UNIVERSITATEA „ALEXANDRU IOAN CUZA” IAŞI

**FACULTATEA DE INFORMATICĂ**



LUCRARE DE LICENŢĂ

**Clasificarea automată a melodiilor în genuri muzicale**

**propusă de**

***Dorin – Andrei - Beniamin Miron***

**Sesiunea:** *iulie, 2017*

**Coordonator ştiinţific**

**Conf. Dr. Liviu Ciortuz**

**UNIVERSITATEA ALEXANDRU IOAN CUZA IAŞI**

**FACULTATEA DE INFORMATICĂ**

**Clasificarea automată a melodiilor în genuri muzicale**

***Dorin – Andrei - Beniamin Miron***

**Sesiunea:** *iulie, 2017*

**Coordonator ştiinţific**

**Conf. Dr. Liviu Ciortuz**

DECLARAŢIE PRIVIND ORIGINALITATE ŞI RESPECTAREA DREPTURILOR DE AUTOR

Prin prezenta declar că Lucrarea de licenţă cu titlul „*Clasificarea automată a melodiilor în genuri muzicale”* este scrisă de mine şi nu a mai fost prezentată niciodată la o altă facultate sau instituţie de învăţământ superior din ţară sau străinătate. De asemenea, declar că toate sursele utilizate, inclusiv cele preluate de pe Internet, sunt indicate în lucrare, cu respectarea regulilor de evitare a plagiatului:

* toate fragmentele de text reproduse exact, chiar şi în traducere proprie din altă limbă, sunt scrise între ghilimele şi deţin referinţa precisă a sursei;
* reformularea în cuvinte proprii a textelor scrise de către alţi autori deţine referinţa precisă;
* codul sursă, imaginile etc. preluate din proiecte *open*-*source* sau alte surse sunt utilizate cu respectarea drepturilor de autor şi deţin referinţe precise;
* rezumarea ideilor altor autori precizează referinţa precisă la textul original.

Iaşi, *data*

*Dorin – Andrei - Beniamin Miron*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

DECLARAŢIE DE CONSIMŢĂMÂNT

Prin prezenta declar că sunt de acord ca Lucrarea de licență cu titlul „*Clasificarea automată a melodiilor în genuri muzicale*”, codul sursă al programelor şi celelalte conţinuturi (grafice, multimedia, date de test etc.) care însoţesc această lucrare să fie utilizate în cadrul Facultăţii de Informatică.

De asemenea, sunt de acord ca Facultatea de Informatică de la Universitatea „Alexandru Ioan Cuza” Iași să utilizeze, modifice, reproducă şi să distribuie în scopuri necomerciale programele-calculator, format executabil şi sursă, realizate de mine în cadrul prezentei lucrări de licenţă.

Iaşi, *data*

*Dorin – Andrei - Beniamin Miron*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

# Rezumat

Genurile muzicale sunt etichete create de oameni pentru a caracteriza piesele muzicale și sunt folosite, în mod special, pentru a le organiza pe categorii. Un gen muzical este caracterizat de caracteristicile comune care sunt promovate de membri comunităților ce le susțin. Aceste trăsături, în mod normal, se referă la instrumentele muzicale folosite, la structura ritmului sau la armonia melodiei.

Clasificarea automată a melodiilor poate asista sau poate chiar înlocui modul în care se face în prezent acest proces: manual, de către oameni. Clasificarea automată pe baza genului poate oferi diverse tehnici pentru dezvoltarea și evaluarea atributelor pentru orice fel de analiză a fișierelor audio pe baza conținutului lor.

Această lucrare prezintă un sistem de clasificare automată a melodiilor în 10 genuri, în funcție de variația energiei din jurul mai multor frecvențe. Setul de date folosit este unul ce de-a lungul timpului s-a consacrat în probleme de acest gen. El este public și disponibil sub numele GTZAN. Clasificarea se face folosind mai mulți algoritmi din domeniul învățării automate. Se aplică, pe rând, 6 clasificatori, iar apoi, pe baza rezultatelor obținute este creat un clasificator expert care va fi antrenat pentru a acorda diferite ponderi clasificatorilor deja creați.

Cuprins

[Rezumat 1](#_Toc486426339)

[Introducere 5](#_Toc486426340)

[1 Definirea problemei 7](#_Toc486426341)

[2 Cercetări din domeniu 8](#_Toc486426342)

[2.1 “Clasificarea semnalelor audio în genuri muzicale” 8](#_Toc486426343)

[2.2 “Recunoașterea genurilor muzicale” 9](#_Toc486426344)

[*2.3* “Clasificarea genurilor muzicale” 9](#_Toc486426345)

[2.4 “Studiu pentru înțelegerea fișierelor audio” 10](#_Toc486426346)

[3 Setul de date si trăsăturile folosite 11](#_Toc486426347)

[3.1 Setul de date 11](#_Toc486426348)

[3.2 Trăsăturile MFCC 11](#_Toc486426349)

[3.2.1 Împărţirea în cadre (frame-uri) scurte 11](#_Toc486426350)

[3.2.2 Estimarea periodogramei spectrului de energie 12](#_Toc486426351)

[3.2.3 Logaritmarea energiilor 13](#_Toc486426352)

[3.2.4 Transformarea cosinusului discret 13](#_Toc486426353)

[4 Algoritmi utilizaţi 14](#_Toc486426354)

[4.1 Random Forest 14](#_Toc486426355)

[4.1.1 Arbori de decizie 14](#_Toc486426356)

[4.1.2 Bagging 15](#_Toc486426357)

[4.1.3 Îmbunătățirea algoritmului Bagging 16](#_Toc486426358)

[4.1.4 Pseudocod Random Forest 17](#_Toc486426359)

[4.2 Rețele neuronale artificiale 18](#_Toc486426360)

[4.2.2 Neuroni artificiali 18](#_Toc486426361)

[4.2.3 Structura rețelelor neuronale 19](#_Toc486426362)

[4.3 K-nearest neighbors (KNN) 21](#_Toc486426363)

[5.3.1 Pre-procesarea datelor pentru KNN 22](#_Toc486426364)

[5.3.2 Pseudocod KNN 22](#_Toc486426365)

[4.4 Regresia Logistică 23](#_Toc486426366)

[4.4.1 Calcularea probabilităților folosind regresia logistică 23](#_Toc486426367)

[4.4.2 Învăţarea modelului de regresie logistică 24](#_Toc486426368)

[4.4.3 Clasificarea cu regresia logistică 25](#_Toc486426369)

[5Gradient Boosting 26](#_Toc486426370)

[5.1 AdaBoost 26](#_Toc486426371)

[5.2 Gradient Boosting 27](#_Toc486426372)

[5.2.1 Funcția de eroare 27](#_Toc486426373)

[5.2.2 Clasificatorii slabi 27](#_Toc486426374)

[5.2.3 Îmbunătăţirea modelului 28](#_Toc486426375)

[6 Detalii de implementare 29](#_Toc486426376)

[6.1 Limbajul de programare 29](#_Toc486426377)

[6.2 Biblioteca scikit-learn 29](#_Toc486426378)

[6.2.1 Documentația 30](#_Toc486426379)

[6.2.2 Expertiza contribuitorilor 30](#_Toc486426380)

[6.2.3 Variația domeniilor acoperite 30](#_Toc486426381)

[6.2.4 Simplitate 31](#_Toc486426382)

[7 Metodologia de clasificare a melodiilor 32](#_Toc486426383)

[7.1 Modele individuale 33](#_Toc486426384)

[7.1.1 Regresie Logistică 33](#_Toc486426385)

[7.1.2 Rețele Neuronale Artificiale 35](#_Toc486426386)

[7.1.3 Gradient Boosting 36](#_Toc486426387)

[7.1.4 Random Forest 36](#_Toc486426388)

[7.1.5 KNN 36](#_Toc486426389)

[7.2 Asamblarea modelelor individuale 36](#_Toc486426390)

[8.Rezultate obţinute 38](#_Toc486426391)

[Concluzii şi direcţii de continuare a acestui studiu 39](#_Toc486426392)

[Anexa 1 40](#_Toc486426393)

[Anexa 2 42](#_Toc486426394)

[Blues VS. Rest 42](file:///C:\work\resources\eu\licenta%20final-%20Clasificarea%20melodiilor.docx#_Toc486426395)

[Clasic VS. Rest 42](file:///C:\work\resources\eu\licenta%20final-%20Clasificarea%20melodiilor.docx#_Toc486426396)

[Country VS. Rest 43](file:///C:\work\resources\eu\licenta%20final-%20Clasificarea%20melodiilor.docx#_Toc486426397)

[Disco VS. Rest 43](file:///C:\work\resources\eu\licenta%20final-%20Clasificarea%20melodiilor.docx#_Toc486426398)

[Jazz VS. Rest 44](file:///C:\work\resources\eu\licenta%20final-%20Clasificarea%20melodiilor.docx#_Toc486426399)

[Hiphop VS. Rest 44](file:///C:\work\resources\eu\licenta%20final-%20Clasificarea%20melodiilor.docx#_Toc486426400)

[Metal VS. Rest 45](file:///C:\work\resources\eu\licenta%20final-%20Clasificarea%20melodiilor.docx#_Toc486426401)

[Pop VS. Rest 45](file:///C:\work\resources\eu\licenta%20final-%20Clasificarea%20melodiilor.docx#_Toc486426402)

[Raggae VS. Rest 46](file:///C:\work\resources\eu\licenta%20final-%20Clasificarea%20melodiilor.docx#_Toc486426403)

[Rock VS. Rest 46](file:///C:\work\resources\eu\licenta%20final-%20Clasificarea%20melodiilor.docx#_Toc486426404)

[Anexa 3 47](#_Toc486426405)

[Anexa 4 49](#_Toc486426406)

[Bibliografie şi Webografie 51](#_Toc486426407)

# Introducere

Dimensiunea mare de informații multimedia de pe internet face necesară construirea unor noi tipuri de utilitare automate capabile să clasifice fișierele audio în genuri muzicale, pe baza conținutului lor. Importanța clasificării melodiilor în genuri muzicale a fost subliniată de către A.C.North și David J Hargreaves în studiul lor[[1]](#endnote-1). Aceștia au făcut un experiment prin care au vrut să observe cât de mult îi influentează pe ascultători genul muzical în care este susținută o piesă muzicală. Rezultatul acestui studiu a fost că ascultătorii asociază mai ușor melodii diferite care aparțin aceluiași gen, decât aceeași melodie cântată în genuri diferite.

Necesitatea unui mecanism eficient de clasificare automată a melodiilor crește direct proporțional cu numărul de melodii disponibile pe internet. Existența unor algoritmi capabili să clasifice automat melodiile în diferite genuri sunt de un real folos celor ce lucrează în domeniul muzical, în special în cazul administratorilor unor colecții mari de melodii. Aceste colecții, în momentul de față, sunt grupate manual. Ca orice alt proces manual și acesta este greoi și anevoios. Mai mult decât atât în cazul clasificării manuale poate apărea și subiectivitatea și se poate ajunge la situația în care diferiți oameni clasifică aceeași melodie în două genuri diferite, în funcție de preferințele lor. Acest lucru poate duce la multe inconsistențe.

Chiar și muzicologii și-au arătat interesul față de aplicații practice care să clasifice fișierele audio după gen. În momentul de față există o cunoaștere limitată a mecanismului pe care noi, ca oameni, îl folosim, în mod natural, pentru a grupa melodiile în genuri muzicale. Nu știm exact atributele ce sunt utilizate precum nici cât de importante sunt. Un sistem care să poată clasifica, în mod automat fisierele audio, ar putea oferi aceste informații.

Lucrarea de față prezintă un sistem de clasificare a melodiilor în 10 genuri, în funcție de coeficientii MFCC. Aceşti coeficienţi au fost calculaţi pentru prima dată de Davis şi Mermelstein în 1980[[2]](#endnote-2) şi de atunci sunt un punct de referinţă în domeniul procesării fişiserelor audio. Atunci când au fost creați acești coeficienţi s-a încercat să se înţeleagă modul în care urechea umană modelează sunetele şi să se reproducă acest proces.

Setul de date folosit pentru antrenarea modelului este GTZAN. Acesta a fost folosit pentru prima dată în domeniul analizei pieselor audio de către G.Tzanetakis si P.Cook[[3]](#endnote-3). Fişierele sunt colectate în anul 2000-2001 din mai multe surse inclusiv CD-uri personale, radio şi chiar înregistrări cu microfonul pentru a reprezenta o varietate de condiţii de înregistrare a unei piese audio. Setul de date constă în 1000 de melodii de lungime de 30 de secunde. Aceste melodii sunt împărţite în 10 genuri; câte 100 pentru fiecare gen muzical: blues, clasic, country, disco, hip-hop, jazz, metal, pop, reggae şi rock.

Prima abordare a fost aceea de a găsi un singur clasificator, cel mai bun, care să obţină cea mai mare acurateţe. S-au încercat, pe rând, următorii clasificatori: Regresie Logistică, Reţele Neuronale, Maşini pe Vector Suport (SVM), Gradient Boosting, Random Forest şi KNN, fiecare cu acurateţea lui proprie. Unii clasificatori s-au pliat pe această problemă, având o acurateţe destul de bună, alţii nu s-au potrivit la fel de bine. Cea mai bună acurateţe a fost cea a algoritmului Random Forest. Rata sa de succes a fost de 77.975%.

Apoi, în încearcarea de a obţine o acurateţe mai bună, s-a mai folosit un clasificator care are rolul de a acorda diferite ponderi celorlalţi algoritmi aminiţi deja. Această tehnică este cunoscută ca folosirea unui clasificator de experţi. S-a dovedit că această abordare a dus la obţinerea unei acuraţi de 88.275%.

Cercetarea în acest domeniu poate avea implicaţii mai mari decât doar clasificarea după genuri muzicale. Tehnicile dezvoltate pentru crearea sistemelor de clasificare a melodiilor în diferite genuri pot fi adaptate şi pentru alte tipuri de clasificare, spre exemplu clasificarea stilului sau a perioadei de timp când a apărut o melodie.

# 1 Definirea problemei

Tehnologia actuală permite memorarea semnalelor sonore în format digital şi pot fi reprezentate sub forma unei secvenţe, unde si reprezintă semnalul sonor de la momentul i, iar N reprezintă dimensiunea fişierului audio sau numărul de momente memorate. Această secvenţă conţine foate mule informaţii acustice, trăsături ale ritmului, precum şi armonia melodiei ce pot fi extrase şi interpretate. Iniţial se extrag anumite trăsături la nivel de cadre scurte ale semnalului sonor, apoi aceste trăsături se agregă pentru a obţine o imagine de ansamblu despre întreaga secventă ce formează fişierul audio. Aşadar, din secvenţa iniţială se poate extrage un vector de trăsături în care fiecare atribut xj este extras din S printr-o anumită procedură. În lucrarea de faţă vectorul de trăsături reprezintă coeficienţii MFCC.

Având un vector ce conţine anumite trăsături ale unei piese muzicale putem să definim problema de clasficare automată a melodiilor după gen ca o problemă clasică de clasificare, folosind atributele obţinute drept valori de intrare. Din mulţimea finită de genuri muzicale G={blues, clasic, country, disco, hip-hop, jazz, metal, pop, reggae şi rock}, trebuie aleasă o singură categrorie care reprezintă cel mai bine genul muzical al melodiei asociate secvenţei de semnal digital S.

Dintr-o perspectivă statistică modalitatea de a găsi cel mai probabil gen, dându-se vectorul de caracteristici, este definită în felul urmator:

unde reprezintă probabilitatea ca dându-se vectorul de atribute X, acesta să descrie o piesă muzicală care să aparţină clasei g, P(g) reprezintă probabilitatea apriori ca o nouă melodie să aparţină unui anumit gen (această probabilitate poate fi estimată din frecvenţa genului g din setul de date), iar P() reprezintă probabilitatea de apariţie a vectorului Ultima probabilitate este necunoscuta, de obicei, dar se poate calcula ştiind că . Deci probabilitatea cautată poate fi calculată, din punct de vedere statistic, pentru fiecare cu ajutorul următoarei formule

**.**

# 2 Cercetări din domeniu

Din perspectiva recunoasterii şi creării de modele, clasificarea melodiilor în funcţie de gen este o problemă interesantă de cercetare, din moment ce melodiile au o dinamică a semnalului audio foarte variată. Un alt aspect interesat este că clasificarea genurilor este o problemă cu mai mult de două categorii. Pentru a putea face clasificări cu mai mult de doua clase există cel puţin două abordări:

1. Folosirea tehnicilor şi a algoritmilor care pot clasifica instanţele în mai multe clase, în mod natural, producând o separare complexă a spaţiului variabilelor de intrare. Ca exemplu astfel de algoritmi pot fi amintiţi arborii de decizie, KNN sau reţelele neuronale.

