**UNIVERSITATEA BABEŞ-BOLYAI CLUJ-NAPOCA**

**FACULTATEA DE MATEMATICĂ ŞI INFORMATICĂ**

**SPECIALIZAREA INFORMATICĂ**

# LUCRARE DE LICENŢĂ

# APLICAȚIE PENTRU DETECȚIA EMOȚIILOR DIN MUZICĂ ȘI GENERARE DE RECOMANDĂRI

**Conducător științific:**

**Lect. Dr. Adriana Mihaela COROIU**

**Absolvent:**

**Andrei PETRESCU**

**Cluj-Napoca**

**2021**

# Abstract

In this paper we present a way for emotion detection from music and voice using supervised learning methods. As a representation of the audio signal, we will consider its mel-spectrogram from which the convolutional neural network will determine the valence and energy values. These predictions are then used in the development of a mobile application, called "Spotimy", whose aim is to generate recommendations based on the user's mood. This can be suggested by the app given their voice or by manually setting parameters.

The data for training the models was collected using the API freely provided by the company "Spotify" and the "Ryerson Audio-Visual Database of Emotional Speech and Song"[21] dataset for emotion detection using voice. For this process, not only the diversity of musical genres was considered, but also the coverage of the value space with data that can reflect a wide range of emotions. Also, the data collected for the preparation of this paper reveals the cultural and linguistic diversity specific to each historical and cultural era.

Following the training phase, good results were obtained for predicting emotions from both music and voice using the same neural network architecture. Specifically, according to the experiments, the accuracy for the training data is about 89% for songs and 90% for voice. For the test data, the accuracy is 86% and 89% respectively. Having obtained these results, the models can be used for correlating predictions towards the desired recommendation mechanism.

Petrescu Andrei, Cluj-Napoca, 2021

# Abstract

În această lucrare vom prezenta o modalitate pentru detecția emoțiilor din muzică și din voce, utilizând metode de învățare supervizate. Ca reprezentare a semnalului audio se va considera spectrograma mel a acesteia din care rețeaua neuronală convoluțională va determina valorile valenței și a energiei. Aceste predicții sunt ulterior utilizate în dezvoltarea unei aplicații mobile, intitulate ”Spotimy”, al cărei scop este de a genera recomandări pe baza stării de spirit a utilizatorului. Aceasta poate fi sugerată de aplicație având în vedere vocea acestuia sau prin setare manuală a parametrilor.

Datele pentru antrenarea modelelor au fost colectate utilizând API-ul pus la dispoziție gratuit de compania ”Spotify” și setul de date ”Ryerson Audio-Visual Database of Emotional Speech and Song”[21] pentru detecția emoțiilor cu ajutorul vocii. Pentru acest proces s-au avut în vedere nu doar diversitatea genurilor muzicale, ci și acoperirea spațiului de valori cu date ce pot reflecta o gamă largă de emoții. De asemenea, datele colectate pentru întocmirea prezentei lucrări relevă diversitatea culturală și lingvistică specifică fiecărei epoci istorice și culturale.

În urma etapei de antrenament s-au obținut rezultate bune pentru predicția emoțiilor atât din muzică, cât și din voce, utilizând aceeași arhitectura a rețelei neuronale. Mai precis, conform experimentelor, acuratețea pentru datele de antrenament este de aproximativ 89% în cazul melodiilor și 90% pentru voce. Pentru datele de testare, acuratețea este de 86% respectiv 89%. Obținând aceste rezultate, modelele pot fi utilizate pentru corelând predicțiile spre obținerea mecanismului de recomandare dorit.

Petrescu Andrei, Cluj-Napoca, 2021

# Conținut

# Listă de figuri

# Introducere

Muzica a fost mereu un aspect important al vieții culturale, dar a devenit mult mai proeminentă în viața cotidiană. Începând cu avansul tehnologic, această formă de artă a devenit accesibilă omului de rând, contribuind la educația acestuia dar mai ales la bunăstarea lui, fiind și o formă de divertisment. De la plăcile de vinil, la casete și până la CD-uri, modul în care consumăm muzica a suferit transformări. Astăzi, muzica trăiește și se dezvoltă în mediul digital sub forma platformelor de streaming. Dintre acestea se evidențiază Spotify, Apple Music și YouTube Music, transformând distribuția de muzică într-o afacere valoroasă, atât pentru aceștia cat și pentru artist.

Însă în toata această schemă unde se află consumatorul? Cum i te adresezi acestuia având în vedere cantitatea enormă și diversă de muzică existentă pe aceste platforme? Soluțiile oferite de aceste platforme constau în sisteme de recomandări, bazate pe muzica ascultată de acesta și preferințele personale, deoarece se adresează unei audiențe ce poate avea gusturi diverse, unele neîncadrate în norme sau poate considerate excentrice. Aceste platforme au oferit până acum posibilitatea de a asculta liste de redare create pe baza preferințelor muzicale, însă, din propria experiență, nu conțin doar melodii pe placul nostru. O astfel de abordare constă în lista de redare „Daily mix” oferită de Spotify, unde acesta include melodii dintr-un anumit gen muzical pe care utilizatorul îl ascultă frecvent. Cu toate acestea, nu de multe ori, utilizatorii revin la muzica deja cunoscută lor.

Până acum am exemplificat recomandările bazate pe genuri, dar dacă muzica recomandată nu se potrivește situației în care se află? A stării lui de spirit? O altă problemă constă în faptul că recomandările în acest caz sunt subiective. Oamenii trăiesc sentimente diferite și sunt influențați diferit de muzica pe care o ascultă. De exemplu, un ascultător de muzica heavy metal poate să resimtă tristețea transmisă de melodie, dar un alt ascultător nu va avea aceeași capacitate. Așadar, lista de redare trebuie sa conțină muzică similară sau apreciată de acesta.

Lucrarea își propune a oferi o soluție la aceasta problemă prin aplicația „Spotimy”. Aceasta este capabilă sa realizeze recomandări bazate pe starea de spirit a utilizatorului, având ca punct de plecare muzica apreciată de acesta. Înclinăm să ascultăm, cu precădere, muzica adăugată recent în librărie. După cum am menționat, gusturile sunt subiective. Acestea sunt influențate de o

multitudine de factori externi, cum ar fi vremea, evenimente recente, etc. Așadar, recomandările trebuie sa conțină muzica recentă pentru un impact mai bun asupra utilizatorului.

Folosind inteligența artificială, putem astăzi detecta din fragmente audio aspecte ce pot transmite sau influența comportamentul ascultătorului. Dintre acestea, două măsuri sunt mai importante: energia melodiei și valența (exprimă cât de „pozitiv" este conținutul)[16]. Măsurile acestea vor juca un rol important în predicția datelor de model[9].

Lucrarea este structurată pe multe capitole. Acestea vor explica mai întâi teoria necesară din spatele mecanismului de predicție și formularea problemei. Apoi discuția se va îndrepta către aplicație, unde se vor prezenta mai întâi tehnologiile folosite și arhitectura aplicației împreună cu designul acesteia. La final se vor prezenta datele utilizate, modul de colectare și preprocesarea acestora.

Primul capitol intitulat "Invățare supervizată" tratează aspectele teoretice din spatele învățării asistate. Acest capitol dezbate cele două probleme de învățare des întâlnite în alte implementări ce au ca subiect detecția emoțiilor din muzică sau voce.

Al doilea capitol extinde informația teoretică spre tărâmul procesării semnalului audio. Se vor descrie aspectele teoretice ce compun un semnal audio, împreună cu mărimile fizice utilizate. De asemenea se va introduce noțiunea de scară melodică (scara mel) și reprezentarea grafică a acesteia.

Următoarele două capitole Tehnologii utilizate și Descrierea aplicației, vor detalia partea practică a acestei lucrări. Pe parcurs s-au utilizat o multitudine de tehnologii în dezvoltarea aplicației, acestea având un capitol separat pentru descrierea amplă a celor mai importante din acestea, dar și a librăriilor ce nu au jucat un rol major dar important. În final se începe descrierea implementării acestei aplicații, accentul fiind pus pe fiecare funcționalitate în parte, împreună cu modul de utilizare a fiecăreia.

Ultimul capitol este destinat experimentelor din parcursul dezvoltării aplicației. Se va explica modul de colectare și procesare a datelor, împreună cu augmentările necesare pentru fișierele audio ce conțin vocea utilizatorului. Capitolul se încheie cu rezultatele în urma procesului de învățare.

# 1. Învățare supervizată

## 1.1 Context

În continuare, se vor descrie aspectele teoretice utilizate în crearea și dezvoltarea aplicației. În prima parte, se discuta elementele din inteligența artificială ce joaca un rol în detectarea emoțiilor din muzica. Acesta este folosit în rezolvarea problemei de regresie pentru predicția unor valori ce pot descrie starea de spirit oferite de un fișier audio, cu precădere o melodie.

Pentru a oferi o mai mare generalitate, modelul trebuie sa poată furniza predicții satisfăcătoare, indiferent de genul muzical. Pentru acest lucru vom utiliza elemente de inteligență artificială pentru ca un calculator să se poată adapta cerințelor diverse.

Algoritmii inteligenți pornesc de la un set de date intrare și prin producerea unui model acestea sunt capabili să rezolve probleme făcând predicții cu rezultat în același domeniu. Aceștia se împart în doua mari categorii: învățare supervizata și învățare nesupervizata. În cazul învățării supervizate se pornește cu un set de date de forma ($x_i$, $y_i$), unde i=1,...,n. Pentru fiecare răspuns $y_i$, există un set de date $x_i$. Acest tip de învățare antrenează modelul să identifice cu precizie cât mai bună răspunsurile pornind de la setul de date de intrare. Astfel se obține un model cu o acuratețe suficient de mare pentru a genera predicții cât mai precise. Pe de alta parte, învățarea nesupervizată nu mai conține un răspuns pentru fiecare set de date de intrare, scopul fiind gruparea acestora pe baza similitudinilor dintre date. În continuare concentrarea va fi pusa pe învățarea supervizata.

### 1.1.1 Problema de regresie

Regresia este o metodă statistică utilizată în multe domenii unde este nevoie de determinarea unei relații dependente între variabile. De obicei, este nevoie de determinarea unei valori, cunoscând una sau mai multe variabile. Variabilele pot fi cantitative sau calitative. În cazul regresiei sunt utilizate variabile cantitative, acestea luând o valoare numerică (ex: valoarea unui bun, temperatura, înălțimea, lungimea etc.). Din acest motiv, problemele în care avem nevoie de determinarea unei variabile sunt cunoscute ca probleme de regresie.

Ecuația formulei generale de regresie este:

$$y = mx + b,$$

$$x = (x0, x1, .., xn),$$

$$m = (m0, m1, .., mn)$$ [40]

De precizat este faptul că formula mai sus descrisa este utilizata doar pentru cazul unei regresii liniare [23]. Există și alte categorii de regresii cum ar fi regresia non liniara. În cazul regresiei non liniare, se va determina curba care poate ajusta datele. În continuare ne vom concentra pe regresia liniara.

Pentru rezolvarea unei probleme de regresie este nevoie de ajustarea, prin diverse metode, a coeficienților (notați cu „m" și „b"). Necesitatea o constă nevoia de a acomoda un interval cat mai mare de date pentru a putea genera predicții satisfăcătoare și conforme realității. Pe scurt, într-o regresie variabila x va prezice variabila y.

### 1.1.2 Problema de clasificare

Față de problemele de regresie, problemele de clasificare[27] utilizează variabile calitative cum ar fi: numele unei persoane, marca unui produs, specii de animale etc. Scopul acesteia este de a identifica subsetul sau categoria din care face parte un anumit element.

Pentru rezolvarea unor astfel de probleme se pot folosi metode de învățare supervizate cum ar fi: regresia logistică sau cei mai apropiați k vecini, dar și metode de învățare nesupervizate transformând problema de clasificare într-o problema de grupare (clustering).

Utilitatea vine în momentul în care este nevoia de identificarea unui anumit tipar în care s-ar putea încadra datele de intrare. În cazul identificării unei emoții se poate folosi un clasificator în contextul în care etichetarea ar putea fi realizata cat mai obiectiv posibil.

## 1.2. Arhitectura rețelelor neuronale

Rețelele neuronale artificiale[14] au ca echivalent rețelele neuronale biologice, alcătuirea fiind asemănătoare cu acestea. O rețea neuronală biologică este formată din unul sau mai mulți neuroni, iar informația este transmisă prin intermediul sinapselor. Pentru eficientizarea procesului de transmisie a informației unele legături sunt „favorizate” în detrimentul altora, prin împuternicirea sinapsei.

Rețelele neuronale artificiale reprezintă una din cele mai populare și mai puternice metode de învățare supervizată. Acestea sunt utile în rezolvarea problemelor cu rezultat real, discret sau chiar vectorial. Utilitatea se remarcă prin posibilitatea de procesare a unor date mai complexe cum ar fi imagini, sunete, valori înregistrate de senzori etc.

