Clasificator de genuri muzicale Naiv Bayes

Niță Alexandros, Pascaru Dan-Alexandru, Glod Cosmin Ștefan

November 2024

Cuprins

Scopul acestui raport este de a documenta și de a explica modelul matematic folosit pentru un proiect de clasificare a pieselor în funcție de genul lor muzical.

Examplu de utilizare: navigați în directorul unde ați salvat proiectul în terminal și rulați `python main.py`.

1 Set de date

Am utilizat unul dintre cele mai populare seturi de date pentru antrenarea clasificatoarelor de sunet, și anume GTZAN Dataset. Acest set conține un total de 999 de piese muzicale, împărțite în 10 genuri diferite: blues, classical, country, disco, hip hop, jazz, metal, pop, reggae și rock. Fiecare piesă este salvată în format .wav și are următoarele caracteristici: 22050Hz, Mono, 16-bit.

În general, fiecare gen include câte 100 de piese, cu o singură excepție: în categoria jazz, piesa cu numărul 00054 lipsește, ceea ce înseamnă că acest gen are doar 99 de piese. Aceasta a fost o caracteristică importantă de luat în considerare atunci când am atribuit indici setului de date. Lipsa unei piese în genul jazz a necesitat o ajustare a indicilor pentru a menține coerența în structura setului de date și pentru a evita erorile în timpul procesării și antrenării clasificatorului.

1.1 Extracția Caracteristicilor Audio

Setul de date include majoritatea caracteristicilor audio din fișierul CSV original, însă au fost adăugate și caracteristici suplimentare, precum Amplitude Envelope și Band-Energy Ratio, pentru a oferi mai multe informații.

Algoritmul de extragere a caracteristicilor durează în jur de 20 de minute, dar nu este optimizat. Funcțiile din Librosa repetă anumiți pași, cum ar fi transformarea Fourier pe termen scurt (STFT), pentru fiecare caracteristică de frecvență, ceea ce încetinește procesul.

Caracteristicile audio sunt calculate pe baza scanării pistei audio cu o fereastră mobilă, pentru a observa variațiile în timp. Fișierul .npy rezultat conține media și variația valorilor generate prin această metodă.

1.1.1 Scara cromatică

Scara cromatcă plasează fiecare ton muzical în 12 clase: de la nota muzicală Do la nota muzicală Si considerând semitonurile. Fiecare ton este calculat în funcție de energia pe care o are i. e. de cât de tare se aude.

1.1.2 Root Mean Square (RMS)

RMS este o metodă de a măsura amplitudinea medie a unui val sonor. Se calculează ca rădăcina pătrată a mediei aritmetice a valorilor semnalului la pătrat, luând în calcul toate eșantioanele dintr-un cadru. Acesta este mai puțin sensibil la valori extreme (outliers) decât Amplitude Envelope, oferind o valoare stabilă care reflectă intensitatea generală a sunetului.

1.1.3 Spectral Centroid & Bandwidth

Centrul spectral este calculat ca media ponderată a frecvențelor prezente în spectru, unde ponderile sunt date de magnitudinea fiecărei frecvențe într-un cadru. Cu cât valoarea este mai mare, cu atât sunetul este mai strălucitor, iar cu cât este mai mică, cu atât sunetul este mai închis. Banda spectrală măsoară gama de frecvențe pe care se distribuie energia semnalului: o valoare scăzută indică o concentrare într-o gamă îngustă de frecvențe, iar o valoare mare sugerează o distribuție mai largă, fără a reprezenta variația ca în cazul centrului spectral.

1.2 Spectral Rolloff

Spectral Rolloff indică frecvența sub care se află un anumit procent din energia totală a spectrului. În codul implementat, se calculează frecvența la care se găsește 85% din energia spectrală.

1.2.1 Zero crossing rate

Zero Crossing Rate reprezintă numărul de ori în care semnalul trece prin axa de amplitudine zero într-un anumit interval de timp. Cu alte cuvinte, arată cât de des oscilează sunetul, fiind util pentru a distinge sunetele rapide de cele mai constante.

