Stancu Denisa, 342

Proiect Inteligenta Artificiala

Scopul acestui proiect este de a clasifica imagini CT ale creierului in doua categorii: contine anomalii (etichetata 1) și normala (eticheta 0). In această lucrare am încercat sa rezolv problema folosind 2 modele de clasificare, si anume SVM si CNN.

Procesarea datelor:

Am definit o functie image_generator pentru a incarca imaginile in batch-uri de dimensiunea batch_size specificata. Am citit fisierele de etichetare pentru antrenare si validare cu ajutorul bibliotecii pandas, si am preluat calea imaginilor si etichetele asociate cu acestea. Am normalizat valorile pixelilor imaginilor .Imaginile au fost redimensionate la dimensiunea 64x64x3. Imaginile sunt normalizate prin scaderea mediei setului de date si impartirea la deviatia standard. Aceasta etapa de preprocesare ajuta la asigurarea ca toate caracteristicile sunt in aceeasi scala si imbunatateste convergenta modelului de retea neuronala in timpul antrenamentului.

Acestea sunt pasii principali pe care i-am urmat pentru a pregati datele pentru modelul SVM si modelul CNN. In plus, pentru modelul CNN am aplicat functia to_categorical asupra etichetelor de antrenare.

Modele:

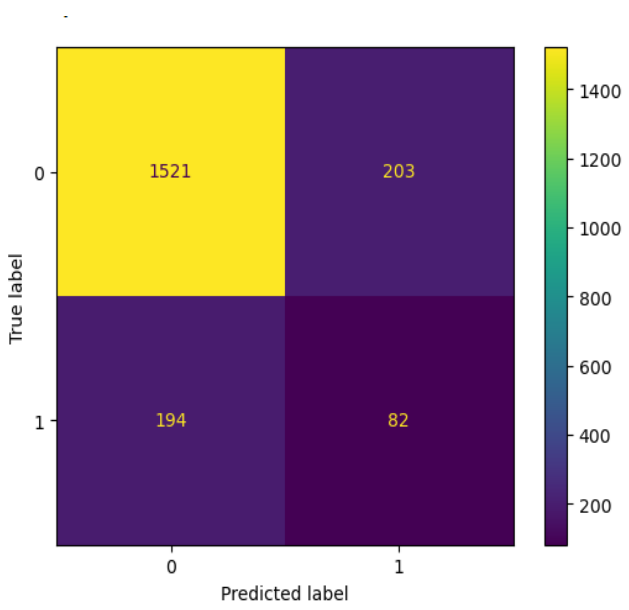
1. SVM:

SVM (Support Vector Machines) este un algoritm de invatare supervizata pentru clasificare si regresie. In cazul clasificarii binare, SVM incearca sa gaseasca un hiperplan care separa perfect cele doua clase de obiecte, astfel incat fiecare obiect dintr-o clasa sa fie de o parte a hiperplanului, iar fiecare obiect din cealalta clasa sa fie de cealalta parte. Hiperplanul poate fi descris printr-o ecuatie liniara.

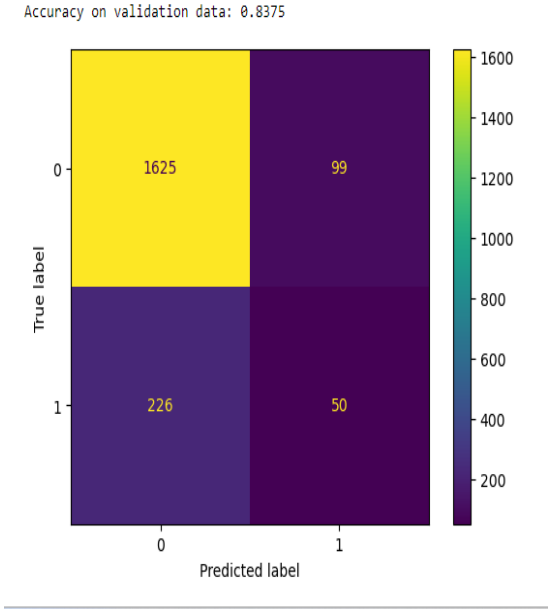
Pentru acest model am incercat mai multe variante schimband valorile kernelului si a C-ului. Am folosit functia compute_class_weight din biblioteca Scikit-learn pentru a calcula class_weights deoarece am observat ca cele doua clase nu sunt echilibrate, clasa 0 este mult mai comuna decat clasa 1, insa aceasta functie nu a imbunatatit rezultatele.

