Monochrome Dreams Classification

Task

Am obținut setul de date și informația necesara rezolvării problemei de pe Kaggle. Scopul proiectului este antrenarea modelului pentru clasificarea unor imagini monocrome în noua clase. Setul de date de antrenare conține 30,000 de imagini, cel de validare 5,000, iar cel de testare 5,000.

Preprocesarea datelor

Pentru a manipula datele de training, validare și test, am încărcat fiecare imagine folosindu-mă de metoda cv2.imread() din librăria OpenCV și am converit-o la grayscale, urmând că după aceasta etapa imaginea să fie codificata că un numpy.ndarray de float-uri utilizandu-mă de funcția img_as_float din utilitarul skimage..

```
for img in tqdm(os.listdir(path_train)):
    img_array = cv2.imread(os.path.join(path_train,img), cv2.IMREAD_GRAYSCALE)
    img_array = img_as_float(img_array)
    img_array = img_array.flatten()
```

Modelul a fost antrenat pe datele de training (training_data) și testat în vederea obținerii acurateții pe datele de validare (validation_data). După calcularea acurateții, modelul a fost testat pe datele de test(test_data) în vederea obținerii clasificării necesare submisiei.

Clasificatori folositi si performanta acestora:

1. **KNNClassifier**

Acest clasificator poate fi folosit atât pentru probleme de regresie, cât și pentru probleme de clasificare. Principiul pe care se bazează este următorul: pentru un punct nou care trebuie clasificat, algoritmul vă caută cele mai apropiate k puncte de acesta prin calcularea de distante. Cele k puncte găsite sunt deja etichetate. În funcție de acest criteriu, se vă alege clasa majoritara. În cazul în care apare ambiguitatea că mai multe clase să fie “majoritare”, se va alege una în mod aleator. Am ales să aplic acest clasificator luând în considerare cei mai apropiați noua vecini.

Pe acest model, acuratețea a fost calculata ținând cont de numărul de predicții valide pentru fiecare label obținut în urma clasificării fiecărei imagini.

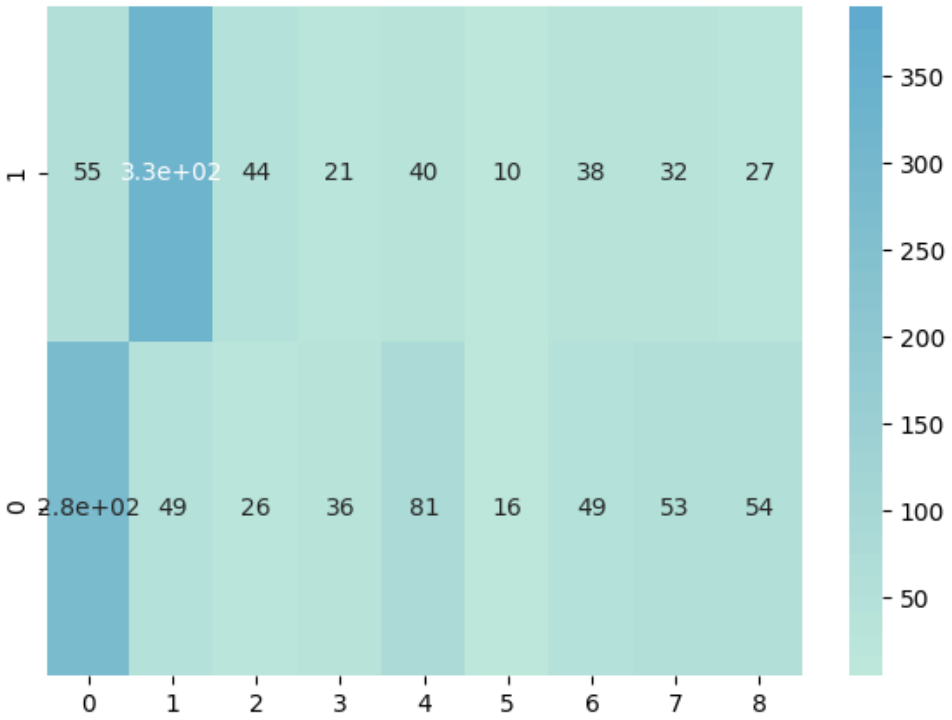
```
correct_count = np.sum(pred_labels == y_validation)
total_count = len(y_validation)

accuracy = correct_count / total_count
```

2. Support Vector Machine

SVM este un model liniar pentru probleme de regresie și clasificare. Ideea acestuia este că el creează o linie sau un hiperplan care separa datele în clase. Astfel, algoritmul găsește cele mai apropiate puncte de linie din toate cele 9 clase. Aceste puncte găsite se numesc vectori de support. Următorul pas este calcularea distantelor dintre linie și vectorii de support. Hiperplanul petru care marginea este maxima este hiperplanul optim. Pentru această metodă, am folosit un SVM cu kernel ‘linear’ și un parametru de regularizare de 1.

Matricea de confuzie obițnută pe datele de validare:



Pentru stocarea datelor necesare submisiei, a matricei de confuzie, m-am folosit de dataframe-uri din pandas.

Dependințe:

- SVM din sklearn
- pandas
- Matplotlib.pyplot
- seaborn

Concluzie

Scorul obținut pe fiecare dintre clasificatori, conform Kaggle este:

- KNNClassifier -> acuratețe 0.48400
- SVM -> acuratețe 0.62480