1.2.2 Componentele Armonice și Percusive

Sunetul armonic este cel care creează melodii și acorduri (ceea ce auzim ca ton muzical). Este realizarea acustică a unei unde sinusoidale și apare ca o linie orizontală într-o spectrogramă. În schimb, sunetul percutiv este perceput ca un zgomot sau o lovitură (cum ar fi sunetul de tobă), apărând ca o linie verticală în spectrogramă.

1.2.3 Tempo

Tempo-ul este măsurat în bătăi pe minut (BPM) și indică viteza sau ritmul piesei. Practic, oferă o idee generală despre dinamica melodiei. În Librosa, calculul BPM este o estimare, deci poate fi mai puțin precis pentru piese cu ritmuri complexe sau schimbări de tempo.

1.2.4 Coeficienții Cepstrali pe Scară Mel (MFCC)

MFCC-urile imită modul în care urechea umană percepe frecvențele. Fiecare coeficient reprezintă o parte din spectrul sonor, cu accent pe diferențele dintre frecvențele joase și înalte. Algoritmul calculează primii 20 de coeficienți pentru a surprinde diverse detalii despre sunet.

1.2.5 Amplitude Envelope

Amplitude Envelope este o caracteristică adăugată care arată amplitudinea maximă absolută într-un cadru audio (calculată pentru fiecare cadru). Este destul de sensibilă la valori extreme (outliers) și oferă o estimare aproximativă a intensității sunetului

1.2.6 Band Energy Ratio

Band Energy Ratio este o caracteristică suplimentară, absentă în setul nostru de date. Cu o frecvență de prag setată la 2000 Hz (în algoritm), această caracteristică arată raportul dintre puterea benzilor de frecvență joasă și puterea benzilor de frecvență înaltă pentru fiecare cadru. Este utilă pentru a înțelege echilibrul de energie între frecvențele joase și cele înalte dintr-o piesă.

1.2.7 Short-Time Fourier Transform

STFT este utilizat pentru a calcula spectrogramele fișierelor audio. Această tehnică împarte semnalul audio în segmente mici (numite ferestre) și aplică transformarea Fourier pe fiecare segment pentru a obține spectrul de frecvență în timp. Transformarea Fourier permite descompunerea semnalului în frecvențele sale componente, arătând amplitudinile respective. Fiindcă semnalele audio variază în timp, STFT oferă o reprezentare a acestor schimbări prin spectrograma rezultată, ideală pentru analiza sunetului variabil cum este muzica.

2 Principal Component Analysis (PCA)

Scopul acestui algoritm este de a reduce din dimensionalitatea datelor extrase. Am măsurat 61 de caracteristici pentru 999 de piese, care sunt salvate ca o matrice cu 999 de linii și 61 de coloane. Inițial vom calcula matricea de covarianță pentru aceste date, după ce le centrăm. Pentru a obține cât mai multă informație din aceste date, vom dori să schimbăm sistemul de componente folosit calculând componente noi. Deoarece când înmulțim repetat un vector aleator cu matricea de covarianță acesta tinde spre o componenetă ce obține variantă maximă, ne putem îndrepta atenția către vectorii proprii ai matricei de covarianță deoarece la înmulțirea acestora cu matricea de covarianță, aceștia nu își vor schimba direcția. În plus, variantă pe care o introduce fiecare vector propriu va fi chiar valoare lui proprie. Deoarece datele colectate au o dimensionalitate mare, vom păstra doar primii K vectori proprii asociați primelor K valori proprii în ordine descrescătoare. K este primul indice care respectă proprietatea:

$$\frac{\sum_{i=1}^{K} \lambda_i}{\sum_{j=1}^{N} \lambda j} \geq p$$

unde N este dimensiunea matricei de covarianță (adică numărul de vectori proprii), λ_i este a i-a valoare proprie în ordine descrescătoare și p este un coeficient ce reprezintă proporția din variantă pe care o dorim ($p \in [0, 1]$). Din testele noastre, obținem acuratețea maximă când $p = 0.8$. În final, vom proiecta datele noastre pe noile componente pe care le-am obținut.

