UNIVERSITATEA TEHNICĂ „Gheorghe Asachi” din IAȘI
FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE
DOMENIUL: Calculatoare și Tehnologia Informației
SPECIALIZAREA: Tehnologia Informației

Analiza stărilor emoționale generate de stimuli audiovizuali

LUCRARE DE DIPLOMĂ

Absolvent
Ceucă Roxana

Coordonator științific
Prof. Dr. Ing. Florina Ungureanu

Iași, 2021

DECLARAȚIE DE ASUMARE A AUTENTICITĂȚII PROIECTULUI DE DIPLOMĂ

Subsemnatul CEUCĂ ROXANA,
legitimat cu CI seria MZ nr. 570039, CNP 2990116226742,
autorul lucrării ANALIZA STĂRILOR EMOȚIONALE GENERATE DE STIMULI AUDIOVIZUALI

elaborată în vederea susținerii examenului de finalizare a studiilor de licență, programul de studii TEHNOLOGIA INFORMAȚIEI organizat de către Facultatea de Automatică și Calculatoare din cadrul Universității Tehnice „Gheorghe Asachi" din Iași, sesiunea IULIE 2021 a anului universitar 2020-2021, luând în considerare conținutul Art. 34 din Codul de etică universitară al Universității Tehnice „Gheorghe Asachi" din Iași (Manualul Procedurilor, UTI.POM.02 - Funcționarea Comisiei de etică universitară), declar pe propria răspundere, că această lucrare este rezultatul propriei activități intelectuale, nu conține porțiuni plagiate, iar sursele bibliografice au fost folosite cu respectarea legislației române (legea 8/1996) și a convențiilor internaționale privind drepturile de autor.

Data
6.07.2021

Semnătura

Cuprins

Analiza stărilor emoționale generate de stimuli audiovizuali

Ceucă Roxana

Rezumat

În cadrul proiectului de diplomă, se urmărește realizarea unei analize comparative între evaluarea stării emoționale (fericire și tristețe) pe baza unor stimuli audiovizuali, în etapa de procesare a semnalelor EEG și în cea de procesare a spectrogramelor semnalelor. Se pornește de la modelul circumplex 2D propus de James Russell, bazat pe 3 dimensiuni independente (valență, excitare, dominanță).

Se utilizează setul de date **DEAP**, conceput pentru analiza stărilor umane afective și care conține date EEG și semnale fiziologice periferice, precum: mișcarea ochilor, temperatura pielii, răspunsul galvanic al pielii, presiunea volumului de sânge și ritmul respirației, stimulii fiind muzicali și vizuali. În faza experimentală, au fost înregistrate semnalele pentru 32 de participanți, de sex masculin și feminin, în timp ce acestora li s-au prezentat o serie de videoclipuri muzicale. Participanții au vizionat în total 40 de videoclipuri, cu durata de un minut fiecare. Aceștia au oferit note privind nivelul de excitare, valență, plăcere, dominanță și familiaritate.

Pentru a evalua în mod corect starea emoțională, semnalele achiziționate trebuie preprocesate și clasificate folosind algoritmi de învățare automată. Etapele procesului de analiză a semnalelor sunt:

- parsarea datelor obținute și aplicarea filtrelor necesare;
- extragerea benzilor de frecvență prin aplicarea unui filtru chebyshev invers de ordin 5;
- extragerea trăsăturilor din benzile de frecvență: entropia șablon, densitatea puterii spectrale, rădăcina medie pătratică și deviația/abaterea standard;
- afișarea grafică a benzilor de frecvență;
- clasificarea datelor folosind algoritmul Support Vector Machine;
- generarea spectrogramelor pentru fiecare stimul și canal;
- extragerea trăsăturilor din spectrograme: Histogram of Oriented Gradients, histograma unei imagini din spațiul de culori HSV, trăsături calculate cu Scale Invariant Feature Transform, valorile texturii prin metoda Haralick;
- clasificarea spectrogramelor semnalelor folosind algoritmul SVM.

În urma etapei de clasificare folosind algoritmul SVM, s-a obținut o acuratețe de 78.75% în cazul clasificării semnalelor EEG și de 81.25% în cazul clasificării spectrogramelor semnalelor. Un impact semnificativ asupra rezultatelor l-au avut trăsăturile extrase, atât pentru semnale, cât și pentru spectrograme. Se poate concluziona că algoritmul Support Vector Machine prezintă un grad de potrivire mai ridicat pentru clasificarea de imagini și mai scăzut pentru cea a semnalelor. De asemenea, trebuie luat în vedere faptul că, semnalele EEG au fost împărțite în subtrial-uri, trăsăturile fiind calculate pentru cel de la jumătatea intervalului, pe când spectrogramele s-au generat pentru întreg semnalul. Efortul de calcul în contextul procesării semnalelor EEG a fost considerabil mai mare față de efortul depus pentru procesarea spectrogramelor, care implică un grad destul de ridicat al utilizării resurselor memoriei.

Introducere

Paradigma de calcul omniprezentă devine o realitate; ajungem la un nivel de automatizare și calcul în care oamenii și dispozitivele interacționează perfect. Cu toate acestea, una dintre principalele provocări este dificultatea pe care o au utilizatorii în interacțiunea cu aceste sisteme din ce în ce mai complexe. Folosind electroencefalograma (EEG) ca senzor de biosemnal, starea afectivă a unui utilizator poate fi modelată și ulterior utilizată pentru a realiza un sistem care să recunoască și să reacționeze la emoțiile utilizatorilor.

Emoția umană este un fenomen extrem de complex care are la bază activitatea cerebrală, însă nu există cunoștințe clare despre mecanismul său de generare. Fiziologii și informaticienii îl studiază de zeci de ani. De exemplu, Ekman a propus șase emoții de bază care sunt universale și se găsesc în fiecare cultură. Posner a fondat un model bidimensional în care emoțiilor li s-au acordat coordonate ce denotă gradul de valență (calitatea pozitivă sau negativă a emoției) și excitare (cât de receptiv sau energic este subiectul).

Au fost propuse diverse teorii și principii de către psihologi în ceea ce privește definirea stărilor emoționale, cum ar fi, a fi cognitiv sau a fi necognitiv. În această dezbatere care continuă să progreseze, există o afirmație care susține că, în loc să se clasifice emoțiile în categorii comune, cum ar fi, fericire, tristețe, furie sau frică, emoțiile pot fi clasificate pe baza modelului valență-excitare.

În raport cu aplicațiile inginerești, definirea și clasificarea emoțiilor sunt importante pentru a decide ce variabile ar trebui luate în considerare și ce măsurători sunt necesare în timpul proiectării unui sistem.

Emoțiile unei persoane pot fi măsurate pornind de la caracteristici externe, cum ar fi expresiile faciale și tonul vocii ce pot fi capturate prin fotografii, înregistrări video și audio. Cu toate acestea, emoția are ca rezultat și alte modificări corporale, cum ar fi variația ritmului cardiac, tensiunea musculară, respirația și conductanța pielii și, desigur, activitatea creierului.

Detecția emoțiilor având la bază semnale EEG are o varietate largă de aplicații practice, evidențiată mai ales în domeniul medicinii, al cercetării științifice și al calculului afectiv care se referă la înglobarea emoțiilor în interacțiunea om-calculator oferind mașinilor un grad de inteligență emoțională. În acest sens, s-a propus ca pentru utilizarea sistemelor de învățare automată să fie incluse medii multimedia, cum ar fi sistemele de recomandare și etichetare, jocurile și filmele care să răspundă la emoțiile utilizatorilor.

Pentru a proiecta un sistem de recunoaștere a emoțiilor folosind semnalele EEG, principalele provocări sunt reprezentate de extragerea eficientă a trăsăturilor și de clasificarea optimă a acestora. Semnalele EEG sunt neliniare, nestaționare, afectate de zgomot și aleatorii. Manipularea și extragerea caracteristicilor semnificative din semnalele EEG joacă un rol crucial în proiectarea unui sistem de recunoaștere a emoțiilor.

Lucrarea de față își propune să realizeze o analiză a stărilor emoționale bazându-se pe comparația dintre recunoașterea de trăsături caracteristice în semnalele de tip EEG și în spectrogramele generate pornind de la acestea. Proiectul este structurat în două etape: o etapă de analiză și clasificare a semnalelor EEG și o etapă de procesare și clasificare a spectrogramelor semnalelor.

Conținutul lucrării scrise este structurat în 4 capitole:

- **Capitolul 1** prezintă aspecte teoretice asupra temei propuse, incluzând referințe către studii similare, introducerea noțiunilor necesare înțelegerii scopului lucrării, definirea trăsăturilor aferente semnalelor și spectrogramelor;
- **Capitolul 2** prezintă setul de date utilizat, structura aplicației, funcționalitatea fiecărui modul în parte și bibliotecile utilizate;
- **Capitolul 3** furnizează informații legate de implementarea modulelor, însoțită de secvențe de cod corespunzătoare;
- **Capitolul 4** prezintă o analiză a rezultatelor experimentale obținute.

Capitolul 1. Aspecte teoretice privind tema propusă

1.1. Studii similare

Studiul emoțiilor în interacțiunea om-calculator a suferit o creștere în ultimii ani datorită nevoii tot mai mari de aplicații informatice capabile să detecteze starea emoțională a utilizatorilor. O mare parte din cercetările acestui domeniu au explorat detectarea emoțiilor din informații faciale și vocale, sistemele informatice actuale fiind capabile de o clasificare a acestora cu o precizie considerabilă. Cu toate acestea, emoțiile nu se manifestă întotdeauna prin intermediul expresiilor faciale și al informațiilor vocale. Psihologii realizează o distincție între excitarea fiziologică, expresia comportamentală și experiența conștientă a emoțiilor. Informațiile faciale și vocale sunt legate numai de expresia comportamentală care poate fi controlată și modificată în mod conștient și a cărei interpretare este adesea subiectivă. Astfel, au fost propuse alte abordări pentru detectarea emoției, care se concentrează pe diferite informații fiziologice, cum ar fi ritmul cardiac, conductanța pielii și dilatarea pupilei. Un domeniu relativ nou al cercetării într-o interacțiune creier-calculator (Brain-Computer Interface) încearcă să detecteze emoția folosind electroencefalograma (semnalele EEG). Au existat mai multe abordări pentru detectarea emoțiilor bazate pe EEG, dar există încă puțin consens cu privire la concluziile definite.

Rafael Ramirez și Zacharis Vamvakousis au prezentat în lucrarea lor [1], o metodă nouă de detecție a emoțiilor din semnale EEG, măsurate cu ajutorul dispozitivului Emotiv EPOC, alcătuit din 14 electrozi de colectare a datelor și 2 electrozi de referință. Locațiile disponibile au fost: AF3, F7, F3, FC5, T7, P7, O1, O2, P8, T8, FC6, F4, F8 și AF4. Cei 6 subiecți (3 bărbați și 3 femei) au fost instruiți să privească o cruce pe ecranul unui computer și să rămână așezați în timpul experimentului, ce a constat în audiția a 12 stimuli sonori preluați din librăria *International Affective Digitized Sounds* (IADS), situați la extreme pe planul excitare-valență. În partea experimentală, au făcut înregistrări EEG pe canalele AF3, AF4, F3, F4, aflate în cortexul prefrontal. Semnalele EEG au fost împărțite în două benzi particulare, alfa și beta, de interes în cercetarea emoțiilor atât pentru valență, cât și pentru excitare. Calculând raportul dintre undele beta și alfa, s-a prezis starea de excitare a fiecărui utilizator. Pentru a determina nivelul de valență, adică starea sufletească negativă sau pozitivă, s-au comparat nivelele de activare ale celor două emisfere corticale. Folosind metoda de învățare *Support Vector Machine* (SVM), s-a obținut o performanță de clasificare de 77,82% pentru excitare scăzută/crescută și 80,11% pentru valență negativă/pozitivă.

Veerabathini Srinivas a folosit setul de date DEAP pentru recunoașterea emoțiilor în lucrarea sa [2]. Împărțirea pe patru benzi de frecvență, alfa, beta, theta și gamma, s-a realizat prin metoda descompunerii Wavelet. Trăsăturile au fost clasificate folosind *Multi-Layer Perceptron* (MLP) și *Radial Basis Function* (RBF). Cele mai bune rezultate în domeniul frecvență au fost obținute cu 54.54% din RBF și 63.63% din MLP.

Un alt studiu bazat pe setul de date DEAP este prezentat în lucrarea „Evaluating Classifiers for Emotional Signal on DEAP Dataset” [3]. Autorii au folosit o metodă bazată pe descompunerea Wavelet, obținând cele 5 benzi de frecvență, iar datele au fost analizate statistic

cu Principal Component Analysis (PCA). Activitatea fiecărei benzi pentru cele 32 de canale a fost stocată într-un vector pentru a fi în continuare etichetat prin valență și excitare. Clasificarea a fost făcută cu 4 tipuri de algoritmi de învățare: *SVM*, *K-Nearest Neighbor* (KNN), *Naive Bayes* (NB) și *Random Forest* (RF). Acuratețea cea mai bună a fost obținută cu RF, 98%, și cu SVM, 96%. KNN și NB au prezentat o precizie mai scăzută de 14%, respectiv 24%.

1.2. Emoția și sentimentele [4]

În psihologie, emoția este adesea definită ca o stare complexă care are ca rezultat schimbări fizice și psihologice care influențează gândirea și comportamentul. Emoția este asociată cu o serie de fenomene psihologice, inclusiv temperament, personalitate, dispoziție și motivație. Potrivit autorului David G. Myers, emoția umană implică „excitare fiziologică, comportamente expresive și experiență conștientă".

În timp ce emoțiile sunt asociate cu reacții corporale care sunt activate prin neurotransmițători și hormoni eliberați de creier, sentimentele sunt experiența conștientă a reacțiilor emoționale. Provenind din regiunile neocorticale ale creierului, sentimentele sunt generate de emoții și modelate de experiențe personale, credințe, amintiri și gânduri legate de acea emoție specială.

Emoțiile pot fi clasificate folosind modelul circumplex propus de James Russell care încadrează stările emoționale într-un spațiu circular, bidimensional, reprezentat de valență și excitare. Excitarea reprezintă axa verticală, valența reprezintă axa orizontală, iar centrul cercului reprezintă o valență neutră și un nivel mediu de excitare, după cum este prezentat în Figura 1.1.
În cadrul proiectului de diplomă s-a optat pentru analiza a două stări emoționale opuse: fericire și tristețe. Fericirea este caracterizată de valori mari ale valenței și excitării și se situează în primul cadran, în timp ce tristețea se încadrează în cel de-al treilea cadran și presupune valori mici ale perechii valență-excitare.

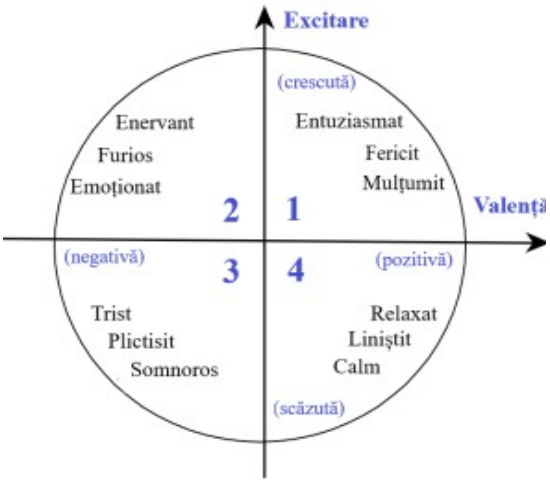

Figura 1.1: Modelul 2D dezvoltat de James Russell

Mehrabian și Russell au completat modelul 2D prin adăugarea celei de-a treia dimensiuni, dominanța, prezentat în Figura 1.2. Valența variază de la o stare de continuă de plăcere la o stare de nemulțumire, excitarea indică nivelul de implicare afectivă, iar dominanța marchează controlul asupra stării emoționale.

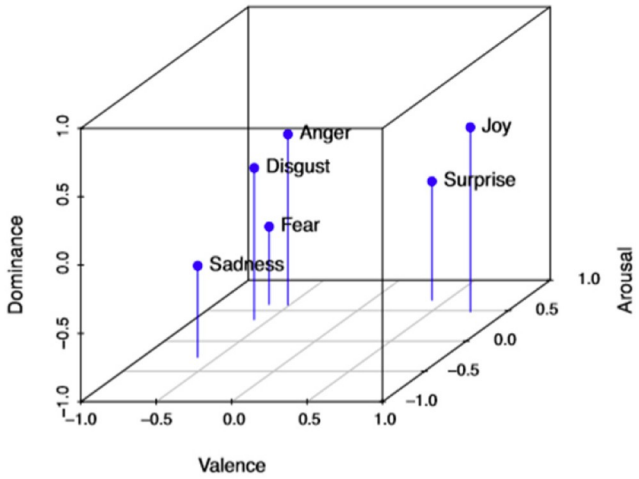

Figura 1.2: Modelul propus de Mehrabian și Russell

1.3. Semnale cerebrale și activitatea cortexului cerebral[5]

Activitatea creierului produce activitate electrică și magnetică. Prin utilizarea unor senzori, se pot detecta diferite tipuri de schimbări în activitatea electrică sau magnetică, la anumite momente de timp și pe anumite zone ale creierului.

Electroencefalografia (EEG) constituie înregistrarea activității electrice cerebrale prin intermediul unor senzori plasați pe suprafața scalpului. Această metodă este standardizată și folosită de câteva decenii atât în mediul clinic, cât și în cercetare. În Figura 1.3 se prezintă o interfață BCI bazată pe EEG. Echipamentul pentru înregistrarea EEG-ului este relativ ieftin, de dimensiuni și masă reduse și ușor de utilizat. Rezoluția temporală, adică abilitatea de a detecta schimbările într-un anumit interval de timp, este foarte bună.

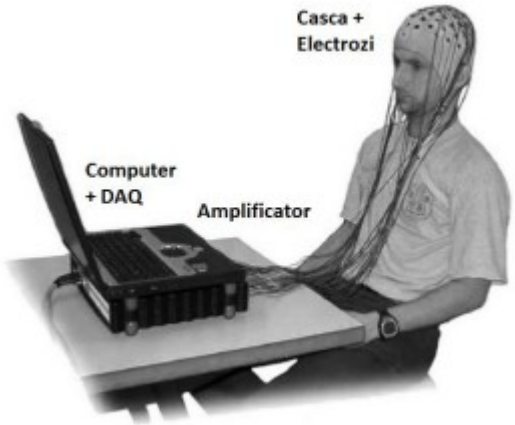

Figura 1.3: Componentele principale ale unei interfețe BCI, Graimann et al.(2010)

Cu toate acestea, EEG-ul prezintă câteva dezavantaje: rezoluția spațială (topografică) și domeniul de frecvențe limitate. De asemenea, EEG este susceptibilă la așa-numitele „artefacte” cauzate de alte activități electrice, cum ar fi, activități bioelectrice provocate de mișcarea ochilor sau clipiri (activități electrooculografice, EOG) și activități ale mușchilor (activități electromiografice, EMG) apropiați de zonele de înregistrare.

Pentru a se obține informații utile, semnalele EEG trebuie preprocesate, însemnând aplicarea unor filtre pentru a evidenția benzile de frecvență. Se disting cinci benzi de frecvență cu o anumită distribuție asupra scalpului: alfa, beta, delta, gamma și theta.

- **Delta**: domeniul de frecvență este 0-4 Hz. Apar la nou-născuții în dezvoltare și la adulți, în timpul somnului adânc. În plus, pot fi observate la adulții ale căror zone ale creierului au fost afectate de inflamație, de tumoare sau de blocaj vascular.

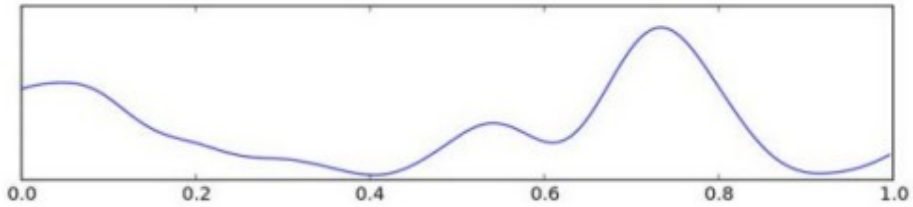

Figura 1.4: Oscilații delta [5]

- **Theta**: domeniul de frecvență este 4–7 Hz. Sunt întâlnite mai ales la copii și mai rar la adulți, în timpul somnului. Dacă apar în alte cazuri, acestea pot indica disfuncții cerebrale. Un exemplu ar fi starea de comă cauzată de medicamente sedative sau infecții severe care determină semnalele EEG să fie predominate de astfel de oscilații.

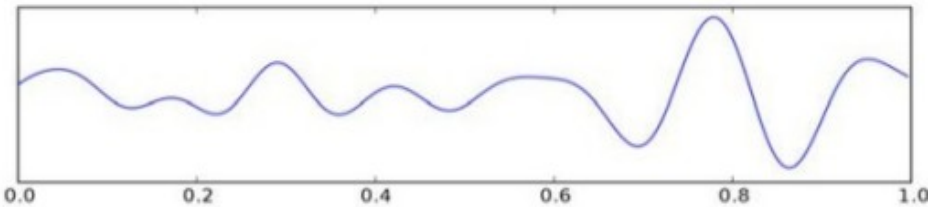

Figura 1.5: Oscilații theta [5]

- **Alpha**: domeniul de frecvență 7-13 Hz. Apar la adulți în timpul relaxării cu ochii închiși și se atenuează dacă există efort fizic sau concentrare asupra unei anumite sarcini specifice.

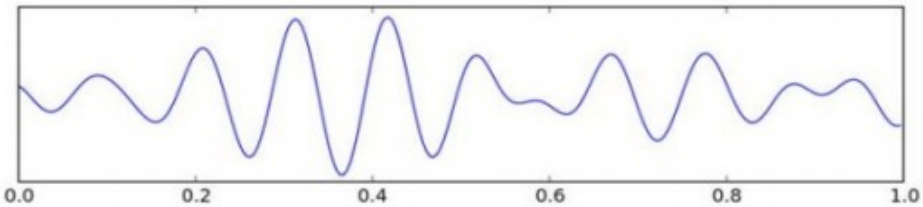

Figura 1.6: Oscilații alpha [5]

- **Beta**: domeniul de frecvență 14-30. Sunt frecvente în momentul atenției și concentrării la sarcini sau stimuli. Dacă un pacient deschide ochii sau începe activitatea mentală, undele alfa scad sau se atenuează, pentru a fi înlocuite cu unde beta.