C	KERNEL	S-a folosit class_weights	Acuratete
1	linear	Nu	80,1%
3	poly	Nu	84,9%
3	poly	Da	81,7%
3	rbf	Nu	85,5%
3	rbf	Da	80,6%
5	rbf	Nu	85,5%

Desi modelul prezenta o acuratete ridicata, analizand matricele de confuzie am observat ca avea dificultati in prezicerea imaginilor etichetate cu clasa 1, prezentand o marja semnificativa de eroare.

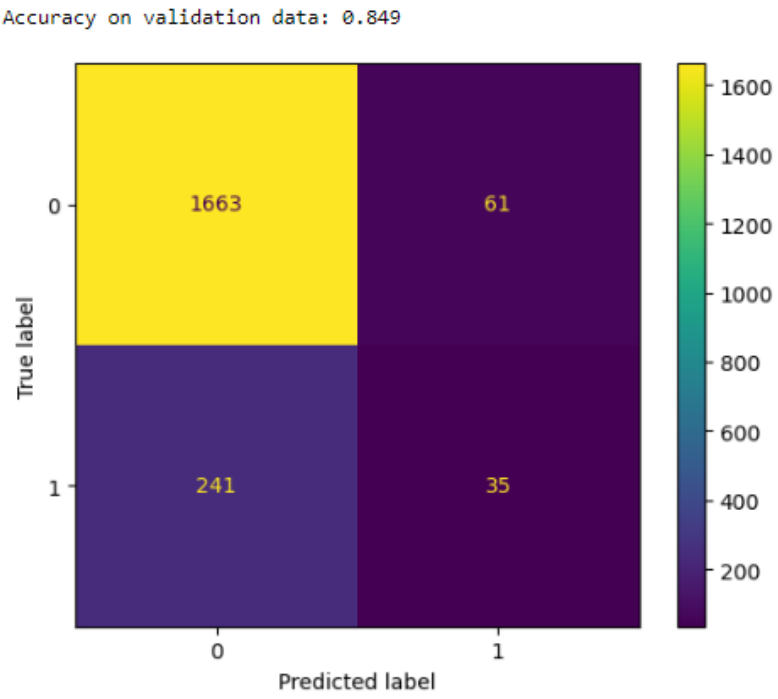


C= 3; kernel = linear



C=10; kernel = poly

Dupa mai multe incercari, am ales sa folosesc modelul cu C = 3 ; kernel = poly fara class_weights ce a avut acuratetea de 84,9% pe datele de validare.



Nu eram multumita de rezultate si am trecut la urmatorul model.

2. CNN:

CNNs (Convolutional Neural Networks) sunt o clasa de retele neuronale profunde specializate in prelucrarea datelor de tip imagini. Acestea sunt inspirate de modul in care functioneaza creierul uman in a recunoaste obiecte vizuale, prin detectarea de caracteristici de niveluri diferite.

CNN-urile folosesc un set de filtre convolutionale care detecteaza caracteristici specifice in imaginea de intrare, cum ar fi margini, colturi sau forme geometrice. Aceste filtre sunt aplicate asupra intregii imagini de intrare, iar rezultatele sunt agregate intr-un map de caracteristici.

