**DOCUMENTATIE**

**GHADAMIYAN LIDA 312**

**1) Alegerea clasificatorului**

Am ales sa folosesc MultinomialNB, in urma mai multor runde de selectie. Incercarie au fost cu urmatorii clasificatori: SVC, Logistic Regression, KNN, Random Forest Classifier, MLP, perceptron si Logistic Regression CV. Dupa ce am testat aceste modele pe datele de validare, i-am ales pe cei mai buni 4 pentru a fi testati si pe platforma, obtinand urmatoarele rezultate :

1. LogisticRegression – 0.70798

2. MultionamialNB – 0.70419

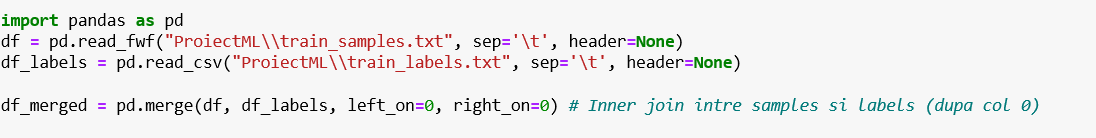
3. MLP – 0.68650

4. SVM(C=10) – 0.64614

In continuare, am incercat sa fac si un ansamblu cu toate modelele, obtinand un scor de 0.70228, motiv pentru care am renuntat la ansamblu, gandindu-ma ca este mai efficient sa caut cel mai bun claificator.

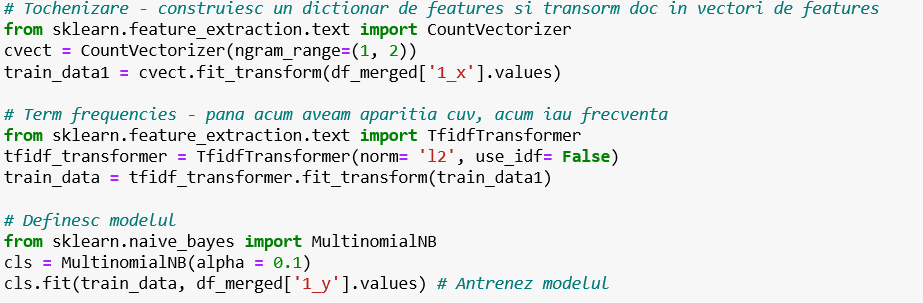
In urma acestor rezultate am continuat sa lucrez cu MultinomialNB si LR. Codul a fost acelasi (citirea si prelucrarea datelor), singurul lucru diferit fiind doua linii de cod ( importarea modelului si definirea acestuia, iar dupa grid search si parametrii de la countVectorizer si tfidfTransformer).

Citirea a fost efectuata cu functia pd.read\_fwf(), deoarece cu pd.read\_csv() nu se citeau toate propozitiile, iar cu alte tipuri de citire nu aveam avantajele celor din pandas. Apoi, am folosit pd.merge() pentru a corela propozitiile cu id-urile, printr-un procedeu de tipul inner join intre cele doua data frame uri (a unit cele doua structuri dupa coloana indicata –id-, sortand, apoi alegand liniile cu id care coincide). Pentru varianta finala am folosit ca date de antrenare datele din fisierele de train si cela de validare, concatenand fisierele intr-un set de date.

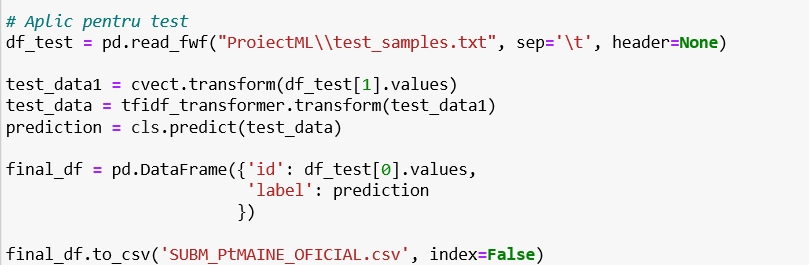


In vedere prelucrararii datelor am utilizat CountVectorizer pentru a calcula aparitia cuvintelor, apoi TfidfTransformer pentru a transforma aparitia in frecvente, impartind datele obtinute dupa CV prin numarul total de cuvinte. Dupa aceasta prelucrare, aplic clasificatorului ales.