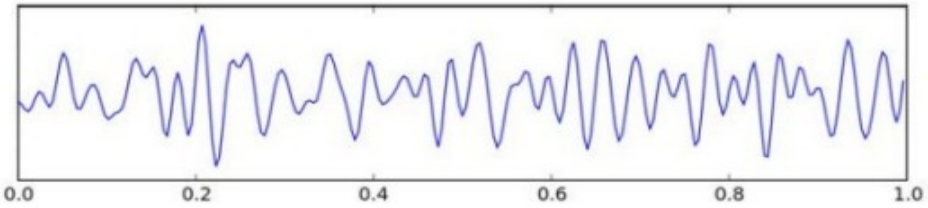

Figura 1.7: Oscilații beta [5]

- **Gamma**: domeniul de frecvență 30-100. Pot fi observate în cazul unui traumatism cranian, în cazul expunerii la toxine precum metale grele sau a prezenței unei boli generative a creierului, cum ar fi demența sau boala Alzheimer.

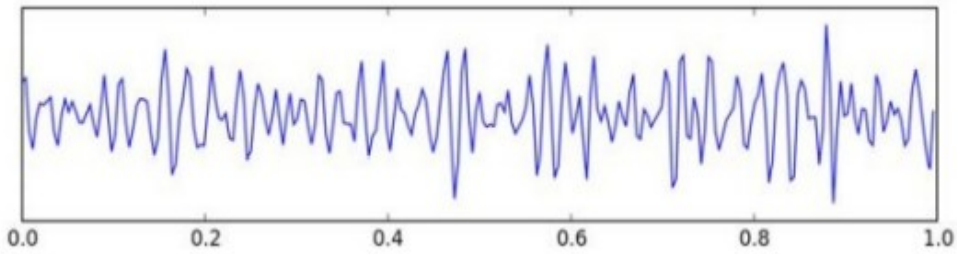

Figura 1.8: Oscilații gamma [5]

Dat fiind faptul că benzile delta și gamma sunt prezente în situații extreme, în timpul somnului adânc, respectiv în momentul apariției unui traumatism sau a unei boli, s-a optat pentru utilizarea benzilor alfa, beta și gamma deoarece sunt mai relevante pentru recunoașterea și analiza emoțiilor utilizatorilor.

Creierul este împărțit în două jumătăți - emisfera cerebrală stângă și emisfera cerebrală dreaptă. Fiecare emisferă este împărțită în lobi, după cum este prezentat în Figura 1.9:

- lobul frontal– este considerat centrul de control emoțional și este implicat în funcțiile motorii, rezolvarea de probleme, spontaneitate, memorie, controlul impulsurilor și comportamentul social și sexual;
- lobul parietal– controlează senzația tactilă, răspunsul la stimuli interni, înțelegerea senzorială, citirea și unele funcții vizuale;
- lobul occipital– intevine în procesul de recepționare și procesare a informațiilor vizuale, influențează modul în care sunt percepute culorile și formele;
- lobul temporal– este implicat în procesarea auditivă și găzduiește cortexul auditiv primar. Conține hipocamul și are un rol important în formarea memoriei pe termen lung.

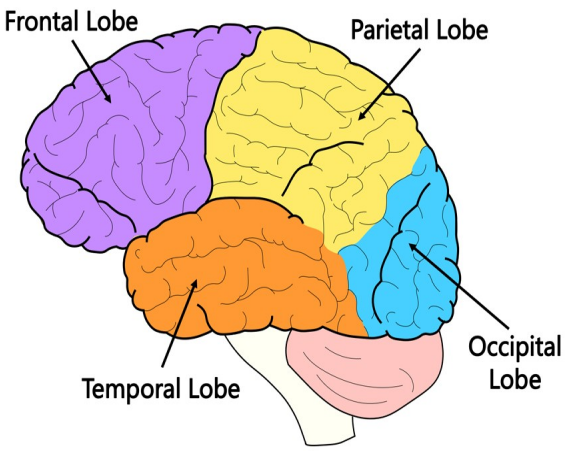

Figura 1.9: Lobii creierului

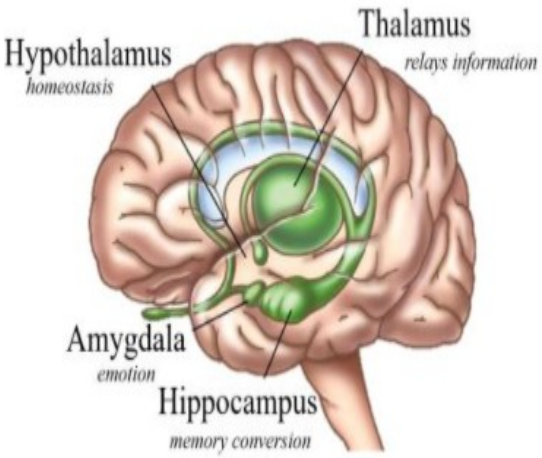

Figura 1.10: Sistemul limbic

Sistemul limbic, prezentat în Figura 1.10, este alcătuit dintr-un set de structuri cerebrale evolutive de bază sau primitive situate sub cortex, deasupra trunchiului cerebral. Acest sistem conectează hipotalamusul și talamusul cu alte zone ale lobilor frontali și temporali, inclusiv amigdala și hipocampusul. Acestea sunt implicate în multe dintre emoțiile și motivațiile noastre,

în special în cele legate de supraviețuire, controlând experiența și exprimarea emoțiilor, precum și unele funcții automate ale corpului.

Hipotalamusul are rol în activarea sistemului nervos simpatic care influențează reacțiile emoționale și în manifestarea emoțiilor. Părțile laterale ale hipotalamusului sunt implicate în emoții cum ar fi plăcerea și furia, în timp ce partea mediană este asociată cu aversiunea, nemulțumirea și o tendință de râs incontrolabil.

Talamusul servește ca releu senzorial; neuronii săi proiectează semnale atât către amigdală, cât și către regiunile corticale superioare pentru procesare ulterioară.

Amigdala, situată în lobii temporali stâng și drept ai creierului, a primit o mare atenție din partea cercetătorilor care investighează baza biologică a emoțiilor, în special a fricii și anxietății. Modificările structurii și funcției amigdalei au fost constatate la adolescenții care fie sunt supuși riscului fie au fost diagnosticați cu tulburări de dispoziție.

Hipocampusul integrează experiența emoțională cu cunoașterea. Trimite amintiri către emisfera cerebrală pentru a fi stocate pe termen lung și le recuperează la nevoie.

Având în vedere cele două emoții analizate în cadrul lucrării de diplomă, fericirea și tristețea, acestea au următoarele caracteristici. Pe de o parte, fericirea este asociată cu o stare de bine, de mulțumire și satisfacție totală și activează mai multe zone ale creierului, inclusiv cortexul frontal drept, precuneul, amigdala stângă și insula stângă, implicând conexiuni între conștientizare (cortexul frontal și insulă) și punctul central de control al sentimentelor (amigdala). Pe de altă parte, tristețea echivalează cu o stare de dezamăgire, slăbiciune și suferință. Este asociată cu o activitate crescută a lobului occipital drept, a insulei stângi, a talamusului stâng, a amigdalei și hipocampusului.

1.4. Analiza semnelelor cerebrale

În cadrul primei etape a proiectului, după împărțirea semnalelor EEG pe cele trei benzi de frecvență considerate, a fost necesară alegerea unor trăsături care să fie calculate pentru fiecare stimul/canal selectat, respectiv pentru fiecare bandă de frecvență asociată. S-au ales 4 trăsături importante: deviația standard (STD Dev), entropia șablon (Sample Entropy), rădăcina medie pătratică (RMS) și densitatea puterii spectrale (PSD).

1.4.1. Deviația standard

Abaterea sau deviația standard este o statistică care măsoară dispersia unui set de date în raport cu media sa. O valoare scăzută a acestei statistici indică faptul că datele analizate sunt apropiate de media setului (numită și valoare preconizată), în timp ce o valoare mare a abaterii indică faptul că valorile sunt distribuite pe o gamă mai largă de valori. Deviația standard este calculată ca rădăcina pătrată a variației prin determinarea abaterii fiecărei valori în raport cu media, formula generală putând fi observată mai jos:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n} (x_i - \bar{x})^2}{n-1}} \tag{1}$$

Valorile folosite în formula (1) reprezintă:

- σ - deviația standard a populației
- x_i - o valoare a populației
- $\bar{x}$ - valoarea medie a populației
- n – dimensiunea populației

Abaterea standard se calculează după cum urmează:

- Valoarea medie se calculează prin adăugarea tuturor valorilor populației și împărțirea la dimensiunea acesteia.
- Varianța se calculează scăzând media din fiecare valoare a setului de date, iar rezultatul se ridică la pătrat și se împarte la dimensiunea populației-1. Fiecare dintre aceste valori rezultate este apoi pătrată și rezultatele sunt însumate.

1.4.2. Entropia [6]

Entropia se definește ca informația conținută în distribuția unui semnal sau a unei serii de timp, devenind tot mai populară pentru utilizarea în periodicitatea sau regularitatea în date umane. Două dintre cele mai importante metode pentru date biologice sunt entropia aproximativă (*Approximate Entropy*) și entropia șablon (*Sample Entropy*).

Entropia aproximativă [6] (ApEn)

Entropia aproximativă constituie o tehnică utilizată pentru a cuantifica regularitatea și imprevizibilitatea fluctuațiilor datelor din serii de timp sau semnale. Pe măsură ce valoarea entropiei aproximative crește, devine tot mai dificil de prezis. Pentru a calcula entropia aproximativă sunt necesari următorii pași:

1. Se formează o serie de date u(1), u(2), u(3), ..., u(N). Acestea reprezintă N valori brute, măsurate la intervale de timp egale.

2. Se fixează un număr întreg, m, reprezentând lungimea datelor, și un număr real pozitiv, r, reprezentând nivelul de filtrare.

3. Se formează o secvență de vectori x(1), x(2), x(3), ..., x(N-m+1) în R^m , spațiul m-dimensional real definit de x(i)=[u(i), u(i+1),... , u(i+m−1)].

4. Folosind secvența x(1), x(2), ..., x(N-m+1), se construiește, pentru fiecare i, $1 \leqslant i \leqslant N-m+1$:

$C_i^m(r)$ = (numărul de x(j) astfel încât d[x(i), x (j)]≤r)/(N−m+1) (1)

, unde d [x , x‘] este definit ca: $d[v, v'] = \max_a |u(a) - u'(a)|$. (2)

În (2), u(a) sunt cele m componente scalare, d reprezintă distanța dintre vectorii v(i) și v(j) dată de diferența maximă în respectivele componente scalare.

5. Se definește:

$$\Phi^m(r) = (N-m+1)^{-1} \sum_{i=1}^{N-m+1} \log(C_i^m(r)) \quad (3)$$

7. Se definește entropia aproximativă drept:

$$ApEn = \Phi^m(r) - \Phi^{(m+1)}(r) \quad (4)$$

Entropia șablon[7] (SampEn)

Entropia șablon constituie o modificare a entropiei aproximative și este utilizată pentru evaluarea complexității chiar și în cazul unor serii de timp de dimensiuni mai mici. Entropia șablon are două avantaje importante față de cea aproximativă: independența lungimii datelor și o implementare relativ fără probleme. De asemenea, comparația dintre vectorul șablon și restul vectorilor include comparația cu vectorul însuși, astfel încât probabilitățile $C_i^{('m)}(r)$ nu vor avea valori nule.

Pentru o dimensiune m dată, o valoare de toleranță r și N, reprezentând numărul de puncte ale ale seriei, SampEn este echivalentă cu logaritmul natural negativ al probabilității ca, dacă două seturi de date simultane de lungime m au distanța mai mică decât r, atunci seturile de date de lungime m+1 au, de asemenea, distanța mai mică decât r.

Se consideră o serie de timp N = ($x_1, x_2, x_3, \dots, x_N$) cu un interval de timp egal, t. Se definește un vector template, de lungime m, $X_m(i)$ =($x_i, x_{(i+1)}, x_{(i+2)}, \dots, x_{(i+m-1)}$) și o funcție distanță $d[X_m(i), X_m(j)](i \neq j)$ care poate fi o distanță Chebîșev sau oricare alta, incluzând-o pe cea euclidiană. Se definește entropia șablon ca fiind:

$$SampEn = -\log\left(\frac{A}{B}\right) \tag{5}$$

În formula (5), valoarea A reprezintă numărul de perechi din vectorul template care au distanța $d[X_{(m+1)}(i), X_{(m+1)}(j)] < r$, iar B este egal cu numărul de perechi care au $d[X_m(i), X_m(j)] < r$. Prin urmare, entropia șablon va fi fie egală cu 0, fie va avea o valoare pozitivă.

Cu cât valoarea entropiei șablon este mai mică, cu atât gradul de similaritate a datelor va fi mai mare sau prezența zgomotului mai redusă.

1.4.3. Densitatea puterii spectrale[8]

Puterea spectrală $S_{xx}(f)$ a unei serii de timp, x(t), descrie distribuția puterii în componente de frecvență care compun semnalul. Conform analizei Fourier, orice semnal fizic poate fi descompus într-un număr de secvențe discrete sau într-un spectru de frecvențe pe un interval de timp continuu. Media statistică a unui anumit semnal (incluzând și semnalele afectate de zgomot) determinată în frecvență pentru toate componentele sale, se numește spectru.

Când energia semnalului se focusează pe un interval de timp finit și mai ales dacă energia sa totală este finită, atunci se poate calcula densitatea spectrală a energiei. Totuși, densitatea spectrală a puterii rămâne cea mai utilizată formă pentru acest tip de calcul, deoarece se aplică semnalelor care se manifestă pe o perioadă foarte mare de timp.

Densitatea spectrală de putere (Power Spectral Density - PSD) reprezintă măsurarea conținutului de putere al semnalului față de frecvență. PSD este de obicei utilizată pentru a

caracteriza semnalele aleatorii de bandă largă. Amplitudinea sa este normalizată prin rezoluția spectrală folosită pentru digitalizarea semnalului.
Pentru calcularea acestei trăsături, a fost utilizată metoda Welch. Aceasta este aplicată pentru a estima spectrul de putere al unui semnal dat. În fiecare secvență de timp dată este folosită o fereastră care poate sau nu să aibă posibilitatea de a se suprapune. Considerând {xd(n)} drept secvența de timp aleasă, unde d poate lua valori de la 1 până la L (numărul de intervale) și având M drept lungimea intervalului, atunci densitatea puterii spectrale, folosind metoda Welch este definită de formula: PSD poate fi estimată folosind metoda lui Welch, numită și metoda periodigramei. Dacă un semnal x(t) are transformata Fourier X(f), atunci densitatea spectrală de putere va fi:

$$\hat{pd}(f)=\frac{1}{MU}\left|\sum_{n=0}^{M-1} xd(n)\,w(n)\,\mathrm{e}^{-j2\pi f}\right|^2 \tag{6}$$

, pentru care U este factorul de normalizare al puterii în funcția fereastră, având reprezentarea din formula de mai jos. De asemenea, w(n) reprezintă datele după aplicarea ferestrei.

$$U=\frac{1}{M}\sum_{n=0}^{M-1}|w(n)|^2 \tag{7}$$

Astfel, spectrul puterii calculat cu funcția Welch este media tuturor periodogramelor definite prin:

$$PWelch(f)=\frac{1}{L}\sum_{i=0}^{L-1} Pd(f) \tag{8}$$

1.4.4. Rădăcina medie pătratică [9]

Rădăcina medie pătratică (RMS) se definește ca media aritmetică a pătratului unui set de numere sau pătratul funcției care descrie o formă de undă continuă. Este cunoscută și sub denumirea de medie cuadratică și constituie un caz particular al mediei generalizate cu exponent 2.

În cazul unui set alcătuit din n valori $(x_1, x_2, \ldots, x_n)$, RMS se calculează conform formulei (9):

$$x_{RMS}=\sqrt{\frac{1}{n}\left(x_1^2+x_2^2+\ldots+x_n^2\right)} \tag{9}$$

Formula echivalentă pentru o formă de undă continuă $f(t)$ definită pe intervalul $T_1 \le t \le T2$ este:

$$f_{RMS}=\sqrt{\frac{1}{T2-T1}\int_{T1}^{T2}|f(t)|^2 dt} \tag{10}$$

Dacă se calculează pe tot intervalul de timp, atunci relația (9) devine:

$$f_{RMS}=\lim_{T\to\infty}\sqrt{\frac{1}{2T}\int_{-T}^{T}|f(t)|^2 dt} \tag{11}$$

Dacă forma de undă este sinusoidală, atunci relația dintre amplitudini (vârf-la-vârf) și RMS sunt fixate și cunoscute, așa cum sunt pentru toate undele periodice și continue. Pentru cazurile în care a fost calculată și componenta PSD a semnalului, înălțimea vârfului obținut pentru aceasta este o estimare a amplitudinii RMS.

1.5. Analiza spectrogramelor semnalelor

1.5.1. Spectrograma semnalului [10]

O spectrogramă constituie o reprezentare vizuală a spectrului de frecvențe ale unui semnal, deoarece variază în funcție de timp. Atunci când sunt aplicate unui semnal audio, spectrogramele sunt uneori numite sonografe, amprente vocale sau voicegrame. Conform figurilor 1.11, 1.12 și 1.13, spectrogramele pot fi reprezentate sub forma unui grafic cu două dimensiuni geometrice: o axă reprezintă timpul, iar cealaltă axă reprezintă frecvența; o a treia dimensiune care indică amplitudinea unei anumite frecvențe la un anumit moment dat este reprezentată de intensitatea sau culoarea fiecărui punct din imagine. Pot fi generate pornind de la un semnal în două moduri:

- aproximate prin aplicarea unor filtre trece-bandă (acesta a fost singurul mod înainte de apariția procesării moderne a semnalului digital); această metodă se bazează pe procesarea analogică pentru a împărți semnalul de intrare în benzi de frecvență;
- calculate dintr-un semnal din domeniul timp folosind transformata Fourier; se bazează pe procesarea digitală. Datele eșantionate sunt împărțite în mai multe porțiuni care, de regulă, se suprapun. Se aplică transformata Fourier pentru a calcula ieșirea spectrului de frecvență pentru fiecare porțiune. Fiecare astfel de porțiune va corespunde unei linii verticale în spectrogramă, iar spectrele rezultate vor fi așezate fie unul lângă celălalt pentru a forma imaginea, fie parțial suprapuse, formând ferestre.

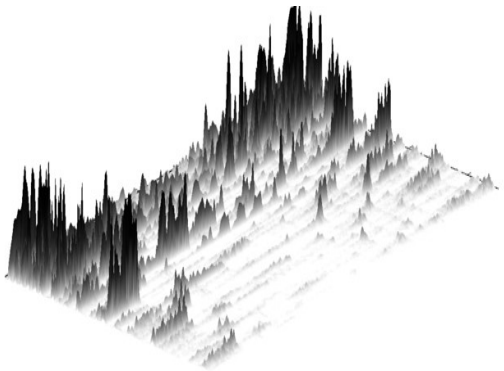

Figura 1.11: Spectrograma 3D a unei părți dintr-o melodie

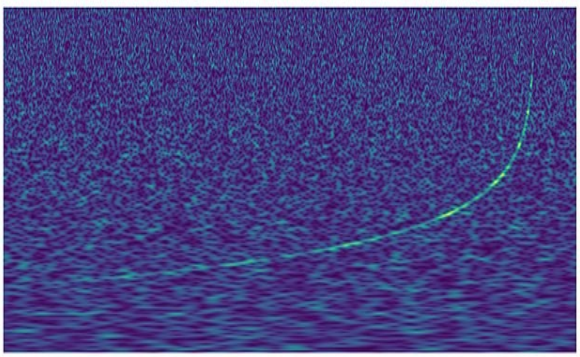

Figura 1.12: Spectrograma unei unde gravitaționale

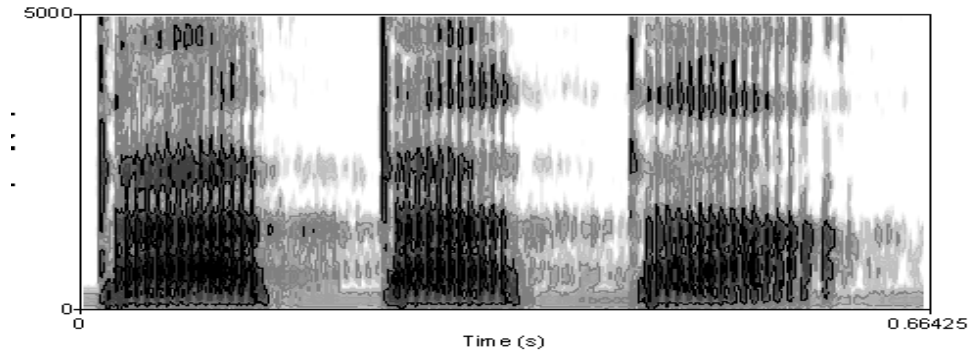

Figura 1.13: Spectrograma unei voci masculine care rostește „ta ta ta”

1.5.2. Extragerea trăsăturilor din spectrograme

În cadrul etapei a doua a lucrării de diplomă, se generează câte o spectrogramă a semnalului pentru fiecare canal selectat, iar pentru fiecare astfel de spectrogramă se calculează 4 trăsături importante: histograma în spațiul HSV, histograma gradienților orientați (Histogram of Oriented Gradients-HOG), textura folosind metoda Haralick și detecția keypoint-urilor cu Scale Invariant Feature Transform (SIFT).

1.5.2.1. Histograma imaginii în spațiul HSV [11]

În procesarea de imagini, histograma de culoare constituie o reprezentare a distribuției culorilor în imagine. Poate fi construită pentru orice spațiu de culoare, deși este mai utilizată în cadrul spațiilor tridimensionale, precum RGB sau HSV.

Spatiul de culoare HSV corespunde percepției vizuale umane a culorii. Poate fi reprezentat ca un hexacon tridimensional, în care axa verticală constituie valoarea intensității ce poate lua valori de la 0 la 255. Are trei componente:

- nuanță (Hue)- constituie porțiunea de culoare a modelului, exprimată ca un număr de la 0 la 360 de grade;
- saturație (Saturation)- descrie cantitatea de gri exprimată printr-un număr de la 0 la 100%. Cu cât saturația are valori mai apropiate de zero, cu atât se introduce mai mult gri și se crează un efect estompat;
- valoare sau strălucire (Value)- descrie luminozitatea sau intensitatea culorii, exprimată printr-un număr de la 0 (complet negru) la 100% (culoarea cu cea mai mare valoare a intensității).

În Figura 1.13 este reprezentată histograma de culoare a unei imagini pe toate cele trei componente.