Modelul folosit de mine este proiectat pentru a clasifica imagini in doua categorii si are urmatoarea arhitectura:

- O serie de straturi convolutionale si de impachetare pentru a extrage caracteristici din imagini
- Un strat Flatten pentru a transforma caracteristicile intr-un vector
- Un strat Dropout pentru a preveni suprapunerea
- Doua straturi complet conectate pentru clasificare

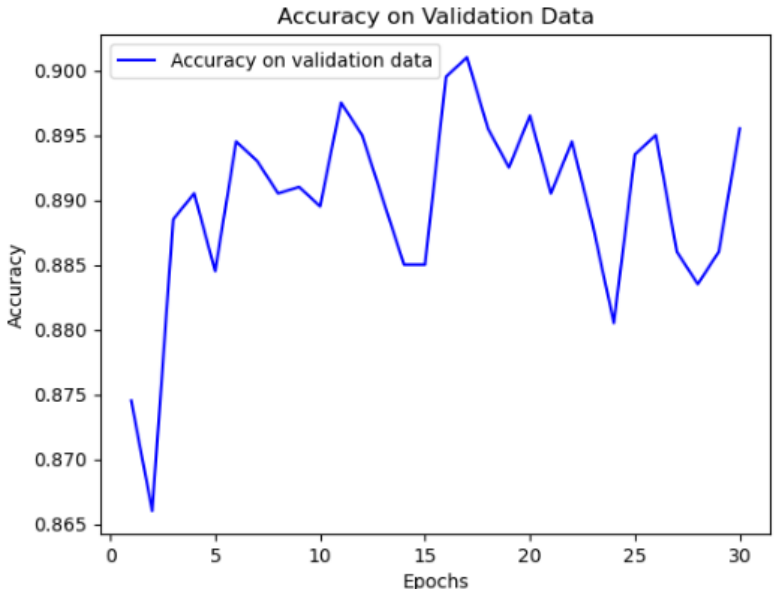
Modelul a fost antrenat si validat folosind un set de date de imagini preprocesate, iar performanta sa a fost evaluata pe baza acuratetii de clasificare pe un set de date de

validare. Acesta a obtinut o acuratete de aproximativ 80% in ceea ce priveste clasificarea imaginilor.

Prima data am incercat cu loss='binary_crossentropy' si epochs = 15:

```
Epoch 1/15
235/235 [==============================] - 32s 131ms/step - loss: 0.3751 - accuracy: 0.8531 - val_loss: 0.3405 - val_accuracy:
0.8725
Epoch 2/15
235/235 [==============================] - 33s 140ms/step - loss: 0.3340 - accuracy: 0.8577 - val_loss: 0.3179 - val_accuracy:
0.8780
Epoch 3/15
235/235 [==============================] - 35s 149ms/step - loss: 0.3186 - accuracy: 0.8619 - val_loss: 0.2936 - val_accuracy:
0.8895
Epoch 4/15
235/235 [==============================] - 35s 149ms/step - loss: 0.2948 - accuracy: 0.8732 - val_loss: 0.3334 - val_accuracy:
0.8845
Epoch 5/15
235/235 [==============================] - 36s 152ms/step - loss: 0.2811 - accuracy: 0.8821 - val_loss: 0.2674 - val_accuracy:
0.8915
Epoch 6/15
235/235 [==============================] - 37s 156ms/step - loss: 0.2633 - accuracy: 0.8899 - val_loss: 0.2646 - val_accuracy:
0.8935
Epoch 7/15
235/235 [==============================] - 36s 155ms/step - loss: 0.2483 - accuracy: 0.8953 - val_loss: 0.2569 - val_accuracy:
0.9005
Epoch 8/15
235/235 [==============================] - 37s 156ms/step - loss: 0.2340 - accuracy: 0.9019 - val_loss: 0.2631 - val_accuracy:
0.8930
Epoch 9/15
235/235 [==============================] - 36s 155ms/step - loss: 0.2173 - accuracy: 0.9079 - val_loss: 0.2777 - val_accuracy:
0.8915
Epoch 10/15
235/235 [==============================] - 36s 155ms/step - loss: 0.2032 - accuracy: 0.9152 - val_loss: 0.2856 - val_accuracy:
0.8845
Epoch 11/15
235/235 [==============================] - 37s 157ms/step - loss: 0.1904 - accuracy: 0.9178 - val_loss: 0.2879 - val_accuracy:
0.8805
Epoch 12/15
235/235 [==============================] - 37s 155ms/step - loss: 0.1690 - accuracy: 0.9291 - val_loss: 0.3097 - val_accuracy:
0.8800
Epoch 13/15
235/235 [==============================] - 37s 156ms/step - loss: 0.1529 - accuracy: 0.9376 - val_loss: 0.3654 - val_accuracy:
0.8940
Epoch 14/15
235/235 [==============================] - 36s 155ms/step - loss: 0.1396 - accuracy: 0.9423 - val_loss: 0.3544 - val_accuracy:
0.8685
Epoch 15/15
235/235 [==============================] - 38s 160ms/step - loss: 0.1265 - accuracy: 0.9492 - val_loss: 0.3456 - val_accuracy:
0.8820
Validation accuracy: 88.20%
```

