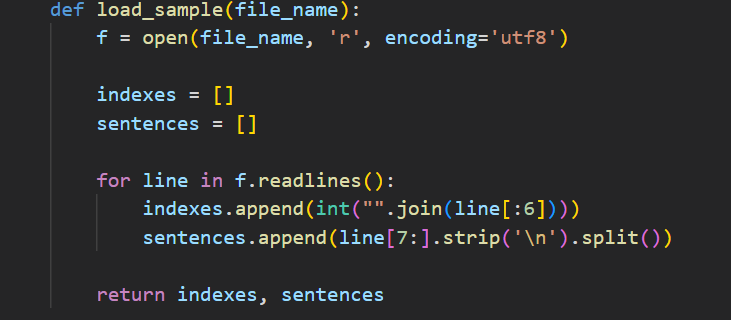
Documentatie Proiect Machine Learning

Cuciureanu Dragos-Adrian

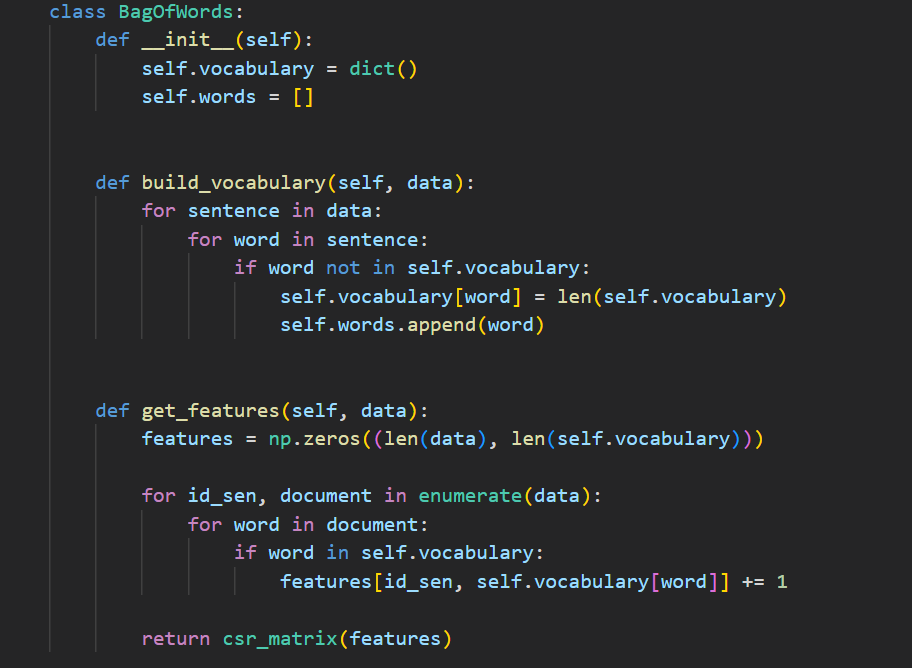
Submisiile mele finale de pe Kaggle sunt: proiect\_ma\_crossover\_v4 si proiect\_ma\_crossover\_v5. Restul de fisiere py atasate au fost ajutatoare (pentru determinarea celor mai buni parametrii si crearea matricelor de confuzie). Am atasat in forlder doar modelele pentru care am luat acurateti de peste ~70%.

In proiectul de machine learning initial am testat urmatoarele modele: Adaboost, BernoulliNB, ComplementNB, Decision Tree, Gradient Boost, Linear SVC, MultinomialNB, Neural Network, NUSVC, Random Forest, SVC si XGBoost.

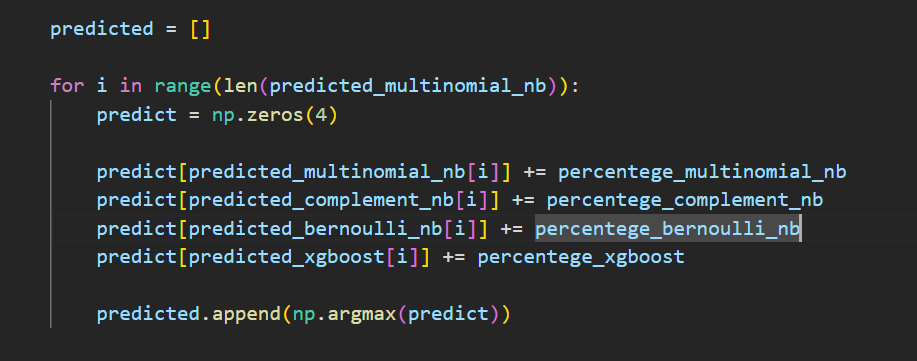
Citirea datelor am realizat-o intr-un clasic de python (nefolosind np.loadtxt).



Reprezentarea caracteristicilor am realizat-o initial cu Bag Of Words (BOW) realizat la labolatorul de machine learning, apoi am testat Count Vectorizer, insa BOW a avut rezultate mai bune pentru mine. BOW am incercat pentru toate modelele enumerate mai sus, iar Count Vectorizer am testat doar pentru: BernoulliNB, ComplementNB , MultinomialNB si Gradient Boost. Dupa aceste teste am incercat sa optimizez functia de extragere a carateristicilor in mai multe moduri. Prima data am incercat sa trec in loc de frecventa cuvintelor, lungimea acestora inmultita cu frecventa la care apar, insa a avut o acuratete putin mai scazuta decat BOW-ul initial, dupa aceasta am incecat sa iau doar cele mai importante n caracteristici, insa cu acelasi rezultat. Apoi pentru o performanta mai buna am sparsat matricea de caracteristici pentru a reduce timpul de executie. In aceasta ultima iteratie a ramas si la finalul proiectului.

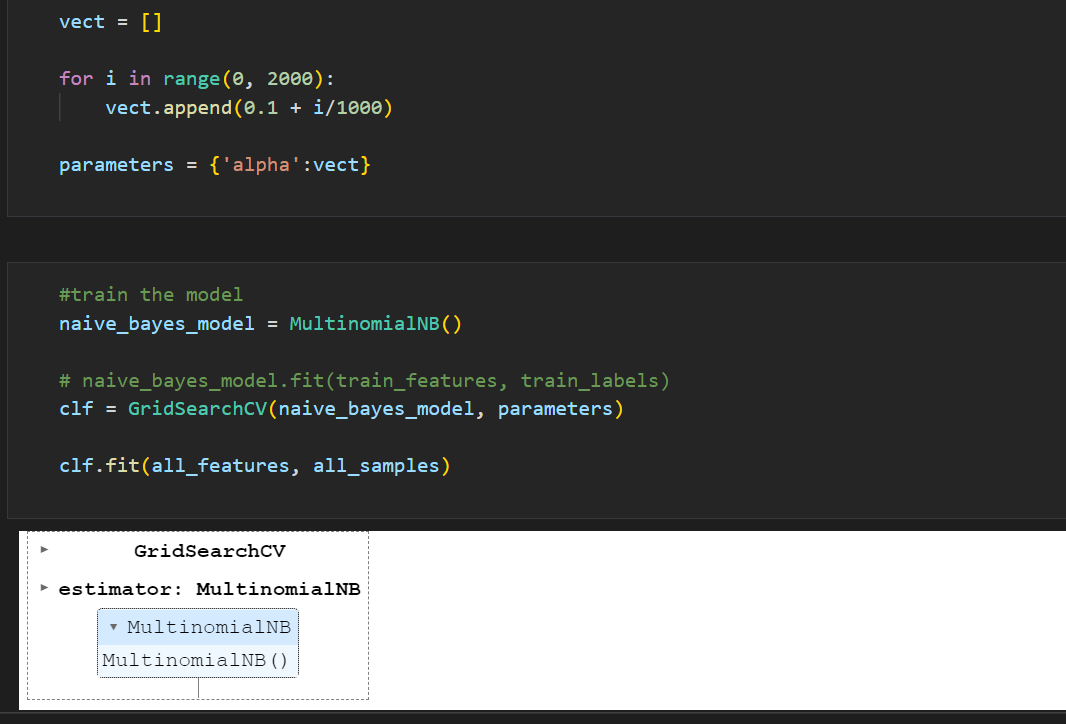


Dupa aceasta etapa am incercat sa identific care sunt cele mai bune modele pentru a incerca sa le optimizez parametrii si sa le fac crossover dupa aceea. Cele mai bune 5 modele in cazul meu au fost: BernoulliNB, ComplementNB, MultinomialNB, Neural Network si XGBoost, astfel in prima iteratie am facut crossover intre aceste 5 modele fara sa le fac optimizez parametrii, insa rezultatul a fost destul de bun si fara aceasta. Dupa mai multe teste realizate manual am ajuns sa gasesc parametrii mai buni printr-o pseudo cautare binala (spre exemplu pentru modelele de tip Naïve Bayes am modificat parametrul alpha pana amobtinut rezultate cat mai bune initial pornind de la 1 -> 0 -> 0.5 -> 075, iar la primele iteratii aceste modele au avut alpha = 0.75). La inceput pentru crossover am realizat predictul pentru fiecare model, realizam un fel de vector de frecventa pentru rezultatele de labeluri si apoi luam np.argmax dintre acestea, totusi am obervat ca o problema destul de recurenta era ca vectorul de frecventa sa fie sub forma 2-2-1, si apoi se selecta primul label. Pentru a face acest lucru mai putin probabil am ramas cu doar 4 modele (am renuntat la Neural Network, celelalte modele luau si fara crossover aproximativ 73% accuracy, pe cand acest model avea putin mai putin, pe la 71-72%). Inca mai putea exista cazul in care 2 labeluri sa aiba valoarea 2, iar al treilea 0, asa ca am decis ca in loc sa adaug cate 1 pentru fiecare predict sa adaug valoarea acccuracy-ului (chair daca acesta era overfitted, deoarece am antrenat si pe train si validation data).



