

Analiză de sentiment folosind clasificatori bayesieni naivi

Secțiune Teoretică



Student:

Nozohor Aida

Coordonatori:

Asistent dr. ing. Trandafir-Liviu Serghei

Profesor dr. ing. Ichim Loretta

Cuprins

1. Alegerea metodologiei de lucru	3
2. Caracteristicile setului de date selectat	4
3. Bibliotecile Python utilizate	5
4. Etapele de preprocesare și normalizare a datelor	6
5. Dezvoltarea modelului	8
6. Procesul de antrenare al modelului	9
7. Procesul de testare al modelului	10
8. Rezultate	11

1. Alegerea metodologiei de lucru

În învățarea automată, clasificatorii bayesieni naivi reprezintă o familie de „clasificatori probabilistici” simpli, bazați pe aplicarea teoremei lui Bayes cu ipoteze puternice (naive) de independență între descriptori.

Naive Bayes este o tehnică simplă pentru construirea clasificatorilor: modele care atribuie etichete de clasă pentru instanțe noi, reprezentate ca vectori de valori pentru diverse caracteristici, unde etichetele de clasă fac parte dintr-o mulțime finită. Nu există un singur algoritm pentru antrenarea acestui tip de clasificatori, ci o familie de algoritmi bazați pe un principiu comun: toți clasificatorii bayesieni naivi presupun că valoarea unei anumite caracteristici este independentă de valoarea oricărei altă caracteristici, dată fiind variabila de clasă. ¹

De exemplu, un fruct poate fi considerat a fi un măr dacă este roșu, rotund, și de aproximativ 10 cm în diametru. Un clasificator naiv bayesian consideră că fiecare dintre aceste caracteristici contribuie în mod independent la probabilitatea ca acest fruct să fie un măr, indiferent de eventualele corelații între culoare, rotunjime și diametru.

Am evaluat această metodă drept cea mai adecvată pentru implementarea proiectului meu, având în vedere următoarele considerente:





1. Naive Bayes are o eficiență computațională bună și poate gestiona seturi de date mari, ceea ce este adesea întâlnit în cazul recenziilor de filme, unde există o cantitate mare de date textuale.
2. Această metodă se potrivește bine cu datele textuale datorită faptului că se bazează pe frecvența cuvintelor pentru clasificare.
3. Este ușor de implementat și scalabil, ceea ce face posibilă folosirea sa pe seturi de date variate și extinse.
4. Recenziile de filme pot fi adesea împărțite în două categorii, pozitive și negative. Naive Bayes are o performanță bună în clasificarea binară, astfel încât poate furniza rezultate satisfăcătoare pentru aceste scenarii.

Am luat în considerare și posibilitatea utilizării algoritmului Support Vector Machines. Astfel, am elaborat un tabel de comparație între cei doi algoritmi pentru a face cea mai bună alegere între tehnicile de implementare disponibile.

Naive Bayes	Support Vector Machines
Avantaje	
Eficient în timpul antrenării, de obicei, datorită presupunerii naive de independență între caracteristici.	Poate oferi performanțe bune în probleme complexe de clasificare.
Se descurcă bine cu seturi de date mari și este ușor de implementat și interpretat.	Poate trata eficient seturi de date mici sau seturi de date cu caracteristici complexe.
Dezavantaje	
Face presupuneri simplificate despre date și poate avea performanțe mai slabe în capturarea relațiilor complexe în text.	Poate necesita timp mai îndelungat pentru antrenare, în special pe seturi de date mari.

Link catre sursa: https://en.wikipedia.org/wiki/Naive_Bayes_classifier [1]

2. Caracteristicile setului de date selectat

 review 	 sentiment 
49582 unique values	2 unique values
One of the other reviewers has mentioned that after watching just 1 Oz episode you'll be hooked. The...	positive
A wonderful little production. The filming technique is very unassuming- very old-time-B...	positive
I thought this was a wonderful way to spend time on a too hot summer weekend, sitting in the air con...	positive
Basically there's a family where a little boy (Jake) thinks there's a	negative

Setul de date IMDB conține 50.000 de recenzii de filme destinate procesării limbajului natural sau analizei de text. Acesta este un set de date pentru clasificare binară a sentimentelor ce conține 25.000 de recenzii de filme puternic polarizate pentru antrenare și 25.000 pentru testare.²

