Nume: Barbu Andrei-Catalin
Grupa: 331AA

TEMA 1 - TIA

I. Prezentarea temei

Tema aleasa este Digit Classifier. Algoritmul de clasificare este antrenat pentru a recunoaste cifrele scrie de mana din diferite poze.

II. Modul de obtinere si organizare a setului de date

Setul de date este contributie proprie. In continuare voi prezenta pasii urmati pentru alcatuirea setului de date:

1. Crearea imaginilor: am utilizat MS Paint pentru a scrie fiecare cifra de mai multe ori, cu grosimi diferite, modificand totodata modul de scriere pentru o mai mare diversitate. Am realizat in total 500 de imagini cu cifre de la 0 la 9, de dimensiune 200x200 pixeli;
2. Redimensionarea imaginilor: dupa o scurta documentatie am observat ca nu este nevoie de o dimensiune atat de mare, avand in vedere tema aleasa, si am optat spre redimensionarea acestora la 28x28 pixeli, folosind o aplicatie numita Image Resizer;
3. Redenumirea imaginilor: fiecare clasa de imagini trebuie sa aiba o denumire proprie pentru a adauga label-uri cu usurinta, asa ca am denumit imagine cu numele cifrei pe care o reprezinta;
4. Alcatuirea csv-ului: am creat un nou proiect in Python care sa citeasca fiecare imagine si sa-l adauge label-ul si fiecare pixel intr-un csv. Label-ul a fost adaugat cu ajutorul unui dictionar, care avea ca si key numele pozei, iar ca valoare cifra pe care o reprezinta (exemplu: “zero”: “0”);
5. Formatarea csv-ului: setul de date arata dezordonat, asa ca am sters spatiile din csv si l-am modificat astfel incat prima coloana sa contina label-urile, iar a doua coloana sa contina toti pixelii pozei separati prin spatiu.

III. Algoritmul utilizat si parametrii utilizati

Am utilizat algoritmul Random Forest pentru a rezolva problema de clasificare a cifrelor. Acesta functioneaza in felul urmator:

1. Antrenarea cu setul etichetat: algoritmul incarca imaginile rezprezentate ca un vector de pixeli impreuna cu etichetele lor. CSV-ul pare ca are valoarea 255 constanta, insa acest lucru se datoreaza faptului ca primii pixeli din fiecare imagine sunt albi (255 fiind valoarea pentru alb)

	Labels	Pixels
1	8	255 255 255 255 255 255 255 255 255 255 255 255 2…
2	8	255 255 255 255 255 255 255 255 255 255 255 255 2…
3	8	255 255 255 255 255 255 255 255 255 255 255 255 2…
4	8	255 255 255 255 255 255 255 255 255 255 255 255 2…
5	8	255 255 255 255 255 255 255 255 255 255 255 255 2…
6	8	255 255 255 255 255 255 255 255 255 255 255 255 2…
7	8	255 255 255 255 255 255 255 255 255 255 255 255 2…
8	8	255 255 255 255 255 255 255 255 255 255 255 255 2…
9	8	255 255 255 255 255 255 255 255 255 255 255 255 2…
10	8	255 255 255 255 255 255 255 255 255 255 255 255 2…

Figure 1. O parte din csv-ul cu setul de date

2. Extragerea de caracteristici: in cazul dat, caracteristicile sunt reprezentate de pixelii imaginilor de dimensiune 28x28, care au fost redimensionate la un vector de 784 de caracteristici pentru a fi utilizate in modelul RandomForestClassifier.

```
def load():
    data = pd.read_csv(dataset_path)
    pixels = data['Pixels'].tolist()
    labels = data['Labels'].tolist()
    width, height = 28, 28
    faces = []
    for pixel_sequence in pixels:
        face = [int(pixel) for pixel in pixel_sequence.split(' ')]
        face = np.asarray(face).reshape(width, height)
        a = face
        face = np.resize(face.astype('uint8'), image_size)
        faces.append(face.astype('float32'))
    faces = np.asarray(faces)
    pixels = faces
    faces = np.expand_dims(faces, -1)
    return pixels, labels

pixels, labels = load()
pixels = pixels.reshape(500, -1)
```

3. Impartirea setului de date: am utilizat functia train_test_split pentru a imparti setul de date in felul urmator: 75% setul de antrenament, 15% setul de validare si 10% setul de testare.

```
train_ratio=0.75
val_ratio=0.15
test_ratio=0.10
x_train, x_aux, y_train, y_aux = train_test_split(pixels, labels, test_size=(1
-train_ratio))
x_val, x_test, y_val, y_test = train_test_split(pixels, labels, test_size=(1 -
val_ratio))
```