Pentru aceasta parte a codului am urmarit urmatorul tutorial: https://scikit-learn.org/stable/tutorial/text\_analytics/working\_with\_text\_data.html

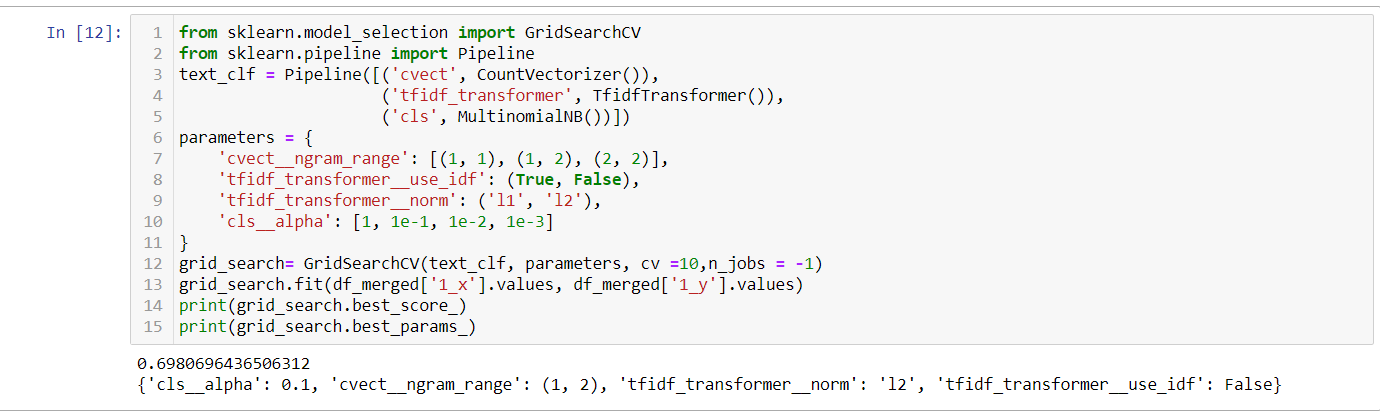


In final, am citit datele pentru test, am aplicat acelasi procedeu de mai sus pentru prelucrare, cu exceptia faptului ca am folosit doar transform, nu si fit.



**2) Alegerea parametrilor**

Pentru alegerea parametrilor am folosit Pipeline cu Gridsearch pentru modelele alese, in felul urmator.

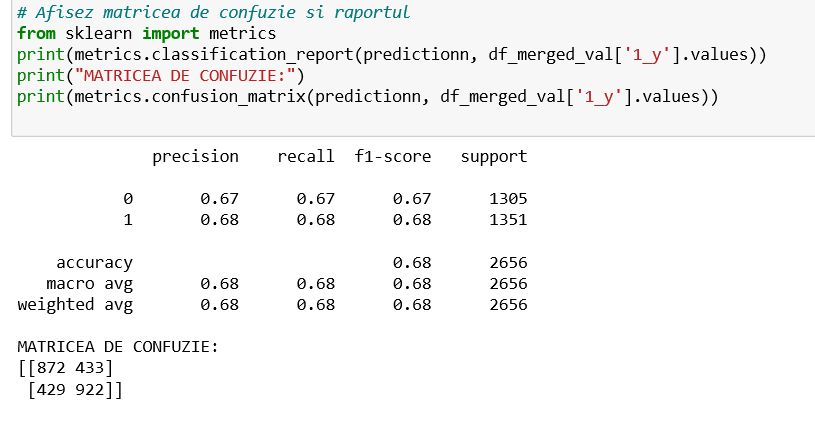


Astfel, am obtinut parametrii din figura atasata, iar cel mai bun scor pentru acestia este 0.6980 (pe datele de antrenare definite prin concatenarea celor de train si validare). Acesta a fost scorul dupa care m-am ghidat foarte mult in alegerea acestui model ca varianta finala.

**3) Matrice de confuzie si f1\_score**

Pentru a afisa matricea de confuzie si un raport cu precizie, recall si f1\_score am folosit metrics din sklearn.

Pentru MultinomialNB am obtinut urmatoarele rezultate:



Aceste procedee le-am facut si pentru LogisticRegression, obtinand rezultate mai slabe. Astfel ca, pentru cele doua submisii finale am ales MultinomialNB, deorece a avut cele mai bune rezultate pe datele de validare, si LogisticRegression, deoarece a avut cele mai bune rezultate pe platforma (si pentru ca puteam sa aleg doua rezultate finale ☺ )