Link catre sursa: <https://www.kaggle.com/datasets/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews/data> [2]

3. Bibliotecile Python utilizate

```
Clasificare_Recenzii.py x
1 import numpy as np
2 import pandas as pd
3 import re
4 from nltk.corpus import stopwords
5 from nltk.tokenize import word_tokenize
6 from nltk.stem import SnowballStemmer
7 from sklearn.feature_extraction.text import CountVectorizer
8 from sklearn.model_selection import train_test_split
9 from sklearn.naive_bayes import GaussianNB, MultinomialNB, BernoulliNB
10 from sklearn.metrics import accuracy_score
11 import pickle
```

import numpy as np	Importă biblioteca NumPy pentru manipularea eficientă a datelor în formă de matrice sau vectori.
import pandas as pd	Importă biblioteca Pandas pentru lucrul cu structuri de date tabulare, precum DataFrame.
import re	Importă modulul de expresii regulate (regular expressions) pentru manipularea șabloanelor de text.
from nltk.corpus import stopwords	Importă lista de cuvinte de oprire (stop words) din biblioteca Natural Language Toolkit (NLTK).
from nltk.tokenize import word_tokenize	Importă funcția de tokenizare a cuvintelor din NLTK pentru descompunerea textului în cuvinte.
from nltk.stem import SnowballStemmer	Importă algoritmul SnowballStemmer pentru efectuarea operațiunii de stemming (eliminarea sufixe).
from sklearn.feature_extraction.text import CountVectorizer	Importă CountVectorizer din scikit-learn pentru crearea unui model Bag of Words.
from sklearn.model_selection import train_test_split	Importă funcția train_test_split din scikit-learn pentru divizarea setului de date în seturi de antrenare și testare.
from sklearn.naive_bayes import GaussianNB, MultinomialNB, BernoulliNB	Importă clasificatorii bayesieni naivi cu distribuții specifice.
from sklearn.metrics import accuracy_score	Importă funcția accuracy_score din scikit-learn pentru evaluarea acurateții unui model.
import pickle	Importă modulul pickle pentru serializarea și deserializarea obiectelor Python.

4. Etapele de preprocesare și normalizare a datelor

I. Înlăturarea tag-urilor HTML

Această etapă implică curățarea textului din recenzii prin eliminarea tuturor tagurilor HTML. Funcția `clean` primește un text și utilizează expresii regulate pentru a identifica și înlocui orice șiruri care încep cu `<` și se încheie cu `>`, adică tagurile HTML. Astfel, aceste taguri sunt eliminate din text, iar rezultatul curățat este stocat în coloana `review` a setului de date `data`.

Linia `data.review = data.review.apply(clean)` aplică această funcție de curățare asupra întregii coloane de recenzii din setul de date. `data.review[0]` afișează rezultatul curățat al primei recenzii din setul de date.

```
37 def clean(text):
38     cleaned = re.compile(r'<.*?>')
39     return re.sub(cleaned, repl: '', text)
40
41 data.review = data.review.apply(clean)
42 print()
43 print("1. Datele curatate de taguri HTML:")
44 print(data.review[0])
```

II. Înlăturarea caracterelor speciale

Funcția `is_special` primește un text și elimină toate caracterele speciale și non-alfanumerice, păstrând doar literele și cifrele. Această etapă este importantă în preprocesarea datelor text pentru a reduce zgomotul și a standardiza formatul, facilitând astfel analiza ulterioară.

```
47 def is_special(text):
48     rem = ''
49     for i in text:
50         if i.isalnum():
51             rem = rem + i
52         else:
53             rem = rem + ' '
54     return rem
55
56 data.review = data.review.apply(is_special)
57 print()
58 print("2. Datele curatate de caractere speciale:")
59 print(data.review[0])
```

III. Convertirea tuturor caracterelor în litere mici

Prin funcția `to_lower`, se transformă toate caracterele majuscule în litere mici, asigurând uniformitatea și coerența în manipularea ulterioară a informațiilor.

```
62 def to_lower(text):
63     return text.lower()
64
65 data.review = data.review.apply(to_lower)
66 print()
67 print("3. Datele convertite in lowercase:")
68 print(data.review[0])
```

IV. Eliminarea cuvintelor de oprire

Prin intermediul funcției `rem_stopwords`, se elimină cuvintele de oprire (stop words) din text, adică cuvintele frecvente care nu aduc o valoare semnificativă în analiza sentimentului. Eliminarea acestor cuvinte permite concentrarea pe cuvintele cheie, îmbunătățind precizia și relevanța analizei sentimentului din recenzii.

```
71 def rem_stopwords(text):
72     stop_words = set(stopwords.words('english'))
73     words = word_tokenize(text)
74     return [w for w in words if w not in stop_words]
75
76 data.review = data.review.apply(rem_stopwords)
77 print()
78 print("4. Datele fara cuvinte de oprire:")
79 print(data.review[0])
```

V. Eliminarea sufixelor și a altor componente ce pot varia (stemming)

Stemming este un proces de reducere a cuvintelor la forma lor de rădăcină sau la o formă bază comună. Scopul este de a grupa cuvintele care au aceeași origine sau rădăcină lingvistică, astfel încât să poată fi tratate uniform în procesarea limbajului natural. De exemplu, aplicarea stemmingului pe cuvintele "alergare", "alerga", "alergat" ar duce la forma de rădăcină "alerg".

```
82 def stem_txt(text):
83     ss = SnowballStemmer('english')
84     return " ".join([ss.stem(w) for w in text])
85
86 data.review = data.review.apply(stem_txt)
87 print()
88 print("5. Datele fara sufixe:")
89 print(data.review[0])
90
91 print()
92 print(data.head())
```

5. Dezvoltarea modelului

I. Crearea unui Bag Of Words

Această etapă se ocupă de transformarea recenziilor (textelor) într-o reprezentare numerică utilizând o tehnică numită "Bag of Words" (sac de cuvinte). Fiecare recenzie este reprezentată printr-un vector de lungime fixă, în care fiecare element corespunde frecvenței de apariție a unui cuvânt specific din vocabularul întregului set de recenzii. `X` reprezintă matricea de caracteristici rezultată, iar `y` reprezintă vectorul de etichete sentimentale asociate fiecărei recenzii. `CountVectorizer` convertește textul într-o matrice de frecvențe de cuvinte, iar `max_features=1000` specifică că se vor utiliza doar primele 1000 de cuvinte cele mai frecvente din vocabular. Afișarea formei (`shape`) arată dimensiunile matricii rezultate.

```
97 X = np.array(data.iloc[:, 0].values)
98 y = np.array(data.sentiment.values)
99 cv = CountVectorizer(max_features=1000)
100 X = cv.fit_transform(data.review).toarray()
101 print()
102 print("X.shape = ", X.shape)
103 print("y.shape = ", y.shape)
```

II. Împărțirea datelor în seturi de antrenare, validare și testare

Această porțiune de cod împarte datele inițiale în trei subseturi: un set de antrenare (train), un set de validare (val) și un set de testare (test). `train_test_split` este folosită de două ori pentru a realiza împărțirea. În prima apelare, datele sunt împărțite inițial în seturile de antrenare și testare, iar apoi setul de antrenare este împărțit din nou în seturile de antrenare și validare.

```
113 trainx, testx, trainy, testy = train_test_split(*arrays: X, y, test_size=0.2, random_state=9)
114 trainx, valx, trainy, valy = train_test_split(*arrays: trainx, trainy, test_size=0.25, random_state=9)
115
116 print()
117 print("Train shapes : X = {}, y = {}".format(*args: trainx.shape, trainy.shape))
118 print("Validation shapes : X = {}, y = {}".format(*args: valx.shape, valy.shape))
119 print("Test shapes : X = {}, y = {}".format(*args: testx.shape, testy.shape))
```


6. Procesul de antrenare al modelului

I. Definirea și antrenarea modelelor

Se definesc trei modele diferite de clasificatori bayesieni naivi: `GaussianNB`, `MultinomialNB`, și `BernoulliNB`. Fiecare dintre aceste modele este apoi antrenat pe setul de date de antrenare (`trainx` și `trainy`) pentru a învăța relațiile între caracteristici și etichetele corespunzătoare.

```
126 gnb, mnb, bnb = GaussianNB(), MultinomialNB(alpha=1.0, fit_prior=True), BernoulliNB(alpha=1.0, fit_prior=True)
127 gnb.fit(trainx, trainy)
128 mnb.fit(trainx, trainy)
129 bnb.fit(trainx, trainy)
```