4. Initializarea RandomForestClassifier: un obiect “RandomForestClassifier” este initializat cu parametrii “n_estimators” si “random_state”.
    - “n_estimators” reprezinta numarul de arbori de dezitie. Cu cat acesta este mai mare, cu atat modelul poate avea mai multa complexitate si poate imbunatati performanta pe setul de antrenare. Totusi, un numar prea mare de arbori va creste timpul de antrenare. Am incercat mai multe variante ale acestui parametru si am ajuns la concluzia ca 300 este cel mai potrivit, avand constant o acuratete de peste 95%.
    - “random_state” este folosit pentru a initializa generatorul de numere pseudo-aleatoare. Valoarea sa nu este importanta, ci folosirea aceleiasi in mai multe rulari pentru a asigura coerenta rezultatelor.

```
model = RandomForestClassifier(n_estimators=300, random_state=15)
```

5. Antrenarea modelului: setul de antrenarea este folosit pentru a antrena modelul “RandomForestClassifier”. Fiecare arbore este antrenat pe o submultime aleatoare a setului de antrenare, iar predictiile acestora sunt combinate in final.

```
model.fit(x_train, y_train)
```

6. Evaluarea modelului: dupa antrenare, performanta modelului este evaluate pe un set de validare. Acuratetea este o metrica comuna pentru a evalua performanta clasificatorului. Setul de validare m-a ajutat la ajustarea parametrilor modelului.

```
y_pred=model.predict(x_val)
print("Validation_accuracy")
print(model.score(x_val,y_val))
print(accuracy_score(y_pred, y_val))
```

7. Testarea modelului: dupa ce parametrii modelului au fost alesi si acuratetea a ajuns la nivelul dorit, am utilizat setul de test pentru a verifica modelul si am afisat una dintre prezicerile din acesta.

```
print("\nTest accuracy")
y_pred2 = model.predict(x_test)
print(model.score(x_test, y_test))
print(accuracy_score(y_pred2, y_test))
y_predicted = model.predict(x_test[5].reshape(1, -1))
pixel = x_test[5]
label = y_predicted
pixel = pixel.reshape((28,28))
plt.title(f'Predicted digit is {label}'.format(label=label))
plt.imshow(pixel, cmap='gray')
plt.show()
```

IV. Rezultate

Rezultatele obtinute sunt urmatoarele:

1. Setul de validare
   1.1. Acuratete: 84%
   1.2. Raport de clasificare

```
Raport de clasificare:
              precision    recall  f1-score   support

           0       0.83      1.00      0.91         5
           1       1.00      0.75      0.86         4
           2       0.92      0.80      0.86        15
           3       0.62      1.00      0.77         5
           4       1.00      0.55      0.71        11
           5       0.71      0.83      0.77         6
           6       0.71      1.00      0.83         5
           7       1.00      0.82      0.90        11
           8       0.83      1.00      0.91         5
           9       0.80      1.00      0.89         8

    accuracy                           0.84        75
   macro avg       0.84      0.87      0.84        75
weighted avg       0.87      0.84      0.84        75
```

   1.3. Matrice de confuzie

```
Matricea de confuzie:
 [[ 5  0  0  0  0  0  0  0  0  0]
 [ 0  3  1  0  0  0  0  0  0  0]
 [ 0  0 12  2  0  0  0  0  1  0]
 [ 0  0  0  5  0  0  0  0  0  0]
 [ 1  0  0  0  6  2  1  0  0  1]
 [ 0  0  0  0  0  5  1  0  0  0]
 [ 0  0  0  0  0  0  5  0  0  0]
 [ 0  0  0  1  0  0  0  9  0  1]
 [ 0  0  0  0  0  0  0  0  5  0]
 [ 0  0  0  0  0  0  0  0  0  8]]
```

`

2. Setul de testare

2.1. Acuratete: 80%

2.2. Raport de clasificare

```
Raport de clasificare:
              precision    recall  f1-score   support

           0       1.00      0.86      0.92         7
           1       1.00      0.57      0.73         7
           2       1.00      0.60      0.75         5
           3       0.67      0.86      0.75         7
           4       1.00      0.67      0.80         6
           5       0.50      1.00      0.67         2
           6       1.00      1.00      1.00         4
           7       0.75      1.00      0.86         3
           8       0.71      1.00      0.83         5
           9       0.60      0.75      0.67         4

    accuracy                           0.80        50
   macro avg       0.82      0.83      0.80        50
weighted avg       0.86      0.80      0.80        50
```