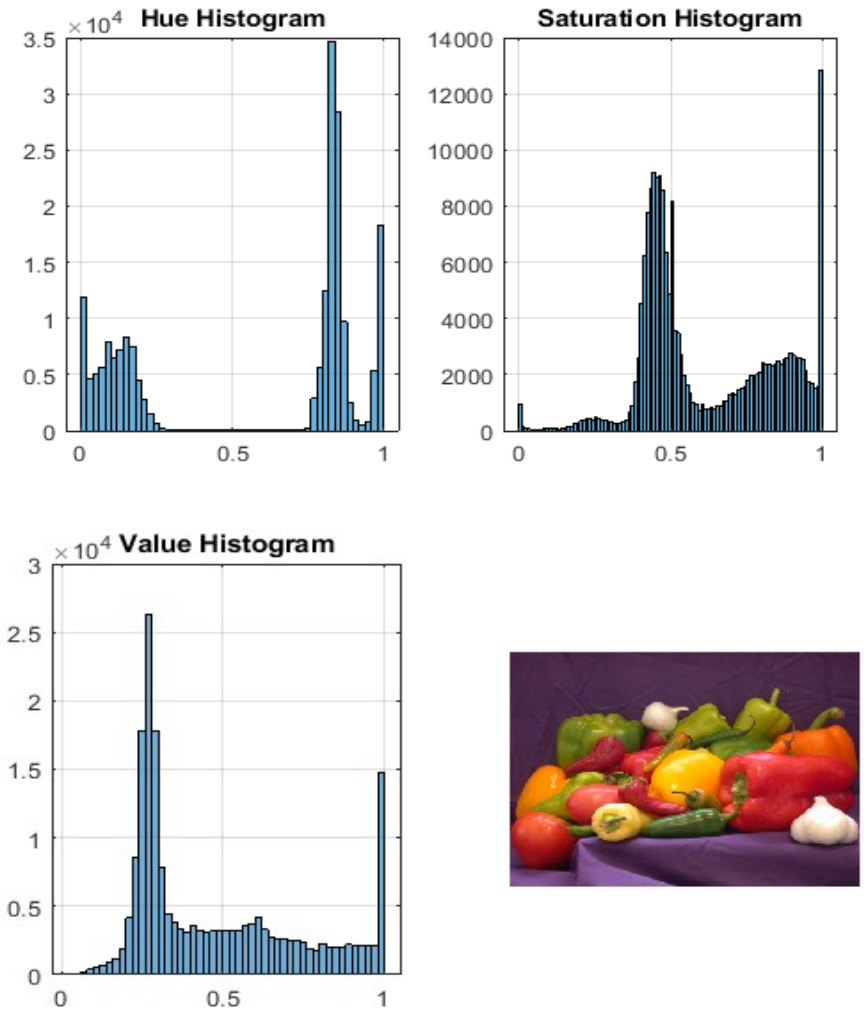

Figura 1.13: Histograma de culoare a unei imagini

1.5.2.2. Histograma Gradienților Orientați [12]

Histograma Gradienților Orientați reprezintă un descriptor utilizat în computer vision și procesare de imagini, pentru detecția obiectelor. Metoda numără aparițiile orientării gradientului în porțiuni localizate ale imaginii, asemănându-se cu histograma orientării muchiilor și descriptorii SIFT (Scale Invariant Feature Transform). Cu toate acestea, diferă prin faptul că este calculată pe o rețea densă de celule uniform distanțate și utilizează normalizarea contrastului local suprapus pentru o precizie îmbunătățită.

Principiul de calcul al acestei histograme se bazează pe faptul că apariția obiectelor locale și forma unei imagini pot fi descrise ca distribuție a intensității gradienților sau a intensității direcțiilor muchiilor. Imaginea este împărțită în regiuni conectate, de dimensiuni mici, numite celule, iar pentru pixelii din interiorul fiecărei celule, se calculează histograma pentru direcțiile gradientului. Descriptorul constituie concatenarea tuturor histogramelor calculate. Pentru o precizie îmbunătățită, histogramele locale pot fi normalizate prin contrast,

calculând o măsură a intensității pe o regiune mai mare a imaginii, numită bloc, și apoi folosind această valoare, se pot normaliza toate celulele din bloc.

Descriptorul HOG prezintă câteva avantaje cheie față de alți descriptori. Deoarece operează pe celule locale, este invariant la transformări geometrice și fotometrice. Mai mult, după cum au descoperit Dalal și Triggs, eșantionarea spațială grosieră, eșantionarea cu orientare fină și normalizarea fotometrică locală permit ca mișcarea individuală a corpului pietonilor să fie ignorată atât timp cât mențin o poziție aproximativ verticală. Descriptorul HOG este astfel adecvat în special pentru detectarea umană în imagini. [12]

1.5.2.3. Determinarea texturii prin metoda Haralick [13]

Trăsăturile de textură Haralick sunt descriptori utilizați în analiza imaginilor, mai ales pentru identificarea obiectelor și regiunilor de interes din imagine. Pentru a determina valorile acestor trăsături, nivelurile de gri ale imaginii se reduc, proces numit cuantizare. În lucrarea sa [15], Robert Haralick s-a concentrat asupra dezvoltării unor trăsături care să fie invariante la numărul cuantizării nivelurilor de gri. Redefinind matricea de co-ocurență la nivel de gri (Gray-Level Co-occurence Matrix-GLCM) ca o funcție de densitate de probabibilitate discretizată, aceasta devine invariantă asimptotic la cuantizare. Invariantul și trăsăturile originale au fost comparate folosind clasificarea prin metoda regresiei logistice, pentru a separa două clase în funcție de valorile texturii. Clasificatorii antrenați pe caracteristicile invariante au prezentat precizii mai mari și au avut performanțe similare pentru antrenări și imagini de testare cu cuantificări foarte diferite. Concluzia lucrării a fost că, folosind caracteristicile invariante Haralick, un model de imagine va genera aceleași valori ale caracteristicilor texturii, independent de cuantificarea imaginii.

Pentru a determina trăsăturile Haralick, sunt necesari o serie de pași:

- Se mapează nivelurile de gri din imaginea inițială, de dimensiune $M \times K$, I', din intervalul [a, b], in imaginea cuantizată, I, în intervalul [1, N], care are numărul de nivele de gri egal cu N. Se definește o funcție de cuantizare:

$$\varphi : [a,b]^{(M\times K)} \rightarrow [1,N]^{(M\times K)} \tag{12}$$

Se obține imaginea cuantizată:

$$I = \varphi(I') \tag{13}$$

- Se construiește matricea de co-ocurență a nivelelor de gri (GLCM), X, naturală, contorizând de câte ori fiecare pereche de nivele de gri apare în vecinătatea imaginii I sau într-o zonă arbitrară a imaginii. Fiecare element $X(i,j)$ din matrice se calculează conform formulei (14)

$$X(i,j) = \sum_{m=1}^{M}\sum_{k=1}^{K} 1, dacă\, I(m,k) = i \wedge I(m+d_x, k+d_y) = j \tag{14}$$

$$X(i,j) = \sum_{m=1}^{M}\sum_{k=1}^{K} 0, în\, caz\, contrar \tag{14'}$$

, unde d_x, d_y reprezintă deplasarea pe x și y în pixeli, iar $I(m,k)$ este elementul de pe pozițiile m,k în imaginea cuantizată, I.

- Se construiește matricea normalizată, P , în care fiecare element reprezintă probabilitatea estimată a fiecărei combinații de perechi de niveluri de gri învecinate din imagine.

$$P=\frac{X}{\sum_{i=1}^{N}\sum_{j=1}^{N}X(i,j)} \tag{15}$$

- Se calculează trăsăturile de textură din matricea de co-ocurență normalizată. Aceste trăsături sunt funcții ale elementelor și ale indicilor corespondenți din matrice și pot fi scrise conform formulei (16)

$$f=\sum_{i=a}^{A}\sum_{j=b}^{B}\Phi(i,j,g(P))\Psi(p(i,j)) \tag{16}$$

,unde Ψ reprezintă funcția unui element al matricei P, g este o funcție vectorială a lui P, iar Φ o funcție a indicilor și a lui g.

În Figura 1.14 poate fi observat modul de calcul al trăsăturilor Haralick prin efectuarea pașilor prezentați:

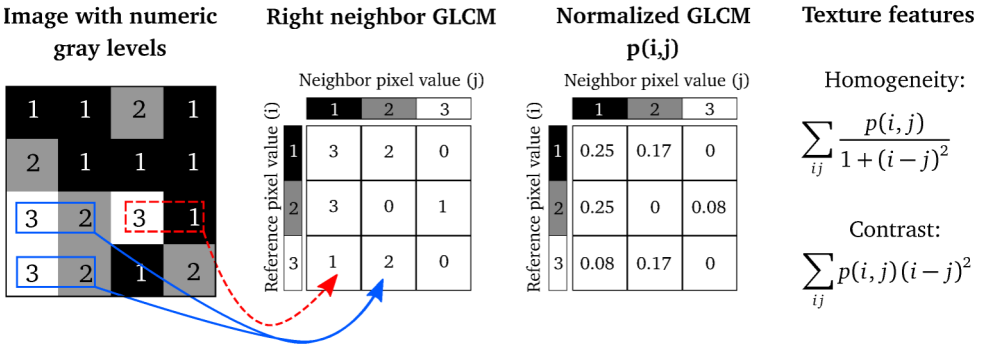

Figura 1.1: Modul de calcul al texturii pe baza nivelelor de gri

1.5.2.4. Scale Invariant Feature Transform (SIFT)

În 2004, D.Lowe, a propus un nou algoritm, *Scale Invariant Feature Transform (SIFT)*, utilizat pentru extragerea punctelor-cheie (*keypoints*) și a descriptorilor săi, în lucrarea [14].
În general, acest algoritm poate fi împărțit în 4 etape:

1. **Detecția punctelor-cheie**
   Algoritmul SIFT este invariant la scalare, ceea ce îi oferă întâietate în rândul celorlați algoritmi, cum ar fi, detectorul Harris. Acesta nu este caracterizat de inavarianță la scalare și, drept urmare, un colț poate deveni o margine numai dacă scala se modifică, după cum este ilustrat în Figura 1.15.

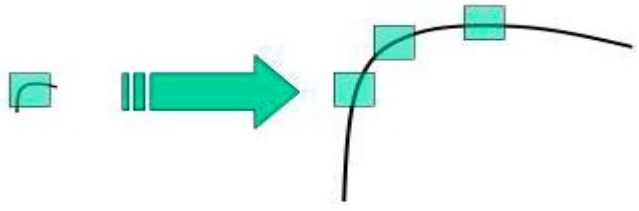

Figura 1. 15: Un colț se poate transforma în margine doar dacă scala se modifică

Scala unui reper din imagine este egală cu diametrul în imagine. Se notează cu σ, care se măsoară în pixeli.

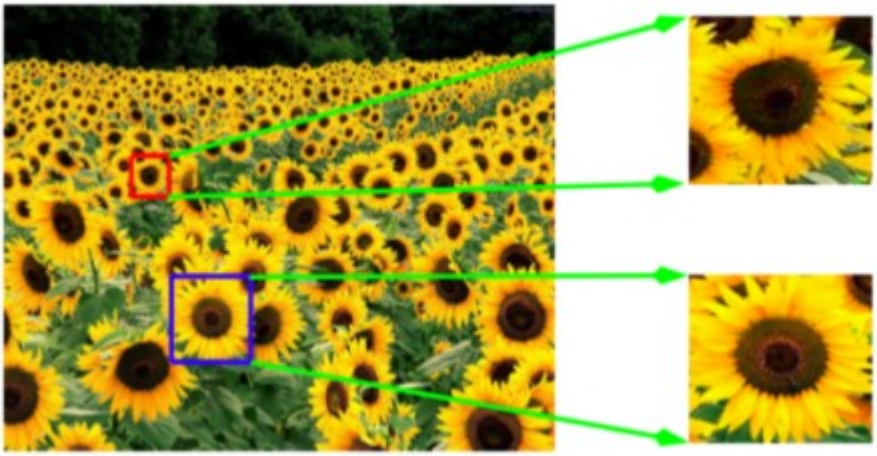

Figura 1.16: Trăsături similare la scale diferite

2. **Localizarea punctelor-cheie**
   Se detectează câteva puncte-cheie situate în regiunea celui mai apropiat vecin. Deoarece scala σ este cuantificată în puțini pași, aceste keypoint-uri sunt slab localizate. Se redefinește locația punctelor-cheie, numită localizare sub-pixel, eliminându-se acele puncte care prezintă trăsături mai puțin relevante. Aceasta etapă presupune utilizarea unei expansiuni Taylor care să se potrivească cu o suprafață 3D (în x, y și σ) din zona locală, pentru a interpola maximele sau minimele. Se folosește diferența Gaussianului (*Difference of Gaussian*) pentru a elimina punctele-cheie marginale și cele cu contrast scăzut, evidențiindu-se doar cele de interes.
3. **Asignarea unei orientări**
   Se apelează la calculului histogramei gradientului orientat (*HOG*). O histogramă de orientare este formată din orientările de gradient ale punctelor de eșantionare dintr-o regiune din jurul punctului-cheie. Fiecare eșantion adăugat este ponderat de magnitudinea gradientului său și de o fereastră circulară ponderată, de tip Gaussian, cu un σ care este de 1.5 ori mai mare decât scala punctului-cheie. Astfel, se consideră o fereastră de dimensiune 16×16 în jurul trăsăturii detectate, se determină orientarea muchiilor pentru fiecare pixel, se elimină muchiile slabe și se construiește histograma pentru orientările rămase.

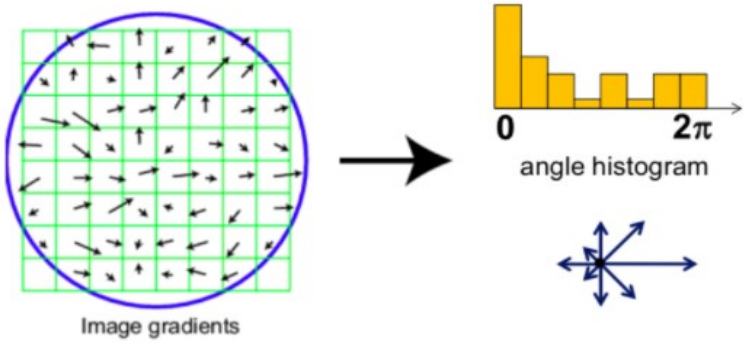

Figura 1.17: Determinarea histogramei pe baza orientărilor

4. **Generarea descriptorilor**

Se generează descriptorul care reprezintă o histogramă formată din gradientul imaginii gri. Se împarte fereastra de dimensiune 16×16 într-o grilă de celule de dimensiune 4×4 . Se calculează histograma orientărilor pentru fiecare celulă, rezultând un descriptor de dimensiune 128 ($16\,celule\times8\,orientări$).

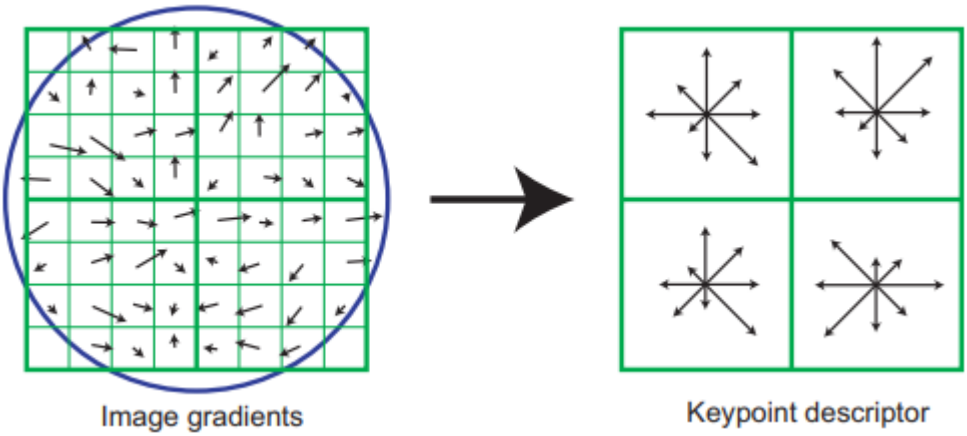

Figura 1.18: Generarea descriptorului SIFT

1.6. Support Vector Machine

Maşinile cu vectori suport (*Support Vector Machines-SVM*) reprezintă o metodă de învăţare de nouă generaţie (Shawe-Taylor & Cristianini, 2000), cu o fundamentare matematică riguroasă, bazată pe conceptul de maximizare a „marginii” care separă instanţele din două clase diferite, iar maximizarea este rezolvată analitic, nu empiric. Datorită acestei fundamentări, maşinile cu vectori suport au demonstrat performanţe foarte bune pentru probleme reale cum ar fi clasificarea textelor, recunoaşterea caracterelor scrise de mână, clasificarea imaginilor etc. În general, SVM-urile sunt considerate unele dintre cele mai bune metode de clasificare cunoscute la ora actuală.

Mașinile cu suport vectorial sunt o clasă de algoritmi supervizaţi de învăţare automată care utilizează concepte din teoria învăţării computaţionale. Suportul este o suprafaţă de separare optimală tipic neliniară în spaţiul de intrare şi liniară într-un spaţiu cu mai multe dimensiuni. În cadrul acestor algoritmi, se urmărește:

- determinarea unui hiperplan optimal între două clase;
- rezolvarea unor probleme liniar neseparabile ;
- proiectarea datelor într-un spațiu cu mai multe dimensiuni unde este mai simplă clasificarea cu suprafețe de decizie liniare.

SVM-urile pot fi de două tipuri:

- **liniari**-utilizați pentru date separabile liniar, ce pot fi clasificate în două clase folosind o singură linie dreaptă;
- **neliniari**-utilizați pentru date neseparabile liniar, adică nu pot fi clasificate folosind o linie dreaptă.

În figurile 1.19 și 1.20 pot fi vizualizată clasificarea datelor de intrare în două clase, prin determinarea unui hiperplan de separare, aplicând un algoritm liniar, respectiv neliniar.

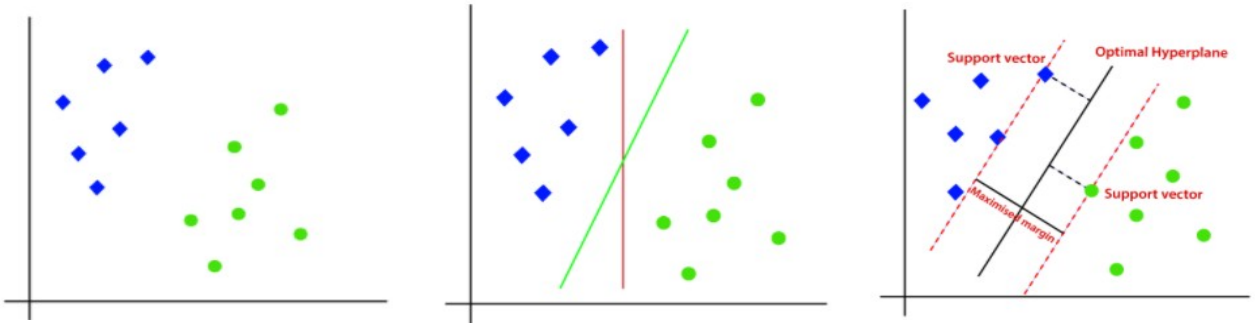

Figura 1.19: Determinarea hiperplanului în cazul unui SVM liniar

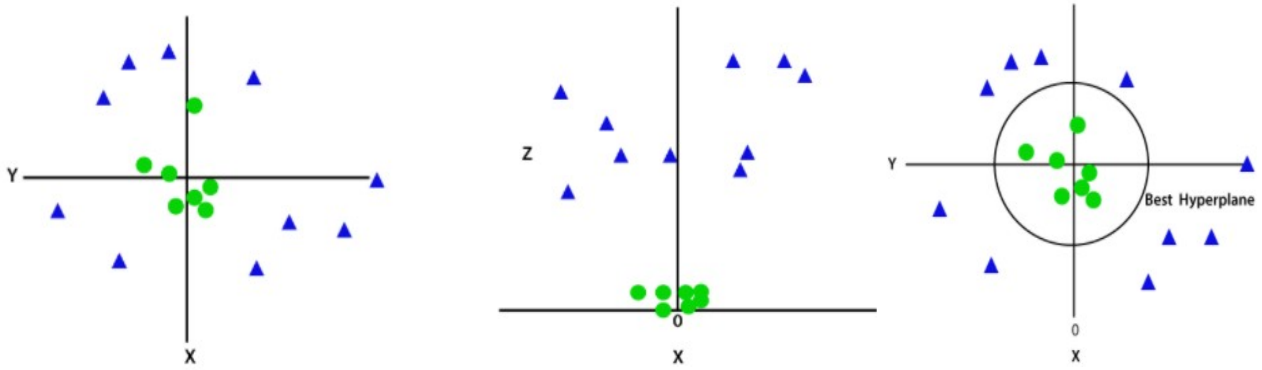

Figura 1.20: Determinarea hiperplanului în cazul unui SVM neliniar

Capitolul 2. Proiectarea aplicației

2.1. Setul de date DEAP

În cadrul proiectului de diplomă s-a ales un set de date potrivit pentru analiza stărilor emoționale ale utilizatorilor, și anume, ***Database for Emotion Analysis using Physiological Signals (DEAP)***. Aceasta conține date despre stimulii utilizați în experimente și semnalele achiziționate, informații despre participanți, împreună cu notele acordate de aceștia pentru fiecare stimul în funcție de aprecierea lor în ceea ce privește valența, excitarea și dominanța emoțiilor resimțite, și diverse alte detalii legate de modul în care a fost realizată preprocesarea semnalelor achiziționate.

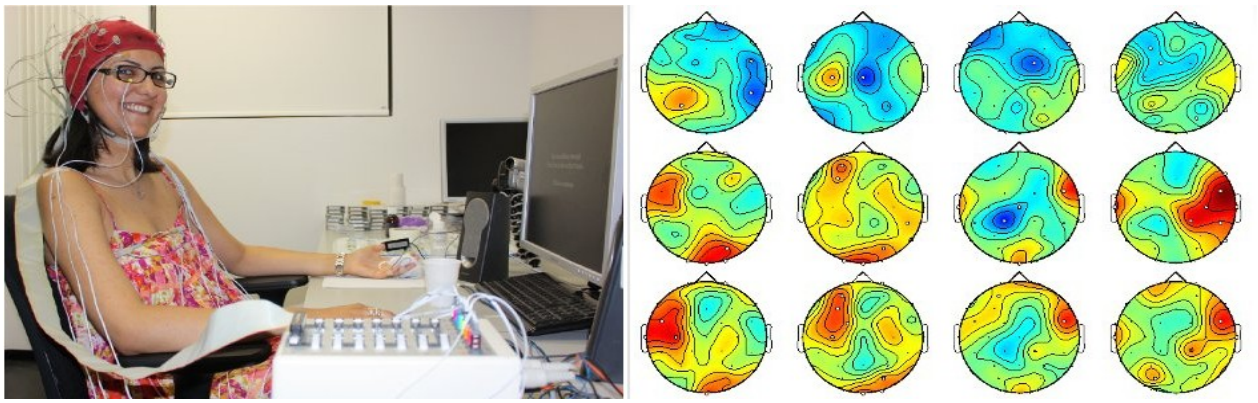

Figura 2.1: Imagini din timpul derulării experimentelor

Cercetătorii care au realizat baza de date au pus la dispoziția persoanelor interesate înregistrările semnalelor fiziologice și EEG a 32 de participanți, de vârste și genuri diferite. Fiecare dintre participanți a fost supus unui test, acesta constând în vizionarea a 40 de videoclipuri cu un diferit impact vizual și auditiv, timp în care au fost monitorizați cu senzori EEG și GSR. Evaluarea emoțională a videoclipurilor a fost făcută folosind grafica manechinelor de autoevaluare (*Self-Assesment Manikins – SAM*), prezentată în Figura 2.2 [16]. Au fost evaluate prin selecția celei mai potrivite grafici pentru fiecare experiment, pe o scară de la 1 la 9.