2. Folosirea unei strategii pentru descopunerea problemei de clasificare în mai multe clase într-o serie de subprobleme de clasificare binară, folosind algoritmi ce vor clasifica spaţiul de soluţii în mod binar. Algoritmi de acest fel sunt SVM sau clasificare logistică.

De câteva ori s-a încercat o abordare în manieră asamblistă. Aceste abordări s-au dovedit a fi un succes. Ele constau în aplicarea a mai multor clasificatori, nu doar a unuia singur, fiecare specializat pe o anumită parte a problemei. În acest fel, fiecare clasificator este antrenat pe câte un subset diferit de atribute, iar rezultatul final va fi combinaţia acestor clasificatori de către un alt clasificator ce este antrenat în îmbinarea acestor modele în mod dinamic. Această abordare poate fi privită şi ca o mixtură de experţi aplicată la problema clasificării. De asemenea, poate fi privită şi ca o variaţie a algoritmului boosting.

O posibilă explicaţie a succesului acestei metode este că aplicarea clasificatorilor pe doar un subset de atribute face ca suprafaţa de decizie să fie mult mai simplă, iar clasificatorii pot să producă rezultate cu acurateţe mai mare.

## 2.1 “Clasificarea semnalelor audio în genuri muzicale” 3

Tzanetakis şi Cook au fost primii care au abordat şi definit această problemă. În studiu3 lor de cercetare ei au propus un set cuprinzător de trăsături ce pot reprezenta o piesă muzicală. Aceste trăsături sunt obţinute în urma procesării semnalelor sonore şi includ printre altele şi atribute ale timbrului vocal, ale ritmului şi ale notelor muzicale folosite. Ca algoritmi de clasificare ei au folosit clasificarea gaussiană, mixtura gausiană şi clasificatorul KNN (K Nearest-Neighbors).

Experimentul s-a făcut pe setul de date GTZAN, acelaşi set pe care l-am folosit şi noi, în această lucrare. Setul de date constă în 1000 de instanţe ce împărţite în 10 genuri muzicale, cu trăsături extrase din primele 30 de secunde ale fiecărei melodii. Rezultatele obţinute de ei indică o acurateţe în jur de 50% folosind toate cele 10 genuri.

## 2.2 “Recunoașterea genurilor muzicale” [[4]](#endnote-4)

K. Kosina a dezvoltat MUGRAT (**Mu**sic **G**enre **R**ecognition by **A**nalysis of **T**exture) în lucrarea sa de dizertaţie. Acesta este un prototip capabil să recunoască genul muzical al unei melodii bazat pe subsetul de date oferit de cei de la MARSYAS. În acest caz atributele au fost selectate doar din segmente de lungime de 3 secunde alese în mod aleator din întregul fişier audio.

Experimentele au fost făcute într-o bază de date alcătuită din 186 de melodii din 3 genuri diferite. Folosind algoritmul de clasificare 3-NN, Kosina a obţinut un model ce are acurateţea de aproximativ 88%. În această cercetare, autoarea studiului confirmă faptul că încercarea manuală de a clasifica piesele muzicale după gen duce la rezultate inconsistente.

## *2.3* “Clasificarea genurilor muzicale” *[[5]](#endnote-5)*

Li, Ogihara şi Li au prezentat un studiu comparativ între atributele incluse în setul de date MARSYAS şi un alt set de atribute bazate pe DWCH (**D**aubechies **W**avelet **C**oefficient **H**istograms), folosind alţi algoritmi de clasificare precum SVM (**S**upport **V**ector **M**achines) si LDA (**L**inear **D**iscriminant **A**nalysis). Pentru comparare ei au luat două seturi de date: (a) setul de date folosit iniţial de Tzanetakus şi Cook (GTZAN), cu atributele extrase de la inceputul melodiei şi (b) un set de date compus de 755 de piese muzicale împărţite în 5 genuri, cu atributele extrase din fiecare piesă muzicală de la secunda 31 până la secunda 61. Aceste experimente au arătat faptul că algoritmul de clasificare SVM este mai bun decat celelalte: în cazul (a) având o acurateţe de 72% folosind atributele originale şi o acurateţe de 78% folosind atributele DWCH. În cazul (b) acurateţea a fost de 71% pe atributele originale şi 74% pe atributele generate de DWCH.

Autorii studiului au mai făcut încă o comparaţie a strategiilor de descompunere a problemei de clasificare a mai multor categorii în mai multe subprobleme de clasificare binară. Problema iniţială de a împărţi setul de date în 5 genuri a fost descompusă într-o serie de probleme de clasificare binară: atât folosind strategia OAA (**O**ne-**A**gainst-**A**ll) cât şi folosind RR (**R**ound-**R**obin). Rezultatele cele mai bune au fost obţinute folosind algoritmul SVM şi metoda OAA de descompunere a problemei.

## 2.4 “Studiu pentru înțelegerea fișierelor audio”*[[6]](#endnote-6)[[7]](#endnote-7)*

Grimaldi, Cunningham şi Kokaram au încercat diverse strategii de descopunere a acestei probleme folosind algoritmi de clasificare specializaţi doar pe o parte din problemă şi apoi combinarea rezultatelor cu un clasificator expert. Autorii au descompus problema originală folosind OAA (**O**ne-**A**gainst-**A**ll), RR (**R**ound-**R**obin) şi o tehnică de selectarea aleatoare în subspaţii.

Ei au mai comparat şi diferite metode de selecţie a trăsăturilor ordonându-le după importanţa lor folosind mai multe tehnici de ordonare a lor precum IG (**I**nformation **G**ain) sau GR (**G**ain **R**atio) sau folosind algoritmi de combinare a atributelor precum PCA (**P**rincipal **C**omponent **A**nalysis).

Cercetarea lor s-a bazat pe un set de date ce conţine 200 de piese muzicale ce sunt grupate în 5 genuri şi au folosit KNN ca algoritm de clasificare. Trăsăturile au fost obţinute de pe întregul fişier audio folosindu-se DWT (Discrete Wavelet Transform). Pentru clasificatorul KNN transformarea PCA a fost cea mai eficientă tehnică de selecţie a trăsăturilor ajungându-se la o acurateţe de 79%. Metoda RR de asamblare a reuşit să obţină o acurateţe de 81% şi când s-a folosit IG dar şi atunci când s-a folosit GR.

# 3 Setul de date si trăsăturile folosite

## 3.1 Setul de date

Setul de date folosit pentru antrenarea modelului este GTZAN. Acesta a fost folosit pentru prima dată în domeniul analizei pieselor audio de către G.Tzanetakis si P.Cook3.

Fişierele sunt colectate în anul 2000-2001 din mai multe surse inclusiv CD-uri personale, radio şi chiar înregistrări cu microfonul pentru a reprezenta o varietate de condiţii de înregistrare a unei piese audio. Setul de date constă în 1000 de melodii de lungime de 30 de secunde. Aceste melodii sunt împărţite în 10 genuri; câte 100 pentru fiecare gen muzical: blues, clasic, country, disco, hip-hop, jazz, metal, pop, reggae şi rock.

## 3.2 Trăsăturile MFCC

Primul pas ce trebuie făcut pentru clasificarea melodiilor este acela de a extrage anumite trăsături din ele. Partea cea mai importantă este încerarea de a înţelege modul în care urechea umană modelează sunetele şi cât de bine acest proces este reprodus.

**M**el **F**requency **C**oefficients (MFCC) sunt caracteristici folosite foarte frecvent în vorbirea automată şi în recunoaşterea vocală. Aceşti coeficienţi au fost calculaţi prima dată de Davis şi Mermelstein[[8]](#endnote-8) în 1980 şi de atunci au rămas un punct de referinţă în domeniul procesării fişiserelor audio.

Atunci când vrem să calculăm aceşti coeficienţi trebuie să trecem prin mai multe etape:

1. Împărţirea în cadre (frame-uri) scurte;
2. Pentru fiecare frame se va face o estimare a periodogramei spectrului de energie;
3. Se logaritmează fiecare energie;
4. Se calculează DCT din valorile logaritmate la pasul anterior;
5. Se păstrează primii 13 coeficienţi.

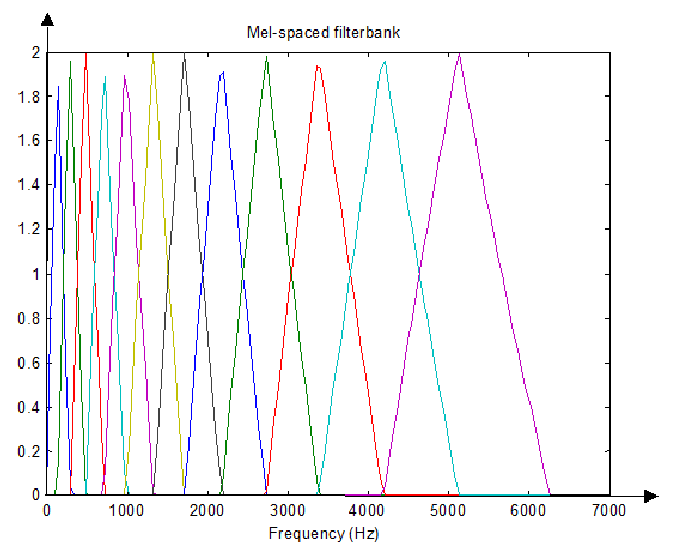
### 3.2.1 Împărţirea în cadre (frame-uri) scurte

Un semnal audio se schimbă în continuu, dar pentru simplitate putem presupune că pentru perioade scurte de timp, semnalul audio nu se schimbă foarte mult (această presupunere este făcută din punct de vedere statistic; este evident faptul că se schimbă chiar şi în intervale foarte mici de timp). De aceea vom împărţi semnalul audio în frame-uri de 20-30 milisecunde. Dacă frame-ul ar fi mai mic atunci nu ar fi existat destule date pentru a realiza o estimare rezolabilă a variaţiei energiei, iar dacă frame-ul ar fi mai mare atunci diferenţele dintre frame-uri nu ar putea fi atribute rezonabile pentru problema identificării genului unei melodii.

### 3.2.2 Estimarea periodogramei spectrului de energie

Următorul pas este să calculăm energia din jurul mai multor frecvenţe pentru fiecare frame. Acest pas simulează funcţia cohleei, denumită şi melcul membranos. Acesta este un organ din urechea internă umană care vibrează în diferite locuri în funcţie de frecvenţa pe care o are sunetul. În funcţie de locul în care vibrează cohleea, diferiţi nervi se activează şi primesc informaţii pe care le vor transporta către creier. Periodograma face cam acelaşi lucru, identificând frecvenţa din fiecare frame.

Estimarea făcută de periodogramă încă mai conţine multe informaţii nefolositoare. În special cohleea nu poate face diferenţa dintre două frecvenţe apropiate. Acest efect devinte din ce în ce mai pronunţat pe măsură ce frecvenţa creşte. Din acest motiv, vom grupa mai multe estimări şi le vom aduna pentru a avea o idee despre cât de mult variază energia în diferite regiuni ale melodiei. Acest lucru este făcut de filtrarea Mel. Filtrarea Mel cuprinde o grupare de filtre: primul filtru ne va spune cât de multă energie exista aproape de 0 Hertz. Cu cât frecvenţa creşte, filtrele se adaptează astfel încât diferenţa de energie să nu mai fie o problemă.



Filtrarea MEL15

### 3.2.3 Logaritmarea energiilor

După ce s-a filtrat energia din sunetul audio, pe mai multe frecvențe, trebuie să le logaritmăm. Această logaritmare se face pentru a ne apropia cât mai mult de modul în care funcționează urechea umană: noi nu auzim intensitatea sonoră pe o scală liniară. În general, pentru a dubla volumul unui sunet trebuie să fie consumată de până la 8 ori mai multă energie. Acest lucru înseamnă că variațiile mari de energie pot să nu sune la fel de diferit dacă sunetul era mult mai intens la început. Această operație face ca atributele noastre să fie mai asemănătoare cu ceea ce auzim.

### 3.2.4 Transformarea cosinusului discret

Ultimul pas este o transformare numită DCT (transformarea discretă a cosinusului). Ea este aplicată peste valorile obținute la pasul anterior. Există cel puțin 2 motive mari pentru care se execută această operație. În primul rând, filtrele pe care le-am aplicat se suprapun și sunt destul de corelate între ele. DCT reușește să decoreleze energiile. În al doilea rând, coeficienții MFCC conțin doar 13 valori, nu 26, câți rezultă după filtrarea MEL. Acest lucru se întâmplă deoarece ultimii 13 coeficienți conțin schimbări foarte bruște de energie și acest lucru afectează procesarea ulterioară.

# 4 Algoritmi utilizaţi

## 4.1 Random Forest

### 4.1.1 Arbori de decizie

Arborii de decizie constituie o metodă foarte folosită în practică. Ei pot fi folosiți atât pentru clasificare, cât și pentru regresie. Arborii de decizie antrenați pot fi reprezentați și ca un set de reguli de tip if-else, pentru a putea fi citiți mai usor. Iar etapa de antrenare nu este altceva decât învățarea ordinii atributelor ce trebuie testate și a condițiilor ce trebuie validate.

Bărbat

Bărbat

Femeie

DA

NU

DA

NU

Arborii de decizie clasifică instanțele sortându-le de-a lungul drumului de la rădăcină la frunze. Fiecare nod din arbore este, de fapt, un test asupra unui atribut al instanței ce urmează să fie clasificat. Fiecare ramură ce descinde dintr-un nod corespunde unei posibile valori a atributului ce tocmai a fost testat.

Clasificarea unei noi instanțe pornește de la rădăcina arborelui, apoi se testează, pe rând, atributele specificate în nodul curent, pentru ca, mai apoi, să se treacă la subarborele ce corespunde valorii atributului instanței în cauză. Acest proces se repetă până când se ajunge la o frunză a arborelui. Valoarea din frunza la care s-a ajuns este categoria în care va fi încadrată noua instanță.

Algoritmul învață arboriele de dicizie construindu-i de sus (de la rădăcină) în jos (spre frunze). Cea mai importantă alegere pe care o face acest algoritm este selectarea trăsăturii de test pentru fiecare nod din arbore. La fiecare pas se alege acel atribut care este cel mai folositor pentru clasificare. Pentru a măsura care atribut este cel mai potrivit pentru un anumit pas s-a introdus o proprietate statistică numită “informația câștigată” (**I**nformation **G**ain), ce măsoară cât de bine separa datele de antrenare atributul ales.

Pentru a putea vorbi de IG (**I**nformation **G**ain) trebuie definită o modalitate de măsurare a (im)puritații unei colecții de exemple (de instanțe), pe care o vom numi **entropie**. O colecție este mai ușor de clasificat dacă ea este mai pură. Când se alege un atribut pentru un nod, acesta trebuie să împartă colecția de exemple în subcolecții a căror sumă de impurități sa fie cat mai mică decât impuritatea colecției inițiale. Altfel, acea divizare nu a adus nici un caștig de informație.

Dându-se o colecție de instanțe S, cu exemple atât pozitive cât și negative, entropia colecției S relativă la clasificarea booleană este:, unde este proporția exemplelor pozitive din S, iar este proporția exemplelor negative din S. În calcule se va considera că 0 \* log 0 este egal cu 0.

Având o modalitate de a măsura impuritatea datelor de antrenare, se poate defini castigul informational, adică putem măsura cât de bun este un atribut pentru clasificarea datelor. IG-ul (Information Gain) nu face altceva decât să spună dacă atunci când se împart datele de antrenare după un atribut entropia va fi mai mica. Cu alte cuvinte, avem un câștig de informație atunci când, împărțind colecția de date după un anumit atribut, suma entropiilor subcolecțiilor este mai mică decât entropia colecției inițiale. Formal putem scrie acest lucru în felul urmator:

Unde P este o colecție de persoane, iar h este un atribut al persoanelor și anume înălțimea.

### 4.1.2 Bagging

Bootstrap Aggregation (Bagging) este o metodă foarte simplă dar și foarte puternică de asamblare. O metodă de asamblare este o tehnică ce combină predicția făcută de mai mulți clasificatori pentru a obține o predicție ce are acuratețea mai mare decât orice alt model luat individual.