Am incercat si cu loss = 'categorical_crossentropy', epochs = 30:

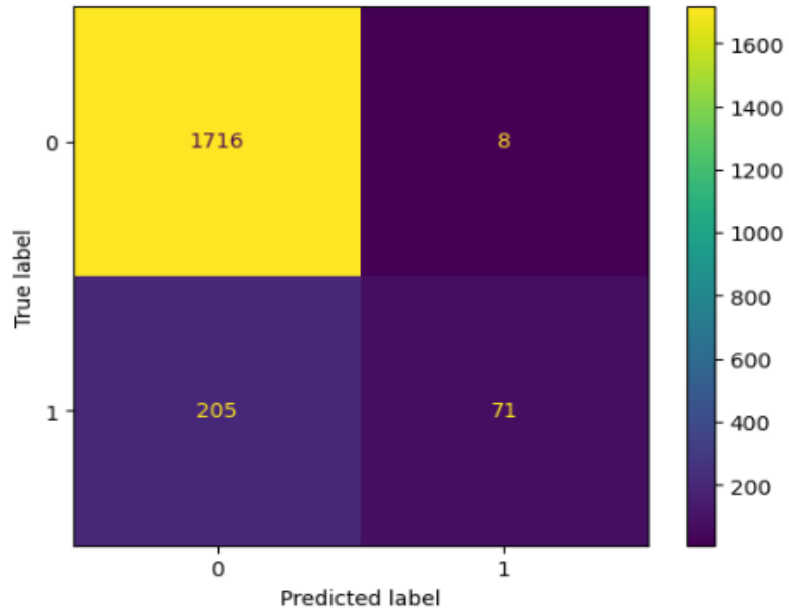


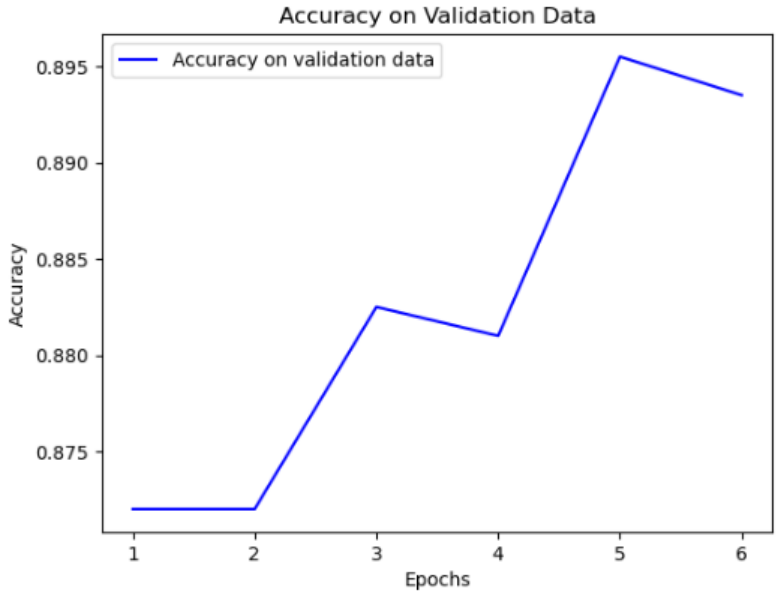
Apoi am incercat cu loss='categorical_crossentropy', epochs = 12 si am folosit si functia compute_weight:

```
Epoch 1/12
150/150 [==============================] - 28s 183ms/step - loss: 0.5892 - accuracy: 0.6098 - val_loss: 0.4744 - val_accuracy:
0.6545
Epoch 2/12
150/150 [==============================] - 32s 211ms/step - loss: 0.5177 - accuracy: 0.6665 - val_loss: 0.5305 - val_accuracy:
0.6650
Epoch 3/12
150/150 [==============================] - 32s 212ms/step - loss: 0.4893 - accuracy: 0.7159 - val_loss: 0.5867 - val_accuracy:
0.6285
Epoch 4/12
150/150 [==============================] - 32s 215ms/step - loss: 0.4730 - accuracy: 0.7320 - val_loss: 0.5794 - val_accuracy:
0.7035
Epoch 5/12
150/150 [==============================] - 32s 216ms/step - loss: 0.4475 - accuracy: 0.7665 - val_loss: 0.5928 - val_accuracy:
0.6010
Epoch 6/12
150/150 [==============================] - 33s 220ms/step - loss: 0.4338 - accuracy: 0.7746 - val_loss: 0.5048 - val_accuracy:
0.7165
Epoch 7/12
150/150 [==============================] - 34s 224ms/step - loss: 0.4078 - accuracy: 0.7799 - val_loss: 0.4562 - val_accuracy:
0.7515
Epoch 8/12
150/150 [==============================] - 34s 225ms/step - loss: 0.3799 - accuracy: 0.8075 - val_loss: 0.3721 - val_accuracy:
0.8165
Epoch 9/12
150/150 [==============================] - 34s 228ms/step - loss: 0.3551 - accuracy: 0.8204 - val_loss: 0.6256 - val_accuracy:
0.6570
Epoch 10/12
150/150 [==============================] - 35s 236ms/step - loss: 0.3318 - accuracy: 0.8355 - val_loss: 0.4448 - val_accuracy:
0.7785
Epoch 11/12
150/150 [==============================] - 36s 237ms/step - loss: 0.3032 - accuracy: 0.8515 - val_loss: 0.3969 - val_accuracy:
0.8165
Epoch 12/12
150/150 [==============================] - 35s 231ms/step - loss: 0.3011 - accuracy: 0.8460 - val_loss: 0.3904 - val_accuracy:
0.8010
Validation accuracy: 80.10%
```

Dupa mai multe incercari si experimente cel mai bun rezultat a fost folosind functia loss='categorical_crossentropy' si epochs = 6 cu acuratetea de 89,35%(acest procent oscileaza intr-un mod neglijabil) pe datele de antrenare si 58,9% pe datele de testare conform Kaggle.

```
Epoch 1/6
150/150 [==============================] - 30s 197ms/step - loss: 0.3801 - accuracy: 0.8481 - val_loss: 0.3195 - val_accuracy:
0.8720
Epoch 2/6
150/150 [==============================] - 31s 207ms/step - loss: 0.3347 - accuracy: 0.8561 - val_loss: 0.2999 - val_accuracy:
0.8720
Epoch 3/6
150/150 [==============================] - 31s 204ms/step - loss: 0.3124 - accuracy: 0.8649 - val_loss: 0.3084 - val_accuracy:
0.8825
Epoch 4/6
150/150 [==============================] - 33s 219ms/step - loss: 0.3009 - accuracy: 0.8725 - val_loss: 0.2986 - val_accuracy:
0.8810
Epoch 5/6
150/150 [==============================] - 33s 219ms/step - loss: 0.2797 - accuracy: 0.8855 - val_loss: 0.2595 - val_accuracy:
0.8955
Epoch 6/6
150/150 [==============================] - 33s 223ms/step - loss: 0.2631 - accuracy: 0.8905 - val_loss: 0.2782 - val_accuracy:
0.8935
Validation accuracy: 89.35%
```


1716
8
205
71
0
1
True label
0
1
Predicted label
1600
1400
1200
1000
800
600
400
200


Accuracy on Validation Data
Accuracy on validation data
0.895
0.890
0.885
0.880
0.875
Accuracy
1
2
3
4
5
6
Epochs