3 Clasificare Bayes

Vom calcula probabilitatea ca o anumită piesă (dată de proiecția pe noile componenete ale vectorului de caracteristici pe care l-am extras) să fie clasificată în fiecare

gen. Apoi vom alege cea mai mare probabilitate dintre acestea. Vom calcula fiecare probabilitate folosindu-ne de Teorema lui Bayes:

$$\mathbb{P}(C_i|x) = \frac{\mathbb{P}(x|C_i)\mathbb{P}(C_i)}{\mathbb{P}(x)}$$

$\mathbb{P}(C_i|x)$ este probabilitatea ca piesa să fie în genul C_i știind că are vectorul de caracteristici x, $\mathbb{P}(C_i) = \frac{1}{10}$ deoarece considerăm fiecare gen la fel de probabil. $\mathbb{P}(x)$ poate fi ignorat deoarece este o constantă pentru un vector x, deci nu influențează ordinea valorilor $\mathbb{P}(C_i|x)$. Vom folosi următoare formulă pentru $\mathbb{P}(x|C_i)$:

$$\mathbb{P}(x|C_i) = \frac{1}{\sqrt{(2\pi)^d|\Sigma|}} \exp\left(-\frac{(x-\mu)^T\Sigma^{-1}(x-\mu)}{2}\right)$$

unde d este numărul de caracteristici pe care le avem (13 în cazul nostru, valoare obținută după aplicarea PCA-ului), μ este media datelor de test (vector) și Σ este matricea de covarință.

Folosim această formulă deoarece modelul pe care vrem să îl analizăm este unul continuu, având nevoie de o funcție de distribuție a probabilității. Vom folosi cea mai des utilizată: cea gaussiană. În plus, deoarece lucrăm cu vectori folosim versiunea multivariată a acestei formule.

4 Documentație cod

- **genres_original** - Dataset-ul
- **GaussianBayes.py** - Clasificare Bayes
- **createLabels.py** - Distributia pe categorie
- **featureExtraction.py** - Extragerea caracteristicilor
- **featureReduction.py** - Pastrarea caracteristicilor importante
- **features.npy**
- **labels.npy**
- **main.py** - Codul principal
- **pca.npy**

4.1 Fișierul `GaussianBayes.py`

Funcția `gaussianNaiveBayes` implementează un clasificator Gaussian Naive Bayes pentru a recunoaște genuri muzicale pe baza caracteristicilor extrase din fișiere audio. Pentru aceasta, folosim două importuri esențiale:

- `numpy` (importat ca `np`): `numpy` este utilizat pentru manipularea eficientă si rapidă a datelor în matrice și pentru calculul statistic. În funcție, `numpy` ne permite:

- **Organizarea datelor pe genuri muzicale** prin filtrarea datelor de antrenament (`trainData`) conform etichetelor (`trainLabels`).
- **Calculul mediei și covarianței** fiecărui gen, folosind `np.mean` și `np.cov`. Aceste valori sunt necesare pentru modelul Gaussian, care presupune că datele fiecărui gen sunt distribuite normal (Gaussiene).
- **Gestionarea probabilităților**: `np.vstack` organizează probabilitățile într-o matrice, iar `np.where` și `np.log` ne ajută să tratăm valorile zero și să calculăm logaritmii, pentru a evita erori și pentru a transforma produsul probabilităților într-o sumă.

- `scipy.stats.multivariate_normal`: `multivariate_normal` din `scipy.stats` este folosit pentru a calcula probabilitățile ca un set de caracteristici să aparțină unui gen specific, având în vedere distribuția multivariată Gaussiană. Folosim metoda `.pdf` pentru a obține densitatea de probabilitate pentru datele de testare (`testData`), bazându-ne pe media și covarianța fiecărui gen. Fiecare gen primește astfel o probabilitate, ajustată cu o probabilitate de bază de 10% (presupunând distribuție uniformă).

4.1.1 Explicația Procesului

1. **Separarea Datelor pe Genuri**:
   Datele de antrenament sunt împărțite pe genuri muzicale pe baza etichetelor. Astfel, fiecare gen („blues", „classical", etc.) are propriul set de date, care ajută la calculul statistic specific.