O rețea neuronala artificiala este alcătuită din:

- Noduri – acestea joacă rolul neuronilor. Nodurile conectate la mediul extern pot fi utilizați ca noduri de intrare sau ieșire.
- Muchii – au rolul de a lega neuronii intre ei. Aceștia sunt caracterizații prin „greutate” (reprezentata printr-o valoare numerica) și au scopul de a păstra informațiile. Greutatea este afectată de importanța sau utilitatea legăturii ce o formează.
- Straturi neuronale – unul sau mai multe noduri dispuse pe același nivel în graful rezultat. Din stratul de intrare fac parte doar nodurile cu rol de intrare, respectiv în stratul de ieșire doar neuronii ce vor furniza rezultatele. Straturile intermediare celor doua se numesc straturi ascunse

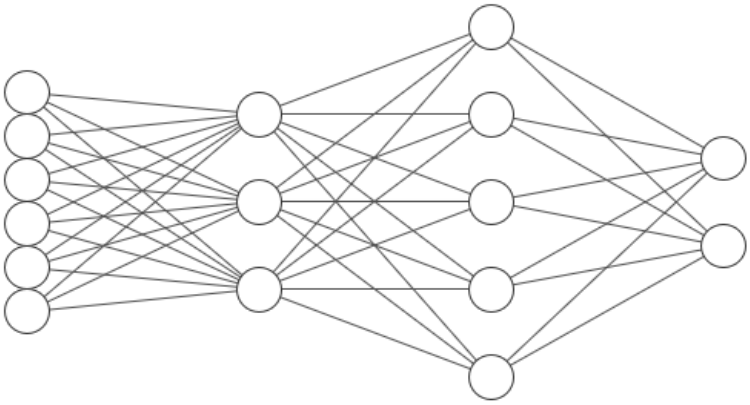

*Fig 1. 1 Rețea neuronala cu 1 strat de intrare, 2 straturi ascunse și un strat de ieșire*

La baza rețelelor neuronale stă perceptronul [32]. Acesta este cel mai primitiv sistem de rețea neuronală folosit îndeosebi pentru reprezentarea unor funcții booleene cum ar fi AND, OR, NAND sau XOR. Perceptronul primește ca intrare un vector x cu valorile x0, x1,...,xn și furnizează răspunsul cu valoarea 1 sau -1, în funcție de pragul stabilit. Acesta calculează o combinație liniară a datelor de intrare pentru a furniza răspunsul.

Funcția de ieșire este:

$o(x_1, \dots, x_n) = \begin{cases} 1, w_1 * x_1 + w_2 * x_2 + \cdots + w_n * x_n > 0 \\ -1, altfel \end{cases}$ , unde $w_1,...,w_n$ reprezintă greutățile și determină gradul de contributivitate a fiecărei valori $x_n$ la rezultatul perceptronului [32].

Astfel, procesul de învățare al perceptronului reprezintă ajustarea greutăților pentru a acomoda un set cat mai mare de date din același domeniu. Se poate începe cu greutăți aleatoare, iar la fiecare pas să se ajusteze cu o valoare reprezentata de eroarea de predicție, numită în acest caz regula perceptronului. Procesul se repetă până când se atinge un număr maxim de iterații, dar de preferat atunci când perceptronul reușește sa obțină rezultate exacte pentru setul de date de antrenament. La fiecare pas, greutățile se schimba în funcție de regula perceptronului, și se reiterează prin întregul set de date de antrenament.

Revizuirea unei greutăți se realizează astfel:

$$w_i \leftarrow w_i + \Delta w_i$$

$$unde\ \Delta w_i \leftarrow \eta(t - o)w_i \text{ [32]}$$

- t – reprezintă valoarea reala a datei de ieșire
- o – reprezintă valoarea obținută
- η – constantă pozitivă ce reprezintă rata de învățare. Controlează rata de schimbare a valorii greutăților (de obicei intre 0 și 1)

Folosind regula perceptronului este posibil sa nu se ajungă niciodată la convergenta. Totuși, chiar și prin aplicarea unei alte reguli, se produc doar funcții liniare. Așadar nu se obțin predicții precise pentru un domeniu de date variat. Astfel pentru a transforma dintr-o problema de regresie liniară într-una neliniară, trebuie folosite altfel de funcții, în acest caz neliniare. Astfel de funcții se mai numesc și funcții de activare[26].

Astfel de funcții des folosite pot fi:

1. funcția tangentă

$$f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \text{ [26]}$$

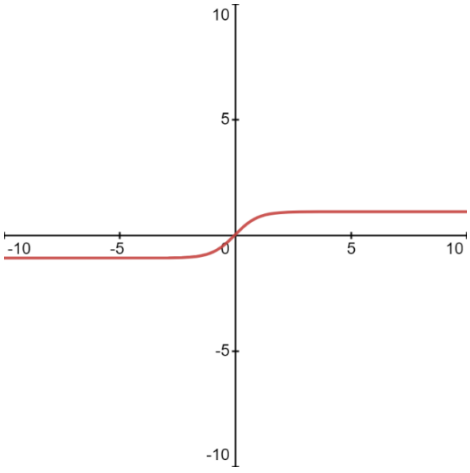


*Fig 1. 2 Funcția tanh pe intervalul [-10,10]*

2. funcția sigmoid

$$f(x) = \frac{1}{1+e^{-x}}$$ [26]

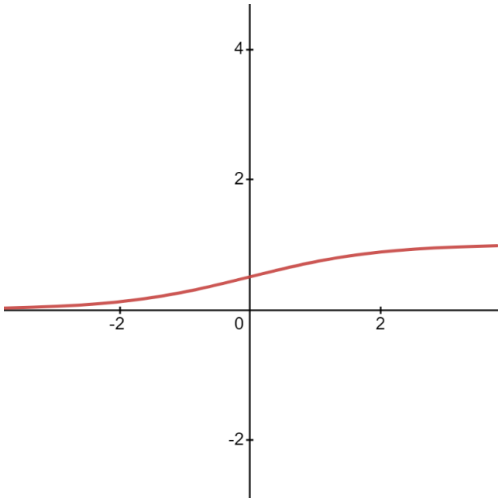


*Fig 1. 3 Funcția sigmoid*

3. Funcțiile ReLU (Rectified Linear Unit) și Leaky ReLU

$$f(x) = \begin{cases} x, & x > 0 \\ 0, & x \leq 0 \end{cases}$$ [26]

funcția ReLU este cea mai eficientă din punct de vedere computațional. Funcțiile sigmoid și tangentă hiperbolică pot cauza ca majoritatea neuronilor sa fie activați în același fel. Așadar ReLU este preferat pentru rețelele neuronale cu multe straturi. Cu toate acestea, problema apare în momentul în care valorile se apropie foarte mult de 0 sau devin negative și algoritmul de backpropagation nu se poate executa. Soluția constă în funcția Leaky ReLU care aduce o ajustare funcției inițiale.

$$f(x) = \begin{cases} x, & x \geq 0 \\ 0.01 * x, & x < 0 \end{cases}$$ [26]

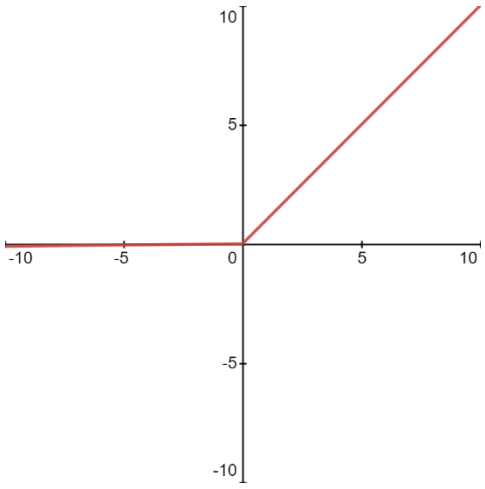


*Fig 1. 4 Funcția Leaky ReLU pe intervalul [-10,10]*

În cazul perceptronului, pentru date care nu sunt liniar separabile se poate folosi regula delta, cunoscută și ca regula Widrow-Hoff [33]. Aceasta are la bază algoritmul gradientului descrescător pentru căutarea greutăților optime. Față de regula perceptronului, nu se dorește identificarea exactă a rezultatului, ci determinarea unui rezultat cât mai apropiat de cel real. Această regulă stă la baza algoritmului de backpropagation.

Pentru calcularea erorii se utilizează formula:

$$E(w) = \frac{1}{2}\sum_{d\epsilon D}(t_d - o_d)^2, unde\ w = (w_0, \dots, w_n)\ [6]$$

- D reprezintă mulțimea datelor de antrenament
- $t_d$ este valorea reală pentru elementul d
- $o_d$ este valoarea calculată pentru elementul d

Față de perceptron, care poate exprima doar decizii liniare, algoritmul backpropagation permite învățarea unui sistem cu mai multe straturi. Astfel, rezultate mai apropiate de cele reale se obțin prin minimizarea erorii. Datorită faptului că acest algoritm poate acomoda un sistem cu mai mulți neuroni, eroarea este redefinită ca suma erorilor tuturor ieșirilor. Astfel formula pentru calcularea erorii va arăta astfel:

$$E(w) = \frac{1}{2}\sum_{d\epsilon D}\sum_{k\epsilon I}(t_{kd} - o_{kd})^2, unde\ w = (w_0, \dots, w_n)\ [6]$$

- I reprezintă mulțimea nodurilor de ieșire
- $t_{kd}$ și $o_{kd}$ reprezintă valorile reale și prezise pentru nodul k de ieșire având data d

Algoritmul backpropagation[6] începe cu o rețea neuronala, având greutățile setate aleator cu valori cât se poate de mici. Datele de intrare sunt de forma (x, y), unde x = (x0,...,xn) și y = (y0,...yn) reprezentând soluția reala pentru setul de date x. Până la finalul iterațiilor sau când se întâlnește situația de oprire se executa astfel:

1. Se parcurge rețeaua obținându-se o predicție y'.
2. Fiecărei unități de ieșire i se calculează eroarea $\delta_k$

$$unde\ \delta_k = y'_k(1 - y'_k)(y_k - y'_k)\ [6]$$

3. Pentru fiecare nod din straturile ascunse se calculează eroarea propagata folosind formula:

$$\delta_h = y'_h(1 - y'_h)\sum_{k\epsilon I} w_{kh}\delta_k\ , unde\ I\ este\ mulțimea\ unităților\ de\ ieșire\ [6]$$

4. Se modifică greutățile după formula:

$$w_{ij} \leftarrow w_{ij} + \Delta w_{ij}$$

$$\text{unde } \Delta w_{ij} = \eta\delta_i x_{ij}\ [6]$$

5. Se revine la pasul 1

În lucrul cu date complexe cum ar fi imaginile, texte, semnale audio etc. este nevoie de un număr mult mai mare de neuroni, așadar și mai multe straturi. Pentru acest lucru putem apela la un domeniu al inteligenței artificial denumit Deep Learning[10] sau Învățare profunda. Profunzimea vine de la numărul mare de neuroni per strat și numărul mare de straturi ce pot exista. O categorie speciala de rețele neuronale constă în rețelele neuronale convoluțiunile[1] ce pot folosi informații dintr-un domeniu mai larg din setul de date, nu doar o singură instanță. Cu alte cuvinte, pentru procesarea imaginilor putem crea și procesa un contur întreg nu doar un singur pixel.

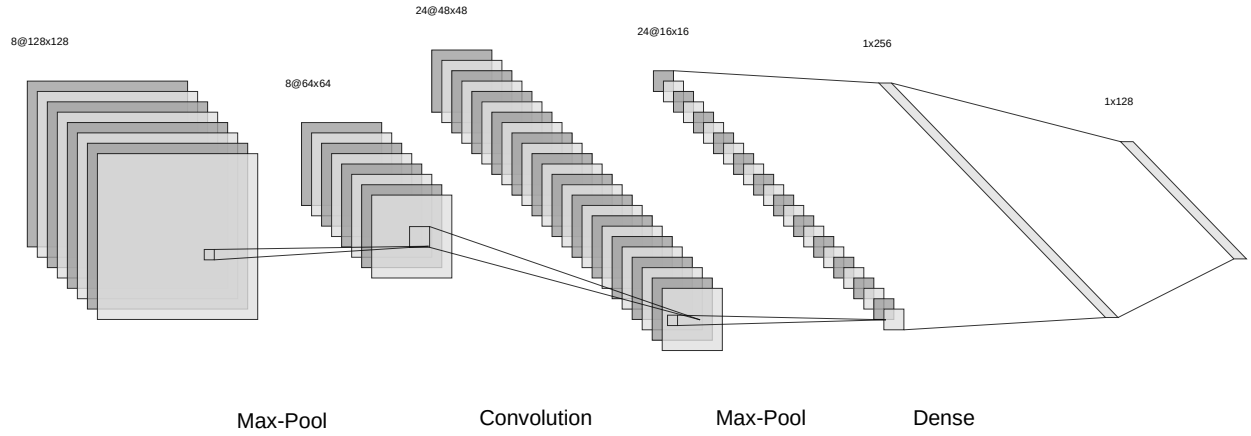


*Fig 1. 5 Exemplu de rețea neuronala convoluțională*

Convoluția este o operație matematică asupra a doua funcții (f și g) ce arată cum una din ele influențează forma celeilalte. Operația se notează cu semnul „*” și are următoarea expresie:

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau \text{ [15]}$$

Un strat convolutional combină un vector de intrare cu un kernel. Acest kernel joaca rolul de filtru pentru vectorul de intrare. În urma operației se obține un alt vector ce va fi procesat în continuare. În cazul procesării imaginilor, filtrele pot fi folosite pentru a scoate în evidenta anumite zone, de exemplu pentru a „trasa” un contur în jurul zonei de interes. Kernelul este „deplasat” peste datele de intrare și se recalculează fiecare element folosind operația de convoluție.

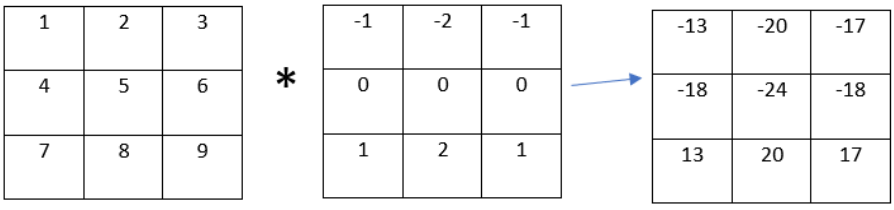


*Fig 1. 6 Exemplu de convoluție*

O altă categorie de filtre este folosită pentru a agrega informația și a rezolva probleme la nivel global. Aceste metode se numesc metode de Pooling. Prin această metodă se învață o reprezentare extinsă a problemei dar se păstrează toate avantajele rețelelor neuronale convoluționale . Cele două operații de Pooling sunt Maximum Pooling[38] și Minimum Pooling. Față de filtrele descrise anterior, acestea nu au un parametru și lucrează doar cu elementele datelor de intrare.

