Raport Proiect: Clasificare imaginilor

-Snowboarders vs skiers-

Scopul si descrierea proiectului

Scopul proiectului este utilizarea algoritmilor **k-Nearest Neighbors (k-NN)** si **Naive Bayes** pentru a obține clasificarea imaginilor in snowboarderi si schiori. Acest lucru poate fi util in dezvoltarea unui sistem automatizat pentru recunoașterea acestora într-un eveniment sportiv, pentru a monitoriza raportul dintre schiori si snowboarderi de pe o pârtie pentru o soluție marketing, sau, in cazul meu, pentru a învăța si a înțelege cum funcționează acești algoritmi.

Modul de obținere si de organizare a datelor

Setul de date de antrenare l-am obținut inițial prin utilizarea unui script făcut in python pentru a descărca imagini de pe internet (un image scraper). Astfel am obținut aproximativ 400 de imagini pentru fiecare categorie (schiori si snowboarderi). Acest lucru l-am făcut pentru a scuti munca de a descărca manual fiecare poza de pe internet, însă după utilizarea scriptului, am verificat fiecare imagine si am făcut o triere a lor astfel încât ele sa fie relevante temei alese. Așadar setul de date de antrenare are 179 de poze pentru categoria „schiori” si 126 de date pentru categoria „snowboarderi”. Acest set de date de antrenare este organizat in subfoldere („schiori”, „snowboarderi”), necesare etichetărilor.

In scriptul făcut, împărțirea setului de date in set de date de antrenare, de validare si de testare este făcută din cod:

```
# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split( *arrays: X, y, test_size=0.2, random_state=42)
```

Parametrul „test_size=0.2” din funcția apelata „train_test_split” reprezintă faptul ca din setul de date, 20% este alocat pentru testare, iar restul este pentru antrenare si validare.

Algoritmii utilizați si parametrii acestora

Primul algoritm utilizat este **k-Nearest Neighbors (k-NN),** care este o metoda de învățare supervizata. Este un algoritm conceptual simplu, dar foarte puternic si, din aceste motive, este unul dintre cei mai populari algoritmi de învățare automata. Este utilizat atât pentru clasificare cat si pentru regresie. In cazul meu, acesta este utilizat pentru clasificarea imaginilor. Algoritmul k-NN este un algoritm de învățare leneș, deoarece nu creează un model in comparație cu alte metode de clasificare, ci prezice direct pe vaza datelor de antrenament. Pe scurt, in clasificare, un obiect este clasificat cu votul majorității vecinilor săi, obiectul fiind atribuit clasei cele mai frecvente din k-NN.

Parametrul utilizat este „k_neighbors=3”. Acesta se refera la numărul de vecini cei mai apropiați care sunt luați in considerare. Fiind atribuita valoarea 3, aceasta reprezintă faptul ca algoritmul ar trebui sa se uite la cei 3 vecini cei mai apropiați (bazat pe o metrica de distanta, de obicei distanta Euclidiana) fata de punctul de interogare dau de punctul testat. Valoarea „k_neighbors” **aleasa influențează flexibilitatea modelului:** o valoarea **mai mica** a lui „k_neighbors” poate duce la un model mai complex, potențial capturând zgomotul din datele de antrenament, ceea ce ar putea duce la supra adaptare. O valoare **mai mare** a lui „k_neighbors” ar putea duce la un model mai simplu, potențial trecând cu vederea nuanțele din datele de antrenament, ceea ce ar putea duce la sub adaptare.