Arborii de decizie sunt sensibili la datele pe care sunt antrenați. Cu alte cuvinte, dacă datele de antrenare sunt schimbate, sau dacă arborele e antrenat pe un subset din datele de intrare, atunci modelul rezultatat poate fi destul de diferit și poate chiar să facă predicții diferite.

Să presupunem că avem un set de date de 1000 de instanțe și vrem să creem arbori de decizie. Metoda Bagging funcționează în felul următor: în primul rând se crează mai multe sub-seturi de date, împărțind setul inițial (ex. se vor crea 100 subseturi). Apoi pentru fiecare subset de date se va antrena câte un arbore de decizie. Ultima fază este cea de predicție: pentru a reuși să facem o predicție pentru o nouă instanță, o vom clasifica cu fiecare arbore de decizie creat, iar predicția finală va fi categoria în care a fost încadrată de cele mai multe ori noua instanță.

Spre exemplu să presupunem ca au fost antrenați 5 arbori de decizie care fac următoarele predicții pentru o nouă instanță: femeie, femeie, barbat, femeie, barbat. Vom considera că instanța aparține categoriei care a apărut de cele mai multe ori. În acest exemplu vom face predicția “femeie”.

### 4.1.3 Îmbunătățirea algoritmului Bagging

Random Forest este unul dintre cei mai cunoscuți, dar in același timp și unul din cei mai puternici algoritmi din învățarea automată. Este o îmbunătațire a algoritmului de învațare automată descris mai sus, Bootstrap Aggregation sau Bagging.

Problema arborilor de decizie este că sunt construiți în manieră greedy. Adică, la fiecare pas trebuie să aleagă o variabilă după care să facă divizarea spațiului datelor de intrare astfel încât să minimizeze eroarea de antrenare (și, în mod implicit și eroarea de testare/validare). Chiar dacă aplicăm metoda Bagging, arborii de decizie rezultați pot avea o structură similară, iar predicția lor va fi corelată.

Combinarea predicțiilor de la multiple modele are rezultate mai bune atunci când predicțiile sub-modelelor sunt necorelate sau cât mai puțin corelate. Random Forest schimbă algoritmul de creare al arborilor într-o manieră în care predicția lor va fi mai puțin corelată.

Când se construiesc, în modul clasic, arborii de decizie, atunci când algoritmul trebuie să selecteze o trăsătură pentru a face divizarea, el are la dispoziție toate atributele datelor de intrare și toate valorile posibile pentru a selecta divizarea optima. Algoritmul de Random Forest modifică această procedură în așa fel încât algoritmul de învățare să fie limitat la un eșantion aleator de atribute din care să facă alegerea.

Numărul de atribute din care se poate alege la fiecare divizare poate fi specificat ca parametru pentru algoritm. Pentru a găsi cel mai bun numar de trăsături se folosesc diferite valori și se compară rezultatele de la cross-validare. Valoarea standard pentru clasificare este radical din numărul total de atribute, iar pentru regresie se va alege doar o treime din totalul de atribute.

### 4.1.4 Pseudocod Random Forest

Parametri: Un set de date de antrenare S=(x1, y1), ..., (xn, yn), atributele F şi numărul de arbori folosiţi B

1 function RandomForest(S, F)

2 H ←

3 for

4 S(i) ← Se seleactează un eşantion din S

5 hi ← CreazaArboreNou(S(i), F)

6 H ← H

7 end for

8 return H

9 end function

10 function CreazaArboreNou(S, F)

11 Pentru fiecare nod:

12 f ← o submulţime foarte mică a lui F

13 Împarte folosind cea mai buna trăsătură din f

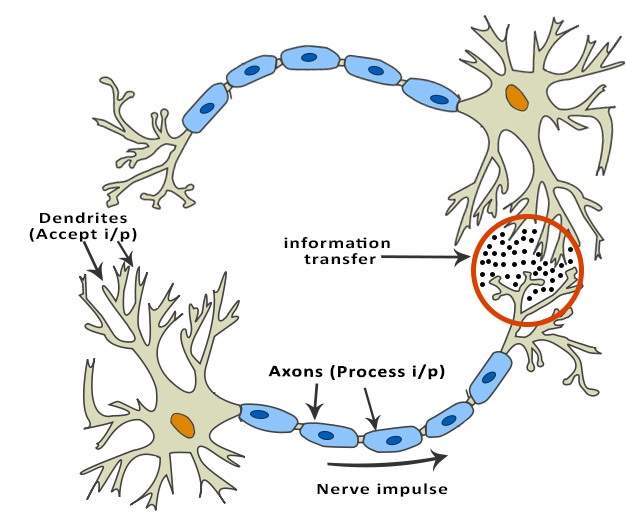
14 return Arborele creat

15 end function

## 4.2 Rețele neuronale artificiale

Ideea din spatele rețelelor neuronale artificiale se bazează pe ipoteza că ceea ce se întâmplă în creierul nostru atunci când luam deciziile corecte, poate fi imitat folosind calculatorul, simulând în acest fel neuronii și dendritele.

Corpul uman este compus din 100 de miliarde de celule nervoase. Ele sunt conectate unele cu altele prin axoni. Stimulii din mediu extern sau datele de intrare de la organele senzoriale sunt acceptate de dendrite. Acești stimuli crează impulsuri electrice care traversează rețeaua neuronală. Un neuron poate trimite mesajul la un alt neuron pentru a se ocupa de problemă sau poate decide să nu-l trimită mai departe, caz in care stimulul extern este ignorat.



Exemplu de funcționare a neuronului14

Rețelele Neuronale Artificiale (**A**rtificial **N**eural **N**etworks - ANN) sunt compuse din noduri multiple, care încearcă să imite funcționalitatea neuronilor biologici din creierul uman. Neuronii sunt conectați și interacționează unii cu alții prin legăturile dintre ei. Nodurile primesc valori de intrare și aplică operații simple asupra acestor date. Rezultatul acestor operații este pasat altor neuroni.

Puterea rețelelor neuronale vine din abilitatea lor de a învăța, pe de o parte, reprezentarea datelor de antrenare și, pe de alta parte, relația dintre aceste date cu variabila de ieșire, care trebuie prezisă. În acest sens, rețelele reuronale învață o mapare. Din punct de vedere matematic, ele sunt capabile să învețe orice funcție și a fost demonstrat că pot fi privite ca un algoritm universal de aproximare.

### 4.2.2 Neuroni artificiali

Unitatea de bază dintr-o rețea neuronală sunt neuronii. Aceștia sunt unități simple de calcul ce au câte o pondere pentru fiecare valoare de intrare și se calculează valoarea de ieșire folosind o anumită funcție de activare.

Funcția de activare

Output

Input

Pondere

Pondere

Pondere

Ponderile sunt de obicei inițializate cu valori aleatoare mici, cuprinse în intervalul [0,0.3], dar există și modalități mai complexe de inițializare a lor, dar pe care nu le vom discuta aici.

Funcția de activare transmite o valoare mai departe în funcție de suma produselor dintre ponderi și atributele de intrare. Această valoare reprezintă rezultatul neuronului. Daca suma este mai mare decat un anumit prag (sa spunem mai mare de 0.5), atunci rezultatul neuronului va fi 1, altfel va fi 0.

Cel mai mult des folosite funcții de activare sunt cele neliniare. Aceste funcții permit rețelelor neuronale să combine mai multe valori de intrare în moduri mult mai complexe, îmbunătățind astfel capabilitatea modelului creat de întreaga rețea. Un exemplu de funcție neliniară este funcția logistică (sau sigmoid). Rezultatul este o valoare reală între 0 și 1.

### 4.2.3 Structura rețelelor neuronale

Neuronii sunt grupați și formează, împreună, rețele neuronale.

Un rând de neuroni formează un strat din rețeua neuronală. În practică, rețelele neuronale sunt construite din mai multe strate.

#### 4.2.3.1 Nivel de intrare

Primul strat se mai numește stratul de intrare (“Input Layer”) sau stratul vizibil deoarece cu acest strat se interacționează, fiind partea expusa din rețeaua neuronală care intermediază trecerea de la datele de intrare la restul rețelei. De obicei, acest strat are câte un neuron pentru fiecare atribut pe care îl au instanțele cu care se lucrează. Acest strat nu conține același tip neuroni ca cei pe care i-am descris mai sus, ci sunt pur și simplu niște intermediari pentru a face legătura dintre celelalte straturi și datele de intrare.

#### 4.2.3.2 Nivelurile ascunse

Următoarele straturi, după cel de intrare sunt numite “straturi ascunse” deoarece ele nu interacționează cu datele de intrare ci, ele, primesc ca valori de intrare, rezultatul stratului anterior. Cea mai simplă structură de rețea reuronală este cea care are un singur neuron în stratul ascuns. Rezultatul acestui neuron va constitui și rezultatul final al rețelei.

Având în vedere puterea de calcul disponibilă și ușor de accesat[[9]](#endnote-9)dar și bibliotecile eficiente ce există, se pot construi așa numitele “deep neural networks”. Acestea sunt speciale, deoarece conțin multe straturi ascunse. În trecut, dura foarte mult ca acestea să fie antrenate, dar acum acest timp a fost redus la câteva minute folosind noi tehnici și hardware modern.

#### 4.2.3.3 Nivelul de iesire

Ultimul strat se numește stratul de ieșire și este responsabil pentru a întoarce o valoare sau un vector de valori ce corespund formatului cerut de problemă. La ultimul strat se pune problema a cât de mulți neuroni trebuie să existe aici și ce funcție de activare este mai bună având în vedere problema care trebuie rezolvată.

Pentru o problemă de regresie acest strat este construit dintr-un singur neuron și nu este necesară nici o funcție de activare.

Pentru o problemă de clasificare binară e recomandat să fie tot un singur neuron pe acest strat și să se folosească funcția sigmoid (numită și funcția logistică ce este folosită la regresie logistică), funcție a cărui rezultat este o valoare între 0 și 1 reprezentând probabilitatea ca instanța curentă să aparțină primei categorii. Iar apoi clasificarea se va face folosind un prag cu valoarea de 0.5. Dacă probabilitatea este mai mare de 0.5 atunci instanța va fi clasificată în prima categorie, altfel va fi clasificată în a doua categorie.

Pentru o problemă de clasificare, cu mai mult de două categorii, este bine să avem mai mulți neuroni pe ultimul strat. În practică se folosește câte un neuron pentru fiecare categorie. În acest caz funcția de activare recomandată este softmax a cărui rezultat este probabilitatea prezisă de rețea ca instanța testată să aparțină fiecărei categorii. Clasificarea constă în a alege categoria cu cea mai mare probabilitate din aceste valori.

## 4.3 K-nearest neighbors (KNN)

KNN nu construieşte alt model decât acela de a memora întregul set de date de antrenare, deci nu este nevoie de o etapă de învăţare. Deoarece setul de date este memorat, aceste date trebuie să fie consistente. În practică, se recomandă ca setul de date sa fie pre-procesat. Actualizarea modelului cu date noi se poate face foarte uşor, fără să fie nevoie de procesări complexe.

Predicţia pentru o nouă instanţă (X) se bazează pe căutarea în întregul set de date a celor mai asemănătoare K instanţe (a celor mai apropiaţi K vecini). În cazul regresiei rezultatul va fi media valorilor vecinilor, iar în cazul clasificării rezultatul va fi categoria cu cel mai mare număr de vecini.

Pentru a determina care sunt cele mai apropiate K instanţe cu instanţa ce trebuie clasificată trebuie să se folosească o măsură de distanţă. Pentru valorile reale ale variabilelor de intrare, cea mai cunoscută măsură de distanţă este distanţa euclidiană. Aceasta se defineşte ca fiind rădăcina pătrată a sumei diferenţelor valorilor de pe fiecare dimensiune la pătrat dintre punctul nou (Xnou) si punctul existent (Xi). Aceasta este distanţa clasică dintre două puncte.

Alte măsuri de distanţe populare sunt:

* Distanţa Hamming: Calculează distanţa dintre doi vectori binari;
* Distanta Manhattan: Calculează distanţa dintre doi vectori reali, sumând valoarea absolută a diferenţelor;
* Distanta Minkowski: este o generalizare a distanţei euclidiene şi a distanţei Manhattan.

Pentru fiecare problemă trebuie căutată masura de distanţă care se potriveşte cel mai bine în funcţie de proprietăţile datelor cu care se lucrează. Atunci când nu se cunoaşte cea mai bună măsură de distanţă, se pot experimenta mai multe metrici de distanţă cu diferite valori pentru K şi se va alege modelul care va avea cea mai bună acurateţe.

Spre exemplu, distanţa Euclidiană este o metrică bună atunci când instanţa de intrare are atribute numerice la o scală asemănătoare (ex.lungimea şi lăţimea dreptunghiurilor). Distanţa Manhattan este o metrică bună pentru cazul în care variabila de intrare nu are atribute de acelaşi tip (ex.vârsta, sexul, greutatea unei persoane).

Valoarea cea mai bună pentru K poate fi găsită prin încercarea mai multor valori şi selectarea celei mai bune. Este o buna idee sa se foloseasca valori mai mici decat 21.

Complexitatea algoritmului KNN creşte pe măsură ce cresc şi datele de antrenare. Pentru seturi de date foarte mari, KNN va alege random un eşantion din care va calcula cei mai apropiaţi K vecini.

Când se foloseşte KNN pentru clasificare, predicţia va fi calculată în funcţie de clasa care are cele mai multe apariţii în cei mai apropiaţi K vecini. Fiecare instanţă, în esenţă, votează pentru clasa căreia îi aparţine şi clasa cu cele mai multe voturi câştigă.

Se poate calcula şi probabilitatea cu care o instanţă aparţine unei clase. Spre exemplu, pentru o problemă de clasificare binară: P(bărbat) = Numărul de instanţe aproapiate de tipul bărbat raportat la numărul total de vecini, K.

Daca există un număr par de clase, atunci este o bună idee ca valoarea lui K să fie impară, pentru a evita situaţiile în care două clase pot avea acelaşi număr de voturi, situaţie în care apare un caracter random în algoritm. Iar dacă avem un număr impar de clase, este o buna idee ca numărul de vecini pe care-i vom lua in considerare (K) sa fie un numar par.

### 5.3.1 Pre-procesarea datelor pentru KNN

* Re-scalarea datelor: KNN funcţionează bine dacă toate datele sunt la aceeaşi scală. Se recomandă ca datele să se normalizeze într-un interval cuprins între 0 şi 1.
* Ştergerea instanţelor care nu au toate atributele: dacă lipsesc atribute, înseamnă că nu se poate calcula similaritatea cu noile instanţe. Acest lucru înseamnă că aceste date oricum nu pot fi folosite şi este natural, să le scoatem din setul de date.
* Numarul de dimensiuni: KNN se pliază pentru problemele în care datele sunt dispunse într-un spaţiu cu dimensiuni cât mai puţine. Se poate aplica şi pe mai multe dimensiuni (de ordinul sutelor de atribute), dar nu va avea aceeaşi acurateţe ca alte tehnici de clasificare. Pentru KNN se poate aplica o reducere de dimensiuni asupra atributelor de intrare.

### 5.3.2 Pseudocod KNN

Parametri: Un set de date de antrenare S=(x1, y1), ..., (xn, yn), o instanţă de testare I

function kNN(S, I)

* 1. Iterarea prin fiecare item din setul de date si calcularea distantei de la acesta la instanta de testare
  2. Clasifica noua instanta cu clasa majoritara dintre cei mai apropiati vecini ai sai din setul de date

end function

## 4.4 Regresia Logistică

Regresia logistică este o tehnică împrumutată de învățarea automată din statistică. Această metodă se aplică, în special, în cazul problemelor de clasificare binară. Numele vine de la funcția nucleu care stă in centrul acestei metode: funcția logistică.   
 Funcția logistică, numită și funcția sigmoid, a fost creată de statisticieni pentru a descrie proprietățile creșterii populației într-un anumit mediu. Această funcție poate lua ca argument orice număr real şi îi atribuie o valoarea între 0 și 1, dar niciodată exact aceste limite.

Regresia logistică foloseşte o ecuaţie ca model al datelor din acest motiv se aseamană foarte mult cu regresia liniară. Pentru fiecare valoare de intrare (X) se aplică o combinaţie liniară folosind diferite ponderi (coeficienţi) pentru a se prezice o valoare de ieşire (y). Diferenţa faţă de regresie liniară este că valoarea de ieşire este binară, luând valori din mulţimea {0, 1} şi nu valori numerice.