# 2. Reprezentarea semnalului audio

## 2.1 Semnalul audio

Semnalul audio[20] este o reprezentare a sunetului, determinat fie de tensiunea curentului sau de semnalul analog al undei audio. În domeniul digital, acesta este reprezentat sub forma unor serii de numere binare a semnalului digital. Față de semnalul analog, semnalul digital nu este continu. Semnalul audio poate fi caracterizat de mai mulți parametrii dinte care: lățimea de unda, frecvența sau puterea (amplitudinea) măsurată în decibeli (dB).

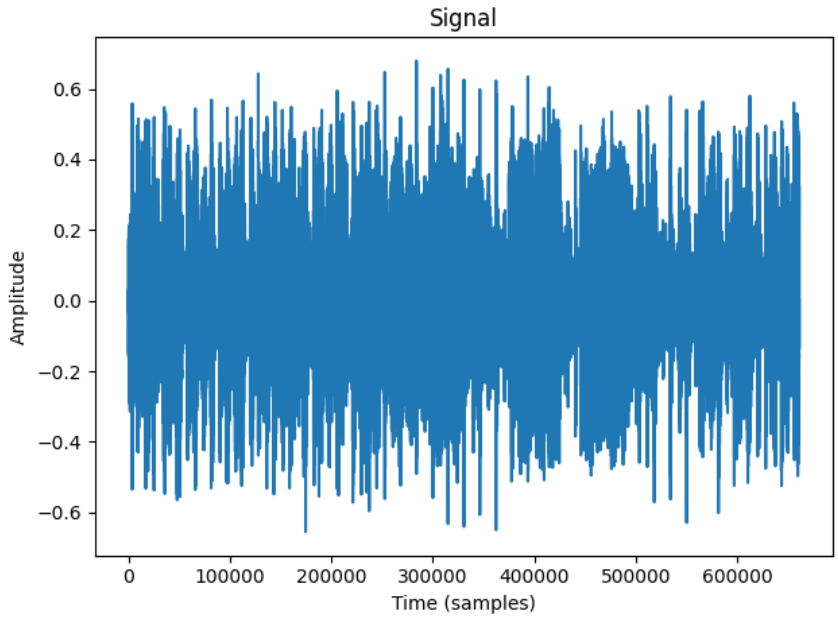


*Fig 2. 1 Exemplu grafic al semnalului audio*

Observând imaginea de mai sus, semnalul audio este descris doar prin amplitudine pe o durată determinată de timp. Cu toate acestea, el este format din mai multe unde de sunet cu anumite frecvențe.

## 2.2 Transformare Fourier

Pentru a descompune semnalul audio în frecvențe individuale[20] și amplitudinea frecvenței se folosește transformarea Fourier[37]. Acesta convertește semnalul măsurat în funcție de timp, într-unul dependent de frecventă. Rezultatul se numește spectru.

Formula discreta a transformării Fourier este:

$X_k = \sum_{n=0}^{N-1} x_n * e^{-i2\pi kn/N}$, unde k=0,1,2,...,n și $x_0$, $x_1$,...,$x_n$ sunt numere complexe[37]

În practică, se folosește algoritmul de transformare Fourier rapidă ( FFT – Fast Fourier Transformation[7]) ce poate furniza rezultate mult mai eficient. Utilitatea vine pentru reprezentarea unui semnal care nu variază în timp.

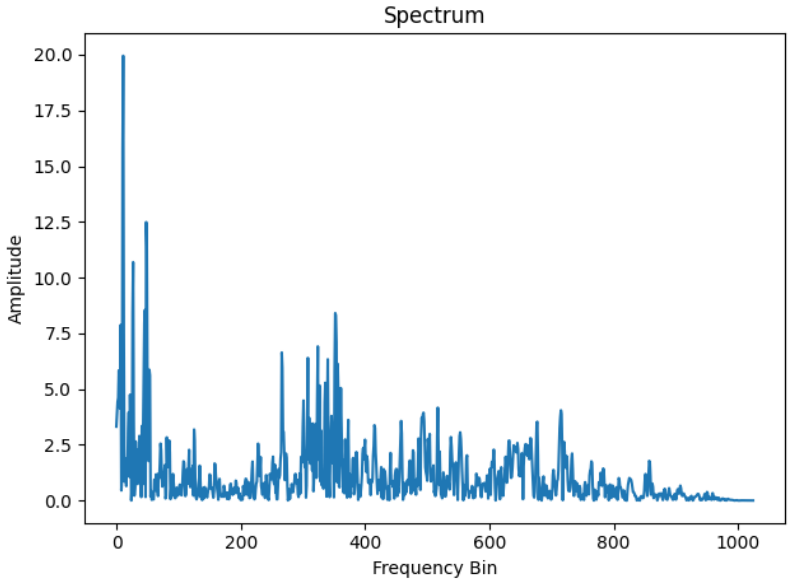


*Fig 2. 2 Semnalul audio după transformare*

## 2.3 Spectrograma mel

Semnalele audio care variază în timp, numite și neperiodice pot fi reprezentate folosind spectrograma semnalului audio. Soluția este sa executam algoritmul FFT pe mai multe porțiuni continue din semnalul audio[20]. Prin adunarea rezultatelor într-un singur grafic obținem astfel o spectrogramă a semnalului audio.

Prin aceasta reprezentăm cu ușurință amplitudinea și varianța acesteia în funcție de timp. De precizat este că și frecventa la rândul ei este transformată în scara logaritmica pentru a putea fi reprezentata. Aceste transformări sunt realizate pentru că oamenii pot percepe un interval foarte mic de frecvențe și amplitudini.

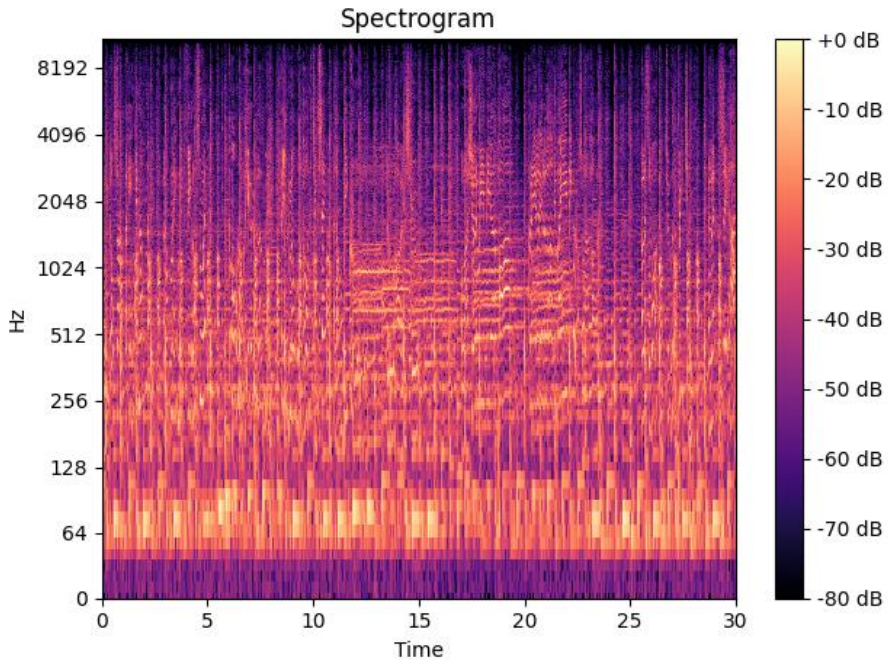


*Fig 2. 3 Spectrograma semnalului audio*

Urechea umană este limitată din punctul de vedere al perceperii frecventelor. Putem detecta cu ușurință frecventele mici intre 500 și 1000 Hz, făcând distincția dintre sunetele din acest interval. Pe de altă parte, sunetele de frecvență înaltă sunt mult mai greu de distins de urechea umana. Acest

lucru se datorează faptului că percepția frecvenței de către aceasta nu este una liniara, ci una logaritmică[39].

Din acest motiv s-a introdus scara mel (provenit de la cuvântul „melodie”) pentru conversie. Scara mel este o scara perceptuala a înălțimii sunetului pentru care, din punctul de vedere al ascultătorului, se află la aceeași distanță una față de cealaltă. Ca punct de referință, se consideră 1000 Hz ca fiind egali cu 1000 meli.

Există mai multe formule de conversie însă una din cele mai populare este:

$$m = 2595 * \log\left(1 + \frac{f}{500}\right) \text{ [18]}$$

În formula de mai sus m este valoarea în meli a frecvenței f. Invers se poate transforma după formula:

$$f = 700(10^{\frac{m}{2595}} - 1) \text{ [18]}$$

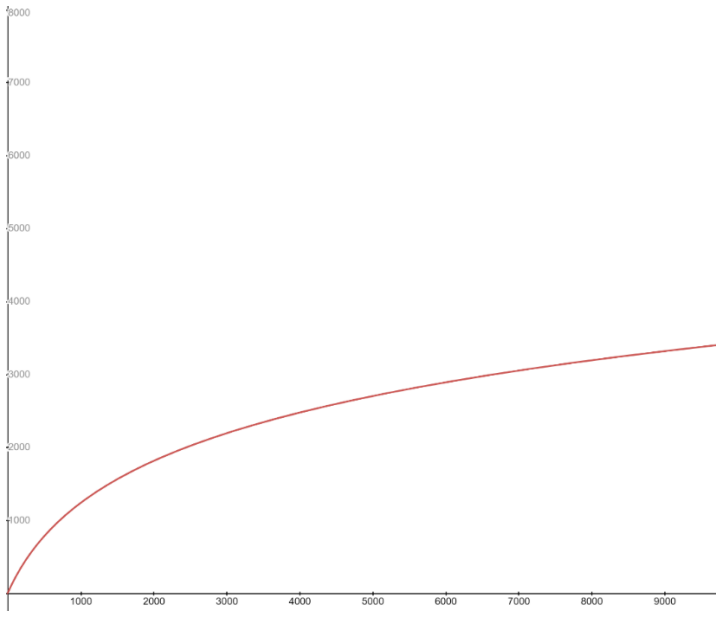


*Fig 2. 4 Graficul scării melodice în raport cu scara hertz*

După realizarea conversiei pentru valorile frecvențelor din scara hertz în scara mel, se poate genera spectrograma mel[20] cu noile valori. Spectrograma mel poate fi utilizată în foarte multe probleme de învățare supervizata cum ar fi: clasificarea după genurile muzicale, clasificarea instrumentelor muzica sau în cazul de fata detecția stării de spirit.

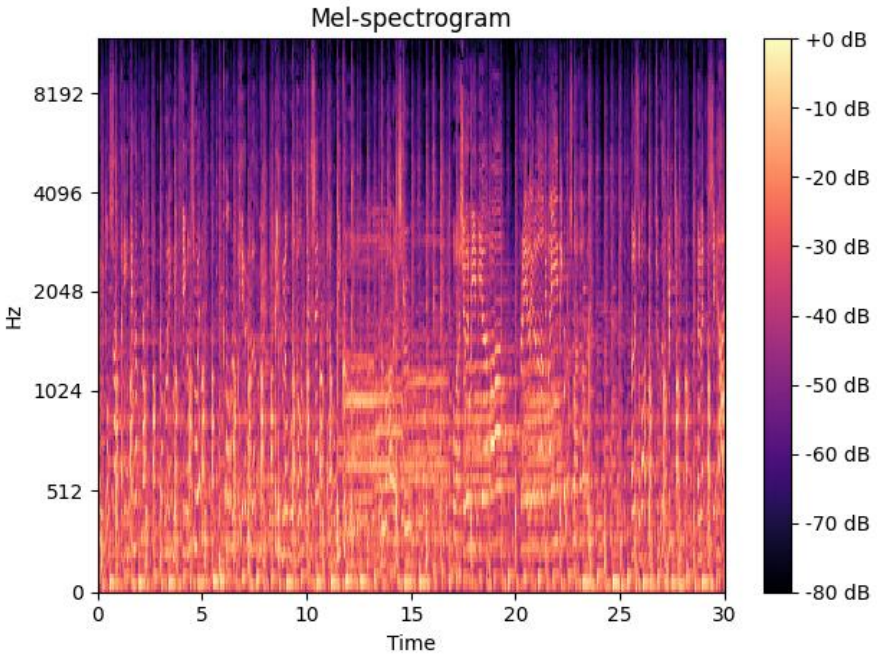


*Fig 2. 5 Spectrograma mel corespunzătoare spectrogramei din Fig 2.4*

# 3. Tehnologii utilizate

În următorul capitol voi descrie tehnologiile utilizate în realizarea aplicației precum și motivele alegerii acestora. Deoarece aplicația are componente frontend precum și backend, mă voi concentra pe descrierea separată a acestora. De asemenea, Spotify API [29] este utilizat atât pentru partea de backend cat și pentru frontend pentru diverse operațiuni specifice ce urmează a fi explicate în următoarele subcapitole.

## 3.1 Tehnologii frontend

Fiind o aplicație mobilă, există o multitudine de tehnologii utilizate în întreaga industrie, cum ar fi Kotlin pentru dezvoltarea aplicațiilor Android Native sau Swift pentru deviceuri iOS. Cu toate acestea abordarea aleasa este una „cross-platform” (funcționarea pe mai multe platforme fără a fi limitați de un anumit sistem de operare), pentru a concentra resursele spre funcționalitate, nu spre platformă. Din acest motiv, pentru dezvoltarea aplicației client am decis sa utilizez framework-ul Ionic.