II. Predicții și acuratețe pentru a alege cel mai potrivit clasificator pe baza setului de validare

Codul de mai jos efectuează predicții folosind modelele antrenate (gnb, mnb, bnb) pe setul de validare (valx), apoi calculează acuratețea acestora prin comparație cu etichetele reale din setul de validare (valy). Rezultatele sunt afișate sub forma acuratetii pentru fiecare model (Gaussian, Multinomial, Bernoulli) în etapa de validare.

```
133 ypg_val = gnb.predict(valx)
134 ypm_val = mnb.predict(valx)
135 ypb_val = bnb.predict(valx)
136 print()
137 # accuracy_score compara rezultatele predicțiilor cu etichetele reale din setul de validare
138 print("Acuratete validare - Gaussian = ", accuracy_score(valy, ypg_val))
139 print("Acuratete validare - Multinomial = ", accuracy_score(valy, ypm_val))
140 print("Acuratete validare - Bernoulli = ", accuracy_score(valy, ypb_val))
```

III. Predicții și acuratețe pentru a alege cel mai potrivit clasificator pe baza setului de testare

Codul de mai jos efectuează predicții folosind modelele antrenate (gnb, mnb, bnb) pe setul de testare (testx), apoi calculează acuratețea acestora prin comparație cu etichetele reale din setul de testare (testy). Rezultatele sunt afișate sub forma acuratetii pentru fiecare model (Gaussian, Multinomial, Bernoulli) în etapa de testare.

```
143 ypg_test = gnb.predict(testx)
144 ypm_test = mnb.predict(testx)
145 ypb_test = bnb.predict(testx)
146 print()
147 # accuracy_score compara rezultatele predicțiilor cu etichetele reale din setul de testare
148 print("Acuratete testare - Gaussian = ", accuracy_score(testy, ypg_test))
149 print("Acuratete testare - Multinomial = ", accuracy_score(testy, ypm_test))
150 print("Acuratete testare - Bernoulli = ", accuracy_score(testy, ypb_test))
```

IV. Salvarea Modelului și a Resurselor Asociate

Această porțiune de cod salvează modelul antrenat (bnb) într-un fișier utilizând modul de serializare pickle. Astfel, modelul poate fi păstrat și folosit ulterior fără a necesita reantrenarea. De asemenea, vocabularul Bag of Words (word_dict) obținut prin aplicarea vectorizatorului CountVectorizer (cv) este și el salvat într-un fișier pentru a fi utilizat în viitor. Această practică de salvare a modelelor și a resurselor asociate (cum ar fi vocabularul) este utilă pentru a evita necesitatea repetării procesului de antrenare și vectorizare.

```
152 # Salvez modelul antrenat (bnb) într-un fișier utilizând modul de serializare pickle
153 pickle.dump(bnb, open('model1.pkl', 'wb'))
154
155 # Salvare vocabular Bag Of Words
156 word_dict = cv.vocabulary_
157 pickle.dump(word_dict, open('bow.pkl', 'wb'))
```

7. Procesul de testare al modelului

Generăm o nouă recenzie în vederea testării modelului:

```
160 rev = ""Terrible. Complete trash. Brainless tripe. Insulting to anyone who isn't an 8 year old fan boy. Im actually
161 pretty disgusted that this movie is making the money it is - what does it say about the people who brainlessly
162 hand over the hard earned cash to be 'entertained' in this fashion and then come here to leave a positive 8.8 review??
163 Oh yes, they are morons. Its the only sensible conclusion to draw. How anyone can rate this movie amongst the pantheon
164 of great titles is beyond me. So trying to find something constructive to say about this title is hard...I enjoyed Iron Man?
165 Tony Stark is an inspirational character in his own movies but here he is a pale shadow of that...About the only 'hook'
166 this movie had into me was wondering when and if Iron Man would knock Captain America out...Oh how I wished he had :(
167 What were these other characters anyways? Useless, bickering idiots who really couldn't organise happy times in a brewery.
168 The film was a chaotic mish mash of action elements and failed 'set pieces'... I found the villain to be quite amusing.
169 And now I give up. This movie is not robbing any more of my time but I felt I ought to contribute to restoring the obvious
170 fake rating and reviews this movie has been getting on IMDb.""
```

Se parcurg etapele de prelucrare a datelor:

```
171 f1 = clean(rev)
172 f2 = is_special(f1)
173 f3 = to_lower(f2)
174 f4 = rem_stopwords(f3)
175 f5 = stem_txt(f4)
```

