Ion Bianca-Andreea
333AA

Traffic Sign Classification

-Implementare-

Temă: clasificarea semnelor de circulație (trecere de pietoni, cedează trecerea, stop).

Dataset:
View dataset

- Creat prin combinarea imaginilor dintr-un set de date cu mai multe semne de circulație și imagini descărcate de pe google.
- Etichetarea s-a făcut manual
- Alcătuit din 3 foldere: *crosswalk*(117 imagini), *give-way*(108 imagini), *stop*(110 imagini)
- Preprocesare:
  - Toate imaginile au extensia *.png*
  - Imaginile sunt modificate pentru a avea rezoluția *50*x*50 / 150x150*
  - Imaginile sunt convertite într-un format ușor de procesat de un algoritm de machine learning: *.flatten()*
- Impărțirea în seturile de antrenare, validare și testare se face cu ajutorul funcției *train_test_split()*, 80% din dataset fiind păstrat pentru antrenare, 10% pentru validare și 10% pentru testare; împărțirea se face prin apelarea de 2 ori a funcției *train_test_split()* pentru a evita pierderea de date

Algoritm:

I. K-Nearest Neighbors

- Pentru antrenare am folosit clasificatorul *K-Nearest Neighbors* – o metodă de învățare supervizată.
- Acest algoritm este unul leneș, deoarece nu creează un model în comparație cu alte metode de clasificare, ci prezice direct pe baza datelor de antrenament.
- Rezultatul utilizării algoritmului *k-NN* în operații de clasificare reprezintă apartenența la o clasă. Un obiect este clasificat cu votul majorității vecinilor săi, obiectul fiind atribuit clasei cele mai frecvente din *k-NN*.
- Avantaje:
  - Este simplu de implementat.
  - Nu este necesară nicio pregătire a datelor înainte.
  - Este flexibil în alegerea tipurilor de distanțe sau a caracteristicilor.
- Dezavantaje:
  - Clasificatorul *k-NN* este nepotrivit pentru seturi de dimensiuni mari.
  - Alegerea valorii lui *K* poate fi dificilă.
  - Nu face predicții pentru lucruri rare.
- *KNeighborsClassifier(n_neighbors, weights, algorithm, leaf_size, p)*
  - *n_neighbors* – numărul vecinilor considerați
  - *weights* – modul în care sunt ponderate contribuțiile vecinilor
  - *algorithm* – algoritmul folosit pentru a calcula cei mai apropiați vecini
  - *leaf_size* – dimensiunea frunzei utilizată pentru algoritmii BallTree sau KDTree
  - *p* – parametrul de putere pentru metrica Minkowski

Ion Bianca-Andreea
333AA

II. Naïve Bayes

- Pentru antrenare am folosit clasificatorul *Naïve Bayes*– o metodă de învățare supervizată.
- Tehnică de clasificare statistică.
- Se bazează pe Teorema lui Bayes: $P(A|B) = \frac{P(A|B)*P(A)}{P(B)}$
- Tehnica presupune existența unui set de date deja clasificate.
- Scopul este reprezentat de determinarea clasei în care se încadrează o nouă instanță de același tip cu cele deja existente.
- Clasificatorul este *naiv*, deoarece presupune că toate atributele unui punct sunt independente unele de altele, însemnând că prezența sau absența unei caracteristici nu afectează probabilitatea unei alte caracteristici.
- Avantaje:
  - Funcționează rapid și poate prezice cu ușurință clasa.
  - Potrivit pentru rezolvarea problemelor de predicție cu mai multe clase.
  - Nu necesită un volum mare de date de antrenare.
- Dezavantaje:
  - Presupune că toate caracteristicile sunt independete, ceea ce se întâmplă rar în viața reală.
  - Algoritmul se confruntă cu *problema de frecvență zero*.
- *GaussianNB(priors, var_smoothing)*:
  - *priors* – probabilitățile anterioare ale claselor
  - *var_smoothing* – porțiunea din cea mai mare varianță a tuturor caracteristicilor, care se adaugă la variații pentru stabilitatea calculului

III. K-Nearest Neighbors + Naïve Bayes

- Pentru antrenare am folosit clasificatorul *K-Nearest Neighbors*.
- Pentru filtrarea rezultatelor obținute cu algoritmul *KNN* am utilizat clasificatorul *Naïve Bayes*.

Ion Bianca-Andreea
333AA

Librării:

- cv2
- numpy
- sklearn
  - model_selection ➔ train_test_split
  - neighbors ➔ KNeighborsClassifier
  - naive_bayes ➔ GaussianNB
  - metrics ➔ accuracy_score, classfication_report, confusion_matrix
- matplotlib

Acurateţe:

I. K-Nearest Neighbors

Pentru parametrii aleşi, *KNeighborsClassifier(n_neighbors=1, weights=distance, algorithm=auto, p=2)* am obţinut un procent de acurateţe de:

- Validare: 57.58%
- Testare: 67.65%

II. Naïve Bayes

Pentru parametrii aleşi, *GaussianNB(priors=None, var_smoothing=1)* am obţinut un procent de acurateţe de:

- Validare: 54.55%
- Testare: 55.88%

III. K-Nearest Neighbors + Naïve Bayes

Pentru parametrii aleşi, *KNeighborsClassifier(n_neighbors=1, weights=distance, algorithm=auto, p=2)* am obţinut un procent de acurateţe de:

- Validare KNN: 63.64%
- Testare KNN: 61.76%
- Testare NB pe KNN: 58.82%

Raport de clasificare pentru setul de testare:

- Precision – capacitatea clasificatorului de a nu eticheta un eşantion negativ ca fiind pozitiv. $\frac{T_p}{Tp+F_p}$
- Recall – capacitatea clasificatorului de a găsi toate probele pozitive. $\frac{T_p}{Tp+F_n}$
- F1-Score – o medie armonică ponderată a preciziei şi a sensibilităţii, unde un scor F-beta atinge cea mai bună valoare la 1 şi cel mai slab scor la 0.
- Support - numărul de apariţii ale fiecărei clase în *y_true*.

Ion Bianca-Andreea
333AA

Afișare rezultate:

I. *KNN*

```
Classification Report for Test Set (KNN):
              precision    recall  f1-score   support

   crosswalk       0.67      0.67      0.67        12
    give-way       0.70      0.64      0.67        11
        stop       0.58      0.64      0.61        11

    accuracy                           0.65        34
   macro avg       0.65      0.65      0.65        34
weighted avg       0.65      0.65      0.65        34
```

II. Naïve Bayes

```
Classification Report for Test Set (NB):
              precision    recall  f1-score   support

   crosswalk       0.53      0.75      0.62        12
    give-way       0.75      0.55      0.63        11
        stop       0.44      0.36      0.40        11

    accuracy                           0.56        34
   macro avg       0.57      0.55      0.55        34
weighted avg       0.57      0.56      0.55        34
```

III. KNN + Naïve Bayes

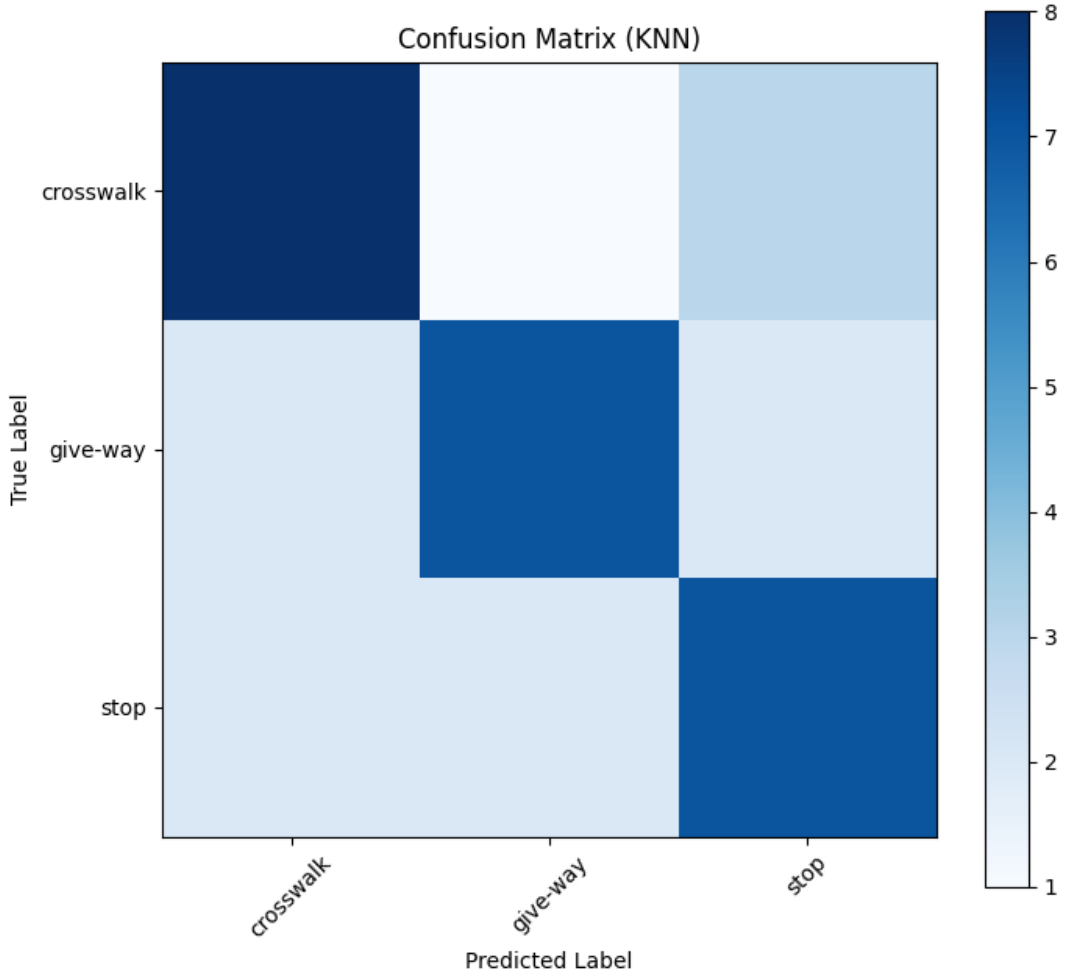
```
Classification Report for Test Set (Naive Bayes on KNN Predictions):
              precision    recall  f1-score   support

   crosswalk       0.50      0.67      0.57        12
    give-way       0.75      0.55      0.63        11
        stop       0.40      0.36      0.38        11

    accuracy                           0.53        34
   macro avg       0.55      0.53      0.53        34
weighted avg       0.55      0.53      0.53        34
```

Ion Bianca-Andreea
333AA

Matrice de confuzie:

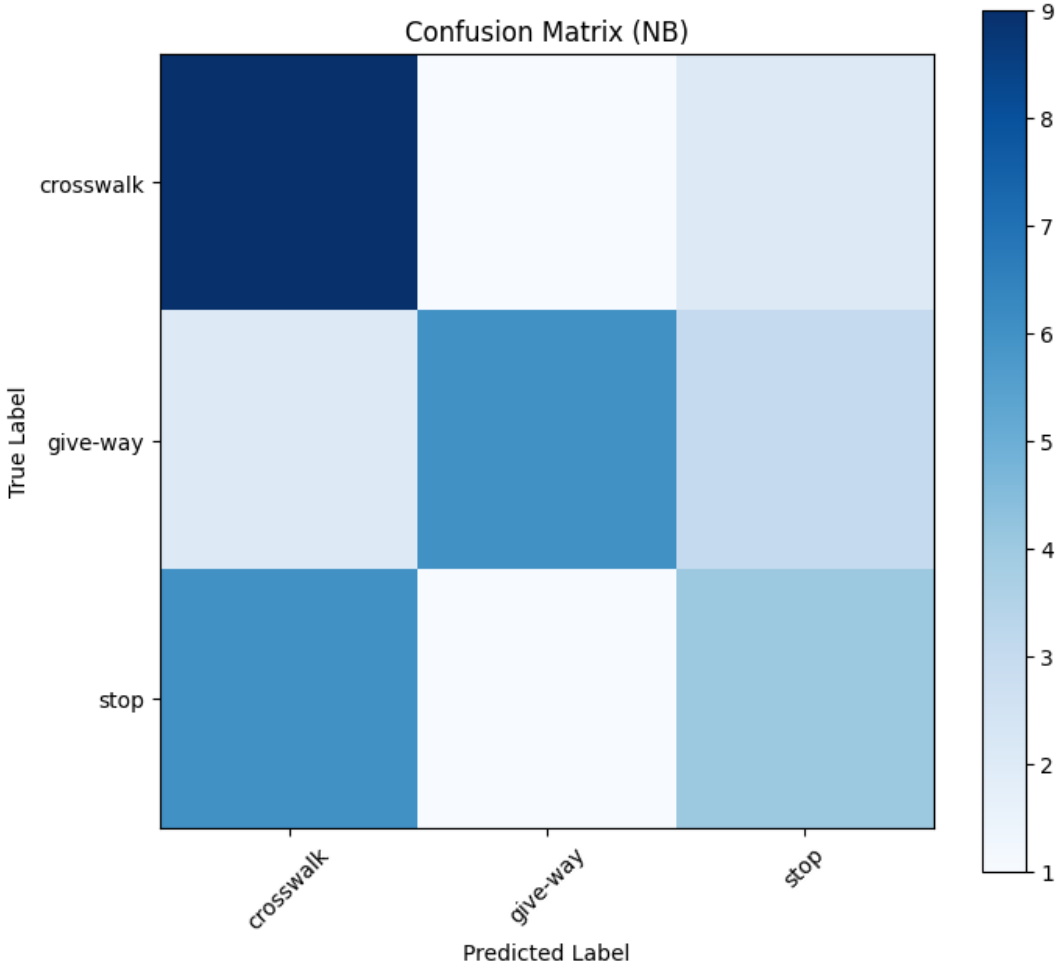
I. *KNN*

```
Confusion Matrix for Test Set (KNN):
 [[8 1 3]
 [2 7 2]
 [2 2 7]]
```



II. Naïve Bayes

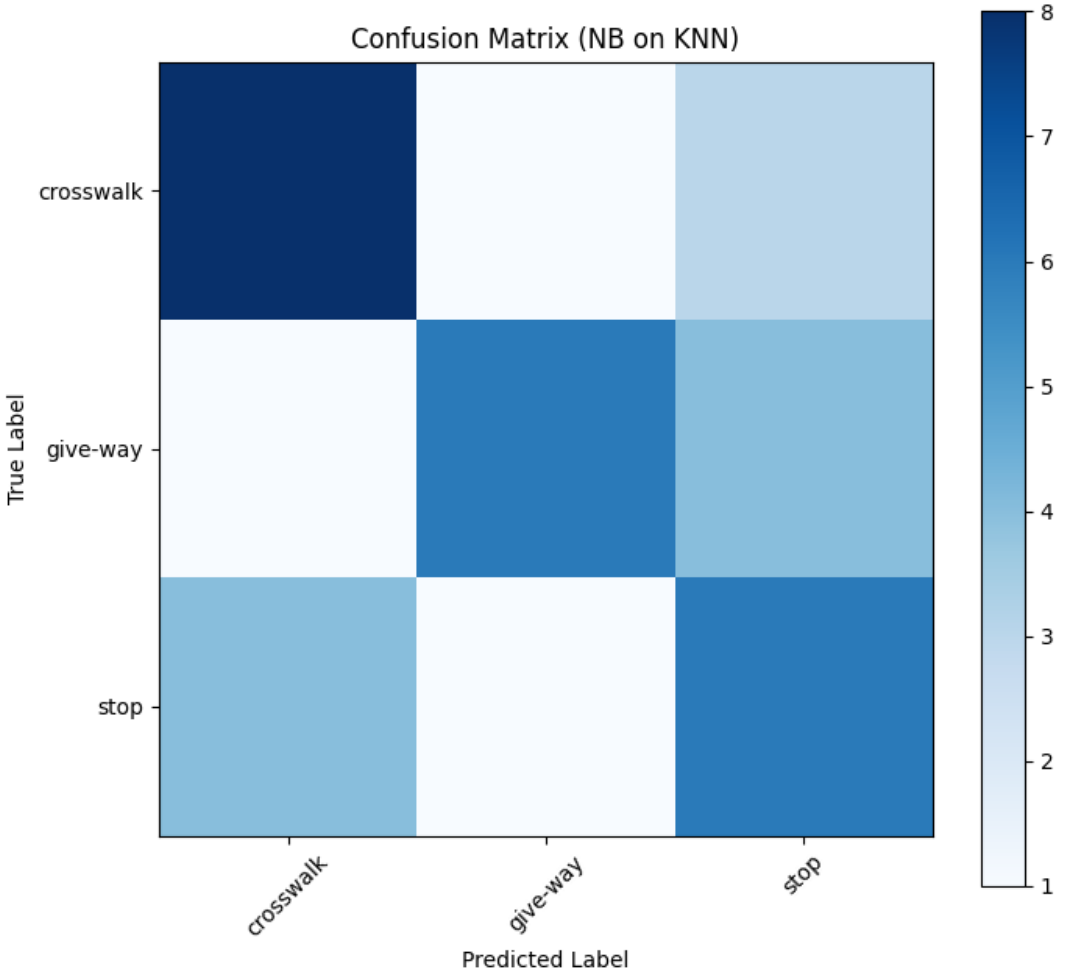
```
Confusion Matrix for Test Set (NB):
 [[9 1 2]
 [2 6 3]
 [6 1 4]]
```



Ion Bianca-Andreea
333AA

III. KNN + Naïve Bayes

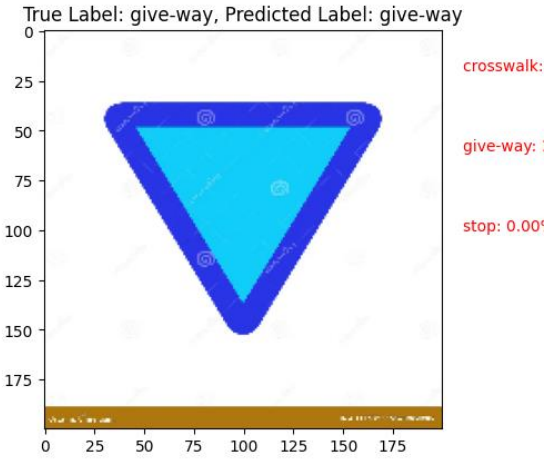
```
Confusion Matrix for Test Set (Naive Bayes on KNN Predictions):
 [[8 1 3]
 [1 6 4]
 [4 1 6]]
```

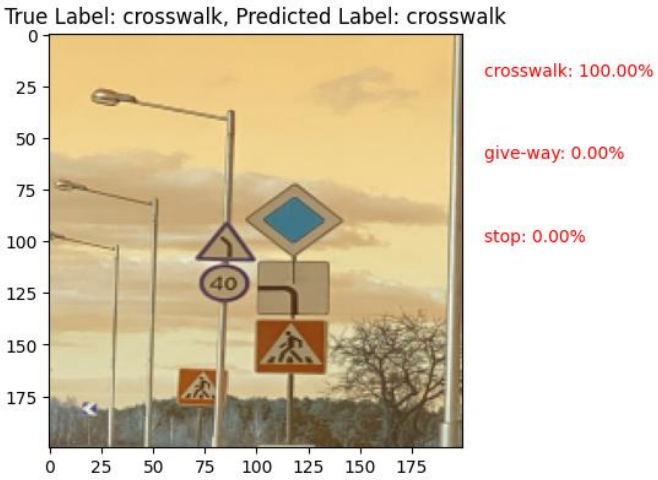


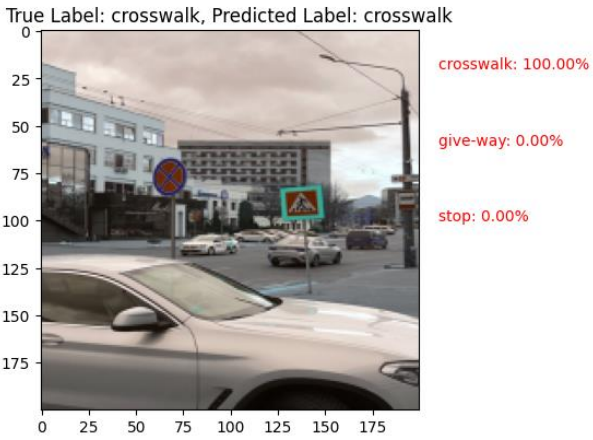
Ion Bianca-Andreea
333AA

Afișare teste:

View results







Ion Bianca-Andreea
333AA

Rezultate:

I. KNN

- Pentru *n_neighbors=5, weights=distance, algorithm=auto, p=2* am obținut o acuratețe a testelor de **58.82%**
- Pentru *n_neighbors=3, weights=distance, algorithm=auto, p=2* am obținut o acuratețe a testelor de **61.76%**
- Pentru *n_neighbors=3, weights=distance, algorithm=ball_tree, leaf_zise=20, p=2* am obținut o acuratețe a testelor de **61.76%**
- Pentru *n_neighbors=5, weights=uniform, algorithm=brute, p=2* am obținut o acuratețe a testelor de **55.88%**
- Pentru *n_neighbors=1, weights=distance, algorithm=auto, p=2* am obținut o acuratețe a testelor de **61.76%**
- Prin modificarea dimensiunii imaginilor, păstrând parametrii de mai sus, am obținut:
  - *300x300* ➔ acuratețea testelor de **58.82%**
  - *200x200* ➔ acuratețea testelor de **61.76%**
  - *150x150 + interpolation = cv.INTER_AREA* ➔ acuratețea testelor de **67.65%**
  - *50x50* ➔ acuratețea testelor de **64.71%**

II. Naïve Bayes

- Pentru *GaussianNB(priors=None, var_smoothing=1e-1)* am obținut o acuratețe a testelor de **50.00%**
- Pentru *GaussianNB(priors=None, var_smoothing=2)* am obținut o acuratețe a testelor de **52.94%**
- Pentru *GaussianNB()* am obținut o acuratețe a testelor de **47.06%**
- Pentru *GaussianNB(priors=None, var_smoothing=1)* am obținut o acuratețe a testelor de **55.88%**
- Pentru *GaussianNB(priors=[0.3, 0.3, 0.4], var_smoothing=1)* am obținut o acuratețe a testelor de **55.88%**

III. KNN + Naïve Bayes

- Pentru *KNeighborsClassifier(n_neighbors=5), GaussianNB()* am obținut o acuratețe a testelor de **55.88%**
- Pentru *KNeighborsClassifier(n_neighbors=3, weights=distance, algorithm=auto, p=2), GaussianNB(priors=None, var_smoothing=1)* am obținut o acuratețe a testelor de **52.94%**
- Pentru *KNeighborsClassifier(n_neighbors=3, weights=distance, algorithm=ball_tree, leaf_size=20, p=2), GaussianNB(priors=None, var_smoothing=1e-9)* am obținut o acuratețe a testelor de **58.82%**

Ion Bianca-Andreea
333AA

Concluzii & Observații:

- Clasificatorii *KNN* și *Naïve Bayes* obțin o acuratețe mai scăzută față de clasificatorul *SVC* utilizat în tema 1. Acest lucru se poate datora setului de date (nu sunt suficiente imagini sau acestea nu sunt potrivite pentru o antrenare foarte bună).
- Acuratețea este influențată de parametrii funcțiilor, cât și de preprocesarea datelor (rezoluția imaginilor).
- Modelul potrivit pentru setul meu de date s-a dovedit a fi clasificatorul *SVC*. Acest fapt se poate datora robusteții algoritmului (este mai puțin sensibil la valori aberante, poate face distincția mai bine între un pattern adevărat și zgomot).
- Clasificatorul *KNN* a mai fost utilizat în lucrări de clasificare a semnelor de circulație.
- Am adaptat algoritmul utilizat la Tema 1 pentru rezolvarea cerințelor impuse. Am modificat parametrii funcțiilor *KNeighborsClassifier()* și *GaussianNB()* pentru a obține cea mai bună acuratețe.

Ion Bianca-Andreea
333AA

Referințe:

- sklearn.neighbors.KNeighborsClassifier
- naive_bayes
- sklearn.naive_bayes.GaussianNB
- Confusion matrix
- Classification report
- Traffic Sign Detection
- Laborator
- Curs