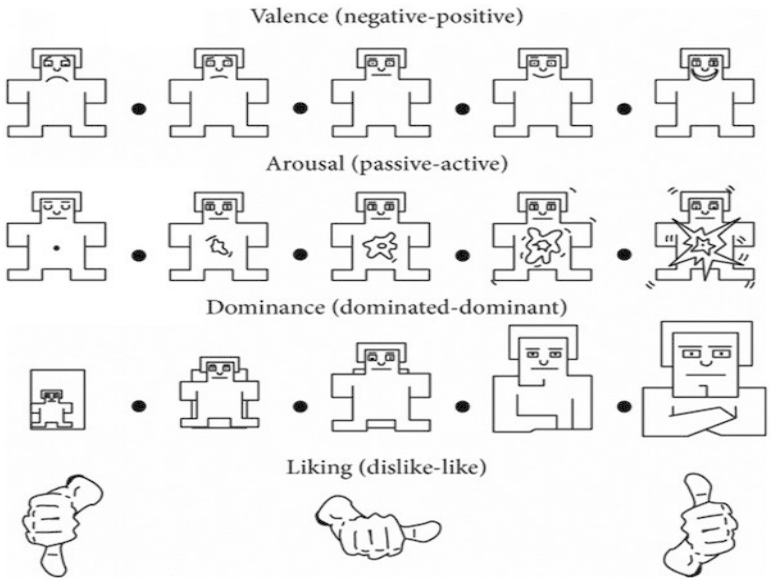

Figura 2.2: Grafica SAM folosită pentru evaluarea emoțiilor

Setul de date conține o serie de fișiere dintre care s-au folosit:

- **Video_list.xls**: acest fișier conține toate videoclipurile utilizate în experiment și informații relevante referitoare la: numărul experimentului, artistul, titlul și media valorilor pentru valență, excitare și dominanță.
- **Data_preprocessed**: aceste fișiere conțin toate datele EEG și înregistrările datelor fiziologice preprocesate.

Varianta bazei de date utilizate în cadrul lucrării de față este versiunea în care semnalele, deși inițial achiziționate cu o frecvență de 512 hertzi, au fost preprocesate pentru a putea eficientiza procesul de utilizare al acestora. Astfel, datele au suferit următoarele modificări:

- Datele au fost sub-eșantionate la 128 Hz.
- Au fost îndepărtate artefactele de tip ElectroOculoGramă (*EOG*), apărute ca rezultat al mișcării ochilor în timpul achiziției semnalelor.
- A fost aplicat un filtru trece-bandă de frecvență 4.0 – 45.0 hertzi.
- Au fost făcute modificări pentru ordinea canalelor EEG.
- Datele au fost segmentate în 40 de experimente (trials) a câte 60 de secunde.
- Experimentele au fost ordonate astfel încât toți utilizatorii să aibă aceeași ordine de prezentare a stimulilor, cu toate că în realitate acestea au fost prezentate în mod aleator.

Fiecare participant are un fișier asociat ce conține datele pentru cele 40 de canale și cele 4 etichete, împărțite în doi vectori:

- **data**: conține un vector de dimensiuni 40x40x8064 (experiment x canal x timp);
- **labels**: conține un vector de dimensiune 40x4 (experiment x eticheta) cu valorile pentru excitare, valență, dominanță și plăcere evaluate de participant pentru fiecare videoclip.

Organizarea canalelor a fost făcută astfel:

- 1 – 32: canale EEG;
- 33 – 36: canale EOG, care urmăresc mișcarea ochilor;

- 37– 40: răspunsul galvanic al pielii (Galvanic Skin Response – GSR), temperatură, centură de respirație și pletismograf. În cadrul temei alese, s-au folosit doar informațiile oferite de EEG.

2.2. Structura aplicației

Aplicația este structurată în două etape: în prima etapă s-a realizat preprocesarea, procesarea, clasificarea și analiza datelor semnalelor EEG, iar în etapa a doua, procesarea imaginilor semnalelor și clasificarea acestora. În acest sens, au fost proiectate câteva module principale pentru a realiza fiecare sarcină în parte.

Fiecare script din cadrul aplicației reprezintă o etapă a procesului de analiză a datelor, așa cum este prezentat în Figura 2.3:

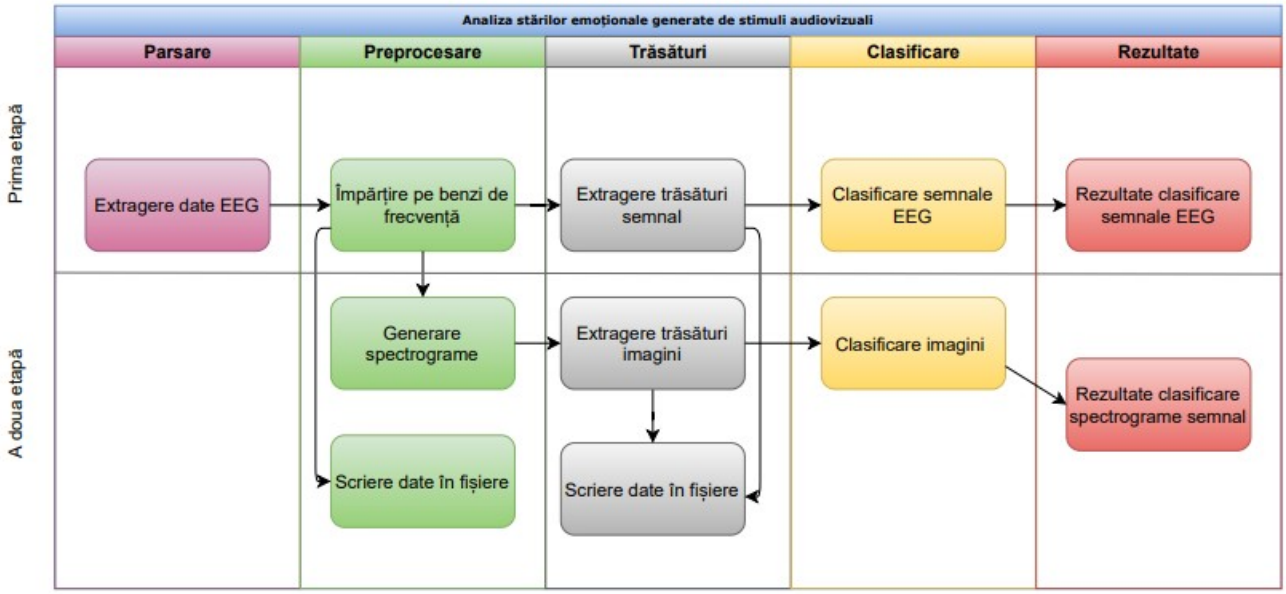

Figura 2.3: Modelul de comunicare între modulele aplicației

2.3. Modulul de extragere a semnalelor brute

În scriptul **ExtragereDate.py** se realizează preluarea și extragerea datelor EEG, urmate de etapa de selecție a etichetelor. Pentru fiecare utilizator, din fișierul cu extensia .dat au fost extrași și prelucrați separat cei doi vectori. Inițial, fiecare stimul avea 7680 de valori (128 Hz x 60 s). Pentru a reduce dimensiunea datelor, s-a împărțit lungimea fiecărui stimul în 10 intervale a câte 6 secunde și s-a extras doar un interval pentru fiecare canal.

Pentru fiecare utilizator s-au creat trei dataframe-uri:

- în fișierul **ProcesareDate/DataframeEtichete.dat** se stochează dataframe-ul cu valorile pentru valență, excitare, dominanță și plăcere;
- În fișierul **ProcesareDate/DataframeCanale.dat** se stochează datele grupate în funcție de stimul/canal;

- în fișierul **ProcesareDate/FisierSubtrial.dat** se stochează dataframe-ul corespunzător subtrial-ului extras pentru fiecare canal.

În faza de selecție a etichetelor s-a avut în vedere diferențierea valorilor mari (mai mari ca 5.5) de valorile mici (mai mici decât 3.5) ale valenței și excitării, eliminându-se astfel valorile neutre cu scopul de a obține un scor de acuratețe cât mai bun în urma etapei de clasificare. Aceste valori au fost binarizate conform celor două stări emoționale care se doresc a fi analizate:

- valorile mici pentru cele două etichete vor fi adnotate cu valori de 0, indicând o stare emoțională negativă-tristețe;
- valorile mari pentru cele două etichete vor fi adnotate cu valori de 1, indicând o stare emoțională pozitivă-fericire.

2.4. *Modulul de extragere a benzilor de frecvență și de calcul al trăsăturilor*

În scriptul **CalculTrasaturiSemnalePeBenzi.py** se realizează extragerea benzilor de frecvență și a trăsăturilor aferente fiecărei benzi. S-au ales 14 canale EEG (electrozi) de interes, prezentați în Tabelul 2.1.

Tabelul 2.1: Electrozii de interes și lobii asociați

Electrozi	*Lobi*
Cz, P7, P8	Lobul parietal
Fz, FC1, FC2, Fp1, Fp2, F3, F4	Lobul frontal
O1, O2	Lobul occipital
T7, T8	Lobul temporal

Pentru fiecare utilizator, benzile de frecvență vor fi extrase din dataframe-ul fiecărui canal rezultat din etapa de extragere a datelor. Obținerea benzilor se realizează prin aplicarea unui filtru Chebyshev invers de ordin 5. Frecvențele alese pentru benzi sunt prezentate în Tabelul 2.2. Descompunerea va fi aplicată pentru fiecare stimul. Dintre cele 4 benzi, au fost alese doar alpha, beta și gamma pentru că s-au dovedit a fi mai relevante pentru observarea emoțiilor participanților. După obținerea benzilor, datele rezultate vor fi stocate în 3 fișiere separate pentru benzile alpha, beta și gamma prezente în directorul **ImpartirePeBenziDeFrecventa**.

Tabelul 2.2: Frecvențele alese pentru benzile de frecvență

Bandă de frecvență	*Frecvență*
Delta	0
Theta	[4 – 7.5]
Alpha	[7.5 - 13]
Beta	[13 - 32]
Gamma	[32 - 64]

Scriptul **CalculTrasaturiSemnale.py** conține metode separate pentru calculul trăsăturilor alese: densitatea puterii spectrale, entropia șablon, deviația standard și rădăcina medie pătratică. Pentru fiecare bandă de frecvență extrasă se vor calcula cele patru trăsături. Datele rezultate vor fi stocate în fișierul **CalculTrasaturi/Trasaturi_semnale_EEG.csv**.

Pentru a vizualiza benzile extrase, s-a creat un script separat **PloteazaBenziFrecventa.py** ce conține metode pentru diferite afișări grafice.

2.5. Modulul de generare a spectrogramelor și de calcul al trăsăturilor

În scriptul **CalculTrasaturiSpectrograme.py** s-a generat, pentru fiecare participant, experiment și canal EEG selectat, câte o spectrogramă a semnalului. În total, pentru fiecare subiect, s-a creat un număr de spectrograme egal cu produsul dintre numărul de experimente relevante pentru emoțiile studiate și numărul de electrozi selectați.

Se va realiza o procesare a spectrogramelor, calculându-se patru trăsături importante: histograma în spațiul HSV, histograma gradienților orientați (HOG), textura folosind metoda Haralick și detecția keypoint-urilor cu Scale Invariant Feature Transform (SIFT). S-a optat pentru aplicarea aceluiași algoritm de clasificare folosit pentru date, și anume, Support Vector Machine (SVM), dar de această dată, intrările sunt reprezentate de trăsăturile calculate pentru spectrogramele semnalelor. Rezultatele calculate se scriu în fișierul **Spectrograme/Trasaturi_spectrograme.csv**.

2.6. Modulul de clasificare

În scripturile **ClasificareSemnal.py** și **ClasificareSpectrograme.py** este realizată clasificarea trăsăturilor calculate în funcție de două emoții: fericire și tristețe. Pentru fiecare bandă de frecvență se iau stimulii ce marchează aceste emoții și trăsăturile corespunzătoare. În Tabelul 2.3. sunt remarcați câțiva stimuli relevanți pentru emoțiile studiate.

Tabelul 2.3: Emoțiile și stimulii aferenți

Emoție	*Număr stimul*	*Artist*	*Titlu*	*Link video*
Fericire	11	Michael Franti	Say Hey (I Love You)	http://www.youtube.com/watch?v=eoaTl7IcFs8
	14	Jason Mraz	I'm Yours	http://www.youtube.com/watch?v=EkHTsc9PU2A
Tristețe	24	James Blunt	Goodbye My Lover	http://www.youtube.com/watch?v=wVyggTKDcOE
	23	Johnny Cash	Hurt	http://www.youtube.com/watch?v=AO9dbmJ_2zU

Clasificarea se realizează pe baza modelului circumplex propus de James Russell, prezentat în Figura 1.1., cele două emoții situându-se în cadranul 1, respectiv 3.

S-a utilizat algoritmul Support Vector Machine (SVM) ale cărui intrări sunt reprezentate de trăsăturile aferente semnalelor și imaginilor, calculate în etapa anterioară pentru fiecare subiect, iar ca ieșiri se consideră valorile binarizate pentru valență și excitare rezultate în urma etapei de selecție a etichetelor. S-au împărțit datele în 80% pentru antrenare și 20% pentru testare.

2.7. Mediul de dezvoltare și biblioteci utilizate

În cadrul aplicației pentru analiza datelor EEG, s-a ales folosirea limbajului de programare **Python** datorită librăriilor și a instrumentelor ce vin în ajutorul proiectelor de tipul Data Science. De la operațiuni pe date până la selecție de trăsături și implementarea diverselor modele pentru învățarea automată, Python poate oferi soluții pentru toate acestea datorită librăriilor sale. De asemenea, programele scrise în acest limbaj presupun o mentenanță scăzută și sunt foarte robuste, având o sintaxă ușor de folosit și de înțeles.

Aplicația a fost dezvoltată în **PyCharm Community Edition 2020.2.1 x64**, un mediu de dezvoltare integrat (IDE) utilizat în programarea computerelor, în special pentru limbajul Python. Este dezvoltat de compania cehă JetBrains (cunoscută anterior ca IntelliJ). Oferă facilități, precum: analiză de cod, debug grafic, testare unitară, integrare cu sisteme de control al versiunilor (VCSes).

Pentru a obține rezultatele dorite, a fost necesară instalarea câtorva biblioteci:

- ***pickle*** – încărcarea fișierelor cu extensia .dat; pentru fiecare participant există un fișier în format binar care conține informații despre date și etichete;
- ***pandas*** **–** crearea dataframe-urilor care să stocheze valori pentru etichete și canale, în etapa de parsare a datelor;

- ***numpy*** **-** calculul funcțiilor matematice (logaritm, maxim, deviație standard, radical) implicate în extragerea trăsăturilor;
- ***scipy*** **–** aplicarea unui filtru Chebyshev invers, de ordin 5, necesară împărțirii pe benzi de frecvență;
- ***sklearn*** **–** implementarea algoritmului Support Vector Machine pentru clasificarea semnalelor și a spectrogramelor semnalelor;
- ***matplotlib*** **–** reprezentarea grafică a benzilor de frecvență și generarea spectrogramelor
- ***cv2*** **–** procesarea spectrogramelor semnalului (citire, convertire la modelul HSV sau grayscale, redimensionare, calculul histogramei)
- ***skimage*** **–** determinarea histogramei gradientului orientat (HOG)
- ***mahotas*** **–** calculul texturii spectrogramelor
- ***statistics*** **–** determinarea mediei trăsăturilor calculate în cazul spectrogramelor semnalelor.

2.8. Dificultăți întâmpinate

Au fost întâmpinate dificultăți în etapa de împărțire pe benzi de frecvență, căci inițial s-a optat pentru descompunerea Wavelet care s-a dovedit a fi nepotrivită, deoarece nu se calculau toate trăsăturile pentru utilizatori. Ulterior, s-a ales o altă abordare, și anume, un filtru Chebyshev invers de ordin 5, care a permis o împărțire mai exactă pe benzile de frecvență și care, din fericire, a funcționat. O reală provocare a fost și înțelegerea modului în care trebuie furnizate date către algoritmul de clasificare astfel încât să se obțină o acuratețe cât mai mare. În acest sens s-a realizat selecția valorilor valenței și excitării în funcție de un prag minim și unul maxim, eliminându-se valorile nerelevante, cuprinse între 3.5 și 5.5.

Volumul mare de date de intrare face ca algoritmul de clasificare să dureze destul de mult. În acest sens, în prima fază de clasificare a datelor, s-a ales un subtrial al semnalului pe fiecare stimul și nu valorile pe întreg semnalul. De asemenea, este necesar de menționat că, datorită volumului de date și a efortului de calcul, s-a optat pentru utilizarea unei stații de la facultate și nu a laptopului personal.

Capitolul 3. Implementarea aplicației

3.1. Extragerea semnalelor brute

În **ExtragereDate.py** se realizează încărcarea fișierelor pentru fiecare participant și descompunerea datelor din fișier în cei doi vectori: *data* și *labels*. Având un format binar, fișierele sunt încărcate prin apelul metodei *load* din modulul *pickle*. Secvența de cod corespunzătoare este:

```
if subiect % 1 == 0:
    if subiect < 10:
        numeSubiect = '%0*d' % (2, subiect + 1)
    else:
        numeSubiect = subiect + 1

fisierDataframeCanale.write("\nSubiect " + subiect.__str__() + "\n")
numeFisier = "data_preprocessed/s" + str(numeSubiect) + ".dat"
# dictionar pentru canale selectate
dictionarCanale = {'Cz': None,'Fp1': None,'F3': None,'Fp2': None,
                   'F4': None, 'O2': None,'O1': None,'P8': None,
                   'P7': None,'T7': None,'T8': None,'FC1':None,
                   'FC2':None,'Fz':None}
# Se citesc datele din fisierele .dat folosind metoda load din biblioteca
pickle
dateCuPickle = pickle.load(open(numeFisier, 'rb'), encoding='latin1')
dateCanale = dateCuPickle['data']
```

```
if subiect % 1 == 0:
    if subiect < 10:
        numeSubiect = '%0*d' % (2, subiect + 1)
    else:
        numeSubiect = subiect + 1
numeFisier = "data_preprocessed/s" + str(numeSubiect) + ".dat"
fisierDataframeEtichete.write("\nSubiect " + subiect.__str__() + "\n")
dateCuPickle = pickle.load(open(numeFisier, 'rb'), encoding='latin1')
etichete = dateCuPickle['labels']
```

Vectorul *labels* conține evaluările participanților, corespunzătoare celor patru etichete (valență, excitare, dominanță, plăcere), valori care se vor stoca într-un DataFrame *pandas*:

```
# dictionar de etichete
dictionarDeEtichete = {'Valenta': list(etichete[:, 0]),
                       'Excitare': list(etichete[:, 1]),
                       'Dominanta': list(etichete[:, 2]),
                       'Placere': list(etichete[:, 3])}
# Se creaza un dataframe pentru etichete
dataframeEtichete = pd.DataFrame(dictionarDeEtichete)
```

Se consideră două valori prag (3.5 pragul inferior și 5.5 pragul superior) pentru selecția valenței și excitării, cu scopul de se evita stimulii nerelevanți pentru observarea stării emoționale, ceea ce determină obținerea unei acurateți mai bune, în urma etapei de clasificare.

```
dfEtichete = extrageDateEtichete(subiect)
for experiment in range(numarExperimente):
    if (dfEtichete['Valenta'][experiment] <= 3.5 and dfEtichete['Excitare']
[experiment] <= 3.5) or (
            dfEtichete['Valenta'][experiment] >= 5.5 and
dfEtichete['Excitare'][experiment] >= 5.5):
        # valorile etichetelor ce se incadreaza in cele 4 intervale se vor
stoca in vectorii corespunzatori, de asemenenea si indecsii experimentelor
relevante
        valori_valenta.append(dfEtichete['Valenta'][experiment])
        valori_excitare.append(dfEtichete['Excitare'][experiment])
        experimente.append(experiment)
    else:
        dfEtichete = dfEtichete.drop(labels=experiment, axis=0)
```

Se realizează o binarizare a valorilor selectate în etapa descrisă mai sus pentru a putea fi utilizate ca ieșiri pentru algoritmul *Support Vector Machine*. Valorile mari ale valenței și excitării (mai mari decât 5.5) sunt adnotate cu 1, iar valorile mici (mai mici decât 3.5) cu 0.

```
# Se va realiza o binarizare a datelor pentru etichete in scopul de a putea
fi utilizate ca iesiri pentru algoritmul de clasificare
def binarizeazaDate (subiect, margineInf, margineSup, experiment):
    # Se extrag mai intai valorile etichetelor
    dfEtichete = extrageDateEtichete(subiect)
    # Daca valorile pentru valenta si excitare sunt mai mici sau egale cu
marginea inferioara (3.5), vor deveni 0
    if dfEtichete['Valenta'][experiment] <= margineInf and
dfEtichete['Excitare'][experiment] <= margineInf:
        return 0
    # Daca valorile pentru valenta si excitare sunt mai mari sau egale cu
marginea superioara (5.5), vor deveni 1
    elif dfEtichete['Valenta'][experiment] >= margineSup and
dfEtichete['Excitare'][experiment] >= margineSup:
        return 1
```

Se împarte semnalul EEG în 10 intervale a câte 6 secunde fiecare și se reține subtrial-ul de la jumătatea semnalului într-un DataFrame *pandas:*

```
for channel in CANALE_SELECTATE:
    #Se imparte semnalul in 10 intervale  a cate 6 s
    for split in range(0, splits):
        #Se ia un sub-interval de la jumatatea semnalului
        if split == 5:
                canalDict={channel: dataFrameCanale[channel][split * 768:
(split + 1) * 768]}
                dictionarSubtrial[channel]=canalDict[channel]
#Se va crea un dataframe care sa contina acest subtrial
df = pd.DataFrame(dictionarSubtrial, columns=['Cz', 'Fz', 'FC1', 'FC2',
'Fp1', 'Fp2', 'F3', 'F4', 'O1', 'O2', 'P7', 'P8', 'T7', 'T8'])
```

3.2. *Împărțirea pe benzi de frecvență*

În **CalculTrasaturiSemnalePeBenzi.py** se realizează extragerea benzilor de frecvență alpha, beta, theta și gamma. Pentru fiecare participant, benzile de frecvență vor fi extrase din DataFrame-ul fiecărui canal rezultat în etapa de parsare. În acest sens, a fost utilizat un filtru Chebyshev II, cunoscut și sub numele de filtru Chebyshev invers. Pentru implementare s-a folosit metoda *cheby2* al modulului *signal* din cadrul bibliotecii Python numită *scipy*. Metoda proiectează un filtru Chebyshev invers, analog sau digital, de ordin n, și returnează coeficienții filtrului. Funcția face parte dintr-o serie de metode de proiectare a unor filtre asemănătoare modalității de obținere a acestora în MATLAB. În acest caz a fost utilizat un filtru trece-bandă cu o atenuare de 3 decibeli, de ordin 5, care în funcție de tipul benzii dorite, are diferite frecvențe de tăiere. Cele 3 funcții implementate în acest scop pot fi observate mai jos:

```
def aplicaChebyshevInvers(frecventa_low, frecventa_high, fs, order=5):
    frecventa_nyq = 0.5*fs
    low = frecventa_low / frecventa_nyq
    high = frecventa_high / frecventa_nyq
    b, a = signal.cheby2(order, 3, [low, high], btype='bandpass')
    return b, a
def aplicaFiltruChebyshevInvers(data, frecventa_low, frecventa_high, fs,
order=5):
    semnal_eeg = np.array(data)
    b, a = aplicaChebyshevInvers(frecventa_low, frecventa_high, fs,
order=order)
    semnal_filtrat = signal.lfilter(b, a, semnal_eeg)
    return semnal_filtrat
def verificaTipBanda(tipBanda):
    fs = 128
    if tipBanda == "Alpha":
        frecventa_low = 7.5
        frecventa_high = 12.5
    elif tipBanda == "Beta":
        frecventa_low = 13.0
        frecventa_high = 30.0
    else:
        frecventa_low = 30.0
        frecventa_high = 63.5
    return frecventa_low, frecventa_high, fs
```

3.3. *Extragerea trăsăturilor semnalelor*

În **CalculTrasaturiSemnale.py** s-au implementat metode de calcul al trăsăturilor benzilor de frecvențe extrase. După cum a fost precizat și în primul capitol, acestea sunt rădăcina medie pătratică (RMS), deviația standard (STD Dev), densitatea puterii spectrale (PSD) și entropia șablon (SampEn). Pentru RMS, PSD și STD Dev au fost extrase valorile maxime din mulțimile de valori obținute.

Pentru determinarea densității puterii spectrale și a rădăcinii medii pătratice s-a folosit librăria specializată pentru calcule matematice *scipy*. Calculul cu ajutorul metodei *welch* este prezentat în următoarea secvență de cod. Această metodă calculează o estimare a densității puterii spectrale prin împărțirea datelor în segmente suprapuse, estimând mai apoi o periodogramă modificată prin calculul mediei periodogramelor segmentelor. Motivul pentru care a fost utilizată și în calculul RMS este acela că amplitudinea căutată pentru acest proiect este reprezentată de rădăcina obținută din cel mai înalt vărf calculat în PSD, după cum se poate observa și în cod, în formula folosită.

```
#Metoda care calculeaza densitatea puterii spectrale
def calculeazaDensitateaPuteriiSpectrale(banda, low, high):
    low = 4*low
    high = 4*high
    frecventa, psd = signal.welch(banda,  nperseg=len(banda),
scaling='spectrum')
    #Se preiau datele din domeniul de frecventa pentru fiecare banda
    psd = psd[int(low):int(high)]
    return np.max(psd)
```

```
#Metoda care calculeaza radacina medie patratica
def calculeazaRMS(date):
     sampling_rate = 128
     frecventa, psd = signal.welch(date, sampling_rate, scaling='spectrum')
     amplitudineRMS = np.sqrt(psd.max())
     return amplitudineRMS
```

```
#Metodă pentru calcularea entropiei sablon
def calculeazaEntropieSablon(date, m, r = None):
    def determinaDistantaMaxima(xi, xj):
        return max([abs(ua - va) for ua, va in zip(xi, xj)])
    def determinaPhi(m):
        x = [[date[j] for j in range(i, i + m - 1 + 1)] for i in range(N -
m + 1)]
        C = [len([1 for j in range(len(x)) if i != j and
determinaDistantaMaxima(x[i], x[j]) <= r])for i in range(len(x))]
        return sum(C)
    N = len(date)
    return -np.log(determinaPhi(m+1) / determinaPhi(m))
```

Pentru calculul deviației standard a fost utilizată funcția *std* din cadrul *numpy*.

```
#Metoda care calculeaza deviatia standard
def calculeazaDeviatiaStandard(date):
    semnal = np.asarray(date)
    max = np.max(np.std(semnal))
    return max
```

3.4. Clasificarea semnalelor EEG

În **ClasificareSemnaleEEG.py** s-a folosit librăria *Scikit-learn* pentru implementarea algoritmilor de învățare automată. Librăria conține o varietate de algoritmi de clasificare, regresie și clusterizare. Pe lângă algoritmii de învățare automată, au fost folosite metrici pentru evaluarea performanței metodelor din Scikit-learn, *Accuracy score* și *confusion_matrix*. Pentru fiecare participant, se realizează o selecție a valorilor etichetelor și se reține doar acei stimuli relevanți pentru observarea celor două emoții studiate. În funcție de fiecare stimul/experiment, se împarte semnalul pe benzi de frecvență și se calculează trăsăturile ce vor reprezenta intrările algoritmului de clasificare. De asemenea, se efectuează o binarizare a valorilor valenței și excitării aferente fiecărui stimul, iar rezultatele obținute, echivalente celor două clase (0 – tristețe, 1 - fericire), vor constitui ieșirile algoritmului. Secvența de cod următoare este relevantă în acest sens:

```
# Se realizeaza o filtrare a etichetelor functie de marginea inferioara
(3.5) si cea superioara (5.5)
    # Se determina experimentele relevante
    valenta, excitare, experimente =filtrareDupaEtichete(subiect)
    valoriEtichete = extrageDateEtichete(subiect)
    #Se calculeaza trasaturile
    for exp in experimente:
        print("Experiment "+ exp.__str__())
        tr = impartireBenziFrecventa(subiect=subiect,experiment=exp)
        trasaturi.append(tr)
        # Se binarizeaza datele pentru cele 2 etichete: valenta si excitare
pentru experimentul exp
        valoarebinara = binarizeazaDate(subiect, 3.5, 5.5, exp)
        dateBinarizate.append(valoarebinara)
        if (valoriEtichete['Valenta'][exp] <= 3.5 and
valoriEtichete['Excitare'][exp] <= 3.5) :
            fisierTristete.write(str(subiect)+" "+str(exp)+"
"+str(valoriEtichete['Valenta'][exp])+" "+str(valoriEtichete['Valenta']
[exp])+"\n")
        elif (valoriEtichete['Valenta'][exp] >=5.5 and
valoriEtichete['Excitare'][exp] >= 5.5):
            fisierFericire.write(str(subiect)+" "+str(exp)+"
"+str(valoriEtichete['Valenta'][exp])+" "+str(valoriEtichete['Valenta']
[exp])+"\n")
# Lungimea datelor binarizate este egala cu suma numarului de experimente
relevante pentru utilizatori
lungime  = len(dateBinarizate)
# Pentru fiecare experiment la care este supus utilizatorul se calculeaza
168 de trasaturi (4 trasaturi X 3 benzi X 14 canale selectate)
trasaturi = np.array(trasaturi).reshape(lungime, 168)
print(trasaturi)
print(dateBinarizate)
# Iesirile vor fi reprezentate de doua clase:
#0-indica o stare negativa
#1-indica o stare pozitiva
dateBinarizate=np.array(dateBinarizate)
```

S-a folosit metoda *train_test_split* din *sklearn.model.selection*, pentru crearea seturilor de date de antrenare și a celui de testare. Această metodă extrage dintr-un set de date inițial, partiții alese aleatoriu pentru a crea subsetul de antrenare și cel de test. S-a ales ca 80% din setul de date să fie alocat pentru antrenare, iar 20% pentru testare. Pentru a implementa algoritmul de clasificare *Support Vector Machine* s-a apelat metoda *SVC* din modelul *sklearn.svm*, cu un kernel liniar, adică hiperplanul de separare dintre cele două clase va fi reprezentat de o singură linie dreaptă. Un exemplu de utilizare este prezentat în următoarea secvență de cod.

```
# Se impart datele: 80% pentru antrenare si 20% pentru testare
# intrarile sunt reprezentate de trasaturile calculate, iar iesirile de
valorile binarizate (clase)
x_antrenare, x_testare, y_antrenare, y_testare= train_test_split(trasaturi,
dateBinarizate, test_size=0.20, random_state=0)
# Se aplica Support Vector Machine (SVM) cu o functie obiectiv liniara
clasificator_SVM = SVC(kernel='linear')
# Se incearca potrivirea datelor de antrenare
clasificator_SVM.fit(x_antrenare, y_antrenare)
# Se face predictia pe datele de testare
y_predictie = clasificator_SVM.predict(x_testare)
```

Pentru a verifica dacă evaluările participanților au fost extrase corect, acestea vor fi stocate în fișiere csv în directorul **RezultateClasificare**. Acest director va conține două subdirectoare Fericire și Tristete.

3.5. Generarea spectrogramelor semnalelor și extragerea trăsăturilor pentru acestea

Se generează câte o spectrogramă a semnalelor pentru fiecare canal, prin apelul metodei *specgram* din librăria *matplotlib.* Datele aferente canalelor sunt împărțite în segmente de lungime *Non-uniform Fast Fourier Transform* (*NFFT*) și se calculează spectrul fiecărei secțiuni. Se aplică o fereastră fiecărei secțiuni, iar numărul de puncte de suprapunere a segmentelor este dat de parametrul *noverlap*, care default are valoarea 256. Spectrograma este reprezentată ca o hartă de culori (folosind *imshow*). Se salvează în fișiere cu extensia png în directorul **Spectrograme**, având o denumire specifică, după cum poate fi observat în codul ce urmează:

```
# Metoda care genereaza spectrograma semnalului
def genereazaSpectrograma (subiect, experiment, canal):
    print("Se genereaza spectrograma pentru subiect ", subiect, ", experiment
", experiment, "canal ", canal)
    denumire_imagine = 'Spectrograme/Spectrograma_semnal_Subiect_' +
str(subiect) + '_experiment_' + str(experiment) + '_canal_' + canal
    # Se extrag datele pentru canalul, subiectul si experimentul dat ca
parametru
```

```
  dateCanal = extrageDateCanale(subiect = subiect, experiment=experiment)
[canal]
    fig = plt.figure()
    ax = plt.subplot(111)
    # Se genereaza spectrograma semnalului
    ax.specgram(dateCanal, Fs=128)
    # Se salveaza ca imagine fara a se retine axele de coordonate
    extent =
ax.get_window_extent().transformed(fig.dpi_scale_trans.inverted())
    fig.savefig(denumire_imagine+'.png', bbox_inches=extent)
    plt.close(fig)
```

S-au creat 4 funcții pentru calculul trăsăturilor aferente spectrogramelor. S-a trecut de la modelul BGR la grayscale prin apelul metodei *cvtColor* din *opencv* și s-a utilizat metoda *calcHist* din aceeași librărie, pentru determinarea histogramei în spațiul HSV, luându-se în considerare media valorilor obținute.

```
# Metoda care calculeaza histograma imaginii semnalului in spatiul HSV,
luand media pentru trasaturile calculate
def colorHistogramOfHSVImageMean(img):
    trasaturiHue = []
    trasaturiSat = []
    trasaturiVal = []
    hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
    hue_hist = cv2.calcHist([hsv], [0], None, [180], [0, 180])
    sat_hist = cv2.calcHist([hsv], [1], None, [256], [0, 256])
    val_hist = cv2.calcHist([hsv], [2], None, [256], [0, 256])
    for h in hue_hist:
        trasaturiHue.append(h[0])
    for s in sat_hist:
        trasaturiSat.append(s[0])
    for v in val_hist:
        trasaturiVal.append(v[0])
    return statistics.mean(trasaturiHue), statistics.mean(trasaturiSat),
statistics.mean(trasaturiVal)
```

Histograma gradientului orientat (*HOG*) s-a determinat apelând metoda *hog* din *skimage*, cu normalizare în a doua formă normală.

```
# Metoda care determina histograma gradientilor orientati (HOG)
def histogramOfOrientedGradients(img):
    img = cv2.resize(img, (128, 256))
    (hog, hog_image) = feature.hog(img, orientations=9,
                                   pixels_per_cell=(8, 8),
cells_per_block=(2, 2),
                                   block_norm='L2-Hys', visualize=True,
transform_sqrt=True)
    return statistics.mean(hog)
```

Valorile texturii se calculează prin apelul metodei *haralick* din modelul *mahotas.features*. Se folosește o matrice de co-apariție a nivelului de gri (*GLCM*) pătratică, cu dimensiunea

numărului de niveluri de gri N din regiunea de interes, care numără co-apariția nivelurilor de gri învecinate din imagine.

```
# Metoda care calculeaza valorile pentru textura folosind metoda Haralick
def HaralickTexture(img):

    img_resized = cv2.resize(img, (300, 300))
    img_gray = cv2.cvtColor(img_resized, cv2.COLOR_BGR2GRAY)
    image_har = mahotas.features.haralick(img_gray).mean(axis=0)
    return statistics.mean(image_har)
```

Pentru a determina descriptorii SIFT se utilizează metode din *opencv*. Se creaza un obiect pentru extragerea trăsăturilor SIFT cu *SIFT_create*, se detectează punctele cheie din spectrogramă cu *detect* și se calculează descriptorii cu *compute*.

```
# Metoda care calculeaza descriptorii SIFT
def siftDetector(img):
    descriptorValues = []
    grayImage = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    sift = cv2.SIFT_create(400)
    keypoints_sift = sift.detect(grayImage, None)
    descriptors_sift = sift.compute(grayImage, keypoints_sift)
    img = cv2.drawKeypoints(grayImage, keypoints_sift, img)
    for d in descriptors_sift[1]:
        descriptorValues.append(statistics.mean(d))
    return statistics.mean(descriptorValues)
```

3.6. Clasificarea spectrogramelor semnalelor

Etapa de clasificare a spectrogramelor este asemănătoare cu cea de clasificare a semnalelor EEG, diferența fiind că, în primul caz, se utilizează ca intrări în algoritmul SVM trăsăturile calculate pentru semnale, iar în cel de-al doilea caz, intrările sunt reprezentate de trăsăturile determinate pentru spectrograme. Ieșirile sunt reprezentate de cele două clase, 0 și 1, care marchează stările emoționale considerate. Prin apelul metodei *train_test_split* din librăria *sklearn.model.selection*, se împart datele în 80% date de antrenare și 20% date de test. S-a apelat metoda *SVC* din modelul *sklearn.svm*, cu un kernel liniar, adică hiperplanul de separare dintre cele două clase va fi reprezentat de o singură linie dreaptă. Secvența de cod următoare este relevantă în acest sens:

```
# Se determina experimentele relevante
    valenta, excitare, experimente =filtrareDupaEtichete(subiect)
    #Se calculeaza trasaturile
    for exp in experimente:
        print("Experiment "+ exp.__str__())
        tr, l=calculTrasaturiImagini(subiect, exp)
        trasaturi.append(tr)
        # Se binarizeaza valorile etichetelor
        valoarebinara = binarizeazaDate(subiect, 3.5, 5.5, exp)
        dateBinarizate.append(valoarebinara)
print("Lungime date binarizate ", len(dateBinarizate))
print('Lungime trasaturi ', len(trasaturi))
# Lungimea datelor binarizate este egala cu suma numarului de experimente
relevante pentru utilizatori
lungime  = len(dateBinarizate)
trasaturi = np.array(trasaturi).reshape(lungime, 84)
# Iesirile vor fi reprezentate de doua clase:
# 0-indica o stare negativa
# 1-indica o stare pozitiva
dateBinarizate=np.array(dateBinarizate)
x_antrenare, x_testare, y_antrenare, y_testare = train_test_split(trasaturi,
dateBinarizate, test_size=0.20, random_state=0)
# # Se aplica algoritmul SVM cu o functie liniara
clasificatorSVM = SVC(kernel='linear')
clasificatorSVM.fit(x_antrenare, y_antrenare)
y_predictii = clasificatorSVM.predict(x_testare)
```

3.7. Reprezentarea semnalelor în benzile de frecvență

A fost realizat un modul destinat afișării unor grafice ale benzilor de frecvență, grafice care sunt utile în analiza rezultatelor obținute în cadrul proiectului. Astfel, au fost create funcții pentru a arăta diferențe sau asemănări între benzile de frecvență obținute și de reprezentare a densității puterii spectrale.

```
#Metodă pentru afișarea benzilor de frecvență pentru un user
def ploteazaBenziFrecventa(subiect, experiment, canal):
     valoriBandaAlpha = impartireBenzi(subiect, experiment, 'Alpha', canal)
     valoriBandaBeta = impartireBenzi(subiect, experiment, 'Beta', canal)
     valoriBandaGamma = impartireBenzi(subiect, experiment, 'Gamma', canal)
     plt.figure(figsize=[10, 10])
     plt.title('Power bands')
     plt.subplot(5, 1, 1)
     plt.xlim([0, 100])
     plt.ylim([-100, 100])
```

```
     plt.plot(valoriBandaAlpha, "r-")
     plt.ylabel('Alpha')
     plt.subplot(5, 1, 2)
     plt.xlim([0, 175])
     plt.ylim([-100, 100])
     plt.plot(valoriBandaBeta)
     plt.ylabel('Beta')
     plt.subplot(5, 1, 3)
     plt.xlim([0, 350])
     plt.ylim([-100, 100])
     plt.plot(valoriBandaGamma, color="black")
     plt.ylabel('Gamma')
#Metodă pentru afișarea densității puterii spectrale
def ploteazaPSD(subiect, experiment, canal):
     valoriBandaAlpha = impartireBenzi(subiect, experiment, 'Alpha', canal)
     valoriBandaBeta = impartireBenzi(subiect, experiment, 'Beta', canal)
     valoriBandaGamma = impartireBenzi(subiect, experiment, 'Gamma', canal)
     freq_alpha, psd_alpha = signal.welch(valoriBandaAlpha, FREQ, nperseg =
len(valoriBandaAlpha),
    scaling='spectrum')
     freq_beta, psd_beta = signal.welch(valoriBandaBeta, FREQ, nperseg =
len(valoriBandaBeta),
    scaling='spectrum')
     freq_gamma, psd_gamma = signal.welch(valoriBandaGamma, FREQ, nperseg =
len(valoriBandaGamma),
    scaling='spectrum')
     plt.figure(figsize = (10, 8))
     plt.subplot(1, 3, 1)
     plt.plot(freq_alpha, psd_alpha)
     plt.xlabel('Frecventa [Hz]')
     plt.xlim([7.5, 13])
     plt.ylabel('Densitatea puterii spectrale')
     plt.ylim([0, 250])
     plt.title("Periodigrama Welch pentru banda Alfa")
     plt.subplot(1, 3, 2)
     plt.plot(freq_beta, psd_beta)
     plt.xlabel('Frecventa [Hz]')
     plt.xlim([13, 32])
     plt.ylabel('Densitatea puterii spectrale')
     plt.ylim([0, 50 ])
     plt.title("Periodigrama Welch pentru banda Beta")
     plt.subplot(1, 3, 3)
     plt.plot(freq_gamma, psd_gamma)
     plt.xlabel('Frecventa [Hz]')
     plt.xlim([32, 64])
     plt.ylabel('Densitatea puterii spectrale')
     plt.ylim([0, 15])
     plt.title("Periodigrama Welch pentru banda Gamma")
     plt.tight_layout()
     plt.show()
```

Capitolul 4. Rezultate experimentale

4.1. Selecția evaluărilor acordate de participanți

În urma etapei de preprocesare a semnalelor EEG prin selecția evaluărilor acordate de utilizatori, s-au obținut stimulii relevanți pentru stările de fericire sau tristețe, având drept indicatori valorile valenței și excitării. Astfel, valorile mici ale valenței și ale excitării (mai mici sau egale cu pragul inferior de 3.5) marchează o stare negativă, de tristețe. Valorile situate în intervalul (3.5, 5.5) denotă faptul că stimulii corespunzători sunt neutri din punctul de vedere al observării stării emoționale. Valorile mari (mai mari sau egale cu 5.5) marchează o stare pozitivă, de fericire. În tabelele următoare se pot observa rezultatele în urma selecției etichetelor, pentru subiecții 15 și 22:

Tabelul 4.1: Selecția etichetelor pentru participantul nr. 15

	A	B	C	D	E
1		Valenta	Excitare	Dominanta	Placere
2	2	6.74	5.83	6.37	6.83
3	3	7.09	5.67	2.53	5.65
4	4	5.51	7.9	9	8.14
5	17	6.56	5.72	5.22	7.37
6	18	2.86	2.94	3.9	2.4
7	24	2.47	3.45	2.4	2.13
8	27	2.85	3.38	3.44	6.69
9	28	3.23	2.79	3.09	2.62
10	36	6.09	5.5	6.44	5.83
11	38	6.13	5.97	8.49	6

Tabelul 4.2: Selecția etichetelor pentru participantul nr. 22

	A	B	C	D	E
1		Valenta	Excitare	Dominanta	Placere
2	0	7.71	6.56	7.71	7.74
3	1	9	9	9	9
4	3	8.21	6.97	8.27	8.42
5	4	6.94	7.01	7.05	6.01
6	6	8.94	7.81	7.99	8.99
7	13	8.01	7.01	7.85	8.01
8	16	2.86	1	3.51	1.06
9	20	1.91	1	7.38	1
10	21	2.97	1	5.06	2.03
11	31	7.97	7.04	8.28	8.06
12	33	6.72	6.35	6.79	7.91

În Tabelul 4.1 se poate observa că stimulii 2, 3, 4, 17, 36 și 38 prezintă valori mari [5.5, 9] ale valenței și excitării, de unde rezultă că participantul cu numărul 15 a avut o stare pozitivă în timpul vizionării acestor videoclipuri. Pe de altă parte, stimulii 18, 24, 27, 28 au avut un impact negativ în ceea ce privește starea emoțională a subiectului 15, dovada fiind valorile mici, situate în intervalul [1, 3.5]. În aceeași manieră se tratează și rezultatele din Tabelul 4.2: stimulii 0, 1, 3, 4, 6, 13, 31, 33 marchează o stare de fericire, iar stimulii 16, 20, 21 indică o stare de tristețe.

