Invatare Automata - Tema 1
Clasificare

Teodor-Stefan Dutu

Universitatea Politehnica Bucuresti
Facultatea de Automatica si Calculatoare
Grupa 341C3

Cuprins

1 Introducere

Scopul temei este de a compara performantele unor algoritmi de clasificare atat supervizata (Random Forest [1], XGBoost [2], SVM [3] si Naive Bayes [4]) cat si nesupervizata (K-Means [**?**]) si se vor propune strategii pentru imbunatatirea modelelor de baza.

Astfel, fiecare algoritm va fi analizat in 2 variante: una neoptimizata, ce foloseste parametrii impliciti, din biblioteca `scikit-learn`, ai algoritmilor de mai sus, si alta optimizata prin diverse tehnici (adaugarea de noi caracteristici, tunarea hiperparametrilor etc.). Folosind *5-fold cross-validation*, am comparat modelul imbunatatit cu cel de baza, pana cand am obtinut obtine cele mai bune performante, dupa care am testat modelul optim pe un set separat de testare, iar la final am comparat rezultatele.

2 Procedura de imbunatatire a algoritmilor

In primul rand, marit numarul de caracteristici folosite de modele. Pentru acest lucru, am adaugat caracteristicilor din `audio_features` pe cele din `temporal_features`. Temandu-ma ca ar rezulta un set de date cu un numar prea mare de caracteristici, deci vulnerabil la overfitting, am ales pentru fiecare model un numar optim de caracteristici, folosind `SelectKBest` [6]. Aceasta augmentare a caracteristicilor a produs o crestere generala a tuturor scorurilor urmarite (acuratete, precizie, regasire, F1) cu aproximativ 10%. Singurul algoritm pentru care nu am augmentat caracteristicile este *SVM*, motivul fiind descris in Sectiunea 4.

Folosind aceste numere optime de caracteristici, am variat parametrii fiecarui model unul cate unul, dupa care am ales valorile optime (daca produceau o imbunatatire relevanta a rezultatelor) si le-am combinat intre ele pentru a obtine cele mai bune performante.

3 Random Forest

Modelul pe care l-am folosit este `RandomForestClassifier` [1]. Conform graficului din Figura 1, performantele algoritmului nu mai cresc de la un numar de caracteristici ≥ 500, drept care acesta este numarul de caracteristici pe care l-am ales.

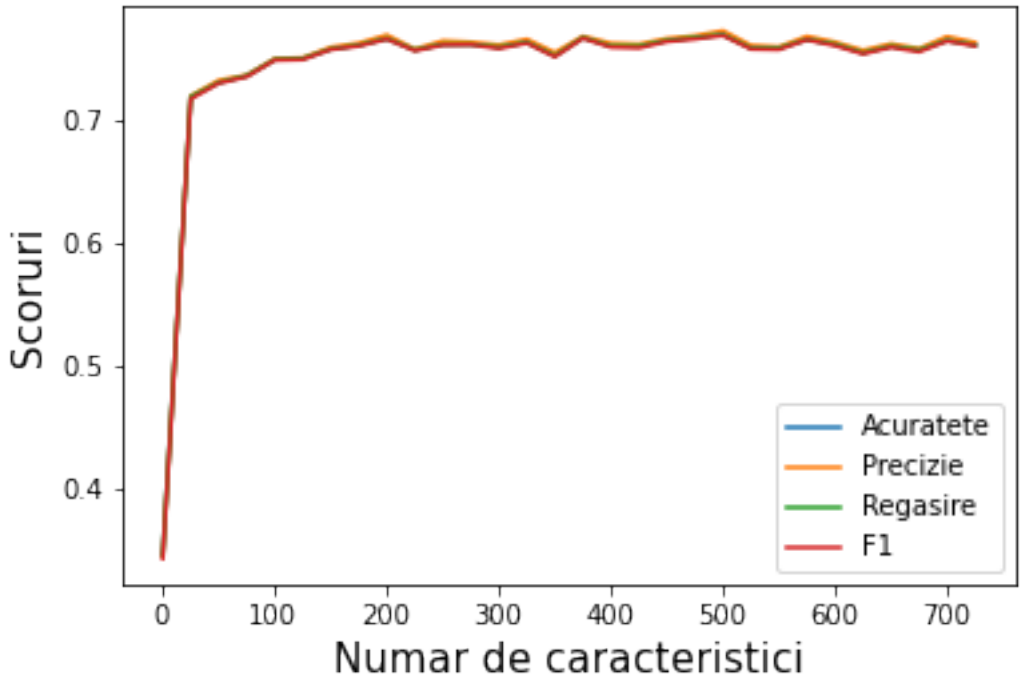


Fig. 1: Performantele modelului *Random Forest* in functie numarul de caracteristici

Variind criteriul de clasificare intre `gini` si `entropy`, am observat, conform Figurii 2, ca `gini` este criteriul superior. De asemenea, pentru a evita overfittingul, dar si underfittingul, cel mai bun criteriu de selectie a caracteristicilor din fiecare arbore si din fiecare nod este cel ce foloseste radicalul, confirm Figurii 3. De asemenea, un impact simtitor il are numarul de estimatori (arbori de decizie), a carui valoare optima se situeaza la la 250, dupa cum se poate vedea in Figura 4.

Alti parametrii, precum `max_depth` si `max_samples` nu influenteaza performantele modelului intr-o masura substantiala.

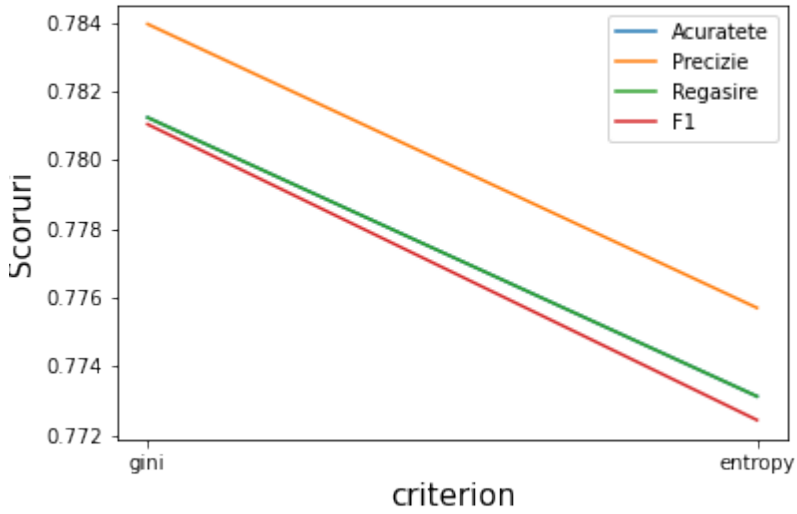


Fig. 2: Performantele modelului *Random Forest* in functie de parametrul `criterion`

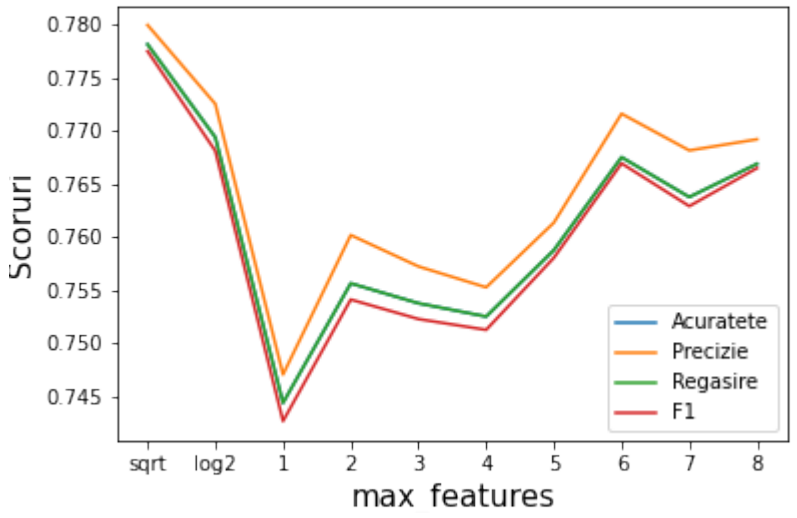


Fig. 3: Performantele modelului *Random Forest* in functie de parametrul `max_features`

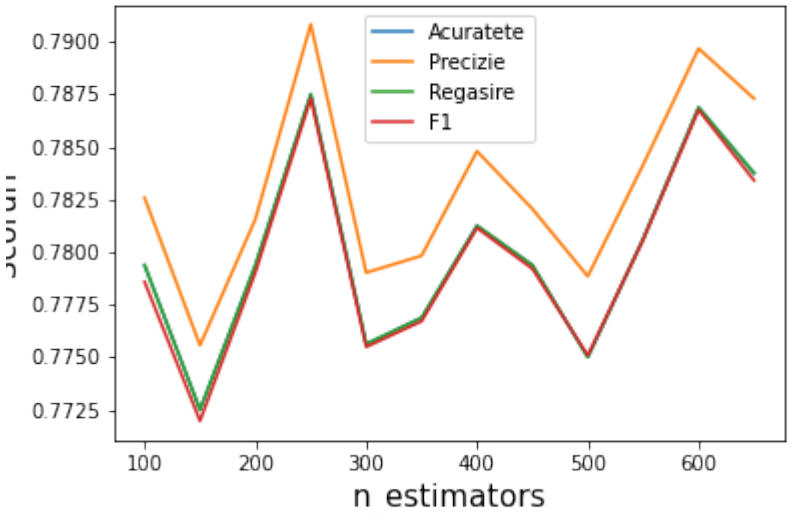


Fig. 4: Performantele modelului *Random Forest* in functie de parametrul `n_estimators`

In urma imbunatatirii hiperparametrilor si a numarului de caracteristici, performanta modelului *Random Forest* creste, in medie, cu 10%, asa cum se observa in Figura 5. De asemenea, chiar si deviatiile standard sunt la o treime din cele initiale, cu toate ca acestea deja erau mici (< 0.035) chiar si pentru modelul de baza.

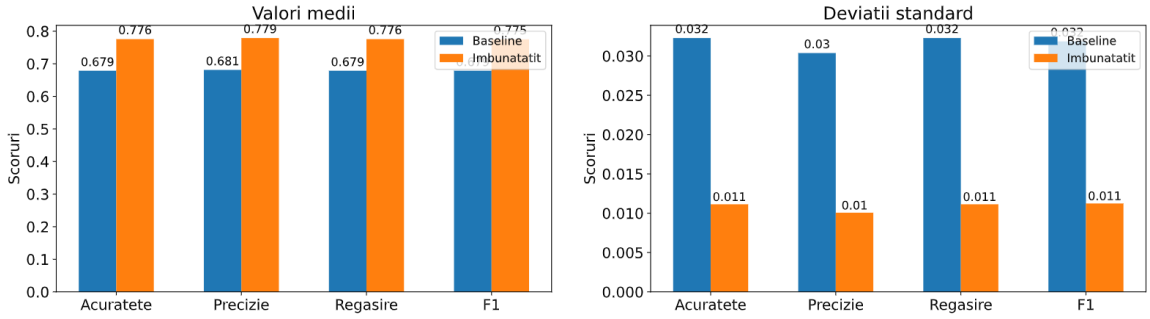


Fig. 5: Cresterea de performanta a modelului *Random Forest* imbunatatit

4 SVM

Modelul pe care l-am folosit este `SVC` [3]. Conform graficului din Figura 6, performantele algoritmului nu mai cresc de la un numar de caracteristici ≥ 500, drept care acesta este numarul de caracteristici pe care l-am ales.

