Documentatie-ML

CT – Scan Classification

Negulescu Stefan
GRUPA 232

Descrierea task-ului

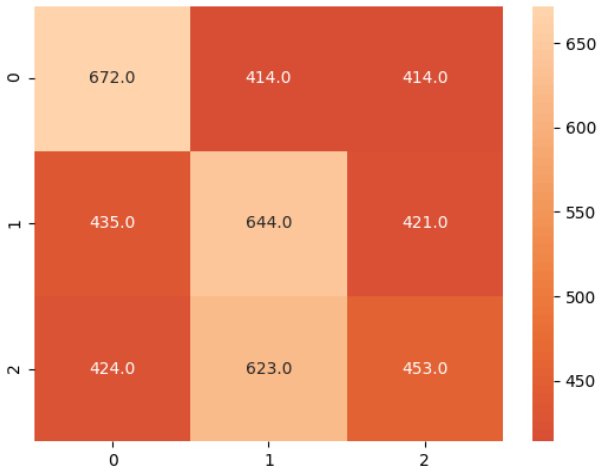
Task-ul presupune crearea unui model de machine-learning si antrenarea sa cu scopul de a obtine o acuratete cat mai mare a clasificarii datelor de test in 3 categorii, datele reprezentand tomografii computerizate ale plamanilor.

Setul de antrenare cuprinde 15.000 de imagini (cate 5.000 din fiecare categorie), setul de validare 4.500, iar setul de test 3.900. Imaginile au dimensiunile 50x50, acestea fiind deja grayscaled. Astfel, nu a fost nevoie de preprocesare in acest sens.

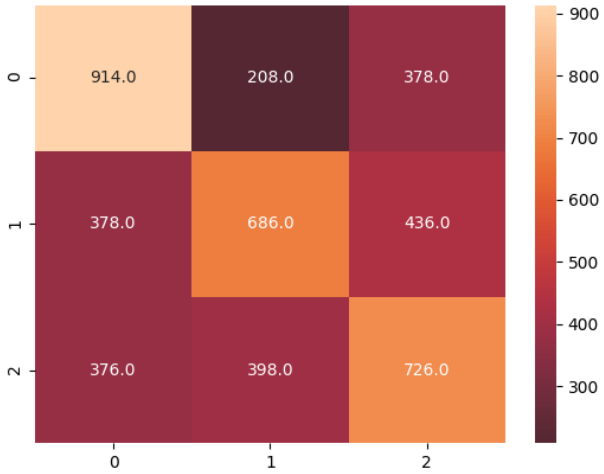
Categorie	**0**	**1**	**2**
Exemplu de imagine			

Modelul 1: k-NN

In acest caz, am utilizat un model asemanator cu cel de la laborator. Pentru k = 3, pe setul de validare, utilizand distanta l1 am obtinut scorul 0.3931, iar utilizand distanta l2 am obtinut 0.5168. De asemenea am observant faptul ca acuratetea creste odata cu parametrul k, insa nu in mod semnificativ. Pentru acelasi model cu k=3 si l2, pentru setul de test, am obtinut scorul de 0.48410 (public score).



Matricea de confuzie pentru k=3 si l1.



Matricea de confuzie pentru k=3 si l2.

Modelul 2: SVM

Pentru preprocesare am utlilizat **BagOfWords** (lab4)**,** inlocuind cuvintele din exemplul original cu pixelii imaginilor. Astfel pentru fiecare imagine se retine frecventa fiecarui pixel. Apoi, noile date obtinute sunt normalizate prin diferite metode (standard utilizand StandardScaller, l2 utilizand normalize cu norm='l2', minmax utlizand minmax).

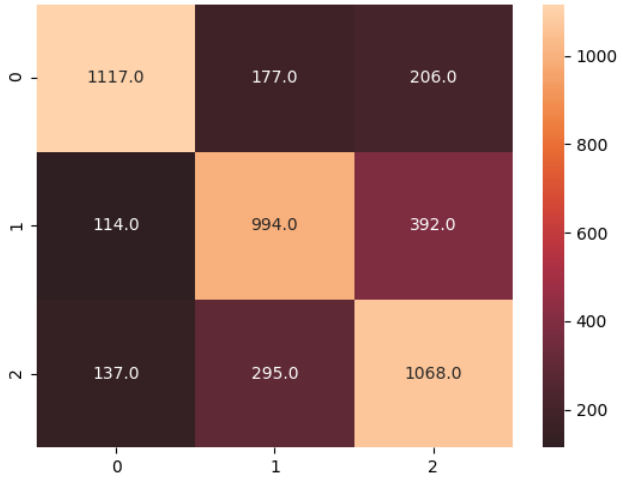
Atat pentru preprocesare, cat si pentru clasificator am utlizat libraria sklearn de unde am importat preprocessing, respectiv SVC. Am constatat ca, in cazul acestui clasificator hiperparametrii joaca un rol important in obtinerea unei precizii cat mai bune. Astfel, mai intai am testat cele 3 valori posibile pentru parametrul kernel, acestea fiind linear, rbf, si poly. In acest caz cea mai mare acuratete a fost obtinuta cu rbf (0.6942), apoi cu poly (0.6686), ultimul fiind linear cu o acuratete mult mai mica decat precedentele (0.5362). Pentru kernel='poly' am testat cateva valori pentru degrees = [0,1,2,3,4]. Pentru degree=0, precizia este 0.33, pentru urmatoarele valori precizia creste, ajungand la

valoarea maxima obtinuta (0.668), dupa care scade. Pentru kernel='rbf' am utilizat GridSearchCV din sklearn pentru a gasi cele mai bune valori pentru gamma si C. Pentru C = [0.1, 1, 10, 100] si gamma = [0.01, 0.1, 1, 10] cele mai bune 5 rezultate sunt prezentate in urmatorul tabel:

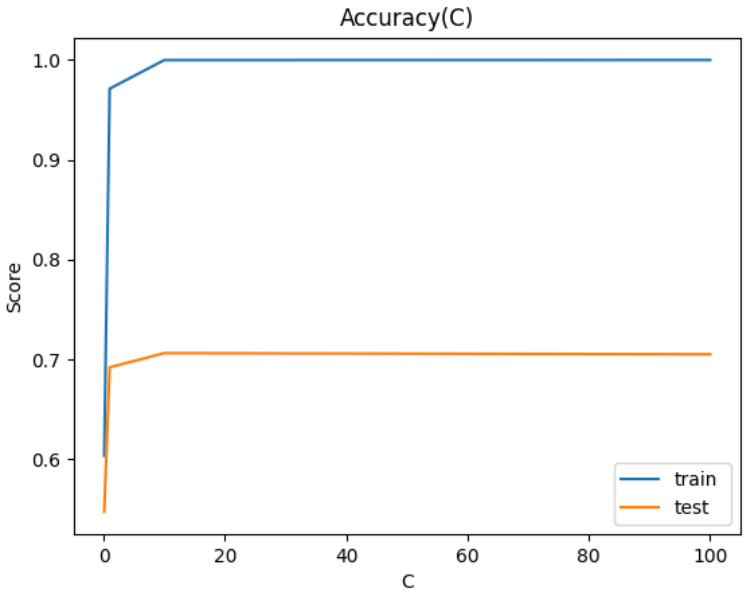
C	GAMMA	SCORE
100	0.01	0.7801
10	0.01	0.7796
1	0.01	0.7578
0.1	0.01	0.5368
10 / 100	0.1/ 0.1	0.4815

Mai departe sunt prezentate rezultatele pentru primele 3 locuri pentru setul de validare:

C	GAMMA	SCORE
100	0.01	0.6922
10	0.01	0.7064
1	0.01	0.7053



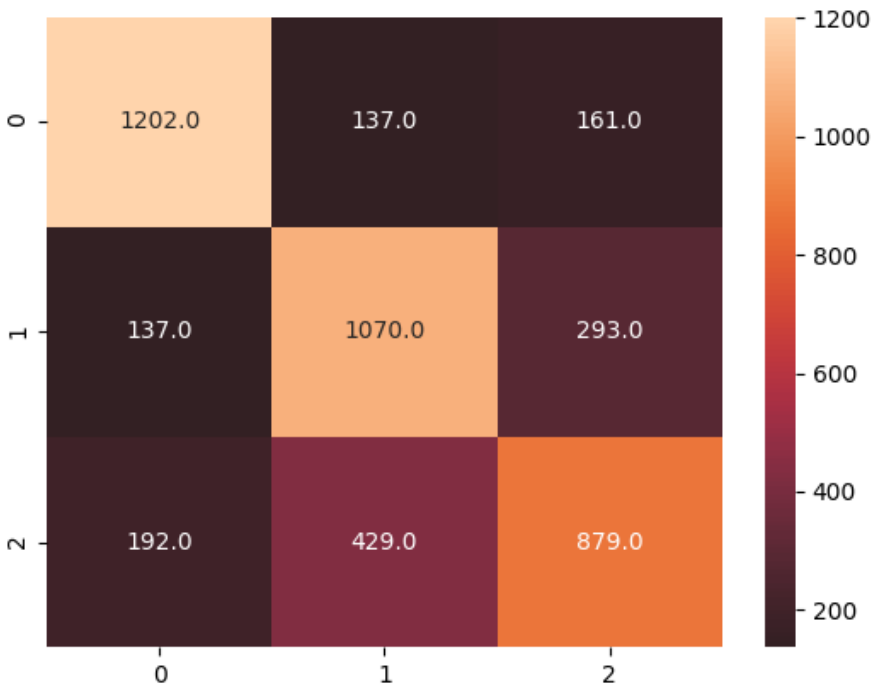
Matricea de confuzie pentru C=10 si gamma=0.01



Acurateata pentru train si test in functie de C

Modelul 3: MLP

In acest caz, preprocesarea este aceeasi cu cea de la modelul 2, diferenta dintre metodele de normalizare nefiind una semnificativa. Pentru clasificator am utilizat MLPClassifier() din libraria sklearn. Pentru acest model am ales sa folosesc varianta default pentru parametrul hidden_layer_sizes, si anume sa folosesc un singur layer, pentru ca scorul nu este imbunatatit in cazul in care sunt folosite mai multe layers, dar si din motive de imbunatatire a timpului de executie. Au fost testate diferite configuratii de parametrii, insa variant default (activation='relu', max_iter=200) a avut cel mai bun scor (0.7002).



Matricea de confuzie pentru MLPClassifier default

Se remarca faptul ca foarte multe elemente din categoria 1 sunt clasificate in categoria 2 si un numar foarte mare de elemente din categoria 2 sunt clasificate in 1. Am crezut ca acest aspect se datoreaza preprocesarii cu BoW, insa testand acelasi model fara Bow scorul este foarte mic (0.3868).

Concluzie

Cel mai bun model s-a dovedit a fi SVM impreunan cu preprocesarea respective. MLP este destul de aproape ca scor, timpul de executie scazand. Cel mai neperformant model este k-NN, acesta fiind de asemenea si cel mai lent.