4.2. Analiza trăsăturilor calculate pentru semnalele EEG

Pentru a observa corectitudinea datelor trăsăturilor extrase pentru semnalele EEG, acestea au fost salvate în fișiere csv. În continuare vor fi prezentate trăsăturile calculate pentru emoția cu valență și excitare mari ([5.5, 9]) și cele extrase pentru emoția cu valență și excitare neutre ((3.5, 5.5)), în cazul celui de-al 3-lea participant. Valorile pentru trăsăturile inițiale au fost înregistrate pentru stimulul 2, ce ar trebui să exprime un sentiment de fericire, și pentru stimulul 5, neutru din punctul de vedere al observării stării emoționale.

Tabelul 4.3: Trăsăturile extrase în cazul unui stimul neutru

	A	B	C	D	E
1	Canal	PSD	SampEn	RMS	StdDev
2	Cz	6.7236	0.4412	6.0571	17.7882
3	Fz	10.7675	0.871	6.4032	10.8201
4	FC1	127.2668	0.611	21.5813	34.7049
5	FC2	35.9344	0.5871	12.0948	19.0239
6	Fp1	224.9829	0.7197	28.4356	45.1338
7	Fp2	72.0895	0.6631	15.4018	25.0283
8	F3	69.6205	0.5596	16.4144	25.9793
9	F4	11.687	0.8972	5.7641	10.1304
10	O1	5.0524	0.7469	5.1987	8.622
11	O2	478.7163	0.564	43.3138	68.5827
12	P7	9.8065	0.9333	5.3694	9.3171
13	P8	17.3434	0.7163	8.4378	13.3099
14	T7	10.5535	1.1738	3.7667	10.2488
15	T8	6.3084	0.851	4.7926	8.3909

Tabelul 4.4: Trăsăturile extrase în cazul unui stimul pozitiv

	A	B	C	D	E
1	Canal	PSD	SampEn	RMS	StdDev
2	Cz	3.1087	0.4723	11.4136	20.0056
3	Fz	15.8392	1.1012	4.5522	10.1557
4	FC1	174.1213	0.8467	16.5541	31.8383
5	FC2	47.1896	0.8231	8.8547	17.3582
6	Fp1	321.1077	0.9105	20.6746	42.6984
7	Fp2	120.3872	0.8719	11.967	25.289
8	F3	93.1229	0.7786	12.5956	23.9201
9	F4	20.094	0.9972	4.9804	10.6047
10	O1	9.128	0.9239	3.8519	7.4433
11	O2	650.0255	0.8147	32.8937	62.903
12	P8	27.0291	0.935	5.8952	12.6366
13	T7	14.539	0.994	6.0765	13.9707
14	T8	14.8933	0.9436	3.9927	8.8244

Valorile din tabelele de mai sus prezintă fluctuații mari de la un canal la altul, ceea ce este și normal în cazul datelor noastre, deoarece starea emoțională a unui participant oscilează la prezentarea unui stimul. Valorile mai mici pentru densitatea puterii spectrale, rădăcina medie pătratică și deviația standard marchează un impact emoțional mai mare. Valorile pentru entropia șablon din Tabelul 4.3 sunt semnificativ mai mici în comparație cu cele din Tabelul 4.4. O valoare mică pentru entropia șablon denotă un impact emoțional nesemnificativ.

Făcându-se o comparație între efectul unui stimul pozitiv (2) și efectul unui stimul negativ (36) asupra participantului, se poate spune că, sentimentul de tristețe este resimțit mult mai intens decât cel de fericire. În tabelele următoare se poate vedea că valorile entropiei șablon sunt mai mari, iar valorile densității puterii spectrale, ale rădăcinii medii pătratice și ale deviației standard sunt mai mici, pentru un stimul cu valori ale valenței și excitării scăzute, în comparație cu cele pentru un stimul pozitiv. Trăsăturile sunt calculate pentru subiectul 3.

Tabelul 4.5: Trăsături extrase pentru un stimul negativ

	A	B	C	D	E
1	Canal	PSD	SampEn	RMS	StdDev
2	Cz	4.9356	0.5146	7.6228	16.541
3	Fz	9.8617	1.6559	3.525	9.9106
4	FC1	98.8431	1.6017	11.0319	31.6661
5	FC2	28.606	1.5939	6.0039	17.4781
6	Fp1	149.5376	1.5995	14.1738	40.7142
7	Fp2	40.6952	1.5824	8.5879	22.9627
8	F3	57.0013	1.5675	8.1236	23.7857
9	F4	8.9388	1.6058	3.7815	9.1361
10	O1	5.6877	1.5508	2.8534	7.5417
11	O2	406.5061	1.5724	22.019	62.9594
12	P7	5.7654	1.5121	3.1953	8.3428
13	P8	10.6516	1.5534	4.0529	11.8836
14	T7	10.8194	1.4538	4.7014	9.4549
15	T8	4.88	1.5826	2.6439	7.2939

Tabelul 4.6: Trăsături extrase pentru un stimul pozitiv

	A	B	C	D	E
1	Canal	PSD	SampEn	RMS	StdDev
2	Cz	3.1087	0.4723	11.4136	20.0056
3	Fz	15.8392	1.1012	4.5522	10.1557
4	FC1	174.1213	0.8467	16.5541	31.8383
5	FC2	47.1896	0.8231	8.8547	17.3582
6	Fp1	321.1077	0.9105	20.6746	42.6984
7	Fp2	120.3872	0.8719	11.967	25.289
8	F3	93.1229	0.7786	12.5956	23.9201
9	F4	20.094	0.9972	4.9804	10.6047
10	O1	9.128	0.9239	3.8519	7.4433
11	O2	650.0255	0.8147	32.8937	62.903
12	P8	27.0291	0.935	5.8952	12.6366
13	T7	14.539	0.994	6.0765	13.9707
14	T8	14.8933	0.9436	3.9927	8.8244

O altă analiză comparativă s-a efectuat asupra emoțiilor participanților în funcție de sexul acestora. În continuare vor fi prezentate graficele benzilor de frecvență extrase pentru doi participanți de sexe diferite, având drept stimul declanșator un videoclip ce marchează o stare de fericire/extaz. Pe axa orizontală se reprezintă timpul, iar pe axa verticală valorile benzilor.

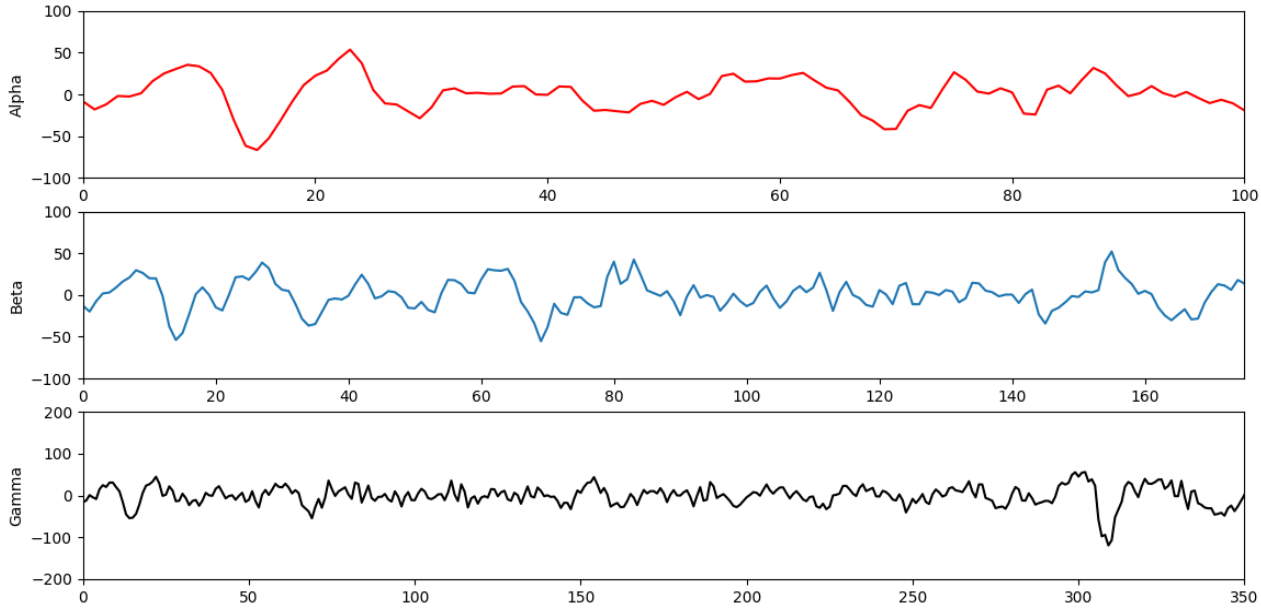

Figura 4.1: Benzile de frecvență extrase de pe canalul Fp2 pentru un participant de sex masculin

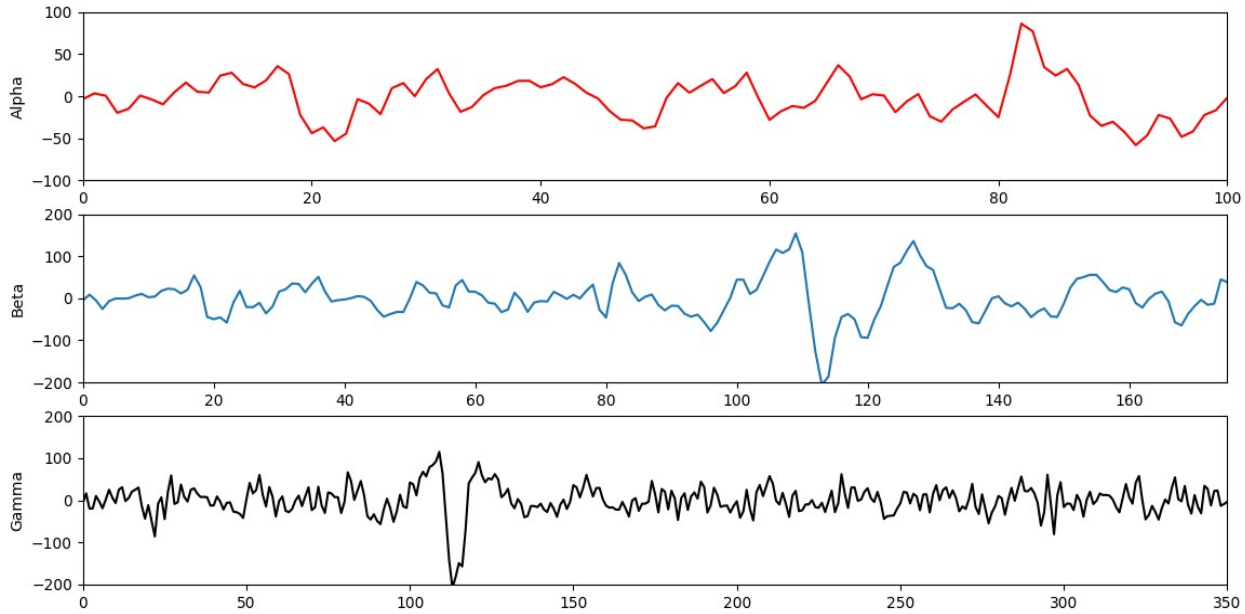

Figura 4.2: Benzile de frecvență extrase de pe canalul Fp2 pentru un participant de sex feminin

În Figura 4.1 și Figura 4.2 se pot observa diferențele clare dintre cei doi participanți în benzile beta și gamma. Participantul de sex feminin a prezentat o emoție crescută pe parcursul vizionării videoclipului, dar subiectul de sex masculin și-a păstrat o atitudine destul de constantă. Acest lucru este de așteptat în studiul reacțiilor participanților de sexe diferite, deoarece activitatea neuronală a celor două sexe este declanșată diferit la prezentarea aceluiași stimul.

În Figura 4.3 și Figura 4.4 este prezentă densitatea puterii spectrale pentru benzile alpha, beta și gamma pe canalul Fp1, având drept stimul declanșator un videoclip ce marchează o stare de fericire/extaz. Pe axa orizontală se reprezintă frecvența exprimată în [Hz], iar pe axa verticală se reprezintă densitatea puterii spectrale, măsurată în [W/Hz]. În banda gamma se poate observa că ambii participanți prezintă o fluctuație semnificativă a emoțiilor, în beta se observă o amplitudine mai mare în cazul subiectului de sex feminin, iar în banda alfa se remarcă o amplitudine mai mare în cazul subiectului masculin.

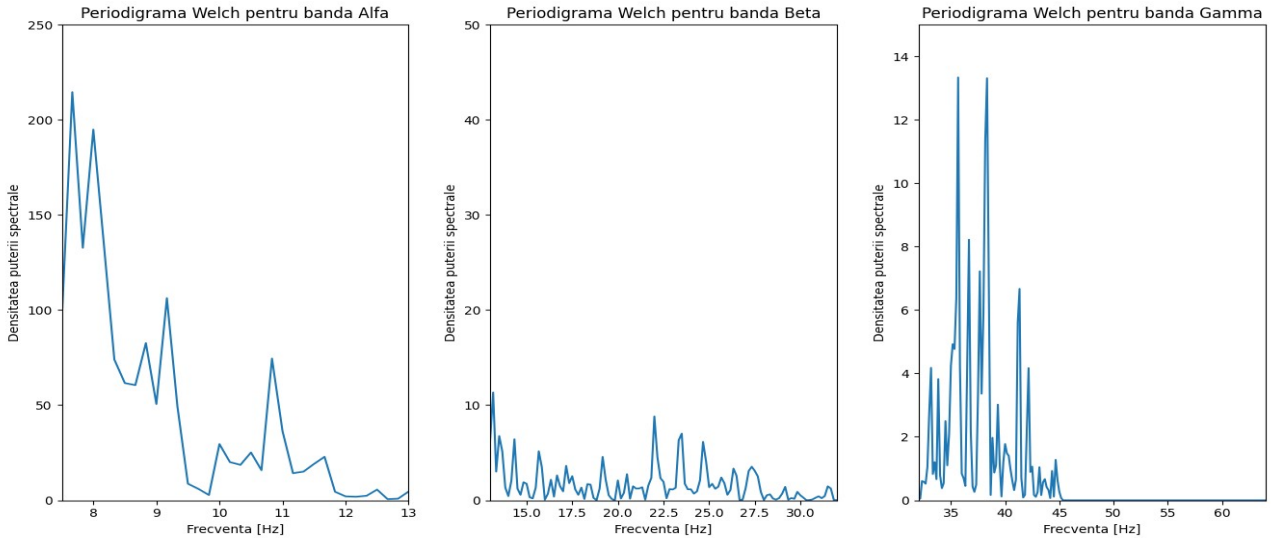

Figura 4.3: Periodigrama Welch a unui participant de sex masculin

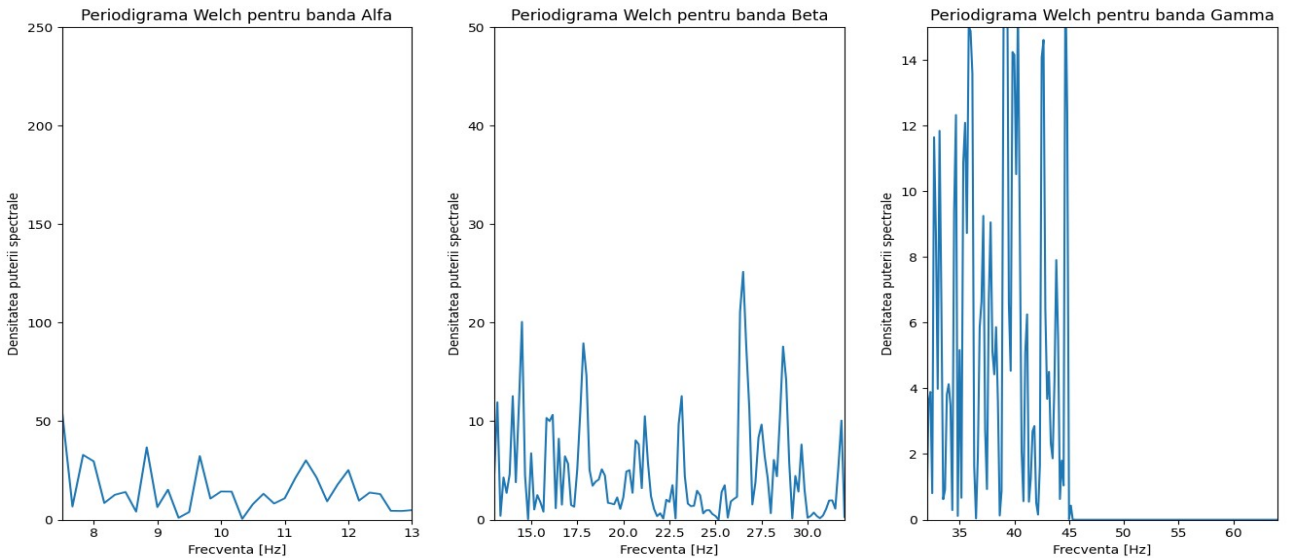

Figura 4.4: Periodigrama Welch a unui participant de sex feminin

4.3. Analiza spectrogramelor semnalelor EEG

Spectrogramele sunt utilizate pentru analiza activității cerebrale în timpul vizionării stimulilor, marcând astfel prezența unei stări emoționale. În Figura 4.5 și 4.6 sunt prezentate spectrogramele generate în cazul unui stimul cu valori ale valenței și excitării neutre, respectiv scăzute, pentru canalul F3. Pe axa orizontală este reprezentat timpul măsurat în secunde, iar pe axa verticală se reprezintă frecvența, exprimată în hertzi.

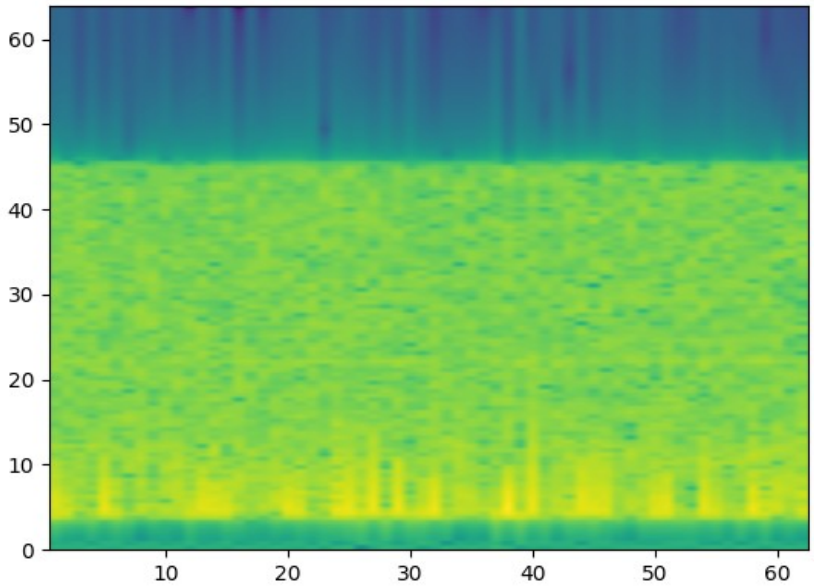

Figura 4.5: Spectrograma semnalului în cazul unui stimul nerelevant pentru observarea stării emoționale, pentru canalul F3 și un participant de gen masculin

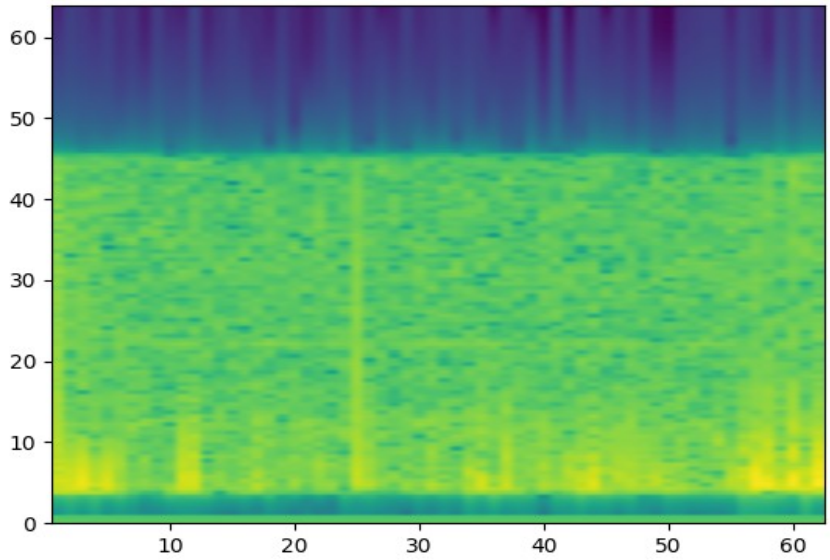

Figura 4.6 Spectrograma semnalului în cazul unui stimul cu valență și excitare negative, pentru canalul F3 și un participant de gen masculin

Se poate observa că în Figura 4.7 activitatea cerebrală pe canalul F3 este relativ constantă, stimulul neavând vreun impact emoțional ridicat asupra participantului. Pe de altă parte, în Figura 4.12 se remarcă prezența unor zone cu o frecvență mai mare a semnalului, la începutul și finalul videoclipului, care semnifică prezența stării de dezgust/tristețe a subiectului. Același stimul negativ are un impact mai redus asupra unui participant de gen feminin, după cum se poate vedea în Figura 4.7, în care semnalul rămâne oarecum constant.

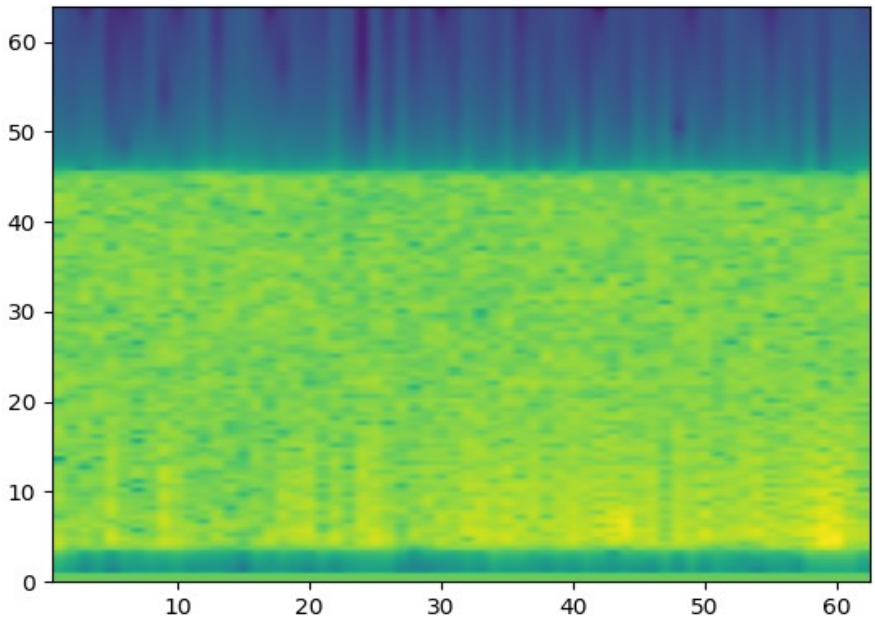

Figura 4.7: Spectrograma semnalului în cazul unui stimul cu valență și excitare negative, pentru canalul F3 și un participant de gen feminin

Tabelul 4.7: Trăsături calculate pentru un stimul pozitiv și un participant de gen feminin

	A	B	C	D	E	F	G
1	Canal	MedieHue	MedieSat	MedieVal	HOG	Textura	SIFT
2	Cz	1016.8	714.9375	714.9375	0.149000248646571	118.646256901823	25.182154
3	Fz	1016.8	714.9375	714.9375	0.136453048105132	528.367651097496	22.096191
4	FC1	1016.8	714.9375	714.9375	0.133533331205231	547.031625011069	22.950407
5	FC2	1016.8	714.9375	714.9375	0.133106400170416	513.500407577311	22.719063
6	Fp1	1016.8	714.9375	714.9375	0.134637041111795	529.911518926857	21.961824
7	Fp2	1016.8	714.9375	714.9375	0.134698944218902	572.459709953701	23.101004
8	F3	1016.8	714.9375	714.9375	0.133402957428673	494.241793120469	21.92678
9	F4	1016.8	714.9375	714.9375	0.135882143620932	637.465035708467	23.850962
10	O1	1016.8	714.9375	714.9375	0.133711310593613	576.813837647592	23.322836
11	O2	1016.8	714.9375	714.9375	0.132964603131524	520.734269884435	23.513159
12	P7	1016.8	714.9375	714.9375	0.13466855415973	557.47362337045	21.64523
13	P8	1016.8	714.9375	714.9375	0.134919842715832	529.255668338905	22.118515
14	T7	1016.8	714.9375	714.9375	0.136385426191619	715.961737957536	23.49508
15	T8	1016.8	714.9375	714.9375	0.134845111930418	587.399005574086	24.880154

În Tabelul 4.7 sunt prezentate trăsăturile calculate pe canale pentru un stimul pozitiv, vizionat de către un subiect de gen feminin. Valorile sunt asemănătoare deoarece toate canelele

marchează aceeași stare emoțională. Valorile pentru nuanță, saturație și valoare sunt identice pe toate canalele întrucât pentru generarea spectrogramelor se folosește aceeași gamă de culori cu aceleași proprietăți. Variațiile cele mai semnificative se înregistrează în cazul texturii.

Tabelul 4.8: Trăsături calculate pentru un stimul nerelvant din punctul de vedere al observării stării emoționale, pentru un participant de gen feminin

	A	B	C	D	E	F	G
1	Canal	MedieHue	MedieSat	MedieVal	HOG	Textura	SIFT
2	Cz	1016.8	714.9375	714.9375	0.149467450337225	114.404568642604	25.579546
3	Fz	1016.8	714.9375	714.9375	0.135932585829771	616.140821164061	22.907969
4	FC1	1016.8	714.9375	714.9375	0.13500625534195	654.239041475213	24.857355
5	FC2	1016.8	714.9375	714.9375	0.134858684832564	635.915236760169	25.075367
6	Fp1	1016.8	714.9375	714.9375	0.135092312818906	610.981283011212	23.490625
7	Fp2	1016.8	714.9375	714.9375	0.136562704642349	561.208683781874	23.166065
8	F3	1016.8	714.9375	714.9375	0.135163261495633	462.659922932613	22.400276
9	F4	1016.8	714.9375	714.9375	0.13693227278536	567.08222525891	22.076647
10	O1	1016.8	714.9375	714.9375	0.136171186121413	632.94314584847	24.260044
11	O2	1016.8	714.9375	714.9375	0.13568643745748	612.211885086289	24.190971
12	P7	1016.8	714.9375	714.9375	0.136282278399574	629.548485877158	23.030382
13	P8	1016.8	714.9375	714.9375	0.134261084494009	608.762015449837	23.18698
14	T7	1016.8	714.9375	714.9375	0.136941843823154	555.546522714276	20.704203
15	T8	1016.8	714.9375	714.9375	0.136474218386486	666.091057559169	22.367872

Comparând tabelele 4.7 și 4.8, se poate observa că odată cu scăderea valorilor pentru histograma gradientului orientat, a valorilor SIFT și ale texturii, impactul emoțional devine mai semnificativ. Drept urmare, rezultatele obținute în cazul unui stimul pozitiv sunt relativ mai mici în comparație cu cele pentru un stimul neutru.

4.4. Rezultatele clasificării semnalelor și spectrogramelor

Pentru a obține rezultate bune în cazul clasificării cu algoritmul Support Vector Machine, s-au folosit ca date de antrenare trăsăturile calculate pentru stimulii corespunzători emoțiilor Fericire/Tristețe și evaluările pentru toți cei 32 participanți. Astfel, setul de antrenare va conține 896 valori (2 stimuli x 14 canale x 32 participanți) pentru fiecare trăsătură: PSD, SampEn, RMS și StdDev.

Evaluările acordate de participanți pentru gradul de valență/excitare constituie valorile țintă, ce vor fi binarizate, considerând valorile mai mici sau egale cu 3.5 drept valență/excitare scăzută, iar cele mai mari sau egale cu 5.5 vor fi considerate drept valență/excitare crescută. Pentru fiecare utilizator se vor calcula 168 de trăsături (14 canale x 3 benzi x 4 trăsături). Acestea reprezintă intrările algoritmului Support Vector Machine, iar ieșirile sunt formate din valorile binarizate ale valenței și excitării pentru fiecare stimul.

Pentru a realiza o comparație corectă, algoritmul SVM a fost aplicat atât pentru clasificarea semnalelor EEG, cât și pentru clasificarea spectrogramelor semnalelor. În faza de clasificare a semnelelor EEG, a rezultat o acuratețe de 78.75%, iar în etapa de clasificare a spectrogramelor, s-a obținut o acuratețe de 81.25%. După cum se poate observa, algoritmul

SVM a determinat obținerea unui rezultat mai bun în etapa de clasificare a imaginilor semnalelor.
Motivul ar fi că, în prima etapă, datorită dimensiunilor mari, semnalele EEG au fost împărțite în subtrial-uri de câte 6 secunde și s-a utilizat doar subtrial-ul de la jumătatea semnalului pentru extragerea trăsăturilor; astfel, nu s-a realizat o clasificare a întregului semnal, ci doar a unei ferestre a semnalului. În a doua etapă, spectrogramele au fost generate pentru întreg semnalul, iar trăsăturile s-au calculat integral.

Tabelul 4.9: Rezultatele clasificării

Tipul clasificării	Algoritmul utilizat	Acuratețe	F1 score	AUC
Clasificarea semnaleleor EEG	Support Vector Machine	0.7875	0.872	0.62
Clasificarea spectrogramelor semnalelor	Support Vector Machine	0.8125	0.89	0.51
Clasificarea semnaleleor EEG	Random Forest	0.78	0.879	0.5
Clasificarea spectrogramelor semnalelor	Random Forest	0.77	0.874	0.59

În Tabelul 4.9 sunt prezentate rezultatele obținute în urma clasificării semnalelor și spectrogramelor, utilizând și un al doilea algoritm de învățare automată, Random Forest. Se poate observa că folosind algoritmul Support Vector Machine s-au obținut rezultate mai bune decât în cazul aplicării algoritmului Random Forest.

Concluzii

Lucrarea de diplomă a avut două obiective principale. Primul obiectiv s-a bazat pe analiza prin extragerea de trăsături și clasificarea semnalelor EEG, în funcție de două emoții țintă, folosind algoritmul de învățare automată Support Vector Machine. Al doilea obiectiv s-a concentrat asupra procesării și clasificării spectrogramelor semnalelor EEG, tratând aceleași stări emoționale și utilizând același algoritm de clasificare.

Rezultatele experimentale obținute pentru clasificarea binară, descrise în 4.2, prezintă valori ridicate ale acurateții, ajungând până la 78.75% în cazul clasificării semnalelor și 81.25% în cazul clasificării spectrogramelor. Aceste rezultate sunt datorate alegerii valenței și a excitației drept etichete țintă pentru trăsăturile extrase. Astfel, cele două emoții au fost clasificate în funcție de gradul scăzut/crescut de valență/ excitare, fiind încadrate în modelul 2D circumplex al lui Russell, prezentat în Figura 1.1.

S-a observat că selecția benzilor de frecvență și a trăsăturilor a jucat un rol important în stabilirea unei relații între date și emoțiile țintă. De asemenea, pe baza trăsăturilor extrase s-au putut stabili diferențele dintre participanții de sexe diferite, aceștia având răspunsuri emoționale diferite când le-a fost prezentat același tip de stimul.

Pentru extinderea direcției proiectului, se pot monitoriza mai multe semnale fiziologice periferice, precum: ritmul cardiac (Heart Rate Variability – HRV), respirația și mișcarea ochilor (Eye Tracking). Prin monitorizarea mai multor semnale fiziologice se poate crea o imagine clară a emoției unei persoane. Ritmul cardiac monitorizat prin electrocardiogramă (Electrocardiogram – ECG) oferă informații directe despre contracția mușchiului cardiac și forma sa de undă, putând fi observat un grad ridicat de excitare. Monitorizarea de tip Eye Tracking poate pune în evidență efortul cognitiv și nivelul de atenție al persoanei pe baza privirii, a clipirii și a dimensiunii pupilei.

De asemenea, pentru a obține o clasificare mai bună a datelor se poate utiliza un clasificator mai performant, precum o rețea neuronală, starea emoțională fiind determinată utilizând modelul 3D propus de Russell și Mehrabian, prezentat în Figura 1.2.

În concluzie, prelucrarea corectă a semnalelor EEG şi a spectrogramelor generate pe baza acestora, împreună cu analiza datelor rezultate prin extragerea de trăsături, pot aduce mari beneficii în contextul clasificării diferitelor stări emoționale. Evaluarea afectivă pe baza acestor parametri ar putea influența pozitiv domenii precum psihologia, psihiatria, marketing etc. prin dezvoltarea unor maşinării capabile să înțeleagă sau să anticipeze emoții. Această metodă implică totuşi un efort de calcul considerabil, mai ales în cazul procesării semnalelor, şi un grad ridicat de utilizare a resurselor memoriei, în cazul generării spectrogramelor.

Bibliografie

[1] Rafael Ramirez, Zacharis Vamvakousis, „Detecting Emotion from EEG Signals Using Emotive Epoc Device", Department of Information and Communication Technologies, vol. 7670, pp. 175- 184, 2012.

[2] Veerabathini Srinivas, „Wavelet Based Emotion Recognition Using RBF Algorithm", International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering, vol. 4, pp. 29-33, 2016.

[3] Gaurav Kumar Raghav, Kishorjit Nongmeikapam, Anurag Dixit, Sachin Bose,Dhananjay Singh, „Evaluating Classifiers for Emotion Signal on DEAP Dataset", IEEE International Conference on Machine Learning and Data Science, 2018.

[4] Emotions [Online], Disponibil la adresa:https://www.verywellmind.com/what-are-emotions-2795178, Accesat: 2021.

[5] Brain anatomy [Online], Disponibil la adresa: https://my-ms.org/anatomy_brain_part2.htm, Accesat: 2021.

[6] Approximate entropy [Online], Disponibil la adresa: https://en.wikipedia.org/wiki/Approximate_entropy, Accesat: 2021.

[7] Sample Entropy [Online], Disponibil la adresa: https://en.wikipedia.org/wiki/Sample_entropy, Accesat: 2021.

[8] Spectral density [Online], Disponibil la adresa: https://en.wikipedia.org/wiki/Spectral_density, Accesat: 2021.

[9] Root mean square [Online], Disponibil la adresa: https://en.wikipedia.org/wiki/Root_mean_square, Accesat: 2021.

[10] Spectrogram [Online], Disponibil la adresa: https://en.wikipedia.org/wiki/Spectrogram, Accesat: 2021.

[11] Linda Shapiro, "Computer Vision", The University of Washington, pp. 221-236, 2000.

[12] Navneet Dalal, Bill Triggs, "Histogram of Oriented Gradients for Human Detection".

[13] Haralick texture [Online], Disponibil la adresa: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0212110, Accesat:2021.

[14] David G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints", Computer Science Department University of British Columbia, pp.3-4, 2004.

[15] Sander Koelstra, Christian Muhl, Mohammad Soleymani, Jong-Seok Lee, Ashkan Yazdani,Touradj Ebrahimi, Thierry Pun, Anton Nijholt, Ioannis Patras, „DEAP: A Database for Emotion Analysis Using Physiological Signals", IEE Transactions on Affective Computing, vol. 3, pp. 18-31, 2011.

Anexe

Anexa 1. Codul sursă pentru aplicația „Analiza și clasificarea datelor”

- ExtrageDate.py

```
import pickle
import pandas as pd
#dictionar ce contine denumirea canalelor si indexul in data_preprocessed
dictPozitiiCanale = { 'Cz': 32, 'Fz': 31, 'FC1': 5, 'FC2': 26 , 'Fp1': 1,
'F3': 4,  'Fp2': 30, 'F4': 27,   'O2': 17,'O1': 15, 'P8': 20, 'P7': 11,
'T7': 7 , 'T8': 24 }
#Se selecteaza 16 utilizatori/subiecti
SUBIECTI = [1, 5, 6, 7, 10, 11, 12, 13, 14, 15, 22, 24, 27, 28, 29, 30]
#Canale EEG
CANALE_EEG = ['P3', 'P7', 'PO3', 'O1', 'Oz', 'Pz', 'Fp2', 'AF4', 'Fz', 'F4',
'F8', 'FC6', 'FC2',
'Cz', 'C4', 'T8', 'CP6', 'CP2', 'P4', 'P8', 'PO4', 'O2', 'Fp1', 'AF3', 'F3',
'F7', 'FC5', 'FC1', 'C3', 'T7', 'CP5', 'CP1']
#Se selecteaza 14 de canale EEG cele mai relevante pentru analiza starilor
emotionale (din cele 32)
CANALE_SELECTATE=['Cz', 'Fz', 'FC1', 'FC2', 'Fp1', 'Fp2', 'F3', 'F4', 'O1',
'O2', 'P7', 'P8', 'T7', 'T8']
numarEtichete, numarExperimente, numarUtilizatori, numarCanale,
momenteDeTimp = 4, 40, 32, 40, 8064
#Metoda care extrage valorle corespunzatoare etichetelor din
data_preprocessed  pentru un subiect si creaza dataframe-uri folosind
biblioteca pandas
def extrageDateEtichete(subiect):
    fisierDataframeEtichete = open("ProcesareDate/DataframeEtichete.dat",
'w')
    if subiect % 1 == 0:
        if subiect < 10:
            numeSubiect = '%0*d' % (2, subiect + 1)
        else:
            numeSubiect = subiect + 1
    numeFisier = "data_preprocessed/s" + str(numeSubiect) + ".dat"
    fisierDataframeEtichete.write("\nSubiect " + subiect.__str__() + "\n")
    dateCuPickle = pickle.load(open(numeFisier, 'rb'), encoding='latin1')
    etichete = dateCuPickle['labels']
    # dictionar de etichete
    dictionarDeEtichete = {'Valenta': list(etichete[:, 0]),'Excitare':
list(etichete[:, 1]), 'Dominanta': list(etichete[:, 2]), 'Placere':
list(etichete[:, 3])}
    # Se creaza un dataframe pentru etichete
    dataframeEtichete = pd.DataFrame(dictionarDeEtichete)
    fisierDataframeEtichete.write(dataframeEtichete.__str__())
    return dataframeEtichete
```

```
   #se vor alege doar valorile mari pentru valenta si arousal (>=5.5),
respectiv valorile mici (<=3.5)
#Se vor returna: vectorul cu valori pentru valenta, excitare si cela mai
relevante experimente
def filtrareDupaEtichete(subiect):
    fisierFiltrareEtichete = open('ProcesareDate/FiltrareEtichete.dat',
'a')
    valori_valenta = []
    valori_excitare = []
    experimente = []
    dfEtichete = extrageDateEtichete(subiect)
    for experiment in range(numarExperimente):
        if (dfEtichete['Valenta'][experiment] <= 3.5 and
dfEtichete['Excitare'][experiment] <= 3.5) or (dfEtichete['Valenta']
[experiment] >= 5.5 and dfEtichete['Excitare'][experiment] >= 5.5):
            # valorile etichetelor ce se incadreaza in cele 4 intervale se
vor stoca in vectorii corespunzatori, de asemenenea si indecsii
experimentelor relevante
            valori_valenta.append(dfEtichete['Valenta'][experiment])
            valori_excitare.append(dfEtichete['Excitare'][experiment])
            experimente.append(experiment)
        else:
            dfEtichete = dfEtichete.drop(labels=experiment, axis=0)
    # valorile filtrate vor fi scrise in fisier
    fisierFiltrareEtichete.write("\nSubiect " + subiect.__str__() + "\n")
    fisierFiltrareEtichete.write(dfEtichete.__str__())
    return valori_valenta, valori_excitare, experimente
# Se va realiza o binarizare a datelor pentru etichete in scopul de a putea
fi utilizate ca iesiri pentru algoritmul de clasificare
def binarizeazaDate (subiect, margineInf, margineSup, experiment):
    # Se extrag mai intai valorile etichetelor
    dfEtichete = extrageDateEtichete(subiect)
    # Daca valorile pentru valenta si excitare sunt mai mici sau egale cu
marginea inferioara (3.5), vor deveni 0
    if dfEtichete['Valenta'][experiment] <= margineInf and
dfEtichete['Excitare'][experiment] <= margineInf:
       return 0
    # Daca valorile pentru valenta si excitare sunt mai mari sau egale cu
marginea superioara (5.5), vor deveni 1
    elif dfEtichete['Valenta'][experiment] >= margineSup and
dfEtichete['Excitare'][experiment] >= margineSup:
        return 1
#Metoda care extrage datele din data_prepreocessed pentru un subiect si
returneaza dataframe-ul corespunzator canalelor
def extrageDateCanale(subiect, experiment):
    fisierDataframeCanale = open("ProcesareDate/DataframeCanale.dat", 'a')
    if subiect % 1 == 0:
        if subiect < 10:
            numeSubiect = '%0*d' % (2, subiect + 1)
        else:
            numeSubiect = subiect + 1
```

```
    fisierDataframeCanale.write("\nSubiect " + subiect.__str__() + "\n")
    numeFisier = "data_preprocessed/s" + str(numeSubiect) + ".dat"
 # dictionar pentru canale selectate
    dictionarCanale = {'Cz': None,'Fp1': None,'F3': None,'Fp2': None,
                       'F4': None, 'O2': None,'O1': None,'P8': None,
                       'P7': None,'T7': None,'T8': None,'FC1':None,
                       'FC2':None,'Fz':None}
    # Se citesc datele din fisierele .dat folosind metoda load din biblioteca
pickle
    dateCuPickle = pickle.load(open(numeFisier, 'rb'), encoding='latin1')
    dateCanale = dateCuPickle['data']
    fisierDataframeCanale.write("EXperiment "+str(experiment)+"\n")
    for canal in CANALE_SELECTATE:
        canalDict = {canal: dateCanale[experiment][dictPozitiiCanale[canal]]}
        dictionarCanale[canal] = canalDict[canal]
    # Se creaza un dataframe pentru canalele selectate
      dataframeCanale = pd.DataFrame(dictionarCanale,
    columns=['Cz', 'Fz', 'FC1', 'FC2', 'Fp1', 'Fp2', 'F3', 'F4', 'O1', 'O2',
'P7', 'P8', 'T7', 'T8'])
      fisierDataframeCanale.write(dataframeCanale.__str__())
      fisierDataframeCanale.close()
      return dataframeCanale
#Metodă pentru selectarea unui subtrial de 6s dintr-un semnal de 60s
def selecteazaSubtrialDe6sec(subiect, experiment, splits=10):
    dictionarSubtrial={ 'Cz': None,  'Fp1': None, 'F3': None,  'Fp2': None,
                       'F4': None,  'O2': None, 'O1': None,   'P8': None,
                       'P7': None,  'T7': None, 'T8': None, 'FC1':None,
'FC2':None, 'Fz':None}
    # Extragem valorile pe canale pentru subiectul si experimentul trimise ca
parametri
    dataFrameCanale = extrageDateCanale(subiect, experiment=experiment)
    fisierSubtrial=open("ProcesareDate/FisierSubtrial.dat", 'a')
    fisierSubtrial.write('\n\nSubiect '+subiect.__str__())
    for channel in CANALE_SELECTATE:
        #Se imparte semnalul in 10 intervale  a cate 6 s
        for split in range(0, splits):
            #Se ia un sub-interval de la jumatatea semnalului
            if split == 5:
                canalDict={channel: dataFrameCanale[channel][split * 768:
(split + 1) * 768]}
                dictionarSubtrial[channel]=canalDict[channel]
    #Se va crea un dataframe care sa contina acest subtrial
    df = pd.DataFrame(dictionarSubtrial, columns=['Cz', 'Fz', 'FC1', 'FC2',
'Fp1', 'Fp2', 'F3', 'F4', 'O1', 'O2', 'P7', 'P8', 'T7', 'T8'])
    fisierSubtrial.write(df.__str__())
    fisierSubtrial.close()
    return df
if __name__ == "__main__":
    df=selecteazaSubtrialDe6sec(subiect=1, experiment=1, splits=10)
    #print(df)
    #extrageDateCanale(subiect =0, experiment=20)
    # extrageDateEtichete(subiect=0)
```

- CalculTrasaturiSemnalePeBenzi.py

```
from scipy import signal
import numpy as np
import pandas as pd
import CalculTrasaturi
from ExtragereDate import CANALE_SELECTATE, selecteazaSubtrialDe6sec
subtrial = 768
valoriPSD=[]
valoriRMS=[]
valoriSampleEntropy=[]
valoriStdDev=[]
def aplicaChebyshevInvers(frecventa_low, frecventa_high, fs, order=5):
    frecventa_nyq = 0.5*fs
    low = frecventa_low / frecventa_nyq
    high = frecventa_high / frecventa_nyq
    b, a = signal.cheby2(order, 3, [low, high], btype='bandpass')
    return b, a
def aplicaFiltruChebyshevInvers(data, frecventa_low, frecventa_high, fs,
order=5):
    semnal_eeg = np.array(data)
    b, a = aplicaChebyshevInvers(frecventa_low, frecventa_high, fs,
order=order)
    semnal_filtrat = signal.lfilter(b, a, semnal_eeg)
    return semnal_filtrat
def verificaTipBanda(tipBanda):
    fs = 128
    if tipBanda == "Alpha":
        frecventa_low = 7.5
        frecventa_high = 12.5
    elif tipBanda == "Beta":
        frecventa_low = 13.0
        frecventa_high = 30.0
    else:
        frecventa_low = 30.0
        frecventa_high = 63.5
    return frecventa_low, frecventa_high, fs

def impartireBenziFrecventa(subiect, experiment):
    print("Impartire pe benzi de frecventa si calcul de trasaturi pentru
subiectul  "+str(subiect))
    valoriTrasaturi = []
    fisierTrasaturiCSV = open('CalculTrasaturi/Trasaturi_semnale_EEG.csv',
'a')
    fisierTrasaturiCSV.write("\n")
```

```
    for tipBanda in ['Alpha', 'Beta', 'Gamma']:
        print(tipBanda)
        # Se verifica tipul benzii si se returneaza valorile low, high si fs
ale frecventei
        frecventa_low, frecventa_high, fs = verificaTipBanda(tipBanda)
        #Se scrie in fisierele corespunzatoare benzilor
        fisierValoriBenziPerUser =
open('ImpartirePeBenziDeFrecventa/ValoriBanda'+tipBanda+'.dat', 'a')
        fisierValoriBenziPerUser.write("\n Subiect " + subiect.__str__()+"\n")

        # Se selecteaza un subtrial de 6 secunde
        subtrial = selecteazaSubtrialDe6sec(subiect=subiect,
experiment=experiment, splits=10)
        for canal in CANALE_SELECTATE:
            valoriBanda = []
            column = subtrial[canal]
            dateFiltrate = aplicaFiltruChebyshevInvers(column, frecventa_low,
frecventa_high, fs)
            for val in dateFiltrate:
                  valoriBanda.append(round(val, 4))
            # Se scrie in fisierele corespunzatoare benzilor
            fisierValoriBenziPerUser.write("\nCanal "+canal+"\n")
            fisierValoriBenziPerUser.write(valoriBanda.__str__() + "\n")
             # Se calculeaza cele 4 trasaturi
            trasaturi =
CalculTrasaturi.calculeazaTrasaturiBanda(valoare=valoriBanda,
low=frecventa_low, high=frecventa_high)
             #psd
            valoriTrasaturi.append(trasaturi[0])
            #entropie
            valoriTrasaturi.append(trasaturi[1])
            #rms
            valoriTrasaturi.append(trasaturi[2])
            #deviatie
            valoriTrasaturi.append(trasaturi[3])
            #Se scriu datele in fisier .csv
            fisierTrasaturiCSV.write(str(subiect)+"  "+str(experiment)+"
"+tipBanda+"  "+canal + " " + str(trasaturi[0]) + " " + str(trasaturi[1]) + "
" + str(trasaturi[2]) + " " + str(trasaturi[3])+"\n")

      fisierValoriBenziPerUser.close()
      return valoriTrasaturi
if __name__ == "__main__":
    trasaturi = impartireBenziFrecventa(subiect=3, experiment=2)
    print("S-au calculat "+len(trasaturi).__str__()+" trasaturi")
    print(trasaturi)
```

- CalculTrasaturiSemnale.py

```
from scipy import signal
import numpy as np
#Metoda care calculeaza deviatia standard
def calculeazaDeviatiaStandard(date):
    semnal = np.asarray(date)
    max = np.max(np.std(semnal))
    return max
#Metoda care calculeaza radacina medie patratica
def calculeazaRMS(date):
     sampling_rate = 128
     frecventa, psd = signal.welch(date, sampling_rate,
scaling='spectrum')
     amplitudineRMS = np.sqrt(psd.max())
     return amplitudineRMS
#Metoda care calculeaza densitatea puterii spectrale
def calculeazaDensitateaPuteriiSpectrale(banda, low, high):
    low = 4*low
    high = 4*high
    frecventa, psd = signal.welch(banda,  nperseg=len(banda),
scaling='spectrum')
    #Se preiau datele din domeniul de frecventa pentru fiecare banda
    psd = psd[int(low):int(high)]
    return np.max(psd)
#Metodă pentru calcularea entropiei sablon
def calculeazaEntropieSablon(date, m, r = None):
    def determinaDistantaMaxima(xi, xj):
        return max([abs(ua - va) for ua, va in zip(xi, xj)])
    def determinaPhi(m):
        x = [[date[j] for j in range(i, i + m - 1 + 1)] for i in range(N -
m + 1)]
        C = [len([1 for j in range(len(x)) if i != j and
determinaDistantaMaxima(x[i], x[j]) <= r])for i in range(len(x))]
        return sum(C)
    N = len(date)
    return -np.log(determinaPhi(m+1) / determinaPhi(m))
#Se calculeaza PSD, RMS, entropia sablon si deviatia standard pentru
fiecare banda (ALPHA, BETA si GAMMA)
def calculeazaTrasaturiBanda(valoare, low, high):
    psd = calculeazaDensitateaPuteriiSpectrale(valoare, low, high)
    r = 0.2 * np.std(valoare)
    entropieSablon = calculeazaEntropieSablon(valoare, 2, r)
    rms = calculeazaRMS(valoare)
    deviatieStandard = calculeazaDeviatiaStandard(valoare)
    return [round(psd, 4), round(entropieSablon, 4), round(rms, 4),
round(deviatieStandard, 4)]
```

- ClasificareSemnaleEEG.py

```
import numpy as np
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from CalculTrasaturiPeBenzi import impartireBenziFrecventa
from  ExtragereDate import filtrareDupaEtichete, binarizeazaDate, SUBIECTI,
extrageDateEtichete
from sklearn.metrics import accuracy_score

def clasificare_semnale_EEG():
    trasaturi = []
    dateBinarizate = []
    fisierFericireEtichete =
open('RezultateClasificare/Fericire/Etichete.csv', 'a')
    fisierFericireEtichete.write("\n")
    fisierTristeteEtichete =
open('RezultateClasificare/Tristete/Etichete.csv', 'a')
fisierTristeteEtichete.write("\n")
    for subiect in range(1, 32):
        # Se realizeaza o filtrare a etichetelor functie de marginea
inferioara (3.5) si cea superioara (5.5)
        valenta, excitare, experimente =filtrareDupaEtichete(subiect)
        valoriEtichete = extrageDateEtichete(subiect)
      #Se calculeaza trasaturile
      for exp in experimente:
        print("Experiment "+ exp.__str__())
        tr = impartireBenziFrecventa(subiect=subiect,experiment=exp)
        trasaturi.append(tr)
        # Se binarizeaza datele pentru cele 2 etichete: valenta si excitare
pentru experimentul exp
        valoarebinara = binarizeazaDate(subiect, 3.5, 5.5, exp)
        dateBinarizate.append(valoarebinara)

        #Se scriu etichetele in folderele emotiilor
        if (valoriEtichete['Valenta'][exp] <= 3.5 and
valoriEtichete['Excitare'][exp] <= 3.5) :
            fisierTristeteEtichete.write(str(subiect)+" "+str(exp)+"
"+str(valoriEtichete['Valenta'][exp])+" "+str(valoriEtichete['Valenta']
[exp])+"\n")
        elif (valoriEtichete['Valenta'][exp] >=5.5 and
valoriEtichete['Excitare'][exp] >= 5.5):
            fisierFericireEtichete.write(str(subiect)+" "+str(exp)+"
"+str(valoriEtichete['Valenta'][exp])+" "+str(valoriEtichete['Valenta']
[exp])+"\n")
```

```
# Lungimea datelor binarizate este egala cu suma numarului de experimente
relevante pentru utilizatori
lungime  = len(dateBinarizate)
# Pentru fiecare experiment la care este supus utilizatorul se calculeaza 168
de trasaturi (4 trasaturi X 3 benzi X 14 canale selectate)
trasaturi = np.array(trasaturi).reshape(lungime, 168)
print(trasaturi)
print(dateBinarizate)
# Iesirile vor fi reprezentate de doua clase:
#0-indica o stare negativa
#1-indica o stare pozitiva
dateBinarizate=np.array(dateBinarizate)
# Se impart datele: 80% pentru antrenare si 20% pentru testare
# intrarile sunt reprezentate de trasaturile calculate, iar iesirile de
valorile binarizate (clase)
x_antrenare, x_testare, y_antrenare, y_testare = train_test_split(trasaturi,
dateBinarizate, test_size=0.20,

random_state=0)
# # Se aplica Support Vector Machine (SVM) cu o functie obiectiv liniara
# clasificator_SVM = SVC(kernel='linear')
# # Se incearca potrivirea datelor de antrenare
# clasificator_SVM.fit(x_antrenare, y_antrenare)
# # Se face predictia pe datele de testare
# y_predictie = clasificator_SVM.predict(x_testare)
# # Se afiseaza matricea de confuzie
# print(confusion_matrix(y_testare, y_predictie))

# Se aplica Random Forest
clasificatorRandomForest = RandomForestClassifier(n_estimators=8,
random_state=0,
                                                  max_features='sqrt',
criterion='entropy')
# Antrenarea datelor
clasificatorRandomForest.fit(x_antrenare, y_antrenare)
# Predicțiile rezultate
y_predictii = clasificatorRandomForest.predict(x_testare)
# Se afiseaza metricile
print("Folosind Random Forest, s-a obtinut o  acuratete  de ",
accuracy_score(y_testare, y_predictii), ".")
print("F1 score: ", f1_score(y_testare, y_predictii))
false_positive_rate, true_positive_rate, thresholds = roc_curve(y_testare,
y_predictii)
roc_auc = auc(false_positive_rate, true_positive_rate)
print("Area under curve: ", roc_auc)
if __name__ == '__main__':
    clasificare_semnale_EEG()
```

- PloteazaBenziFrecventa.py

```
import matplotlib.pyplot as plt
from scipy import signal
from CalculTrasaturiPeBenzi import impartireBenzi
FREQ = 128
#Metodă pentru afișarea benzilor de frecvență pentru un user
def ploteazaBenziFrecventa(subiect, experiment, canal):
     valoriBandaAlpha = impartireBenzi(subiect, experiment, 'Alpha', canal)
     valoriBandaBeta = impartireBenzi(subiect, experiment, 'Beta', canal)
     valoriBandaGamma = impartireBenzi(subiect, experiment, 'Gamma', canal)
     plt.figure(figsize=[15, 15])
     plt.title('Benzi de frecventa')
     plt.subplot(3, 1, 1)
     plt.xlim([0, 100])
     plt.ylim([-100, 100])
     plt.plot(valoriBandaAlpha, "r-")
     plt.ylabel('Alpha')
     plt.subplot(3, 1, 2)
     plt.xlim([0, 175])
     plt.ylim([-200, 200])
     plt.plot(valoriBandaBeta)
     plt.ylabel('Beta')
     plt.subplot(3, 1, 3)
     plt.xlim([0, 350])
     plt.ylim([-200, 200])
     plt.plot(valoriBandaGamma, color="black")
     plt.ylabel('Gamma')
     plt.show()
#Metodă pentru afișarea densității puterii spectrale
def ploteazaPSD(subiect, experiment, canal):
     valoriBandaAlpha = impartireBenzi(subiect, experiment, 'Alpha', canal)
     valoriBandaBeta = impartireBenzi(subiect, experiment, 'Beta', canal)
     valoriBandaGamma = impartireBenzi(subiect, experiment, 'Gamma', canal)
     freq_alpha, psd_alpha = signal.welch(valoriBandaAlpha, FREQ, nperseg =
len(valoriBandaAlpha),
    scaling='spectrum')
     freq_beta, psd_beta = signal.welch(valoriBandaBeta, FREQ, nperseg =
len(valoriBandaBeta),
    scaling='spectrum')
     freq_gamma, psd_gamma = signal.welch(valoriBandaGamma, FREQ, nperseg =
len(valoriBandaGamma),
    scaling='spectrum')
     plt.figure(figsize = (10, 8))
     plt.subplot(1, 3, 1)
     plt.plot(freq_alpha, psd_alpha)
     plt.xlabel('Frecventa [Hz]')
     plt.xlim([7.5, 13])
     plt.ylabel('Densitatea puterii spectrale')
```

```
    plt.ylim([0, 250])
    plt.title("Periodigrama Welch pentru banda Alfa")
    plt.subplot(1, 3, 2)
    plt.plot(freq_beta, psd_beta)
    plt.xlabel('Frecventa [Hz]')
    plt.xlim([13, 32])
    plt.ylabel('Densitatea puterii spectrale')
    plt.ylim([0, 50 ])
    plt.title("Periodigrama Welch pentru banda Beta")
    plt.subplot(1, 3, 3)
    plt.plot(freq_gamma, psd_gamma)
    plt.xlabel('Frecventa [Hz]')
    plt.xlim([32, 64])
    plt.ylabel('Densitatea puterii spectrale[W/Hz]')
    plt.ylim([0, 15])
    plt.title("Periodigrama Welch pentru banda Gamma")
    plt.tight_layout()
    plt.show()
# ploteazaBenziFrecventa(1, 3, 'Fp1')
ploteazaPSD(10, 2, 'Fp1')
```

- CalculTrasaturiSpectrograme.py

```
from ExtragereDate import *
import matplotlib.pyplot as plt
import cv2
import matplotlib
from skimage import feature
import mahotas
import statistics
matplotlib.use('TKAgg', force=True)
# frecventa de esantionare
frecventaEsantionare =128
def calculTrasaturiImagini(subiect, experiment):
    trasaturiImagini = []
    fisierTrasaturiCSV = open('Spectrograme/trasaturi_spectrograme.csv',
'a')
    fisierTrasaturiCSV.write("\n")

    for canal in CANALE_SELECTATE:
        denumire_imagine
='Spectrograme/Spectrograma_semnal_Subiect_'+str(subiect)
+'_experiment_'+str(experiment)+'_canal_'+canal
        # Se genereala spectrograma pentru subiect, experiment, canal
        genereazaSpectrograma(subiect, experiment, canal)
        # Se citeste spectrograma
        img = cv2.imread(denumire_imagine+'.png')
        print('Se calculeaza HSV Histogram')
        # trasaturiHue, trasaturiSat, trasaturiVal =
colorHistogramOfHSVImageMax(img)
        trasaturiHue, trasaturiSat, trasaturiVal =
colorHistogramOfHSVImageMean(img)
        trasaturiImagini.append(trasaturiHue)
        trasaturiImagini.append(trasaturiSat)
        trasaturiImagini.append(trasaturiVal)
        print('Se calculeaza trasaturi HOG')
        trasaturiHOG = histogramOfOrientedGradients(img)
        trasaturiImagini.append(trasaturiHOG)
        print('Se calculeaza HARALICK texture')
        trasaturiTexturaHaralick = HaralickTexture(img)
        trasaturiImagini.append(trasaturiTexturaHaralick)
        print("Se calculeaza trasaturi detector sift")
        trasaturiSIFT = siftDetector(img)
        trasaturiImagini.append(trasaturiSIFT)
        fisierTrasaturiCSV.write(str(subiect)+"  "+str(experiment)+"
"+canal+"  "+str(trasaturiHue)+"  "+str(trasaturiSat)+"
"+str(trasaturiVal)+"  "+str(trasaturiHOG)+"
"+str(trasaturiTexturaHaralick)+"  "+str(trasaturiSIFT)+"\n")
    fisierTrasaturiCSV.close()
    return trasaturiImagini, len(trasaturiImagini)
```

```
# Metoda care genereaza spectrograma semnalului
def genereazaSpectrograma (subiect, experiment, canal):
    print("Se genereaza spectrograma pentru subiect ", subiect, ",
experiment ", experiment, "canal ", canal)
    denumire_imagine = 'Spectrograme/Spectrograma_semnal_Subiect_' +
str(subiect) + '_experiment_' + str(experiment) + '_canal_' + canal
    # Se extrag datele pentru canalul, subiectul si experimentul dat ca
parametru
    dateCanal = extrageDateCanale(subiect = subiect, experiment=experiment)
[canal]
    fig = plt.figure()
    ax = plt.subplot(111)
    # Se genereaza spectrograma semnalului
    ax.specgram(dateCanal, Fs=128)
    # Se salveaza ca imagine fara a se retine axele de coordonate
    extent =
ax.get_window_extent().transformed(fig.dpi_scale_trans.inverted())
    fig.savefig(denumire_imagine+'.png', bbox_inches=extent)
    # plt.show()
    plt.close(fig)

# Metoda care calculeaza histograma imaginii semnalului in spatiul HSV,
luand media pentru trasaturile calculate
def colorHistogramOfHSVImageMean(img):
    trasaturiHue = []
    trasaturiSat = []
    trasaturiVal = []
    hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
    hue_hist = cv2.calcHist([hsv], [0], None, [180], [0, 180])
    sat_hist = cv2.calcHist([hsv], [1], None, [256], [0, 256])
    val_hist = cv2.calcHist([hsv], [2], None, [256], [0, 256])
    for h in hue_hist:
        trasaturiHue.append(h[0])
    for s in sat_hist:
        trasaturiSat.append(s[0])
    for v in val_hist:
        trasaturiVal.append(v[0])
    print("Trasaturi HSV ", statistics.mean(trasaturiHue), " ",
statistics.mean(trasaturiSat)," ", statistics.mean(trasaturiVal))
    return statistics.mean(trasaturiHue), statistics.mean(trasaturiSat),
statistics.mean(trasaturiVal)
# Metoda care calculeaza histograma imaginii semnalului in spatiul HSV,
luand valoarea maxima a trasaturilor calculate
def colorHistogramOfHSVImageMax(img):
    trasaturiHue = []
    trasaturiSat = []
    trasaturiVal = []
    hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
    hue_hist = cv2.calcHist([hsv], [0], None, [180], [0, 180])
    sat_hist = cv2.calcHist([hsv], [1], None, [256], [0, 256])
    val_hist = cv2.calcHist([hsv], [2], None, [256], [0, 256])
    for h in hue_hist:
        trasaturiHue.append(h[0])
```

```
    for s in sat_hist:
        trasaturiSat.append(s[0])
    for v in val_hist:
        trasaturiVal.append(v[0])
    return max(trasaturiHue), max(trasaturiSat), max(trasaturiVal)
# Metoda care determina histograma gradientilor orientati (HOG)
def histogramOfOrientedGradients(img):
    img = cv2.resize(img, (128, 256))
    (hog, hog_image) = feature.hog(img, orientations=9,
                                   pixels_per_cell=(8, 8),
cells_per_block=(2, 2),
                                   block_norm='L2-Hys', visualize=True,
transform_sqrt=True)
    print("HOG ", statistics.mean(hog))
    return statistics.mean(hog)
# Metoda care calculeaza valorile pentru textura folosind metoda Haralick
def HaralickTexture(img):
    img_resized = cv2.resize(img, (300, 300))
    img_gray = cv2.cvtColor(img_resized, cv2.COLOR_BGR2GRAY)
    image_har = mahotas.features.haralick(img_gray).mean(axis=0)
    print("Haralick texture ", statistics.mean(image_har))
    return statistics.mean(image_har)
# Metoda care calculeaza descriptorii SIFT
def siftDetector(img):
    descriptorValues = []
    grayImage = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    sift = cv2.SIFT_create(400)
    keypoints_sift = sift.detect(grayImage, None)
    descriptors_sift = sift.compute(grayImage, keypoints_sift)
    img = cv2.drawKeypoints(grayImage, keypoints_sift, img)
    for d in descriptors_sift[1]:
        descriptorValues.append(statistics.mean(d))
    print("SIFT ", statistics.mean(descriptorValues))
    return statistics.mean(descriptorValues)
if __name__ == '__main__':
     # genereazaSpectrograma(subiect=3, experiment=36, canal='F3')
     calculTrasaturiImagini(subiect=2, experiment=24)
```

- ClasificareSpectrograme.py

```
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from  ExtragereDate import filtrareDupaEtichete, binarizeazaDate, SUBIECTI
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import roc_curve, auc, accuracy_score, f1_score
from CalculTrasaturiImagini import calculTrasaturiImagini
def clasificare_spectrograme():
    trasaturi = []
    dateBinarizate = []
    for subiect in range(1, 32):
        # Se realizeaza o filtrare a etichetelor functie de marginea
inferioara (3.5) si cea superioara (5.5)
        # Se determina experimentele relevante
        valenta, excitare, experimente =filtrareDupaEtichete(subiect)
        #Se calculeaza trasaturile
        for exp in experimente:
            print("Experiment "+ exp.__str__())
            tr, l=calculTrasaturiImagini(subiect, exp)
            trasaturi.append(tr)
            # Se binarizeaza valorile etichetelor
            valoarebinara = binarizeazaDate(subiect, 3.5, 5.5, exp)
            dateBinarizate.append(valoarebinara)
    print("Lungime date binarizate ", len(dateBinarizate))
    print('Lungime trasaturi ', len(trasaturi))
     # Lungimea datelor binarizate este egala cu suma numarului de
experimente relevante pentru utilizatori
    lungime  = len(dateBinarizate)
    trasaturi = np.array(trasaturi).reshape(lungime, 84)
    # Iesirile vor fi reprezentate de doua clase:
    # 0-indica o stare negativa
    # 1-indica o stare pozitiva
    dateBinarizate=np.array(dateBinarizate)
    x_antrenare, x_testare, y_antrenare, y_testare =
train_test_split(trasaturi, dateBinarizate, test_size=0.20, random_state=0)

    # Se aplica algoritmul SVM cu o functie liniara
    clasificatorSVM = SVC(kernel='linear')
    clasificatorSVM.fit(x_antrenare, y_antrenare)
    y_predictii = clasificatorSVM.predict(x_testare)
    # Se afiseaza metricile
    print("Folosind SVM, s-a obtinut o  acuratete  de ",
accuracy_score(y_testare, y_predictii) , ".")
    print("F1 score: ", f1_score(y_testare, y_predictii))
    false_positive_rate, true_positive_rate, thresholds =
roc_curve(y_testare, y_predictii)
    roc_auc = auc(false_positive_rate, true_positive_rate)
    print("Area under curve: ", roc_auc)
```

```
    # # Se aplica Random Forest
    # clasificatorRandomForest = RandomForestClassifier(n_estimators=8,
random_state=0,
    #                                     max_features='sqrt',
criterion='entropy')
    # # Antrenarea datelor
    # clasificatorRandomForest.fit(x_antrenare, y_antrenare)
    # # Predicțiile rezultate
    # y_predictii = clasificatorRandomForest.predict(x_testare)
    # # Metricile de acurateță
    # print("Folosind Random Forest s-a obtinut o acuratete de : ",
accuracy_score(y_testare, y_predictii), ".")
    # print("F1 score: ", f1_score(y_testare, y_predictii))
    # false_positive_rate, true_positive_rate, thresholds =
roc_curve(y_testare, y_predictii)
    # roc_auc = auc(false_positive_rate, true_positive_rate)
    # print("Area under curve: ", roc_auc)
if __name__ == '__main__':
    clasificare_spectrograme()
```