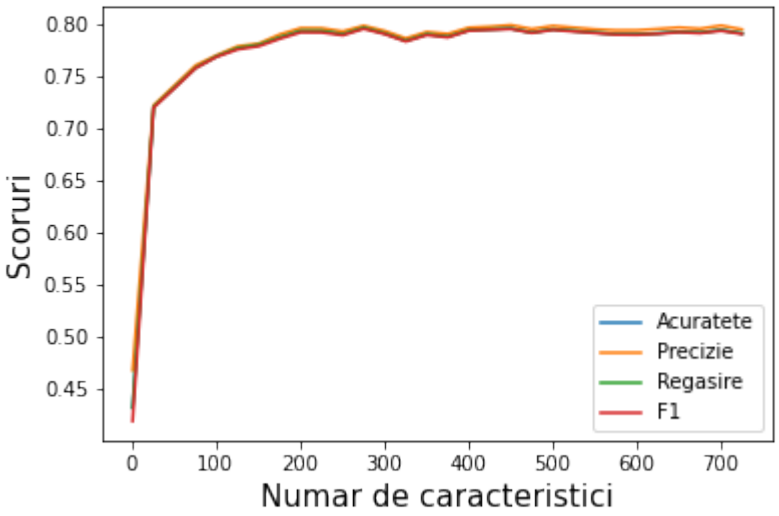


Fig. 6: Performantele modelului *SVM* in functie numarul de caracteristici

Am observat din graficul din Figura 7 ca nucleul optim (dintre cele aplicabile pe acest set de date) este cel polinomial. Astfel, am incercat sa determin gradul optim al acestuia, pe care l-am gasit ca este 4 (Figura 8).

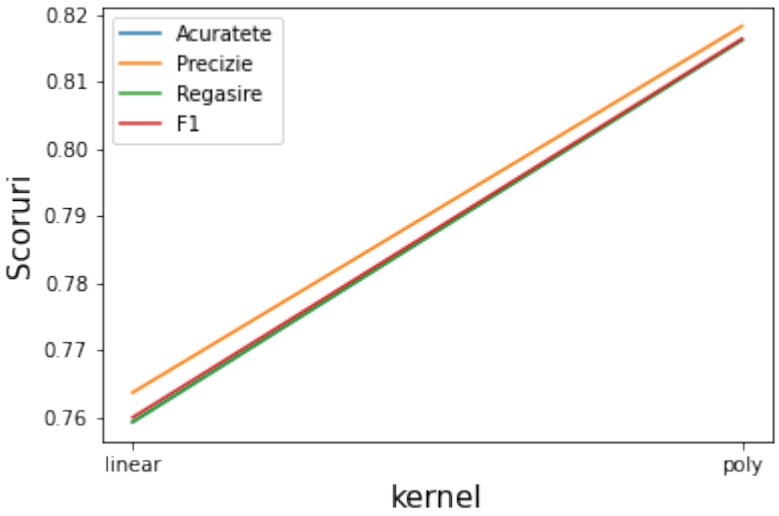


Fig. 7: Performantele modelului *SVM* in functie tipul kernelului

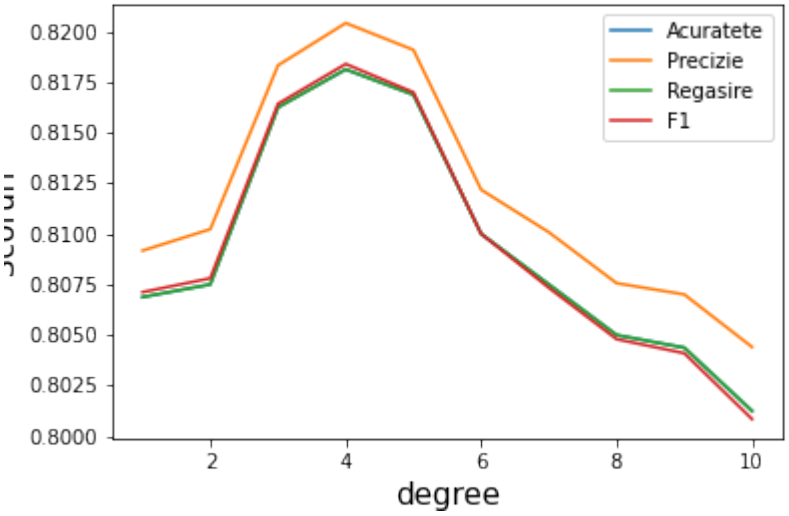


Fig. 8: Performantele modelului *SVM* in functie de gradul polinomului folosit pentru kernel

In urma imbunatatirii hiperparametrilor si a numarului de caracteristici, performanta modelului *SVM* creste, chiar si cum 50%, asa cum se observa in Figura 9. De asemenea, chiar si deviatiile standard scad pana la 0.015, cu toate ca deja erau mici (< 0.025) chiar si pentru modelul de baza.

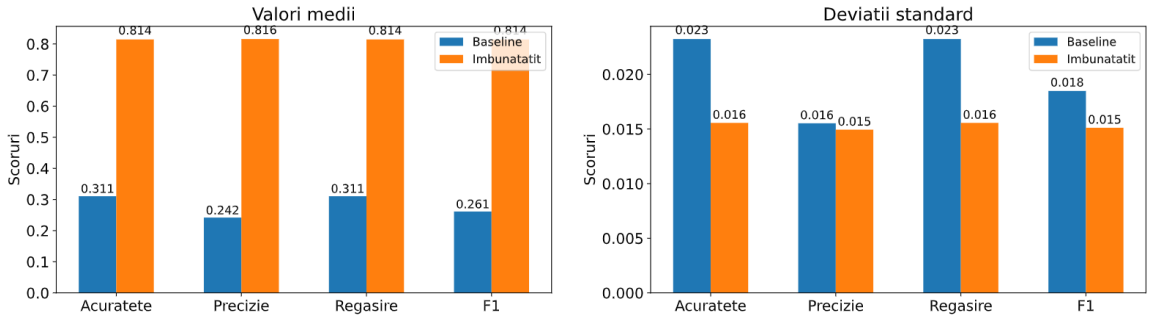


Fig. 9: Cresterea de performanta a modelului *SVM* imbunatatit

5 XGBoost

Modelul pe care l-am folosit este `XGBClassifier` [2]. Conform graficului din Figura 10, performantele algoritmului nu mai cresc de la un numar de caracteristici ≥ 500, drept care acesta este numarul de caracteristici pe care l-am ales.