2. **Calculul Mediei și Covarianței**:
   Pentru fiecare subset de date de antrenament (fiecare gen), se calculează media și matricea de covarianță, esențiale pentru modelul Gaussian.

3. **Estimarea Probabilităților**:
   `multivariate_normal.pdf` calculează probabilitatea ca `testData` să aparțină fiecărui gen muzical, pe baza distribuției Gaussiene definite de media și covarianța respective.

4. **Gestionarea Valorilor Zero**:
   Introducerea unui mic `epsilon` (1e-100) ne permite să evităm erorile cauzate de logaritmul de zero. Valorile zero în matricea de probabilități sunt înlocuite cu `epsilon` înainte de a aplica logaritmul.

5. **Întoarcerea Log-Probabilităților**:
   La final, funcția returnează o matrice de log-probabilități pentru fiecare gen, oferind astfel o bază pentru clasificare, unde genul cu cea mai mare probabilitate logaritmică va fi considerat genul prezis.

4.2 featureExtraction.py

Acest script implementează funcții pentru extragerea diverselor caracteristici audio din fișiere muzicale, folosind `librosa` pentru procesarea audio și `numpy` pentru calcule și manipulări de date. De asemenea, folosește `os` pentru navigarea în directoare și pentru a accesa fișierele audio.

- `os`: `os` este utilizat pentru a interacționa cu sistemul de fișiere, permițând parcurgerea folderului care conține fișierele audio organizate pe genuri muzicale.
- `librosa`: `librosa` este o bibliotecă specializată în procesarea audio, utilizată aici pentru extragerea caracteristicilor audio precum chroma, energie, centroid spectral, etc.
- `numpy` (importat ca `np`): `numpy` este utilizat pentru manipularea eficientă a matricei de caracteristici și pentru calculele statistice (medie, variație).

4.2.1 Explicația Procesului

1. `getSongs()`: Această funcție navighează în folderul `genres_original` și adună toate căile către fișierele audio, organizate pe genuri. Returnează o listă cu căile complete către fiecare fișier audio.
2. `chroma(data)`: Calculează caracteristica **chroma**, care reflectă intensitatea fiecărui pitch în spectrul audio.
3. `rootMeanSquareEnergy(data)`: Calculează energia medie pătratică a amplitudinii (RMS) a semnalului audio, oferind informații despre intensitatea acestuia.
4. `spectralCentroidAndBandwidth(data)`: Calculează **centroidul spectral** (media frecvențelor) și **lățimea de bandă spectrală** (intervalul de frecvențe în care este distribuită energia).
5. `rolloff(data)`: Calculează **spectral rolloff**, frecvența sub care se află 85% din energia spectrală, oferind o idee despre claritatea și "asprimea" sunetului. Returnează media și variația.
6. `zero_crossing_rate(data)`: Măsoară rata de trecere prin zero, indicând cât de brusc sau „tăiat" este sunetul. Returnează media și variația.
7. `decompose_harmonic_percussive(data)`: Descompune semnalul în componente **armonice** (sunete melodice) și **percutive** (sunete ritmice). Returnează media și variația pentru fiecare componentă.
8. `tempo(data)`: Calculează **tempo-ul** piesei în bătăi pe minut (BPM), oferind o măsură a ritmului general al piesei.
9. `band_energy_ratio(data, sr)`: Calculează raportul energiei între benzile de frecvență joasă și înaltă, pentru a identifica cât de „grav" sau „acut" este sunetul.
10. `amplitude_envelope(data)`: Măsoară amplitudinea maximă într-un cadru de semnal, oferind o aproximare a intensității maxime.
11. `mel_frequency_cepstral_coef(data)`: Calculează coeficienții cepstrali pe scară Mel (MFCC), care reprezintă caracteristicile spectrale ale sunetului.

12. `compute_features()`: Funcția principală de calcul a caracteristicilor, care parcurge toate fișierele audio din `genres_original`, aplicând toate funcțiile de mai sus pentru a crea un vector de 61 de caracteristici per piesă. Salvează rezultatele într-un fișier `features.npy` pentru a evita recalculările.