Următoarea ecuaţie reprezintă un exemplu pentru acest mod de clasificare:

Unde y este rezultatul, iar sunt coeficienţii pentru atributele valorii de intrare (X). Dacă datele de intrare sunt formate din mai multe atribute (coloane), atunci pentru fiecare atribut vom avea asociat câte un coeficient b( o valoare reală, constantă) care trebuie învăţat din datele de antrenare.

Reprezentarea modelului pe care trebuie să o memorăm sunt doar coeficienţii ecuaţiei, adică valorile corespunzăoare pentru fiecare b.

### 4.4.1 Calcularea probabilităților folosind regresia logistică

Regresia logistică modelează probabilitatea unei instanţe de a aparţine unei categorii (ex. Prima categorie). De exemplu, dacă am modela sexul oamenilor (bărbaţi/ femei), în funcţie de înălţimea lor, atunci prima categorie poate fi *bărbat,* iar modelul regresiei logistice poate fi scris sub forma de probabilitate: probabilitatea ca dându-se înălţimea unei persoane, aceasta să fie fie bărbat. Mai formal:

.

Generalizând putem spune că regresia logistică modelează probabilitatea ca dându-se o instanţă de intratre (X), aceasta să aparţină unei clase prestabilite (Y=1). Formal putem scrie în felul urmator:

Regresia logistică este o metodă liniară, dar predicţia este transformată de functţa logistică. Din acest motiv nu mai putem să privim rezultatul predicţiei ca o combinaţie liniară a datelor de intrare, aşa cum se întâmplă în cazul regresiei liniare. Aplicând funcţia logistică pe probabilitatea anterioară, putem privi modelul obţinut în felul următor:

În urma unor transformări imediate, se poate observa că:

Logaritmând ajungem la:

Această formulă este folositoare, deoarece se pot face calculele din partea dreaptă în mod liniar (în acelaşi mod ca la regresia liniară), iar rezultatul, din partea stângă, poate fi privit ca logaritm din probabilitatea ca instanta X sa apartina primei categorii.

### 4.4.2 Învăţarea modelului de regresie logistică

Coeficienţii algoritmului de regresie logistică trebuie să fie estimaţi din datele de antrenare. Această estimare se realizează folosind o metodă numită “estimarea probabilităţii maxime” (**M**aximum **L**ikelihood **E**stimation) - MLE.

Această tehnică de estimare a coeficienţilor este comună în algoritmii de învăţare automată, chiar dacă face anumite presupuneri despre distribuţia datelor. Cei mai buni coeficienţi vor determina un model care va face o predicţie foarte aproape de 1 (persoana este bărbat) pentru categoria de bază şi o valoare foarte aproape de 0 (persoana este femeie) pentru cealaltă categorie. În mod intuitiv această tehnică de estimare, pentru cazul regresiei logistice poate fi privit ca o metodă de a căuta cei mai buni coeficienţi astfel încât eroarea (discordanţa dintre categoria prezisă şi categoria reală) să fie cât mai mică.

### 4.4.3 Clasificarea cu regresia logistică

Predicţia categoriei cu regresia logistică este simplă. Să presupunem că avem un model care face predicţia dacă o persoană este bărbat sau femeie, pe baza înălţimii.

Dându-se o persoană de 150 de cm să se determine dacă aceasta este bărbat sau femeie.  
Să presupuem ca am învăţat deja coeficienţii şi aceştia sunt şi. Folosind ecuaţia de mai sus putem calcula probabilitatea ca o persoană de 150 de cm să fie bărbat:

Deci probabilitatea este aproape 0 ca această persoană să fie bărbat. În practică, putem folosi direct probabilitatilea şi vom face clasificarea în felul următor: Dacă probabilitatea este mai mică decât 0.5, atunci acea persoană este catalogată ca fiind bărbat, altfel va fi catalogată femeie.

# 5Gradient Boosting

Gradient Boosting face parte din categoria celor mai puternici tehnici de construire a modelelor din învățarea automată. Ideea unui algoritm de tip boosting (de îmbunătățire) este că un clasificator slab poate fi modificat pentru a deveni mai bun.

Un clasificator slab este definit ca fiind un clasificator a cărui performanță este puțin mai bună decât a unei clasificari aleatoare. Ideea din spatele acestui tip de algoritm este de a filtra instanțele de intrare, lăsându-le doar pe cele pe care clasificatorul slab să se concentreze și să se antreneze pentru a le clasifica corect și astfel să devenă un clasificator mai bun.

*“Ideea este de a folosi o metodă de învățare slabă de câteva ori pentru a reusi să satisfacem ipoteza, de fiecare dată concentrându-ne atenția pe acele instanțe pe care în etapa anterioară le-am clasificat greșit.”* [[10]](#endnote-10)

## 5.1 AdaBoost

Prima implementare a unui alogritm de tip boosting care a avut succes în aplicații practice este Boostingul Adaptiv sau AdaBoost pe scurt.

*“Boosting se referă la problema generală de a produce un model cu acuratețe foarte mare combinând mai multe modele cu acuratețe mai mică.*”[[11]](#endnote-11)

Clasificatorii slabi folosiți de AdaBoost sunt arbori de decizie cu o singură divizare. AdaBoost acordă diferite ponderi instanțelor de intrare. Instanțele care sunt mai greu de clasificat vor avea o pondere mai mare, iar cele care sunt deja clasificate corect vor avea o pondere mai mică. Clasificatorii noi sunt antrenați să se adapteze concentrându-se, în mod special, în clasificarea corectă a acelori instanțe care au ponderi mai mari, adică a acelor instanțe care nu au fost clasificate corect până acum.

*“Acest lucru înseamnă că instanțele care sunt greu de clasificat corect vor primi o pondere mai mare până când modelul creat va reusi să le clasifice corect.”*[[12]](#endnote-12)

Predicția finală se face pe baza majorității voturilor a acestor clasificatori slabi, fiecare având câte o pondere individuală direct proporțională cu acuratețea lor individuală. Varianta de implementare a alogritmului AdaBoost cu cel mai mare succes pentru problema clasificării binare a fost numit AdaBoost.M1.

## 5.2 Gradient Boosting

AdaBoost și ceilalți algoritmi au fost inspirați dintr-o abordare statistică numită inițial ARCing.

*“ARCing este un acronim pentru* ***R****eponderarea* ***A****daptivă și* ***C****ombinatorială. Fiecare iterație constă în a minimiza ponderile urmată de o nouă recalculare atât a clasificatorilor cât și a ponderilor.”* [[13]](#endnote-13)

Această tehnică a fost dezvoltată mai departe de către Friedman și a fost numită inițial de acesta “Gradient Boosting Machines”. Astfel a apărut o nouă soluție de optimizare numerică ce miniminează eroarea modelului prin adăugarea clasificatorilor slabi, pe care va încerca apoi să-i îmbunătățească. Generalizarea făcută de Friedman a permis utilizarea diferitelor funcții de calculare a eroarii și extinderea acestei tehnici și la alte probleme pe lângă clasificarea binară: problema regresiei, problema clasificării în mai multe categorii, dar și în multe alte domenii.

Atunci când vorbim de algoritmul de Gradient Boosting trebuie să vorbim de cel puțin 3 elemente:

* 1. O funcție ce calculează eroarea;
  2. Un clasificator slab care să facă predicții;
  3. O modalitate de a îmbunătăți modelul prin antrenarea clasificatorului să minimizeze eroarea;

### 5.2.1 Funcția de eroare

Această funcție depinde foarte mult de ce fel de problemă trebuie să fie rezolvată. În primul rând trebuie să fie derivabilă, pentru a o putea optimiza mai ușor. Există foarte multe astfel de funcții considerate standard, dar există și posibilitatea de a defini unele noi, atunci când problemele întâlite în practică impun acest lucru. Spre exemplu în problema regresiei cea mai folosită funcție este pătratul erorii, iar în problema clasificării se folosește funcția logaritm.

### 5.2.2 Clasificatorii slabi

Arborii de decizie sunt folosiţi ca şi clasificatori slabi de către gradient boosting. Iniţial, în cazul algoritmului AdaBoost, erau construiti arbori de decizie având o adâncime foarte mică de cele mai multe ori fiind cu doar o singură divizare. Gradient Boosting foloseşte, în general de la 4 până la 8 niveluri de adâncime.

În mod uzual, pentru a constrânge arborii de decizie să rămână slabi, se poate specifica un număr maxim de niveluri, un număr maxim de noduri de decizie sau un număr maxim de frunze. Dar aceştia încă sunt construiţi în manieră greedy.

### 5.2.3 Îmbunătăţirea modelului

Arborii sunt adăugaţi câte unul la fiecare iteraţie. O procedură asemănătoare cu cea a gradientului descendent se foloseşte pentru a minimiza eroarea atunci când se adaugă noul arbore. În mod tradiţional, gradientul descendent este folosit pentru a minimiza un set de parametri precum coeficienţii unei ecuaţii de regresie sau ponderile într-o reţea neuronală. După calcularea erorii ponderile sunt actualizate, iar eroarea se miniminează.

Noul arbore creat se adaugă la secvenţa de arbori deja existenţi într-un efort de a corecta sau de a îmbunătăţi rezultatul final al modelului.

Algoritmul se opreşte fie atunci când a fost atins un anumit număr de arbori, fie atunci când eroarea a atins un punct acceptabil sau atunci când nu se mai pot face îmbunătăţiri.

# 6 Detalii de implementare

## 6.1 Limbajul de programare

Dintotdeauna a fost o provocare alegerea unui limbaj de programare care să fie capabil să le surclaseze pe celelalte, mai ales, când vine vorba de *Data Science*. Dificultatea este și mai accentuată atunci când o persoană nouă dorește să se specializeze în acest domeniu și nu știe ce limbaj de programare să aprofundeze.

Limbajele de programare precum *R*, *Python*, *Octave*, *Matlab*, *Julia*, etc. oferă câte o serie unică de capabilități ce au drept scop reducerea complexității de implementare a operațiilor de analiză a datelor comparativ cu limbajele de programare tradiționale ca *Java*, *C++*, *C* etc.

*Siva Prasad Katru*, architect la institutul de tehnologie din India, a creat o clasificare a limbajelor de programare în funcție de diferite metrici, scopul fiind acela de a-l identifica pe cel care s-a impus în domeniul învățării automate.

Pentru a ajunge la o concluzie obiectivă, a creat o scală de la 0 la 5, notând limbajele cu câte o notă pentru fiecare caracteristică din cele alese:

* Viteza de execuție;
* Dificultatea învățării;
* Diversitatea uneltelor oferite pentru analiza datelor;
* Metode de vizualizare;
* Unelte de dezvoltare (IDE etc.);
* Dificultatea integrării unei noi aplicaţii noi cu una deja existentă;
* Oportunităţile locurilor de muncă de pe piaţă;

Adunând notele fiecărei caracteristici a obținut o notă după care a făcut clasificarea finală.

Analizând rezultatele obținute din anexa 1, el trage fără reținere următoarea concluzie “*Python* conduce detașat, dar *R* vine din urmă destul de puternic”.

## 6.2 Biblioteca scikit-learn

Din multitudinea de utilitare și biblioteci de analiză avansată a datelor (*Spark* -cu biblioteca *MLLib*-, *R*, *Scikit-learn*, *GraphLab* etc.), am optat pentru biblioteca Scikit-learn deoarece este ușor de folosit, este foarte bine documentată și conține implementări a algoritmilor populari din învățarea automată.

### 6.2.1 Documentația

Principalul motiv pentru care am ales să folosesc biblioteca scikit-learn a fost datorită documentației foarte bine organizată. Persoanele care doresc să contribuie la acest proiect open-source sunt nevoite să scrie și documentația aferentă însoțită obligatoriu și de exemple de script-uri care să ruleze pe seturi de date mici, ca exemplu de funcționare. Pe lângă documentație, comunitatea de persoane care s-a format este consecventă, iar contribuțiile noi sunt folositoare și de calitate. De asemenea, ei sunt încurajați să dezvolte, în permanență, noi teste pentru asigurarea calității tuturor funcționalităților disponibile.

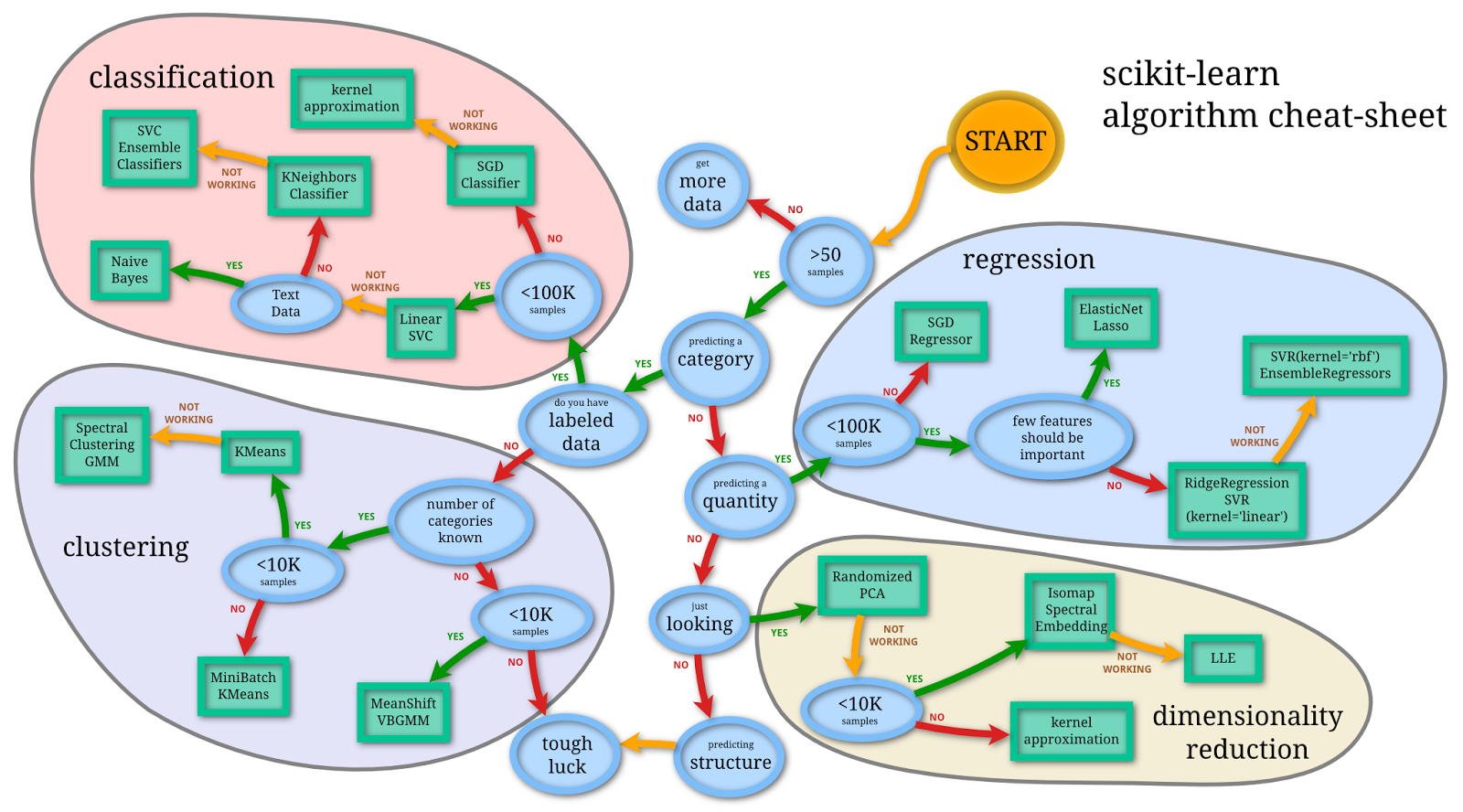
### 6.2.2 Expertiza contribuitorilor

Comunitatea de persoane care contribuie la scikit-learn include și experți din mai multe domenii, printre care și cel al învățării automate și cel al dezvoltării de software. Un mic grup dintre ei dedică o parte din timpul lor profesionl pentru a ajuta la dezvoltarea acestui proiect.

### 6.2.3 Variația domeniilor acoperite

Lista cu uneltele disponibile în scikit-learn cuprinde majoritatea domeniilor din învățarea automată (precum clasterizare, clasificare, regresie, etc). Iar din moment ce scikit-learn este dezvoltat de o comunitate numeroasă, este foarte posibil ca tehnicile oferite să se înmulțească în urmatoarea perioadă.