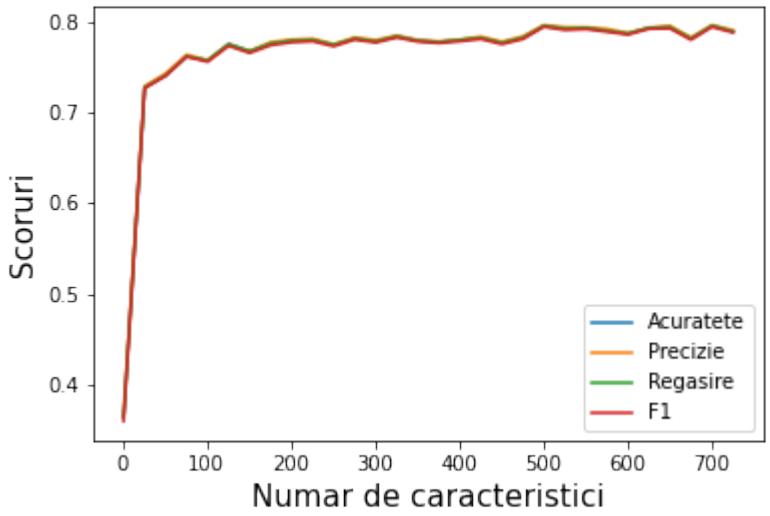


Fig. 10: Performantele modelului *XGBoost* in functie de numarul de caracteristici

Parametrii a caror modificare a produs o imbunatatire a performantelor modelului sunt: `alpha`, `eta` si `min_child_weight`. Astfel, din Figura 11, se observa ca parametrul de regularizare al normei L1, α, obtine rezultate bune cand are valoare de 0.1. Rata de invatare `eta` este optima la valoarea de 0.5, conform Figurii 12, iar valoarea de la care se vor constui noduri frunza (`min_child_weight`) este 0.7, asa cum se poate observa in Figura 13.

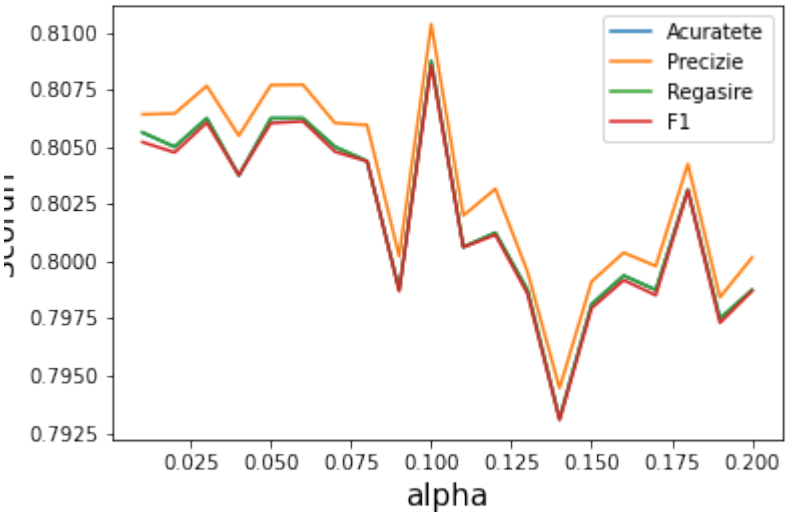


Fig. 11: Performantele modelului *XGBoost* in functie de parametrul `alpha`

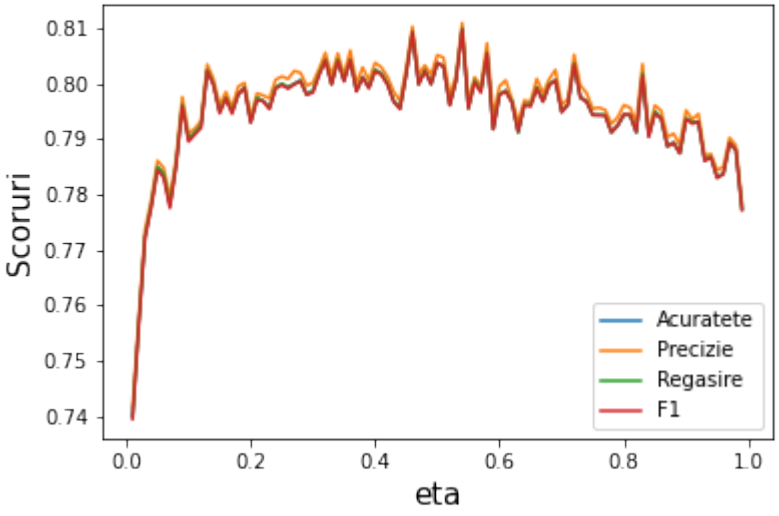


Fig. 12: Performantele modelului *XGBoost* in functie de parametrul `eta`

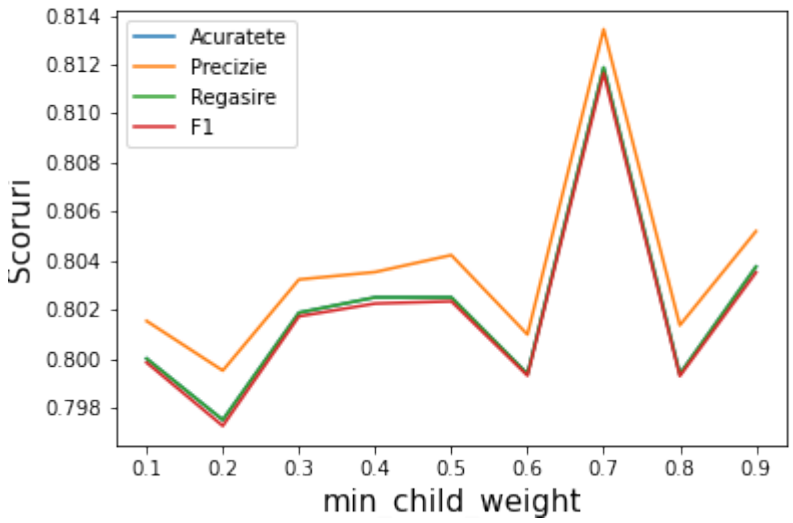


Fig. 13: Performantele modelului *XGBoost* in functie de parametrul `min_child_weight`

In urma imbunatatirii hiperparametrilor si a numarului de caracteristici, performanta modelului *XGBoost* creste, in medie, cu 13%, asa cum se observa in Figura 14. De asemenea, chiar si deviatiile standard se injumatatesc, cu toate ca deja erau mici (< 0.04) chiar si pentru modelul de baza.