Așadar, pentru setul meu de date am ales de cuviința ca valoarea lui „k_neighbors” sa fie egala cu 3

Al doilea algoritm utilizat este Naive Bayes, care este o metoda de învățare supervizata, precum si o tehnica de clasificare statistica, ce presupune existenta unui set de date deja clasificate (in cazul meu, acesta exista, imaginile fiind etichetate conform folderului din care fac parte), scopul fiind de a determina clasa in care se încadrează o noua instanța de același tip cu cele deja existenta, Metoda presupune ca instanțele din setul de date au o serie de valori ale acestor caracteristici sa aparțină claselor in care se încadrează setul de date. De asemenea, face parte dintr-o familie de algoritmi de învățare generativa adică

folosește cunoștințele despre distribuțiile de probabilitate anterioare care generează datele analizate, mai mult este capabil sa genereze noi date. Ceea ce face ca un clasificator Bayes sa fie naiv este presupunerea ca toate atributele unui punct luate in considerare sunt independente unele de altele ceea ce înseamnă ca prezenta sau absenta unei caracteristici nu afectează probabilitatea unei alte caracteristici. Acest algoritm este un clasificator probabilist, bazând-se pe teorema lui Bayes.

Biblioteci Python utilizate

- os: Pentru manipularea structurii de directoare și fișiere.
- matplotlib.pyplot: este folosită pentru a afișa și salva imagini și distribuții de probabilitate.
- - numpy: este utilizată pentru manipularea și procesarea datelor, în special pentru a stoca și manipula imagini.
- pandas: este utilizată pentru a crea și gestiona un DataFrame, care ulterior este utilizat pentru a salva rezultatele într-un fișier CSV.
- skimage.io.imread: funcție din biblioteca scikit-image pentru a citi imagini din fișiere.
- skimage.transform.resize: funcție din scikit-image utilizată pentru redimensionarea imaginilor.
- sklearn.model_selection.train_test_split: funcție din biblioteca scikit-learn este utilizată pentru a diviza setul de date în seturi de antrenament și de testare.
- sklearn.naive_bayes.GaussianNB: implementează algoritmul Naive Bayes pentru clasificarea pe baza distribuției gaussiene.
- sklearn.neighbors.KNeighborsClassifier: implementează algoritmul k-NN pentru clasificare.

Rezultate obținute

In urma rulării programului, am obținut 2 fișiere CSV in care am rezultatele celor 2 algoritmi: true label si predicted label:

True Label ski: pos	Predicted Label snow: neg		
snowboarderi	schiori	0	FN
schiori	schiori	1	TP
schiori	schiori	1	TP
snowboarderi	snowboarderi	1	TN
schiori	schiori	1	TP
schiori	schiori	1	TP
schiori	schiori	1	TP
schiori	schiori	1	TP
schiori	schiori	1	TP
schiori	schiori	1	TP
schiori	schiori	1	TP
schiori	schiori	1	TP
schiori	snowboarderi	0	FP
schiori	schiori	1	TP
schiori	schiori	1	TP
schiori	snowboarderi	0	FP
snowboarderi	snowboarderi	1	TN
snowboarderi	schiori	0	FN
schiori	schiori	1	TP
snowboarderi	snowboarderi	1	TN
snowboarderi	schiori	0	FN
snowboarderi	schiori	0	FN
schiori	schiori	1	TP
schiori	schiori	1	TP
snowboarderi	schiori	0	FN
snowboarderi	schiori	0	FN
schiori	snowboarderi	0	FP
snowboarderi	schiori	0	FN
schiori	snowboarderi	0	FP
schiori	schiori	1	TP
snowboarderi	snowboarderi	1	TN
snowboarderi	schiori	0	FN
schiori	schiori	1	TP
schiori	schiori	1	TP
snowboarderi	schiori	0	FN
schiori	schiori	1	TP
schiori	schiori	1	TP
schiori	snowboarderi	0	FP
schiori	schiori	1	TP
schiori	schiori	1	TP
schiori	schiori	1	TP
snowboarderi	schiori	0	FN
schiori	snowboarderi	0	FP
schiori	schiori	1	TP
snowboarderi	snowboarderi	1	TN
snowboarderi	schiori	0	FN
schiori	snowboarderi	0	FP
snowboarderi	schiori	0	FN
snowboarderi	schiori	0	FN
snowboarderi	schiori	0	FN
snowboarderi	schiori	0	FN
schiori	schiori	1	TP
schiori	schiori	1	TP
snowboarderi	snowboarderi	1	TN
snowboarderi	schiori	0	FN
snowboarderi	schiori	0	FN
schiori	schiori	1	TP
snowboarderi	schiori	0	FN
snowboarderi	snowboarderi	1	TN
snowboarderi	snowboarderi	1	TN

