Mitocaru Irina

343C1

Analiza problemei

Contents

[Introducere 2](#_Toc35966765)

[Structura temei 2](#_Toc35966766)

[Analiza setului de date 3](#_Toc35966767)

[1. Dimensiune 3](#_Toc35966768)

[2. Clase 3](#_Toc35966769)

[3. Atribute 3](#_Toc35966770)

[Analiza clasificatorilor 4](#_Toc35966771)

[K-means 4](#_Toc35966772)

[1. Baseline 4](#_Toc35966773)

[2. Imbunatatiri 4](#_Toc35966774)

[3. Grafice intra-algoritmi 5](#_Toc35966775)

[Random Forest Classifier 6](#_Toc35966776)

[1. Baseline 6](#_Toc35966777)

[2. Imbunatatiri 6](#_Toc35966778)

[3. Grafice intra-algoritmi 7](#_Toc35966779)

[XGBoost Classifier 8](#_Toc35966780)

[1. Baseline 8](#_Toc35966781)

[2. Imbunatatiri 8](#_Toc35966782)

[3. Grafice intra-algoritmi 9](#_Toc35966783)

[SVM (SVC) Classifier 10](#_Toc35966784)

[1. Baseline 10](#_Toc35966785)

[2. Imbunatatiri 10](#_Toc35966786)

[3. Grafice intra-algoritmi 11](#_Toc35966787)

[Grafice inter-algoritmi 11](#_Toc35966788)

[1. Acuratete 12](#_Toc35966789)

[2. Precizie 12](#_Toc35966790)

[3. Recall 13](#_Toc35966791)

[4. F1 13](#_Toc35966792)

[5. Matricea de confuzie 14](#_Toc35966793)

[Concluzie 15](#_Toc35966794)

# Introducere

**!** Pentru rularea temei trebuie data mai intai urmatoarea comanda:

pip install -r requirements.txt

Acest document ilustreaza atat analiza amanuntita a setului de date cu care se lucreaza, cat si analiza 4 modele folosite la clasificarea unui set de date dupa genul muzical.

# Structura temei

Clase:

* KMeansModel
* RandomForestModel
* XGBoostModel
* SVMModel

Fiecare din ele are urmatoarele functii:

* <nume\_model>\_baseline:

implementarea baseline

* <nume\_model>\_improved:

implementarea imbunatatita

* intra\_algorithm\_analysis:

apeleaza functia de afisat bar\_chart pe baza metricilor rezultate

Alte functii:

* inter\_algorithm\_analysis:

afiseaza, pe rand, cate un grafic cu una din cele 5 metrici (acuratete, precizie, recall, f1 si matricea de confuzie)

* get\_classification\_metrics:

calculeaza metricile: acuratete, precizie, recall, f1 pentru un model dat, folosind cross\_validate cu 5-fold

* get\_confusion\_matrix:

returneaza matricea de confuzie pentru 5-fold

* get\_rand\_index:

returneaza rand\_index pentru modelul dat

* get\_avg\_score:

returneaza average de un vector

* plot\_bar\_chart:

afiseaza un bar chart pentru modelul dat si metricile date

* plot\_metrics:

afiseaza un grafic cu performanta algoritmilor Random Forest, XGBoost si SVM pentru metrica data

* plot\_conf\_matrix:

afiseaza matricea de confuzie

* plot\_silhouette:

afiseaza distributia clusterelor si numarul de clustere optim

Grafice generate:

**Intra-algoritmi**

K-Means:

* un grafic care arata alegerea numarul optim de clustere si distributia acestora
* un bar chart care compara randIndex baseline si improved

Random Forest:

* un bar chart care compara metricile: accuracy, precision, recall, f1
* un grafic cu matricea de confuzie

XGBoost:

* un bar chart care compara metricile: accuracy, precision, recall, f1
* un grafic cu matricea de confuzie

SVM:

* un bar chart care compara metricile: accuracy, precision, recall, f1
* un grafic cu matricea de confuzie

**Inter-algoritmi**

* cate un grafic pentru fiecare metrica care compara valorile obtinute de modelele: Random Forest, XGBoost si SVM

# Analiza setului de date

### Dimensiune

Setul de date e impartit in 2000 de exemple pentru setul de invatare si 400 de exemple pentru setul de testare si validare. In total sunt 2400 de exemple.

### Clase

Facand o analiza a distributiei claselor, s-a observat ca toate 4 (Hip-Hop, Electronic, Folk si Rock) sunt impartite in mod egal, fiecare avand 600 de exemple, lucru util deoarece se evita fenomenul de overfitting.

### Atribute

#### Echonest.csv

* audio\_features: 8 atribute, valori numerice si fara NaN
* social\_features: 5 atribute, valori numerice si fara NaN
* ranks: 5 atribute, valori numerice si NaN
* metadata: 7 atribute, valori numerice, categorice si NaN
* temporal\_features: 224 atribute, valori numerice si fara NaN

#### Features.csv

* chroma\_cens, chroma\_cqt, chroma\_stft
* mfcc
* rmse
* spectral\_bandwidth, spectral\_centroid, spectral\_contrast, spectral\_rolloff
* tonnetz
* zcr

Toate sunt in combinatie cu: 'kurtosis', 'max', 'mean', 'min', 'skew' si 'std', avand atat valori numerice, categorice, cat si NaN.

# Analiza clasificatorilor

## K-means

### Baseline

KMeans(n\_clusters=n\_clusters)

### Imbunatatiri

#### Hiperparametri

KMeans(n\_clusters=n\_clusters)

**n\_clusters:**

S-a ales numarul optim de clustere folosind silhouette.

**init:**

* ‘k-means++’: nu a adus imbunatatiri acuratetei
* ‘random’: nu a adus imbunatatiri acuratetei

S-a ales default: ‘k-means++’

**tol:**

S-au incercat mai multe valori fara a aduce imbunatiri. S-a ales default.

**precompute\_distances:**

**S-a incercat ‘True’, fara imbunatatiri. S-a ales default.**

**algorithm:**

* **‘full’: fara imbunatatiri**
* **‘elkan’: fara imbunatatiri**

**S-a ales default: ‘auto’.**

#### Preprocesare

preprocessing.StandardScaler().fit\_transform(x\_train)

S-a aplicat standardizarea.

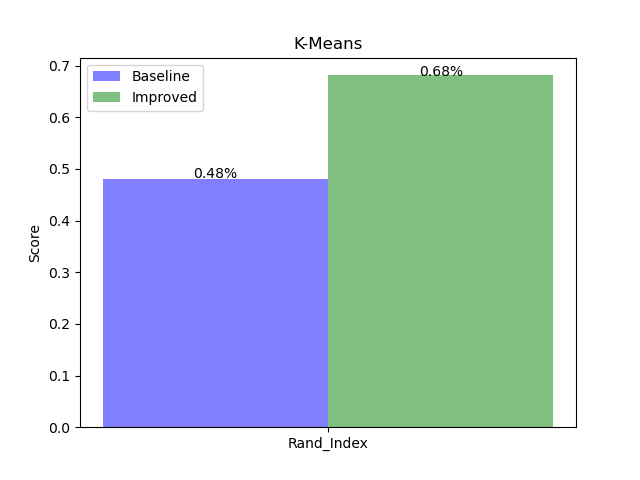
#### Atribute

S-a incercat combinatia dintre audio\_features cu fiecare din celelalte din categoria Echonest si cateva din Features. Cea mai optima solutie a fost folosirea exclusiva de ‘audio-features’.