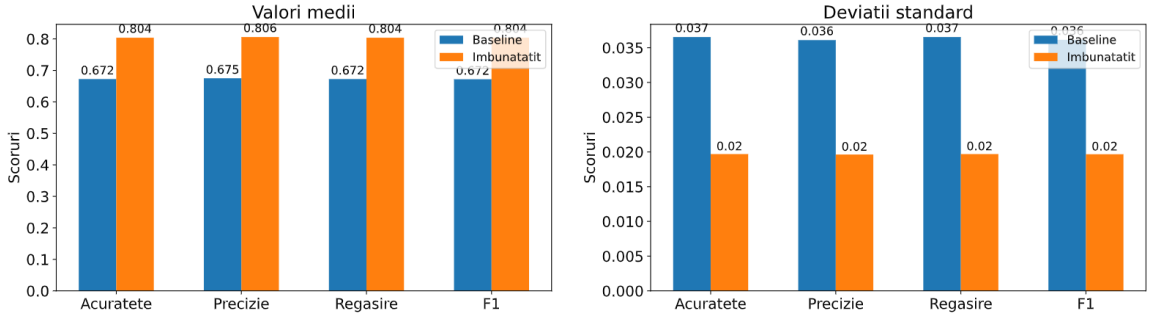


Fig. 14: Cresterea de performanta a modelului *XGBoost* imbunatatit

6 Naive Bayes

Modelul pe care l-am folosit este `ComplementNB` [4]. Conform graficului din Figura 15, performantele algoritmului nu mai cresc de la un numar de caracteristici ≥ 500, drept care acesta este numarul de caracteristici pe care l-am ales.

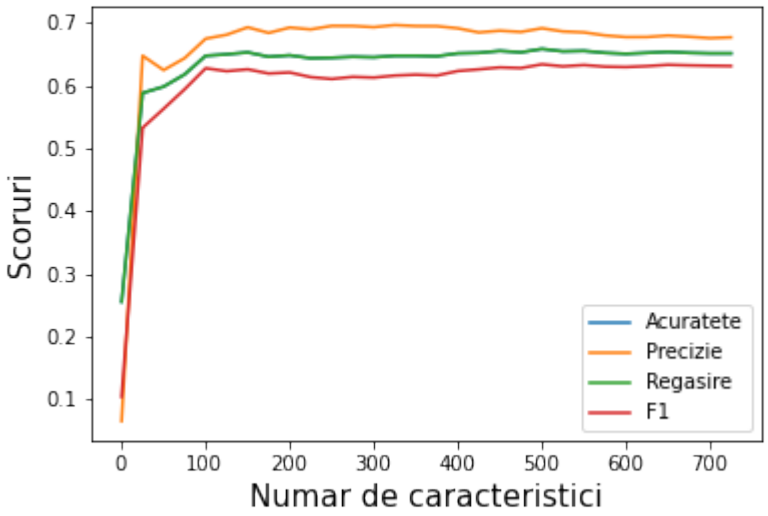


Fig. 15: Performantele modelului *Naive Bayes* in functie numarul de caracteristici

Dintre cei 2 hiperparametri ai modelului `ComplementNB`, `alpha` si `norm`, niciunul nu modifica semnificativ performantele acestuia. Din acest motiv singura imbunatatire a constat in cresterea numarului de caracteristici, iar performanta modelului a crescut, in medie, cu 10%, asa cum se observa in Figura 18. De asemenea, si deviatiile standard scad cate putin, in afara de cea pentru precizie, care se dubleaza. Totusi, pana si aceasta ramane sub valoarea de 0.03.

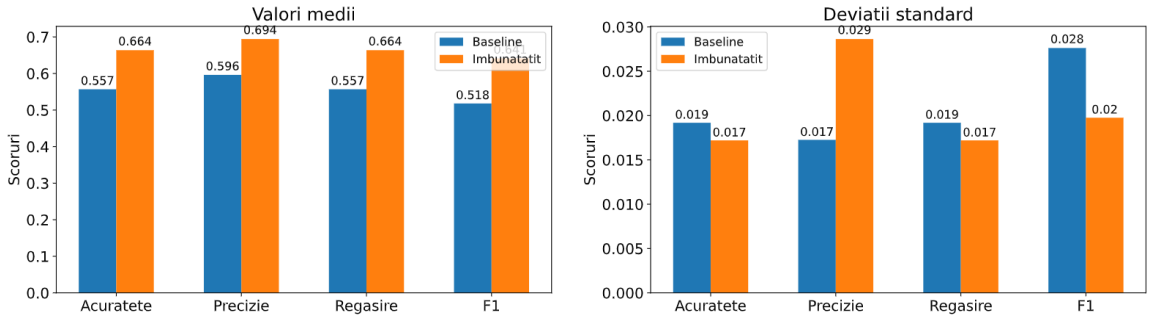


Fig. 16: Cresterea de performanta a modelului *Naive Baies* imbunatatit

7 K-Means

Modelul pe care l-am folosit este `KMeans` [5]. Conform graficului din Figura 17, performantele algoritmului nu mai cresc de la un numar de caracteristici ≥ 125, drept care acesta este numarul de caracteristici pe care l-am ales.