Utilizatorii nu sunt nevoiți să aleagă dintre multiple implementări ale aceluiași algoritm (o problemă cu care se confruntă utilizatorii limbajului R). Pentru a-i ajuta pe utilizatori să găsească modelul care se potrivește cel mai bine cu problema lor, Andreas Muller a creat următoarea diagramă:



Imaginea este preluată de pe al doilea diapozitiv de pe prezentarea pe care a susținut-o Andreas Muller la “SciPy 2016 Conference talk”

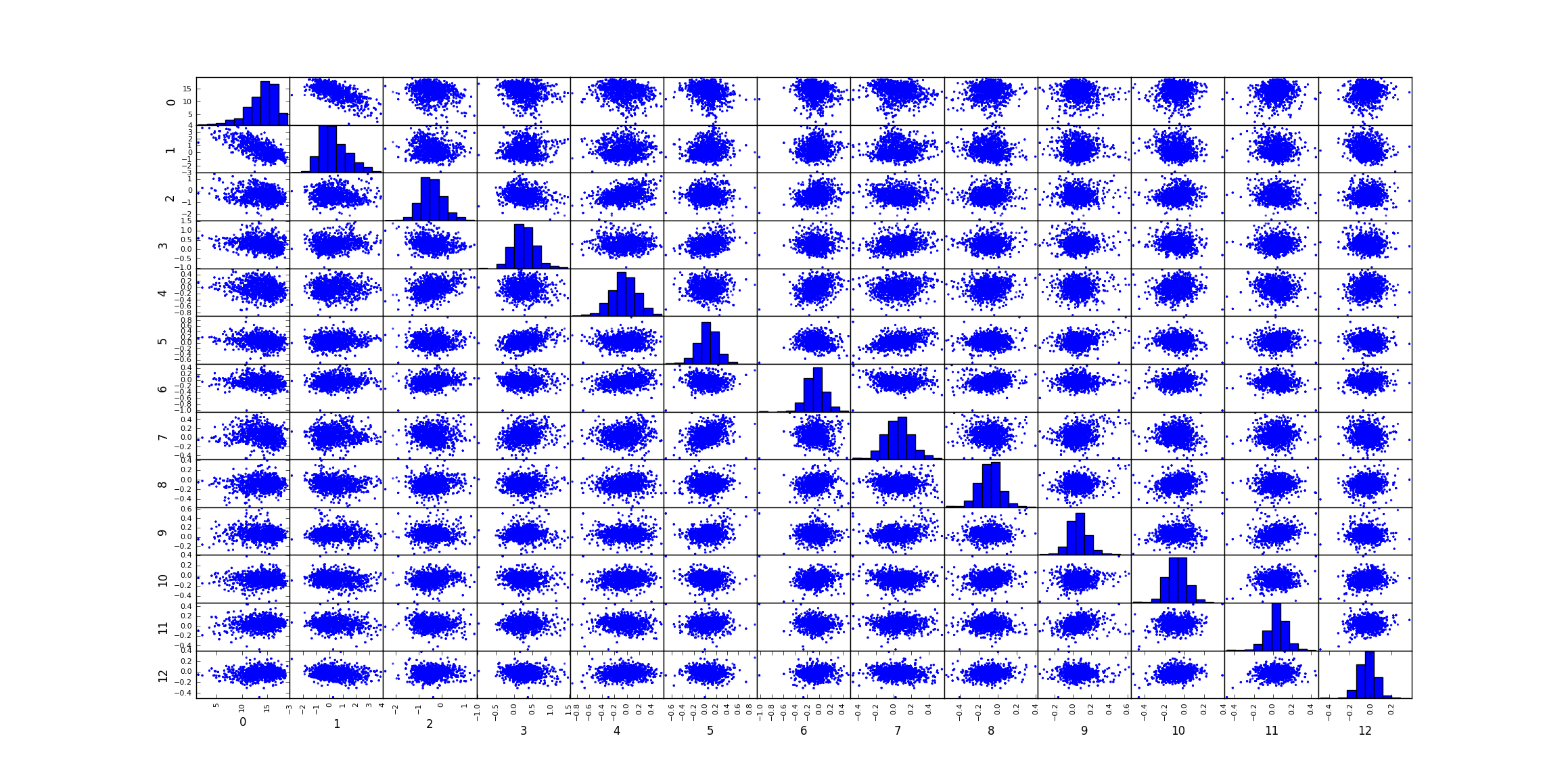
### 6.2.4 Simplitate

Scikit-learn este o bibliotecă de învățare automată. Scopul ei este de a oferi un set de algoritmi obișnuiti utilizatorilor de python, printr-o interfață consistentă. Acest lucru aduce cu sine alegeri dificile care trebuie făcute pentru a discerne ce noi funcționalități pot fi adăugate. De exemplu, comunitatea a observat că algoritmii de Deep - Learning au cateva dependente speciale care ar fi trebuit incluse. Dar pentru a nu complica lucrurile s-a decis renunțarea la includerea lor și implementarea, în schimb, a aunui algortim de tip Multilayer Perceptron.

# 7 Metodologia de clasificare a melodiilor

Problema clasificării automate a melodiilor în genuri diferite poate fi descrisă ca sarcina de a atribui pentru fiecare melodie câte un gen muzical. În acest proces de clasificare se crează un model care să aproximeze cât mai bine datele de antrenare.

În abordarea pe care o vom prezenta, crearea modelului cuprinde mai mulţi paşi. În primul rând, se vor extrage atributele din melodie conform paşilor explicaţi la capitolul corespunzător. În învăţarea automată este importantă studierea datelor de intrare, în special este necesar ca acestea sa nu fie corelate. Sunt algoritmi care sunt sensibili la covariaţia a două sau a mai multor atribute de aceea trebuie să ne asigurăm că atributele extrase sunt reprezentative pentru piesele muzicale. Pentru acest lucru am generat mai multe grafice pe care le-am unit în imaginea de mai jos.



În această imagine putem observa pe diagonală histogramele pentru fiecare din cele 13 atribute. Observăm cum aceste atribute sunt normal distribuite și au valori variate. De asemenea, în restul poziţiilor din imagine avem o serie de puncte ce reprezintă atributele de intrare. Spre exemplu pe prima coloană avem o serie de grafice în care fiecare punct are ca valoare pe abscisă primul atribut, iar pe ordonată, pe rând, fiecare din celelalte atribute.

Analizând aceste grafice putem concluziona că nu există două atribute care să fie dependente. Dacă ar fi existat cel puţin două atribute care să fie covariate atunci graficele ar fi avut forme alungite, eventual în direcţia fie a diagonalei principale, fie în direcţia diagonalei secundare.

După acest pas vom avea setul de date pregătit pentru procesările ulterioare. Acesta constă în 1000 de instanţe de intrare, fiecare având câte 13 atribute, reprezentând media energiilor ce apar pe durata melodiilor grupate în 13 grupe, în funcţie de frecvenţa lor. Prima categorie, cea de antrenare va fi folosită pentru antrenarea simultană şi independentă a 6 clasificatori:

1. Regresie Logistică,
2. Reţele Neuronale,
3. Maşini pe Vector Suport (SVM),
4. Gradient Boosting,
5. Random Forest,
6. KNN.

## 7.1 Modele individuale

Prima abordare a fost aceea de a găsi un singur clasificator, cel mai bun, care să obţină cea mai mare acurateţe. S-au încercat, pe rând, mai mulţi clasificatori, fiecare cu câte o acurateţe proprie. Unii clasificatori s-au pliat pe această problemă, având o acurateţe destul de bună, alţii nu s-au potrivit la fel de bine.

În anexa 2 am generat pentru fiecare clasificator, câte 10 grafice ROC. Fiecare grafic reprezintă acurateţea a câte un model individual în a clasifica un anumit gen în comparaţie cu celelalte. Aceste grafice sunt grupate pe genuri pentru a se face mai uşor comparaţia dintre aceste modele.

### 7.1.1 Regresie Logistică

Primul clasificator încercat a fost Regresia Logistică. Această tehnica de clasificare este folosită, în mod special, în cazul problemelor de clasificare binară. În cazul de faţă, problema cere o clasificare în 10 categorii diferite. De aceea s-a aplicat o tehnica numită în învăţarea automată One - Against -All (Unul – împotriva - Tuturor).

#### 7.1.1.1 One-Against-All (OAA)

Această strategie constă în a antrena câte un clasificator pentru fiecare categorie, specializându-se să facă distincţia între instanţele care aparţin unei singure categorii şi cele care nu aparţin acestei categorii. Fiecare clasificator va returna probabilitatea ca noua instanţă să aparţină clasei pe care s-a specializat el. Iar rezultatul final, va fi genul muzical cu probabilitatea cea mai mare ca noua instanţă să-i aparţină, conform tuturor modelelor.

Spre exemplu dacă am presupune că am vrea să facem o clasificare în 3 categorii de obiecte şi că avem urmatoarele instanţe: triunghiuri, ovale şi dreptunghiuri.

Vor fi create 3 modele, fiecare specializat în a recunoaşte câte o categorie: triunghi, oval, respreciv dreptunghi. Primul clasificator va avea ca rezultat probabilitatea ca o nouă instanţă să aparţinaă categoriei triunghi. Al doilea, va întoarce probabilitatea ca aceeaşi instanţă să aparţină categoriei oval, iar al treilea va returna probabilitatea de a fi dreptunghi. Rezultatul final, al întregii clasificări va fi dat de categoria care are probabilitatea cea mai mare.

P(X= )

P(X= )

P(X= )

Clasificare finala:

Categoria cu probabilitatea cea mai mare



Deoarece, în problema clasificării melodiilor avem 10 genuri muzicale, vom avea 10 modele create cu ajutorul regresiei logistice. Dar rezultatul final va fi, un singur gen muzical. Acestă abordare nu modelează datele nici foarte bine dar nici extraordinar de rău, având o acurateţe de 46%. Spunem că această acurateţe nu este foarte rea în condiţiile în care, probabiliatatea de a alege în mod aleator genul unei noi instanţe şi ca aceasta să fie varianta corectă este doar de 10%. Deci regresia logistică aplicată pe problema clasificării melodiilor în genuri muzicale, este de 4.6 ori mai bună decât un clasificator aleator.

### 7.1.2 Rețele Neuronale Artificiale

Al doilea clasificator încercat este un model creat cu ajutorul reţelelor neuronale. Primul strat din reţeaua creată are 13 neuroni, deoarece datele cu care lucrăm au câte 13 atribute. Reteaua va area un singur strat ascuns, ce va fi format din 100 de neuroni. Iar ultimul strat este format din 10 neuroni, fiecare având câte o probabilitate asociată pentru fiecare gen muzical.

x1

x2

x12

x13

Input

X

y1

y2

Y10

max



Ultimul strat are ca funcţie de activare funcţia softmax. Aceasta este o generalizare a funcţiei logistice şi se foloseşte de obicei în cazul problemelor de clasificare ce au mai mult de două categorii. Această funcţie va avea ca valoare de intrare un vector *z* 10-dimensional şi ca ieşire va returna tot un vector cu 10 valori *y* cu valori între 0 şi 1. Vectorul de ieşire este construit în aşa fel încât suma valorilor lui să fie 1, adică valorile din vectorul *y* reprezintă, de fapt, probabilităţile de aparteneţă a unei instanţe la o anumita categorie.

Funcţia softmax se poate defini în felul următor:

Numitorul este termenul ce se asigură că suma tuturor valor din vectorul de ieşire sunt 1 ( ). Din punct de vedere grafic funcţia softmax poate fi reprezentată ca un strat cu 10 neuroni. Putem scrie probabilităţile categoriilor t=c pentru c = 1...10 dându-se valoare de input z în felul următor:

Unde este probabilitatea ca dându-se valoarea de intrare z ea să reprezinte o instanţă din clasa c.

Rezultatul final al reţelei neuronale va fi categoria care are probabilitatea cea mai mare din cele 10.

### 7.1.3 Gradient Boosting

Acest clasificator este unul de tip asamblist ce crează mai mulţi clasificatori slabi, iar apoi încearcă prin diverse tehnici să-i asambleze şi să creeze alţii noi pentru a rezulta un model cât mai veridic. Ca şi clasificator este ales modelul creat de arborii de decizie. Cel mai bun rezultat a fost obţinut folosind 100 de astfel de clasificatori. Chiar dacă iniţial nu s-a luat în calcul, a trebuit să revenim asupra acestor clasificatori pentru a-i forţa să rămână slabi. Pentru acest lucru am pus două condiţii suplimentare şi anume: fiecare arbore de decizie poate avea maxim 8 frunze şi poate fi divizat de maxim 5 ori. Acuratetea acestui clasificator este una din cele mai bune, alături de cea obţinută de Random Forest, şi anume de 76.3%.

### 7.1.4 Random Forest

Următorul clasificator folosit, pe care îl vom prezenta este Random Forest. Acesta s-a potrivit cel mai bine pe această problemă şi a oferit cele mai bune rezultate individuale. El a asamblat 100 de arbori de decizie. Pentru acest algoritm, arborii de decizie utilizaţi nu au fost constrânşi cu nimic, spre deosebire de cei folosiţi de Gradient Boosting. Acurateţea acestuia a fost de 77.975% .

### 7.1.5 KNN

KNN este unul din puţinii algoritmi care pot să clasifice, prin natura algoritmului toate cele 10 genuri simultan. Ceilalţi algoritmi recurg la diverse metode de descompunere a problemei în subprobleme şi apoi la asamblarea lor. Prin încercări repetate, s-a ajuns la concluzia că cele mai bune rezultate pentru acest clasificator sunt obţinute atunci când luăm în considerare cei mai apropiaţi 5 vecini. Acurateţea este comparabilă cu cea a regresiei logistice sau a reţelelor neuronale şi anume de 53.225%.

## 7.2 Asamblarea modelelor individuale

Setul de date folosit constă în 1000 de instanţe de intrare, fiecare având câte 13 atribute, reprezentând suma energiilor ce apar pe durata melodiilor grupate în 13 grupe, în funcţie de frecvenţa lor. După ce avem setul de date pregătit, acesta este împărţit, pentru început, în 2 sub-categorii: date de antrenare (A1) şi date de testare (T1), în proporţie de 60% resprectiv 40%. Prima categorie (A1), cea de antrenare va fi folosită pentru antrenarea simultană a cei 6 clasificatori.

Fiecare din aceşti clasificatori vor fi antrenaţi pe 600 de instanţe, fiecare având câte 13 atribute. După faza de antrenare, fiecare clasificator urmează să fie testat. Testarea se face folosind (T1).

În această fază de testare se generează câte o matrice de confuzie pentru fiecare clasificator. Aceste matrici pot fi găsite în Anexa 3. Pentru o matrice de confuzie abscisa va reprezenta rata clasificarilor false făcute, adică câte melodii au fost încadrate într-un anumit gen, cu toate că ele aparţineau altuia. Iar pe ordonată, matricea va avea reprezentată rata clasificărilor corecte.

După cum se poate observa, din subcapitolul anterior fiecare clasificator individual, produce câte 10 probabilităţi, câte una pentru fiecare gen. Din aceste 10 probabilităţi clasificatorul o va lua pe cea mai mare şi predicţia sa va fi categoria care corespunde acestei probabilităţi.

Pe lângă această fază de testare, pentru fiecare din aceleaşi 240 de instanţe se memorează probabilităţile fiecărei instanţe de a aparţine la cele 10 genuri. Rezultaul întors de fiecare din aceşti clasificatori este reprezentat de un vector de 10 valori reale, ce se vor concatena formând într-un final un vector de 60 de valori. Aceste 60 de valori vor fi valorile de intrare pentru un alt clasificator de tip Random Forest. Acest clasificator se consideră a fi un clasificator expert pentru că el va atribui diferite ponderi clasificărilor individuale. Pentru a reuşi să-l atrenăm şi să-l testăm corect din punct de vedere statistic vom împărţi datele de testare (T1) încă o dată în două, în aceeaşi proporţie: 60% (TA2) şi 40% (TT2). Doar că aceste date nu vor conţine cele 13 atribute iniţiale, ci vor conţine 60 de atribute: câte 10 probabilităţi de la fiecare clasificator. Cu ajutorul setului de date TA2, care conţine 240 de melodii, se antrenează clasificatorul expert, iar cu setul de date TT2 se va testa clasificatorul expert. Acurateţea acestui clasificator s-a dovedit a fi cu mult mai bună decât a celorlalţi clasificatori. Aceasta este de 88%.