13. `custom_compute_features(custom_song)`: Similar cu `compute_features()`, dar aplicat unei piese specificate de utilizator. Extrage aceleași 61 de caracteristici pentru piesa dată și adaugă vectorul de caracteristici în fișierul `features.npy`.

4.3 featureReduction.py

Funcția `PCA` implementează analiza componentelor principale pentru a reduce dimensionalitatea datelor audio. Aceasta transformă caracteristicile originale în componente principale, păstrând doar cele care contribuie semnificativ la variația totală. Funcția folosește trei importuri esențiale:

- `numpy` (importat ca `np`): `numpy` este folosit pentru manipularea eficientă a datelor numerice în matrice și pentru operațiuni statistice. În funcția `PCA`, `numpy` este esențial pentru:
  - **Încărcarea și centralizarea datelor**: `np.load` încarcă datele din fișierul `features.npy`. Funcția auxiliară `center` este folosită pentru a calcula media și a standardiza fiecare caracteristică, utilizând `np.mean`, `np.std`, și `np.divide` pentru a obține datele centrate și scalate.
  - **Calculul matricei de covarianță și valorilor proprii**: `np.cov` calculează matricea de covarianță a datelor centralizate, iar `np.linalg.eig` este utilizat pentru a obține valorile proprii și vectorii proprii ai matricei de covarianță, necesare pentru identificarea componentelor principale.
  - **Salvarea proiecției**: `np.save` este folosit pentru a salva proiecția finală a datelor reduse dimensional, într-un fișier numit `pca.npy`, astfel încât să poată fi folosită ulterior.

4.3.1 Explicația Procesului

1. **Încărcarea și Centralizarea Datelor**:
   Datele sunt încărcate din fișierul `features.npy` folosind `np.load`. Funcția auxiliară `center` calculează media fiecărei caracteristici și scade media din date, apoi împarte rezultatul la deviația standard pentru a obține datele standardizate.

2. **Calculul Matricei de Covarianță și Valorilor Proprii**:
   Funcția calculează matricea de covarianță a datelor centralizate folosind `np.cov`. Apoi, `np.linalg.eig` este utilizat pentru a obține valorile proprii și vectorii proprii ai acestei matrice, care vor determina direcțiile principale (componentele principale) de variație.

3. **Sortarea și Selectarea Componentelor Principale**:
   Valorile proprii sunt sortate descrescător folosind `np.argsort`, iar funcția identifică numărul minim de componente care explică un procent specificat din variația totală (setat prin `variance`, implicit 95%).

4. **Proiecția Datelor pe Componentele Selectate**:
   Datele centralizate sunt proiectate pe spațiul componentelor principale selectate prin multiplicarea cu vectorii proprii corespunzători. Aceasta reduce dimensionalitatea datelor păstrând informația esențială.

5. **Salvarea Proiecției Finale**:
   Proiecția finală a datelor pe componentele principale este salvată într-un fișier numit `pca.npy` folosind `np.save`, pentru a putea fi utilizată ulterior în analiză.

4.4 main.py

Acest script implementează codul principal pentru un proiect de clasificare a genurilor muzicale folosind analiza componentelor principale (PCA), clasificarea Naive Bayes, și extragerea de caracteristici personalizate. Scopul este de a prezice genul muzical al unui fișier audio și de a evalua performanța modelului prin metrici statistice. Scriptul folosește mai multe importuri esențiale:

- `numpy` (importat ca `np`): `numpy` este utilizat pentru manipularea eficientă a datelor și pentru calcule statistice. În acest script, `numpy` este esențial pentru:
  - **Încărcarea și salvarea datelor**: `np.load` și `np.save` sunt folosite pentru a încărca și salva datele în format `npy`.
  - **Manipularea matricelor**: Utilizăm funcții precum `np.zeros`, `np.argmax`, și `np.where` pentru a prelucra și analiza datele.
  - **Calcule statistice**: Calculăm metrici precum precizia, acuratețea, scorurile F și rata de reamintire folosind operații matriceale.