Fiecare cuvânt unic din recenzie este asociat cu o valoare numerică în vectorul BOW, reprezentând de câte ori apare acel cuvânt în recenzie:

```
177 bow, words = [], word_tokenize(f5)
178 for word in words:
179     bow.append(words.count(word))
```

Fiecare element al vectorului `inp` corespunde frecvenței de apariție a unui cuvânt în recenzia prelucrată (f5). Dicționarul `word_dict` conține cuvintele unice din vocabular și leagă fiecare cuvânt de o anumită poziție în vectorul BOW. Astfel, pentru fiecare cuvânt din `word_dict`, se adaugă în vectorul `inp` de câte ori apare acel cuvânt în recenzia prelucrată (f5). Acest vector `inp` este apoi folosit pentru a face o predicție cu modelul antrenat.

```
181 inp = []
182 for i in word_dict:
183     inp.append(f5.count(i[0]))
```

Se realizează o predicție asupra sentimentului recenziei folosind modelul antrenat (bnb). Vectorul `inp` este transformat într-un array NumPy și apoi remodelat pentru a corespunde formei așteptate de model (1, 1000) în cazul Bag of Words cu 1000 de caracteristici. Apoi, predicția este realizată cu `bnb.predict`, iar rezultatul (y_pred) este verificat pentru a determina dacă recenzia este clasificată ca pozitivă sau negativă.

```
185 y_pred = bnb.predict(np.array(inp).reshape(1, 1000))
186
187 print()
188 if y_pred == 0:
189     print("Recenzia este clasificata ca negativa.")
190 else:
191     print("Recenzia este clasificata ca pozitiva.")
```

8. Rezultate

În urma antrenării:

Rezultatele de mai jos reflectă performanța modelelor pe seturile de validare și de testare. Forma "Train shapes : X = (30000, 1000), y = (30000,)" indică că setul de antrenare conține 30,000 de exemple, fiecare cu 1,000 de caracteristici. Similar, formele seturilor de validare și de testare sunt indicate. Valorile de acuratețe furnizează măsura cât de bine modelele generalizează la datele pe care nu le-au văzut în timpul antrenării. Cu cât acuratețea este mai mare, cu atât modelul are performanțe mai bune. În acest caz, modelul Bernoulli Naive Bayes pare să aibă cea mai bună acuratețe pe setul de testare.

```
Train shapes : X = (30000, 1000), y = (30000,)
Validation shapes : X = (10000, 1000), y = (10000,)
Test shapes : X = (10000, 1000), y = (10000,)

Acuratete validare - Gaussian = 0.7902
Acuratete validare - Multinomial = 0.8339
Acuratete validare - Bernoulli = 0.8398

Acuratete testare - Gaussian = 0.7846
Acuratete testare - Multinomial = 0.8316
Acuratete testare - Bernoulli = 0.8387
```

În urma testării:

Rezultatul, din urma utilizării modelului Bernoulli Naive Bayes pe recenzia furnizată de noi, indică faptul că aceasta a fost clasificată ca fiind negativă de către modelul antrenat, rezultatul fiind în concordanță cu conținutul critic și cuvintele utilizate în recenzie.

```
Recenzia este clasificata ca negativa.
```