Totoal:	36	
	TP:	28
	TN:	8
	FP:	7
	FN:	18
	Error rate:	0.409836066
	Precision:	0.8
	Accuracy:	0.590163934
	Sensibility:	0.608695652

Acestea sunt rezultatele obținute pentru algoritmul k-NN. Folosindu-mă de niște formule in Excel am identificat termenii matricei de confuzie [TP, FP; FN, TN] după cum urmează:

TP = 28; (însemnând ca am 28 de valori interpretate de model ca fiind „schiori” iar valoarea reala sa fie tot „schiori”)

TN = 8; (însemnând ca am 8 de valori interpretate de model ca fiind „snowboarderi” iar valoarea reala sa fie tot „snowboarderi”)

FP = 7; (însemnând ca am 7 de valori interpretate de model ca fiind „schiori” iar valoarea reala sa fie „snowboarderi”)

FN = 18; (însemnând ca am 18 de valori interpretate de model ca fiind „snowboarderi” iar valoarea reala sa fie „schiori”)

Apoi am identificat următoarele:

Rata de eroare = 0.41 (după formula: (FP + FN) / (TP + TN + FP + FN))

Precizia = 0.8 (după formula: TP / (TP + FP)). Aceasta este o măsură a corectitudinii care se realizează in predicția adevărata. Cu alte cuvinte îmi spune cate predicții sunt de fapt pozitive din totalul prezis pozitiv.

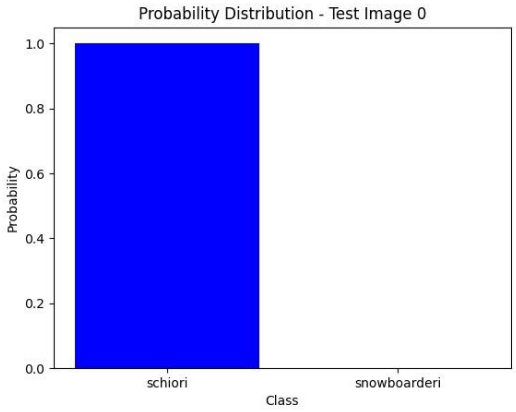
Acuratețea = 0.6 (după formula: (TP + TN) / (TP + TN + FP + FN)). Aceasta măsoară proporția de instanțe clasificate corect din numărul total de imagini. Este raportul dintre numărul de predicții corecte si numărul total de predicții

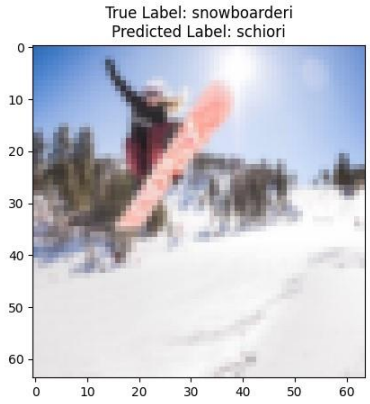
Sensibilitatea = 0.61 (după formula: TP / (TP + FN)).