Clasificator 1

y11

y21

Y101

Clasificator 6

y16

y26

Y106

Random Forest

# 8.Rezultate obţinute

În ciuda naturii foarte complexe a problemei, melodiile pot fi clasificate în mod automat. Iar rezultatele acestui proces automat sunt cu mult mai bune decât clasificarea făcută manual de oameni. Trăsăturile folosite sunt cele propuse de G. Tzanetakis si P. Coo şi poartă numele de MFCC. Antrenând câte un model pentru fiecare din cei 6 clasificatori (Regresie Logistică, Reţele Neuronale, Maşini pe Vector Suport (SVM), Gradient Boosting, Random Forest, KNN) şi analizând rezultatele fiecăruia s-a ajuns la o acurateţe individuală de 77.975%. Acest rezultat aparţine modelului creat cu algoritmul Random Forest.

Apoi, în încearcarea de a obţine o acurateţe mai bună, s-a mai folosit un clasificator care are rolul de a acorda diferite ponderi celorlalţi algoritmi aminiţi deja. Această tehnică este cunoscută ca folosirea unui clasificator a experţilor. S-a dovedit că această abordare a dus la obţinerea unei acuraţi de 88.275%.

Pentru a vizualiza modul în care se face clasificarea melodiilor am încercat combinarea a catorva tehnici din învăţarea automată. În primul rând, pentru a reprezenta datele de intrare este nevoie de un spaţiu în 13 dimensiuni. Cum o astfel de reprezentare nu se poate pune în practica am aplicat, pe datele de intrare, un algoritm de reducere a dimensiunilor şi anume PCA (**P**rincipal **C**omponent **A**nalysis). Astfel s-a ajuns de la 13 dimensiuni, la doar două. Dar pentru a putea vedea concret fiecare instanţă cum a fost clasificată trebuie să se folosească toate cele 13 atrbiute, altfel clasificarea nu este validă. Acestea au fost clasificate folosindu-se toate cele 13 atribute, iar apoi coordonatelor bidimensionale le-a fost atribuit genul cu care a fost clasificată o anumită melodie. Aceste imagini pot fi vizualizate în Anexa 4.

# Concluzii şi direcţii de continuare a acestui studiu

În această lucrare am implementat un sistem software capabil să identifice genul muzical al pieselor muzicale. Acest sistem poate fi îmbunătăţit prin adăugarea de noi atribute cu ajutorul cărora să se facă diferenţa dintre genuri mai bine. Cel mai bun model individual a fost Random Forest, de aceea el a fost ales mai apoi ca şi clasificator expert.

De asemenea, pentru obţinerea unor rezultate care să reflecte cât mai bine realitatea, ar trebui să se facă un studiu comparativ între mai multe seturi de date deoarece în practică, o piesă muzicală, un fişier audio sau un conţinut multimedia audio poate avea o multitudine de forme, cu multe variante şi pentru a vedea cât de bun este un model în practică, ar trebui studiate mai multe seturi de date.

Iar după aceste studii, cred că ar fi foarte benefică oferirea acestei capabilităţi ca şi un serviciu web, accesibil de oricine are nevoie. O astfel de clasificare în timp real ar putea fi folosită atât pentru recunoaşterea genului muzical al unei melodii cât şi pentru recomandarea de melodii pentru utilizatori. Aşa cum au concluzionat A.C.North și David J Hargreaves în studiul lor 1 oamenii asociază mai repede melodiile care aparţin aceluiaşi gen muzical, deci poate fi o îmbunătăţire a modelelor actuale de recomandare a melodiilor.

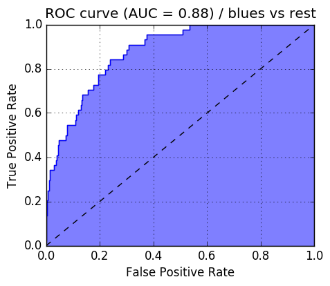
|  |  |  |  |
| --- | --- | --- | --- |
| Limbajul de programare | Avantaje | Dezavantaje | Biblioteci importante |
| R | 1.Open source  2.Bun pentru analiză statistică și procesarea datelor  3. O colecție impresionante de algoritmi | 1. Destul de greu de învățat  2. Comenzi neobișnuite | 1. Gbm  2. RTextTools  3. Dplyr, zoo,  4. Ggplot2,  5. Carot |
| Python | 1. Open source  2. Ușor de învățat  3. Păstrează avantajele unui limbaje de programare general  4. Poate fi folosit pentru BigData | 1.Viteza de execuție  2.Trebuie să se țină cont dacă bibliotecile sunt portate de la versiunea 2.x la 3.x | 1.Scikit-learn  2.Pandas,  3.Matplotlib  4.Numpy  5.Sciy  6.Theano,  7.Nltk |
| MATLAB | Potrivit pentru:  1. procese complexe matematice precum operații cu matrici  2. învățare automată  3. procesarea semnalului audio  4. procesare de imagini | 1. Lipsește un ecosistem open source  2. Există dificultăți atunci când datele nu pot fi reprezentate sub forma de matrice | 1.Statistică și învățare automată  2.Procesare de imagini  3.Optimizare |
| OCTAVE | 1.Open source  2.Potrivit pentru operații numerice  3.Este compatibil cu MATLAB  3.Bun pentru a construi modele preliminare | 1. Inteoperabilitatea cu date externe (baze de date, fișiere csv etc) este destul de slabă | 1.Libsvm,  2.Shogun,  3.Liblinear,  4.Ltfat,  5.Vlfeat |
| Julia | 1.Open source  2.Proiectat pentru calcule numerice și științifice  3.Performanță foarte bună  4.Poate apela funcții Python și C | 1.Este un limbaj nou de programare  2. Nu exista foarte multe biblioteci externe | MLBase  MLUtils  MLKernels,  Clustering  MachineLEarning |

# Anexa 1

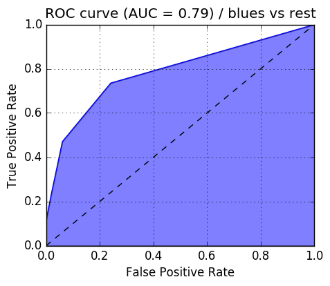
Aceste tabele sunt o traducere a unei analize făcute de Siva Prasad Katru. Preluată de pe https://www.linkedin.com/pulse/r-vs-python-matlab-octave-julia-who-winner-siva-prasad-katru

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Limbajul de programare | Viteza de executie | Dificultatea invatarii | Capabitilati de analiza a datelor | Capabilitati grafice | Utilitati  (IDE, plugins) | Suportul comunitatii | Integrarea cu aplicatiile existente | Joburi disponibile | Scor total |
| R | 3 | 1 | 5 | 4 | 4 | 4 | 3 | 4 | 28 |
| Python | 4 | 4 | 3 | 3 | 3 | 5 | 5 | 5 | 32 |
| MATLAB | 2 | 3 | 4 | 4 | 5 | 3 | 2 | 2 | 25 |
| OCTAVE | 2 | 2 | 3 | 2 | 2 | 3 | 2 | 1 | 17 |
| Julia | 5 | 3 | 4 | 2 | 2 | 2 | 3 | 1 | 22 |

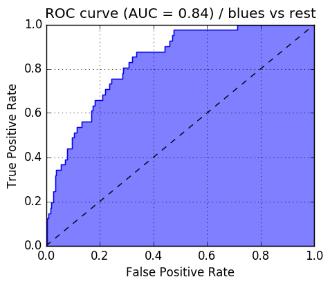
# Anexa 2



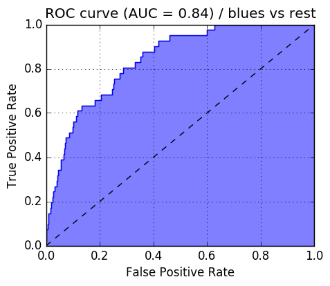
Gradient Boosting



KNN



Rețele Neuronale Artificiale

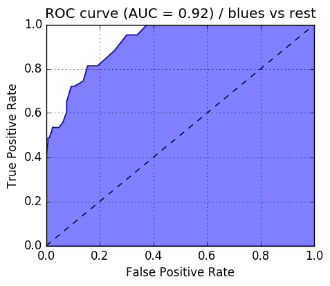


Regresie Logistică

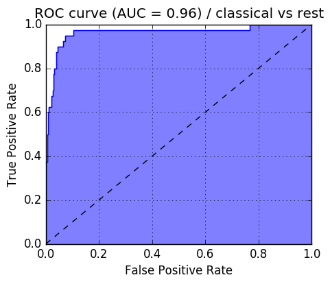
## Blues VS. Rest



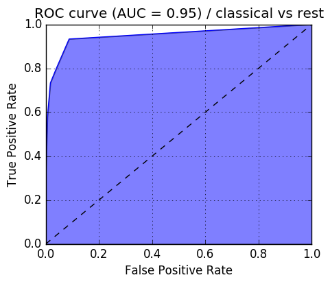
SVM



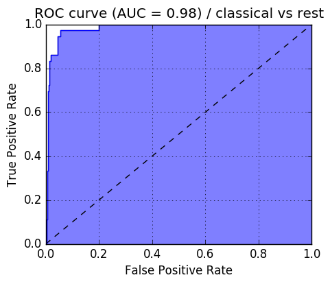
Random Forest



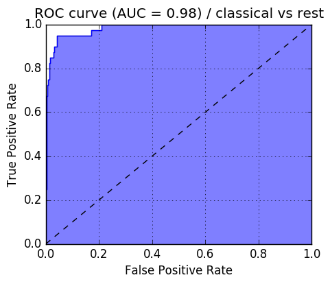
Gradient Boosting



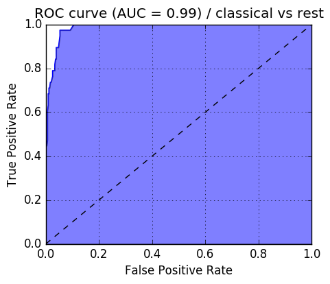
KNN



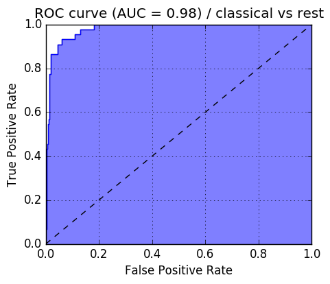
Regresie Logistica



Rețele Neuronale Artificiale

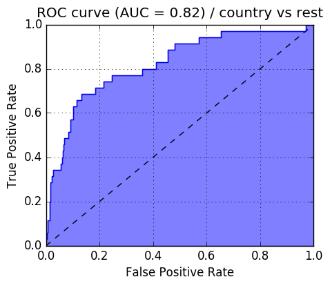


Random Forest

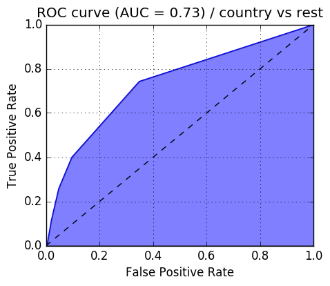


SVM

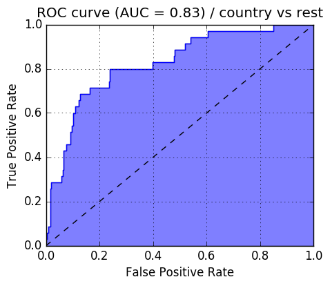
## Clasic VS. Rest



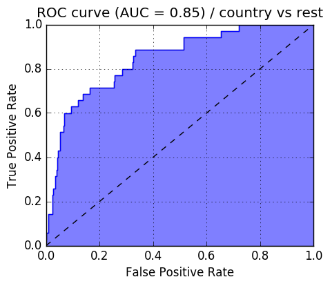
Gradient Boosting



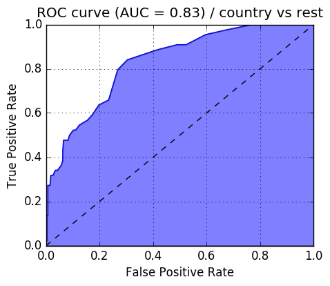
KNN



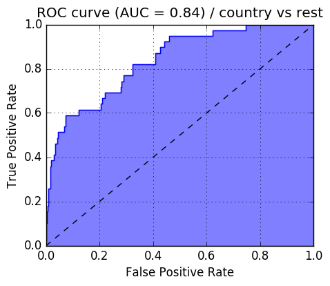
Regresie Logistica



Rețele Neuronale Artificiale

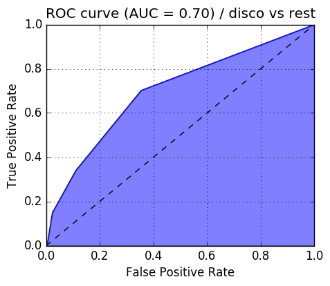


Random Forest

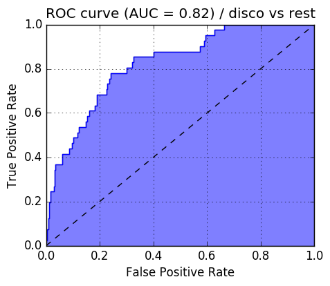


SVM

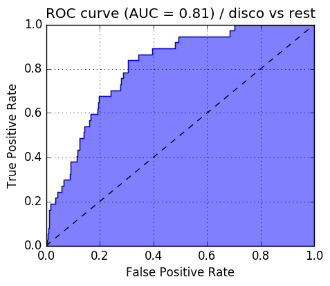
## Country VS. Rest



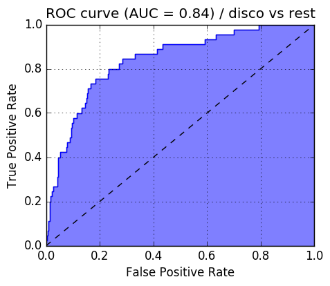
KNN



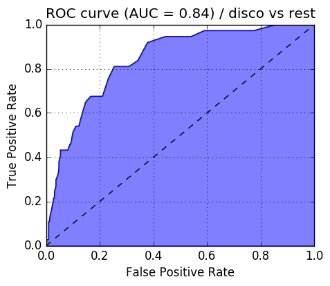
Gradient Boosting



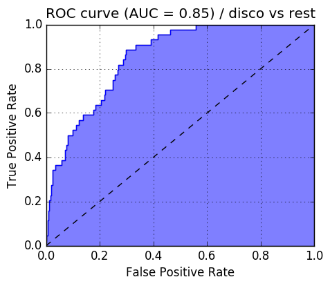
Regresie Logistica



Rețele Neuronale Artificiale

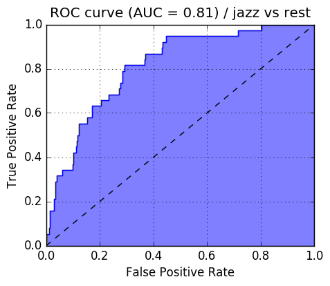


Random Forest

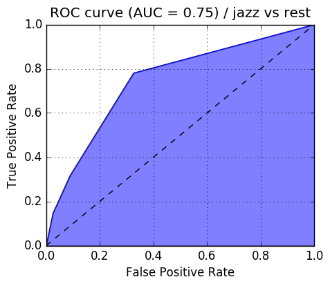


SVM

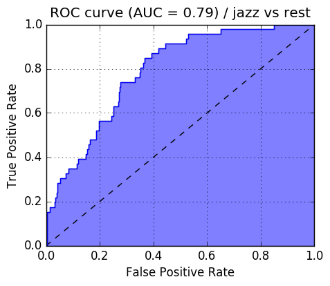
## Disco VS. Rest



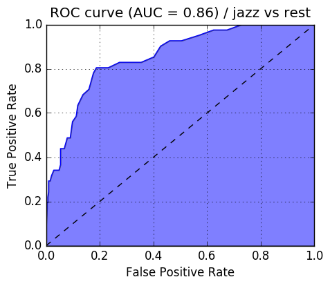
Gradient Boosting



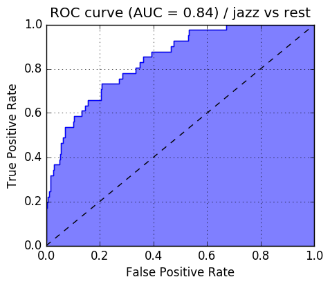
KNN



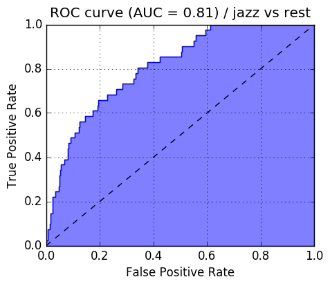
Regresie Logistica



Random Forest

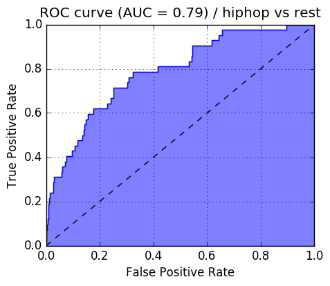


SVM

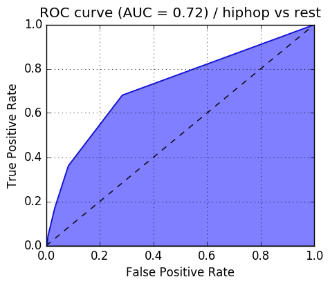


Rețele Neuronale Artificiale

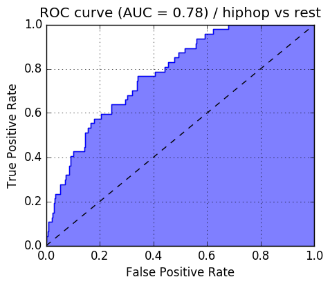
## Jazz VS. Rest



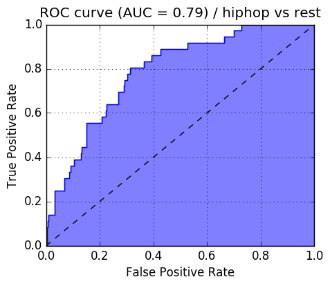
Gradient Boosting



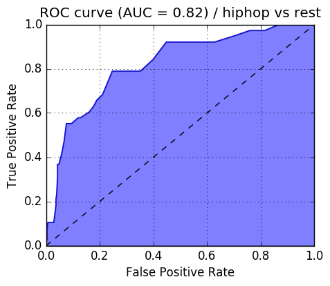
KNN



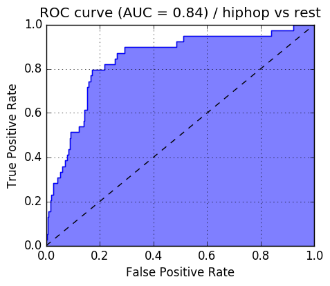
Regresie Logistica



Rețele Neuronale Artificiale



Random Forest

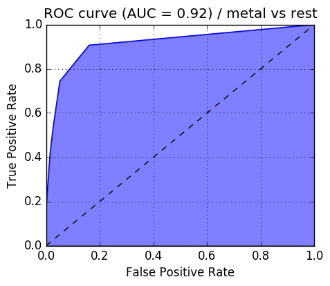


SVM

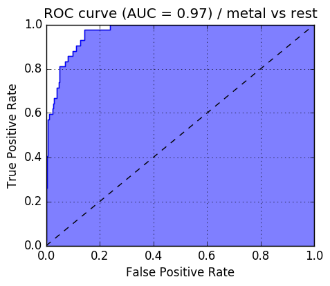
## Hiphop VS. Rest



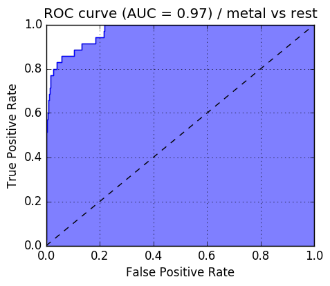
Gradient Boosting



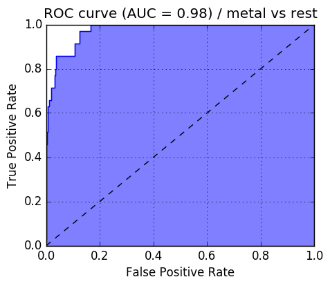
KNN



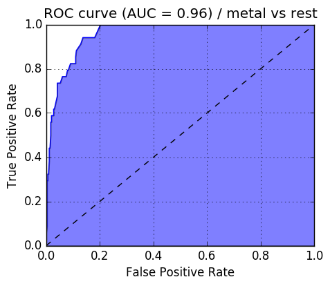
Regresie Logistica



Rețele Neuronale Artificiale

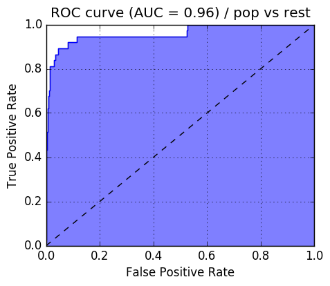


SVM

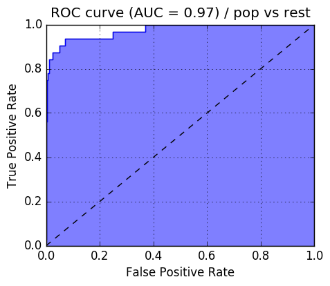


Random Forest

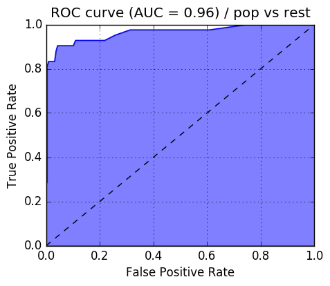
## Metal VS. Rest



Gradient Boosting

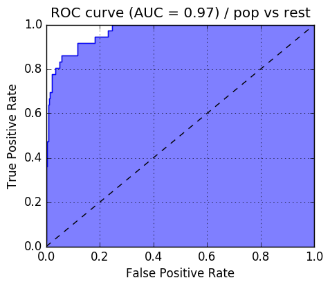
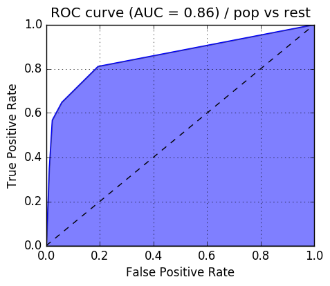


SVM

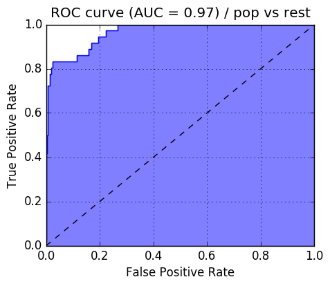


Random Forest

KNN



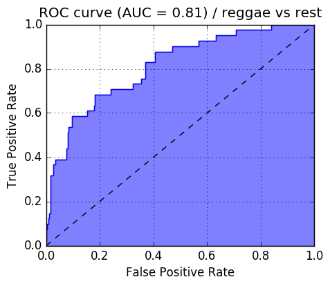
Regresie Logistica



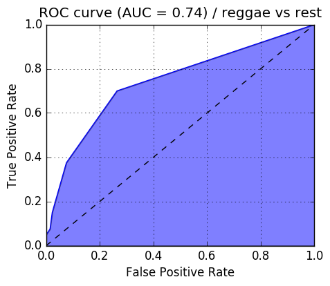
Rețele Neuronale Artificiale

## Pop VS. Rest

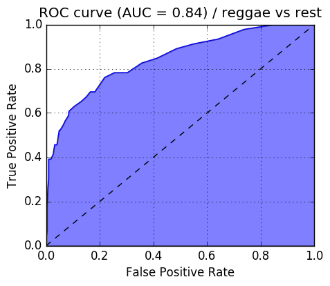
VS. Rest



Gradient Boosting



KNN

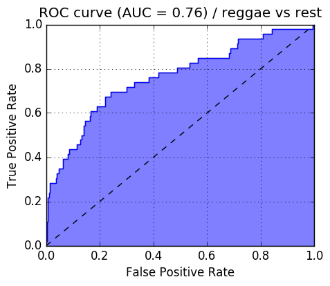
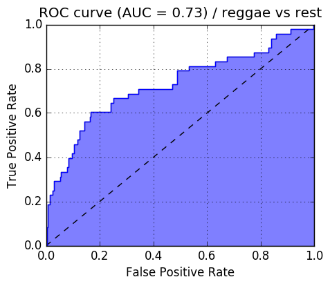