1. Grafice de bare

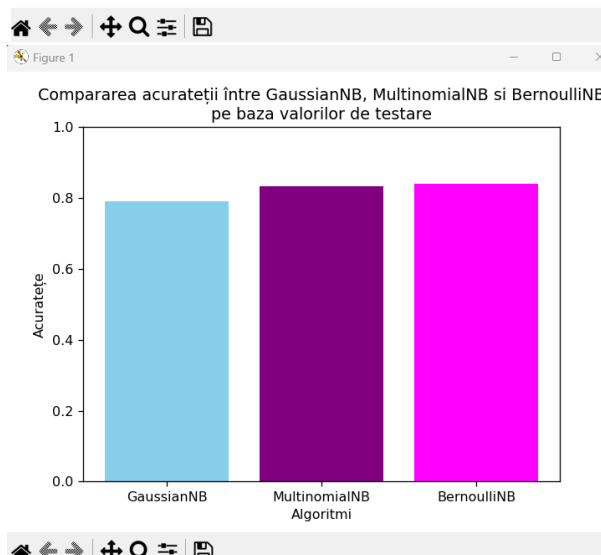
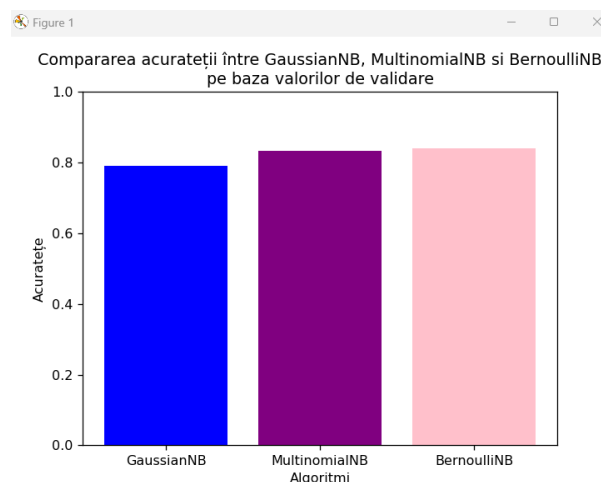
Cu scopul de a evidenția mai clar disparitățile dintre performanțele fiecărui algoritm, am elaborat două grafice de bare, folosind biblioteca `matplotlib`. Acestea furnizează o comparație vizuală a acurateței pentru fiecare model, atât pe baza datelor de validare, cât și pe cele de testare.

```

147 accuracy_val_scores_test = [
148     accuracy_score(valy, ypg_val),
149     accuracy_score(valy, ypm_val),
150     accuracy_score(valy, ypb_val)
151 ]
152
153 plt.bar(models, accuracy_val_scores_test, color=['blue', 'purple', 'pink'])
154 plt.xlabel('Algoritmi')
155 plt.ylabel('Acuratețe')
156 plt.title('Compararea acurateții între GaussianNB, MultinomialNB si BernoulliNB\npe baza valorilor de validare')
157 plt.ylim(0, 1)
158 plt.show()

170 accuracy_test_scores_test = [
171     accuracy_score(testy, ypg_test),
172     accuracy_score(testy, ypm_test),
173     accuracy_score(testy, ypb_test)
174 ]
175
176 plt.bar(models, accuracy_val_scores_test, color=['skyblue', 'purple', 'magenta'])
177 plt.xlabel('Algoritmi')
178 plt.ylabel('Acuratețe')
179 plt.title('Compararea acurateții între GaussianNB, MultinomialNB si BernoulliNB\npe baza valorilor de testare')
180 plt.ylim(0, 1)
181 plt.show()

```



2. Introducerea unui nou set de recenzii

Am definit o listă numită `reviews`, care conține zece elemente, fiecare reprezentând o recenzie pentru un film.

Aceste recenzii sunt împărțite în două categorii: primele cinci recenzii sunt negative, exprimând dezamăgirea sau nemulțumirea față de film, în timp ce ultimele cinci recenzii sunt pozitive, evidențiind încântarea și aprecierea pentru calitatea filmului.

```
193 reviews = [  
194     """Terrible. Complete trash. Brainless tripe. Insulting to anyone who isn't an 8 year old fan boy. Im actually  
195     pretty disgusted that this movie is making the money it is - what does it say about the people who brainlessly  
196     hand over the hard earned cash to be 'entertained' in this fashion and then come here to leave a positive 8.8 review??  
197     Oh yes, they are morons. Its the only sensible conclusion to draw. How anyone can rate this movie amongst the pantheon  
198     of great titles is beyond me. So trying to find something constructive to say about this title is hard...I enjoyed Iron Man?  
199     Tony Stark is an inspirational character in his own movies but here he is a pale shadow of that...About the only 'hook'  
200     this movie had into me was wondering when and if Iron Man would knock Captain America out...Oh how I wished he had :(  
201     What were these other characters anyways? Useless, bickering idiots who really couldn't organise happy times in a brewery.  
202     The film was a chaotic mish mash of action elements and failed 'set pieces'... I found the villain to be quite amusing.  
203     And now I give up. This movie is not robbing any more of my time but I felt I ought to contribute to restoring the obvious  
204     fake rating and reviews this movie has been getting on IMdb.""",  
205  
206     """Absolutely dreadful! A complete waste of time. The plot was incoherent, the acting was wooden, and the special effects were laughably bad.  
207     I can't believe I sat through the entire thing. Save yourself the agony and steer clear of this cinematic disaster.""",  
208  
209     """Terrible from start to finish! The storyline was confusing, the acting was painfully awkward, and the dialogue felt like  
210     it was written by a child. I couldn't wait for it to end. It's astonishing how a film this bad even got made.  
211     Save your money and skip this cinematic catastrophe.""",  
212  
213     """Dreadful film, an absolute letdown. The plot was so predictable, and the characters were one-dimensional.  
214  
215     # Etichetele recenziilor  
216     y_true = np.array([0, 0, 0, 0, 0, 1, 1, 1, 1, 1])
```