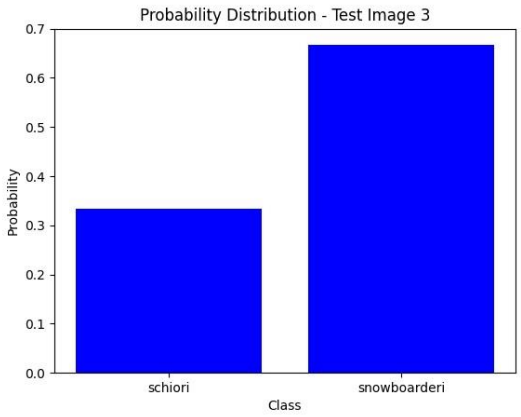
Observ ca rata de eroare este ridicata (0.4), ceea ce înseamnă ca modelul nu este antrenant îndeajuns. Aceeași concluzie o pot trage si din faptul ca acuratețea este foarte mica pentru un model (0.6), având in vedere ca probabilitatea sa aleagă o clasa random este de 0.5. Consider ca acest lucru se datorează setului de date destul de micuț, dar si din cauza greutății clasificării. Imaginile sunt foarte asemănătoare intre ele; majoritatea imaginilor au același fundal si același personaj, in poziții foarte asemănătoare. Diferența o face obiectul in sine (placa de snowboard sau schiurile), care este greu de diferențiat, sau in unele cazuri, poziția

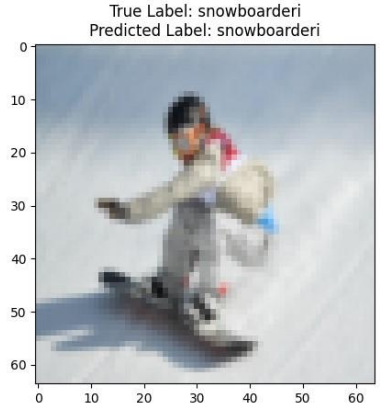
omului pe placa sau schiuri. Pentru aceasta îmi asum vina, pentru ca am ales o tema greu de implementat, însă am învățat din aceasta.

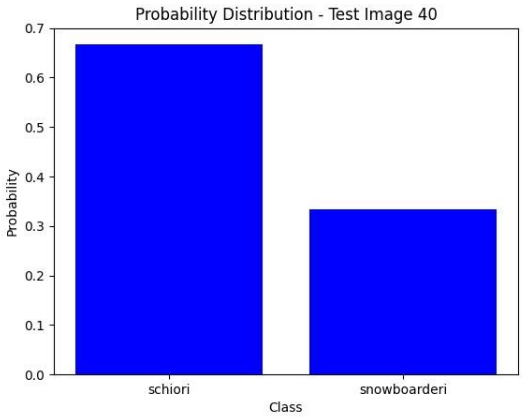
Câteva poze specifice, in urma rulării programului, pentru algoritmul k-NN:

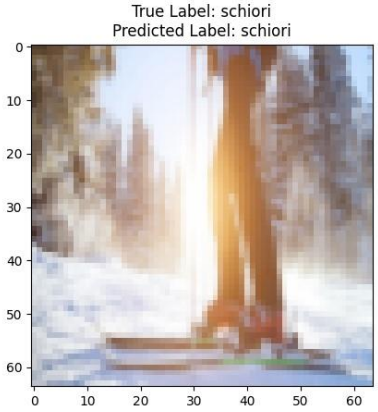


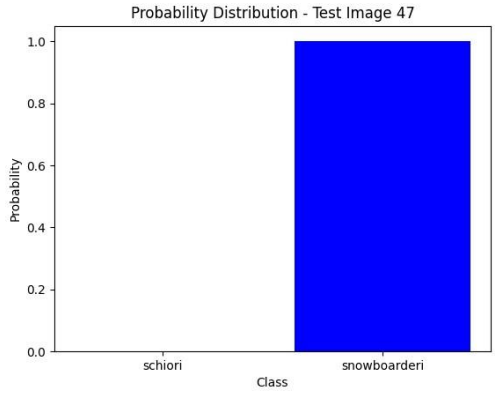


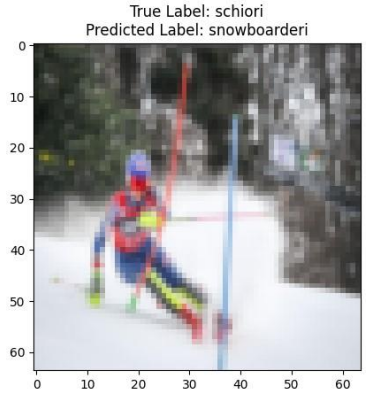


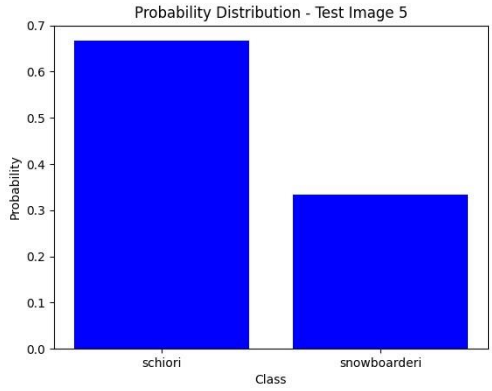


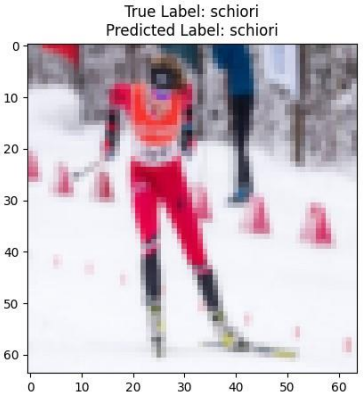


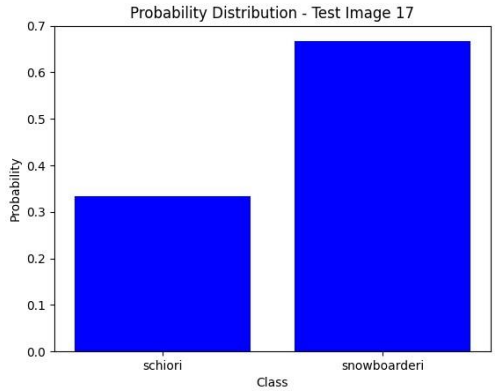


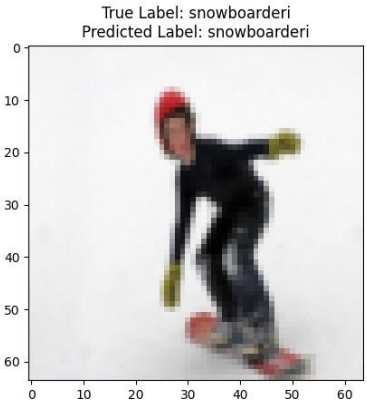

Probability Distribution - Test Image 47
Probability
Class
schiori
snowboarderi
0.0
0.2
0.4
0.6
0.8
1.0


True Label: schiori
Predicted Label: snowboarderi


Probability Distribution - Test Image 5
Probability
Class
schiori
snowboarderi
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7


True Label: schiori
Predicted Label: schiori


Probability Distribution - Test Image 17
Probability
Class
schiori
snowboarderi
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7