- `random`: Biblioteca `random` este folosită pentru a selecta aleatoriu datele de antrenament și testare, în funcție de raportul specificat. Metoda `random.sample` este utilizată pentru a selecta un subset de date pentru antrenare.

- `featureExtraction.custom_compute_features` (importată ca `ccf`): Funcția `ccf` este o metodă personalizată pentru extragerea caracteristicilor audio dintr-un fișier muzical specificat de utilizator. Aceasta extrage trăsături relevante pentru clasificarea genului și le salvează pentru procesare ulterioară.

- `featureReduction.PCA` (importată ca `pca`): Funcția `pca` reduce dimensionalitatea caracteristicilor extrase utilizând analiza componentelor principale (PCA). Aceasta este esențială pentru a simplifica datele și a le face mai ușor de procesat de către clasificator.

- `GaussianBayes.gaussianNaiveBayes` (importată ca `gnb`): Funcția `gnb` implementează clasificarea Naive Bayes pentru a prezice genul muzical bazat pe caracteristicile reduse obținute prin PCA. Aceasta calculează probabilitățile pentru fiecare gen muzical și alege genul cu probabilitatea maximă.

4.4.1 Explicația Procesului

1. `split_data(data_ratio)`: Împarte setul de date în date de antrenament și de testare pe baza unui raport specificat (`data_ratio`). Funcția `pca` este apelată pentru a reduce dimensionalitatea datelor. Returnează datele și etichetele de antrenament, datele și etichetele de testare, și vectorul de împărțire a datelor.

2. `calculate_metrics(metric_true_labels, metric_predictions, num_classes)`: Calculează metrici de performanță (precizie, reamintire și scor F) pentru fiecare clasă. Pentru fiecare gen muzical (clasă), se calculează:

   - **True Positives (TP)**: Numărul de predicții corecte pentru o clasă.
   - **False Positives (FP)**: Numărul de predicții incorecte în care o clasă a fost prezisă greșit.
   - **False Negatives (FN)**: Numărul de instanțe în care o clasă nu a fost prezisă corect.

   Se returnează listele de precizii, reamintiri și scoruri F pentru fiecare clasă.

3. `computing_statistics(data_true_labels, data_predictions)`: Evaluează performanța modelului pentru fiecare gen muzical, afișând metrici precum precizia, reamintirea, scorurile F și acuratețea generală. Datele sunt transformate pentru a fi utilizate în `calculate_metrics`, iar apoi rezultatele sunt afișate pe genuri muzicale.

4. `custom_category()`: Permite utilizatorului să prezică genul muzical al unei piese audio personalizate. Solicită utilizatorului să introducă calea către fișierul audio, extrage caracteristicile folosind `ccf`, aplică `pca` pentru reducerea dimensionalității și utilizează `gnb` pentru a prezice genul muzical. La final, afișează genul muzical prezis.

5. **Afișarea Statisticilor și Evaluarea Modelului**: Scriptul împarte datele în proporția 80:20 (antrenament:test) prin `split_data`, antrenează modelul cu datele de antrenament și face predicții pentru datele de testare folosind `gnb`. Apoi, `computing_statistics` este apelată pentru a afișa precizia, reamintirea, scorul F și acuratețea pe genuri, oferind o evaluare completă a modelului.

5 Bibliografie

1. fauci2013 Fauci, A., Cast, J., & Schulze, R. (2013). *Music Genre Classification.* Retrieved from cs229.stanford.edu

2. ieee2010 Tzanetakis, G., & Cook, P. (2002). *Musical genre classification of audio signals.* IEEE Transactions on Speech and Audio Processing, 10(5), 293-302. Retrieved from ieeexplore.ieee.org

3. ijctt2018 Mohan, S., et al. (2018). *Music Genre Classification using Machine Learning Techniques.* International Journal of Computer Trends and Technology (IJCTT) – Volume 62. Retrieved from ijcttjournal.org

4. youtube2021 Valerio Velardo - The Sound of AI. (2021, January 12). *The Math Behind Music Genre Classification.* [Video file]. Retrieved from youtube.com

5. *Principal Componenet Analysis by Victor Lavrenko* youtube.com

6. *Data set* huggingface.co