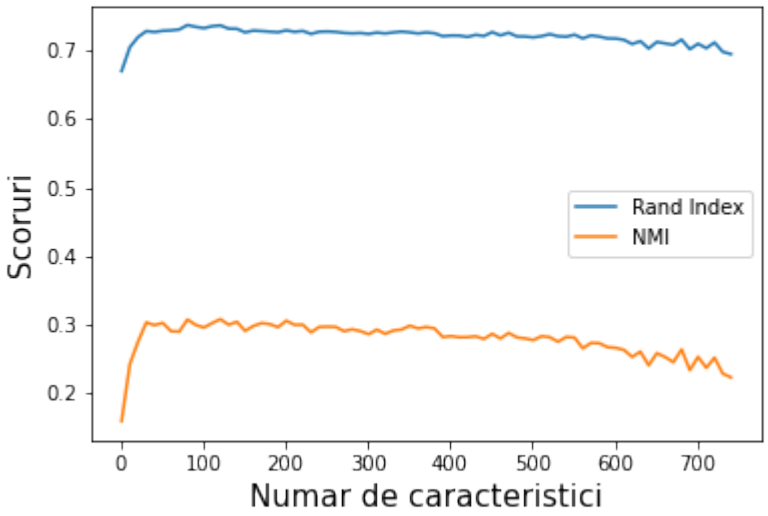


Fig. 17: Performantele modelului *KMeans* in functie numarul de caracteristici

Niciun hiperparametru nu modifica performantele modelului. Asadar, in urma imbunatatirii numarului de caracteristici, performanta modelului *KMeans* creste, in medie, cu 11%, asa cum se observa in Figura 18, iar deviatiile standard raman aproape nemodificate.

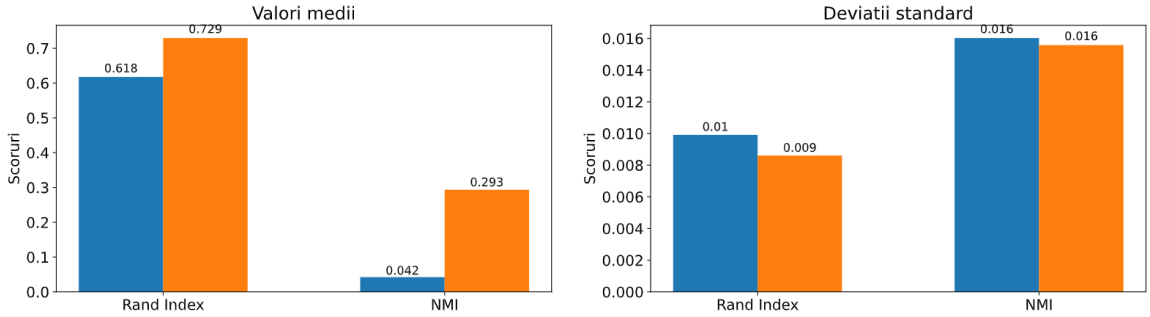


Fig. 18: Cresterea de performanta a modelului *KMeans* imbunatatit

8 Comparatii

8.1 Acuratete

Intrucat i se aplica metrici diferite, am comparat *KMeans* cu ceilalti clasificatori doar prin prisma *Rand Indexului*, pe care l-am comparat cu acuratetile obtinute de acestia. Astfel, din Figura 19 se poate observa ca aproape toti algoritmii obtin acurateti de peste 70%, cu exceptia Naive Bayes, iar cel mai bun din acest punct de vedere este *XGBoost*, urmat la o diferenta mica de *Random Forest*.

Deviatiile standard sunt si ele foarte bune (sub 0.02), dar din aceasta perspectiva cel mai performant este *Random Forest*, si nu *XGBoost*.

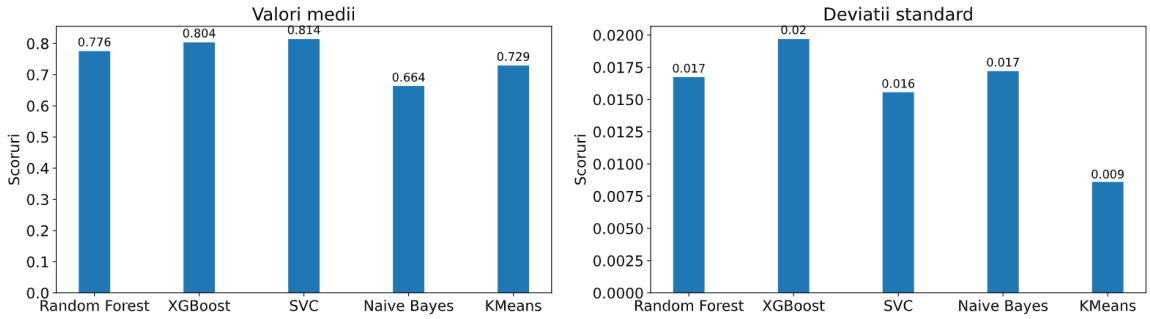


Fig. 19: Diferentele de acuratete dintre modele

8.2 Precizie

Valorile preciziei (Figura 20) sunt foarte similare cu cele ale acuratetii.

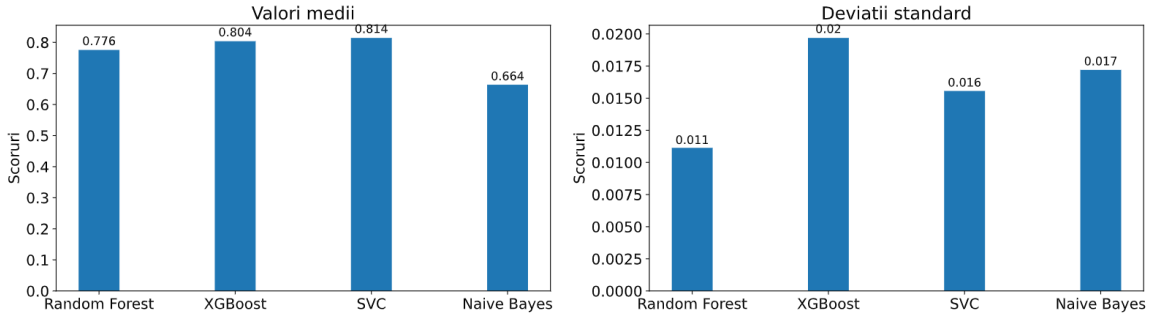


Fig. 20: Diferentele de precizie dintre modele

8.3 Regasire

Valorile regasirii (Figura 21) sunt foarte similare cu cele ale acuratetii.