‘audio\_features’

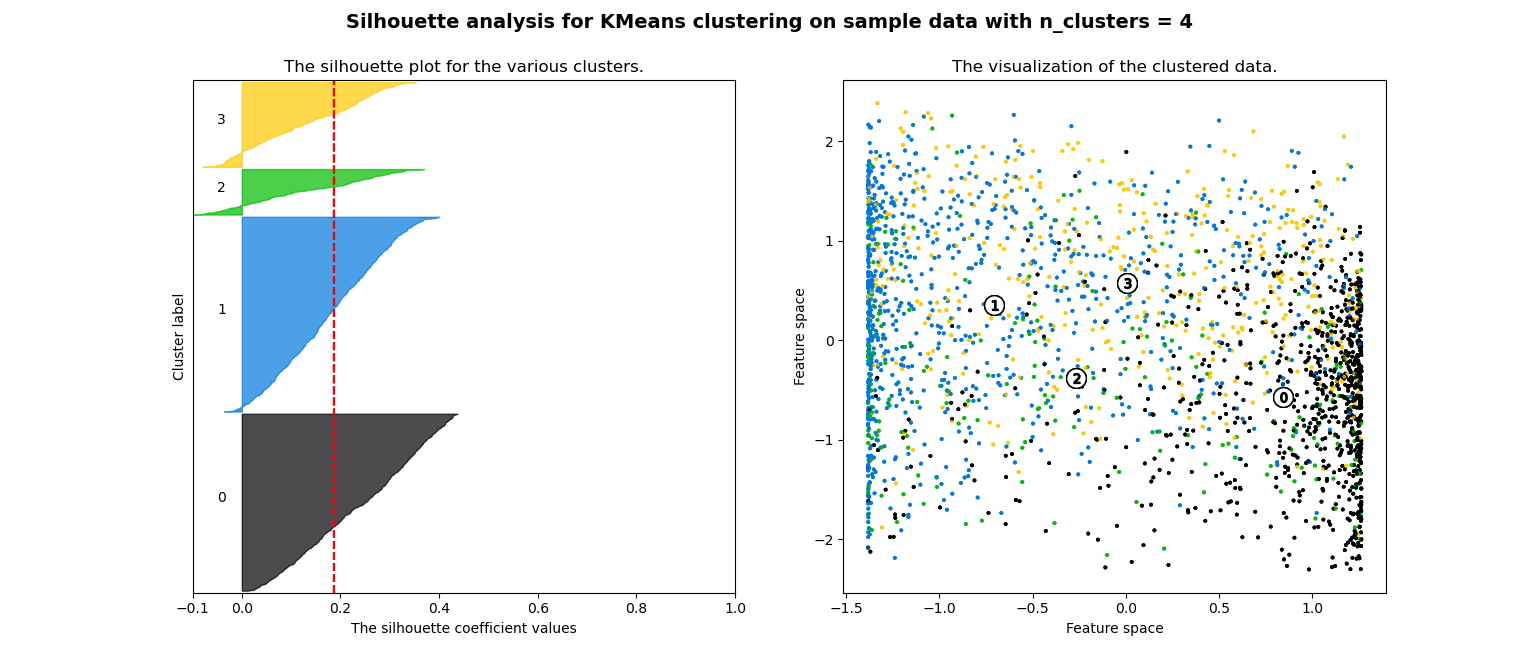
### Grafice intra-algoritmi

* Rand Index: 0.48% baseline, 0.68% varianta imbunatatita



Imbunatatirea majora vine din standardizarea datelor din etapa de preprocesare. Algoritmul poate clusteriza cu o acuratete mai mare in varianta a doua deoarece datele nu mai sunt atat de “indepartate” intre ele, fiind toate centralizate in 0 cu o margine de 1 unitate.

* Alegerea numarului optim de clustere si vizualizarea acestora



## Random Forest Classifier

### Baseline

RandomForestClassifier(random\_state=0)

### Imbunatatiri

#### Hiperparametri

RandomForestClassifier(n\_jobs=-1, n\_estimators=200, max\_depth=depth, random\_state=0)

S-au incercat urmatorii hiper-parametri:

**n\_estimators**:

* (100, 500): acuratete cu 3% mai mare
* >= 500: acuratete la metrica f1\_score cu 2% mai mare, iar la restul cu 3% mai mare

Astfel, s-a ales valoarea 200.

**max\_depth**:

* None: acurarete la fel ca la baseline
* > ¾ din numarul de atribute: acuratete cu 1% mai mare
* < ¾ din numarul de atribute: acuratete mai slaba ca la baseline
* = ¾ din numarul de atribute: acuratete cu 4% mai mare

S-a observat ca o combinatie de numar de estimatori mic (100) cu o limita de adancime mica (1/2 din numarul de atribute) rezulta in performante proaste, deoarece algoritmul nu ia in considerare toate atributele (sau mai multe atribute) care aduc un plus acuratetei.

S-a ales max\_depth = 3/4 din numarul de atribute.

**criterion**:

* ‘gini’: acurarete la fel ca la baseline
* ‘entropy’: acuratete mai slaba ca la baseline

S-a ales ‘gini’.

**n\_jobs**:

S-a ales -1 pentru a folosi tot CPU-ul la maxim, daca este nevoie.

**min\_samples\_split & min\_samples\_leaf:**

* >2: nu avem imbunatatiri

S-a ales valoarea default: 1.

**max\_features:**

* int: nu avem imbunatatiri
* alte optiuni mai putin default: performante mai proaste

S-a ales default: auto.

**max\_leaf\_nodes & min\_impurity\_decrease & min\_impurity\_split & bootstrap & cc\_alpha:**

* alte optiuni mai putin default: performante mai proaste

S-a ales default: None.

**random\_state:**

S-a ales valoarea 0 pentru a face modelul determinist.

#### Preprocesare

depth = int(len(x\_train.columns) \* 3 / 4)

x\_train = SelectKBest(f\_classif, k=depth).fit\_transform(x\_train, y\_train)

Pentru preprocesarea setului de invatare, s-a incercat:

* Standardizare: fara rezultat
* Normalizare: fara rezultat
* SelectKBest: singura metoda care a adus un plus de 2-3% la perfomanta. Si aici s-a ales un numar de atribute = ¾ din numarul total

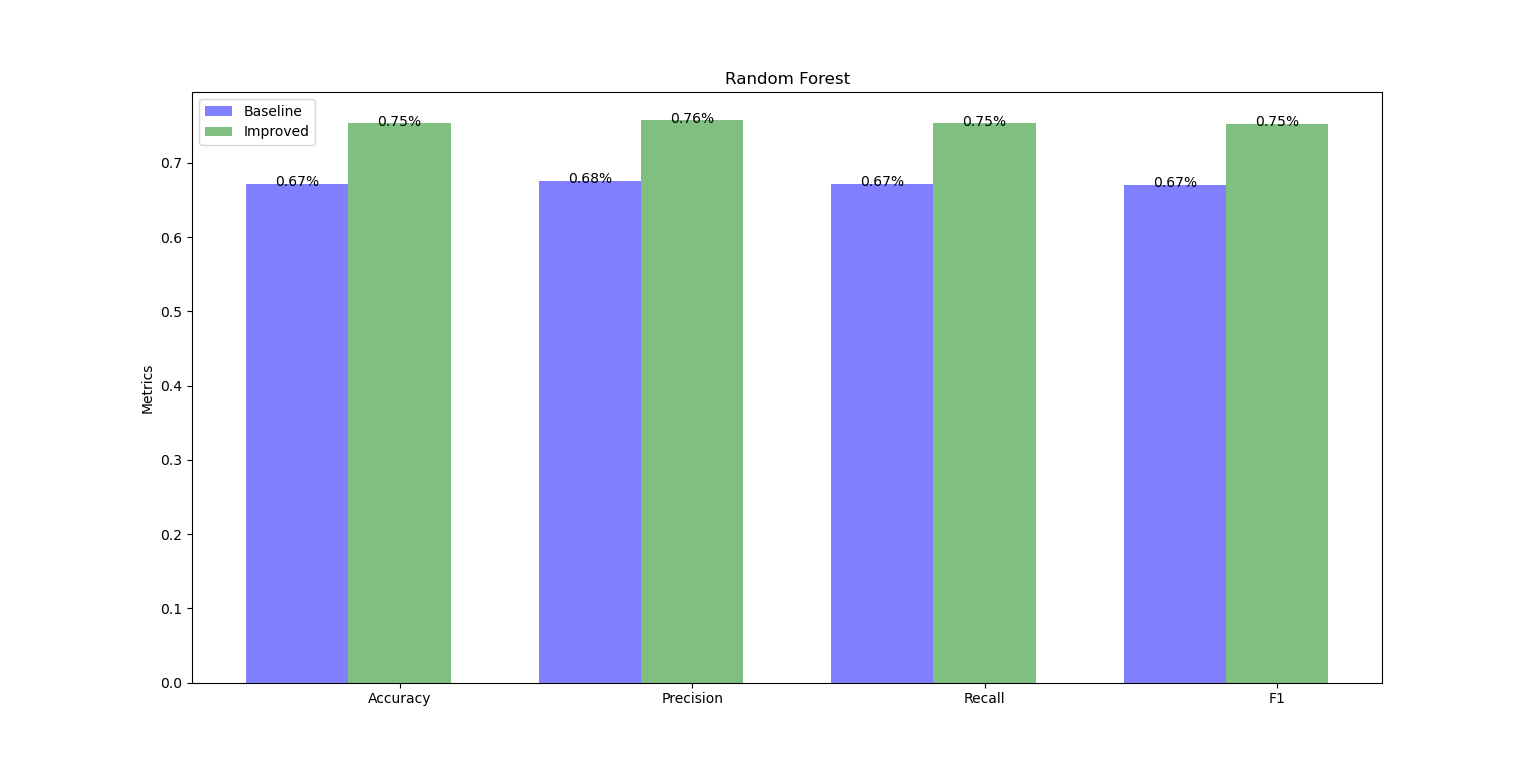
#### Atribute

S-a incercat combinatia dintre audio\_features cu fiecare din celelalte din categoria Echonest si cateva din Features. Singura combinatie care a adus un plus performantei a fost:

‘audio\_features’ cu ‘temporal\_features’

### Grafice intra-algoritmi

* Acuratete: 0.67% baseline, 0.75% varianta imbunatatita
* Precizie: 0.68% baseline, 0.76% varianta imbunatatita
* Recall: 0.67% baseline, 0.75% varianta imbunatatita
* F1: 0.67% baseline, 0.75% varianta imbunatatita



Se observa importanta rularii algoritmului pe un numar de atribute cat mai mare (care sa aiba insa si date echillibrate, adica valori apropriate, fara valori lipsa etc). Este, de asemenea, important si pasul de limitare a adancimii arborelor in acest caz si folosirea unui numar mai mare de estimatori (insa nu prea mare). Pentru acest model, SelectKBest a imbunatatit acuratetea deoarece a reusit sa selecteze cele mai importante atribute, avand in vedere ca nu a redus drastic numarul.

## XGBoost Classifier

### Baseline

XGBClassifier(random\_state=0)

### Imbunatatiri

#### Hiperparametri

XGBClassifier(n\_jobs=-1, random\_state=0, subsample=0.9, objective='multi:softmax', num\_class=4, learning\_rate=0.1)

S-au incercat urmatorii hiper-parametri:

**learning\_rate**:

* = 0.1: performanta imbunatatita cu 7%
* > 0.1: performanta imbunatatita cu 5%
* < 0.1: performanta intre baseline si baseline + 5%

S-a ales 0.1.

**max\_depth:**

* None: acurarete la fel ca la baseline
* > ¾ din numarul de atribute: acurarete la fel ca la baseline
* < ¾ din numarul de atribute: acuratete mai slaba ca la baseline
* = ¾ din numarul de atribute: acurarete la fel ca la baseline

S-a ales default: None.

**booster:**

* ’gbtree’: acurarete la fel ca la baseline
* ’gblinear’: acuratete mai slaba ca la baseline
* ’dart’: acurarete la fel ca la baseline

S-a ales default: None

**num\_parallel\_tree:**

S-au incercat mai multe valori, toate crescand complexitatea temporala si neaducand un plus de performanta. S-a ales default: None

**n\_estimators:**

* >= 100: acurarete la fel ca la baseline

S-a ales default: 100

**importance\_type:**

S-au incercat mai multe valori, fara rezultat. S-a ales default: None

**subsampling:**

* 1: acurarete la fel ca la baseline
* < 0.9: acuratete cu 1-2% mai mare
* = 0.9: acuratete cu 2% mai mare

S-a ales 0.9

**objective:**

S-a ales valoarea 'multi:softmax' deoarece modelul foloseste by default binary:logistic, care se foloseste la clasificator binar.

**num\_class:**

S-a ales valoarea 4 deoarece acesta este numarul de clase a setului nostru de date. Hiper-parametrul se foloseste in combinatie cu ’objective’.

**random\_state:**

S-a ales valoarea 0 pentru a face modelul determinist.

#### Preprocesare

Pentru preprocesarea setului de invatare, s-a incercat:

* Standardizare: fara rezultat
* Normalizare: fara rezultat
* SelectKBest: fara rezultat

Astfel, nu s-a efectuat nicio preprocesare a setului de invatare pentru acest model.

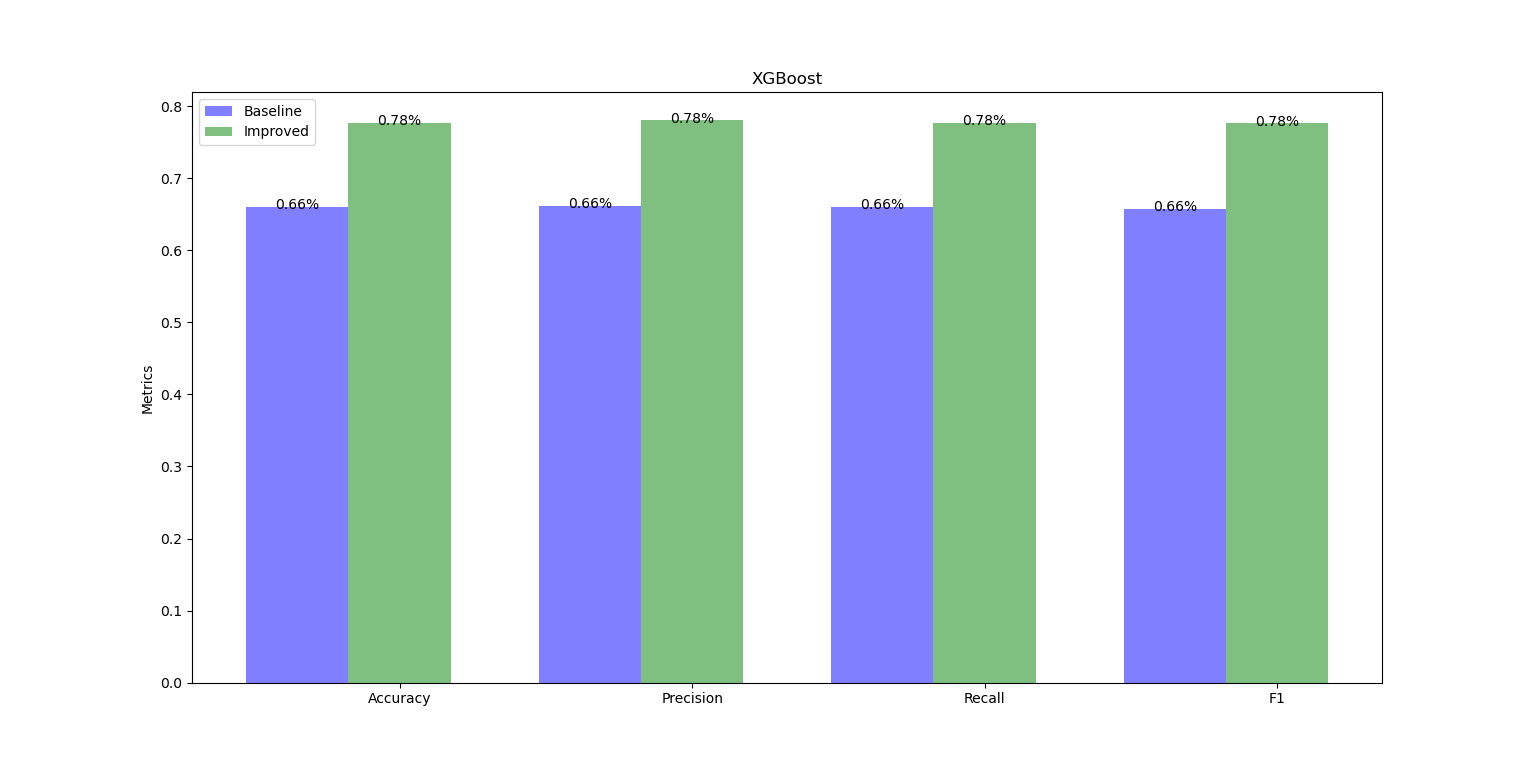
#### Atribute

S-a incercat combinatia dintre audio\_features cu fiecare din celelalte din categoria Echonest si cateva din Features. Singura combinatie care a adus un plus performantei a fost:

‘audio\_features’ cu ‘temporal\_features’

### Grafice intra-algoritmi

* Acuratete: 0.66% baseline, 0.78% varianta imbunatatita
* Precizie: 0.66% baseline, 0.78% varianta imbunatatita
* Recall: 0.66% baseline, 0.78% varianta imbunatatita
* F1: 0.66% baseline, 0.78% varianta imbunatatita



XGBoost reuseste sa imbunatateasca performantele doar prin rularea cu un numar mai mare de atribute. Trebuie avand in vedere ca, desi este cel mai bun model de pana acum, foloseste resursele la maxim, iar rularea dureaza mai mult. Imbunatatirea este adusa si de hiper-parametrul subsample care asigura faptul ca la urmatoarea rulare (din succesiunea secventiala) modelul va fi antrenat si mai bine pe un set de date necunoscut si astfel creste acuratetea si nu apare overfitting.

## SVM (SVC) Classifier

### Baseline

SVC(random\_state=0)

### Imbunatatiri

#### Hiperparametri

SVC(random\_state=0, decision\_function\_shape='ovo', kernel='rbf')

**kernel:**

* ‘rbf’: acuratete imbunatatita
* ‘poly’: acuratete imbunatatita mai putin
* ‘linear’: dureaza foarte mult procesul de invatare
* ‘sigmoid’: acuratate mai proasta decat cea din varianta ‘poly’

S-a ales default: ’rbf’.

**gamma & C:**

S-au incercat diferite valori, toate aducand performante foarte scazute. S-a ales default.

**random\_state:**

S-a ales valoarea 0 pentru a face modelul determinist.

**decision\_fucion\_shape:**

S-a ales valoarea ‘ovo’ (one vs one) pentru a imbunatati perfomanta clasificatorului multiclass.

#### Preprocesare

preprocessing.StandardScaler().fit\_transform(x\_train, y\_train)

Pentru preprocesarea setului de invatare, s-a incercat:

* Standardizare: performanta s-a imbunatati cu 20%
* Normalizare: fara rezultat
* SelectKBest: fara rezultat

Astfel, s-a ales standardizarea.

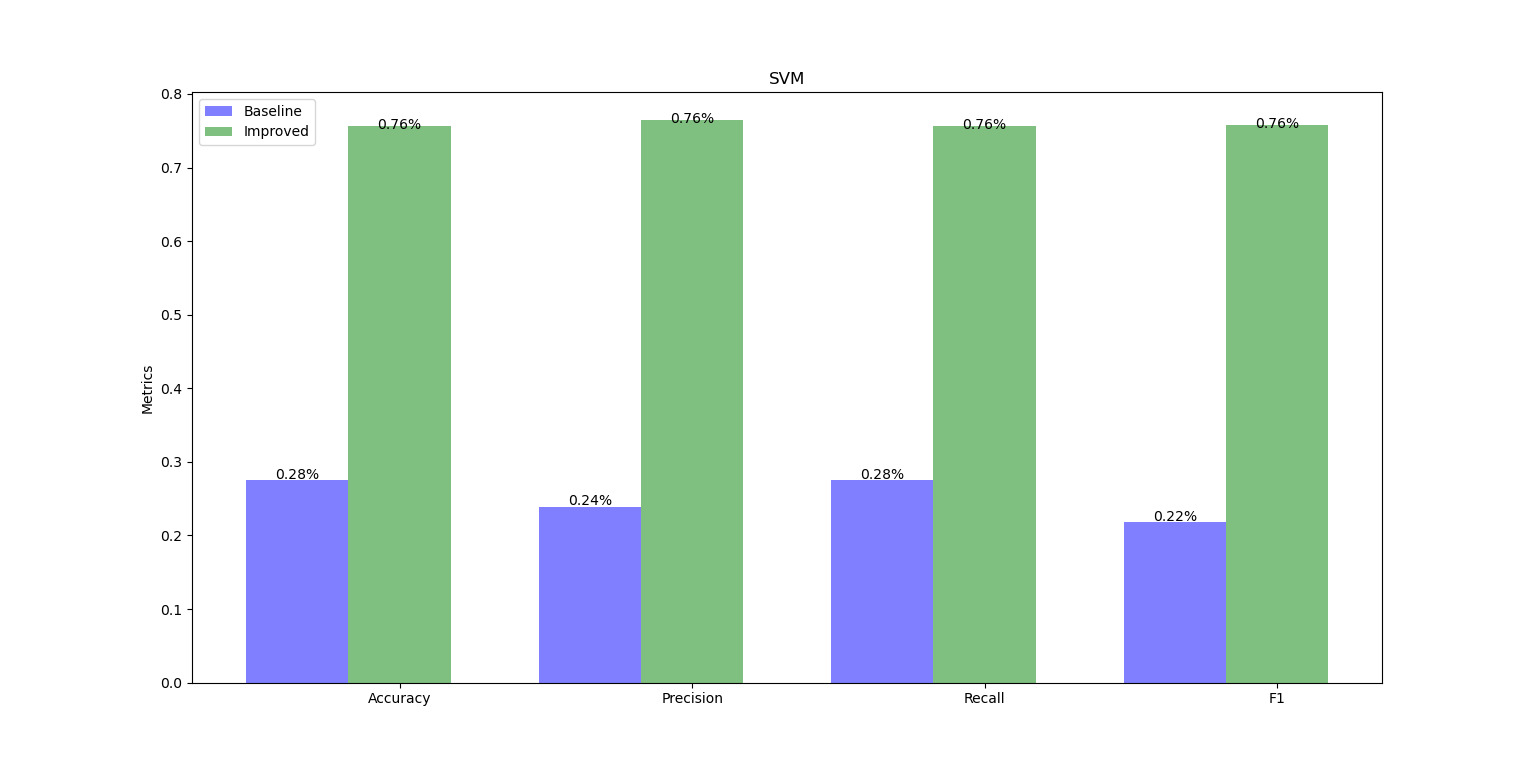
#### Atribute

S-a incercat combinatia dintre audio\_features cu fiecare din celelalte din categoria Echonest si cateva din Features. Singura combinatie care a adus un plus performantei a fost:

‘audio\_features’ cu ‘temporal\_features’

### Grafice intra-algoritmi

1. Acuratete: 0.28% baseline, 0.76% varianta imbunatatita
2. Precizie: 0.24% baseline, 0.76% varianta imbunatatita
3. Recall: 0.28% baseline, 0.76% varianta imbunatatita
4. F1: 0.22% baseline, 0.76% varianta imbunatatita



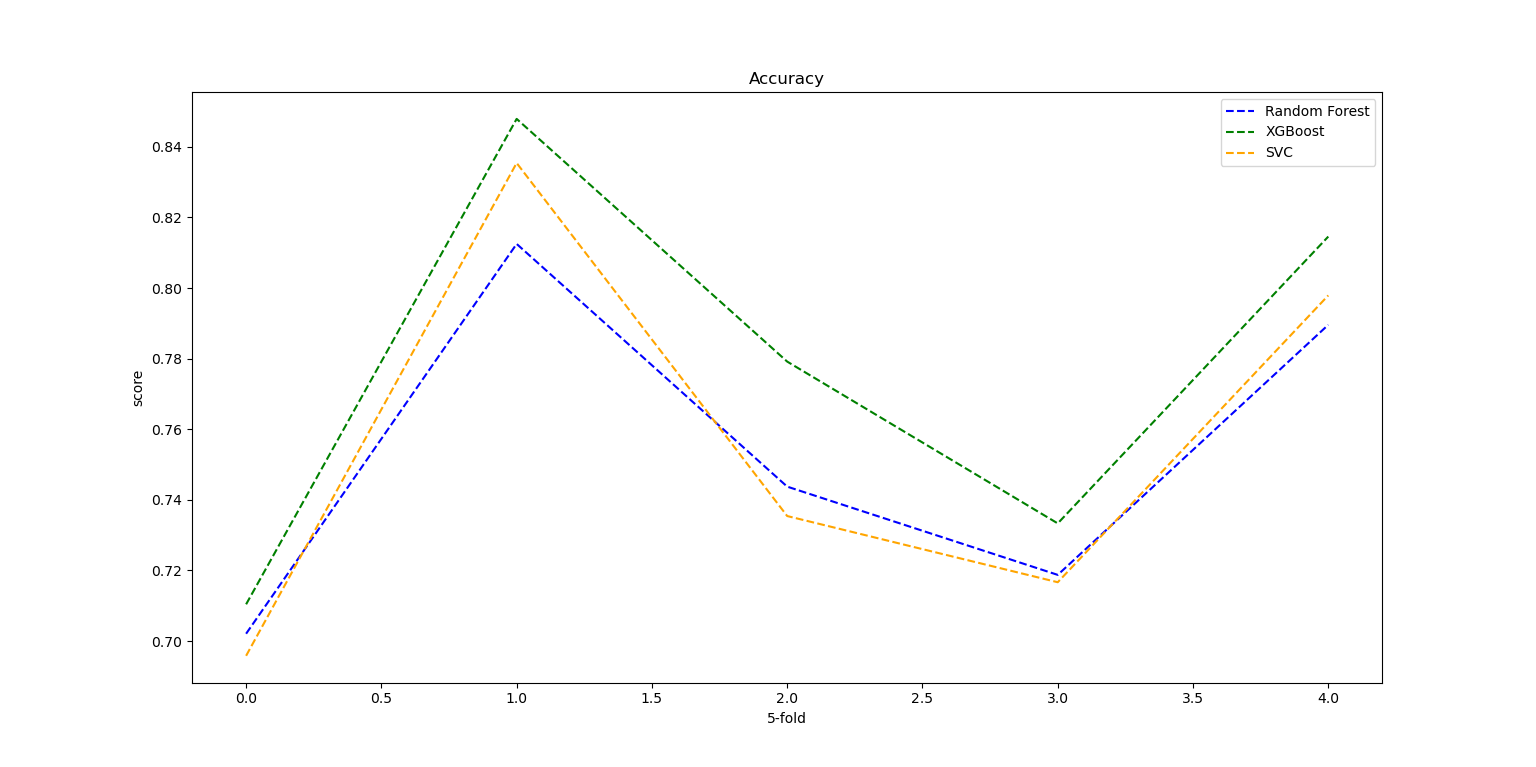
Se observa ca pentru acest model, kernelul ales are o importanta majora. Pe cazul nostru, in care datele sunt raspandite, cel mai bun kernel este RBF. Importanta este si valoarea parametrului ‘C’, standardizarea datelor inainte de aplicarea algoritmului pentru o acuratete mai buna si extinderea numarului de atribute folosite.

# Grafice inter-algoritmi

Pentru evaluare, s-a folosit 5-Fold Cross Validation si functia cross\_validate cu metricile de mai jos pe fiecare model in parte. Asupra vectorilor rezultati s-a aplicat average pentru a calcula procentul.

### Acuratete

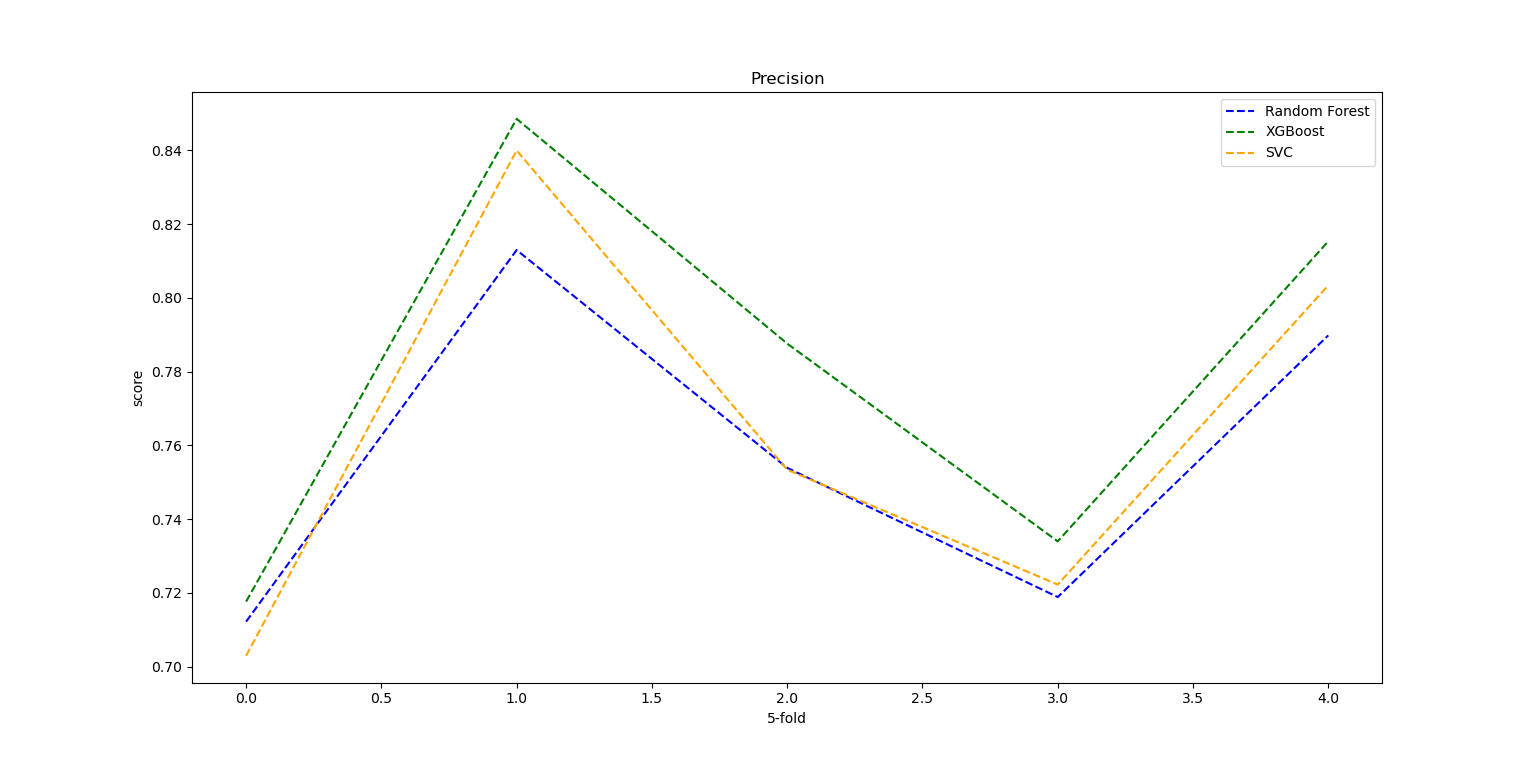
* Random Forest: 0.75%
* XGBoost: 0.78%
* SVM: 0.76%



Pe parcursul celor 5 rulari, performanta difera deoarece se alege mereu un set de date diferit pentru antrenare si pentru testare. In final, XGBoost are acuretetea cea mai buna.

### Precizie

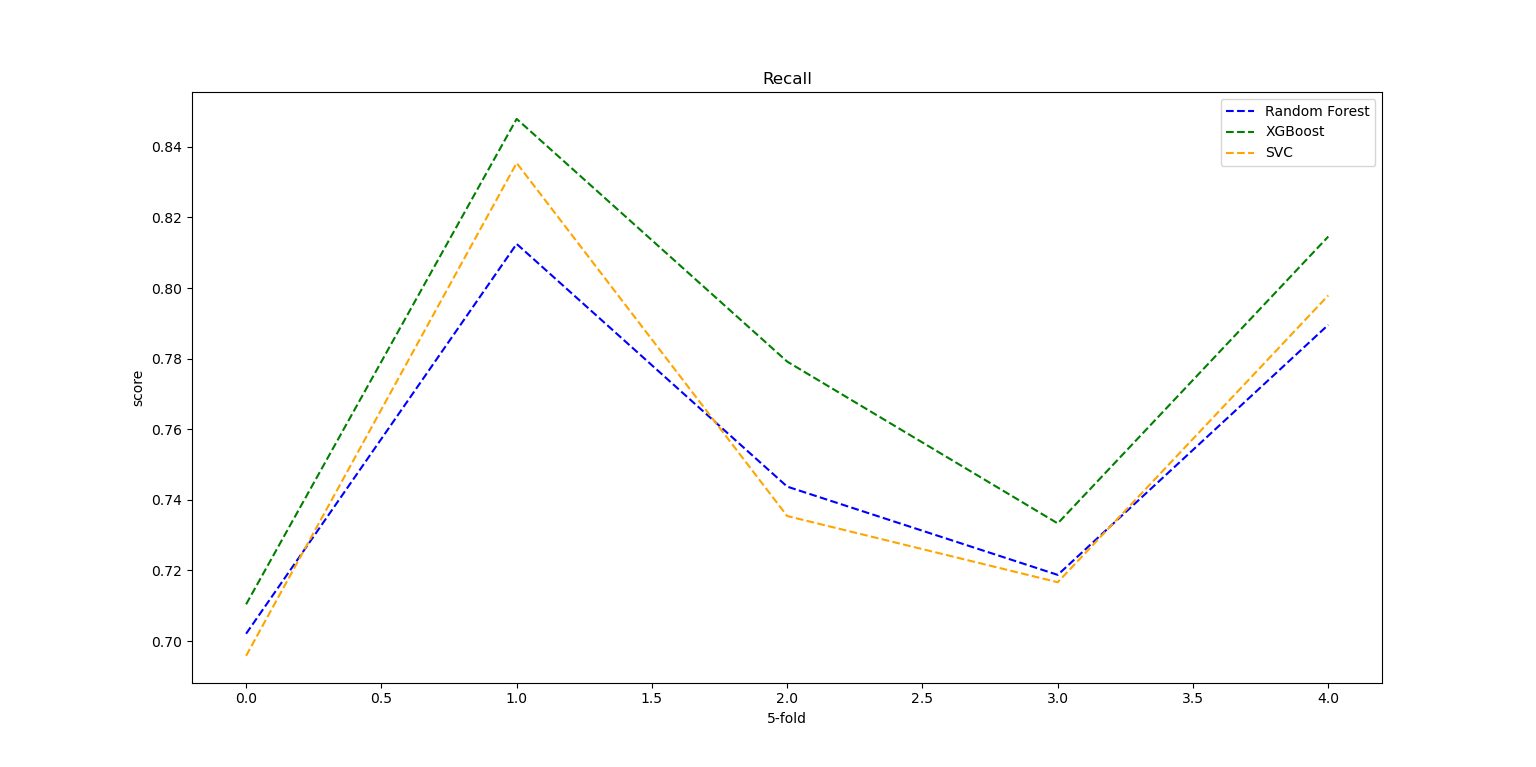
* Random Forest: 0.76%
* XGBoost: 0.78%
* SVM: 0.76%



Pe parcursul celor 5 rulari, performanta difera deoarece se alege mereu un set de date diferit pentru antrenare si pentru testare. In final, XGBoost are precizia cea mai buna.

### Recall

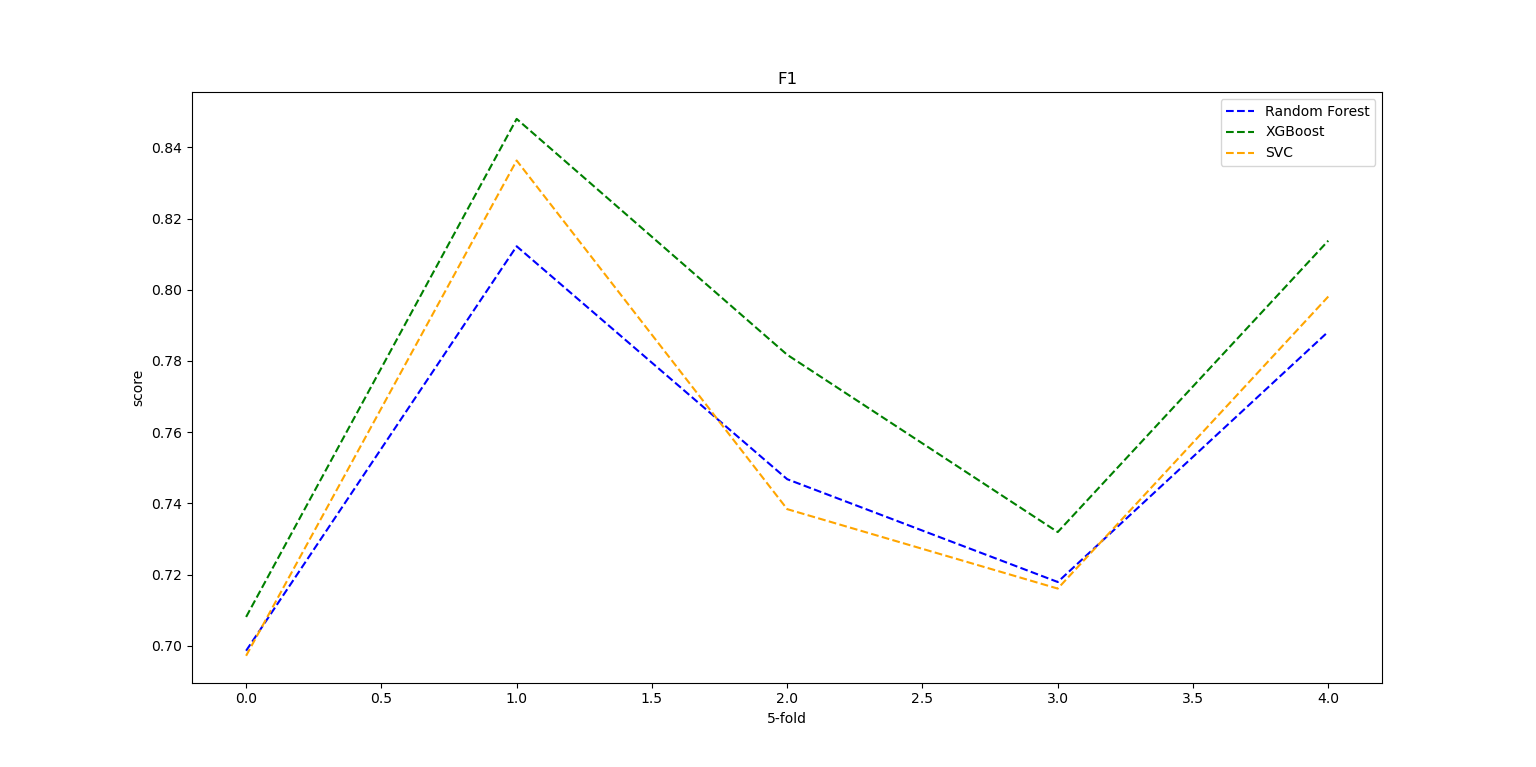
* Random Forest: 0.75%
* XGBoost: 0.78%
* SVM: 0.76%



Pe parcursul celor 5 rulari, performanta difera deoarece se alege mereu un set de date diferit pentru antrenare si pentru testare. In final, XGBoost are recall-ul cel mai bun.

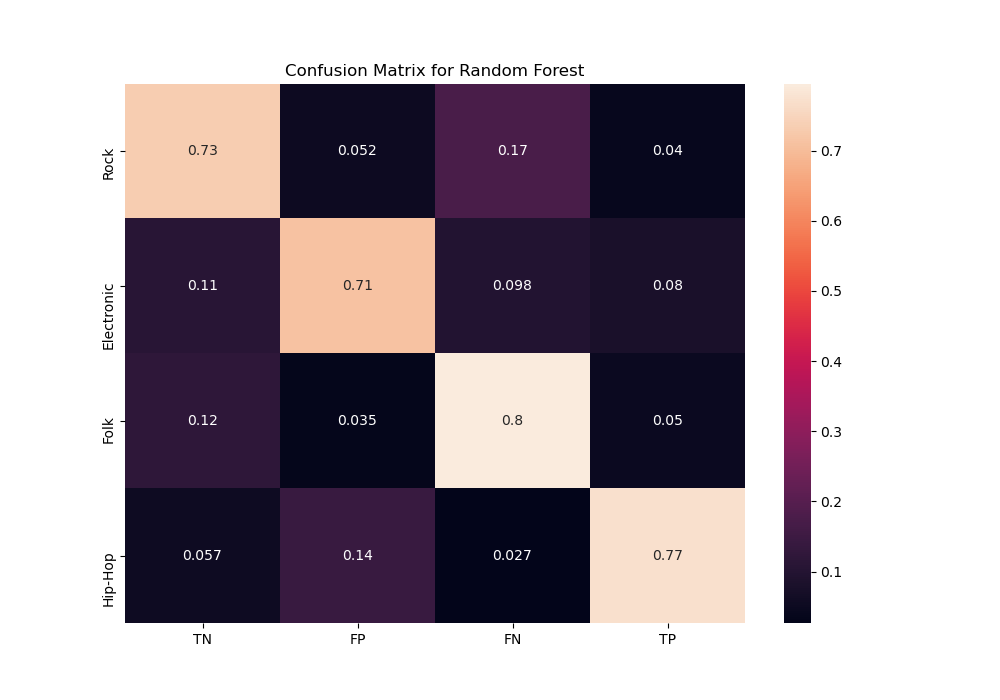
### F1

* Random Forest: 0.75%
* XGBoost: 0.78%
* SVM: 0.76%

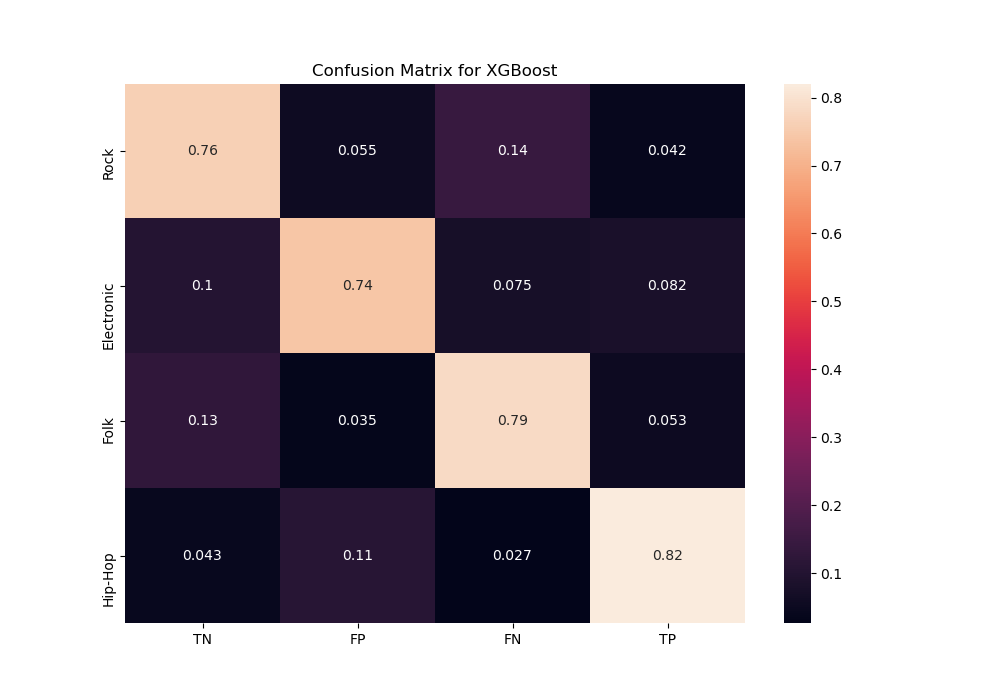


Pe parcursul celor 5 rulari, performanta difera deoarece se alege mereu un set de date diferit pentru antrenare si pentru testare. In final, XGBoost are scorul F1 cel mai bun.

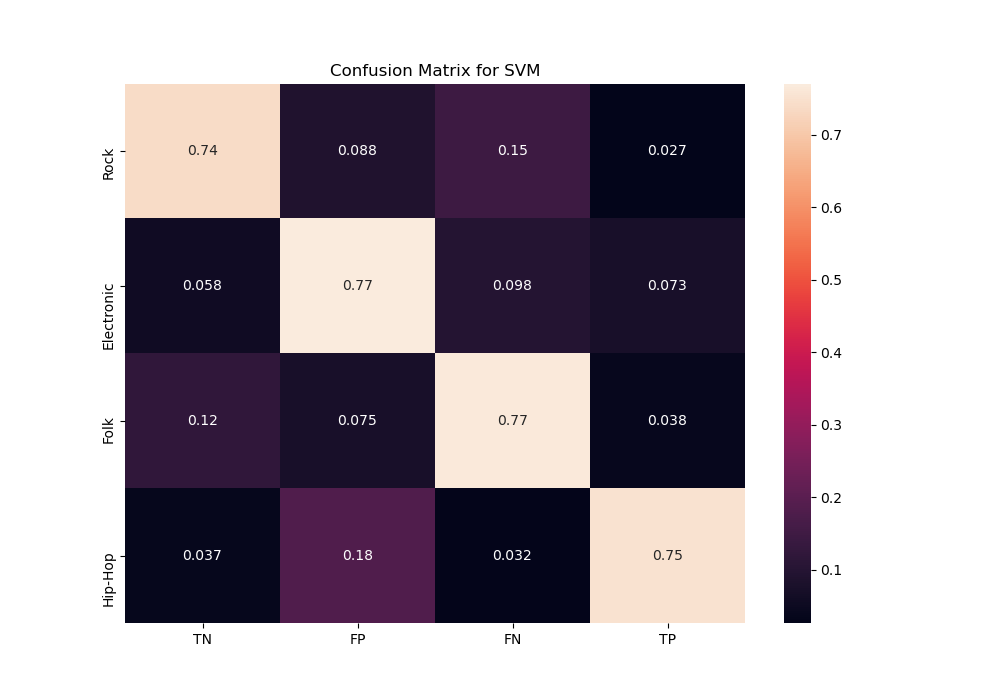
### Matricea de confuzie



* Rock: 80% din exemple sunt prezise corect
* Electronic: <1% din exemple sunt prezise corect
* Folk: 1% din exemple sunt prezise corect
* Hip-Hop: 78% din exemple sunt prezise corect



* Rock: 77% din exemple sunt prezise corect
* Electronic: 2% din exemple sunt prezise corect
* Folk: 2% din exemple sunt prezise corect
* Hip-Hop: 83% din exemple sunt prezise corect



* Rock: 75% din exemple sunt prezise corect
* Electronic: 1% din exemple sunt prezise corect
* Folk: 2% din exemple sunt prezise corect
* Hip-Hop: 76% din exemple sunt prezise corect

# Concluzie

Observand rezultate din matricile de confuzie, genurile: Rock si Hip-Hop sunt prezise corect de cele mai multe ori, pe cand celelalte doua genuri muzicale: Electronic si Folk, sunt prezise gresit foarte des.

Cel mai bun model este XGBoost, cu o acuratete de 78%.

Dezavantajul la acest model este faptul ca dureaza mult deoarece se bazeaza pe principiul de Boosting si faptul ca foloseste CPU-ul la maxim. Este, de altfel, si modelul care dureaza cel mai mult.

Pe locul doi ar fi SVM, cu o acuratete de 76%, apoi Random Forest cu aproximativ aceeasi acuratete (57%), iar K-Means nu poate fi comparat din punct de vedere al metricilor cu ceilalti algoritmi deoarece el doar clusterizeaza / grupeaza, singura metrica folosita pentru a prezice corectitudinea fiind Rand Index.