Random Forest



SVM

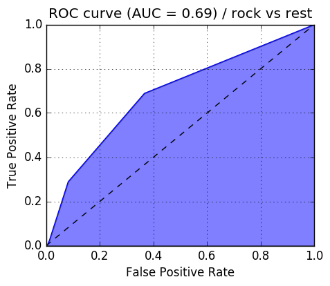
Regresie Logistica



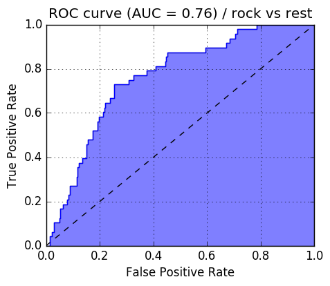
Rețele Neuronale Artificiale

## Raggae VS. Rest

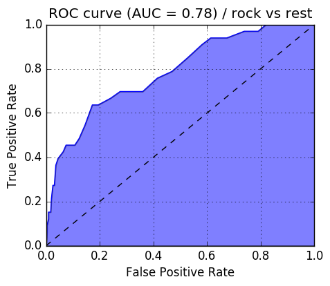
VS. Rest



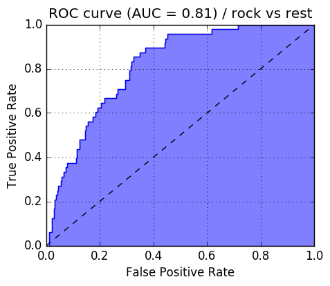
KNN



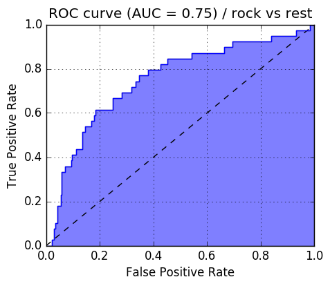
Gradient Boosting



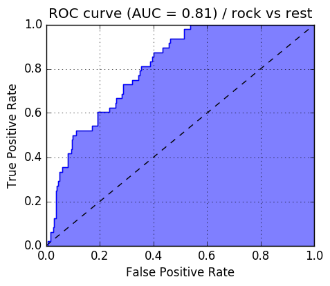
Random Forest



SVM



Regresie Logistica

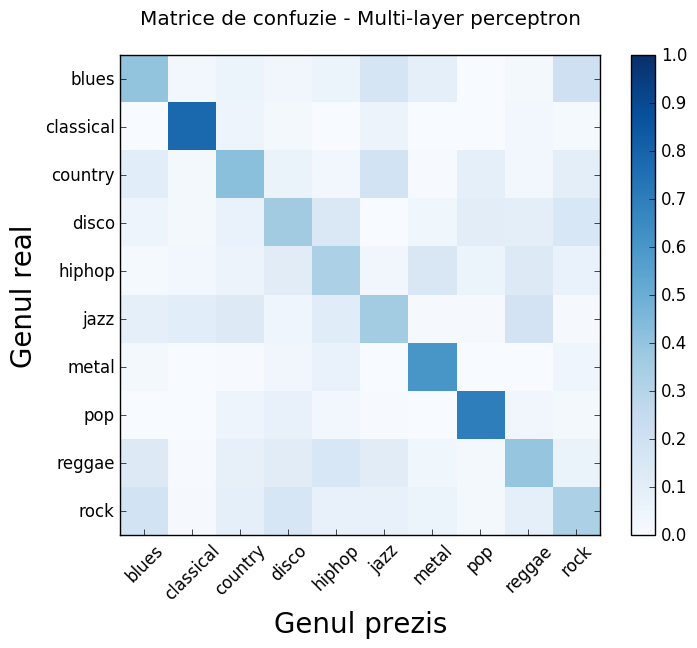


Rețele Neuronale Artificiale

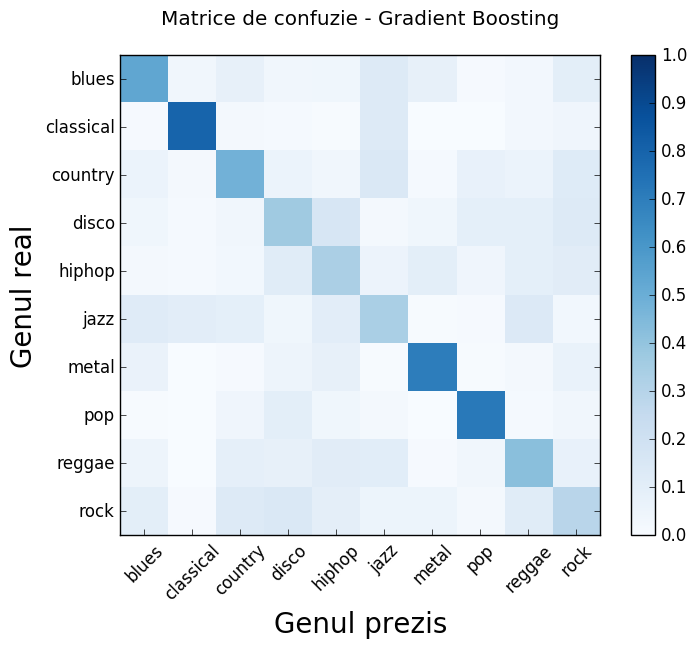
## Rock VS. Rest

VS. Rest

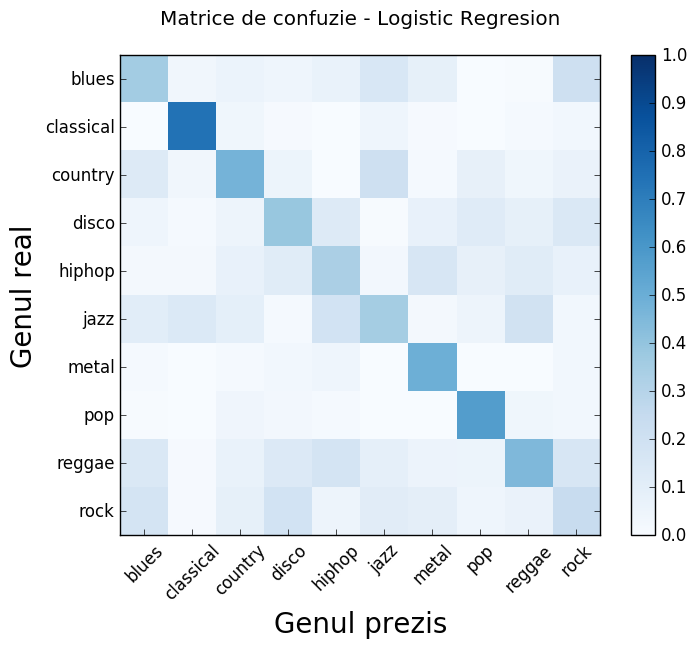
# Anexa 3



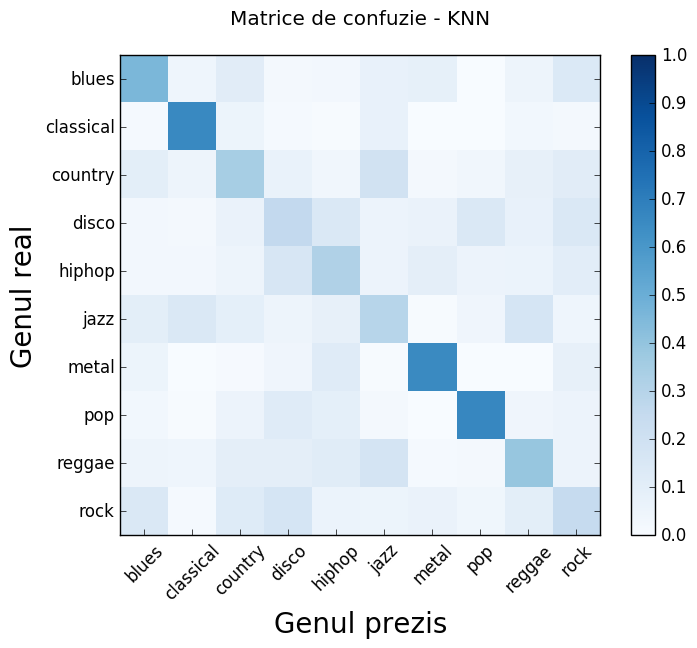
Rețele Neuronale Artificiale



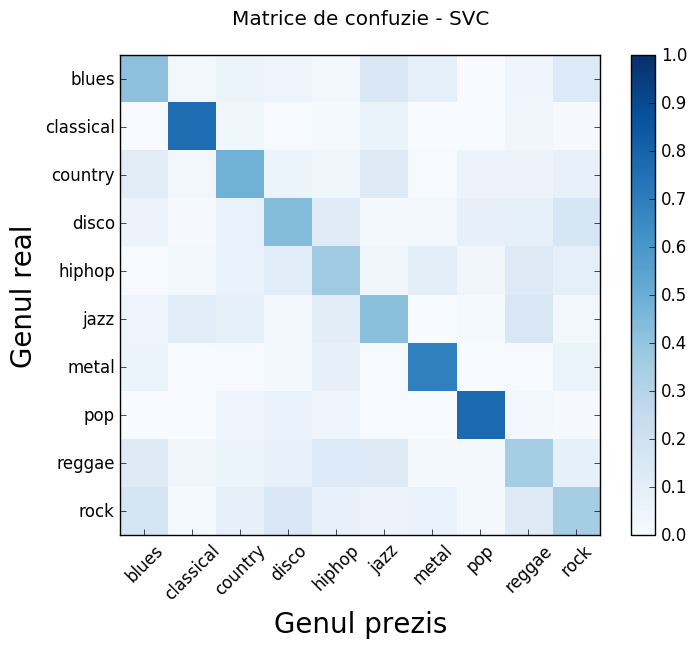
Gradient Boosting



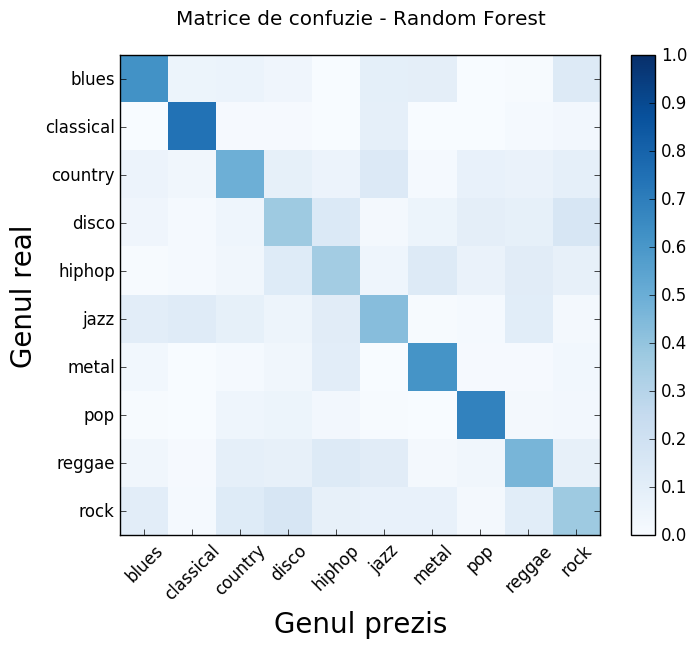
Regresia Logistică



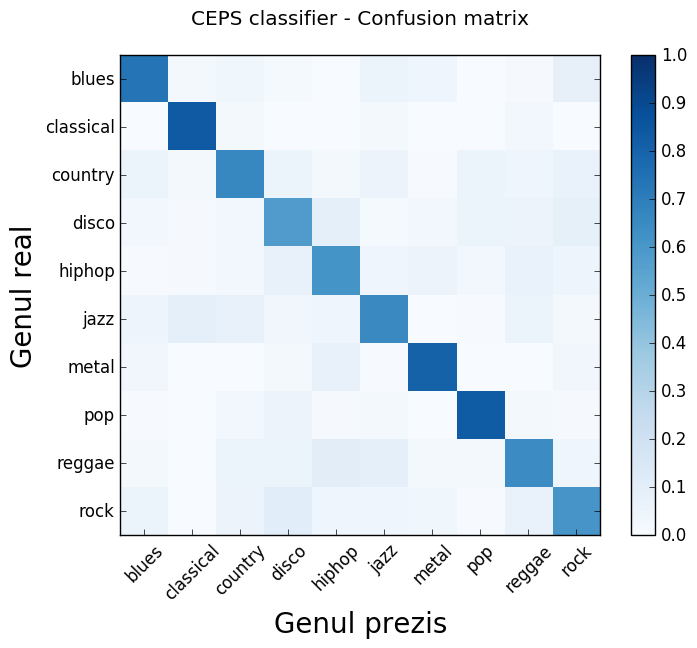
KNN



SVM

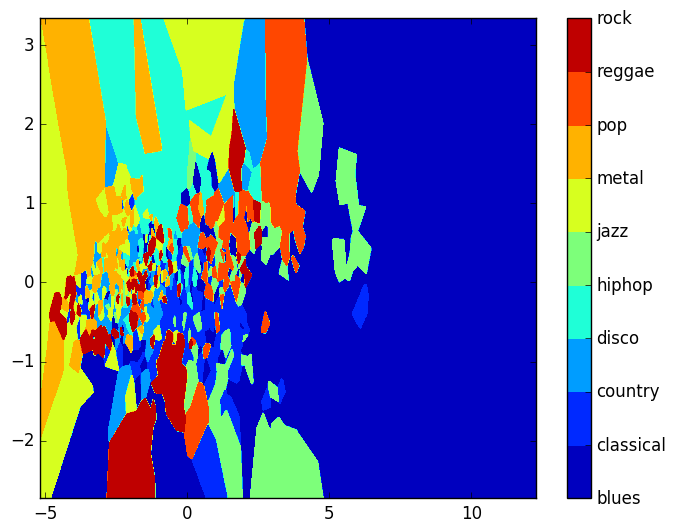


Random Forest

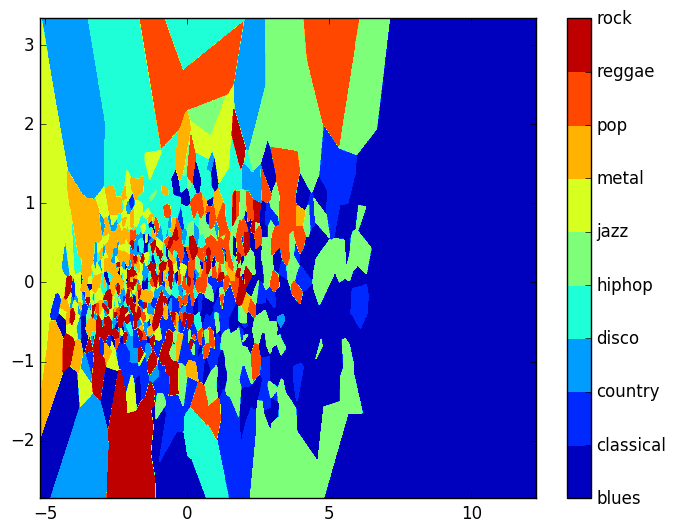


Clasificatorul expert

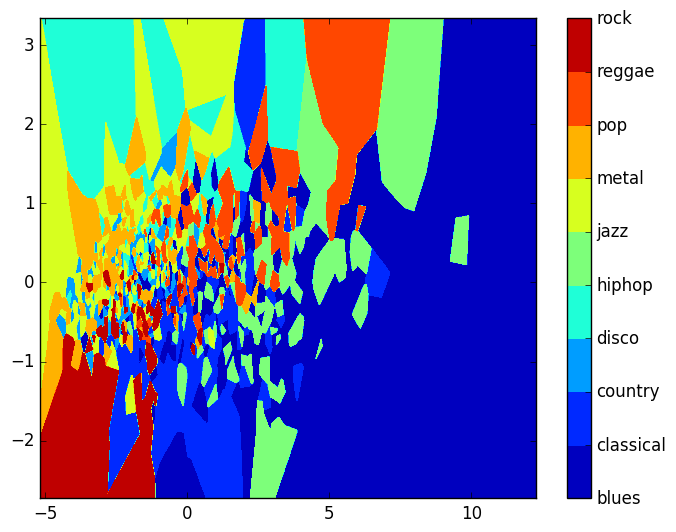
# Anexa 4



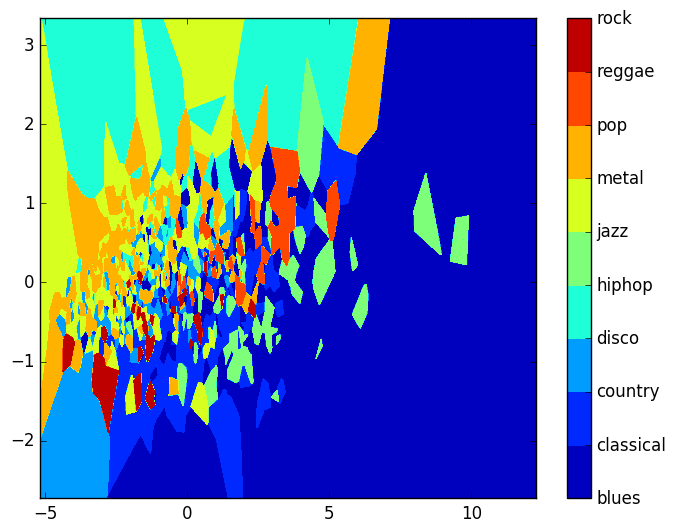
KNN



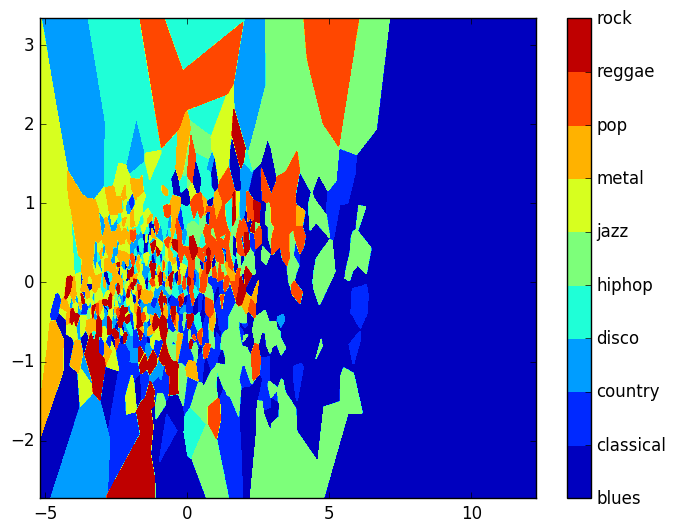
Gradient Boosting



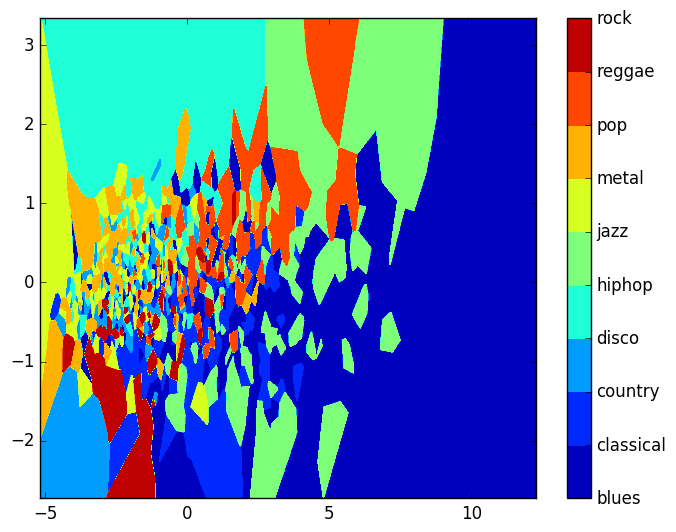
Rețele Neuronale Artificiale



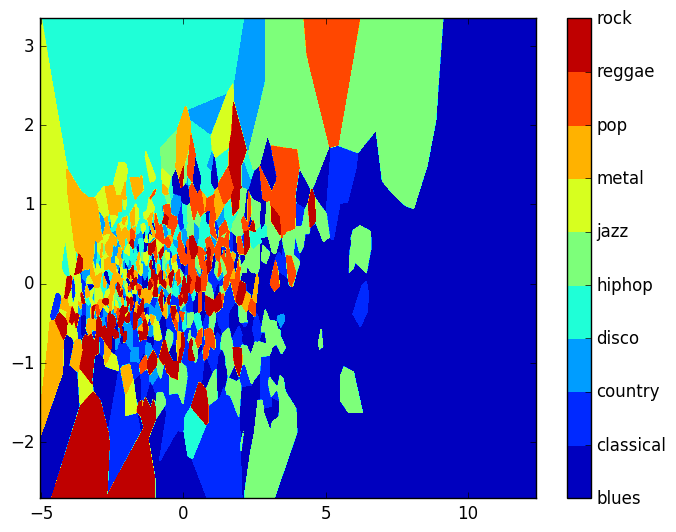
Regresie Logistică



Random Forest



SVM



# Bibliografie şi Webografie

1. North, A. C., and D. J. Hargreaves. 1997.”Liking for musical styles”. Music Scientae 1: 109–28.

   [ https://www.researchgate.net/publication/239066308\_Liking\_for\_musical\_styles ] [↑](#endnote-ref-1)
2. S. B. Davis and P. Mermelstein: “Comparison of parametric representationsfor monosyllabic word recognition in continuously spoken sentences”, IEEE Transactions on Acoustic, Speech, and Signal Processing, Vol. 28, No. 4, August 1980, pp. 357–366. [↑](#endnote-ref-2)
3. Tzanetakis, G., and P. Cook. 2002. Musical genre classification of audio signals. IEEE Transactions on Speech and Audio Processing 10 (5): 293–302.

   [https://www.researchgate.net/publication/220656193\_Musical\_Genre\_Classification\_of\_Audio\_Signals ] [↑](#endnote-ref-3)
4. K. Kosina. Music Genre Recognition. MSc. Dissertation, Fachschule Hagenberg, June 2002.

   [ http://kyrah.net/mugrat/mugrat.pdf ] [↑](#endnote-ref-4)
5. T. Li; M. Ogihara; Q. Li. A Comparative study on content-based Music Genre Classification. Proceedings of the 26th Annual International ACM SI-GIR Conference on Research and Development in Informaion Retrieval, Toronto, ACM Press, pages 282-289, 2003. [↑](#endnote-ref-5)
6. M. Grimaldi; P. Cunningham; A. Kokaram. A wavelet packet representation of audio signals for music genre classification using different ensemble and feature selection techniques. *Proceedings of the 5th ACM SIGMM International Workshop on Multimedia Information Retrieval*, ACM Press, pages 102-108, 2003. [↑](#endnote-ref-6)
7. M. Grimaldi; P. Cunningham; A. Kokaram. An evaluation of alternative feature selection strategies and ensemble techniques for classifying music. *Workshop on Multimedia Discovery and Mining*, 14th European Conference on Machine Learning, 7th European Conference on Principles and Practice of Knowledge Discovery in Databases, Dubrovnik, Croatia, 2003 [↑](#endnote-ref-7)
8. S.B. Davis, and P. Mermelstein (1980), "Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences," in IEEE Transactions on Acoustics, Speech, and Signal Processing, 28(4), pp. 357–366

   [ https://books.google.at/books?id=yjzCra5eW3AC&pg=PA65#v=onepage&q&f=false ] [↑](#endnote-ref-8)
9. https://cloud.google.com/products/machine-learning/ [↑](#endnote-ref-9)
10. Probably Approximately Correct: Nature’s Algorithms for Learning and Prospering in a Complex World, page 152, 2013 [↑](#endnote-ref-10)
11. A decision-theoretic generalization of on-line learning and an application to boosting, 1995 [↑](#endnote-ref-11)
12. Applied Predictive Modeling, 2013 [↑](#endnote-ref-12)
13. Prediction Games and Arching Algorithms [PDF], 1997

    14https://www.tutorialspoint.com/artificial\_intelligence/artificial\_intelligence\_neural\_networks.htm

    15http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/

    Liviu Ciortuz, Alina Munteanu, Elena Bădărău – “Exerciţii de învăţare automată”, Editura Universităţii “Alexandru Ioan Cuza” Iaşi

    https://www.researchgate.net/figure/220942923\_fig1\_Fig-1-An-example-illustrating-the-problem-space-decomposition-strategy

    https://www.researchgate.net/publication/3333877\_Musical\_genre\_classification\_of\_audio\_signals\_IEEE\_Trans\_Speech\_Audio\_Process

    http://machinelearningmastery.com/logistic-regression-for-machine-learning/

    https://www.tutorialspoint.com/artificial\_intelligence/artificial\_intelligence\_neural\_networks.htm

    https://hal.archives-ouvertes.fr/inria-00103955/document

    http://amueller.github.io/

    http://machinelearningmastery.com/logistic-regression-for-machine-learning/

    https://en.wikipedia.org/wiki/Music\_genre

    https://www.linkedin.com/pulse/r-vs-python-matlab-octave-julia-who-winner-siva-prasad-katru

    https://www.oreilly.com/ideas/six-reasons-why-i-recommend-scikit-learn [↑](#endnote-ref-13)