Brain Anomaly Detection - Documentation

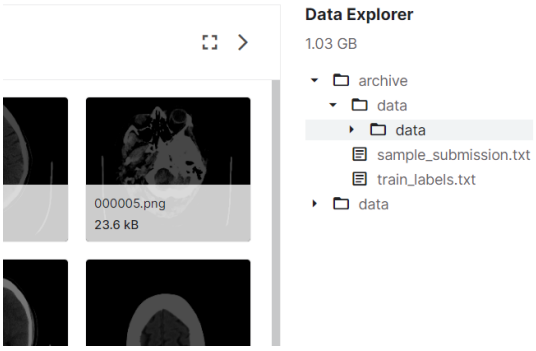
Rus Alexandru

Grupa 241

Brain Anomaly Detection – *Detect anomalies in CT scans of the brain*

Scopul proiectului este să antrenăm un model care reușește să facă diferența dintre două clase de scanare CT a creierului. O clasă care conține anomalii (clasa cu numărul 1) și o clasă normală (clasa cu numărul 0).

Primim un folder cu date care conține un folder cu 22149 de poze de tip PNG, un fisier *text train_labels.txt* care conține datele de antrenare a modelului (id-urile primelor 15.000 de poze, respectiv clasa din care face parte fiecare dintre acestea), un fisier *validation_labels.txt* care conține datele de validare a următoarelor 2000 de poze cu clasa din care ar trebui să facă parte și un fișier cu extensia .csv care reprezinta un exemplu de fișier de submisie.



În acest proiect, am folosit diferite modele pe care le-am antrenat cu primele 15.000 de poze si label-urile respective, model căruia i-am calculat apoi acuratețea, pentru ca în final să îi cerem să prezică singur pentru un număr de 5149 de poze, clasa care ar trebui să corespundă fiecăreia (0 sau 1).

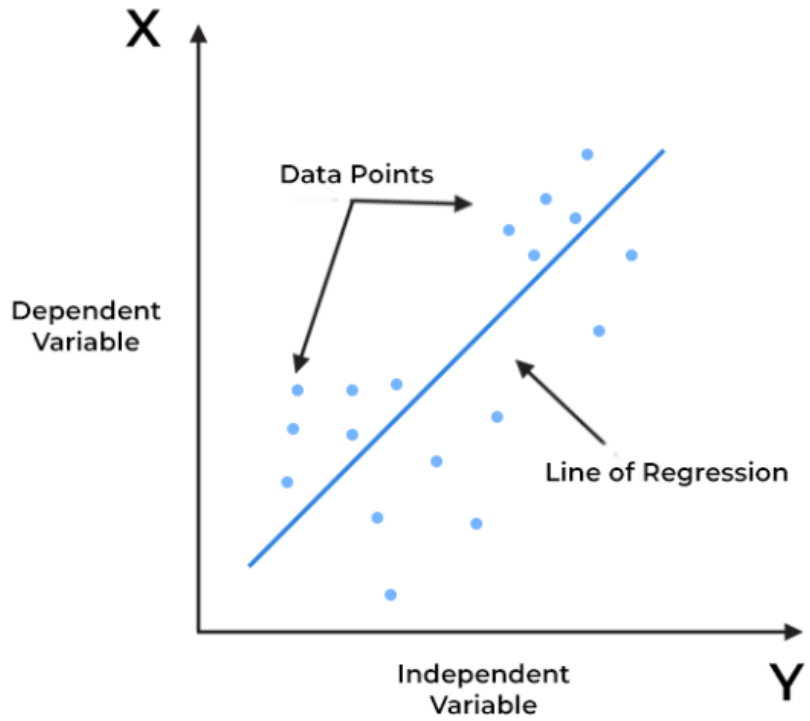
Aceste informații le-am scris intr-un fisier de tip .csv, dupa modelul primit in sample_submission, pe care l-am încărcat în cadrul competiției. Astfel, în funcție

de acuratețea informațiilor noastre, primeam un punctaj și eram clasați în clasament alături de colegii noștri. Scorul minim de trecere era 0.23.

Abordarea mea:

După ce am înțeles exact task-ul și scopul proiectului, am început să caut diferite informații legate de modele, de tipul de modele folosite pentru clasificarea imaginilor, care sunt cele mai ,,accurate", cum putem mări acuratețea acestora pentru a ajunge să obținem un scor cât mai mare în competiție.

Am ales să folosesc un model de tipul *„Linear Models"*, aceste modele statistice presupun o relație liniară între o variabila dependenta și una independenta.



https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-linear-regression/

Am ales modelul de ***Linear Regression***. Acesta încearcă să găsească o relație dintre cele doua variabile (în cazul nostru, poza și mulțimea de clase {0,1}) și să

găsească o ,,regulă” care are o marjă de eroare cât mai mică. Astfel, după ce a fost antrenat cu un anumit set de date, poate face preziceri pentru noi valori independente (poze) cărora să le asocieze valori dependente (clase). Astfel, la final observăm acuratețea acestuia, în funcție de valorile corecte ale variabilelor dependente și cele returnate de modelul nostru.

Am început prin importarea librăriilor necesare**, glob** pentru încarcarea filelor, **numpy** pentru operații pe array-uri**, csv** pentru prelucrarea filelor CSV, **skimage** pentru imagini.

