UNIVERSITATEA DIN BUCUREȘTI
FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ

PROIECT
INTELIGENȚĂ ARTIFICIALĂ

STUDENT:
HANGHICEL RĂZVAN-MIHAI
GRUPA 241

BUCUREȘTI
2023

Cuprins

1. Descrierea proiectului

Proiectul constă în antrenarea unui clasificator pentru a prezice dacă o scanare de tip computer tomograf a creierului arată dacă acesta prezintă o anomalie sau este sănătos.

Datele de antrenare constau într-o colecție de 22149 de imagini. Fiecare exemplu de antrenare conține o scanare de tip computer tomograf a creierului dintr-o anumită secțiune. Fiecare exemplu este etichetat cu un anumit label care poate avea două valori posibile: 0 (reprezentând că imaginea cu secțiunea arată un creier sănătos), 1 (reprezentând că imaginea cu secțiunea arată un creier care prezintă o anomalie). Pentru datele de train și validare cunoaștem și label-urile pentru fiecare imagine și target-ul nostru este să determinăm label-urile pentru imaginile de test și să obținem o acuratețe cât mai bună.

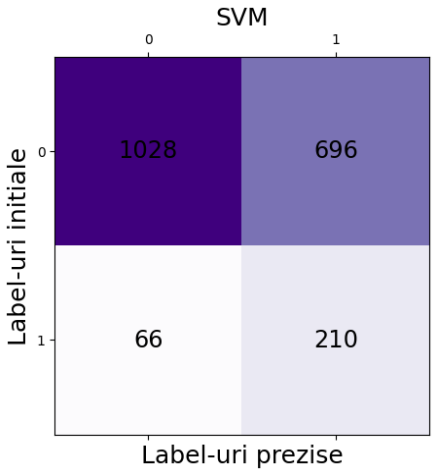
2. Modele

În rezolvarea proiectului am folosit următoarele modele: mașini cu vector suport, random forest classifier și rețele neuronale convoluționale.

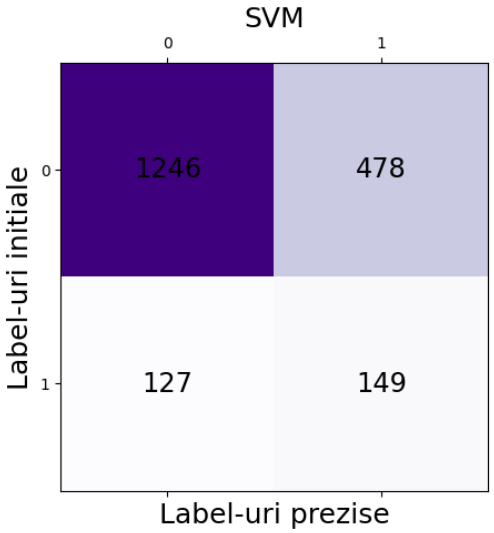
2.1 Masini cu vectori suport (SVM)

Primul model pe care l-am încercat a fost o mașină cu vector suport (SVM).
După aplicarea lui cv2.IMREAD_GRAYSCALE din (224,224,3) va rezulta o imagine cu shape-ul de (224,224). Deci o idee de a micșora numărul de elemente și de a nu pierde prea multe informații din date este să salvăm pentru fiecare imagine media pe fiecare linie, astfel obținând feature-uri de 224x1 și reducem timpul pentru a antrena modelul și astfel ne asigurăm că imaginile o să încapă în RAM.
Am observat că există o diferență destul de mare între imaginile cu label-uri de 1 și cele cu label-uri de 0. Pentru a rezolva această problemă am determinat numărul de label-uri de 1 din testele de train și am calculat un weight pentru 1. Am făcut același lucru și pentru label-urile de 0 și aceste 2 valori le-am adăugat la parametrul de class_weight din crearea modelului. Am testat mai multe valori pentru C și Kernel:

- **C = 1, Kernel = linear** => **accuracy score = 0.619**, **precision score = 0.231**, **recall score = 0.760** si **F1 score = 0.355** și cu următoarea matrice de confuzie:



- **C = 1**, **Kernel = poly**=> **accuracy score = 0.698**, **precission score = 0.237**, **recall score = 0.540** si **F1 score = 0.33** și cu următoarea matrice de confuzie:



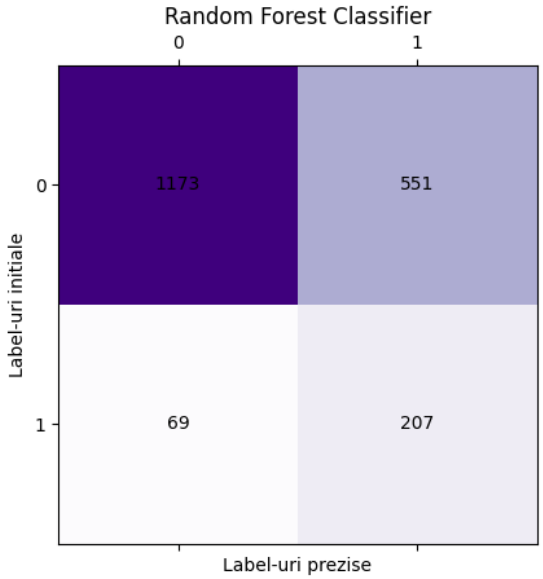
2.2 Random Forest Classifier

Al doilea model pe care l-am folosit este Random Forest Classifier. Am procedat inițial fix la fel ca la modelul anterior, adică partea cu calcularea de weights pentru label-urile de 0 și 1 și adăugarea acestora sub formă de dicționar în declararea modelului în parametrul class_weight. Am setat random_state = 20 și max_depth = 20 pentru toate încercările. După rularea unor cicluri for-uri în care am modificat valorile pentru n_estimators și min_samples_leaf, am obținut următoarele 3 cele mai bune valori pentru acești parametri:

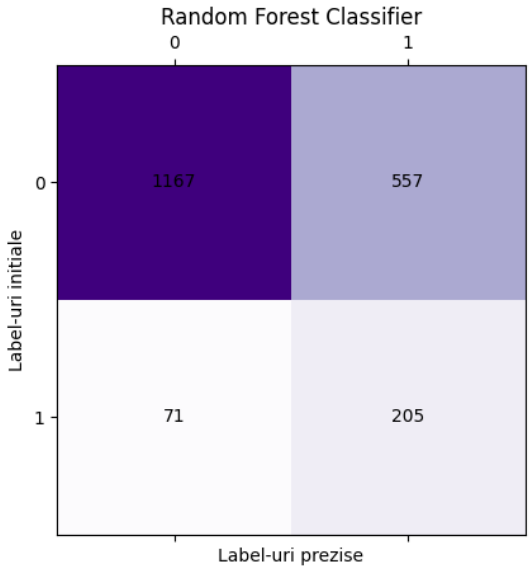
n_estimators	**min_samples_leaf**	**Accuracy score**
123	100	0.69
144	101	0.686

112	110	0.6795

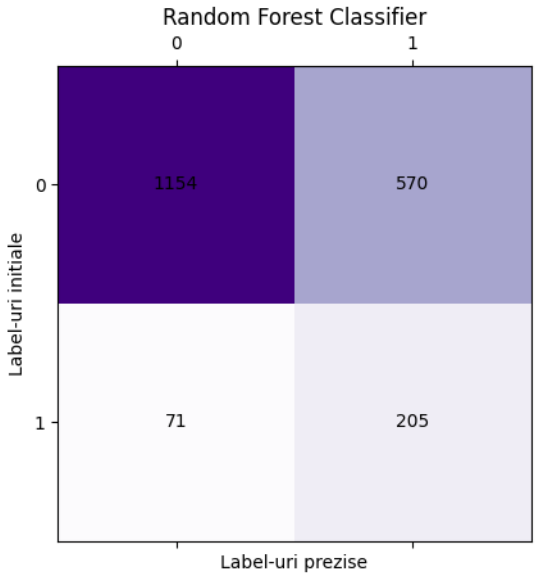
- **n_estimators = 123, min_samples_leaf = 100** => **accuracy score = 0.69**, **precision score = 0.273**, **recall score = 0.75** si **F1 score = 0.4** și cu următoarea matrice de confuzie:



- **n_estimators = 144, min_samples_leaf = 101**=> **accuracy score = 0.686**, **precision score = 0.269**, **recall score = 0.742** si **F1 score = 0.394** și cu următoarea matrice de confuzie:



- **n_estimators = 112, min_samples_leaf = 110**=> **accuracy score = 0.6795**, **precision score = 0.264**, **recall score = 0.742** si **F1 score = 0.390** și cu următoarea matrice de confuzie:



2.3 Convolutional Neural Network

1) Data Augmentation and Data Processing

Pentru fiecare imagine din setul primit am aplicat anumite transformări: RandomVerticalFlip() care rotește vertical imaginea curentă cu o probabilitate de 0.5; RandomHorizontalFlip() care rotește orizontal imaginea curentă cu o probabilitate de 0.5; RandomRotation(80) care rotește imaginea în jurul centrului acesteia cu un unghi între [-80, 80] de grade; Normalize(mean = [0.5,0.5,0.5], std = [0.5,0.5,0.5]) care o să scadă din fiecare imagine media și o să împartă la standard deviation pentru fiecare channel.

2) Structura modelului

a) Layere:

- **Conv2d** este folosit pentru a extrage feature-urile din imaginile primite ca input. Se vor da numărul de channels de intrare și numărul de channels de ieșire care reprezintă numărul de filtre ce sunt învățate de către layer-ul convolutional. Apoi mai specificăm și **kernel_size** care reprezintă dimensiunea filtrului ce trebuie utilizat și este aplicat printr-un procedeu numit “sliding window”. Apoi se specifică **stride**-ul care reprezintă cu cât se va muta kernel-ul prin matrice și **padding**-ul care reprezintă cu cât bordam matricea de 0-uri
- **MaxPool2d** este folosit pentru a reduce dimensiunea în spatiu a matricei de features și de a reține cele mai importante features, astfel eliminând feature-urile care nu ne ajută în clasificarea modelului. Acesta are ca parametru **kernel_size** care determină dimensiunea matricei folosite pentru a alege cel mai mare element și **stride**-ul care

arată cu cât se muta această matrice pentru a putea calcula valorile pentru matricea inițială.

- **Linear** care aplică o transformare liniară asupra datelor de intrare. Primește ca parametri numărul de features de intrare și numărul de features de ieșire.

b) Hiperparametri:

i) Layere convolutionale:

- In si out channels = {64, 128,256,512}
- Kernel_size = (3,3)
- Stride = (1,1)
- Padding = (1,1)
- Funcție de activare ReLU

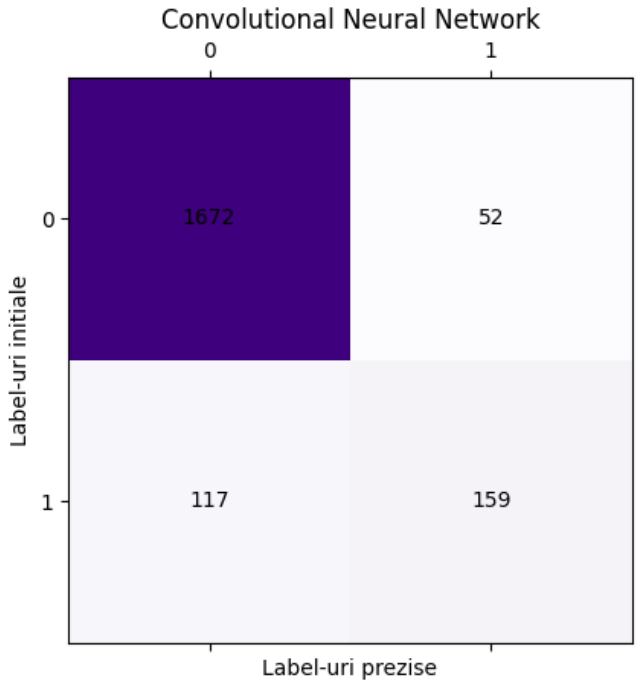
ii) Layere dense:

- Funcție de activare ReLU
- Mulțimea valori cu care reduc numărul de features {128,256,512,1024}
- Pentru ultimul layer o să avem ca output 2, deoarece în acest caz avem de clasificat fiecare imagine în 2 clase

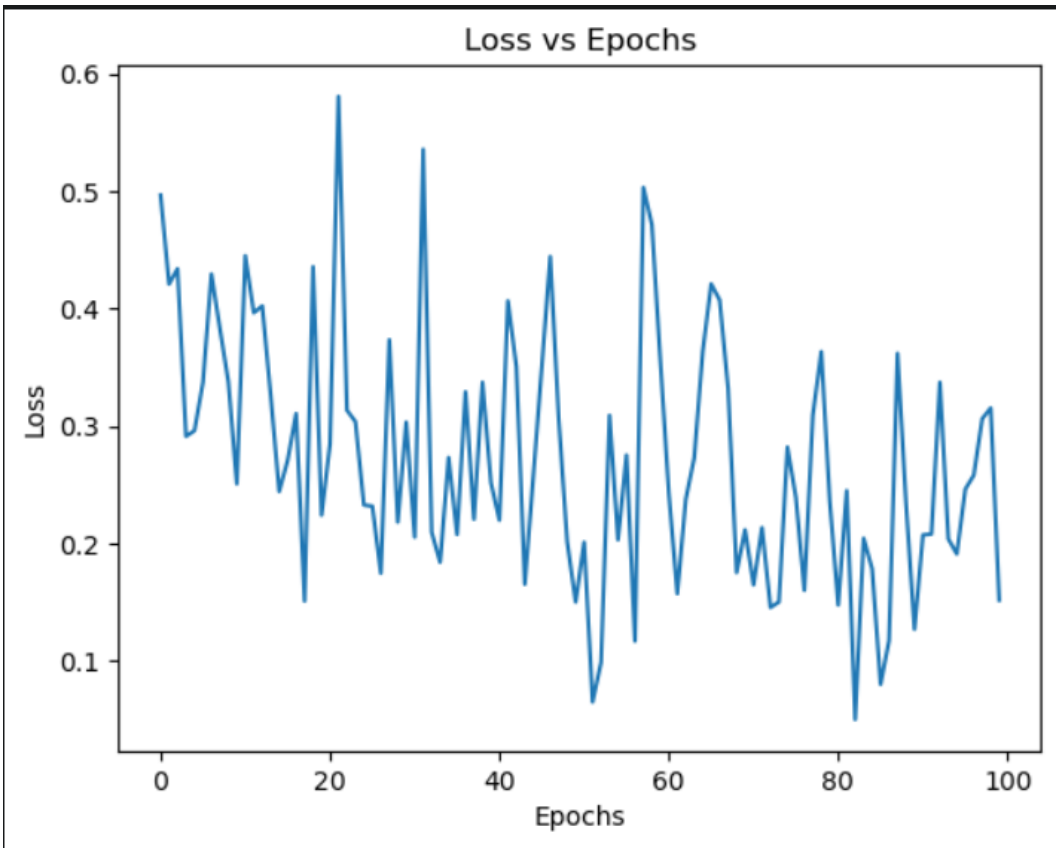
c) Rularea modelului

- Folosesc ca funcție de loss CrossEntropyLoss pentru determinarea loss-ului dintre label-urile prezise și cele adevărate
- Folosesc ca funcție de optimizare Adam

După rularea modelului pentru 100 de epoci am obținut pentru datele de validare următoarele rezultate: **accuracy score = 0.8835**, **precision score = 0.573**, **recall score = 0.605**, **F1 score = 0.589** și cu următoarea matrice de confuzie:



Dupa plotarea loss-ului pentru 100 de epoci:



3.Concluzie

În urma rulării celor 3 modele pe datele primite am obținut pe Kaggle scorul de 0.70188 pentru 20% dintre date și 0.73616 pe următoarele 80% dintre date. Aceste rezultate au fost obținute folosind modelul de CNN descris mai sus.