3. Predicții asupra noului set de date

```
243 results = []  
244  
245 for i in range(10):  
246     rev = reviews[i]  
247     f1 = clean(rev)  
248     f2 = is_special(f1)  
249     f3 = to_lower(f2)  
250     f4 = rem_stopwords(f3)  
251     f5 = stem_txt(f4)  
252  
253     bow, words = [], word_tokenize(f5)  
254     for word in words:  
255         bow.append(words.count(word))  
256  
257     inp = []  
258     for j in word_dict:  
259         inp.append(f5.count(j[0]))  
260  
261     # Se realizeaza predicții pentru fiecare algoritm  
262     y_pred_bnb = bnb.predict(np.array(inp).reshape(1, 1000))  
263     y_pred_mnb = mnb.predict(np.array(inp).reshape(1, 1000))  
264     y_pred_gnb = gnb.predict(np.array(inp).reshape(1, 1000))  
265  
266     # Se adauga rezultatele in lista de rezultate  
267     results.append((y_pred_bnb[0], y_pred_mnb[0], y_pred_gnb[0]))  
268  
269 # Lista de rezultate este transformata intr-un array NumPy  
270 results_array = np.array(results)
```

1. Se definește o listă goală numită `results`, care va fi utilizată pentru a stoca rezultatele predicțiilor pentru fiecare algoritm.
2. Se parcurg primele zece recenzii din lista `reviews`.
3. Pentru fiecare recenzie, se aplică funcțiile de preprocesare asupra textului recenziei (`clean`, `is_special`, `to_lower`, `rem_stopwords`, `stem_txt`) pentru a-l pregăti pentru analiza ulterioară.
4. Se realizează reprezentarea textului preprocesat sub formă de Bag-of-Words (bow), adică se contorizează frecvența cuvintelor.
5. Se construiește un vector de intrare `inp` care reprezintă contorul de frecvență al cuvintelor din recenzie în raport cu un dicționar predefinit `word_dict`.
6. Se efectuează predicții pentru fiecare algoritm (BernoulliNB, MultinomialNB, GaussianNB) pe baza vectorului de intrare.
7. Rezultatele predicțiilor pentru fiecare algoritm sunt adăugate în lista `results`.
8. Lista de rezultate este transformată într-un array NumPy numit `results_array`.

4. Generarea matricilor de confuzie

```
272 # Calcularea si afisarea matricilor de confuzie pentru fiecare algoritm
273 confusion_matrix_bnb = confusion_matrix(y_true, results_array[:, 0])
274 confusion_matrix_mnb = confusion_matrix(y_true, results_array[:, 1])
275 confusion_matrix_gnb = confusion_matrix(y_true, results_array[:, 2])
276
277 print("Matricea de confuzie pentru BernoulliNB:")
278 print(confusion_matrix_bnb)
279
280 print("\nMatricea de confuzie pentru MultinomialNB:")
281 print(confusion_matrix_mnb)
282
283 print("\nMatricea de confuzie pentru GaussianNB:")
284 print(confusion_matrix_gnb)
```

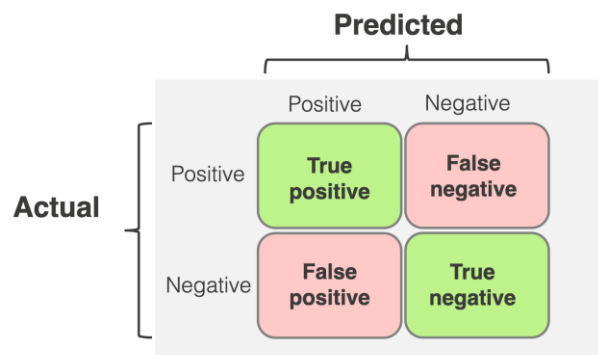
Matricea de confuzie pentru BernoulliNB:
[[4 1]
 [2 3]]

Matricea de confuzie pentru MultinomialNB:
[[0 5]
 [0 5]]

Matricea de confuzie pentru GaussianNB:
[[0 5]
 [2 3]]

5. Interpretarea rezultatelor

Matricele de confuzie sunt utilizate pentru a evalua performanța unui model de clasificare pe un set de date. Acestea sunt de forma:



Pentru BernoulliNB:

1. True Positive (TP): 4
2. True Negative (TN): 3
3. False Positive (FP): 2
4. False Negative (FN): 1

Pentru MultinomialNB:

1. True Positive (TP): 0
2. True Negative (TN): 5
3. False Positive (FP): 0
4. False Negative (FN): 5

Pentru GaussianNB:

1. True Positive (TP): 0
2. True Negative (TN): 3
3. False Positive (FP): 2
4. False Negative (FN): 5

Concluzii pe baza matricilor de confuzie:

Pentru BernoulliNB:

1. Precizie (Precision):

$$\text{Precision} = \frac{4}{4+2}$$

Aproximativ 67% dintre exemplele clasificate pozitiv au fost clasificate corect.

2. Acuratețe (Accuracy):

$$\text{Accuracy} = \frac{4+3}{4+3+2+1}$$

Acuratețea generală a modelului este de aproximativ 70%.

3. Sensibilitate (Recall):

$$\text{Recall} = \frac{4}{4+1}$$

Modelul identifică corect aproximativ 80% din exemplele pozitive.

4. Specificitate (Specificity):

$$\text{Specificity} = \frac{3}{3+2}$$

Modelul are o specificitate de aproximativ 60%.

5. F1 Score:

$$\text{F1 Score} = \frac{2 \times \frac{4}{6} \times \frac{4}{5}}{\frac{4}{6} + \frac{4}{5}}$$

Scorul F1 este o măsură echilibrată între precizie și recall.

Pentru MultinomialNB:

1. Precizie (Precision):

$$\text{Precision} = \frac{0}{0+0}$$

Din cauza lipsei de True Positive în clasificarea pozitivă, precizia este zero.

2. Acuratețe (Accuracy):

$$\text{Accuracy} = \frac{0+5}{0+5+0+5}$$

Acuratețea generală a modelului este de 50%.

3. Sensibilitate (Recall):

$$\text{Recall} = \frac{0}{0+5}$$

Modelul nu identifică niciun exemplu pozitiv.

4. Specificitate (Specificity):

$$\text{Specificity} = \frac{5}{5+0}$$

Modelul are o specificitate de 100%.

5. F1 Score:

$$F1 \text{ Score} = 0$$

Scorul F1 este zero din cauza lipsei de True Positive.

Pentru GaussianNB:

1. Precizie (Precision):

$$\text{Precision} = \frac{0}{0+2}$$

Din nou, din cauza lipsei de True Positive în clasificarea pozitivă, precizia este zero.

2. Acuratețe (Accuracy):

$$\text{Accuracy} = \frac{0+3}{0+3+2+5}$$

Acuratețea generală a modelului este de 30%.

3. Sensibilitate (Recall):

$$\text{Recall} = \frac{0}{0+5}$$

Modelul nu identifică niciun exemplu pozitiv.

4. Specificitate (Specificity):

$$\text{Specificity} = \frac{3}{3+2}$$

Modelul are o specificitate de 60%.

5. F1 Score:

$$F1 \text{ Score} = 0$$

Scorul F1 este zero din cauza lipsei de True Positive.

Pe baza observațiilor din matricile de confuzie, pot trage următoarele concluzii:

1. Modelele MultinomialNB și GaussianNB au dificultăți semnificative în identificarea exemplurilor pozitive, deoarece nu au True Positive în această clasă.
2. BernoulliNB are o performanță relativ mai bună, dar încă întâmpină dificultăți în identificarea corectă a exemplurilor pozitive.
3. Specificitatea este mai bună pentru MultinomialNB și GaussianNB, dar acest lucru se datorează faptului că aceste modele nu au clasificat corect niciun exemplu pozitiv, și nu unei capacități eficiente de identificare a exemplurilor negative.

Concluzia generală este că BernoulliNB pare să fie cel mai robust și precis în contextul acestui set de date specific.