Apoi, scopul meu era să încarc imaginile într-un vector de imagini pe care să îl ,,dau” modelului pentru a se antrena. După ce am încarcat imaginile într-un vector, am constatat că modelul nu accepta acea forma a array-ului. Am aflat forma cu **np.shape()** din numpy. Vectorul meu era 4-dimensional și voiam să îl fac 2-dimensional. Am revenit la citirea imaginilor și am scos atributul de culoare care era inutil în cazul nostru, deoarece imaginile erau alb-negru prin functia **color.rgb2gray()** din skimage. De abia apoi le adăugam în vectorul de imagini. Astfel, vectorul de imagini a ajuns sa fie tridimensional, de forma (no_of_images, 224, 224), iar pentru a-l face bidimensional am folosit funcția reshape() din numpy, iar dupa o verificare cu np.shape(), am vazut ca vectorul de imagini era de forma (no_of_images, 50176)

```
In [2]: images = []
        file = "data/data/*"
        for photo in glob.glob(file):
            image = io.imread(photo)
            image = color.rgb2gray(image)
            images.append(image)
        print(np.shape(images))
        images = np.reshape(images,(-1,224*224))
        print(np.shape(images))
```

```
(22149, 224, 224)
(22149, 50176)
```

Am încărcat și informațiile din *train_labels.txt* și *validation_labels.txt* sub forma unui vector în care stocăm clasele (train_labels = [cls1,cls2,...,cls22150])

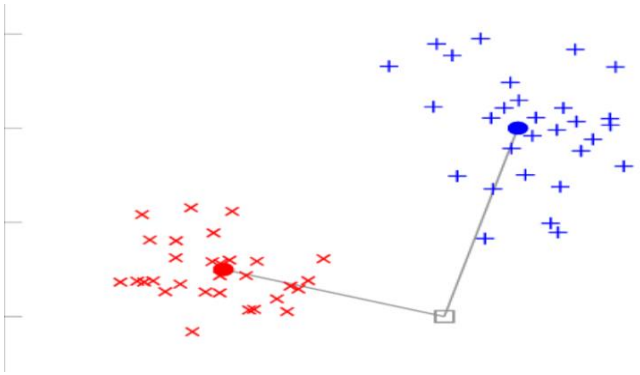
Am initializat modelul (**model = LogisticRegression()),** dupa care am dat inceput sa il antrenez prin **model.fit(images[0:15000], train_labels[0:15000]),** deoarece in cerinta se specifica ca primele 15.000 de poze, respectiv clase sunt pentru antrenarea modelului.

Dupa ce l-am antrenat, am calculat **f1_score** pentru urmatoarele 2000 de imagini, pentru care aveam validation_labels. Am dat **model.predict(images[15000:17000])** si am calculat

f1_score(validation_labels, model.predict(images[15000:17000]))

Acesta ar trebui sa ne ofere o aproximatie a acuratetii acestuia. Scorul a fost in jur de 0.1-0.2, insa cum nu stiam ce valoare reprezinta o valoare ,,buna", am decis sa prezic urmatoarele 5150 de clase pentru urmatoarele 5150 de poze si sa scriu rezultatul in fisierul .csv creat de mine. Asadar, dupa ce am facut toate aceste lucruri, am uploadat fisierul .csv in competitie si am constatat ca scorul meu era undeva pe la 0.28. Trecusem de baseline, dar am decis sa mai incerc si alte modele care poate urmau sa imi ofere un scor mai bun.

Asa ca am inceput sa caut mai multe strategii, am decis sa incerc cu inca un model. Am ales **Nearest Centroid Classifier (NCC)**, acesta utilizeaza centroidul fiecarei clase si, dupa ce este antrenat, poate sa faca predictii, la fel cum am facut in celalalt model utilizat. Atunci când vine vorba de predicții, NCC calculează distanța euclidiană dintre punctul de test și centroidul fiecărei clase. Punctul este clasificat în clasa care are centroidul cel mai apropiat de punctul respectiv.



In aceasta imagine (https://www.researchgate.net/figure/A-visualisation-of-the-nearest-centroid-classification-algorithm-Here-a-training_fig9_286513346), putem observa o vizualizare a acestui algoritm. Datele au fost deja impartite in doua clase, iar predictiile se vor face in functie de distanta catre fiecare centroid din ficare grupa.

Asadar, am considerat ca e un model potrivit pentru problema noastră și am decis să îl încerc. După modelul prezentat mai sus, am dat fit la primele 15.000 de poze pentru a antrena modelul, după care am calculat f1_score pentru a vedea performanța modelului și desigur pentru a o compara cu cea a primului model folosit (Logistic Regression). Dupa ce am antrenat modelul, am dat predict la urmatoarele 2000 de poze pentru care aveam label-uri de validare si am constatat ca scorul f1_score era mai mare decat la modelul anterior.

```
In [7]: #aray = model.predict(images[17000:22151])
        aray = model3.predict(images[17000:22151])

In [8]: testf1 = model3.predict(images[15000:17000])

        print(f1_score(validation_labels,testf1))

        0.3341729638958858
```

Astfel, am scris informatiile in fila mea .csv si am dat submit in competitie, iar rezultatul a fost mai mare, in jur de 0.4

Am incercat diferite metode pentru a imbunătăți acuratețea modelului. Am încercat să îl antrenez pe pozele rotite prin np.flip(), însă acuratețea a rămas aceeași.

În concluzie, după studierea mai multor modele și desigur, implementarea lor, am ales scorul cel mai mare pentru a obține un loc cât mai bun în competiție.