Apoi am realizat mai mult “parameter tuning”, dar de data aceasta am folosit GridSearchCV (tehnic tot ce a trebuit sa fac a fost sa realizez un map cu parametrii si cu valorile pe care doresc sa le atribui, iar la final am folosit metodele best\_score\_ si best\_params\_ pentru a determina cel mai bun scor cu parametrii cu care s-a obtinut acesta). Cele doua submisii finale pe care le -am ales pentru Kaggle difera la parametrii prin modul in care am ales sa-mi creez vocabularul din BOW, in prima varianta (si cea care a avut acuratete mai mica) am ales sa creez vocabularul doar cu cuvintele din train samples, iar in a doua varianta cu cuvintele din train si validation samples. Cum toate cele 3 modele Naive Bayes rulau foarte rapid am putut sa realizez optimizarea parametrilor pana la a treia zecimala.

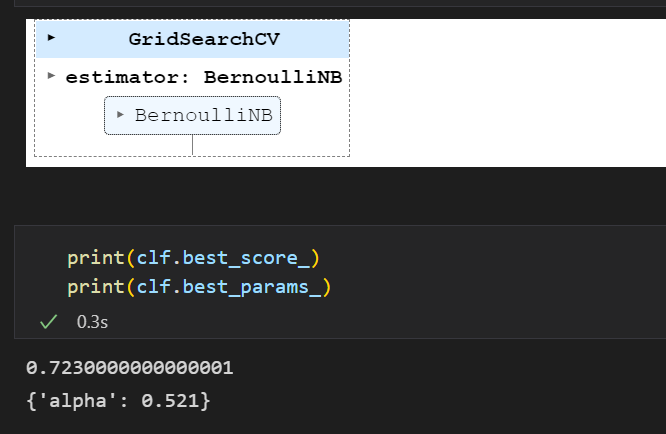
Asa arata maparea parametrului „alpha” la cele trei Naive Bayes:



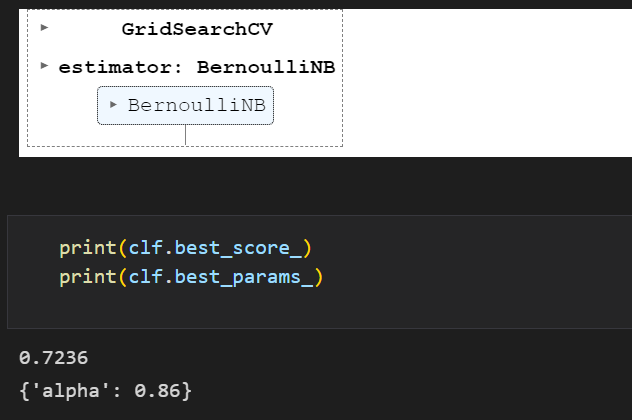
In continuare voi prezenta cel mai bun scor cu cel mai bun parametrii, prima poza de la fiecare model fiind cu vocabularul realizat doar din train samples, iar a doua poza cu train si validation samples.

BernoulliNB:

Caz 1:

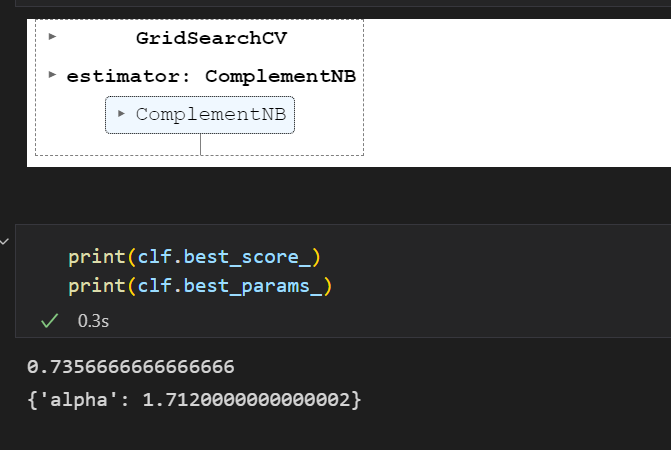


Caz 2:

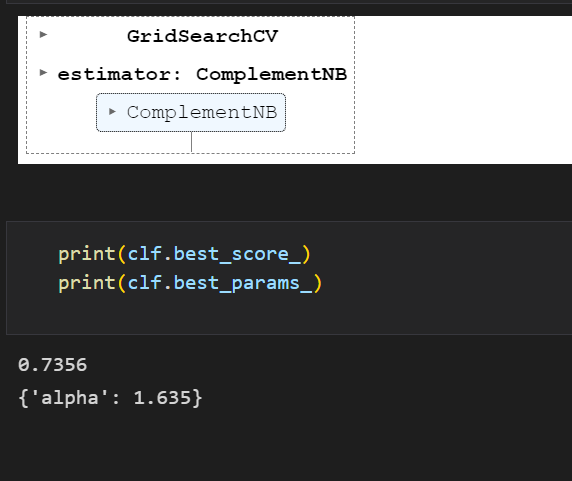


ComplementNB:

Caz 1:

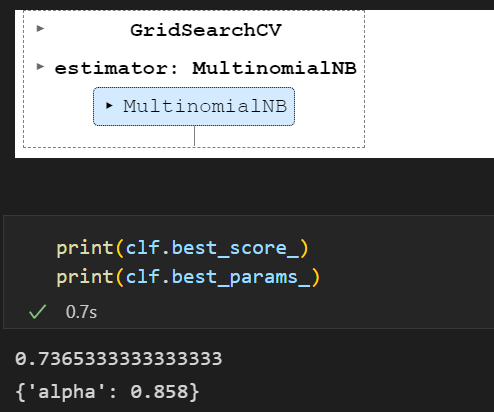


Caz 2:

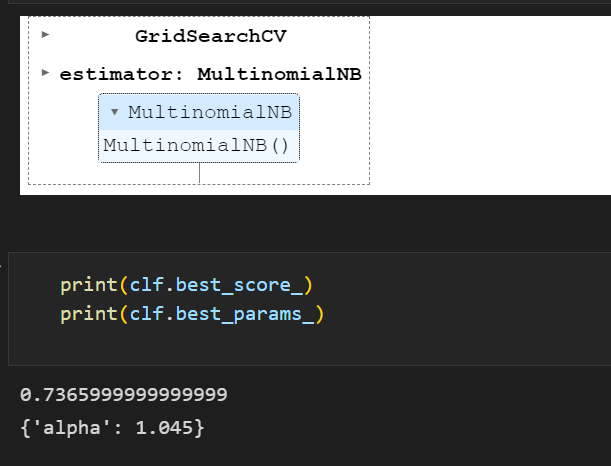


MultinomialNB:

Caz 1:

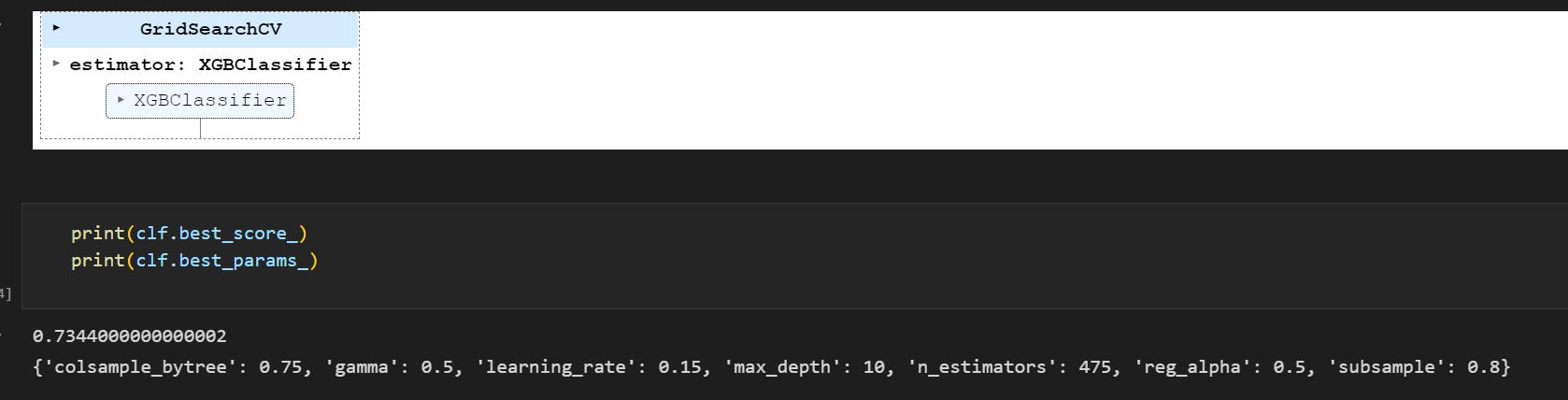


Caz 2:

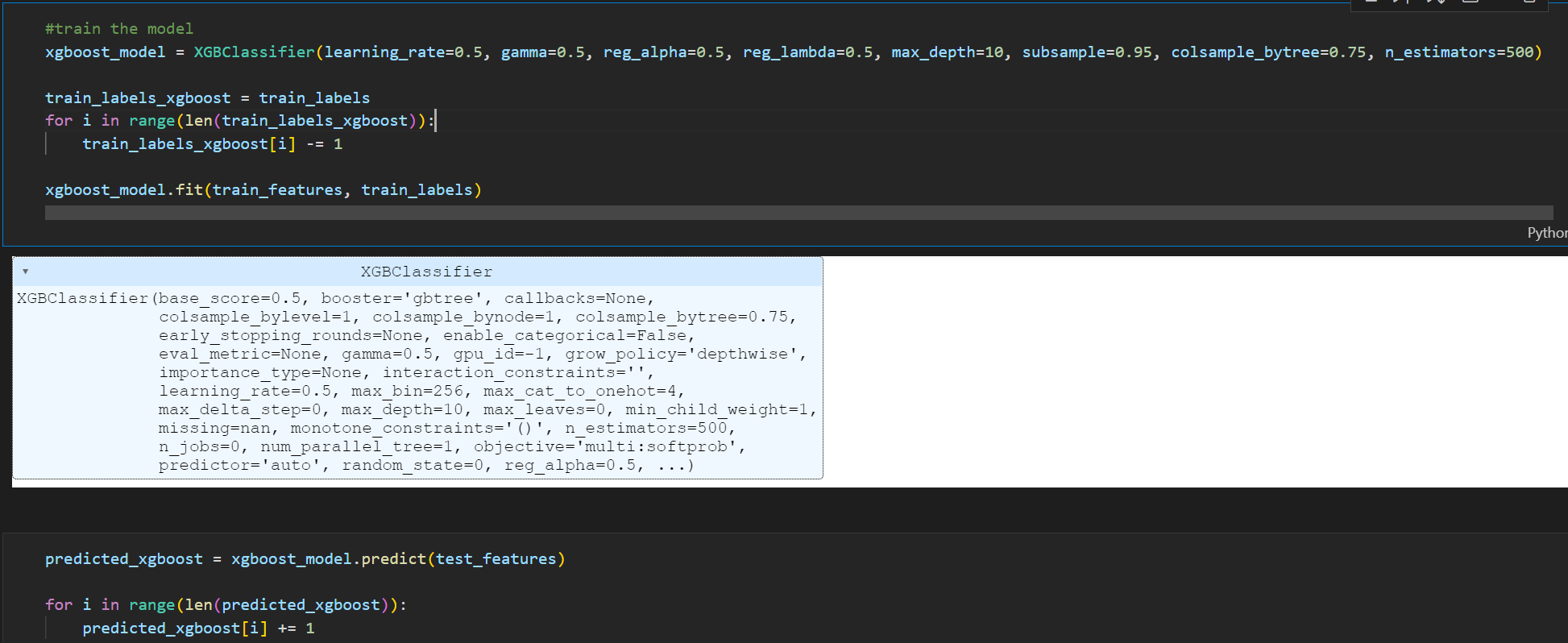


Pentru modelul XBGoost a fost putin mai greu deoarece are mult mai multi parametrii si ruleaza mai incet decat Naive Bayes. Pentru XGBoost am modificat urmatorii parametrii: learning\_rate, gamma, reg\_alpha, reg\_lambda, max\_depth, subsample, colsample\_bytree, n\_estimators. Ultima submisie pe care am facut-o pentru proiect a avut parametrii cei mai optimizat si a luat rezultatul real cel mai bun (doar ca am uitat sa o bifez pentru fina), asa ca pentru XGBoost o sa incarc tot doua poze cu iteratiile prin care a trecut, deoarece intre submisii am pastrat aceiasi parametrii o data.

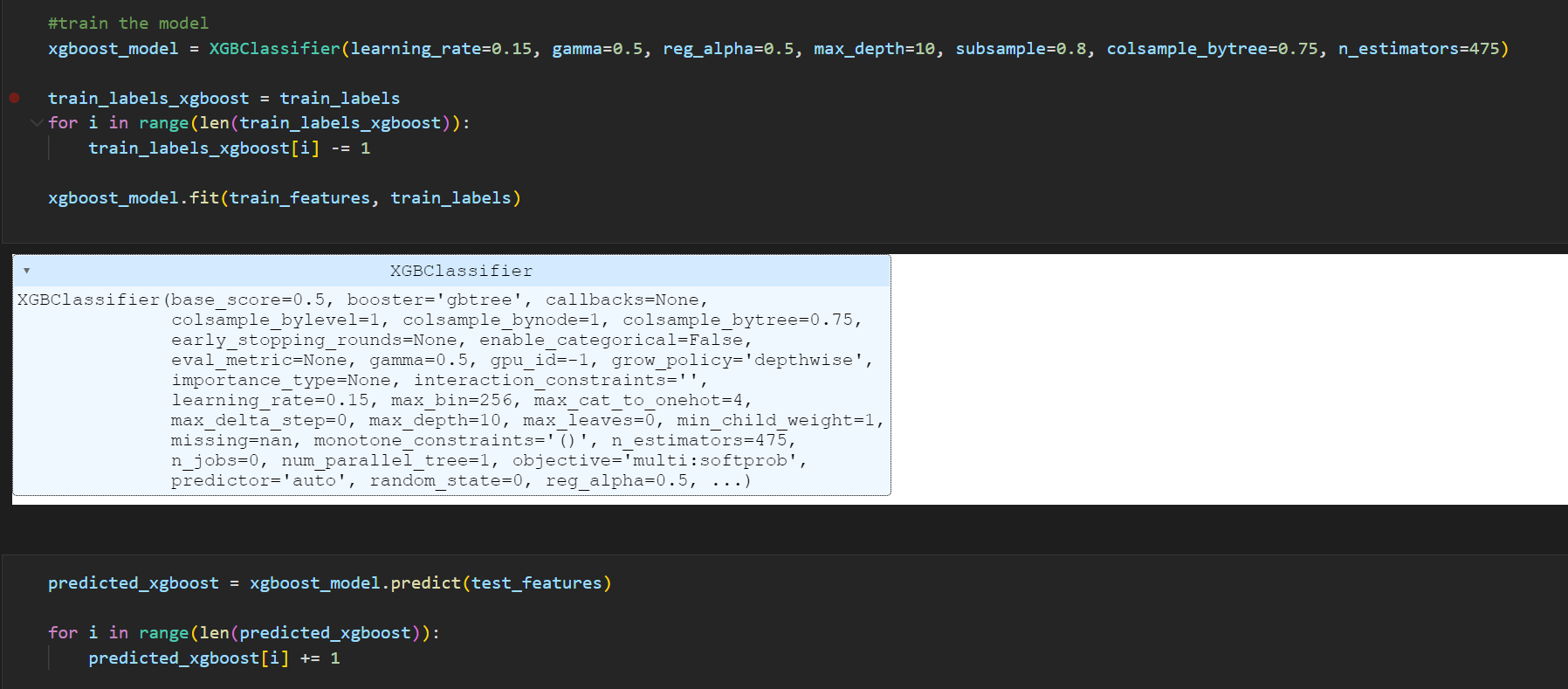
Optimizarea parametrilor:



Parametrii 1:

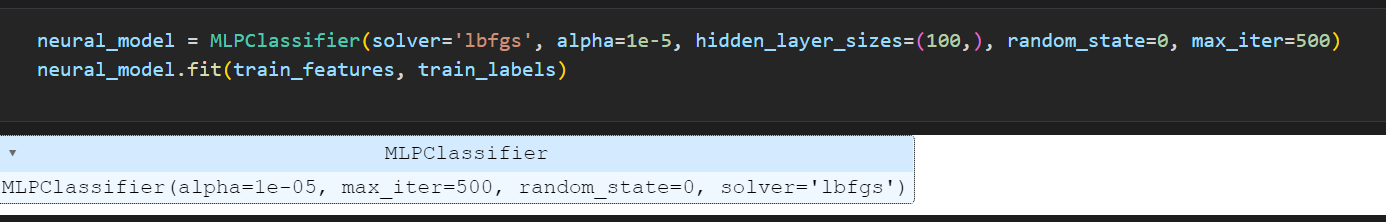


Parametrii 2:



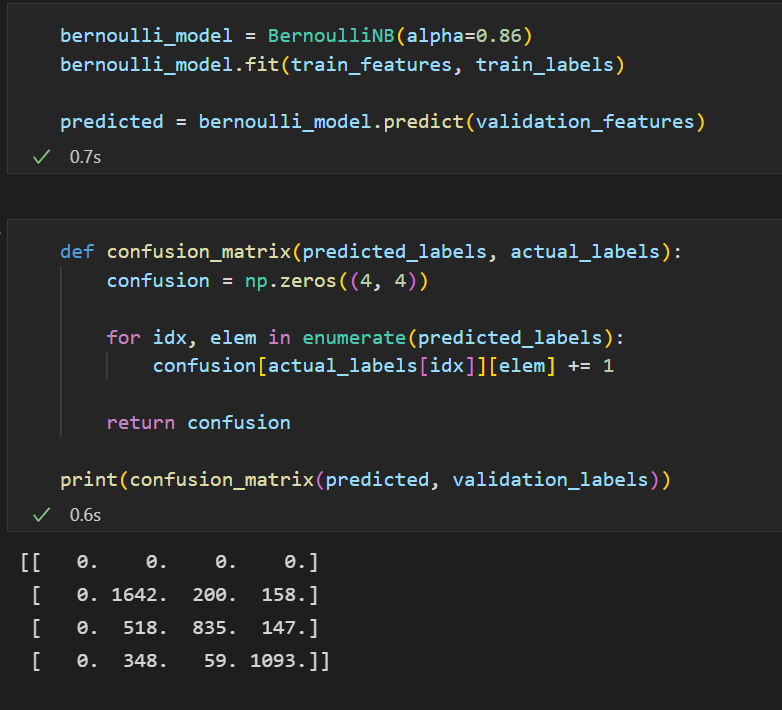
Modelul XGBoost accepta ca labels doar valori cu valori incepand de la 0, asa ca a trebuit mai intai sa micsorez toate lebelurile, iar apoi sa le incrementez inapoi.

Ar trebui amintit ca am optimizat si Neura Network, chiar daca nu l-am folosit in submisiile finale.

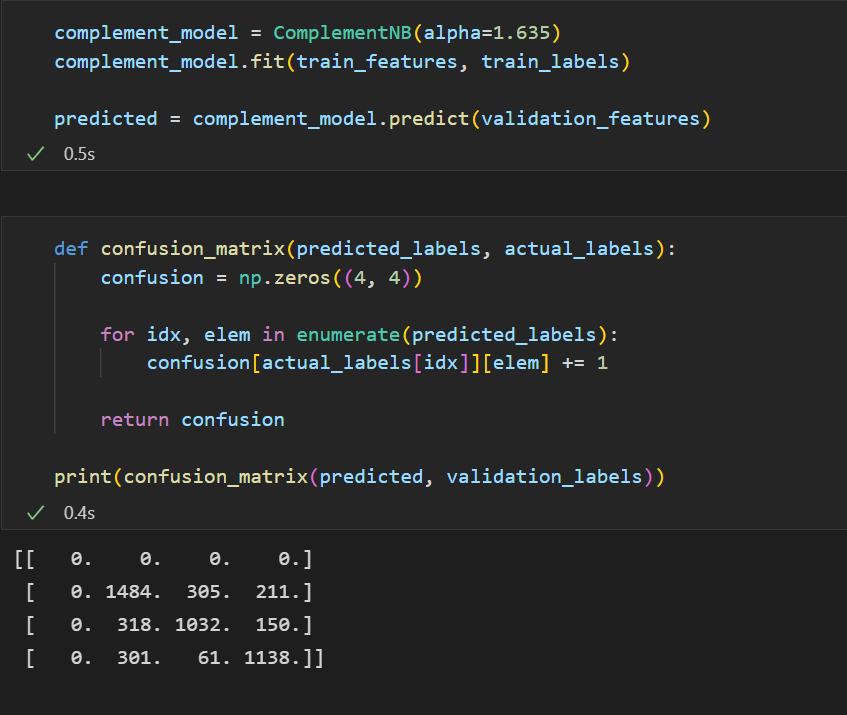


Voi afisa matricile de confuzie pentru cele 4 modele (Naive Bayes-urile si XGBoost) doar pentru cei mai buni parametrii pe care i-am gasit.

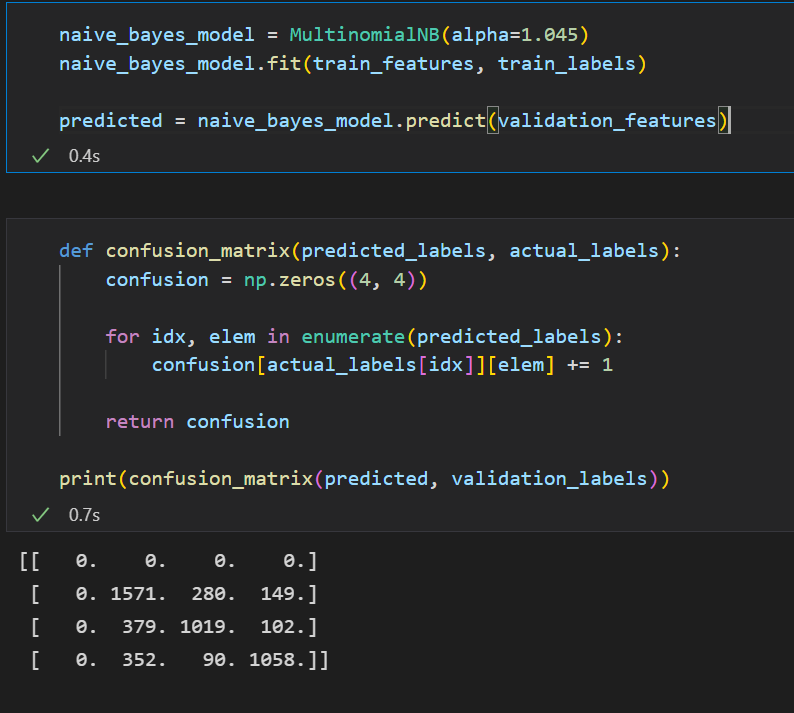
BernoulliNB:



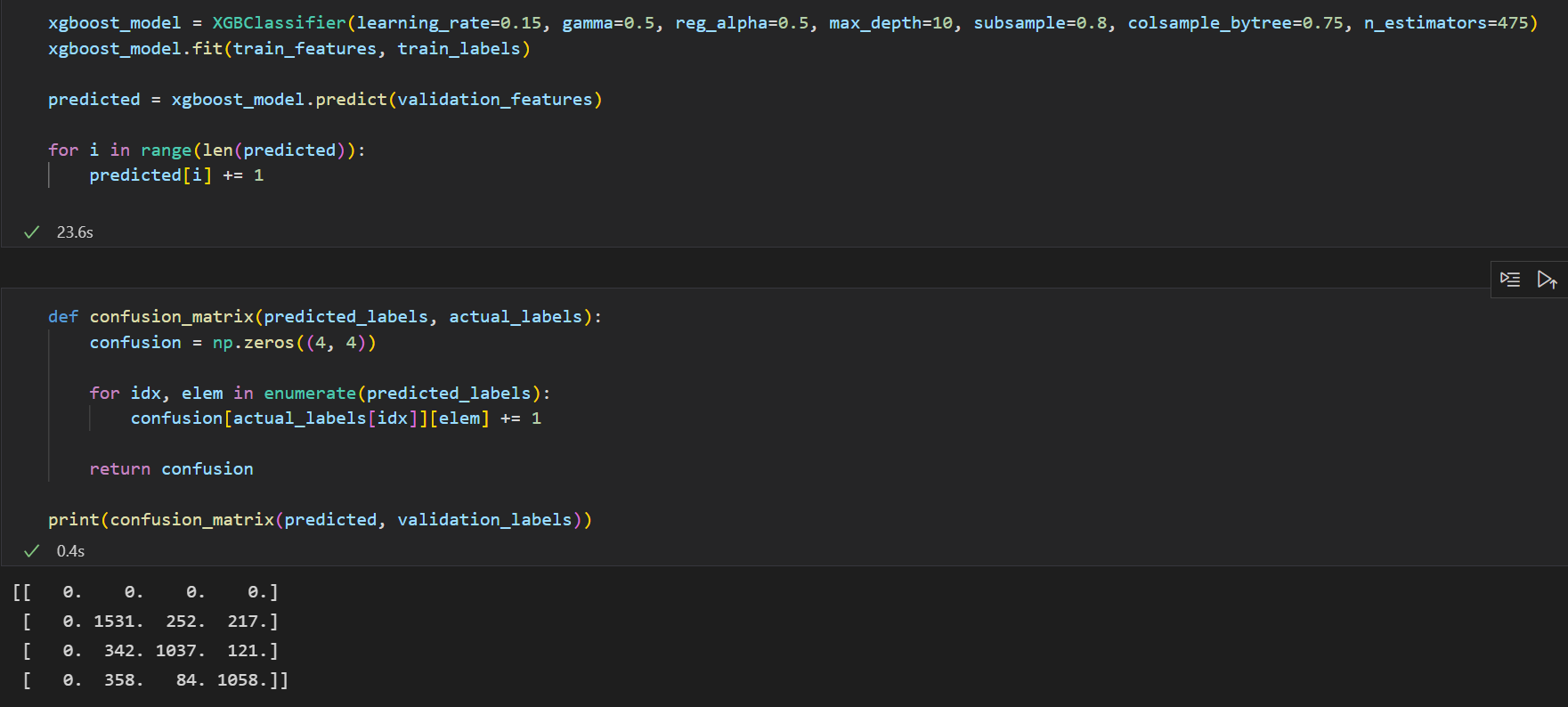
ComplementNB:



MultinomialNB:



XGBoost:



Pentru alte observatii legate de proiect, as dori sa mentionez ca am incercat sa normalizez sau sa standardizez datele, insa am avut rezultate cu acurateti mai scazute.