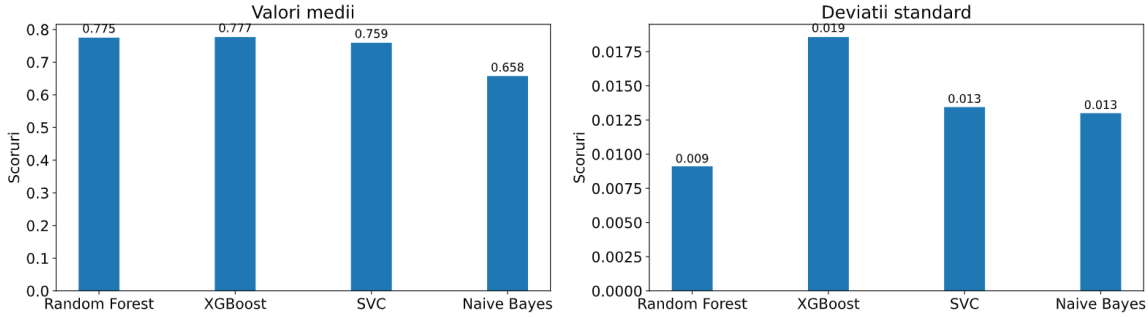


Fig. 21: Diferentele de regasire dintre modele

8.4 F1

Valorile medii ale scorului F1 (Figura 22) sunt foarte similare cu cele ale acuratetii. Deviatiile standard, insa, sunt mai mari la *Naive Bayes*.

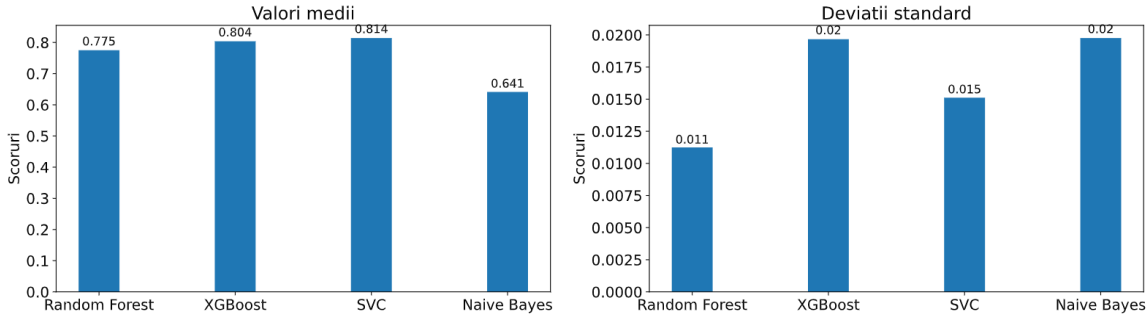


Fig. 22: Diferentele de scor *F1* dintre modele

9 Rezultate pe seturile de testare

9.1 Rezultate numerice

Rezultatele pe seturile de testare pot fi vizualizate in Tabelele 1 si 2. Se observa ca imbunatatirile inregistrate in etapa de tunare a parametrilor prin *5-fild cross validation* se pastreaza si pe seturile de testare, deci nu este vorba de overfitting.

Cele mai performante modele sunt *XGBoost* si *SVC*, ambele inregistrand scoruri de peste 81%. Cea mai buna imbunatatire, scorurile crescand mai de aproape 3 ori, se inregistreaza prin modificarea kernelului *SVM-ului* de la cel implicit (`rbf`), la unul polinomial de grad 4.

Model	**Acuratete/Rand Index**	**Precizie/NMI**	**Regasire**	**F1**
Random Forest	0.6975	0.703	0.6975	0.698
XGBoost	0.665	0.674	0.665	0.667
SVM	0.28	0.194	0.28	0.226
Naive Bayes	0.545	0.585	0.545	0.518
KMeans	0.616	0.043	-	-

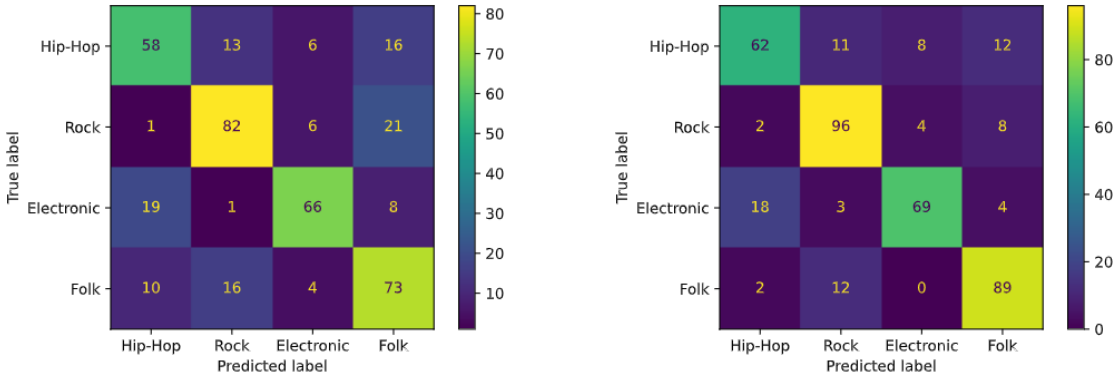
Table 1: Performantele algoritmilor in variantele de baza

Model	**Acuratete/Rand Index**	**Precizie/NMI**	**Regasire**	**F1**
Random Forest	0.79	0.79	0.79	0.787
XGBoost	0.815	0.815	0.815	0.815
SVM	0.8175	0.818	0.8175	0.817
Naive Bayes	0.6525	0.674	0.6525	0.613
KMeans	0.756	0.348	-	-

Table 2: Performantele algoritmilor in variantele imbunatatite

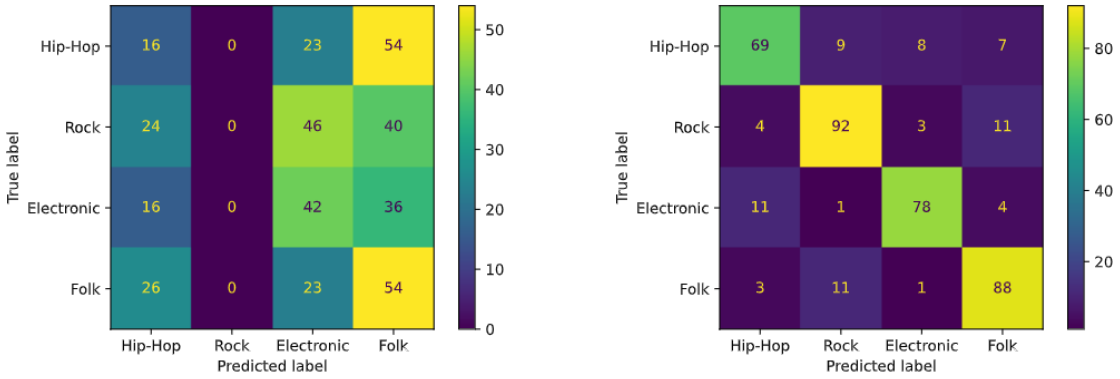
9.2 Matricele de confuzie

Se observa din matricele de confuzie din Figurile 23, 24, 25 si 26 ca toate modelele prezic destul de bine clasa *Rock* si sunt deficitare in prezicerea clasei *Hip-Hop*. Adesea rockul este confundat cu folkul si muzica electronica cu cea hip-hop.



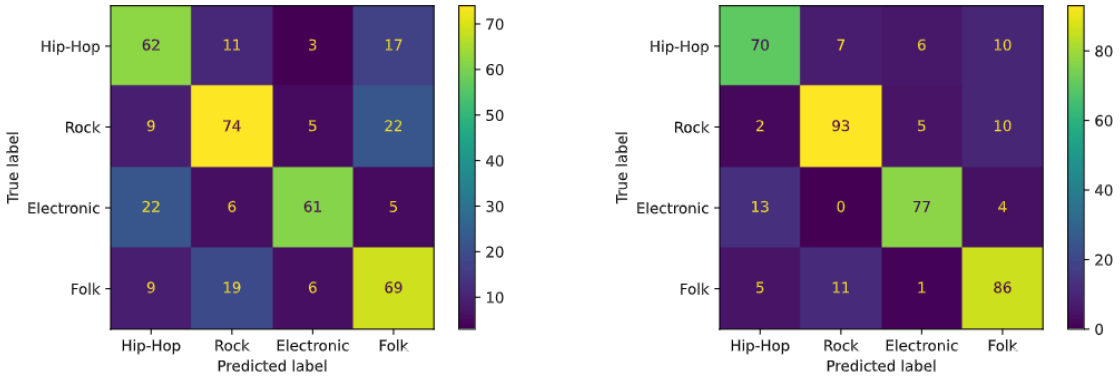
(a) Modelul de baza (b) Modelul imbunatatit

Fig. 23: Matricele de confuzie ale *Random Forest*



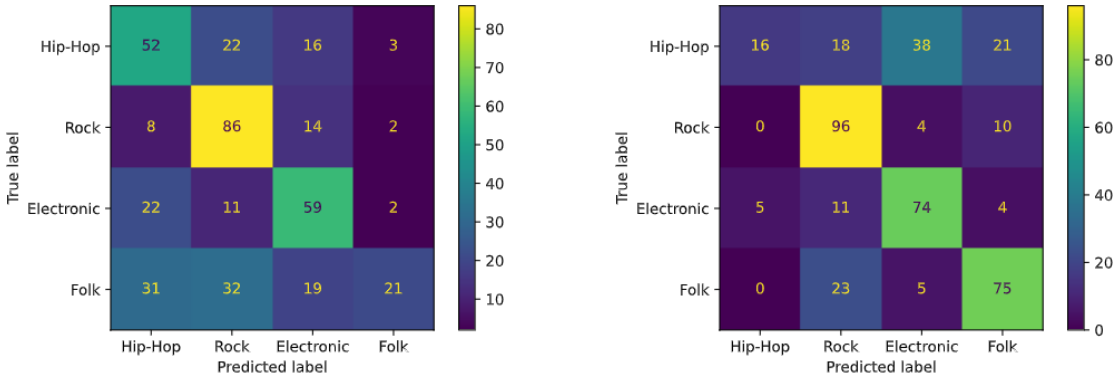
(a) Modelul de baza (b) Modelul imbunatatit

Fig. 24: Matricele de confuzie ale *SVM*



(a) Modelul de baza (b) Modelul imbunatatit

Fig. 25: Matricele de confuzie ale *XGBoost*



(a) Modelul de baza (b) Modelul imbunatatit

Fig. 26: Matricele de confuzie ale *Naive Bayes*

Bibliografie

1. *Parametrii modelului* `RandomForestClassifier`
   `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html`
   Data ultimei accesari: 11 April 2021
2. *Parametrii modelului* `XGBClassifier`
   `https://xgboost.readthedocs.io/en/stable/parameter.html`
   Data ultimei accesari: 11 April 2021
3. *Parametrii modelului* `SVC`
   `https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html`
   Data ultimei accesari: 11 April 2021
4. *Parametrii modelului* `ComplementNB`
   `https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.ComplementNB.html`
   Data ultimei accesari: 11 April 2021
5. *Parametrii modelului* `KMeans`
   `https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html`
   Data ultimei accesari: 11 April 2021
6. *Documentatia pentru* `SelectKBest`
   `https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html`
   Data ultimei accesari: 11 April 2021
7. *Codul pentru scorul* `silhouette`
   `https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_silhouette_analysis.html`
   Data ultimei accesari: 11 April 2021