True Label: snowboarderi
Predicted Label: snowboarderi

True Label snowb = neg	Predicted Label		
snowboarderi	schiori	0	FN
schiori	schiori	1	TP
schiori	schiori	1	TP
snowboarderi	schiori	0	FN
schiori	schiori	1	TP
schiori	schiori	1	TP
schiori	schiori	1	TP
schiori	schiori	1	TP
schiori	schiori	1	TP
schiori	schiori	1	TP
schiori	schiori	1	TP
schiori	schiori	1	TP
schiori	schiori	1	TP
schiori	schiori	1	TP
schiori	snowboarderi	0	FP
schiori	schiori	1	TP
schiori	schiori	1	TP
snowboarderi	snowboarderi	1	TN
snowboarderi	snowboarderi	1	TN
schiori	schiori	1	TP
snowboarderi	snowboarderi	1	TN
snowboarderi	schiori	0	FN
snowboarderi	schiori	0	FN
schiori	schiori	1	TP
schiori	schiori	1	TP
snowboarderi	snowboarderi	1	TN
snowboarderi	schiori	0	FN
schiori	schiori	1	TP
snowboarderi	schiori	0	FN
schiori	schiori	1	TP
schiori	snowboarderi	0	FP
snowboarderi	schiori	0	FN
snowboarderi	snowboarderi	1	TN
schiori	schiori	1	TP
schiori	schiori	1	TP
snowboarderi	schiori	0	FN
schiori	schiori	1	TP
schiori	schiori	1	TP
schiori	schiori	1	TP
schiori	schiori	1	TP
schiori	schiori	1	TP
schiori	schiori	1	TP
snowboarderi	snowboarderi	1	TN
schiori	snowboarderi	0	FP
schiori	schiori	1	TP
snowboarderi	snowboarderi	1	TN
snowboarderi	schiori	0	FN
schiori	schiori	1	TP
snowboarderi	snowboarderi	1	TN
snowboarderi	schiori	0	FN
snowboarderi	schiori	0	FN
snowboarderi	snowboarderi	1	TN
schiori	schiori	1	TP
schiori	snowboarderi	0	FP
snowboarderi	schiori	0	FN
snowboarderi	schiori	0	FN
snowboarderi	schiori	0	FN
schiori	schiori	1	TP
snowboarderi	schiori	0	FN
snowboarderi	snowboarderi	1	TN
snowboarderi	schiori	0	FN

Total	41	
	TP:	31
	TN:	10
	FP:	4
	FN:	16
	Error rat	0.3279
	Precisio	0.8857
	Accurac	0.6721
	Sensibili	0.6596

Acestea sunt rezultatele obținute pentru algoritmul Naive Bayes. Folsindu-ma de niște formule in Excel am identificat termenii matricei de confuzie [TP, FP; FN, TN] după cum urmează:

TP = 31; (însemnând ca am 31 de valori interpretate de model ca fiind „schiori” iar valoarea reala sa fie tot „schiori”)

TN = 10; (însemnând ca am 10 de valori interpretate de model ca fiind „snowboarderi” iar valoarea reala sa fie tot „snowboarderi”)

FP = 4; (însemnând ca am 4 de valori interpretate de model ca fiind „schiori” iar valoarea reala sa fie „snowboarderi”)

FN = 16; (însemnând ca am 16 de valori interpretate de model ca fiind „snowboarderi” iar valoarea reala sa fie „schiori”)

Apoi am identificat următoarele:

Rata de eroare = 0.33 (după formula: (FP + FN) / (TP + TN + FP + FN))

Precizia = 0.89 (după formula: TP / (TP + FP)). Aceasta este o măsură a corectitudinii care se realizează in predicția adevărată. Cu alte cuvinte îmi spune cate predicții sunt de fapt pozitive din totalul prezis pozitiv.

Acuratețea = 0.67 (după formula: (TP + TN) / (TP + TN + FP + FN)). Aceasta măsoară proporția de instanțe clasificate corect din numărul total de imagini. Este raportul dintre numărul de predicții corecte si numărul total de predicții

Sensibilitatea = 0.65 (dupa formula: TP / (TP + FN)).

In cazul acestui model, rata de eroare este puțin mai mica (0.33), iar acuratețea a crescut puțin (0.67). Aceasta îmi da de gândit faptul ca acest algoritm este mai bun pentru tema mea, însă rezultatele tot lasă de dorit, din cauza acelorași motive pe care le-am prezentat la interpretarea rezultatelor obținute la algoritmul anterior.

Câteva poze specifice, in urma rulării programului, pentru algoritmul Naive Bayes: