**Clasificarea animalelor**

*KNN & Naive Bayes*

Tema 2 – Tehnici de învățare automată

Andrei Antonia-Ștefania

333AA

Cuprins

1. Introducere: Motivație, descriere generală temă, tip de algoritm și algoritm ales.
2. Metodologie: Descriere implementare și particularități algoritm.
3. Rezultate obținute: Rezultate EXPLICATE obținute la antrenare, validare și testare. Trebuie să conțină MATRICEA de CONFUZIE explicată și INDICATORI de PERFORMANȚĂ.
4. Concluzii: Concluzii PROPRII desprinse din lucrarea efectuată și comentate pe baza rezultatelor obținute.

1. Introducere

Motivație

Clasificarea imaginilor este un domeniu esențial în inteligența artificială, utilizat în aplicații precum recunoațterea facială, clasificarea obiectelor sau analiza medicală. Scopul acestui proiect este de a analiza și compara performanța a doi algoritmi de clasificare, K-Nearest Neighbors (KNN) și Naive Bayes, în identificarea a 20 de categorii de animale dintr-o bază de date de imagini. Dataset-ul a fost descărcat de pe site-ul : <https://www.kaggle.com/datasets/iamsouravbanerjee/animal-image-dataset-90-different-animals/data> .

Dataset:

Alcătuit din 20 de subfoldere, denumite: bear,butterfly,cat,cow,deer,dog,dolphin,elephant,flamingo,fox,horse,lion,panda, parrot,penguin,pig,sheep,tiger,turtle,zebra. Acestea reprezintă categoriile de animale.

Descriere generală a temei

Proiectul implică utilizarea a trei seturi de date (“Train”, “Validation” și “Test”) pentru a antrena, valida și testa algoritmii. Imaginile sunt preprocesate prin redimensionare și transformare în vectori de caracteristici.

Tip de algoritmi și algoritmi aleși

1. KNN (K-Nearest Neighbors): Un algoritm de clasificare bazat pe similaritatea dintre vectori, unde fiecare punct este clasificat conform celor mai apropiați K vecini.

2. Naive Bayes: Un clasificator probabilistic bazat pe Teorema lui Bayes, care presupune independența caracteristicilor.

2. Metodologie

Descriere implementare

* Preprocesarea datelor:
  + Imaginile din fiecare director (Train, Validation, Test) au fost redimensionate la 128x128 pixeli.
  + Fiecare imagine a fost convertită într-un vector de caracteristici (aplatizare).
* Implementare KNN:

- S-a utilizat biblioteca scikit-learn.

- Parametrii: numărul de vecini K=3, deoarece funcționează bine pentru seturi de date moderate, unde distanțele dintre puncte sunt relativ uniforme. Într-un astfel de caz, alegerea unui k mic (cum ar fi 3) ajută la menținerea unui nivel decent de acuratețe fără a crește excesiv complexitatea.

- Algoritmul clasifică imaginile în funcție de cei mai apropiați 3 vecini din spațiul caracteristicilor.

* Implementare Naive Bayes:

- S-a utilizat modelul GaussianNB din scikit-learn.

- Algoritmul calculează probabilitățile pentru fiecare clasă și clasifică imaginea conform probabilității maxime.

Particularități ale algoritmilor

- KNN este un algoritm fără fază de antrenare explicită, dar costisitor computațional în faza de clasificare.

- Naive Bayes presupune independența caracteristicilor, ceea ce poate fi o limitare în cazul imaginilor cu caracteristici corelate.

3. Rezultate obținute

Performanța algoritmilor pe seturile de date

* **KNN**

1.Antrenare:

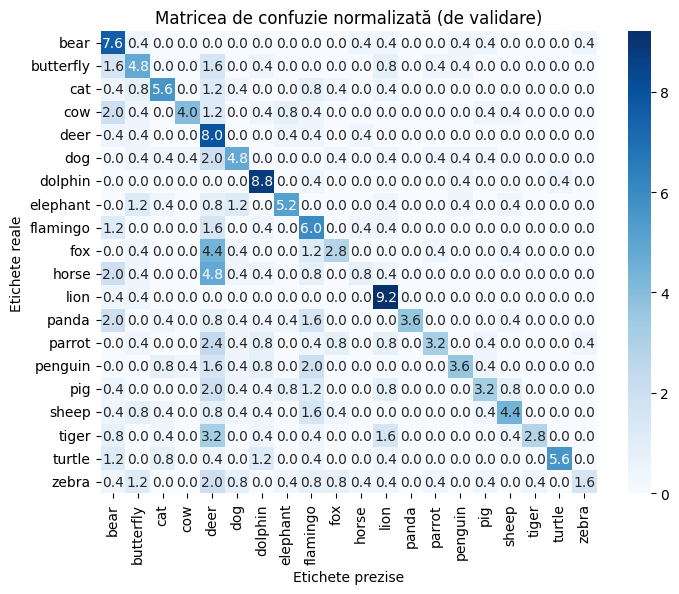
- Acuratețe: 47%.

A graph with numbers and symbols

Description automatically generated

2. Validare:

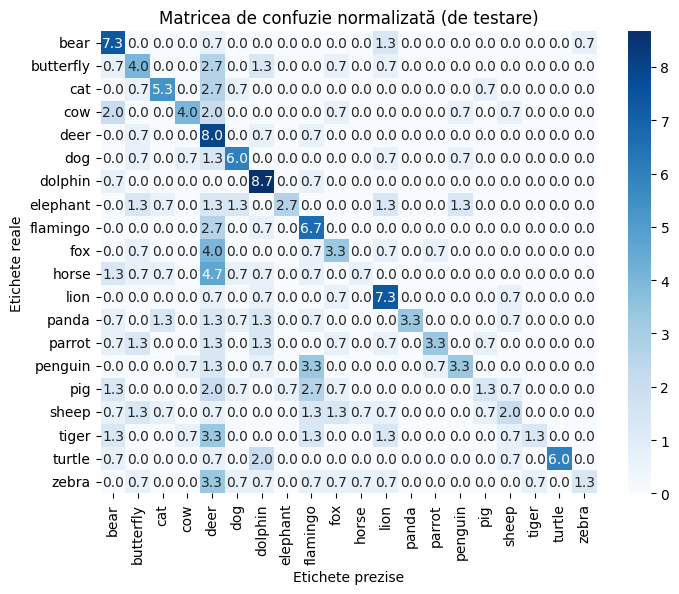
- Accuracitatea: 48%.



3. Testare:

- Din 5 imagini, KNN a clasificat corect 3 imagini (60%).

- Acuratețe : 43%.



Performanța este similară între seturile de antrenare, validare și testare, indicând că modelul nu supraînvață datele de antrenament, dar acuratețea generală este destul de scăzută pentru un model care ar trebui să clasifice 20 de clase.

**Precision:**

* Măsoară cât de multe din predicțiile pozitive ale unei clase sunt corecte. Clasele cu valori mici de precision (ex.: horse, pig, zebra, deer) indică faptul că modelul face multe predicții greșite pentru aceste clase. Precision-ul ridicat pentru clase precum panda și turtle indică faptul că modelul reușește să clasifice bine aceste categorii, deși pot fi mai ușor de distins.

**Recall:**

* Măsoară cât de bine identifică modelul toate instanțele reale ale unei clase. Clase precum deer, dolphin, și lion au valori ridicate ale recall-ului pe toate seturile, ceea ce înseamnă că modelul identifică majoritatea instanțelor reale ale acestora. În schimb, clase precum horse, tiger, și zebra sunt rar recunoscute corect.

**F1-Score:**

* F1-score scăzut (< 0.5) pentru multe clase (horse, pig, zebra) indică faptul că nici precision, nici recall nu sunt bune. Clasele care au F1-score ridicat (ex.: dolphin, lion, turtle) sunt cele unde modelul echilibrează bine precision și recall.

Setul de antrenare:

Acuratețea și f1-score-ul mediu pe setul de antrenare nu sunt foarte ridicate, indicând că modelul are dificultăți în învățarea caracteristicilor specifice claselor. Acest lucru este evident în cazul claselor cu suport mai mare (horse, pig, etc.).

Setul de validare:

Performanța generală este foarte similară cu cea din antrenare, ceea ce sugerează că modelul are o capacitate limitată de a generaliza.

Setul de testare:

Pe setul de testare, valorile scad ușor, ceea ce arată că modelul nu reușește să extrapoleze foarte bine la date noi. Clasele precum tiger, zebra și horse rămân problematice.

**Rezultate algoritm KNN:**

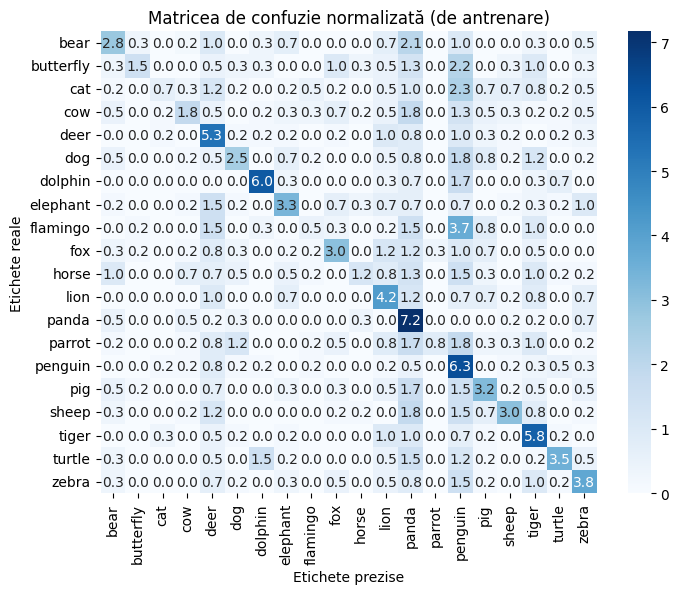
****

****

* **Naive Bayes**

1. Antrenare:

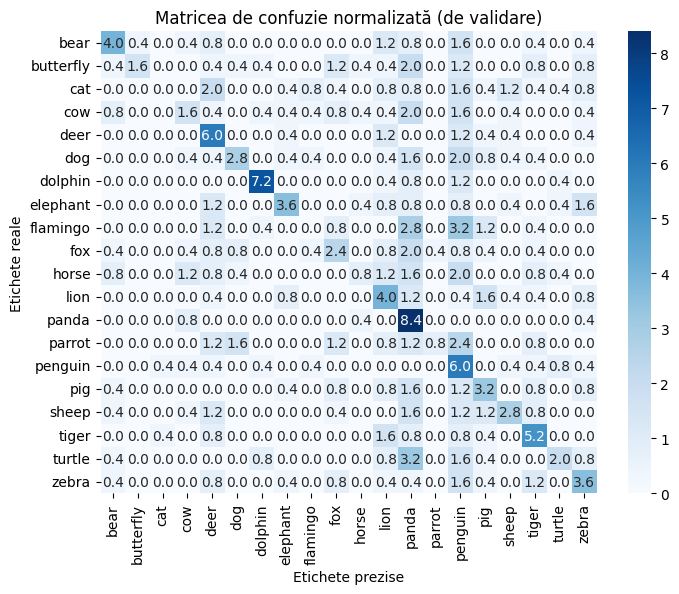
- Acuratețe: 33%.



2. Validare:

- Accuracitatea scade din cauza presupunerii de independență a caracteristicilor.

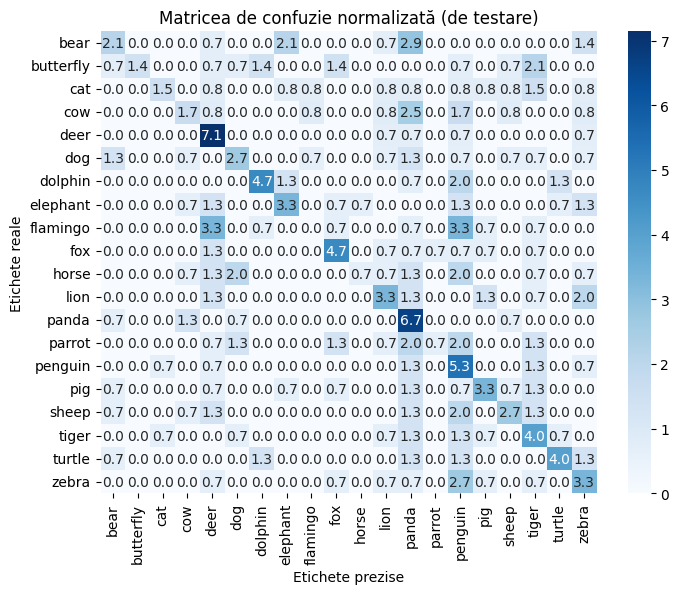
- Acuratețe: 33%.



3. Testare:

- Acuratețe: 32%.

- Din 5 imagini, Naive Bayes a clasificat corect o singură imagine (20%).



*Performanță pe setul de antrenare*

**Precizie și Recall:**

Algoritmul prezintă performanțe variabile pe clase, cu precizii și recall-uri care variază între valori foarte scăzute (ex.: flamingo - 0.23 precizie, 0.05 recall) și valori mai ridicate (ex.: dolphin - 0.67 precizie, 0.60 recall).

Clasele sunt dezechilibrate, unele având un suport mai redus de exemplare sau distribuții probabilistice dificil de clasificat corect de către Naive Bayes.

**F1-Score:**

F1-scores pentru majoritatea claselor sunt scăzute, indicând o performanță slabă în a balansa între precizie și recall. De exemplu:

Butterfly: F1 = 0.24

Tiger: F1 = 0.43 (relativ mai bună față de alte clase)

Valorile mai ridicate pentru clasele precum dolphin sugerează că modelul funcționează mai bine pentru unele distribuții, probabil din cauza unei separabilități mai bune între aceste clase și altele.

Accuratețe generală:

33%, indicând că modelul nu generalizează bine nici pe setul de antrenare.

Macro și Weighted Avg:

Media ponderată și macro sunt similare (F1 ≈ 0.31), ceea ce confirmă distribuția relativ uniformă a claselor.

Performanța pe setul de validare este consistentă cu setul de antrenare, sugerând că modelul are dificultăți de generalizare. Precizia și recall-ul macro sunt similare (0.38 și 0.33), iar F1 scade ușor la 0.30.

Clase problematice: Flamingo și cat nu sunt recunoscute deloc (precizie și recall 0.00), ceea ce indică fie un număr redus de exemple reprezentative în setul de date, fie o suprapunere puternică între distribuțiile lor și alte clase.

Performanțe notabile:

Dolphin: F1 = 0.73, consistentă pe toate seturile.

Panda: F1 = 0.39, cu un recall foarte mare (0.84).

**Rezultate algoritm Naive Bayes:**

A bear walking on a road

Description automatically generated



4. Concluzii

*Observații generale:*

- Algoritmul KNN a avut performanțe mai bune comparativ cu Naive Bayes datorită capacitații de a lua în considerare similaritatea caracteristicilor în mod direct.

- Naive Bayes a fost limitat de presupunerea independenței caracteristicilor, ceea ce este mai puțin realist pentru imagini.

- Analizând rezultatele obținute la Tema 1 cu algoritmul KMeans, am observat că dintre cei 3 algoritmi(KMeans, KNN și Naive Bayes), cel mai eficient și mai precis pentru setul meu de date a fost algoritmul KNN.

Limitări identificate:

- Preprocesarea imaginilor a simplificat problema, dar a pierdut informații importante despre texturi.

- Dimensiunea redusă a setului de testare poate afecta semnificativ estimarea performanței algoritmilor.

Cum pot îmbunătăți proiectul:

- Utilizarea unui set de date mai mare pentru antrenare.

- Crearea unei reprezentări mai avansate a caracteristicilor, de exemplu, utilizând histograme de culori sau descriptorii HOG.

Așadar, KNN este o alegere mai potrivită pentru problema dată, oferind o acuratețe semnificativ mai bună decât Naive Bayes. Totuși, complexitatea computațională a KNN trebuie gestionată în cazul unor seturi de date mai mari.

**Bibliografie**

1. <https://www.geeksforgeeks.org/naive-bayes-classifiers/>
2. <https://www.datacamp.com/tutorial/naive-bayes-scikit-learn>
3. <https://www.w3schools.com/python/python_ml_knn.asp>
4. <https://realpython.com/knn-python/>
5. <https://medium.com/@amirm.lavasani/classic-machine-learning-in-python-k-nearest-neighbors-knn-a06fbfaaf80a>