2.3.Matrice de confuzie

```
Matricea de confuzie:
 [[6 0 0 0 0 0 0 0 0 1]
 [0 4 0 2 0 0 0 1 0 0]
 [0 0 3 1 0 0 0 0 1 0]
 [0 0 0 6 0 0 0 0 1 0]
 [0 0 0 0 4 1 0 0 0 1]
 [0 0 0 0 0 2 0 0 0 0]
 [0 0 0 0 0 0 4 0 0 0]
 [0 0 0 0 0 0 0 3 0 0]
 [0 0 0 0 0 0 0 0 5 0]
 [0 0 0 0 0 1 0 0 0 3]]
```

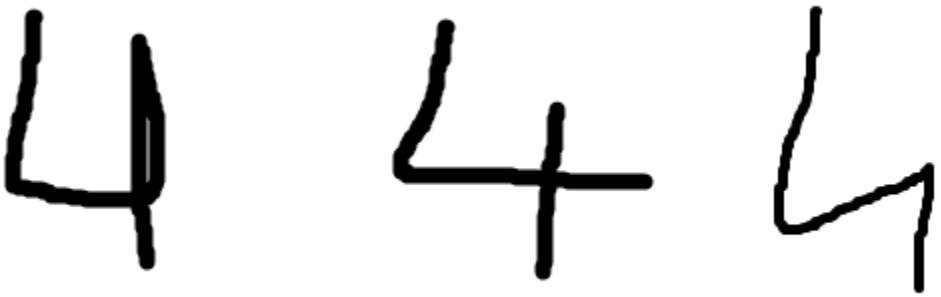
Din raportul de clasificare pentru testare putem observa urmatoarele lucruri:

- Cea mai mare precizie este de 1 pentru clasele 0, 1, 2, 4, 6. Acest lucru inseamna ca toate predictiile positive sunt corecte

- Cea mai mica precizie este de 0.5 pentru clasa 5. Totusi, consider ca precizia mica este cauzata de numarul mic de exemple pentru aceasta clasa (support): 2. Putem vedea ca in raportul de clasificare pentru validare support-ul acestei clase este egal cu 6, iar precizia creste semnificativ (0.71).
- Clasa 1 are cel mai mic recall (0.57), care indica faptul ca doar 57% din exemplele positive au fost identificate corect. Putem observa din matricea de confuzie ca 1 a fost vazut de 2 ori ca 3 si o data ca 7. Confuzia cu 7 se poate datora faptului ca cifra 7 este chiar destul de asemanatoare cu cifra 1.

Din raportul de clasificare pentru vaidare putem observa urmatoarele lucruri:

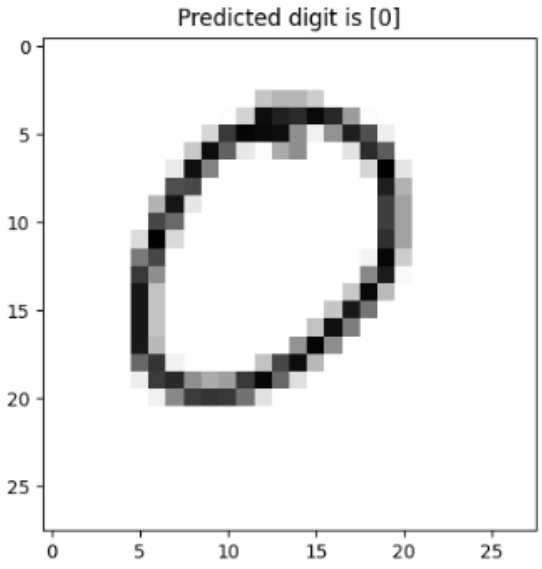
- Clasa 4 are un recall de 0.55, care indica faptul doar 55% din exemplele pozitive au fost identificate corect. Acest lucru se poate datora modurilor diferite de scriere a cifrei 4 din setul de date:

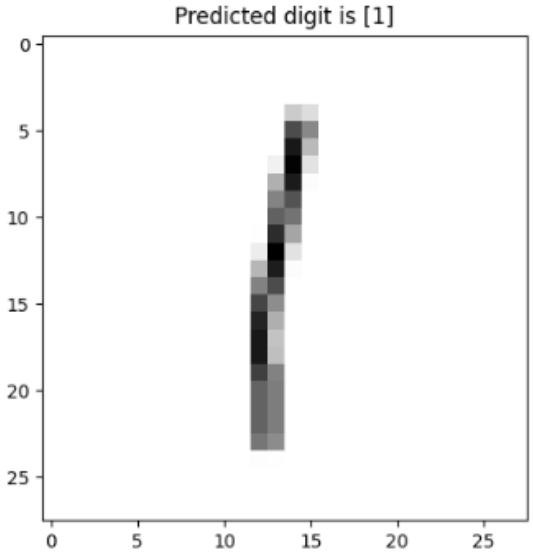


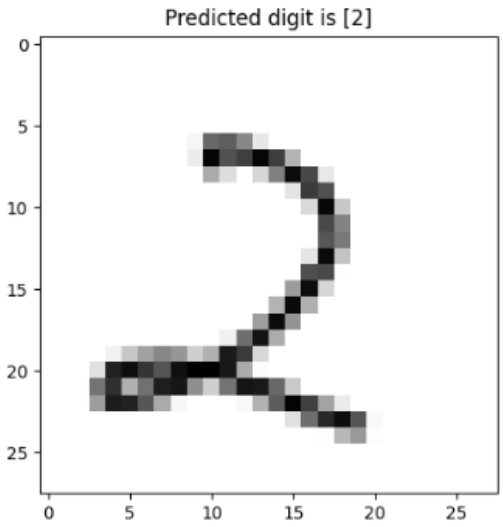
- Toatete F1-score-urile sunt mai mari de 0.7, lucru care sugereaza un echilibru destul de bun dintre precizie si recall
- Diferenta dintre valorile support-ului la fiecare clasa este posibil sa fie prea mare (clasa 2 – 15; clasa 1 – 4);

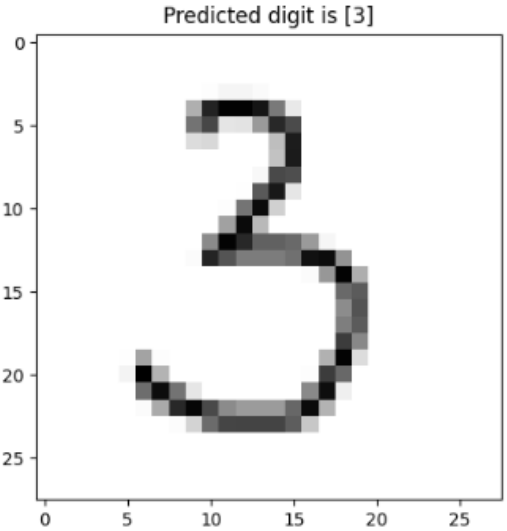
In concluzie, modelul se descurca bine pe tema aleasa, algoritmul RandomForestClassifier fiind potrivit pentru clasificarea de cifre. Totusi, consider ca puteam obtine o precizie mai mare daca aveam un set de date mai mare si diferente mai mici intre pozele pentru fiecare clasa.

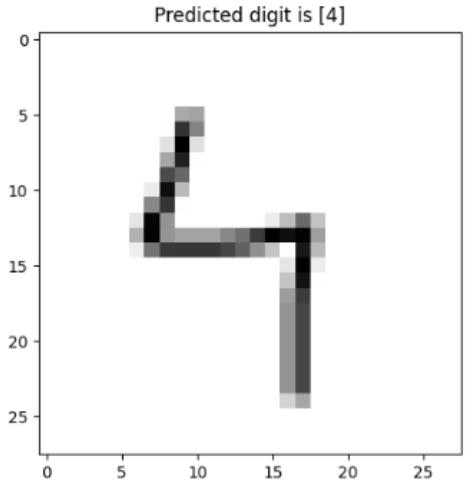
V. Exemple:

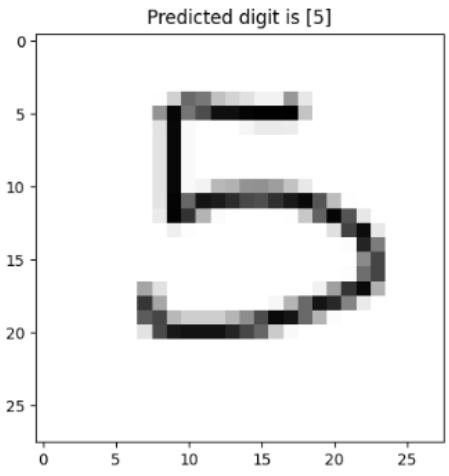

Predicted digit is [0]

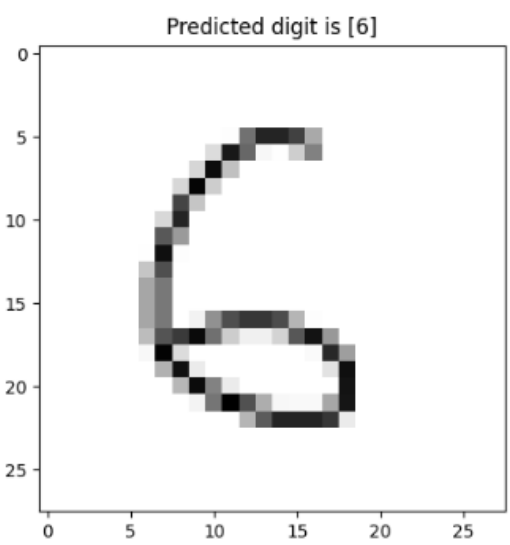

Predicted digit is [1]

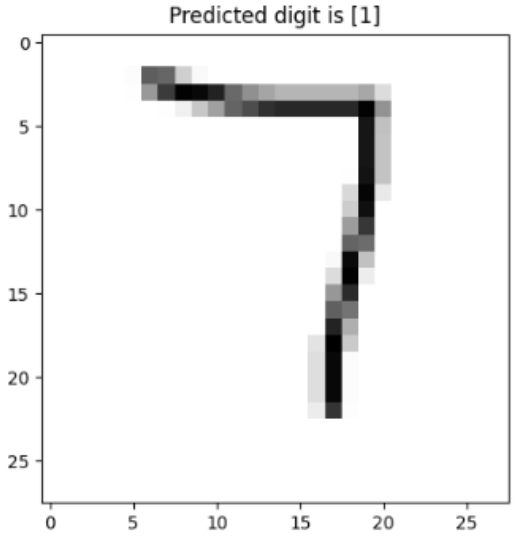

Predicted digit is [2]

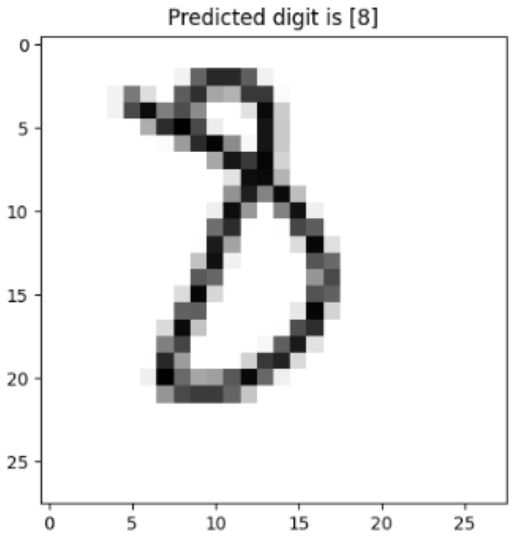

Predicted digit is [3]

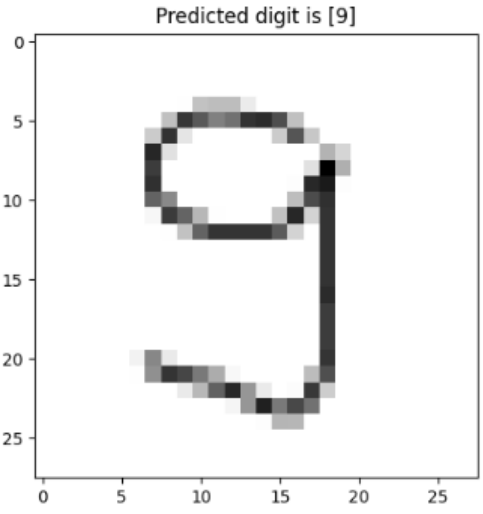

Predicted digit is [4]


Predicted digit is [5]


Predicted digit is [6]


Predicted digit is [1]


Predicted digit is [8]


Predicted digit is [9]

VI. Bibliotecile Python utilizate

Am utilizat urmatoarele biblioteci:

1. Pandas: biblioteca pentru manipularea si analiza datelor, folosita pentru a incarca setul de date.
2. Numpy: biblioteca pentru calcul numeric, folosita pentru lucrul cu pixelii imaginilor
3. Matplotlib: biblioteca pentru crearea de grafice si vizualizari, folosita pentru a afisa rezultatele predictiilor
4. Sklearn: biblioteca pentru invatare automata, folosita pentru a imparti datele in seturi de antrenare, validare si testare, a implementa algoritmul RandomForestClassifier si a evalua modelul creat

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
```