Ionic[5] este un framework utilizat în crearea de interfețe grafice pentru aplicații mobile prin utilizarea unei tehnologii web în implementarea acesteia. Utilizatorul poate alege dintr-o gamă variată de tehnologii web cum ar fi: Angular, React [24] sau Vue.js. Acesta este lansat în anul 2013 în versiunea alfa, având ca unică tehnologie Angular pentru implementarea funcționalităților, ulterior în 2014 începând cu Ionic 4 putând fi utilizate și cele enumerate mai sus. În cazul aplicației curente s-a utilizat React 17 pentru dezvoltare, pentru funcționalitatea numită „hook” [35] ce permite scrierea de cod mult mai eficient și rapid fără a fi nevoie de clase suplimentare pentru actualizarea anumitor variabile. De asemenea, componentele pot fi modificate fără a fi nevoie reîncărcarea întregii pagini.

Limbajul utilizat de Ionic este TypeScript[2], care este construit ca un supraset sintactic pentru limbajul JavaScript. Astfel, programele scrise în limbajul JavaScript sunt compilabile și în TypeScript. Acesta este des folosit în aplicațiile web datorită abilității de a îmbunătăți calitatea

codului dar și pentru a fi mai ușor de înțeles. În ciuda utilității în dezvoltarea aplicațiilor web, frameworkul Ionic utilizează acest limbaj și pentru dezvoltarea aplicațiilor mobile.

Exceptând pluginurile și bibliotecile implementate în Ionic pentru crearea interfeței grafice, folosind Cordova și mai recent Capacitor[4], Ionic oferă utilizatorului capacitatea de a accesa funcționalități gestionate de sistemul de operare cum ar fi: camera, microfonul, gestionarea fișierelor etc. Pentru deployment, frameworkurile Cordova sau Capacitor (folosit în ultima versiune de Ionic) transforma codul HTML și JavaScript intr-unul executabil pe deviceul mobil.

În implementarea aplicației, mai multe biblioteci și pluginuri sunt introduse în crearea funcționalităților. Acestea sunt:

- File Transfer – utilizat pentru transferul fișierelor audio către serverul REST
- Media – utilizat pentru înregistrarea și salvarea fișierelor audio
- AndroidPermissions – utilizat pentru acordarea de permisiuni aplicație
- Axios – utilizat pentru accesarea endpointurilor implementate de serverul REST și apelarea acestora
- Storage – utilizat în salvarea și accesarea datelor de la nivelul aplicației
- SpotifyAuth – utilizat pentru acordarea de permisiuni necesare aplicației de către utilizator și autentificarea pe serverul Spotify. Autentificarea este realizata prin protocolul OAuth 2.0
- SpotifyWebApi – oferă o serie de funcționalități pentru accesarea endpointurilor oferite de Spotify API

## 3.2 Tehnologii backend

Serverul destinat procesării fișierelor audio și realizarea recomandărilor returnate ulterior aplicației client este implementat în limbajul Python [11]. Începând cu anul 1991, au fost publicate mai multe versiuni, însă pentru aceasta aplicație am utilizat versiunea 3.7, fiind compatibilă cu toate bibliotecile necesare dezvoltării. Endpointurile puse la dispoziție de server sunt implementate folosind Flask [12], o bibliotecă și tehnologie Python utilizată în dezvoltarea de servere REST.

Informația deja procesată este salvată într-o baza de date SQLite, implementată în limbajul C pentru accesarea fiabila și rapida a datelor. Utilizând un instrument ORM, în acest caz SQLAlchemy[31] specific Python, poate fi accesată de server prin codul sursă.

Informația este descărcată și salvata în format „mp3”, având la dispoziție biblioteca Wget. Pentru a genera spectrograma mel se utilizează librăria Python Librosa[3] care poate încărca fișierul audio (în formatul mai sus specificat) descărcat anterior și obținerea de informații în crearea și salvarea imaginii ce conține spectrograma.

Rețeaua neuronală artificială și modelul rezultat sunt create folosind platforma Keras[13] și biblioteca Python aferenta. Keras oferă multitudine de funcții și metode pentru crearea de modele, antrenarea și predicția rezultatelor. De asemenea pune la dispoziție unelte pentru evaluarea statistică a modelului.

Colectarea datelor a fost realizată accesând endpointurile furnizate de API-ul Spotify. Acest API oferă informații despre majoritatea melodiilor deținute de acest serviciu. Pe lângă acestea, oferă și informații despre utilizatorul curent sau despre listele de redare ale acestuia. Pentru accesarea datelor este utilizata biblioteca Python „requests”.

Pe lângă acest server, exista un server destinat conectării la Spotify care a fost implementat în limbajul JavaScript. Serverul REST este implementat utilizând biblioteca „express”, iar datele sunt criptate folosind biblioteca CryptoJS [8].

# 4. Descrierea aplicației

În următorul capitol, voi descrie structura și funcționarea aplicației. Aplicația este gândită spre a fi folosită pe orice device mobil. Accentul este pus pe simplitatea utilizării cât și pe ușurința navigării. Aceasta trebuie în primul rând să furnizeze serviciile cât mai facil și să vină în ajutorul utilizatorului în orice context se afla acesta.

Utilizatorul se conectează folosind contul de Spotify la aplicație o singură dată, deoarece datele de conectare sunt apoi reținute și reutilizate de serviciul de conectare către serverul central Spotify. După conectare i se permite detecția stării de spirit folosindu-și vocea sau setarea manuala a indicatorilor. În final, după ce acesta introduce datele specifice prin intermediul indicatorilor, confirmă prin crearea automată a unei liste de redare specifica stării de spirit detectate utilizând ultimele melodii apreciate din aplicația Spotify.

## 4.1 Cazuri de utilizare

După cum se pot desprinde și din scurta descriere de mai sus, aplicația conține mai multe cazuri de utilizare. Utilizatorului, i se permit mai multe funcționalități, unele fiind obligatorii (utilizatorul trebuie sa fie logat utilizând un cont de Spotify). De asemenea, o condiție necesară este ca acesta sa dețină un cont activ de Spotify Premium pentru a putea beneficia de funcțiile utilizate de aplicație (ex: sa creeze liste de redare, sa salveze melodii etc).

În dezvoltarea aplicației se disting următoarele cazuri de utilizare:

- Login – utilizând un cont Spotify Premium. Utilizatorul introduce credențialele de autentificare pe portalul redirecționat
- Detecția stării de spirit din voce (detect mood) – aplicația înregistrează un fragment din vocea utilizatorului și setează valorile pentru valență și energie
- Setarea stării de spirit (Set mood) – utilizatorul are opțiunea de a ajusta valorile setate anterior sau de a le seta el însuși sărind peste funcționalitatea de detecție automata

- Crearea listei de redare – după setarea parametrilor, la comanda utilizatorului aplicația creează o lista de redare conform acestora
- Redare lista (Play created playlist) – după ce playlistul este creat, la comanda utilizatorului acesta poate reda conținutul pe ultimul device activ

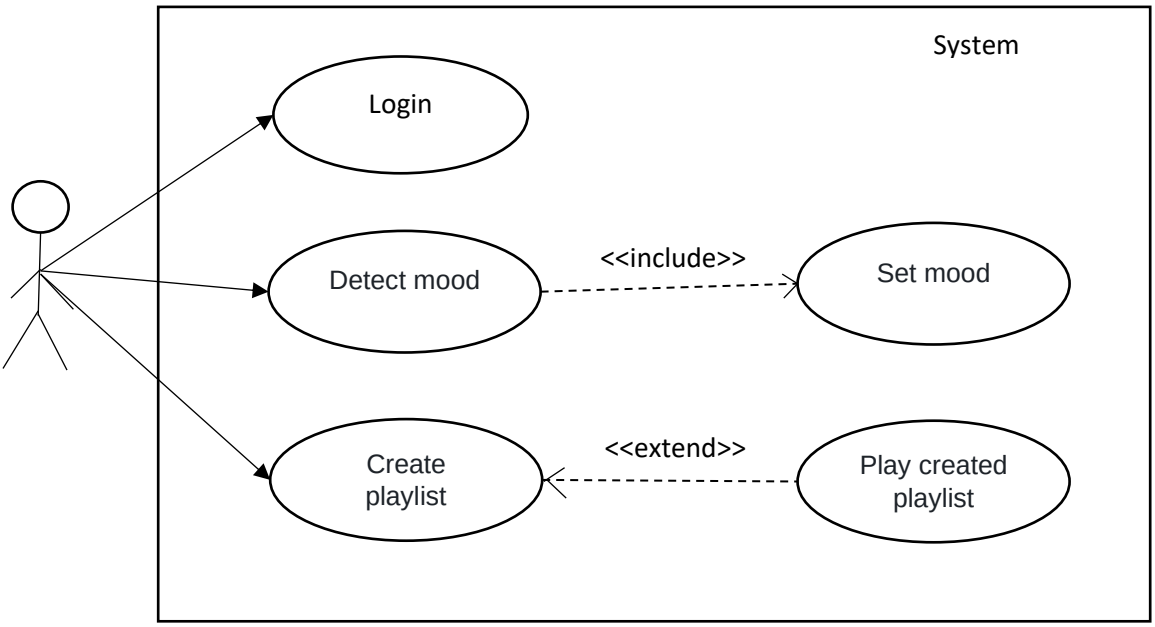


*Fig 4. 1 Diagrama cazurilor de utilizare*

## 4.2 Designul și modurile de utilizare ale aplicației

Am menționat și anterior ca un obiectiv principal este ca experiența utilizatorului în folosirea aplicației sa fie cat mai simpla și clara. Aplicația poate fi folosită în orice context social, deoarece secvența de instrucțiuni este una minimală. Aceste aspecte sunt reflectate și de designul aplicației, care conține un număr minim de comenzi de tip input (butoane și slidere). După prima pornire în care utilizatorului i se cere conectarea folosindu-și credențialele, el poate accesa funcționalitățile fără acest pas intermediar.

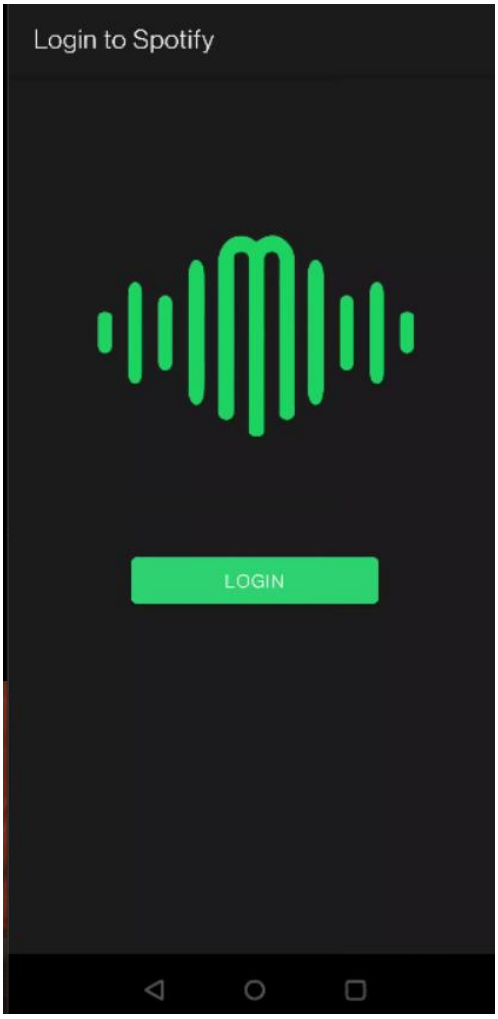


*Fig 4. 2 Pagina de pornire*

Pagina de pornire conține un singur buton pentru pornirea procesului de autentificare. Dacă autentificarea nu a fost executată anterior aplicația nu mai solicita datele necesare și va merge la pasul următor. În caz contrar, aplicația este suspendată și utilizatorul este redirecționat către pagina de autentificare a Spotify.

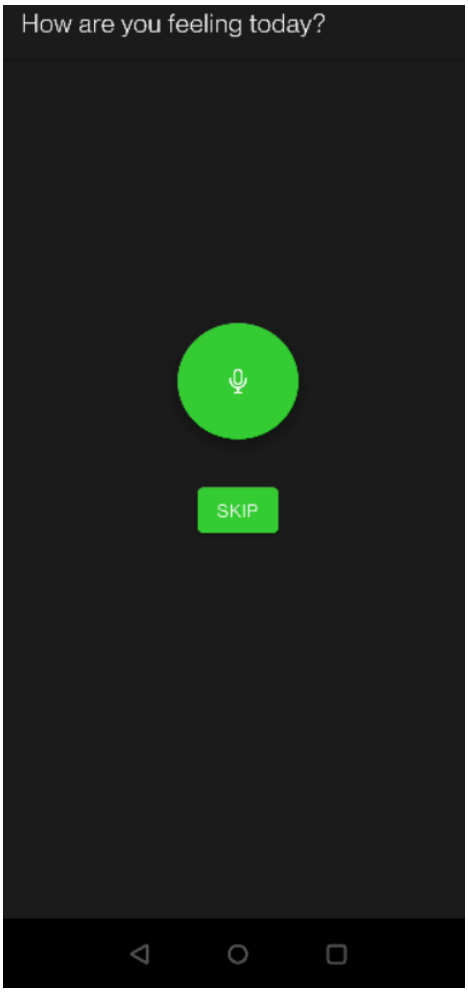


*Fig 4. 3 Pagina pentru detecția emoției utilizatorului*

În aceasta pagina, utilizatorul poate activa microfonul apăsând pe butonul central. La o a doua apăsare, fișierul este trimis către server pentru procesare și după primirea răspunsului, utilizatorul este redirecționat către următoarea pagina. Dacă dorește, poate sa sară peste acest pas apăsând butonul „skip”.

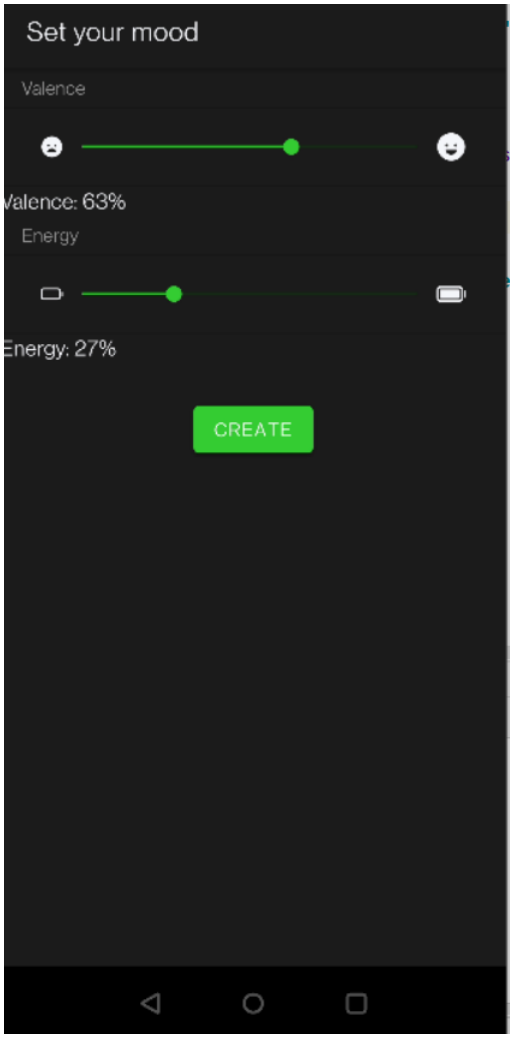


*Fig 4. 4 Pagina de setare a parametrilor emoției*

Dacă la pasul anterior acesta a utilizat funcția de detecție din voce a stării de spirit, cele două slidere vor fi setate cu valorile returnate de server, altfel ele vor fi setate la valoarea 0. În oricare dintre cazuri, utilizatorului i se permite ajustarea valorilor. După aceasta operațiune, prin apăsarea butonului „Create”, playlistul este creat și utilizatorul este redirecționat către pagina următoare.

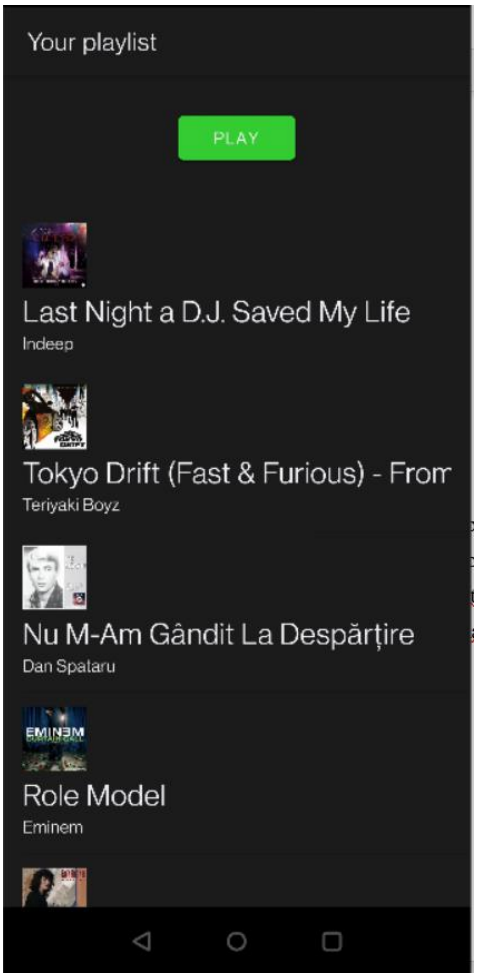


*Fig 4. 5 Pagina conținând lista de redare rezultata*

Această pagină conține lista cu melodiile recomandate și adăugate într-o lista de redare ce poate fi regăsita și pe contul de Spotify al utilizatorului. Acesta poate vizualiza titlul, autorul și imaginea de coperta a albumului. De asemenea, prin apăsarea butonului „Play”, redarea va începe pe ultimul device activ. După începerea redării, textul butonului se va modifica în cuvântul „Pause”, iar prin apăsarea lui redarea va fi oprită temporar.

## 4.3 Arhitectura aplicației și detalii de implementare

Aplicația are două componente: componenta client și componenta server. Aplicația client, după cum am menționat, este o aplicație mobilă pentru platforma Android ce accesează serviciile oferite de un server ce va furniza recomandările, dar și API-ul Spotify pentru obținerea de date legate de utilizator, muzica, liste de redare etc. Serverul pentru recomandări, primește anumite date obținute de aplicația client legate de ultimele melodii apreciate și din acestea va alege și returna melodiile recomandate. De asemenea exista și un server destinat doar mecanismului de autentificare OAuth 2.0 necesar în realizarea acestuia pentru a accesa resursele oferite de API-ul Spotify.

### 4.3.1 Server

Aplicația client are nevoie de cele două servere REST: serverul pentru recomandări și cel destinat autentificării pentru a funcționa. Cele două servere funcționează individual și sunt scrise în limbaje diferite. Serverul pentru recomandări este implementat folosind limbajul Python și frameworkul Flask, iar cel pentru autentificare este scris în limbajul JavaScript.

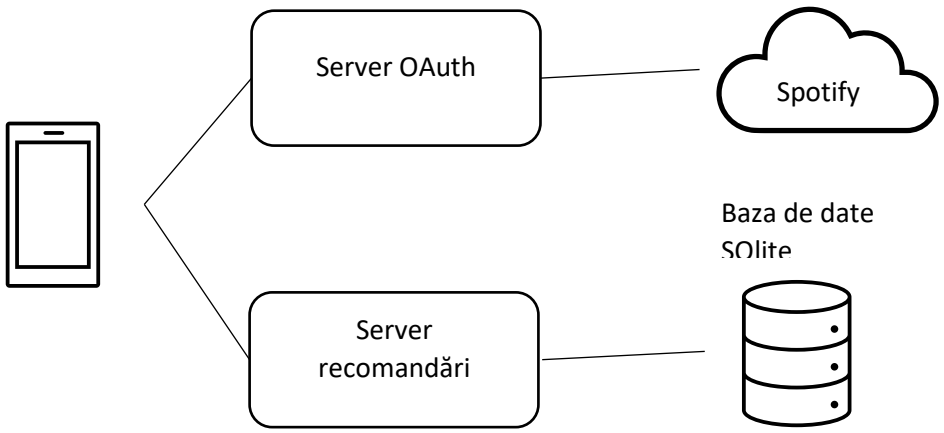


*Fig 4. 6 Diagrama serviciilor accesate de aplicația client*

Serverul destinat autentificării implementează endpointurile necesare autorizării în concordanță cu standardul OAuth 2.0[34]. Acesta funcționează ca un intermediar între aplicația client și serverul ce gestionează conturile Spotify. Pluginul SpotifyAuth[28] pentru frameworkul Cordova ușurează acest proces, având implementate o mare parte din requesturile și funcțiile necesare autentificării păstrând standardul menționat. Cu toate acestea, serverul este necesar pentru gestionarea tokenurilor de acces către aplicație. Cu alte cuvinte, serverul realizează legătura dintre Spotify și aplicație.

Pentru autentificare sunt necesare următoarele resurse stabilite pe platforma destinată developărilor oferita de acesta:

1. Client ID – generat de Spotify la înregistrarea aplicației pentru oferirea de permisiuni
2. Client Secret – de asemenea generat automat și necesar pentru a verifica destinația requestului
3. Scopes – specifică motivul pentru care se obțin autorizările. Acestea vor fi aduse la cunoștință utilizatorului după ce este redirecționat către pagina de login
4. Redirect – reprezintă linkul de redirecționare înapoi către aplicație

Aplicația client va furniza o parte din aceste resurse serverului care prin intermediul endpointurilor va apela serviciul de gestiune al conturilor. Endpointurile implementate și accesate de acesta sunt:

- /exchange – obține tokenul de acces și tokenul de refresh în cazul în care utilizatorul se conectează și oferă acces la datele necesare aplicației (specificate prin resursa scopes)
- /refresh – reînnoiește tokenul de acces și tokenul de refresh fără a mai fi necesară o reconectare a utilizatorului

În implementarea celor doua endpointuri am utilizat o funcție pentru requestul către Spotify și este apelată corespunzător endpointului. Diferența constă în tipul de acces necesar acestora, specificat prin parametrul „grant_type”. Acesta poate fi de tip „authorization_code” pentru obținerea tokenului de acces sau „refresh_token” pentru reînnoirea tokenului de acces.

```
spotifyRequest( params: {
    grant_type:"authorization_code",
    redirect_uri: CLIENT_CALLBACK,
    code: params.code
})
```

*Fig 4. 7 Exemplu de apel pentru funcția spotifyRequest*

Deoarece securitatea este importantă și esențială pentru standardul OAuth, tokenurile de acces și reînnoire sunt criptate folosind librăria CryptoJS. Aceasta necesită un cod secret de criptare (utilizat și în decriptarea aceluiași cuvânt). Pentru ușurință, criptarea și decriptarea sunt implementate în funcțiile aferente, primind ca argument doar cuvântul necesar.

```
function encrypt(str){
    return CryptoJS.AES.encrypt(JSON.stringify( value: {str}), ENCRYPT_SECRET).toString();
}

function decrypt(str){
    var bytes = CryptoJS.AES.decrypt(str, ENCRYPT_SECRET);
    const val = bytes.toString(CryptoJS.enc.Utf8);
    const obj = JSON.parse(val);
    return obj.str;
}
```

*Fig 4. 8 Funcțiile de criptare și decriptare*

Serverul pentru recomandări, pune la dispoziție utilizatorului două endpointuri pentru obținerea listei de redare care va fi creată ulterior și pentru detectarea stării de spirit din vocea utilizatorului. Acesta este creat utilizând frameworkul Flask, pentru ușurința implementării și gestiunea ușoara a requesturilor. Endpointurile sunt accesibile prin metode POST, iar acestea sunt:

- /mood – acest endpoint furnizează o lista cu id-urile pieselor necesare formării unei liste de redare. Ca date de intrare, necesită lista id-urilor melodiilor apreciate furnizate de utilizator împreună cu linkul pentru previzualizare, valența și energia setate de acesta. Acestea trebuie trimise în format JSON. În cazul în care totul decurge bine, status code-ul

este 200. Daca nu se trimit date sau acestea nu conțin toate resursele necesare, codul va fi 401. Altfel, în caz de eroare, se va returna codul 501.

- /profile – acest endpoint este utilizat în determinarea stării de spirit utilizatorului. Acesta necesită o înregistrare audio a vocii acestuia, și se va returna un răspuns în format JSON conținând valorile pentru valenta și energie.

Procesul de recomandare a unei liste de redare începe prin căutarea id-urilor primite de la utilizator în baza de date SQLite. Tabelul „songs” conține id-urile din biblioteca Spotify a melodiilor deja procesate, valenta și energia detectate. Melodiile care nu există în baza de date vor fi procesate și aceasta va fi actualizată cu noile valori. Utilitatea acesteia este pentru a furniza răspunsul cât mai repede și pentru a evita situația unui timeout datorat timpului îndelungat de procesare a melodiilor.

| | id | energy | valence |
|---|---|---|---|
| | Filter | Filter | Filter |
| 1 | 7lLe68z2Hj3ZLEnyEYLaAW | 0.595391929149628 | 0.431295692920685 |
| 2 | 5YWgisUxzDZ28fNFQukf1W | 0.86743700504303 | 0.459577590227127 |
| 3 | 4St5tcazwSomaUQDw6LHJf | 0.631554007530212 | 0.308073073625565 |
| 4 | 7IIsrcbkvVugnDmewbftAv | 0.832797765731812 | 0.533325016498566 |
| 5 | 27kcZEJvhkb1rzZS9gCpdA | 0.608926177024841 | 0.382242560386658 |
| 6 | 26w9NTiE9NGjW1ZvIOd1So | 0.696346282958984 | 0.557423830032349 |
| 7 | 0BCPKOYdS2jbQ8iyB56Zns | 0.774536490440369 | 0.313013881444931 |

*Fig 4. 9 Exemplu de intrări din baza de date*

Procesarea datelor constă în primul rând în determinarea datelor deja procesate și în al doilea rând în actualizarea cu date noi. Melodiile ale căror id-uri nu se regăsesc în baza de date sunt descărcate prin intermediul link-ului de previzualizare, utilizând librăria WGet și salvate în format .mp3. După descărcare, se generează spectrograma mel folosind librăria Librosa, iar imaginea acesteia este salvată în format .png, filtrând culorile în „grayscale” (pe scara alb-negru) pentru a fi mai ușor de procesat de către rețeaua neuronală. Predicția realizată de rețeaua neuronala este adăugată în tabelul din baza de date și utilizată în continuare pentru realizarea recomandărilor. La finalul acestui proces se poate realiza recomandarea pe baza similarității dintre valorile predicțiilor și

starea de spirit a utilizatorului. Aceste valori, după cum am menționat, sunt exprimate prin valenta și energie. Intervalul acestor valori este de la 0 la 1.

Similaritatea dintre aceste valori cuantifică „apropierea" dintre două seturi de valori ce descriu aceleași date. Astfel, reprezentând grafic într-un sistem cartezian aceste date, similaritatea este reprezentata de distanta Euclidiana dintre doua puncte. Cu toate acestea, o formula ce poate determina mai precis similaritatea este abaterea pătratică medie (root mean squared error – RMSE [36]). În cazul problemei noaste, vom măsura similaritatea dintre starea de spirit a utilizatorului și starea transmisa de melodiile pe care acestea le-a apreciat.

$$S = \sqrt{\frac{(x-x0)^2 + (y-y0)^2}{2}}$$ [36]

- S – este valoarea numerica a similarității
- (x, y) și (x0, y0) sunt seturile de date pentru care testam similaritatea

După determinarea similarității dintre aceste date, datele sunt organizate ierarhic descrescător după această valoare. Id-urile primelor 15 melodii din ierarhie sunt trimise în răspunsul către utilizator, pentru ca lista de redare sa conțină în medie cel puțin 50 de minute de muzică.

Pentru predicția valorilor utilizând spectrograma mel a unei melodii sau a vocii utilizatorului, am utilizat o rețea neuronală convoluțională. În implementarea acesteia am utilizat librăria Keras împreună cu Tensorflow[22], accentul fiind pus pe dezvoltarea rețelei nu pe cod. Modelul este antrenat folosind datele puse la dispoziție de Spotify.

Odată stabilită arhitectura modelului, se poate începe antrenarea acestuia. Datele sunt împărțite în doua categorii: date de antrenament și date de testare. Proporția aleasa este de 80/20, unde 80% reprezintă datele de antrenament, iar 20% sunt datele de test. Alegerea acestei proporții pentru antrenarea finală a modelului este detaliată în capitolul legat de experimente.

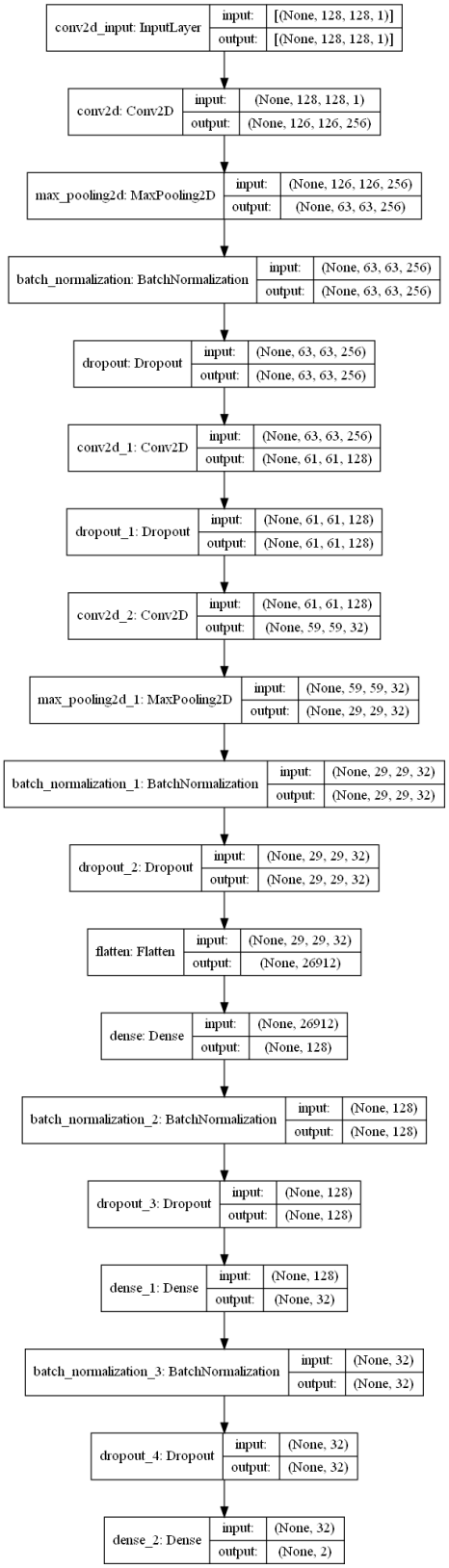


*Fig 4. 10 Modelul rețelei neuronala convoluțiunile utilizat*

### 4.3.2 Client

Aplicația client este implementată cu ajutorul frameworkului Ionic, împreună cu React. Limbajul utilizat este TypeScript. Deoarece frameworkul React este în general utilizat pentru aplicații web, aspectul este definit printr-un fișier HTML, iar stilurile utilizate sunt scrise în fișiere .css. Ecranele interfeței grafice sunt definite prin rute (routes), iar fiecăreia ii revine o componentă pentru controlul fiecărui element grafic sau proces executat în cadrul aplicației. Rutele sunt definite în fișierul App.tsx.

```
<Route path="/home" component={Home} exact={true}/>
<Route path="/recorder" component={Recorder} exact={true}/>
<SpotifyProvider>
  <Route path="/mood" component={Mood} exact={true}/>
  <Route path="/playlist" component={Playlist} exact={true}/>
  <Route path="/playlist/:id" component={Playlist} exact={true}/>
</SpotifyProvider>
```

*Fig 4. 11 Rutele utilizate de aplicație*

Ecranul de start al aplicației este /home. Pe acest ecran, componenta Home desfășoară procesul de autentificare după apăsarea butonului Login. Pentru realizarea autentificării, s-a utilizat un plugin Cordova, numit SpotifyAuth care împreună cu serverul descris mai sus obține tokenul de acces necesar. Acest plugin executa către server și redirecționează utilizatorul către portalul Spotify. Pentru acesta este nevoie de furnizarea datelor prezente în Fig. 4.12.

```
export const spotifyConfig = {
    clientId: clientId,
    redirectUrl: redirect,
    scopes: scopes,
    tokenExchangeUrl: `${authServerUrl}/exchange`,
    tokenRefreshUrl: `${authServerUrl}/refresh`,
};
```

*Fig 4. 12 Datele pentru configurarea pluginului*

După obținerea tokenului de acces, requesturile către serverul Spotify sunt realizate prin intermediul unui alt plugin, SpotifyWebApi[30]. Acest plugin necesită doar un token și după setarea acestuia, pune la dispoziție o serie de funcții care ușurează apelul endpointurilor. Toate endpointurile puse la dispoziție de Spotify pot fi apelate prin intermediul acestuia. Melodiile apreciate sunt preluate înainte de încărcarea paginii /mood, iar lista de redare obținută în urma recomandării este afișată pe pagina /playlist. Funcțiile pluginului utilizate sunt:

- getMySavedTracks – returnează ultimele 50 de melodii apreciate de utilizatorul conectat
- getMe – returnează id-ul utilizatorului
- createPlaylist – creează o lista de redare
- addTracksToPlaylist – adaugă melodii într-o lista de redare, furnizând id-ul listei și lista de id-uri ale melodiilor
- play – redă o lista de melodii pe ultimul device activ
- pause – oprește redarea

După obținerea melodiilor, acestea sunt transferate către serverul de recomandări prin biblioteca Axios. Datele sunt cuprinse printr-o interfața și trimise către server. Răspunsul este de asemenea încapsulat într-o alta interfață pentru ușurința utilizării datelor pe parcursul implementării aplicației.

```
export interface MoodData{
    energy: Number,
    valence: Number,
    songs: SongData[],
}
```

*Fig 4. 13 Exemplu de interfața utilizata pentru încapsularea datelor*

Înregistrarea audio a vocii este realizată prin intermediul pluginului Media, ce are nevoie în primul rând de permisiunea utilizatorului pentru a utiliza microfonul. După salvarea fișierului audio în memoria interna acesta este trimis utilizând pluginul File Transfer.

```
const fileTransfer = FileTransfer.create();
let options: FileUploadOptions = {
    fileKey: 'file',
    fileName: 'record.mp3',
}

return fileTransfer.upload(filePath, url: `${serverURL}/profile`, options);
```

*Fig 4. 14 Transferul fișierului audio către serverul de recomandări*

# 5. Experimente

## 5.1 Colectarea și preprocesarea datelor

În acest capitol, se va descrie modul de colectare a datelor necesare antrenării modelului, inclusive etapa de preprocesare a acestora. Pentru aceasta am utilizat endpointurile "/audio-features" și "/audio-features/:id", puse la dispoziție de Spotify API. Acestea conțin o serie de date legate de aspectele tehnice și statistice din spatele unei sau a mai multor melodii.

Răspunsul unui astfel de apel este în format JSON și formează un așa numit "Audio-features object". Acesta conține mai multe date referitoare la melodie dintre care: valența, energia (necesare aplicației noastre), tempo, cheie, gradul de dansabilitate, etc. În cadrul procesului de colectare am cuprins și aspectele mai sus enumerate pentru eventuale îmbunătățiri considerate în viitor.

Un alt set de date necesar antrenării a constat în generarea spectrogramei mel corespunzătoare melodiei ale căror date tocmai au fost adăugate prin apelul anterior. Răspunsul este oferit tot de acest API, care oferă prin endpointul "tracks/:id" informațiile necesare aplicației primare dintre care: id, numele artistului, poza de coperta, etc, dar cel mai important oferă un link de previzualizate conținând un fișier .mp3 (cu o durata de 30 de secunde) ce poate fi descărcat. Pentru automatizarea procesului am folosit din nou biblioteca WGet din Python. După descărcare am generat spectrograma mel corespunzătoare având asociata id-ul melodiei ale căror date au fost stocate anterior.

Aspectul vital legat de procesul de colectare a constat în alegerea melodiilor ce vor fi adăugate în baza de date pentru antrenamentul modelului. Obiectivul este ca modelul sa poată determina cu o acuratețe suficientă emoțiile dintr-un spectru cât mai larg de melodii, indiferent de gen, limbă, sau raportul dintre coloana instrumentala și cea vocala.

Melodiile alese fac parte din liste de redare diverse, conținând melodii populare zilelor noastre dar și muzica din perioade diferite considerate de succes sau definitorii unor anumiți ani. Limba este de asemenea importanta pentru ca poate exprima emoții diferite de la un ascultător la altul, așa ca am introdus muzica aparținând limbilor latine, slavice, asiatice, africane și chiar și nordice. În acest mod am încercat sa acopăr facil un spectru mai larg de date. Genurile cel mai des întâlnite în setul

de date colectat sunt: pop, rock, hip-hop, clasica, electronica și folk cu subgenurile aferente acestora cum ar fi: house, heavy metal, rap etc.

În urma colectării acestora am proiectat aceste date într-un sistem de axe ortogonal în funcție de cele două caracteristici definitorii: valența și energia, pentru a verifica daca acestea acoperă un spectru suficient de larg și de complex. Totalul melodiilor colectate este de 1203 piese.

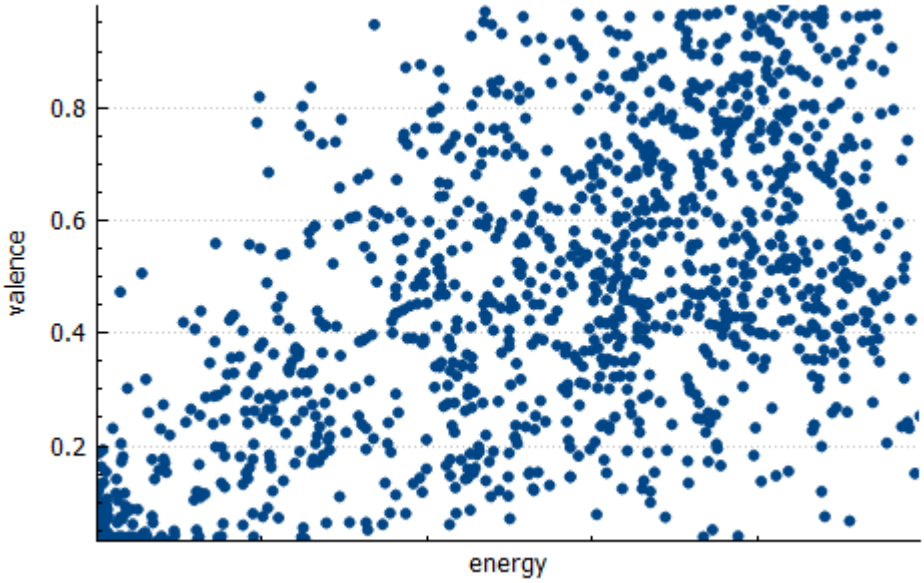


*Fig 4. 15 Dispunerea datelor într-un sistem de axe ortogonal în funcție de valenta și energie*

Pentru antrenarea modelului ce va detecta emoțiile din voce am utilizat setul de date Ryerson Audio-Visual Database of Emotional Speech (RAVDESS)[21]. Acesta conține o serie de fișiere audio cu interpretarea vocală a unor emoții. Actorii de genuri diverse adresează mai multe replici, fiecare având cate o conotație emoționala diferita.

Asupra acestor date, a fost nevoie de o etapa de preprocesarea. Problema constă în faptul ca aceste fișiere corespund unei probleme de clasificare, însă predicția va fi reprezentata de un set de două valori numerice menționate anterior. Parcurgând datele am acordat fiecărei emoții o valoare generata uniform dintr-un interval specific acesteia. Componenta de detecție a emoției din voce este una secundara ce vine în ajutorul utilizatorului în a-și determina singur starea de spirit. Cu alte

cuvinte, nu este necesar ca rezultatul predicției sa corespunda în totalitate cu realitatea trăită de utilizator.

```
moods = {
        '01':{'name':'neutral', 'maxval':0.6, 'minval':0.4, 'maxenerg': 0.6, 'minenerg': 0.4},
        '02':{'name':'calm', 'maxval':0.8, 'minval':0.5, 'maxenerg':0.4, 'minenerg':0.1},
        '03':{'name':'happy', 'maxval':1.0, 'minval':0.7, 'maxenerg':1.0, 'minenerg':0.7},
        '04':{'name':'sad', 'maxval':0.3, 'minval':0.05, 'maxenerg':0.2, 'minenerg':0.05},
        '05':{'name':'angry', 'maxval':0.3, 'minval':0.05, 'maxenerg':1.0, 'minenerg':0.6},
        }
```

*Fig 4. 16 Intervalele pentru valenta și energie în funcție de emoție (not. val = valenta, energ = energie)*

Clipurile audio sunt înregistrate într-un studio cu aparatură performantă. Un device mobil obișnuit nu poate ajunge la aceeași calitate audio. Din acest motiv o serie de filtre audio pentru augmentarea datelor au fost aplicate: zgomot de fundal, mișcarea sursei audio, variații sonore etc.

În cazul filtrului pentru zgomot, augmentarea s-a realizat luând în considerare rata zgomotului în raport cu semnalul audio[17], folosind formula:

$$SNR = 10 * log \frac{{RMS_{signal}}^2}{{RMS_{noise}}^2} \text{ [19]}$$

RMS (Root Mean Squared) reprezintă media pătratică medie, în cazul semnalului sau a zgomotului. SNR (Signal-Noise Ratio) este rata zgomotului în raport cu semnalul audio. Pentru generarea zgomotului se prelucrează formula anterioară obținându-se:

$$RMS_{noise} = \sqrt{\frac{{RMS_{signal}}^2}{10^{\frac{SNR}{10}}}} \text{ [19]}$$

Alte formule utile:

$$RMS_{noise} = \sqrt{\frac{\sum n_i^2}{n}} \text{ [19]}$$

$$STD_{noise} = \sqrt{\frac{\sum (n_i - \mu_{noise})^2}{n}}, unde\ STD\ este\ abaterea\ standard \text{ [19]}$$

De asemenea, media pătratica medie a zgomotului este egala cu deviația standard. În același timp putem genera zgomot utilizând distribuția Gaussiană ce are media egala cu 0. În final, aceasta se adaugă sursei audio de intrare obținându-se un semnal augmentat.

## 5.2 Rezultate experimentale

Datele colectate pentru întocmirea prezentei lucrări relevă diversitatea culturală și lingvistică specifică fiecărei epoci istorice și culturale. De asemenea, acestea aparțin unor multitudini de genuri ce exprimă trăiri și emoții complexe sugerând interpretări originale, dificil de catalogat într-o manieră obiectivă utilizând etichete.

Valorile metricilor prin care emoțiile sunt exprimate (valența și energia) aparțin intervalului [0, 1], formând un spațiu bidimensional. Datele colectate acoperă suficient de mult acest spațiu, obiectivul fiind determinarea unei game largi de emoții. Astfel s-au colectat 1206 melodii, pentru care s-au generat spectrogramele mel aferente folosite ulterior ca intrare pentru model.

După cum am menționat, proporția datelor este de 80/20, împărțite în date pentru procesul de învățare și date pentru testarea modelului obținut[25]. Acest raport a fost ales, în primul rând, considerând numărul relativ mic de date. În al doilea rând, în urma primului experiment pentru datele de testare valoarea acurateței nu a fost suficientă pentru a utiliza modelul pentru a prezice valori ce ar putea fi folosite de sistemul de recomandare. Prima proporție considerată a fost de 60/40, iar în urma testelor s-au obținut următoarele rezultate:

- Loss: 0.0042
- Acuratețe (Accuracy): 0.706

În urma fazei de antrenament pentru proporția finală 80/20 s-au obținut următoarele rezultate pentru datele de testare:

- Loss: 0.0056
- Acuratețe (Accuracy): 0.867

Analizând rezultatele de mai sus, putem observa un loss scăzut față de testul având proporția finală, dar acuratețea este mult mai mică, chiar insuficientă pentru problema noastră. Cu toate acestea, se

poate considera că un număr mai mare de date, ar putea genera rezultate mult mai bune în cazul acestei proporții, probabil comparative cu cele obținute pentru proporția 80/20.

Procesul de învățare a avut loc în mai multe cicluri. Un ciclu reprezintă totalitatea numărului de epoci în care se produce „învățarea" având datele de antrenament. În total s-au considerat 4 cicluri de învățare, având în total 2000 de epoci. Ca date de intrare, modelul va primi o imagine alb-negru a spectrogramei mel de dimensiune 128x128 px și va furniza o predicție constând în perechea de valori valenta și energie (acestea aparținând intervalului [0, 1]).

În arhitectura modelului din Fig. 4.10 se observă existența unor straturi de Dropout după fiecare strat convolutiv. Probabilitatea beta aferentă acestui strat afectează modul în care "noi rute" sunt descoperite. Acesta este utilizat pentru a evita posibilitatea de overfitting.

Valorile testate au fost următoarele: 0.35, 0.75, 0.90. Valoarea stabilită este cea de 0.35, deoarece, în contrast cu rezultatele bune obținute pentru datele de antrenament, datele de test nu au generat predicții suficient de bune. Predicțiile pentru beta mai mare ca 0.35 pot fi vizualizate în Fig. 5.1, considerând un sistem ortogonal de axe unde pe axa Ox (orizontala) se reprezintă valorile valenței, iar pe Oy (verticala) vor fi valorile energiei.

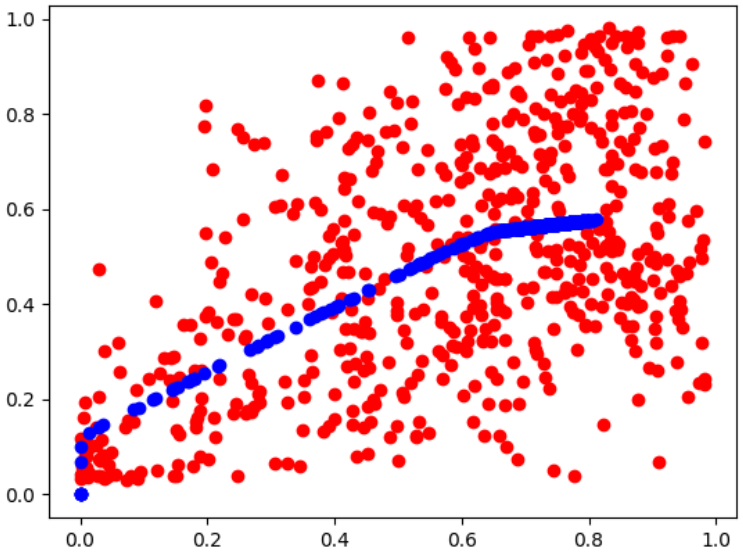


*Fig 5. 1 Dispunerea datelor de test (roşu) și a rezultatelor predicției (albastru) într-un sistem ortogonal de axe pentru beta > 0.35*

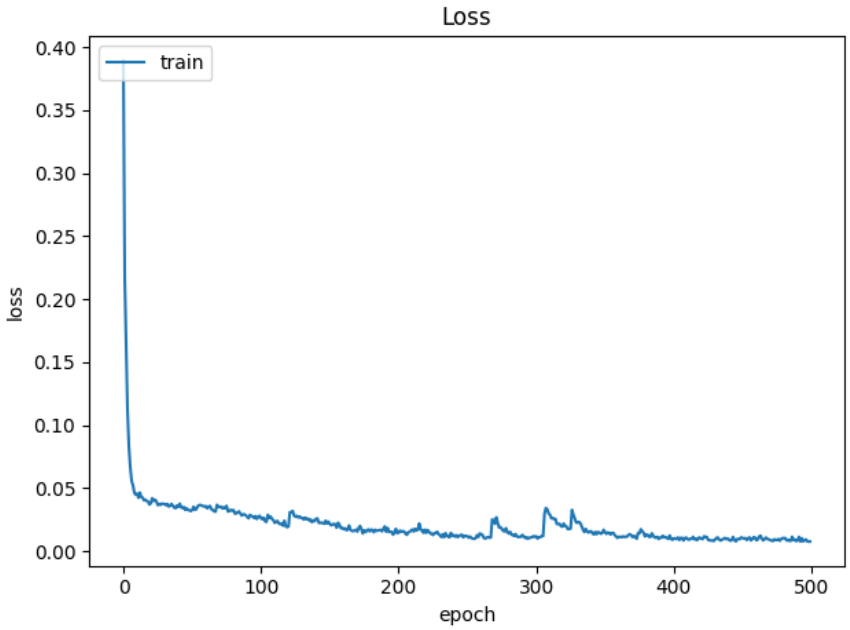


*Fig 5. 2 Graficul evoluției loss-ului după un ciclu de 500 de epoci (beta = 0.35)*

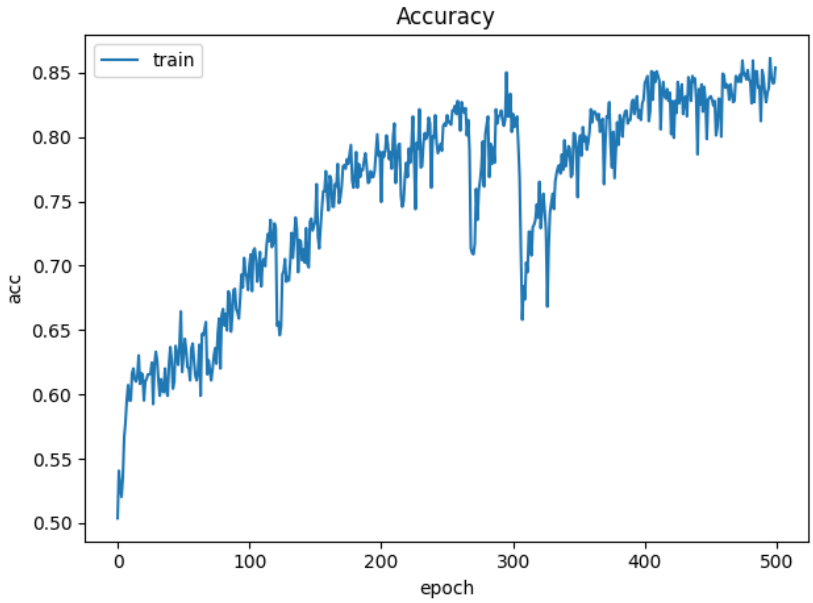


*Fig 5. 3 Graficul evoluției acurateței după un ciclu de 500 de epoci (beta = 0.35)*

De precizat este faptul că acuratețea este măsurată luând în considerare distanta euclidiana dintre valorile reale și cele obținute din urma predicție. Scăderea apărută între primele 300-350 de epoci este datorată stratului de Dropout. Datele sunt reprezentate într-un sistem de axe ortogonale. Pe

axa Ox (orizontala) se reprezintă valorile valentei, iar pe Oy (verticala) vor fi valorile energiei. O astfel de reprezentare după un ciclu de antrenament (adică după ce sunt terminate toate epocile de antrenament setate) și testare este următoarea:

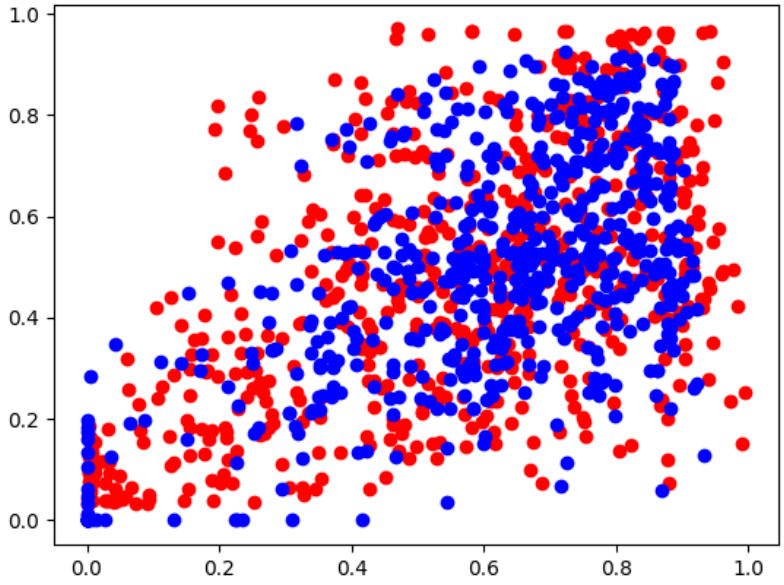


*Fig 5. 4 Dispunerea datelor de test (roșu) și a rezultatelor predicției (albastru) într-un sistem ortogonal de axe și beta = 0.35*

Pentru datele de test al modelului de predicție bazându-se pe vocea umana rezultatele sunt următoarele:

- Loss: 0.0109
- Acuratețe (Accuracy): 0.884

Proporția datelor și în acest caz este de 80/20, din aceleași considerente menționate anterior, iar procesul de învățare a avut loc în mai multe cicluri, asemănător celor de la modelul anterior, arhitectura rețelei fiind complet aceeași. Fiecare ciclu constând în 500 de epoci de învățare. Și în acest caz datele de intrare au constat în imagini alb-negru cu spectrograma mel aferenta semnalului audio, redimensionate la dimensiunea de 128 x 128 px.

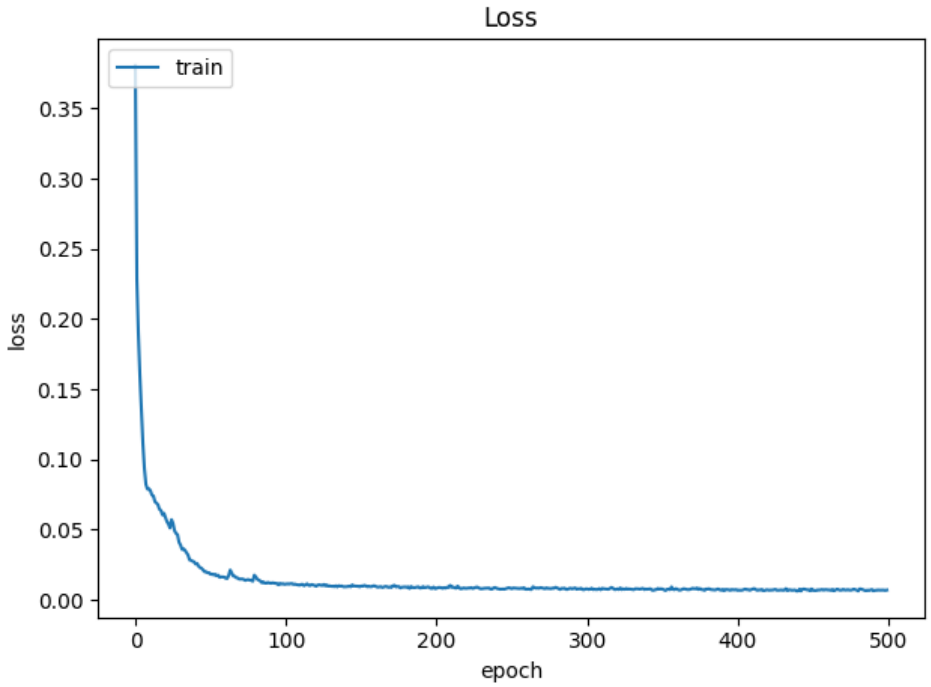


*Fig 5. 5 Graficul evoluției* loss-*ului după un ciclu de 500 de epoci (voce)*

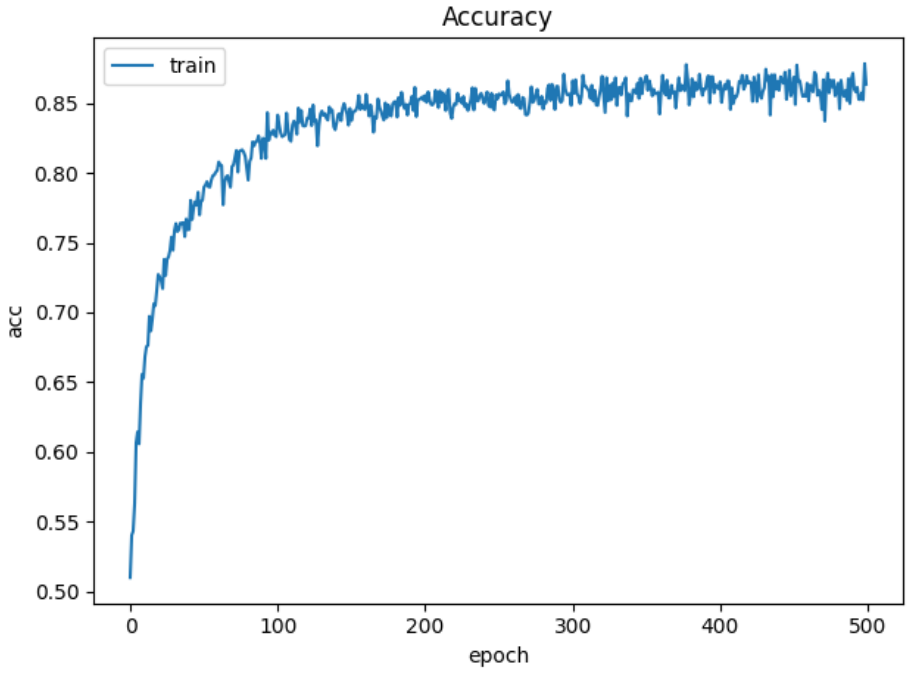


*Fig 5. 6 Graficul evoluției acurateței după un ciclu de 500 de epoci (voce)*

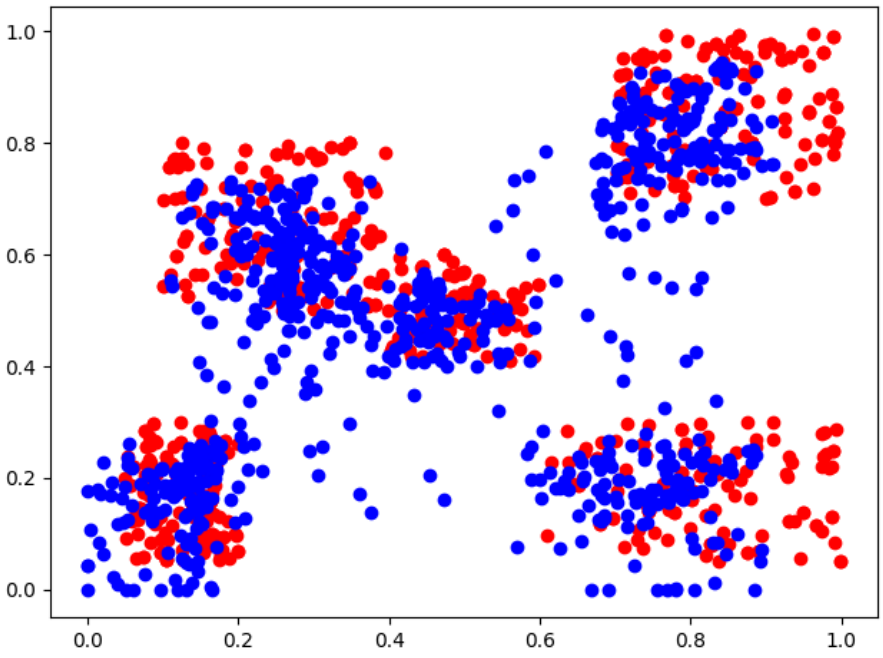


*Fig 5. 7 Dispunerea datelor de test (roșu) și a rezultatelor predicției (albastru) intr-un sistem ortogonal de axe (voce)*

# Concluzii și direcții viitoare de dezvoltare

Muzica ne influențează multe aspecte din viață, dintre care, cel mai important, modul în care resimțim lucrurile. Din acest motiv, un altfel de recomandare se poate face în funcție de starea de spirit trăită la acel moment. Aceasta trebuie sa fie cât mai personală, apelând la istoricul alegerilor muzicale ale utilizatorului. Pentru că volumul de muzică produs este enorm, recomandările trebuie sa reflecte starea curentă a preferințelor.

Aceasta lucrare și-a propus sa prezinte un alt mod prin care putem crea recomandări muzicale, folosind inteligenta artificiala. Acest proces este împachetat într-o aplicație mobila ușor de folosit, ce folosește aplicația Spotify și istoricul aprecierilor utilizatorului pentru a crea astfel de recomandări. Lista de redare obținută fiind adăugată la cele existente de pe platforma pentru a fi audiată și mai târziu, când preferințele curente nu mai sunt în totalitate aceleași cu cele de la momentul creării.

În realizarea modelului, s-a utilizat spectrograma mel a melodiei din care se extrag valoarea valentei și a energiei. Aceeași abordare s-a folosit și pentru vocea utilizatorului, scopul fiind de a ușura procesul în care acesta introduce datele despre starea lui de spirit curenta. Prin acest lucru am evidențiat fiabilitatea modelului pentru orice fel de date audio. Diferența constă doar în datele de antrenament furnizate, existând diferențe majore între spectrograma mel a unei melodii și a unei voci. Modelul rezultat a obținut o acuratețe mare de aproximativ 89% pentru datele de antrenament, procesul de învățare fiind suficient de rapid. În cazul vocii acuratețea este de aproximativ 87%, existând loc de îmbunătățiri prin augmentarea datelor pentru a acomoda o mai mare varietate de situații. În cazul testelor reale, acuratețea s-a apropiat suficient de valorile mai sus menționate.

În final, aplicația client îndeplinește scopul de a obține datele necesare și de a crea lista de redare rezultata. Astfel obiectivul serverului este doar de a crea recomandările, nefiind inclus în procesul de colectare de date în numele utilizatorului.

În urma dezvoltării aplicației am identificat o serie de direcții noi de dezvoltare, incluzând eventuale îmbunătățiri. O posibilă funcționalitate este aceea de a lua în considerare și melodiile propuse de Spotify ca fiind similare cu ce ascultă la momentul curent utilizatorul, în crearea

recomandării, existând riscul ca acesta sa nu includă suficiente melodii în lista de aprecieri. O altă funcționalitate poate consta în alegerea unui număr minim de melodii inclus în recomandări respectiv un număr minim de melodii procesate pentru recomandare.

În ceea ce privește componenta inteligentă a aplicației, arhitectura rețelei utilizate poate fi îmbunătățită, pe de o parte prin simplificarea acesteia și obținerea unor rezultate similare sau chiar mai bune. Pe de altă parte, putem reantrena modelul colectând mai multe date de antrenament, mai diverse din toate punctele de vedere menționate pe parcursul lucrării. Direcția viitoare de dezvoltarea include utilizarea versurilor pentru predicția emoțiilor din muzică, dar și a semnalului audio.

O altă îmbunătățire poate fi adusă componentei inteligente pentru predicția emoției utilizând vocea, prin aplicarea mai multor filtre peste datele existente pentru a simula o mai mare varietate de echipamente de înregistrare mai puțin profesionale, respectiv îmbogățirea setului de date cu înregistrările unor voci în alte limbi.

# Bibliografie


[1] Albawi, S., Mohammed, T. A., & Al-Zawi, S. (2017, August). Understanding of a convolutional neural network. In 2017 International Conference on Engineering and Technology (ICET) (pp. 1-6). Ieee.

[2] Bierman, G., Abadi, M., & Torgersen, M. (2014, July). Understanding typescript. In European Conference on Object-Oriented Programming (pp. 257-281). Springer, Berlin, Heidelberg.

[3] Brian McFee et al. “librosa: Audio and music signal analysis in python”. In: Proceedings of the 14th python in science conference. Vol. 8. 2015.

[4] CapacitorJS, 2021, URL: https://capacitorjs.com/docs, 20.05.2021

[5] Chaudhary, P. (2018). Ionic Framework. International Research Journal of Engineering and Technology, 5(05), 3181-3185.

[6] Cilimkovic, M. (2015). Neural networks and back propagation algorithm. Institute of Technology Blanchardstown, Blanchardstown Road North Dublin, 15, 1-12.

[7] Cooley, J. W., Lewis, P. A., & Welch, P. D. (1969). The fast Fourier transform and its applications. IEEE Transactions on Education, 12(1), 27-34.

[8] CryptoJS, 2021, URL: https://cryptojs.gitbook.io/docs/, 20.05.2021

[9] Delbouys, R., Hennequin, R., Piccoli, F., Royo-Letelier, J., & Moussallam, M. (2018). Music mood detection based on audio and lyrics with deep neural net. arXiv preprint arXiv:1809.07276.

[10] Goodfellow, I., Bengio, Y., Courville, A., & Bengio, Y. (2016). Deep learning (Vol. 1, No. 2). Cambridge: MIT press.

[11] Gowrishankar, S., & Veena, A. (2018). Introduction to Python Programming. CRC Press.

[12] Grinberg, M. (2018). Flask web development: developing web applications with python. " O'Reilly Media, Inc.".

[13] Gulli, A., & Pal, S. (2017). Deep learning with Keras. Packt Publishing Ltd.

[14] Gupta, N. (2013). Artificial neural network. Network and Complex Systems, 3(1), 24-28.

[15] Hirschman, I. I., & Widder, D. V. (2012). The convolution transform. Courier Corporation.

[16] James A Russell. A circumplex model of affect. Journal of personality and social psychology, 39(6):1161, 1980.

[17] Johnson, D. H. (2006). Signal-to-noise ratio. Scholarpedia, 1(12), 2088.

[18] Kamm, T., Hermansky, H., & Andreou, A. G. (1997). Learning the Mel-scale and optimal VTN mapping. In Center for Language and Speech Processing, Workshop (WS 1997). Johns Hopkins University.

[19] Lahiru Nuwan Wijayasingha. Adding noise to audio clips, 2021, URL: https://medium.com/analytics-vidhya/adding-noise-to-audio-clips-5d8cee24ccb8, 20.05.2021

[20] Leland Roberts. Understanding the Mel Spectrogram. 2021. url: https://medium.com/analytics-vidhya/understanding-the-mel-spectrogram-fca2afa2ce53

[21] Livingstone SR, Russo FA (2018) The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS): A dynamic, multimodal set of facial and vocal expressions in North American English. PLoS ONE 13(5): e0196391. https://doi.org/10.1371/journal.pone.0196391.

[22] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Rafal Jozefowicz, Yangqing Jia, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Mike Schuster, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems,2015. Software available from tensorflow.org.

[23] Montgomery, D. C., Peck, E. A., & Vining, G. G. (2021). Introduction to linear regression analysis. John Wiley & Sons.

[24] Phan, H. D. (2020). React framework: concept and implementation.

[25] Reitermanova, Z. (2010). Data splitting. In WDS (Vol. 10, pp. 31-36).

[26] Sharma, S., & Sharma, S. (2017). Activation functions in neural networks. Towards Data Science, 6(12), 310-316.

[27] Song, Y. Y., & Ying, L. U. (2015). Decision tree methods: applications for classification and prediction. Shanghai archives of psychiatry, 27(2), 130.

[28] SpotfyAuth for Ionic, 2021, URL:https://ionicframework.com/docs/native/spotify-auth, 20.05.2021

[29] Spotify Web API, 2021, URL: https://developer.spotify.com/documentation/web-api/, 20.05.2021

[30] SpotifyWebApiJS, 2021, URL: https://www.npmjs.com/package/spotify-web-api-js, 20.05.2021

[31] SQLAlchemy 1.4, 2021, URL: https://docs.sqlalchemy.org/en/14/, 20.05.2021

[32] Stephen, I. (1990). Perceptron-based learning algorithms. IEEE Transactions on neural networks, 50(2), 179.

[33] Stone, G. O. (1986). An analysis of the delta rule and the learning of statistical associations. Parallel distributed processing: Explorations in the microstructure of cognition, 1, 444-459.

[34] The OAuth 2.0 Authorization Framework, 2021, URL: https://datatracker.ietf.org/doc/html/rfc6749, 20.05.2021

[35] Tran, T. B. (2020). Tooling with React.

[36] Wang, W., & Lu, Y. (2018, March). Analysis of the mean absolute error (MAE) and the root mean square error (RMSE) in assessing rounding model. In IOP conference series: materials science and engineering (Vol. 324, No. 1, p. 012049). IOP Publishing.

[37] Winograd, S. (1976). On computing the discrete Fourier transform. Proceedings of the National Academy of Sciences, 73(4), 1005-1006.

[38] Wu, H., & Gu, X. (2015, November). Max-pooling dropout for regularization of convolutional neural networks. In International Conference on Neural Information Processing (pp. 46-54). Springer, Cham.

[39] Wyse, L. (2017). Audio spectrogram representations for processing with convolutional neural networks. arXiv preprint arXiv:1706.09559.

[40] Zou, K. H., Tuncali, K., & Silverman, S. G. (2003). Correlation and simple linear regression. Radiology, 227(3), 617-628.