

## Tema2 - IA

### I. Air Pollution Dataset

#### 1. Explorarea Datelor (Exploratory Data Analysis)

##### a. Analiza tipului de atribute si a plajei de valori a acestora

###### Atribute numerice continue

Dupa cum este indicat si in tabelul din anexa de la sectiunea 5.3. Descrierea atributelor, in acest set de date, atributele numerice continue sunt: "AQI\_Value", "CO\_Value", "Ozone\_Value", "NO2\_Value", "PM25\_Value", "VOCs", "SO2".

Pentru setul de antrenament (air\_pollution\_train.csv), datele cerute pentru atributele numerice continue (returnate de functia .describe()), sunt prezentate in tabelul urmator:

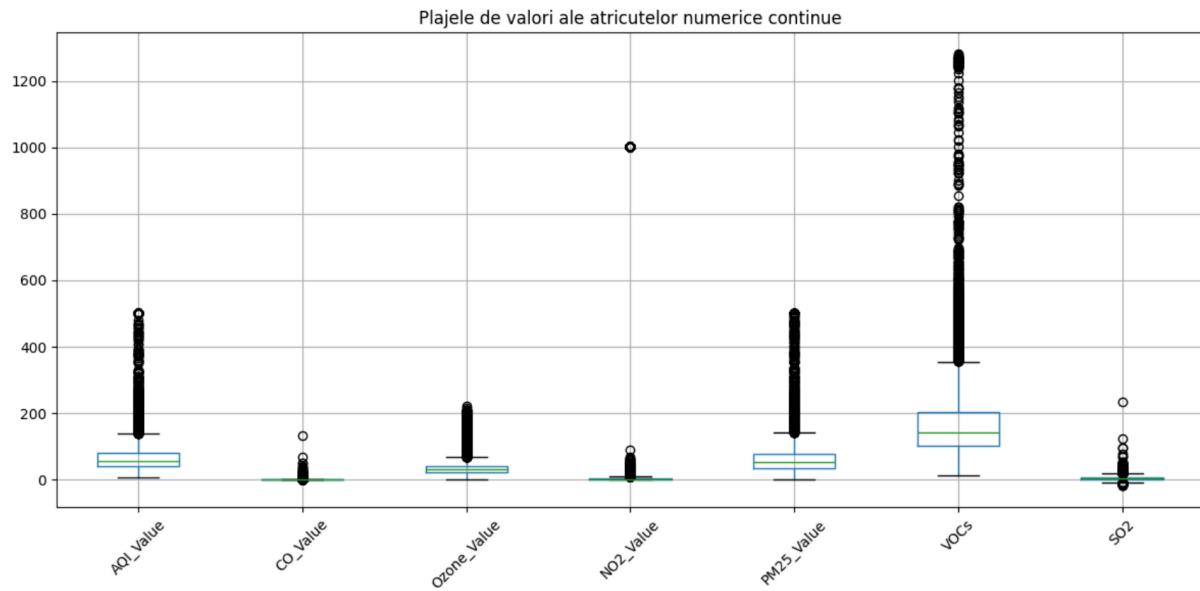
	count	mean	std	min	25%	50%	75%	max
AQI_Value	18770.0	71.981726	56.110722	7.000000	39.000000	55.000000	79.000000	500.000000
CO_Value	18770.0	1.378476	1.932713	0.000000	1.000000	1.000000	1.000000	133.000000
Ozone_Value	16900.0	35.372781	28.422401	0.000000	21.000000	31.000000	40.000000	222.000000
NO2_Value	18770.0	43.133438	196.182302	0.000000	0.000000	1.000000	4.000000	1003.063334
PM25_Value	18770.0	68.490996	54.717105	0.000000	35.000000	54.000000	78.000000	500.000000
VOCs	18770.0	185.006426	140.651248	12.415670	103.092298	142.817708	203.969738	1280.988229
SO2	18770.0	4.461538	6.077151	-18.528019	0.734223	4.286593	7.936256	234.692971

Pentru setul de test (air\_pollution\_test.csv), datele pentru aceleasi atribute sunt prezentate in tabelul urmator:

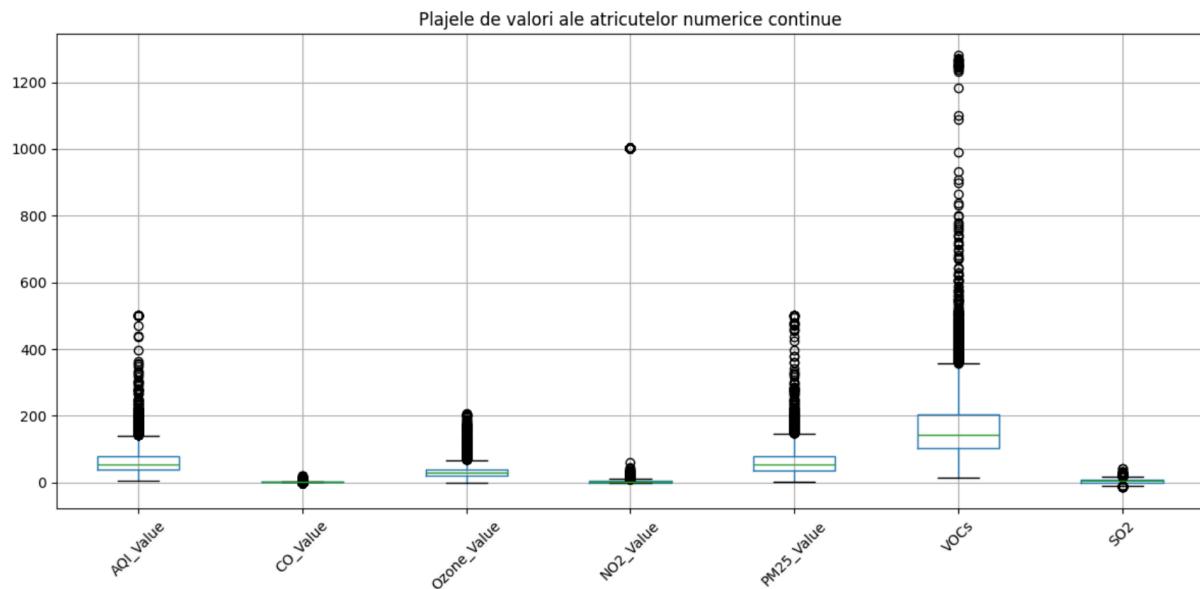
	count	mean	std	min	25%	50%	75%	max
AQI_Value	4693.0	72.127424	55.838493	6.000000	39.000000	55.000000	80.000000	500.000000
CO_Value	4693.0	1.327935	1.356232	0.000000	1.000000	1.000000	1.000000	21.000000
Ozone_Value	4217.0	34.706189	27.023739	0.000000	21.000000	30.000000	40.000000	207.000000
NO2_Value	4693.0	42.887032	195.686936	0.000000	0.000000	1.000000	4.000000	1003.063334
PM25_Value	4693.0	68.634775	55.118326	2.000000	35.000000	54.000000	80.000000	500.000000
VOCs	4693.0	185.239825	139.841721	15.461284	103.767142	143.413477	205.616347	1279.853139
SO2	4693.0	4.393056	5.431688	-13.338278	0.741786	4.288641	7.814964	41.595139

Vizualizarea plajelor de valori pentru atributele numerice continue:

- pentru setul de train:



- pentru setul de test:



Interpretare:

- pentru fiecare atribut, in interiorul boxului albastru avem plaja de valori dintre quartila 1 (Q1, de 25%) si quartila 3 (Q3, de 75%), iar linia verde din mijloc reprezinta mediana (Q2, de 50%)
- cele doua linii orizontale paralele de deasupra si de dedesubtul fiecarui box (numite whiskers) reprezinta distanta de la Q1 la cel mai mic punct care nu este considerat outlier, respectiv distanta de la Q3 la cel mai mare punct care nu este considerat outlier (vom vedea mai tarziu in implementarea agloritmului IQR ca outlierii se afla in afara intervalului  $[Q1 - 1.5 * IQR, Q3 + 1.5 * IQR]$ )
- cercurile negre reprezinta outlierii

Dintre toate attributele, VOCs are cea mai larga plaja de valori, outlierii fiind foarte diversi si intinzandu-se pana la valori foarte mari (peste 1200), ceea ce ar putea indica o scala diferita.

Daca nu se normalizeaza, VOCs ar putea domina predictia pentru ca are cea mai mare plaja de valori.

La CO\_Value si NO2\_Value, majoritatea valorilor sunt foarte mici, box-ul fiind lipit de axa, adica cele 2 laturi ale cutiei coincid (pentru Q1 si Q3 aproape coincid cu linia verde pentru mediana (Q2). Si aici se poate pune problema despre o posibila scalare diferita fata de celelalte atribute si este necesara standardizarea sau normalizarea datelor.

La AQI\_Value si PM25\_value, putem vedea foarte clar delimitata cutie albastra si linia verde orizontala, adica distributia centrala este foarte clar delimitata, dar avem si valori outlier evidente, care vor trebui tratate ulterior prin inlocuire (eliminare + imputare) pentru a nu influenta modelele de invatare automata in mod negativ.

La SO2 distributia este moderat simetrica, avand cei mai putini outlieri.

In concluzie, pentru a elimina diferențele majore de scala intre atribute, se observa necesitatea standardizarii datelor inainte de antrenare, iar pentru a nu influenta negativ modelele de ML, este necesara tratarea outlierilor.

### Atribute discrete sau ordonale

```
discrete_or_ordinal_attributes = [  
    "Country", "City", "CO_Category", "Ozone_Category",  
    "NO2_Category", "PM25_Category", "Emissions", "AQI_Category"  
]
```

In urmatoarele tabele afisam pentru fiecare atribut categorial numarul de exemple care nu au valori lipsa (Not null), respectiv numarul de valori unice:

- pentru setul de train

	count_non_missing	num_unique_values
Country	18421	175
City	18770	18770
CO_Category	16877	2
Ozone_Category	18770	5
NO2_Category	18770	2
PM25_Category	18770	6
Emissions	18770	6
AQI_Category	18770	6

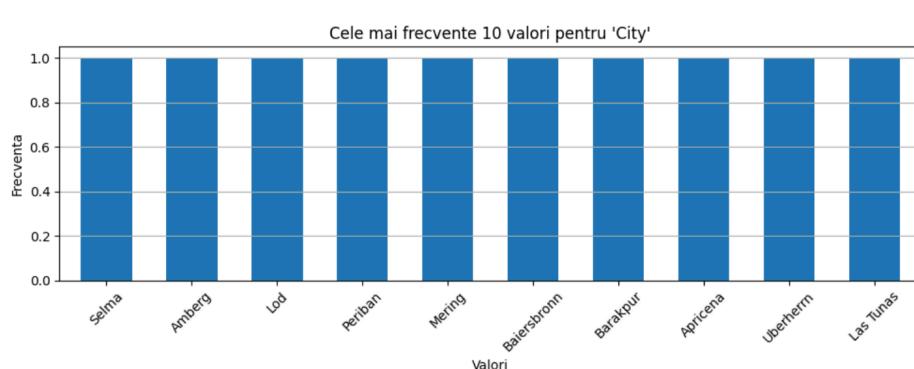
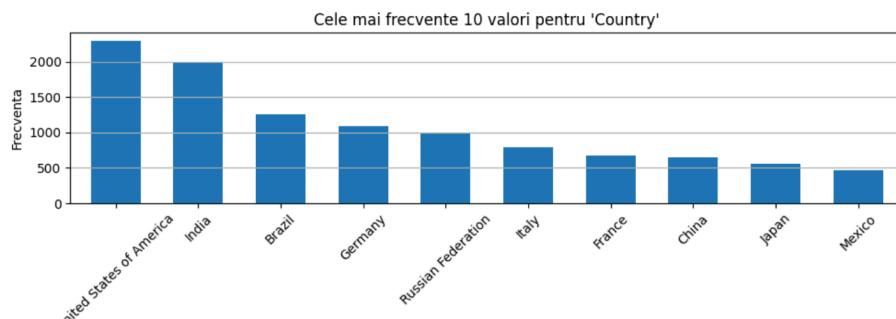
- pentru setul de test

	count_non_missing	num_unique_values
Country	4615	148
City	4692	4692
CO_Category	4240	1
Ozone_Category	4693	5
NO2_Category	4693	2
PM25_Category	4693	6
Emissions	4693	6
AQI_Category	4693	6

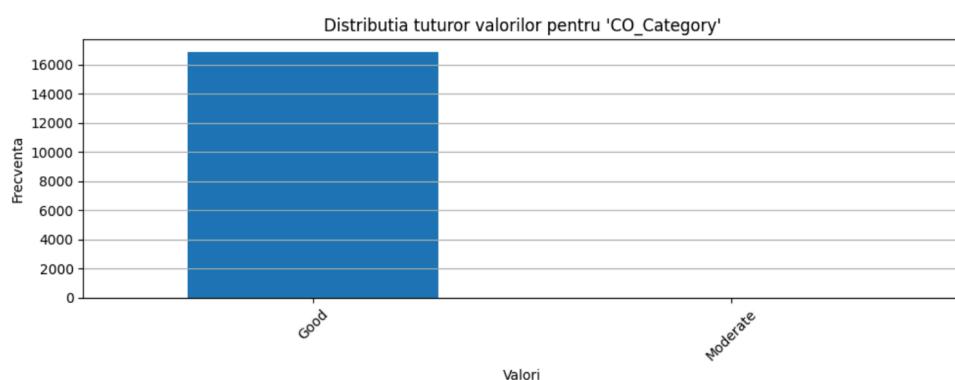
Ceea ceiese in evidenta din aceste doua tabele este faptul ca pentru atributul City, numarul de valori unice este egal cu numarul de valori not null, ceea ce inseamna ca pentru fiecare exemplu vom avea un oras diferit, deci City nu este relevant pentru predictii, asa ca il putem elibera pentru a diminua dimensiunea datelor (fara a afecta in vreun fel predictia).

Grafice de tip histograma pentru a observa vizual distributia valorilor pentru fiecare atribut categoric / ordinal:

- pentru setul de train:

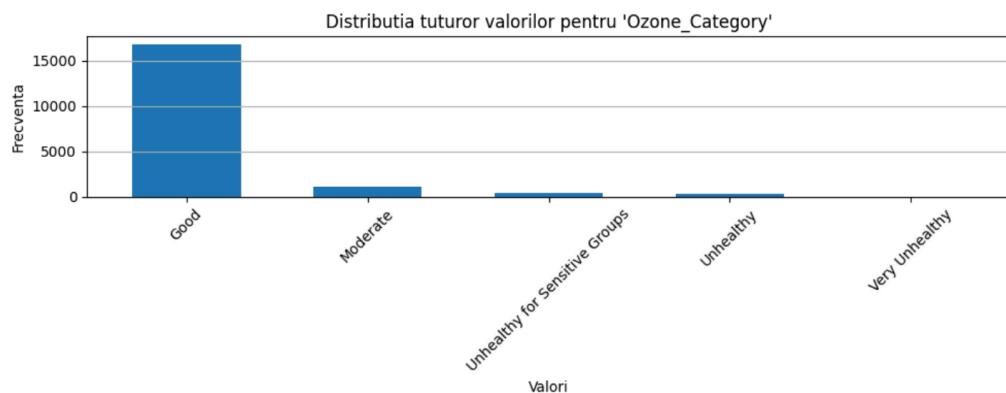


Din nou, se observa ceea ce am concluzionat si din tabel, si anume faptul ca avem cate un oras diferit pentru fiecare exemplu din setul de date (nu se repeta niciun oras), caci noi am afisat in histograma cele mai frecvente 10 valori pentru City, iar toate acestea sunt 1.

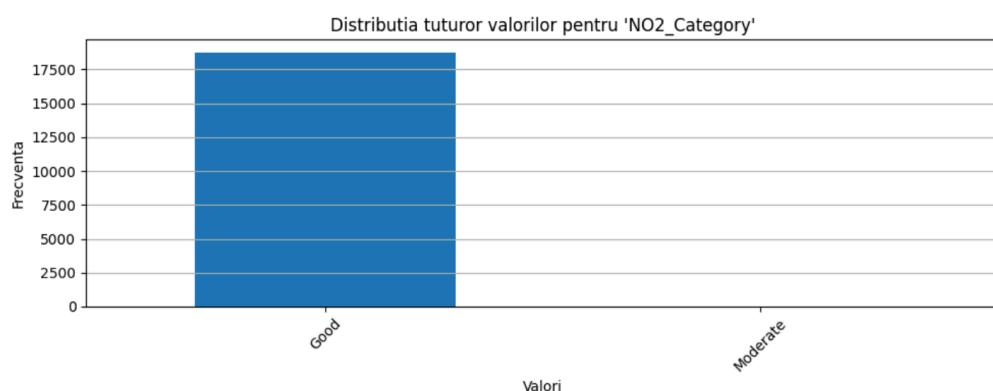


Acest grafic de tip histograma ilustreaza faptul ca setul de date este dezechilibrat pe atributul CO\_Category, intrucat clasa dominanta Good apare cu o frecventa foarte mare (de

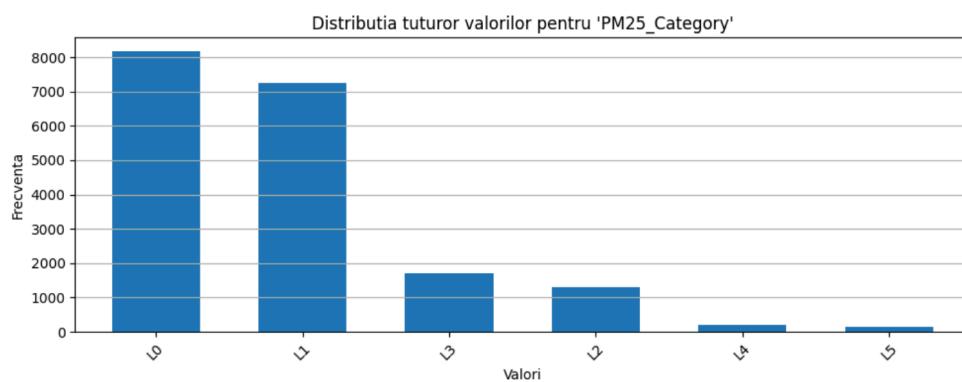
peste 16000 de ori), pe cand clasa Moderate apare extrem de rar. Prin urmare, acest atribut va avea putere de discriminare redusa pentru algoritmi.



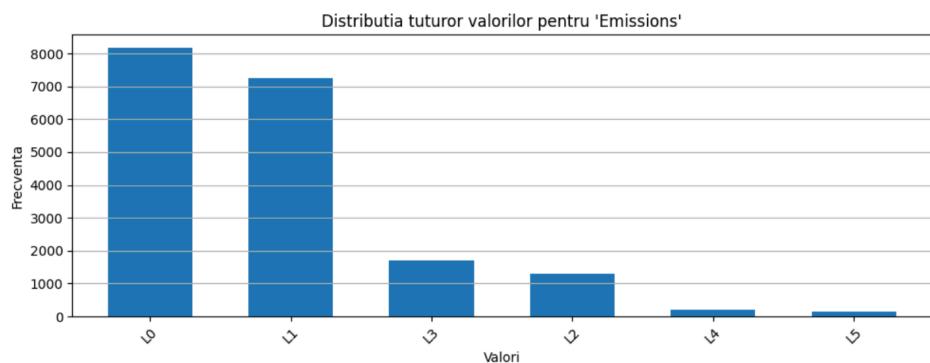
Si in cazul atributului Ozone\_Category, clasa Good este iar majoritara, dar, spre deosebire de atributul CO\_Category, diversitatea valorilor este mai mare (Apar mai multe clase: Moderate, Unhealthy, Very Unhealthy), iar desi dezechilibrul de clase este prezent si aici, nu este extrem.



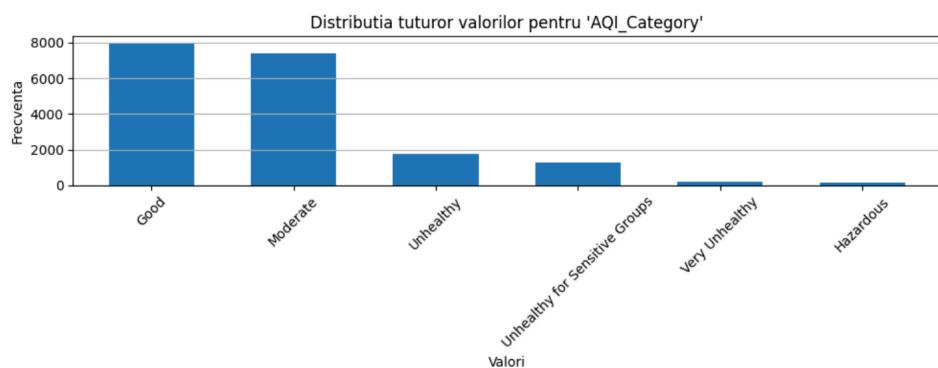
Similar cu situatia atributului CO\_Category, graficul ilustreaza un dezechilibru masiv pentru atributul NO2\_Category, unde aproape toate valorile (peste 17500) sunt Good, iar din clasa Moderate avem foarte putine exemple.



Spre deosebire de atributele prezentate anterior, la PM25\_category avem o distributie mai echilibrata comparativ cu celelalte atribute si o diversitate mai mare.

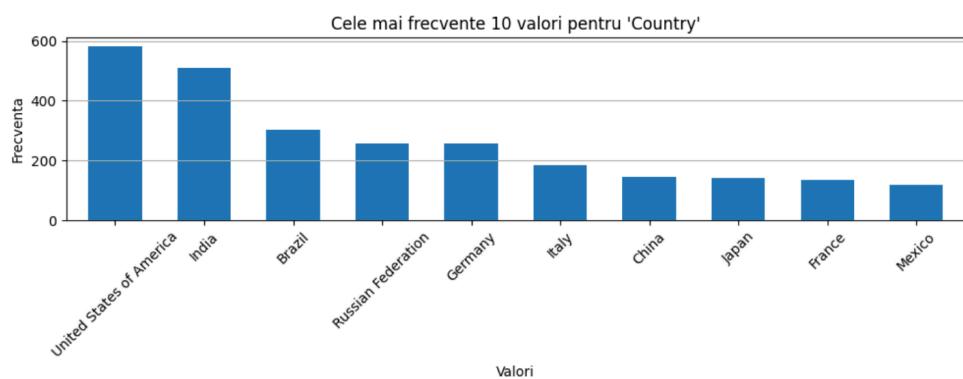


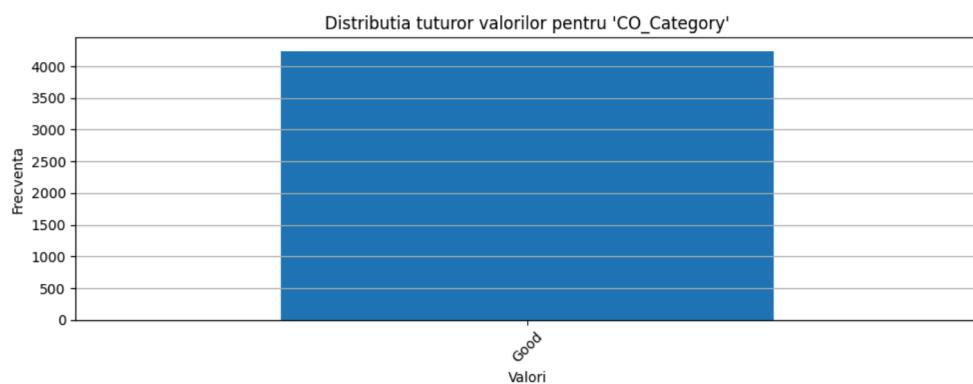
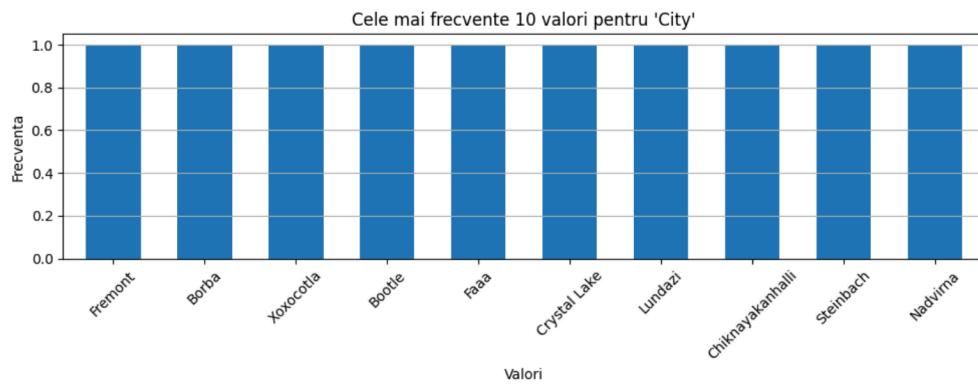
Si aici ca la PM25\_Category, se observa o diversitate mai mare a claselor, ceea ce va fi util pentru invatarea autmata. L0 si L1 sunt in mod evident clasele dominante, dar totusi dezechilibrul este moderat, pentru ca si celelalte clase au un numar rezonabil de aparitii.



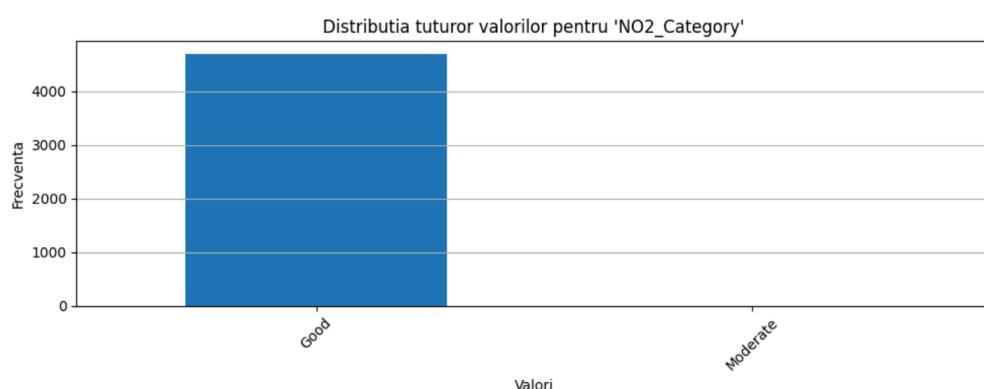
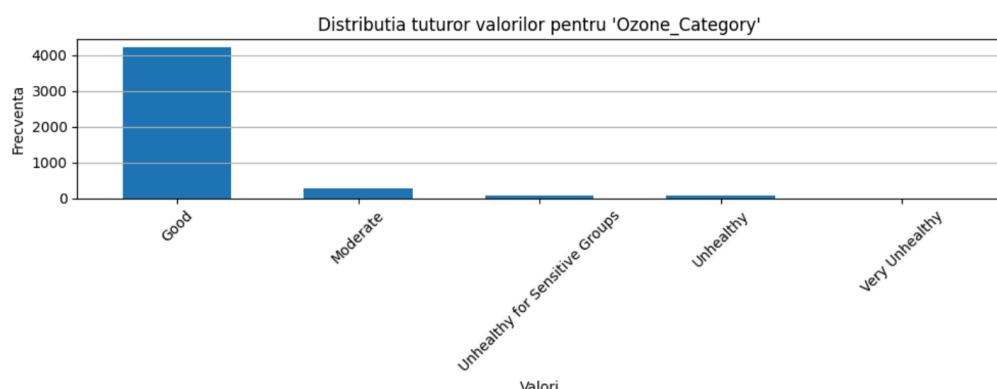
AQI\_Category este eticheta tinta, destul de diversa, dar tot moderat dezechilibrata, cu Good si Moderate clase dominante. Observam faptul ca Very Unhealthy si Hazardous sunt clase foarte rare, ceea ce ar putea ridica dificultati pentru modelele de invatare autmoata, carora le va fi mai greu sa prezice in mod corect clasele rare.

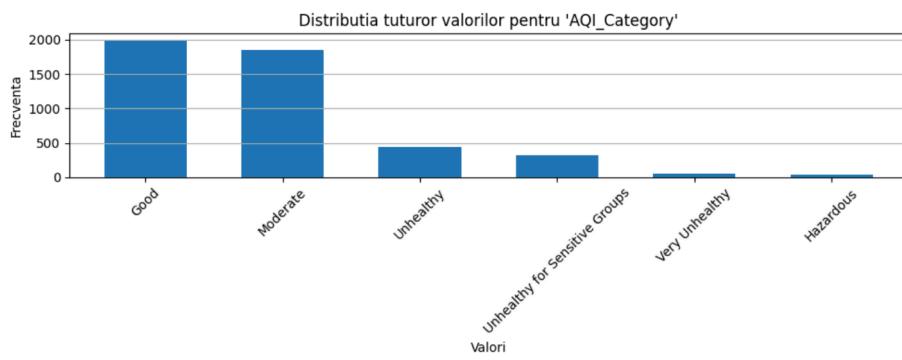
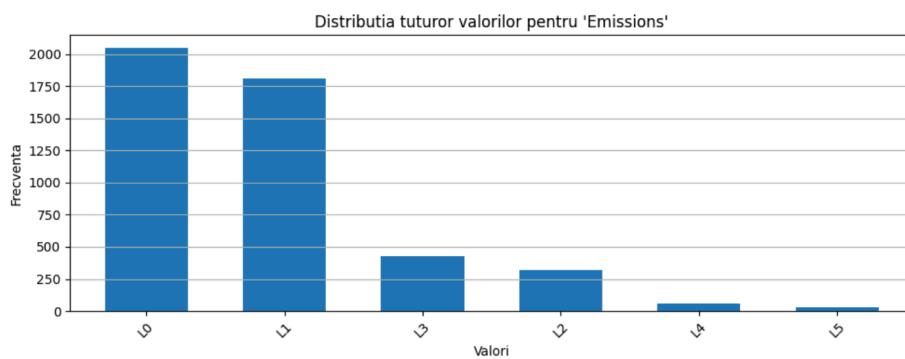
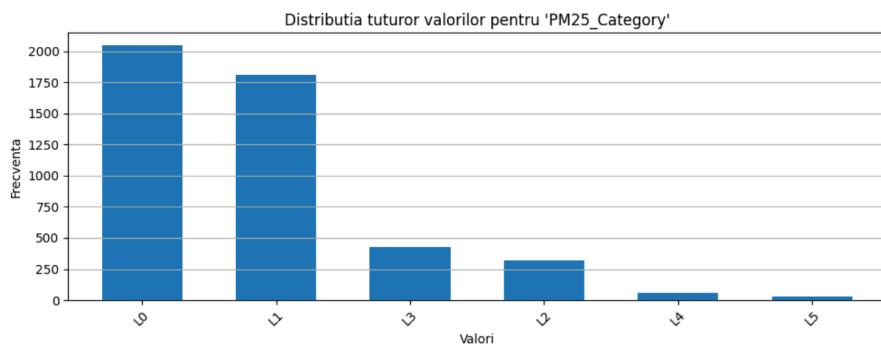
- pentru setul de test:





Se observa faptul ca, spre deosebire de setul de train, in cel de test nu mai avem niciun exemplu pentru Moderate, ci toate valorile pentru CO\_Category vor fi Good.

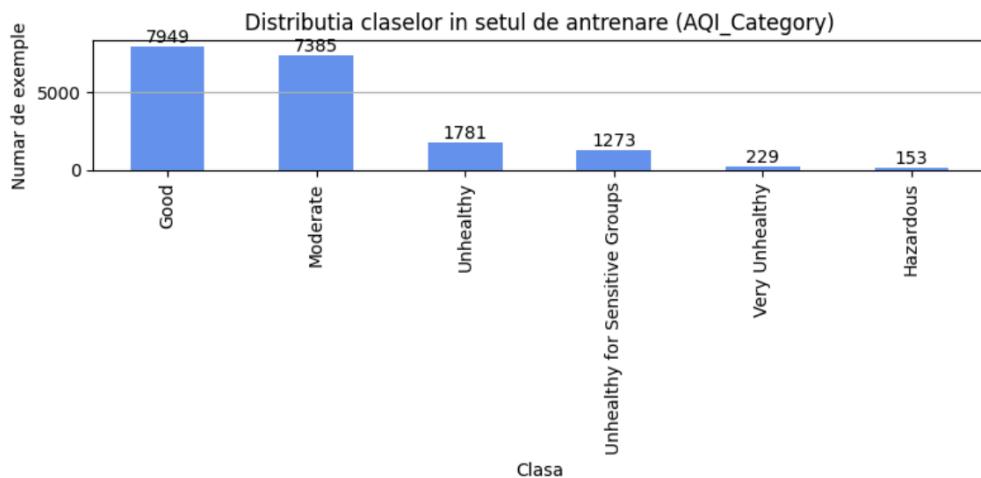


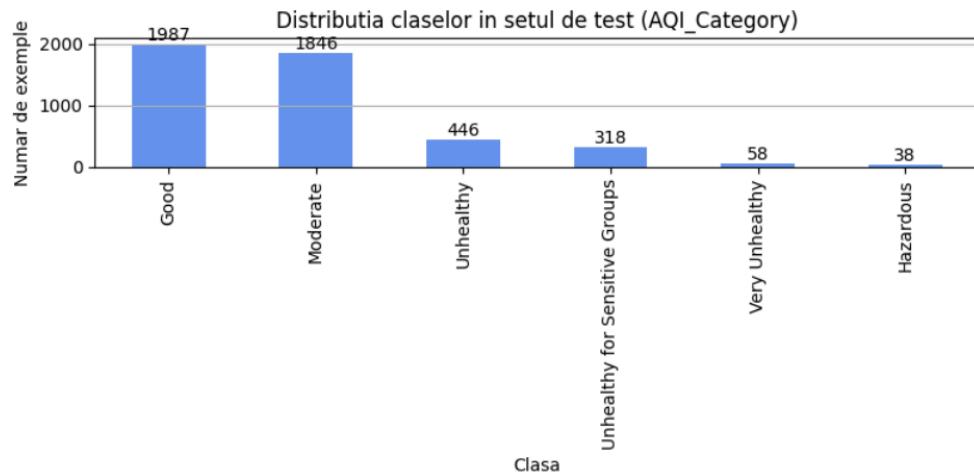


In rest, observatiile facute pentru setul de train raman valabile si pentru set de test.

## 2. Analiza echilibrului de clase

Barplot pentru frecventa de aparitie a fiecarei etichete in setul de date de antrenare / test:

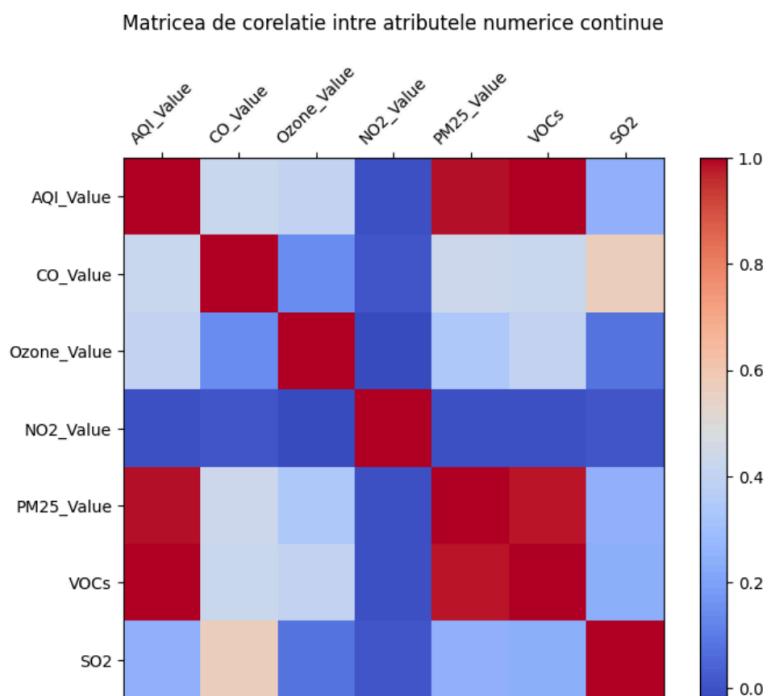




Din aceste 2 barplot-uri se observă un dezechilibru de clase. "Good" și "Moderate" sunt cele mai frecvente clase în ambele seturi (în setul de antrenare, pentru Good avem 7949 exemple, iar pentru Moderate avem 7385 exemple, iar în setul de testare, pentru Good avem 1987 exemple, iar pentru Moderate avem 1846 exemple). Pe de altă parte, clasele Hazardous și Very Unhealthy au frecvențe extrem de mici în ambele seturi (în setul de antrenare, Very Unhealthy are 229 exemple, Hazardous are 153 exemple, iar în setul de testare, Very Unhealthy are 58 de exemple, iar Hazardous are 38 de exemple). Acest dezechilibru între clase poate afecta performanța modelelor de clasificare, care vor invata foarte bine clasele frecvente, dar pot ignora clasele rare.

### 3. Analiza corelatiei intre atrbute

a. Analiza de corelatie intre atributele numerice continue



Doua atribute sunt puternic corelate între ele daca au valori ale indicelui de corelație apropriate de -1 sau 1. Analizand matricea de corelație intre atributele numerice continue, observam urmatoarele valori ale indicelui de corelație apropriate de 1:

- intre AQI\_Value si PM25\_Value: 0.984160
- intre VOCs si AQI\_Value: 0.997472
- intre VOCs si PM25\_Value: 0.981831

Astfel, AQI\_Value este corelat atat cu PM25\_Value, cat si cu VOCs, iar VOCs si PM25\_Value sunt si ele la randul lor corelate intre ele, deci putem elimina atributele PM25\_Value si VOCs, deoarce nu ofera informatie in plus care sa ne ajute la clasificare si maresc setul de date degeaba, ingreunand invataarea, deci sunt atribute redundante.

#### b. Analiza de corelatie intre atributele categorice

Pentru atributele categorice nu am mai construit matricea de corelatie Pearson, ci am evaluat corelatia prin testul statistic Chi-Patrat, care porneste de la premisa ca variabilele sunt independente. Daca testul infirma aceasta ipoteza, putem spune ca variabilele sunt corelate, iar in caz contrar, putem afirma ca, intr-adevar, variabilele sunt independente.

Daca p-value > 0.05 => Ipoteza este acceptata => Variabilele nu sunt corelate.

Daca p-value < 0.05 => ipoteza este respinsa => Variabilele sunt corelate.

- pentru setul de train:

	Attribute 1	Attribute 2	p-value
2	Country	Ozone_Category	0.000000e+00
6	Country	AQI_Category	0.000000e+00
5	Country	Emissions	0.000000e+00
4	Country	PM25_Category	0.000000e+00
25	PM25_Category	Emissions	0.000000e+00
21	Ozone_Category	AQI_Category	0.000000e+00
20	Ozone_Category	Emissions	0.000000e+00
19	Ozone_Category	PM25_Category	0.000000e+00
27	Emissions	AQI_Category	0.000000e+00
26	PM25_Category	AQI_Category	0.000000e+00
3	Country	NO2_Category	3.773165e-232
23	NO2_Category	Emissions	1.400069e-70
22	NO2_Category	PM25_Category	1.400069e-70
24	NO2_Category	AQI_Category	1.399454e-61
13	CO_Category	Ozone_Category	4.819449e-42
17	CO_Category	AQI_Category	4.278661e-20
16	CO_Category	Emissions	6.066628e-14
15	CO_Category	PM25_Category	6.066628e-14
1	Country	CO_Category	1.279980e-04
0	Country	City	4.725000e-01
12	City	AQI_Category	4.947821e-01
10	City	PM25_Category	4.947821e-01
11	City	Emissions	4.947821e-01
8	City	Ozone_Category	4.951957e-01
7	City	CO_Category	4.963809e-01
9	City	NO2_Category	4.965683e-01
18	Ozone_Category	NO2_Category	8.080989e-01
14	CO_Category	NO2_Category	1.000000e+00

Din acest tabel, comparand p-value cu 0.05, putem afirma urmatoarele:

- Country este in corelatie cu urmatoarele atribute:
  - Country - Ozone\_Category
  - Country - AQI\_Category
  - Country - Emissions
  - Country - PM25\_Category (pe acesta l-am eliminat deja)
  - Country - NO2\_Category
  - Country - CO\_Category

Dintre acestea, doar cu Ozone\_Category, PM25\_Category (care e deja eliminat) si AQI\_Category este puternic corelat (p-value = 0), deci ar putea fi eliminat. Country nu masoara direct poluarea, dar poate reflecta nivelul mediu de poluare prin faptul ca tarile dezvoltate au o poluare mai mica (mai multe observatii de Good sau Moderate), in timp ce tarile subdezvoltate si industrializate au un grad de poluare mai mare (in cazul acesta exista mai multe observatii cu Unhealthy, Hazardous). Daca intr-adevar acest tipar este consistent, atunci Country poate fi considerat un atribut redundant si poate fi eliminat.

- Ozone\_Category este corelat cu: AQI\_Category, Emissions, PM25\_Category (pe care am stabilit deja ca il eliminam deci nu mai prezinta importanta)
- NO2\_Category este corelat cu AQI\_Category, Emissions, PM25\_Category
- CO\_Category este corelat cu Ozone\_Category, Emissions, PM25\_Category, AQI\_Category
- City -> pentru toate relatiile, p-value este aproximativ egal cu 0.49, adica mai mare decat 0.05, ceea ce inseamna ca City este independent de toate celelalte atribute. Deci ni s-a confirmat ceea ce observasem si mai devreme, si anume faptul ca City nu este informativ pentru AQI\_Category. Il eliminam deocamdată.
- CO\_Category si NO2\_Category: p\_value = 1 >> 0.05 => complet independente
- Ozone\_Category si NO2\_Category: p\_value = 0.88 => independente
- pentru setul de test:

	Attribute 1	Attribute 2	p-value
5	Country	Emissions	0.000000e+00
6	Country	AQI_Category	0.000000e+00
4	Country	PM25_Category	0.000000e+00
21	Ozone_Category	AQI_Category	0.000000e+00
25	PM25_Category	Emissions	0.000000e+00
26	PM25_Category	AQI_Category	0.000000e+00
27	Emissions	AQI_Category	0.000000e+00
20	Ozone_Category	Emissions	2.231502e-182
19	Ozone_Category	PM25_Category	2.231502e-182
2	Country	Ozone_Category	5.822991e-126
22	NO2_Category	PM25_Category	8.676168e-16
23	NO2_Category	Emissions	8.676168e-16
24	NO2_Category	AQI_Category	8.676168e-16
0	Country	City	4.495532e-01
12	City	AQI_Category	4.895642e-01
11	City	Emissions	4.895642e-01
10	City	PM25_Category	4.895642e-01
8	City	Ozone_Category	4.903912e-01
9	City	NO2_Category	4.931361e-01
18	Ozone_Category	NO2_Category	9.985349e-01
3	Country	NO2_Category	1.000000e+00
1	Country	CO_Category	1.000000e+00
13	CO_Category	Ozone_Category	1.000000e+00
7	City	CO_Category	1.000000e+00
15	CO_Category	PM25_Category	1.000000e+00
14	CO_Category	NO2_Category	1.000000e+00
17	CO_Category	AQI_Category	1.000000e+00
16	CO_Category	Emissions	1.000000e+00

Observatiile facute anterior raman valabile.

## 2. Preprocesarea datelor

### 2.1. Date lipsa pentru un atribut intr-un esantion

```
identify_attributes_with_missing_values(df_train)
✓ 0.0s
CO_Category    1893
Ozone_Value    1870
Country        349
dtype: int64

identify_attributes_with_missing_values(df_test)
✓ 0.0s
Ozone_Value    476
CO_Category    453
Country        78
City            1
dtype: int64
```

Observam ca pe setul de training, pentru atributele CO\_Category, Ozone\_Value si Country, avem valori lipsa, iar pe setul de test, in plus față de atributele mentionate pentru setul de training, avem o valoare lipsa și la City, dar nu este neapărat relevant pentru că oricum am stabilit că eliminăm atributul City.

Acum urmează să facem imputare pentru atributele cu valori lipsa:

- A. **Imputare univariata** - pentru fiecare atribut cu valori lipsa, se completează acestea folosind doar acel atribut
  - Ozone\_Value: acesta este un atribut numeric continuu, putem completa valorile lipsa cu valoarea medianei
  - CO\_Category: pentru că acesta este un atribut categorial, nu numeric (cum era Ozone\_Value), vom completa valorile lipsa cu valoarea cea mai frecventă din setul de date a acestui atribut
  - Country: analog CO\_Category
- B. **Imputare multivariata** - valorile lipsa ale unui atribut sunt prezise pe baza celorlalte atribute
  - imputarea multivariată (Iterative imputation) poate fi aplicată doar pe atributele numerice, deoarece în spate antrenează modele de regresie care nu pot trata valori categorice direct => iterative imputation va funcționa doar pentru Ozone\_Value
  - pentru atributele categorice (CO\_Category și Country) aplicăm tot imputare univariată

Pentru valorice numerice continue, statisticile arată astăzi în urma imputării (de ambele tipuri):

- pentru setul de train:

	Source	count	mean	std	min	25%	50%	75%	max
AQI_Value	Original (cu NaN)	18770.0	71.981726	56.110722	7.000000	39.000000	55.000000	79.000000	500.000000
CO_Value	Original (cu NaN)	18770.0	1.378476	1.932713	0.000000	1.000000	1.000000	1.000000	133.000000
Ozone_Value	Original (cu NaN)	16900.0	35.372781	28.422401	0.000000	21.000000	31.000000	40.000000	222.000000
NO2_Value	Original (cu NaN)	18770.0	43.133438	196.182302	0.000000	0.000000	1.000000	4.000000	1003.063334
PM25_Value	Original (cu NaN)	18770.0	68.490996	54.717105	0.000000	35.000000	54.000000	78.000000	500.000000
VOCs	Original (cu NaN)	18770.0	185.006426	140.651248	12.415670	103.092298	142.817708	203.969738	1280.988229
SO2	Original (cu NaN)	18770.0	4.461538	6.077151	-18.528019	0.734223	4.286593	7.936256	234.692971
AQI_Value	SimpleImputer	18770.0	71.981726	56.110722	7.000000	39.000000	55.000000	79.000000	500.000000
CO_Value	SimpleImputer	18770.0	1.378476	1.932713	0.000000	1.000000	1.000000	1.000000	133.000000
Ozone_Value	SimpleImputer	18770.0	34.937134	27.001145	0.000000	22.000000	31.000000	39.000000	222.000000
NO2_Value	SimpleImputer	18770.0	43.133438	196.182302	0.000000	0.000000	1.000000	4.000000	1003.063334
PM25_Value	SimpleImputer	18770.0	68.490996	54.717105	0.000000	35.000000	54.000000	78.000000	500.000000
VOCs	SimpleImputer	18770.0	185.006426	140.651248	12.415670	103.092298	142.817708	203.969738	1280.988229
SO2	SimpleImputer	18770.0	4.461538	6.077151	-18.528019	0.734223	4.286593	7.936256	234.692971
AQI_Value	IterativeImputer	18770.0	71.981726	56.110722	7.000000	39.000000	55.000000	79.000000	500.000000
CO_Value	IterativeImputer	18770.0	1.378476	1.932713	0.000000	1.000000	1.000000	1.000000	133.000000
Ozone_Value	IterativeImputer	18770.0	35.367977	27.342764	0.000000	22.000000	31.000000	40.000000	222.000000
NO2_Value	IterativeImputer	18770.0	43.133438	196.182302	0.000000	0.000000	1.000000	4.000000	1003.063334
PM25_Value	IterativeImputer	18770.0	68.490996	54.717105	0.000000	35.000000	54.000000	78.000000	500.000000
VOCs	IterativeImputer	18770.0	185.006426	140.651248	12.415670	103.092298	142.817708	203.969738	1280.988229
SO2	IterativeImputer	18770.0	4.461538	6.077151	-18.528019	0.734223	4.286593	7.936256	234.692971

- pentru setul de test:

	Source	count	mean	std	min	25%	50%	75%	max
AQI_Value	Original (cu NaN)	4693.0	72.127424	55.838493	6.000000	39.000000	55.000000	80.000000	500.000000
CO_Value	Original (cu NaN)	4693.0	1.327935	1.356232	0.000000	1.000000	1.000000	1.000000	21.000000
Ozone_Value	Original (cu NaN)	4217.0	34.706189	27.023739	0.000000	21.000000	30.000000	40.000000	207.000000
NO2_Value	Original (cu NaN)	4693.0	42.887032	195.686936	0.000000	0.000000	1.000000	4.000000	1003.063334
PM25_Value	Original (cu NaN)	4693.0	68.634775	55.118326	2.000000	35.000000	54.000000	80.000000	500.000000
VOCs	Original (cu NaN)	4693.0	185.239825	139.841721	15.461284	103.767142	143.413477	205.616347	1279.853139
SO2	Original (cu NaN)	4693.0	4.393056	5.431688	-13.338278	0.741786	4.288641	7.814964	41.595139
AQI_Value	SimpleImputer	4693.0	72.127424	55.838493	6.000000	39.000000	55.000000	80.000000	500.000000
CO_Value	SimpleImputer	4693.0	1.327935	1.356232	0.000000	1.000000	1.000000	1.000000	21.000000
Ozone_Value	SimpleImputer	4693.0	34.228851	25.655699	0.000000	22.000000	30.000000	39.000000	207.000000
NO2_Value	SimpleImputer	4693.0	42.887032	195.686936	0.000000	0.000000	1.000000	4.000000	1003.063334
PM25_Value	SimpleImputer	4693.0	68.634775	55.118326	2.000000	35.000000	54.000000	80.000000	500.000000
VOCs	SimpleImputer	4693.0	185.239825	139.841721	15.461284	103.767142	143.413477	205.616347	1279.853139
SO2	SimpleImputer	4693.0	4.393056	5.431688	-13.338278	0.741786	4.288641	7.814964	41.595139
AQI_Value	IterativeImputer	4693.0	72.127424	55.838493	6.000000	39.000000	55.000000	80.000000	500.000000
CO_Value	IterativeImputer	4693.0	1.327935	1.356232	0.000000	1.000000	1.000000	1.000000	21.000000
Ozone_Value	IterativeImputer	4693.0	34.676230	26.055587	0.000000	22.000000	30.000000	40.000000	207.000000
NO2_Value	IterativeImputer	4693.0	42.887032	195.686936	0.000000	0.000000	1.000000	4.000000	1003.063334

Analog, pentru atrbutele categorice:

- pentru setul de train:

	Source	count	unique	top	freq
Country	Original (cu NaN)	18421	175	United States of America	2290
City	Original (cu NaN)	18770	18770	Selma	1
CO_Category	Original (cu NaN)	16877	2	Good	16875
Ozone_Category	Original (cu NaN)	18770	5	Good	16842
NO2_Category	Original (cu NaN)	18770	2	Good	18756
PM25_Category	Original (cu NaN)	18770	6	L0	8163
Emissions	Original (cu NaN)	18770	6	L0	8163
AQI_Category	Original (cu NaN)	18770	6	Good	7949
Country	SimpleImputer	18770	175	United States of America	2639
City	SimpleImputer	18770	18770	Selma	1
CO_Category	SimpleImputer	18770	2	Good	18768
Ozone_Category	SimpleImputer	18770	5	Good	16842
NO2_Category	SimpleImputer	18770	2	Good	18756
PM25_Category	SimpleImputer	18770	6	L0	8163
Emissions	SimpleImputer	18770	6	L0	8163
AQI_Category	SimpleImputer	18770	6	Good	7949
Country	IterativeImputer	18770	175	United States of America	2639
City	IterativeImputer	18770	18770	Selma	1
CO_Category	IterativeImputer	18770	2	Good	18768
Ozone_Category	IterativeImputer	18770	5	Good	16842
NO2_Category	IterativeImputer	18770	2	Good	18756
PM25_Category	IterativeImputer	18770	6	L0	8163
Emissions	IterativeImputer	18770	6	L0	8163
AQI_Category	IterativeImputer	18770	6	Good	7949

Se observa cum, daca ne uitam la coloana de frecventa, aceasta creste doar pentru atributele pentru care s-a facut imputare (CO\_Category, Country), restul ramand la fel.

- pentru setul de test:

	Source	count	unique	top	freq
Country	Original (cu NaN)	4615	148	United States of America	582
City	Original (cu NaN)	4692	4692	Fremont	1
CO_Category	Original (cu NaN)	4240	1	Good	4240
Ozone_Category	Original (cu NaN)	4693	5	Good	4227
NO2_Category	Original (cu NaN)	4693	2	Good	4692
PM25_Category	Original (cu NaN)	4693	6	L0	2045
Emissions	Original (cu NaN)	4693	6	L0	2045
AQI_Category	Original (cu NaN)	4693	6	Good	1987
Country	SimpleImputer	4693	148	United States of America	660
City	SimpleImputer	4693	4692	Aabenraa	2
CO_Category	SimpleImputer	4693	1	Good	4693
Ozone_Category	SimpleImputer	4693	5	Good	4227
NO2_Category	SimpleImputer	4693	2	Good	4692
PM25_Category	SimpleImputer	4693	6	L0	2045
Emissions	SimpleImputer	4693	6	L0	2045
AQI_Category	SimpleImputer	4693	6	Good	1987
Country	IterativeImputer	4693	148	United States of America	660
City	IterativeImputer	4693	4692	Aabenraa	2
CO_Category	IterativeImputer	4693	1	Good	4693
Ozone_Category	IterativeImputer	4693	5	Good	4227
NO2_Category	IterativeImputer	4693	2	Good	4692
PM25_Category	IterativeImputer	4693	6	L0	2045
Emissions	IterativeImputer	4693	6	L0	2045
AQI_Category	IterativeImputer	4693	6	Good	1987

Ambele tipuri de imputare functioneaza (atât cea univariată, cât și cea multivariată), caci nu vom mai avea valori nule în urma niciunei dintre ele.

## 2.2. Valori extreme pentru un atribut într-un esantion

Primul pas: identificăm și să eliminăm outlierii cu ajutorul metodei IQR (inter-Quartile Range). Initial am urmat indicațiile și am considerat Q1 la 0.25 și Q3 la 0.75, dar se eliminau prea multe valori, care nu erau neapărat outlieri, iar după standardizare, deviația standardă a lui CO\_Value ajungea să fie 0 în loc să fie apropiată de 1, astăzi că am largit puțin intervalul și am luat Q1 la 0.2 și Q3 la 0.8.

Inainte nu mai aveam nicio valoare nula deoarece facusem imputare. În urma eliminării outlier-ilor, au apărut valori nule în proporție în care ne așteptăm încă de la boxplot-ul de la analiza de la început:

- pentru train (număr de alorii Nan aparute la atrbute numerice continue)

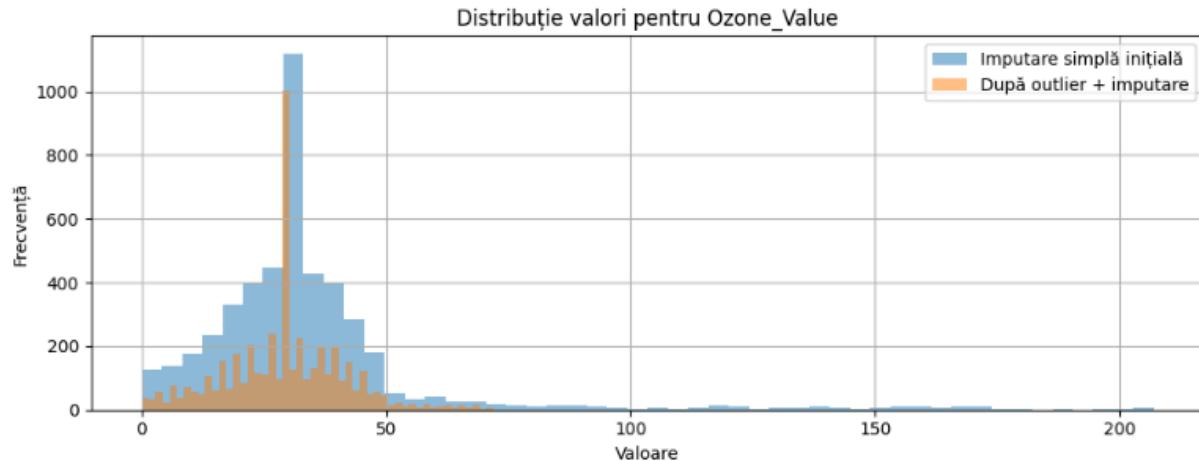
AQI_Value	726
CO_Value	1024
Ozone_Value	1021
N02_Value	1723
PM25_Value	574
VOCs	782
S02	97
dtype:	int64

- pentru test (număr de alorii Nan aparute la atrbute numerice continue)

AQI_Value	182
CO_Value	245
Ozone_Value	237
N02_Value	436
PM25_Value	144
VOCs	199
S02	22
dtype:	int64

Al doilea pas: Acum că avem niste valori lipsă în locul outlierilor, trebuie să le imputăm. Eu am ales să merg pe varianta de SimpleImputer (imputare univariată), care înlocuiește valorile lipsă cu valoarea medianei. În urma imputării, am verificat să nu mai avem nicio valoare lipsă, iar dacă ne uităm la o histogramă pentru un atribut la întâmplare, de exemplu, pentru Ozone\_Value, observăm diferența dintre histograma de dinainte de prelucrarea outlierilor (în care singura prelucrare pe care o facusem asupra setului de date fusese imputarea valorilor lipsă) (albastru) și cea de după înlocuirea outlierilor (galben). Distribuția albastră se întinde mai mult spre dreapta, spre valori mai mari, ceea ce indică prezenta valorilor extreme. Acestea trag media și deviația standardă în sus. În același timp, cea histogramă galbenă este mai compactă și are un varf clar, acest spike în jurul unei singure valori daturându-se faptului

ca outlierii au fost inlocuiti cu valoarea mediana. De asemenea, la distributia galbena este de remarcat faptul ca frecvențele sunt mai grupate, ceea ce inseamna ca modelul va avea mai putine dificultati in a invata.



#### 4. Plaje valorice de marimi diferite pentru attributele numerice

Dupa cum am dedus si din box plot-ul de la analiza datelor, unele attribute numerice pot avea scale diferite, ceea ce ar putea afecta modelele de invatare in sensul ca ar fi puternic dominate de atributul cu valori numerice foarte mari, motiv pentru care se face standardizarea valorilor atributelor numerice.

In urma standardizarii, valorile de pe fiecare coloana ar trebui sa fi fost transformate in asa fel incat pe fiecare coloana media sa fie aproximativ egala cu 0 si deviatia standard aproximativ egala cu 1. Acest lucru se respecta si in cazul setului de train, dupa cum se observa in imaginea de mai jos

	<b>Media (aprox. 0)</b>	<b>Deviația standard (aprox. 1)</b>
AQI_Value	-1.097802e-16	1.000027
CO_Value	-1.025877e-16	1.000027
Ozone_Value	-1.287078e-17	1.000027
NO2_Value	1.059947e-17	1.000027
SO2	5.186167e-17	1.000027

, dar si in cazul setului de test:

	<b>Media (aprox. 0)</b>	<b>Deviația standard (aprox. 1)</b>
AQI_Value	-1.718445e-16	1.000107
CO_Value	-5.601978e-17	1.000107
Ozone_Value	-9.689907e-17	1.000107
NO2_Value	-1.722230e-17	1.000107
SO2	-7.267430e-17	1.000107

### 3. Utilizarea algoritmilor de invatare automata

Inainte de toate, am a trebuit sa convertesc variabilele categoriale intr-o forma numerica, astfel ca variabila tinta (AQI\_Category) a fost encodata folosind LabelEncoder, iar restul variabilelor categorice au fost encodeate folosind OneHotEncoder. Diferenta dintre cele doua este urmatoarea: LabelEncoder atribuie o valoare numerica fiecarei categorii a atributului (ex: Good - 0, Moderate - 1, Hazardous - 2, etc), iar peste aceste coduri care sunt numere intregi se pot aplica relatii de ordine, pe cand OneHotEncoder-ul atribuie fiecarei categorii a unui atribut un vector de valori 0 / 1 (valoarea 1 se afla doar pe pozitia corespunzatoare categoriei respective, in rest vectorul e plin doar cu 0-uri).

Metrici pentru analiza performantei unui algoritm de clasificare ([https://en.wikipedia.org/wiki/Precision\\_and\\_recall](https://en.wikipedia.org/wiki/Precision_and_recall)) :

- precision: dintre toate predictiile pozitive pe care le-a facut, cate sunt corecte?

$$\text{precision} = \text{TruePositive} / (\text{TruePositive} + \text{FalsePositive})$$

- recall: cât de multe dintre exemplele reale pozitive au fost găsite de model

$$\text{recall} = \text{TruePositive} / (\text{TruePositive} + \text{FalseNegative})$$

- f1-score: media armonica intre precision si recall
- support: Numărul de exemple reale din setul de test care aparțin acelei clase.

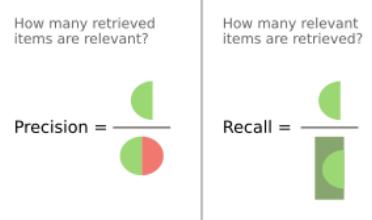
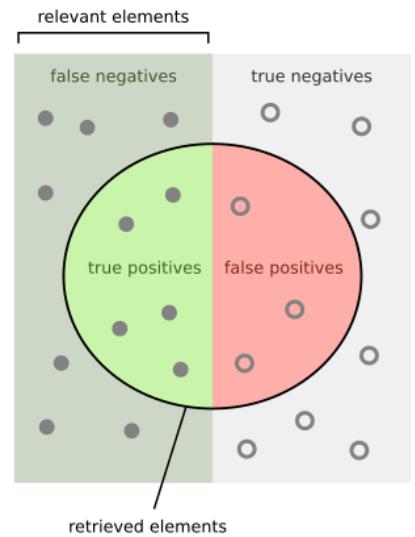
Metrici globale:

- **accuracy**: Procentul de predicții corecte din totalul exemplelor. (Atentie!: poate fi inselatoare daca avem clase dezechilibrate)
- macro avg: Media aritmetică simplă a metricilor (precision, recall, F1) pentru toate clasele, tratând fiecare clasă egal (utilă daca ne interesează performanța pentru toate clasele în mod egal)
- weighted avg: Media ponderată cu support (numărul de exemple per clasă). (ilustrează scorul total generat al modelului luând în considerare dimensiunea claselor)

## 1. Decision Tree

Am pornit de la parametrii default:

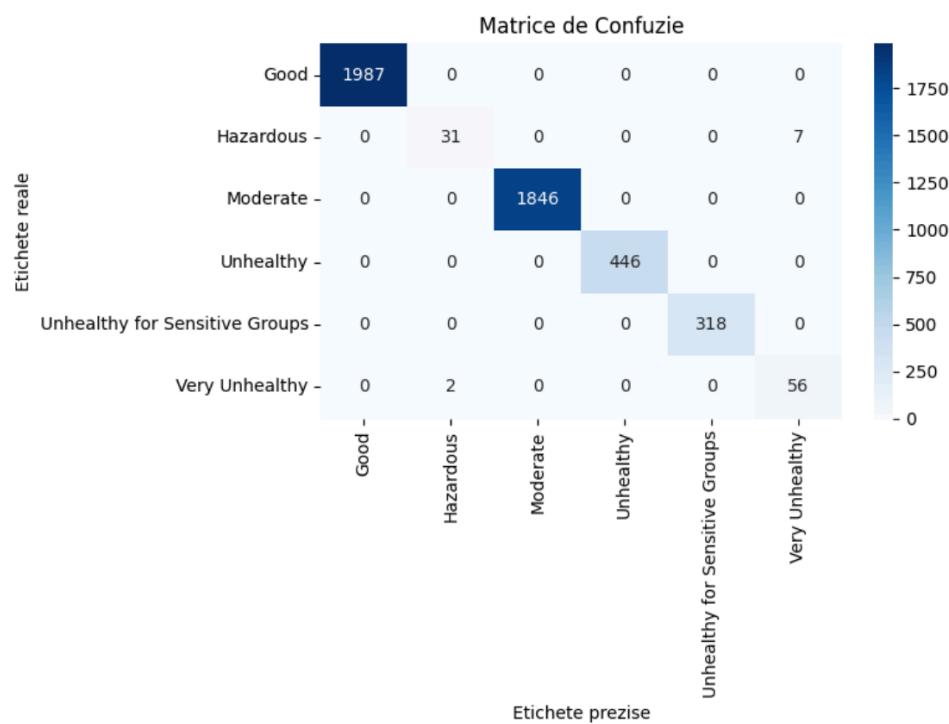
```
tree_model = DecisionTreeClassifier(  
    criterion="gini",  
    max_depth=None,  
    min_samples_leaf=2,  
    class_weight=None,  
    random_state=42
```



)

	precision	recall	f1-score
Good	1.0000	1.0000	1.0000
Hazardous	0.9394	0.8158	0.8732
Moderate	1.0000	1.0000	1.0000
Unhealthy	1.0000	1.0000	1.0000
Unhealthy for Sensitive Groups	1.0000	1.0000	1.0000
Very Unhealthy	0.8889	0.9655	0.9256

Acuratețea generală (fără aproximare): 0.9980822501598124



Observatii: Pentru clasele majore (Good, Moderate, Unhealthy, Unhealthy for sensitive groups), performanta este perfecta, deoarece atat precizia, cat si recall-ul au valoarea 1, ceea ce face ca si f1-score-ul sa aiba valoarea 1. Acestea sunt si clasele care au cel mai mare numar de exemple ( fiecare din aceste clase are un numar de exemple > 300), iar modelul a invatat sa recunoasca foarte bine aceste clase).

Totusi, cand vine vorba despre clasele mai rare (Hazardous, Very Unhealthy), f1-scoare-ul este mai mic, respectiv 0.87 si 0.93, caci acestea nu au fost invatate la fel de bine, neavand un numar asa de mare de exemple. Cu toate acestea, f1-score-urile sunt destul de bune.

Overall, acuratetea modelului este de aproximativ 0.9981, adica foarte apropiata de 1, ceea ce indica o calitate foarte buna.

Matricea de confuzie compara etichetele reale (linii) cu etichetele prezise de model (coloane). Astfel, se observa faptul ca din cele 38 de exemple reale Hazardous, 31 au fost corect clasificate, iar restul de 7 au fost prezise gresit ca Very Unhealthy. Similar, din cele 58

de exemple reale Very Unhealthy, 56 au fost clasificate corect, iar 2 au fost prezise gresit ca hazardous. Pentru restul claselor, toate exemplele au fost prezise corect.

### Modificarea parametrilor pentru DecisionTree

- **max\_depth:** - daca lasam max\_depth-ul None, arborele creste pana cand toate nodurile sunt pure sau nu mai exista exemple de impartit, ceea ce duce la un model foarte complex, care invata si zgromotul din date. Daca setam max\_depth la 10, atunci arborele va fi forțat să se opreasca după 10 niveluri, ceea ce va duce la un model mai simplu, mai general, intrucât nu va mai invata detaliile inutile, adică nu memorează excepțiile din setul de antrenament, ci găsește tipare reale. Cu alte cuvinte, prin setarea max\_depth-ului se poate evita overfitting-ul.
- creșterea numărului minim de exemple per frunză (min\_samples\_leaf) de la 2 la 3: Cu cat **min\_samples\_leaf** este mai mic, cu atât este mai probabil să apară overfitting-ul, pentru că am avea frunze cu 1 sau 2 exemple, ceea ce ar însemna că modelul poate crede că un tipar este important doar pentru că apare o singură dată.

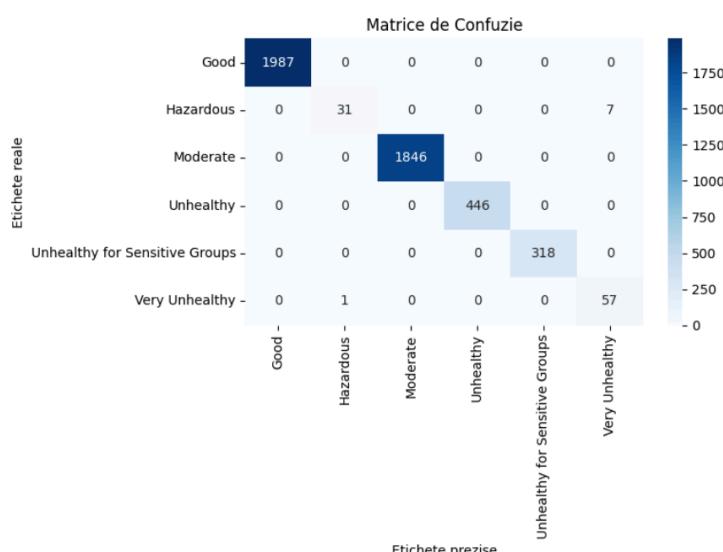
```
tree_model = DecisionTreeClassifier(
    criterion="gini",
    max_depth=10,
    min_samples_leaf=3,
    class_weight=None,
    random_state=42
)


|                                | precision | recall | F1-score |
|--------------------------------|-----------|--------|----------|
| Good                           | 1.0000    | 1.0000 | 1.0000   |
| Hazardous                      | 0.9688    | 0.8158 | 0.8857   |
| Moderate                       | 1.0000    | 1.0000 | 1.0000   |
| Unhealthy                      | 1.0000    | 1.0000 | 1.0000   |
| Unhealthy for Sensitive Groups | 1.0000    | 1.0000 | 1.0000   |
| Very Unhealthy                 | 0.8906    | 0.9828 | 0.9344   |



Acuratețea generală (fără aproximare): 0.9982953334753889


```



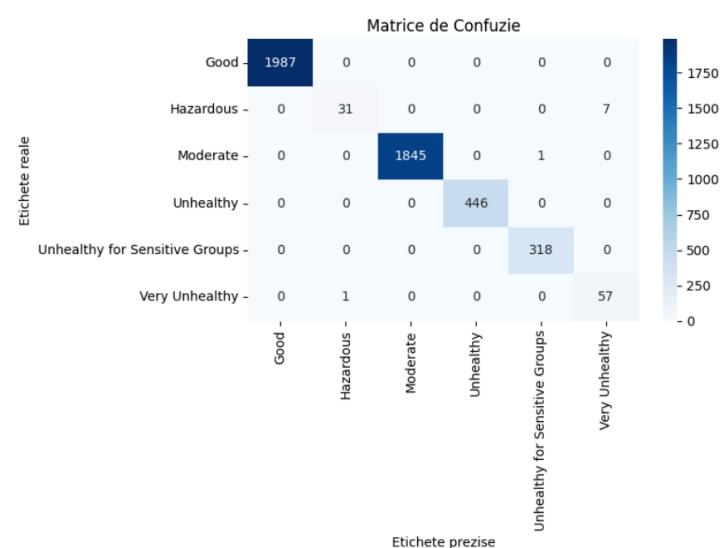
Deja observam ca accuracy-ul a crescut de la 0.9981 la 0.9983, iar daca ne uitam la matricea de confuzie, observam ca modelul reuseste sa clasifice corect cu un exemplu in plus in clasa Very Unhealthy, care este una dintre clasele rare, mai greu de invatat pentru ca avem un numar mult mai mic de exemple (arbore dezechilibrat).

- **class\_weight = “balanced”**: pentru astfel de arbori dezechilibrati, este recomandata setarea parametrului `class_weight = “balanced”`, care reechilibreaza greutatile claselor invers proportional cu frecventa lor in setul de antrenare, mai exact va penaliza mai mult erorile pe clasele rare pentru a preveni ignorarea completa a claselor rare

```
tree_model = DecisionTreeClassifier(
    criterion="gini",
    max_depth=10,
    min_samples_leaf=3,
    class_weight="balanced",
    random_state=42
)
```

	precision	recall	F1-score
Good	1.0000	1.0000	1.0000
Hazardous	0.9688	0.8158	0.8857
Moderate	1.0000	0.9995	0.9997
Unhealthy	1.0000	1.0000	1.0000
Unhealthy for Sensitive Groups	0.9969	1.0000	0.9984
Very Unhealthy	0.8906	0.9828	0.9344

Acuratețea generală (fără aproximare): 0.9980822501598124



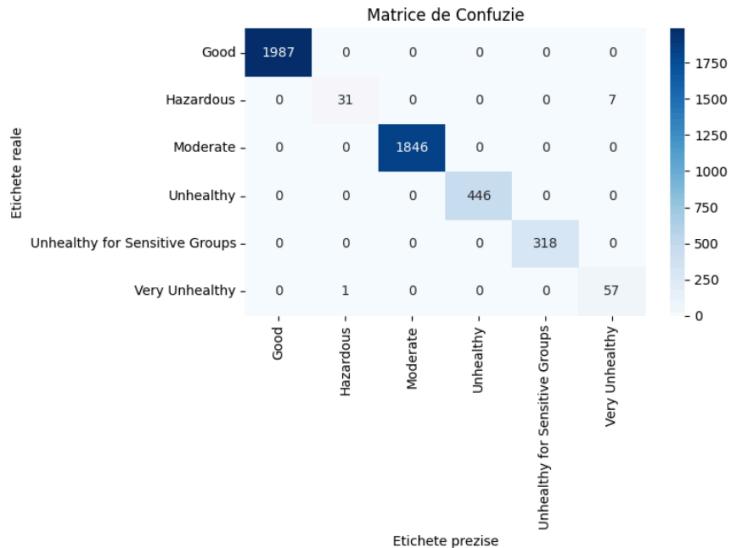
In urma schimbarii parametrului class\_weight din None in balanced, observam ca accuracy-ul a scăzut de la 0.9983 la 0.9981. Totuși, acest lucru nu înseamnă neapărat un model mai slab. Accuracy-ul va scădea pentru că nu se vor mai favoriza clasele frecvente și se va încerca să prezicerea claselor rare, chiar dacă sunt mai puține în test. Astfel, acest parametru balanced este util atunci când avem clase dezechilibrate, pentru că evită stabilirea unui model "partinito", încercând să asigure "echitatea" între clase. De exemplu, dacă să zicem că avem 90% de exemple din clasa Good și 10% din celelalte clase. Un model care prezice doar Good va avea o acuratețe de 90%, dar va avea precision, recall și f1-score = 0 pentru celelalte clase, în timp ce un model care încearcă să prezică și celelalte clase va avea acuratețe mai mici, dar va imbunătăți semnificativ scorurile pe clasele mai rare, chiar dacă va sacrifica puțin din precizia pe Good.

- criterion = "entropy": Entropy e mai atent la cat de "pure" sunt nodurile și poate identifica mai bine un split relevant pentru o clasa rara, deci se folosește atunci când avem clase dezechilibrate. Pe de alta parte, gini este mai rapid, deci e indicat să se folosească atunci când avem seturi mari de date, în care clasele sunt relativ echilibrate, pentru a asigura o viteza mai mare la antrenare.

```
tree_model = DecisionTreeClassifier(
    criterion="entropy",
    max_depth=10,
    min_samples_leaf=3,
    class_weight="balanced",
    random_state=42
)
```

		<b>precision</b>	<b>recall</b>	<b>F1-score</b>
	Good	<b>1.0000</b>	<b>1.0000</b>	<b>1.0000</b>
	Hazardous	0.9688	0.8158	0.8857
	Moderate	<b>1.0000</b>	<b>1.0000</b>	<b>1.0000</b>
	Unhealthy	<b>1.0000</b>	<b>1.0000</b>	<b>1.0000</b>
	Unhealthy for Sensitive Groups	<b>1.0000</b>	<b>1.0000</b>	<b>1.0000</b>
	Very Unhealthy	0.8906	0.9828	0.9344

Acuratețea generală (fără aproximare): 0.9982953334753889



Accuracy-ul a crescut din nou la 0.9983, iar scorurile vor fi egale cu cele de la prima modificare de parametri.

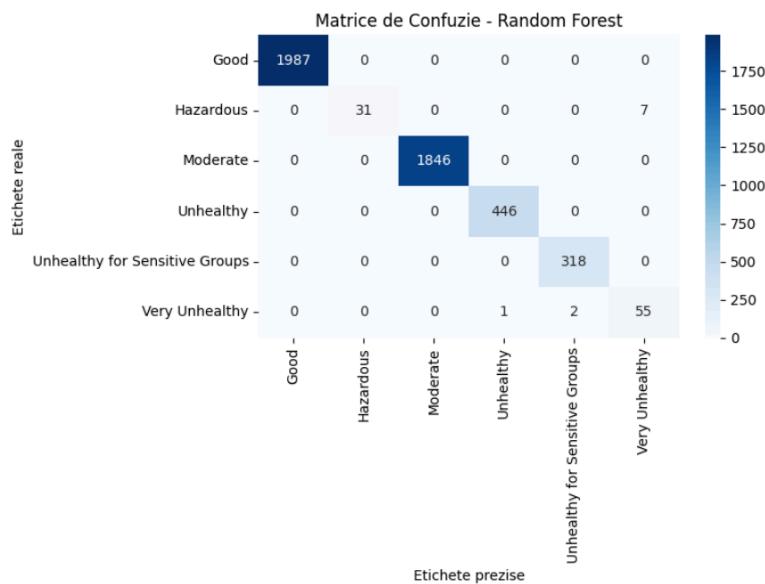
## 2. Random forest

- apelat cu parametrii default:

```
rf_model = RandomForestClassifier(
    n_estimators=100,
    criterion="gini",
    max_depth=None,
    min_samples_leaf=2,
    class_weight=None,
    max_samples=None,
    max_features="sqrt",
    random_state=42,
    n_jobs=None
)
```

	precision	recall	f1-score
Good	<b>1.0000</b>	<b>1.0000</b>	<b>1.0000</b>
Hazardous	<b>1.0000</b>	0.8158	0.8986
Moderate	<b>1.0000</b>	<b>1.0000</b>	<b>1.0000</b>
Unhealthy	0.9978	<b>1.0000</b>	0.9989
Unhealthy for Sensitive Groups	0.9938	<b>1.0000</b>	0.9969
Very Unhealthy	0.8871	0.9483	0.9167
macro avg	0.9798	0.9607	0.9685
weighted avg	0.9980	0.9979	0.9978

Acuratetea generală (fără aproximație): 0.9978691668442361



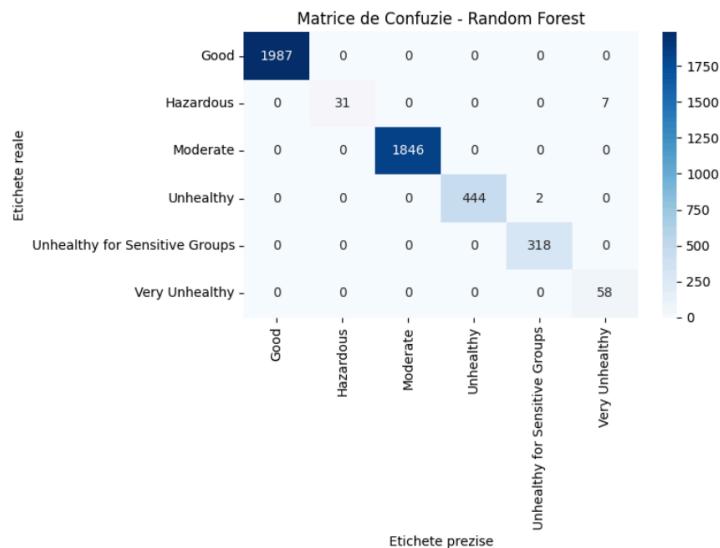
Uitandu-ne la rezultatele de mai sus si urmarind coloana f1-score, se poate observa ca modelul prezice perfect clasele majore (Good si Moderate, care au f1-score = 1), dar are score-uri mai slabe pentru clasele mai mici (cu support mai mic) (avem f1-score = 0.8986 pentru Hazardous si 0.9167 pentru Very Unhealthy).

Pentru a rezolva aceasta problema generata de clasele dezechilibrate si a neputintei modelului de a invata clasele rare, facem urmatoarea modificare: setam parametrul **class\_weight = "balanced"**.

```
rf_model = RandomForestClassifier()
n_estimators=100,
criterion="gini",
max_depth=None,
min_samples_leaf=2,
class_weight="balanced",
max_samples=None,
max_features="sqrt",
random_state=42,
n_jobs=None
)
```

	precision	recall	f1-score
Good	1.0000	1.0000	1.0000
Hazardous	1.0000	0.8158	0.8986
Moderate	1.0000	1.0000	1.0000
Unhealthy	1.0000	0.9955	0.9978
Unhealthy for Sensitive Groups	0.9938	1.0000	0.9969
Very Unhealthy	0.8923	1.0000	0.9431
macro avg	0.9810	0.9686	0.9727
weighted avg	0.9982	0.9981	0.9980

Acuratetea generală (fără aproximatie): 0.9980822501598124



Setarea parametrului `class_weight = "balanced"` modifica importanta relativa a fiecarei clase in timpul antrenarii modelului.

Cum noi avem clase dezechilibrate, este foarte important sa folosim acest parametru setat pe `"balanced"`. Astfel, nu numai ca se observa o crestere a accuracy-ului de la 0.99787 la 0.99808, ci si o crestere a f1-score-ului clasei rare `Very Unhealthy` de la 0.9167 la 0.9431. De asemenea, se poate observa si modificarea matricei de confuzie, indicand faptul ca acum si clasele mai rare au fost prezise preponderent corect.

Pentru a evita overfitting-ul se pot face unele modificari de parametrii, pe care le voi prezenta mai jos, chiar daca in cazul de fata nu poate fi vorba despre overfitting pentru ca noi avem acuratetea foarte mare pe setul de test. Overfitting ar fi fost daca aveam acuratetea mare pe setul de train si mica pe setul de test.

- Impunerea unui `max_depth = 10`

Limitarea adancimii maxime a arborilor de decizie este, dupa cum am explicitat si anterior, o masura de preventire a overfitting-ului. Fara aceasta, arborii pot continua sa se ramifice pana clasifica perfect datele de antrenament (inclusiv zgromotul sau exceptiile), deci pentru a obtine arbori mai generali, care invata modele reale, nu doar cazuri particulare, este indicat sa se foloseasca aceasta limita de adancime. Astfel, modelul este ajutat sa generalizeze mai bine. Totusi, acuratetea overall va scadea pentru ca daca se ajunge la nivelul maxim, chiar daca entropia e inca mare (clasificarea nu e clara), arborele acolo se opreste.

- `max_samples = 0.8`

Bootstrapping: fiecare arbore din Random Forest este antrenat doar pe un eșantion din datele de antrenament (cu repetare). Inainte aveam `max_samples = None`, ceea ce insemanea ca fiecare arbore este antrenat pe toate datele. Cu `max_samples = 0.8`, fiecare arbore este antrenat doar pe 80% din datele de antrenament, selectate aleator (și cu repetare). Astfel, fiecare arbore vede un subset diferit de date, ceea ce scade corelatia dintre arbori, deci padurea devine mai generalizabila, ceea ce reduce overfitting-ul.

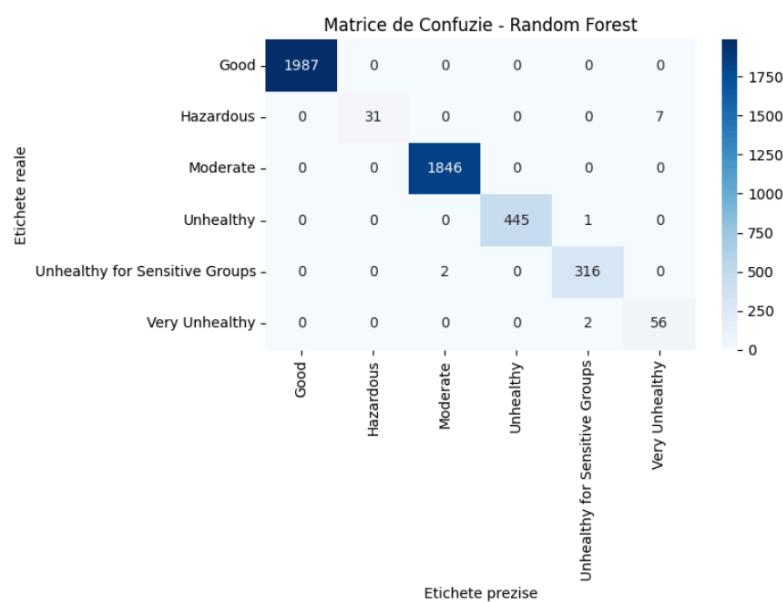
```

rf_model = RandomForestClassifier(
    n_estimators=100,
    criterion="gini",
    max_depth=10,
    min_samples_leaf=2,
    class_weight="balanced",
    max_samples=0.8,
    max_features="sqrt",
    random_state=42,
    n_jobs=None
)

```

		precision	recall	f1-score
	Good	<b>1.0000</b>	<b>1.0000</b>	<b>1.0000</b>
	Hazardous	<b>1.0000</b>	0.8158	0.8986
	Moderate	0.9989	<b>1.0000</b>	0.9995
	Unhealthy	<b>1.0000</b>	0.9978	0.9989
	Unhealthy for Sensitive Groups	0.9906	0.9937	0.9922
	Very Unhealthy	0.8889	0.9655	0.9256
	macro avg	0.9797	0.9621	0.9691
	weighted avg	0.9976	0.9974	0.9974

Acuratețea generală (fără aproximatie): 0.9974430002130833



Dupa cum ne asteptam ca se va intampla in urma introducerii acestor masuri care cresc gradul de generalizare, acuratetea a scazut de la 0.99808 la 0.99744, iar f1-score-urile in general au scazut (in afara de cel al clasei dominante Good, care a ramas in continuarea 1, adica perfect).

## Regresie logistica

Task-ul principal in implementarea acestui algoritm a constat in modificarea codului de la laborator, in care variabila tinta avea doar 2 valori posibile (de obicei yes sau no), in asa fel incat poate fi adaptat pentru o variabila tinta multi-class, in cazul nostru AQI\_Category putand avea mai mult de 2 valori: Good, Moderate, Hazardous, Very Unhealthy, etc. Pentru a efectua modificarile necesare mi-au fost utile urmatoarele 2 link-uri:

<https://medium.com/@jshaik2452/multi-class-logistic-regression-a-friendly-guide-to-classifying-the-many-4a590c2e6c26>

<https://youtu.be/hYBwBmojXoU?feature=shared>

In primul rand, este foarte importanta codarea: variabila tinta a fost codata cu LabelEncoder, iar variabilele categoriale de predictie au fost codate cu OneHotEncoder.

In al doilea rand, am inlocuit functia sigmoid cu softmax, care se calculeaza dupa urmatoarea formula pentru a ne adapta la multi-class:

$$P(y = k) = \frac{e^{z_k}}{\sum_j e^{z_j}}$$

Ca functie de loss, am folosit nll (Negative Log Likelihood), care trebuie minimizata. Atunci cand plotam graficul nll, ar trebui sa observam cum scade pe masura ce modelul devine mai antrenat.

### Optimisation

- Fit model using maximum likelihood. Equivalent to minimising the negative log likelihood:

$$\begin{aligned} J(\mathbf{W}) &= -\log L(\mathbf{W}) \\ &= -\sum_{n=1}^N \log P(y^{(n)} | \mathbf{x}^{(n)}; \mathbf{W}) \\ &= -\sum_{n=1}^N \sum_{k=1}^K \mathbb{I}\{y^{(n)} = k\} \log \frac{e^{\mathbf{w}_k^\top \mathbf{x}^{(n)}}}{\sum_{j=1}^K e^{\mathbf{w}_j^\top \mathbf{x}^{(n)}}} \end{aligned}$$

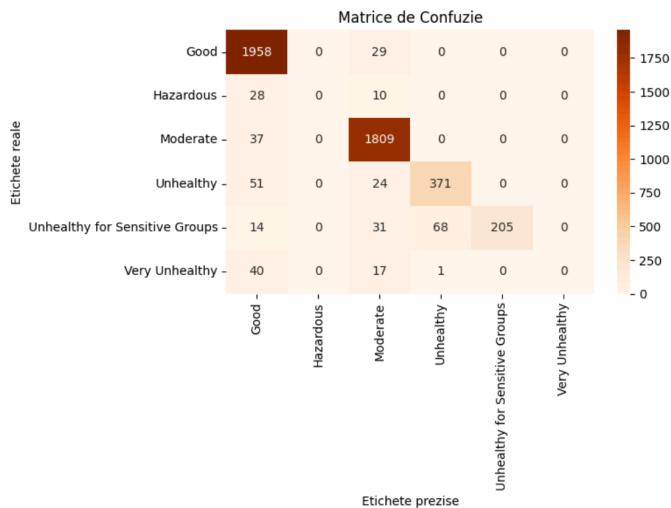
- Derivatives:

$$\frac{\partial J(\mathbf{W})}{\partial \mathbf{w}_k} = -\sum_{n=1}^N (\mathbb{I}\{y^{(n)} = k\} - f_k(\mathbf{x}^{(n)}; \mathbf{w}_k)) \mathbf{x}^{(n)}$$

- Using these derivatives, we can minimise the loss using gradient descent.

Felul in care arata acest grafic ne indica si cat de potrivit este learning rate-ul ales. Dupa cum am spus mai sus, functia nll ar trebui sa fie o curba care scade cu cat modelul invata mai mult. Daca la un moment dat vedem ca aceasta curba o ia in sus, inseamna ca am ales un learning rate prea mare.

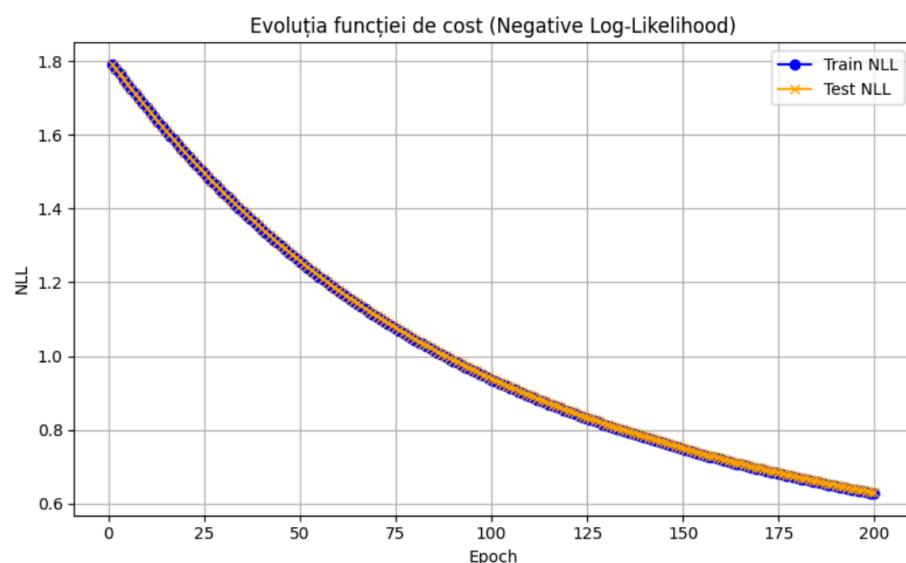
Eu am inceput cu 200 de epoci si un learning rate de 0.02 si am obtinut o acuratete de 0.9254, iar rezultatele au fost urmatoarele:



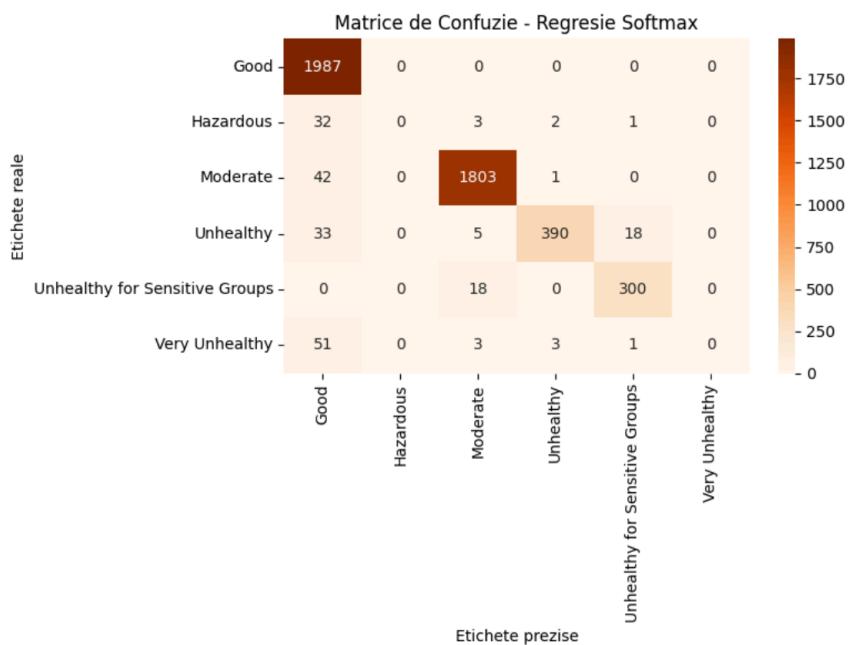
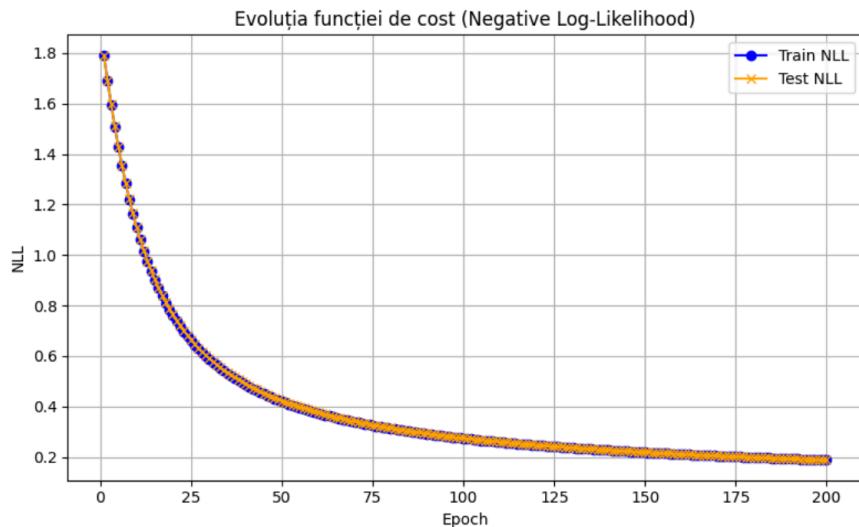
	precision	recall	f1-score
Good	0.9201	<b>0.9854</b>	0.9516
Hazardous	0.0000	0.0000	0.0000
Moderate	0.9422	0.9800	<b>0.9607</b>
Unhealthy	0.8432	0.8318	0.8375
Unhealthy for Sensitive Groups	<b>1.0000</b>	0.6447	0.7839
Very Unhealthy	0.0000	0.0000	0.0000
macro avg	0.6176	0.5736	0.5890
weighted avg	0.9081	0.9254	0.9135

Acuratețea generală (fără aproximație): 0.9254208395482634

Pentru lr = 0.02, functia de loss arata asa:



Am incercat apoi sa cresc treptat learning rate-ul până am ajuns la 0.15, unde am vazut ca se cam plafoneaza. La aceasta valoare de 0.15 curba functiei de loss este mai abrupta, semn ca invatarea are loc intr-un ritm mai rapid, iar acuratetea a crescut la

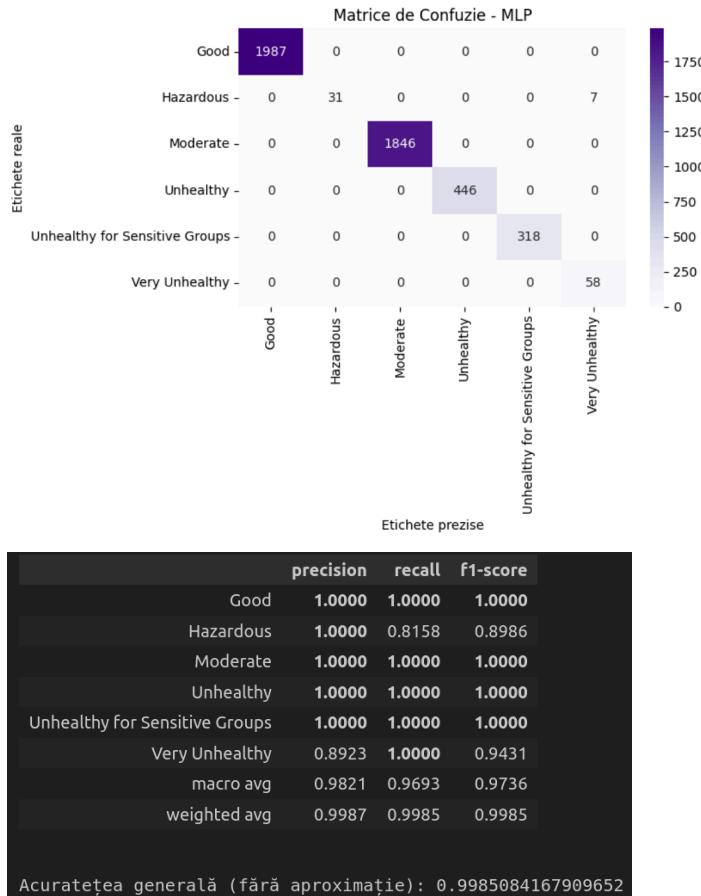


	precision	recall	f1-score
Good	0.9263	<b>1.0000</b>	0.9618
Hazardous	0.0000	0.0000	0.0000
Moderate	0.9842	0.9767	<b>0.9804</b>
Unhealthy	<b>0.9848</b>	0.8744	0.9264
Unhealthy for Sensitive Groups	0.9375	0.9434	0.9404
Very Unhealthy	0.0000	0.0000	0.0000
macro avg	0.6388	0.6324	0.6348
weighted avg	0.9365	0.9546	0.9446

Acuratețea generală (fără aproximatie): 0.9546132537822288

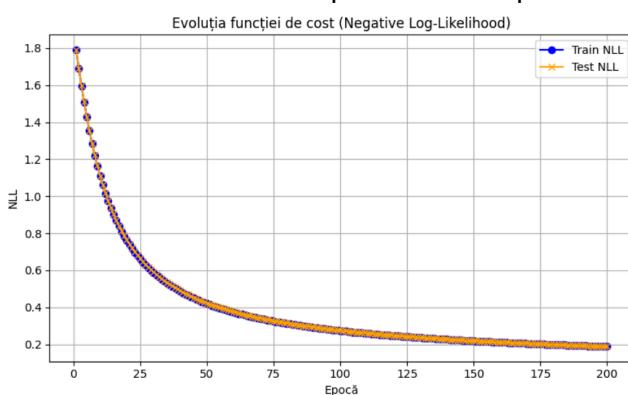
## MLP

- am pornit de la parametrii default si am obtinut din prima o acuratete de 0.9985. Ulterior setat pe True parametrul early\_stopping, care face ca algoritmul sa se opreasca automat dacă performanța pe un set de validare nu se îmbunătășește după un anumit număr de epoci. Desi nu au facut mare diferență în acuratete, timpul de rulare s-a micșorat de la 7.9 s la 3.3s.



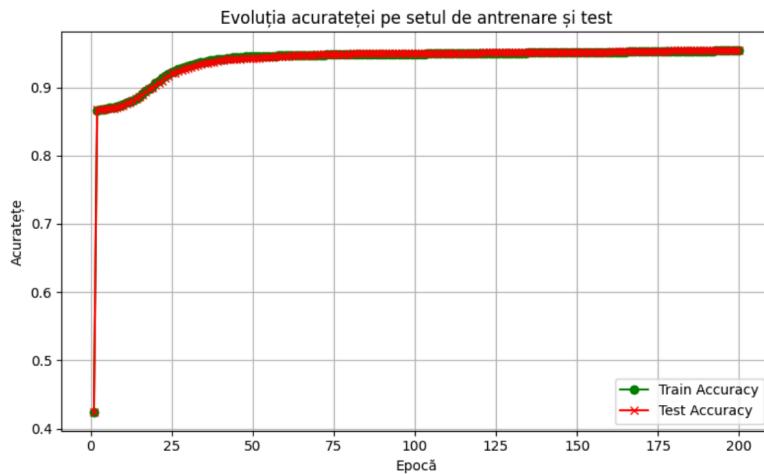
Observam că avem un scor f1 perfect pentru clasele dominante (Good și Moderate), dar și pentru clase care apar mai rar (Unhealthy, Unhealthy for Sensitive Groups). Singurele clase unde rezultatele sunt mai slabe sunt cele rare, putându-se observa în matricea de confuzie că din 38 de etichete hazardous, doar 31 au fost clasificate corect, 7 fiind clasificate ca Very Unhealthy.

Graficele funcției de loss pentru train și pentru test sunt ilustrate în imaginea de mai jos:



Scaderea este destul de abrupta, ceea ce e un semn bun, avem o rata de invatare buna, iar graficele celor doua seturi de date coincid, ceea ce inseamna ca nu avem de-a face cu overfitting. Dupa aproximativ 150 de epoci, scaderea devine mai lenta.

Si cand vine vorba despre evolutia acuratetei pe setul de antrenare si test, cele doua curbe aproape ca se suprapun, ceea ce inseamna ca generalizarea este buna, iar overfitting-ul este aproape inexistent. Cresterea este mai brusca la inceput, ceea ce inseamna ca modelul invata repede din date, iar apoi, dupa aproximativ 50 de epoci, curbele se stabilizeaza, intrucat modelul atinge limita capacitatii sale pe datele respective.



Comparatie intre cei 4 algoritmi:

- Decision tree: are o acuratete aproape perfecta (0.9983) si scoruri f1 1.00 aproape la toate atributele in afara de cele rare. Poate memoriza perfect setul de date, dar este sensibil la overfitting. Din fericire, nu este cazul de asa ceva la setul nostru de date.
- Random Forest: si acesta are o acuratete foarte buna (0.99808), chiar daca este putin mai slaba decat la Decision Tree, Random Forest este mai robust decat Decision Tree, pentru ca vine cu un plus de generalizare. Si acesta a facut cateva greseli tot la clasele rare, dar incercă sa asigure un echilibru intre clase (in special prin setarea parametrului class\_weight la "balanced")
- Logistic Regression: are cea mai mica acuratete dintre toti (0.9546), dar cel mai grav este ca are f1-scores foarte scazute pentru clasele rare (Hazardous si Very Unhealthy).
- MLP: are cea mai buna acuratete dintre toti (0.9985), si f1-score-uri foarte bune, demonstrand o performanta ridicata chiar si pe clasele rare. Dezavantajul sau, insa, este ca are un cost de antrenare mai mare decat arborii.

Tabelul comparativ intre algoritmi se afla in fisierul part1\_air\_pollution.ipynb.

## II. News Popularity

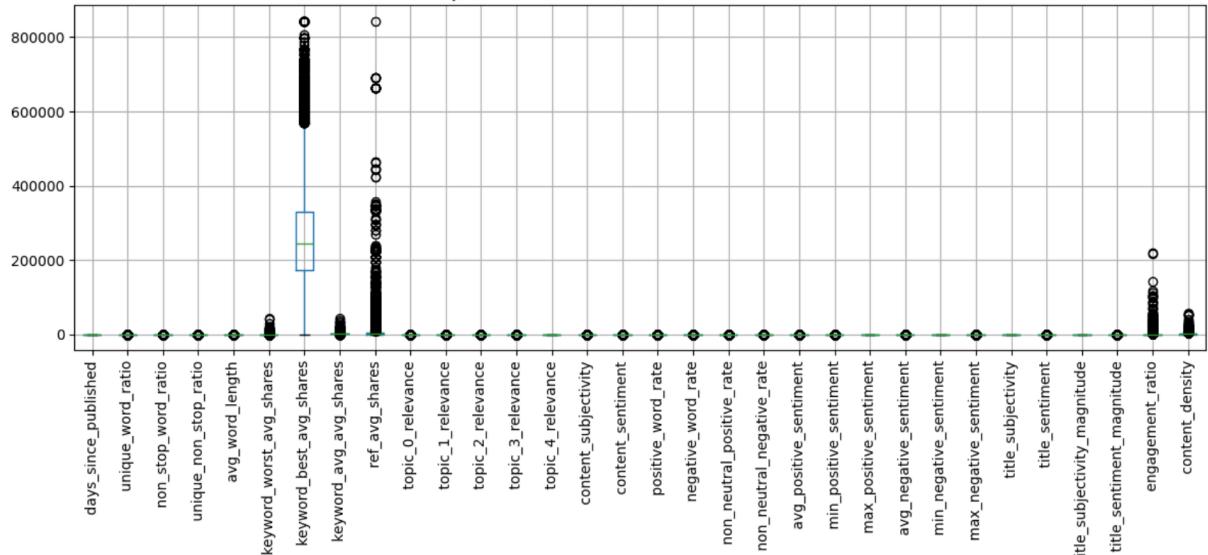
1. Explorarea Datelor (Exploratory Data Analysis)
  - a. Analiza tipului de atributie si a plajei de valori a acestora

## Atribute numerice continue

- Statistici in setul de train:

	count	mean	std	min	25%	50%	75%	max
days_since_published	31715.0	354.837900	214.004340	8.000000	164.000000	339.000000	542.000000	731.000000
unique_word_ratio	31715.0	0.552400	3.935710	0.000000	0.470733	0.539062	0.608857	701.000000
non_stop_word_ratio	31715.0	1.002396	5.848149	0.000000	1.000000	1.000000	1.000000	1042.000000
unique_non_stop_ratio	31715.0	0.692922	3.649410	0.000000	0.625720	0.690554	0.754839	650.000000
avg_word_length	31715.0	4.546125	0.852537	0.000000	4.478261	4.664962	4.855670	8.041534
keyword_worst_avg_shares	31715.0	310.189285	592.178924	-1.000000	142.000000	235.222222	355.800000	42827.857143
keyword_best_avg_shares	31715.0	259713.527912	135812.176630	0.000000	172983.333333	244333.333333	331340.000000	843300.000000
keyword_avg_avg_shares	31715.0	3135.391886	1304.677314	0.000000	2384.380410	2871.904006	3603.698179	43567.659946
ref_avg_shares	31715.0	6371.951353	24145.035816	0.000000	978.775000	2200.000000	5200.000000	843300.000000
topic_0_relevance	31715.0	0.184601	0.263073	0.000000	0.025049	0.033387	0.240609	0.920000
topic_1_relevance	31715.0	0.140735	0.218888	0.000000	0.025012	0.033345	0.150939	0.925947
topic_2_relevance	31715.0	0.217246	0.283001	0.000000	0.028572	0.040004	0.335393	0.919999
topic_3_relevance	31715.0	0.225113	0.295999	0.000000	0.027771	0.040001	0.381367	0.926534
topic_4_relevance	31715.0	0.232273	0.288003	0.000000	0.028574	0.040581	0.397345	0.927191
content_subjectivity	31715.0	0.442996	0.117302	0.000000	0.395991	0.453309	0.508145	1.000000
content_sentiment	31715.0	0.118878	0.097180	-0.393750	0.057353	0.118750	0.177375	0.727841
positive_word_rate	31715.0	0.039600	0.017467	0.000000	0.028348	0.038977	0.050220	0.155488
negative_word_rate	31715.0	0.016616	0.010798	0.000000	0.009615	0.015370	0.021739	0.184932
non_neutral_positive_rate	31715.0	0.681447	0.190774	0.000000	0.600000	0.709677	0.800000	1.000000
non_neutral_negative_rate	31715.0	0.288031	0.156163	0.000000	0.185185	0.280000	0.384615	1.000000
avg_positive_sentiment	31715.0	0.353589	0.104798	0.000000	0.305998	0.358397	0.411024	1.000000
min_positive_sentiment	31715.0	0.095633	0.071728	0.000000	0.050000	0.100000	0.100000	1.000000
max_positive_sentiment	31715.0	0.756480	0.248411	0.000000	0.600000	0.800000	1.000000	1.000000
avg_negative_sentiment	31715.0	-0.259905	0.128557	-1.000000	-0.329027	-0.254074	-0.186806	0.000000
min_negative_sentiment	31715.0	-0.522267	0.290618	-1.000000	-0.700000	-0.500000	-0.300000	0.000000
max_negative_sentiment	31715.0	-0.107628	0.096634	-1.000000	-0.125000	-0.100000	-0.050000	0.000000
title_subjectivity	31715.0	0.282664	0.324494	0.000000	0.000000	0.144444	0.500000	1.000000
title_sentiment	31715.0	0.070873	0.266446	-1.000000	0.000000	0.000000	0.146250	1.000000
title_subjectivity_magnitude	31715.0	0.341824	0.188884	0.000000	0.166667	0.500000	0.500000	0.500000
title_sentiment_magnitude	31715.0	0.156643	0.226890	0.000000	0.000000	0.000000	0.250000	1.000000
engagement_ratio	31715.0	1060.229321	3626.932712	0.041667	220.000000	466.666667	900.000000	221200.000000
content_density	28570.0	1982.199070	2198.508477	32.759785	747.245541	1307.969130	2509.217529	58857.969230

Plajele de valori ale atributelor numerice continue

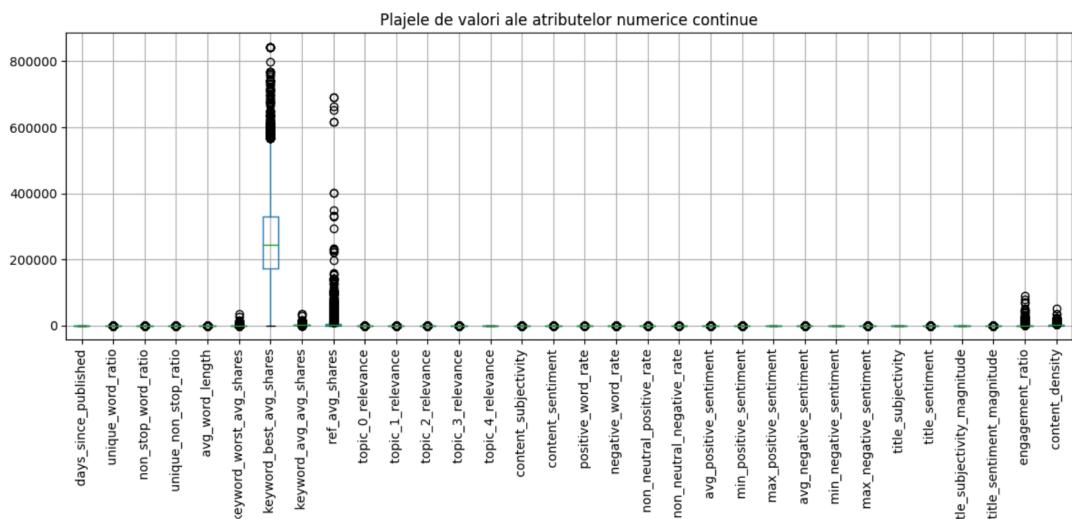


In boxplotul de mai sus pentru setul de train, se observa faptul ca atrubutele 'keyword\_best\_avg\_shares' si 'ref\_avg\_shares' au distributii extreme, cu valori mari, avand foarte multi outlieri. Si la 'engagement\_ratio', 'keyword\_avg\_avg\_shares', 'keyword\_worst\_avg\_shares' si 'content\_density' exista outlieri, dar nu ating valori atat de mari, sunt mai compacti.

De asemenea, de remarcat este si faptul ca avem attribute cu scari complet diferite: majoritatea au valori intre 0 si 1, dar exista cateva cu scari de ordinul zecilor de mii. Asadar, este evident ca va trebui sa folosim scalarea pentru a aduce toate valorile pe aceeasi scara.

- Statistici in setul de test:

	count	mean	std	min	25%	50%	75%	max
days_since_published	7929.0	353.300795	214.809398	8.000000	164.000000	336.000000	543.000000	731.000000
unique_word_ratio	7929.0	0.531480	0.134536	0.000000	0.471449	0.539623	0.607692	0.979592
non_stop_word_ratio	7929.0	0.972758	0.162797	0.000000	1.000000	1.000000	1.000000	1.000000
unique_non_stop_ratio	7929.0	0.674190	0.150791	0.000000	0.625899	0.690162	0.753846	1.000000
avg_word_length	7929.0	4.556697	0.811063	0.000000	4.478652	4.660000	4.850877	6.816754
keyword_worst_avg_shares	7929.0	321.077421	723.962824	-1.000000	139.750000	236.750000	362.142857	34855.125000
keyword_best_avg_shares	7929.0	257555.633196	132218.852468	0.000000	172230.000000	245700.000000	329760.000000	843300.000000
keyword_avg_avg_shares	7929.0	3137.725591	1370.800677	0.000000	2373.128396	2865.091208	3593.588690	36717.233112
ref_avg_shares	7929.0	6520.678737	24475.887596	0.000000	995.000000	2200.000000	5177.000000	690400.000000
topic_0_relevance	7929.0	0.184591	0.262597	0.018279	0.025057	0.033389	0.242588	0.926994
topic_1_relevance	7929.0	0.143340	0.222956	0.018182	0.025016	0.033347	0.150282	0.919976
topic_2_relevance	7929.0	0.212620	0.278681	0.018182	0.025206	0.040002	0.324037	0.919999
topic_3_relevance	7929.0	0.218396	0.291893	0.018183	0.028571	0.040001	0.346937	0.919975
topic_4_relevance	7929.0	0.241053	0.293770	0.018183	0.028574	0.050000	0.420300	0.927119
content_subjectivity	7929.0	0.444867	0.114176	0.000000	0.396875	0.454167	0.508854	1.000000
content_sentiment	7929.0	0.121033	0.059512	-0.380208	0.060000	0.120495	0.179765	0.631746
positive_word_rate	7929.0	0.039723	0.017276	0.000000	0.028617	0.039185	0.050588	0.133080
negative_word_rate	7929.0	0.016598	0.010948	0.000000	0.009615	0.015244	0.021739	0.139831
non_neutral_positive_rate	7929.0	0.684962	0.187902	0.000000	0.600000	0.714286	0.800000	1.000000
non_neutral_negative_rate	7929.0	0.287544	0.156138	0.000000	0.185185	0.277778	0.380282	1.000000
avg_positive_sentiment	7929.0	0.354768	0.103514	0.000000	0.307308	0.360134	0.412245	1.000000
min_positive_sentiment	7929.0	0.094696	0.069639	0.000000	0.050000	0.100000	0.100000	1.000000
max_positive_sentiment	7929.0	0.757719	0.245284	0.000000	0.600000	0.800000	1.000000	1.000000
avg_negative_sentiment	7929.0	-0.257999	0.124340	-1.000000	-0.325556	-0.250758	-0.187143	0.000000
min_negative_sentiment	7929.0	-0.520651	0.288985	-1.000000	-0.700000	-0.500000	-0.300000	0.000000
max_negative_sentiment	7929.0	-0.106988	0.090157	-1.000000	-0.125000	-0.100000	-0.050000	0.000000
title_subjectivity	7929.0	0.281108	0.323275	0.000000	0.000000	0.150000	0.500000	1.000000
title_sentiment	7929.0	0.073634	0.261435	-1.000000	0.000000	0.000000	0.150000	1.000000
title_subjectivity_magnitude	7929.0	0.341916	0.188429	0.000000	0.166667	0.500000	0.500000	0.500000
title_sentiment_magnitude	7929.0	0.153747	0.223896	0.000000	0.000000	0.000000	0.250000	1.000000
engagement_ratio	7929.0	1029.415076	2917.745785	2.500000	220.000000	455.000000	900.000000	90300.000000
content_density	7110.0	2004.082604	2251.238165	36.127614	741.646747	1338.484636	2501.696629	52383.047915



Aceleasi observatii ca la train sunt valabile si pentru boxplotul de test.

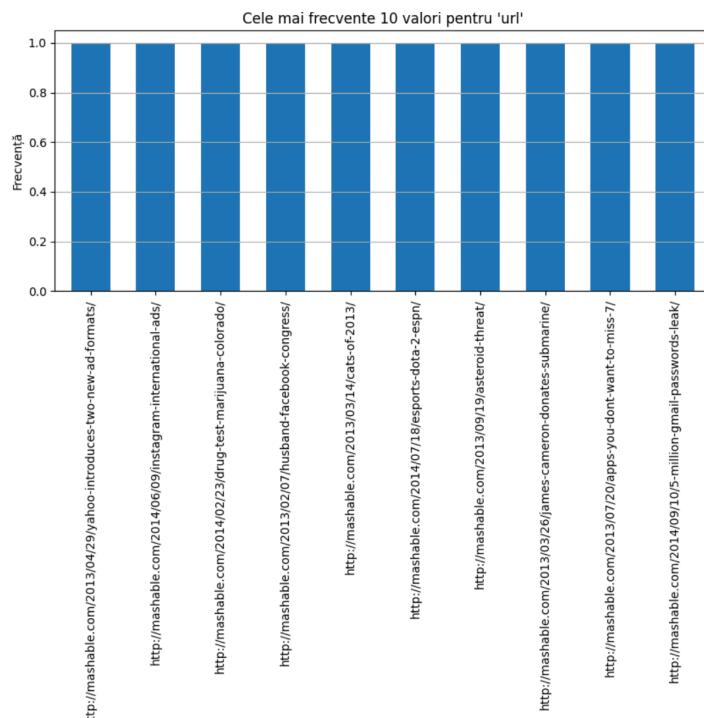
### Atributele discrete sau ordinale

- pentru setul de train:

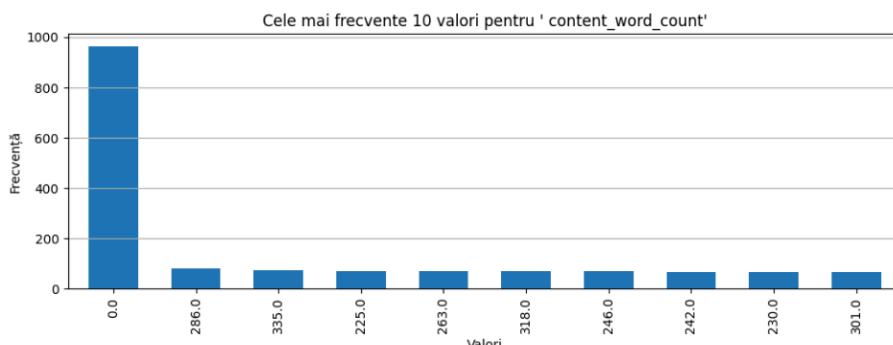
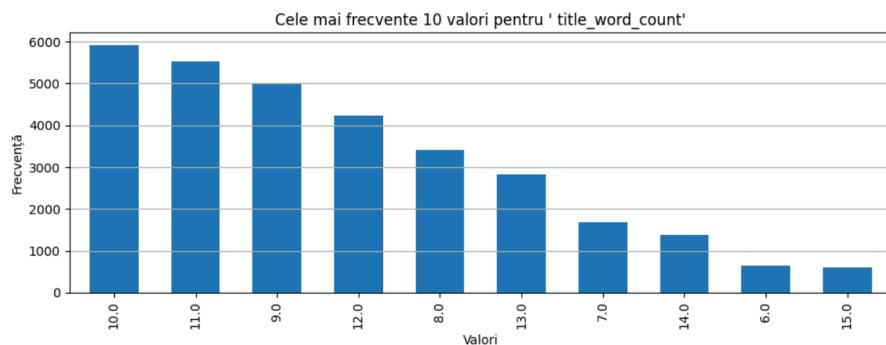
	count_non_missing	num_unique_values
url	31715	31715
title_word_count	31715	20
content_word_count	31715	2284
external_links	31715	1378
internal_links	31715	53
image_count	31715	87
video_count	31715	52
keyword_count	31715	10
channel_lifestyle	28540	2
channel_entertainment	31715	2
channel_business	31715	2
channel_social_media	31715	2
channel_tech	31715	2
channel_world	31715	2
keyword_worst_min_shares	31715	25
keyword_worst_max_shares	31715	1050
keyword_best_min_shares	31715	975
keyword_best_max_shares	31715	34
keyword_avg_min_shares	31715	13413
keyword_avg_max_shares	31715	16619
ref_min_shares	31715	1219
ref_max_shares	31715	1104
day_monday	31715	2
day_tuesday	31715	2
day_wednesday	31715	2
day_thursday	31715	2
day_friday	31715	2
day_saturday	31715	2
day_sunday	31715	2
is_weekend	31715	2
publication_period	31715	2
popularity_category	31715	5

- pentru setul de test:

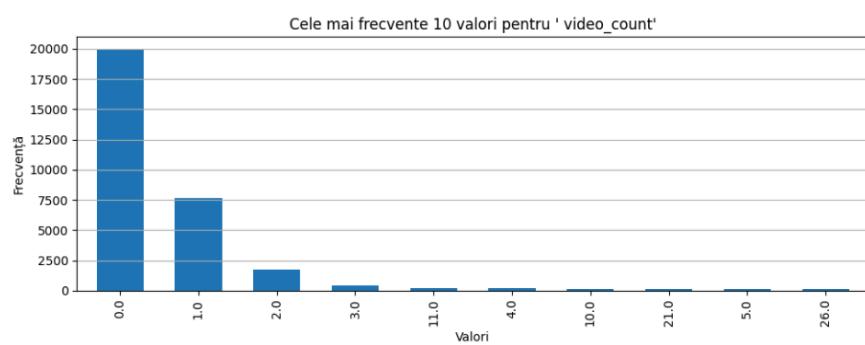
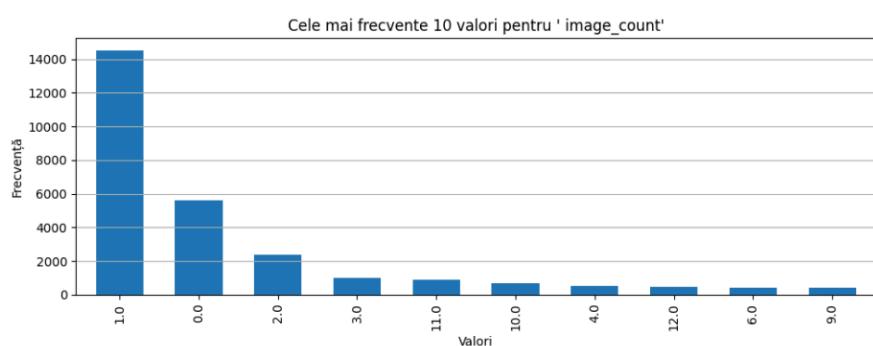
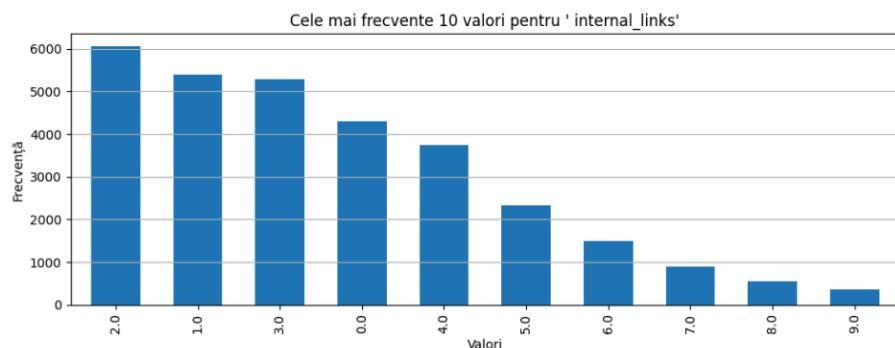
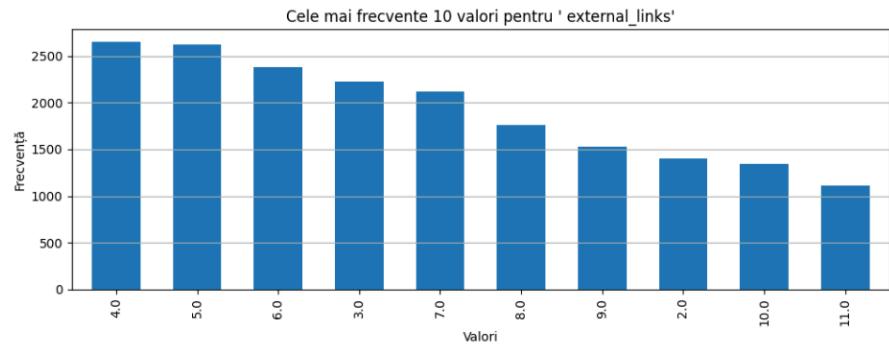
	count_non_missing	num_unique_values
url	7929	7929
title_word_count	7929	17
content_word_count	7929	1588
external_links	7929	425
internal_links	7929	43
image_count	7929	69
video_count	7929	38
keyword_count	7929	10
channel_lifestyle	7140	2
channel_entertainment	7929	2
channel_business	7929	2
channel_social_media	7929	2
channel_tech	7929	2
channel_world	7929	2
keyword_worst_min_shares	7929	10
keyword_worst_max_shares	7929	888
keyword_best_min_shares	7929	598
keyword_best_max_shares	7929	15
keyword_avg_min_shares	7929	3897
keyword_avg_max_shares	7929	5797
ref_min_shares	7929	887
ref_max_shares	7929	747
day_monday	7929	2
day_tuesday	7929	2
day_wednesday	7929	2
day_thursday	7929	2
day_friday	7929	2
day_saturday	7929	2
day_sunday	7929	2
is_weekend	7929	2
publication_period	7929	2
popularity_category	7929	5

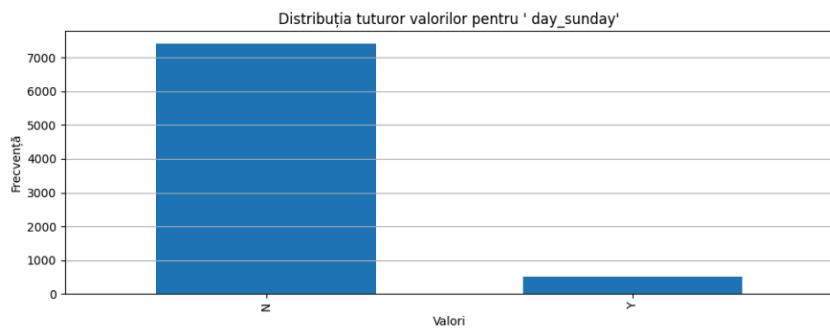


Din graficul de mai sus deducem faptul ca toate url-urile sunt diferite (nu există 2 url-uri identice), deci acesta este un atribut pe care îl putem elimina pentru că nu aduce nicio informație în plus pentru clasificare.



Din graficul de mai sus observăm faptul că 0.0 este de departe cea mai frecventă valoare pentru content\_word\_count, indicând un dezechilibru, caci domina cu o frecvență de aproape 1000, urmatoarea valoare (286) având o frecvență de aproape 100.

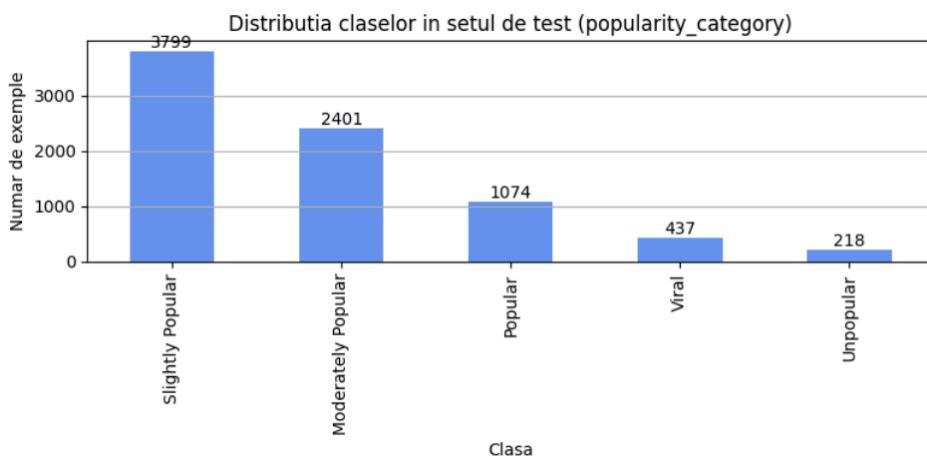
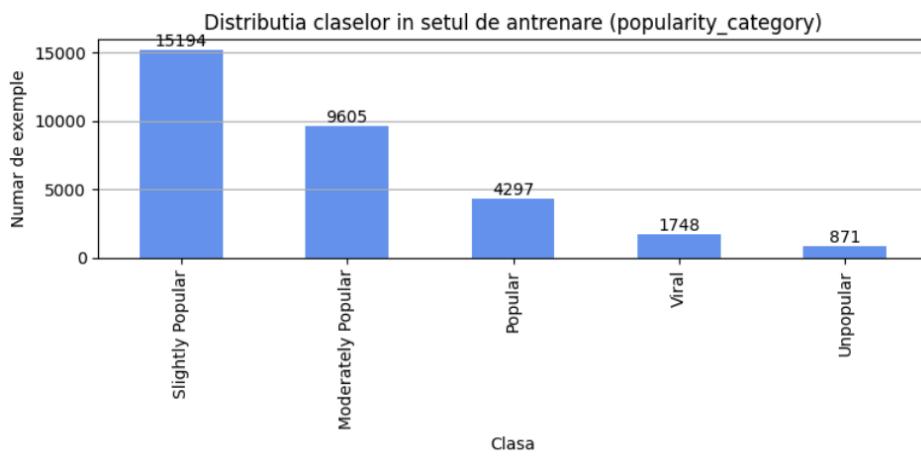




Astfel de grafice ilustreaza tot dezechilibrul claselor, caci avem peste 7000 de exemple de N versus sub 1000 de exemple de Y, ceea ce in mod evident va influenta negativ procesul de invatare, intrucat clasele rare vor trece aproape neobservate si vor fi mult mai usor prezise gresit.

... (restul graficelor sunt disponibile in fisierul .ipynb).

#### b. Analiza echilibrului de clase

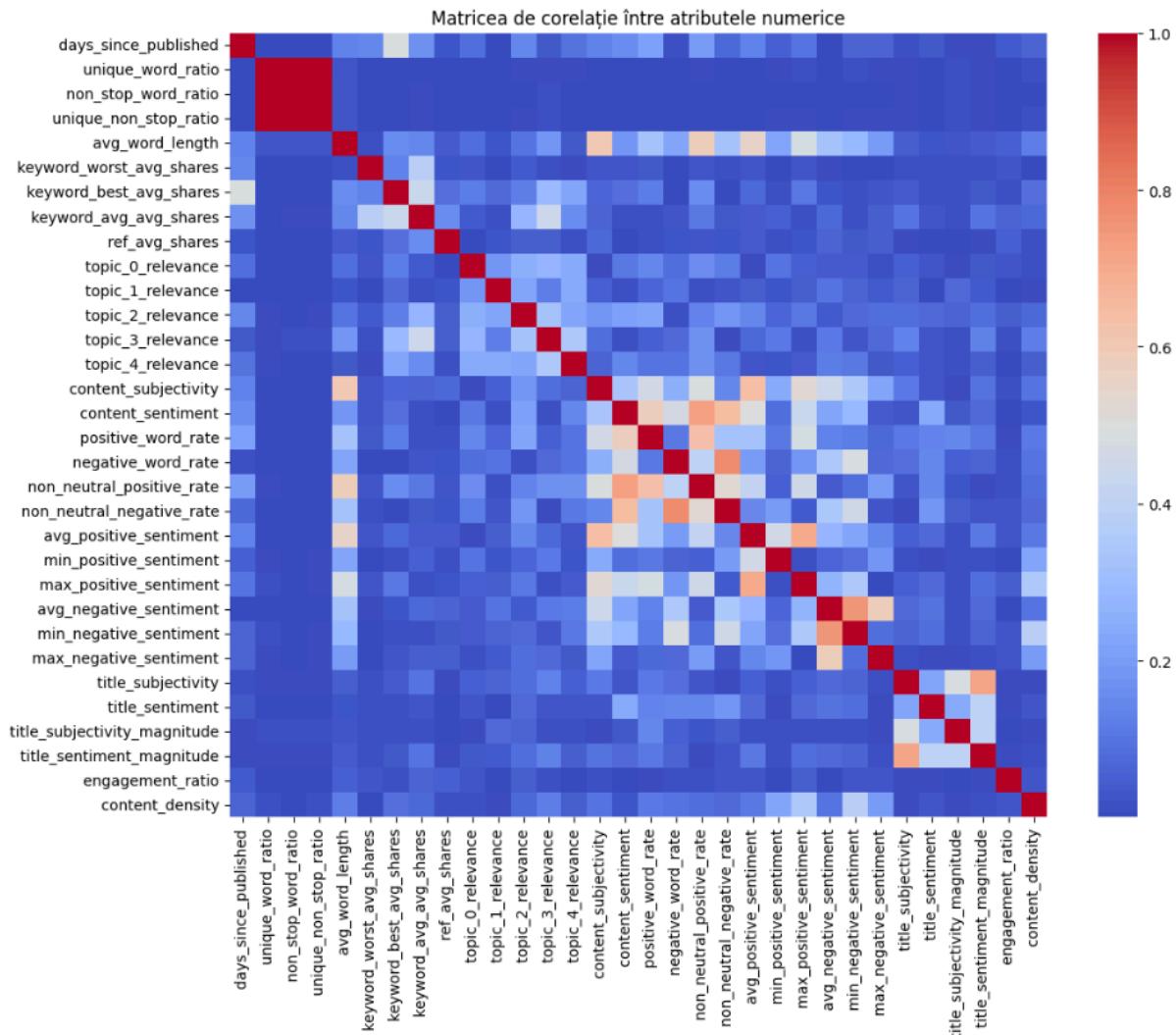


Exista un dexechilibru clar in ambele seturi, clasa Slightly Popular dominand, urmata de Moderately Popular. La polul opus, clasele Unpopular si Viral sunt mai putin frecvente. Astfel, modelele de clasificare vor tinde sa favorizeze clasele cu numar mare de exemple, iar clasele rare (Unpopular si Viral) pot ajunge sa aiba scoruri de recall, precision si f1 mai mici.

### 3. Analize de corelatie

a. intre atribute numerice continue

- matricea de corelatie (Pearson) pentru setul de train:

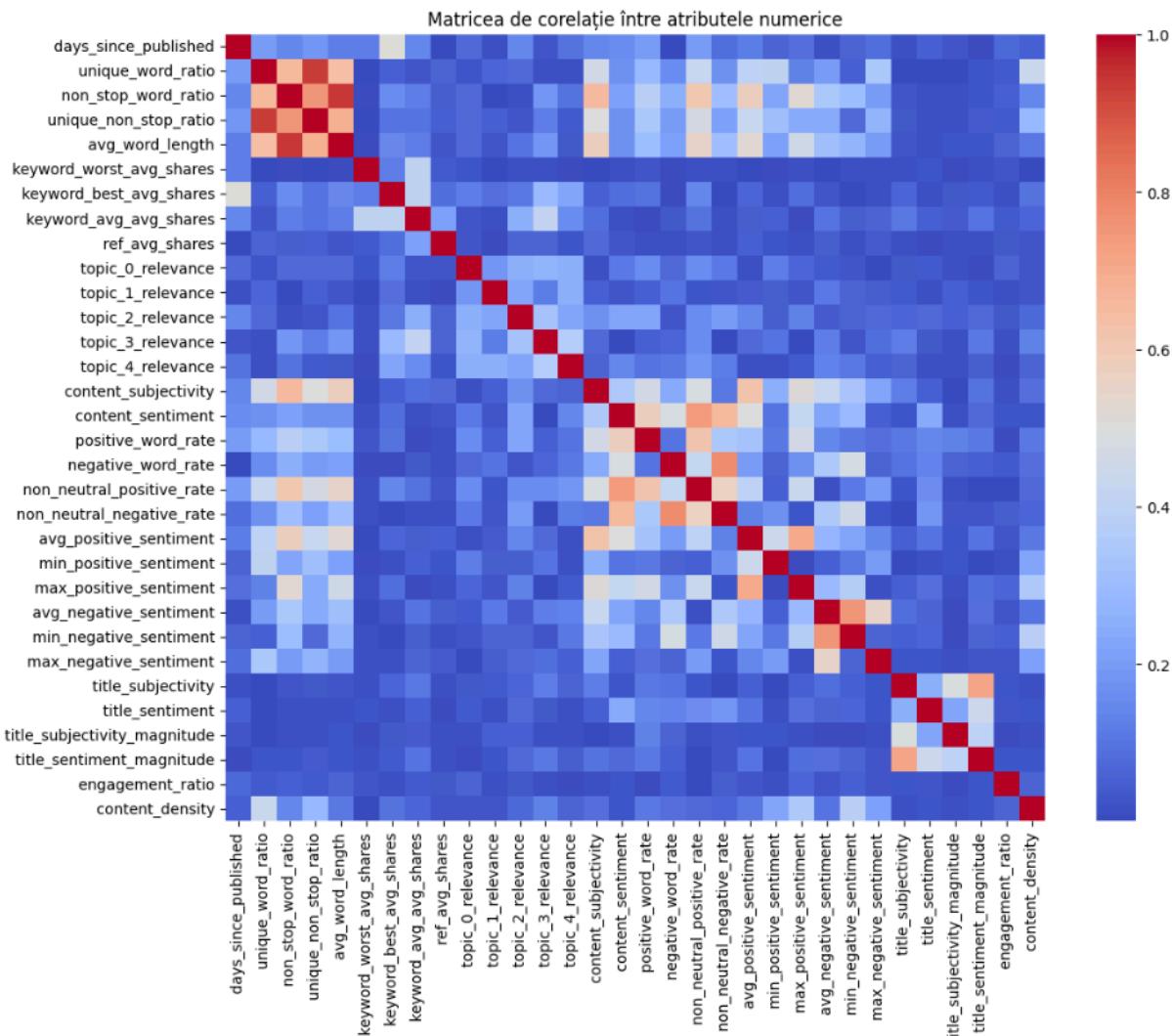


Observam o corelatie sporita intre urmatoarele 3 atribute:

Attribute 1	Attribute 2	Correlation
unique_non_stop_ratio	unique_word_ratio	0.999880
non_stop_word_ratio	unique_word_ratio	0.999657
unique_non_stop_ratio	non_stop_word_ratio	0.999627

Toate aceste 3 atribute sunt corelate intre ele, deci e suficient sa pastram unul singur, caci acesta ne va oferi aceleasi informatii ca celelalte doua, deci putem elimina oricare 2 dintre ele, eu am ales sa elimin "unique\_word\_ratio", "non\_stop\_word\_ratio" (attribute redundante).

- matricea de corelatie (Pearson) pentru setul de test



Aici nu mai este relevata corelatia dintre cele 3 atribute de mai sus, ci sunt gasite alte perechi de atribute cu grad ridicat de corelatie intre ele:

Attribute 1	Attribute 2	Correlation
avg_word_length	non_stop_word_ratio	0.940239
unique_non_stop_ratio	unique_word_ratio	0.936529

### b. intre atribute categorice

- folosim testul Chi-Patrat si comparam p-value cu 0.05, exact ca la setul de date anterior (air\_pollution)

## 2. Preprocesarea datelor

```
identify_attributes_with_na
✓ 0.0s
channel_lifestyle      3175
content_density         3145
dtype: int64

identify_attributes_with_na
✓ 0.0s
content_density         819
channel_lifestyle       789
dtype: int64
```

Atributele care au valori lipsa sunt channel\_lifestyle si content\_density, deci vom face imputare univariata pe ele. content\_density este un numeric continuu, deci ii vom completa valorile lipsa cu media valorilor existente pe acea coloana, iar channel\_lifestyle este un atribut categorial, deci vom completa

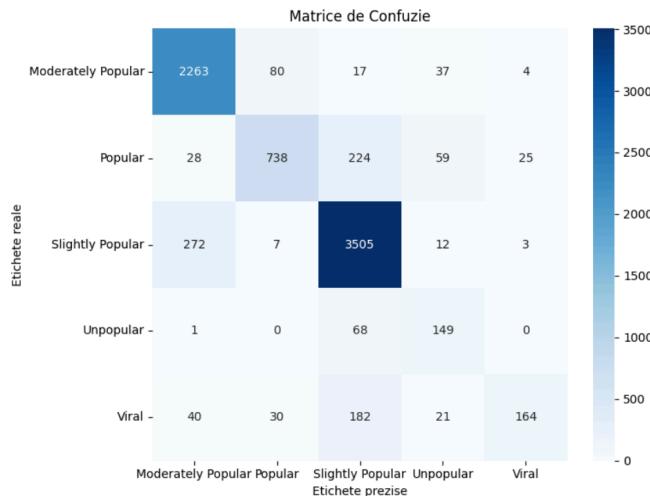
valorile lipsa cu cea mai frecventa valoare existenta pe aceasta coloana.

Pentru prelucrarea valrilor extreme am folosit tot iqr, ca la setul de date anterior.

### 3. Utilizarea algoritmilor de învățare automata

#### 1. Decision Tree

Cu aceeasi hiperparametru folositi si pentru setul de date air\_pollution, acuratetea este mai mica la acest set de date, mai exact 0.86.



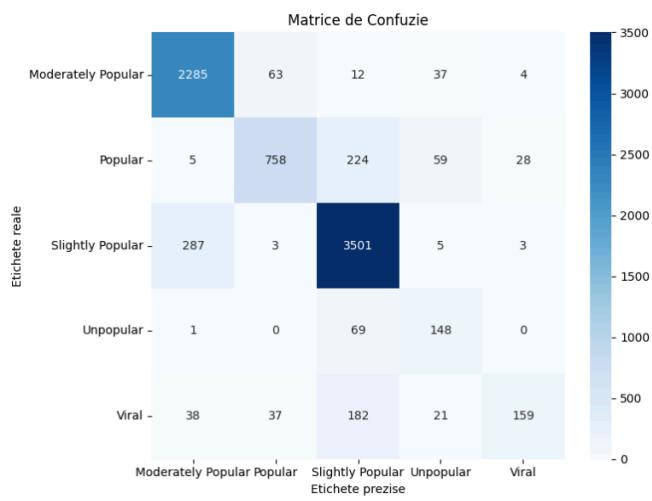
	precision	recall	f1-score
Moderately Popular	0.8690	<b>0.9425</b>	<b>0.9043</b>
Popular	0.8632	0.6872	0.7652
Slightly Popular	<b>0.8771</b>	0.9226	0.8993
Unpopular	0.5360	0.6835	0.6008
Viral	0.8367	0.3753	0.5182

Acuratețea generală (fără aproximare): 0.860007567158532

Avand in vedere ca setul de date este mult mai mare decat air\_pollution, am incercat sa maresc adancimea maxima max\_depth = 50 si min\_samples\_leaf la 5, iar acuratetea a crescut de la 0.86001 la 0.86404:

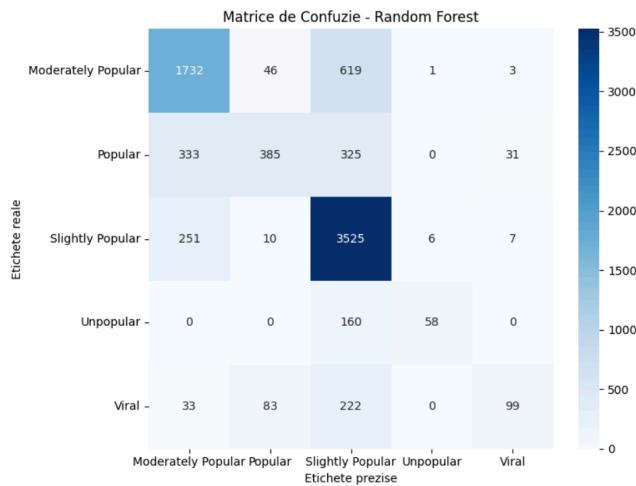
	precision	recall	f1-score
Moderately Popular	0.8735	<b>0.9517</b>	<b>0.9109</b>
Popular	<b>0.8804</b>	0.7058	0.7835
Slightly Popular	0.8779	0.9216	0.8992
Unpopular	0.5481	0.6789	0.6066
Viral	0.8196	0.3638	0.5040

Acuratețea generală (fără aproximare): 0.86404338504225



## 2. Random Forest

- cu hiperparametrii folositi la setul de date anterior:



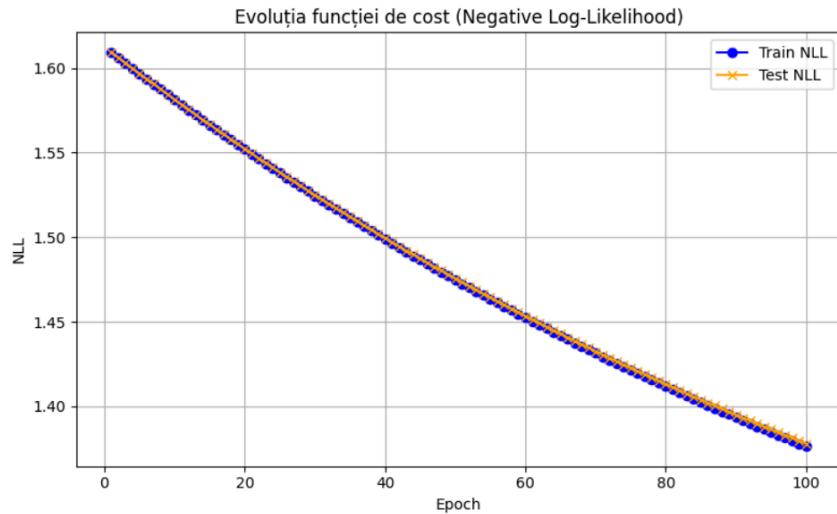
Tabel cu metricele de clasificare per clasă (Random Forest):			
	precision	recall	f1-score
Moderately Popular	0.7373	0.7214	0.7293
Popular	0.7347	0.3585	0.4819
Slightly Popular	0.7267	<b>0.9279</b>	<b>0.8150</b>
Unpopular	<b>0.8923</b>	0.2661	0.4099
Viral	0.7071	0.2265	0.3432
macro avg	0.7596	0.5001	0.5558
weighted avg	0.7345	0.7314	0.7068

Acuratețea generală (fără aproximatie): 0.7313658721150208

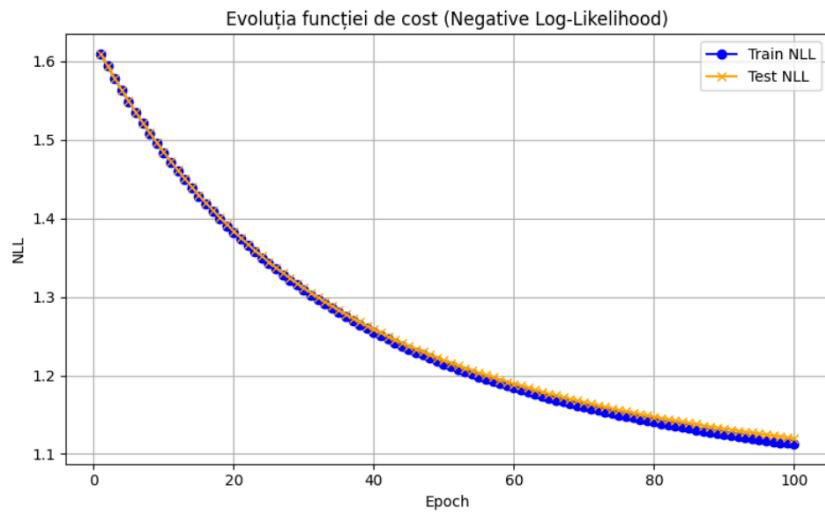
Observatiile despre parametrii discutate la setul 1 de date raman valabile.

## 3. Regresie logistica

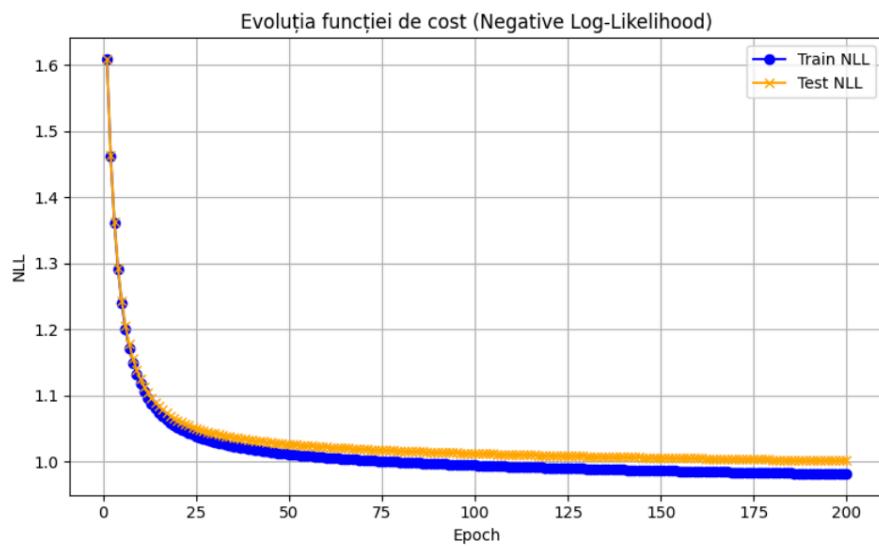
Pentru o rata de invatare de 0.01 si 100 de epoci, acuratetea este undeva la 0.5586, iar functia de cost arata asa:

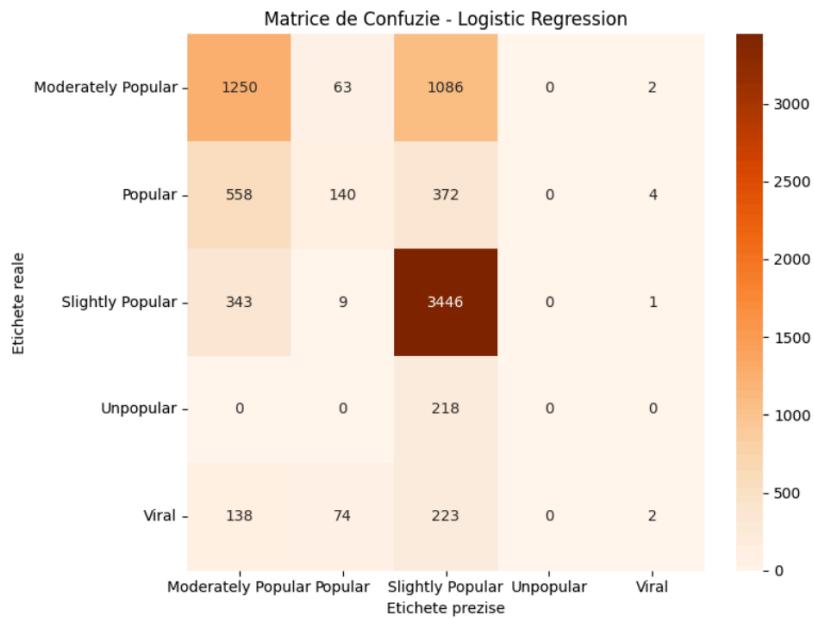


Dacă mai creștem treptat learning rate-ul până la 0.05, acuratețea crește la 0.5752, iar curba funcției de loss este mai abruptă:



Din ce am observat, de la lr = 0.5 în sus și nr\_epoci = 200, acuratețea nu mai crește, se stabilizează la 0.61, iar curba funcției de loss este foarte abruptă:



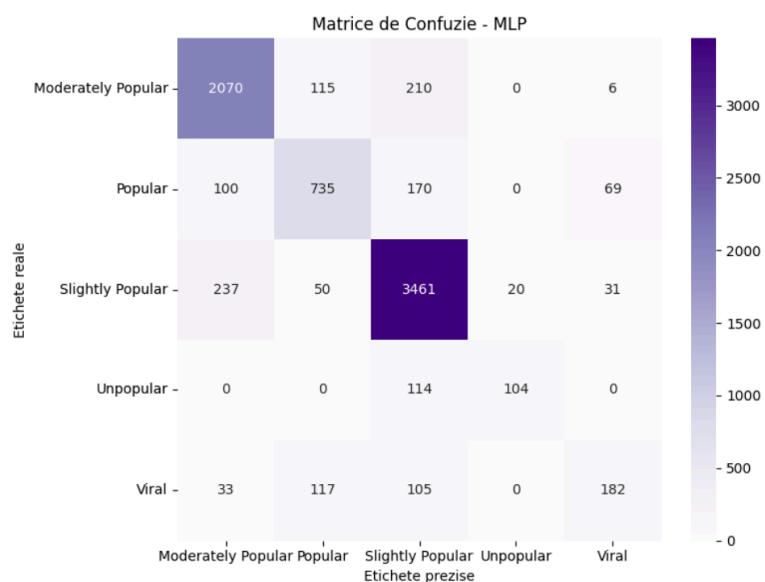


	precision	recall	f1-score
Moderately Popular	0.5461	0.5206	0.5330
Popular	0.4895	0.1304	0.2059
Slightly Popular	<b>0.6447</b>	<b>0.9071</b>	<b>0.7537</b>
Unpopular	0.0000	0.0000	0.0000
Viral	0.2222	0.0046	0.0090
macro avg	0.3805	0.3125	0.3003
weighted avg	0.5528	0.6102	0.5509

Acuratețea generală (fără aproximatie): 0.6101652162946147

#### 4. MLP

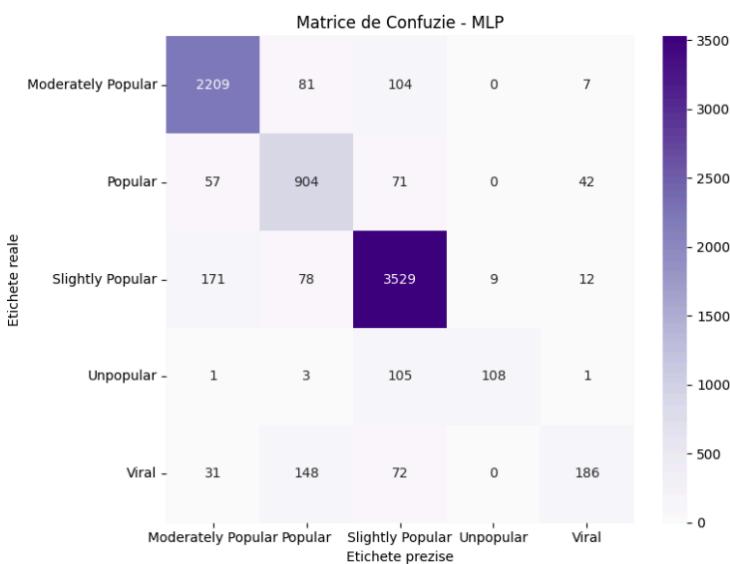
- cu parametrii default, am obtinut urmatoarele rezultate



	precision	recall	f1-score
Moderately Popular	0.8484	0.8621	0.8552
Popular	0.7227	0.6844	0.7030
Slightly Popular	<b>0.8525</b>	<b>0.9110</b>	<b>0.8808</b>
Unpopular	0.8387	0.4771	0.6082
Viral	0.6319	0.4165	0.5021
macro avg	0.7788	0.6702	0.7098
weighted avg	0.8211	0.8263	0.8206

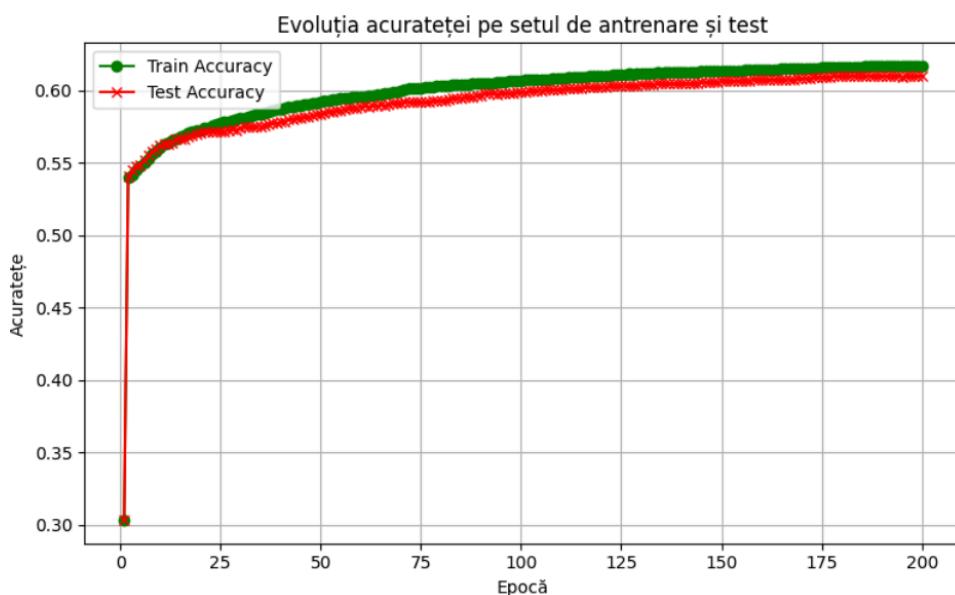
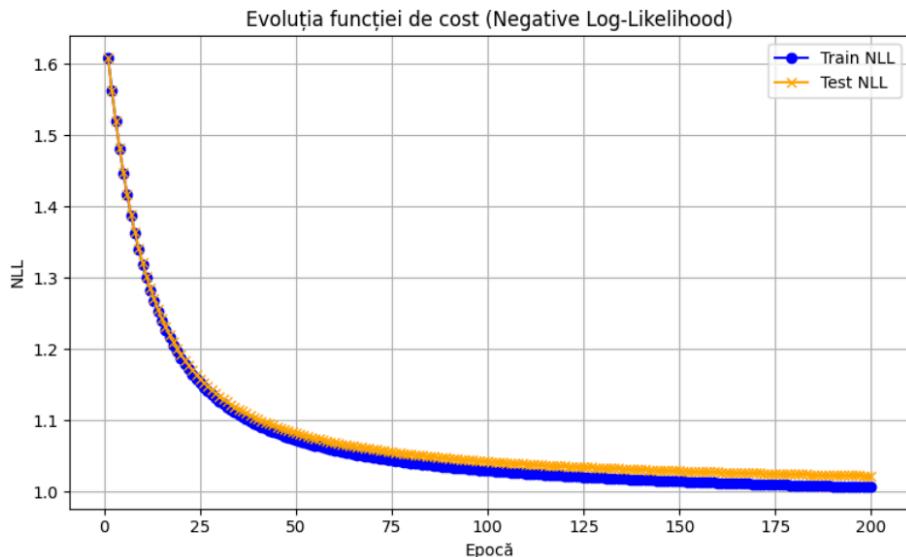
Acuratețea generală (fără aproximație): 0.8263337116912599

Avand în vedere că setul de date este mare, am crescut numarul de layers și numarul de neuroni per layer (hidden\_layer\_sizes=(128,64)). De asemenea, am crescut learning\_rate\_init la 0.005 de la 0.001 (cum era default). Ca urmare a acestor modificări, acuratețea a crescut de la 0.8263 la 0.8747.



	precision	recall	f1-score
Moderately Popular	0.8947	0.9200	0.9072
Popular	0.7446	0.8417	0.7902
Slightly Popular	0.9093	<b>0.9289</b>	<b>0.9190</b>
Unpopular	<b>0.9231</b>	0.4954	0.6448
Viral	0.7500	0.4256	0.5431
macro avg	0.8443	0.7223	0.7608
weighted avg	0.8742	0.8748	0.8697

Acuratețea generală (fără aproximație): 0.874763526295876



Curbele au aceeași formă ca la primul set de date, iar observațiile raman aceleasi.

Adaugari la comparatia algoritmilor:

- decision tree: desi are a doua cea mai buna acuratete, poate fi predispus la overfitting pe teste asa de mari daca nu este controlata adancimea.
- random forest: dupa cum am zis si la setul anterior de date, performeaza constant bine, avand avantajul ca reduce overfitting-ul prin generalizarea datei de utilizarea mai multor arbori
- regresie logistica: cea mai scazuta acuratete, are o limitarea fundamentala: nu reuseste sa surprinda relatiile neliniare complexe din setul de date
- are cea mai buna acuratete, cu o capacitate mare de generalizare pe seturi de date mari, poate invata relatii neliniare complexe. A avut cele mai bune rezultate.

Tabelul comparativ intre algoritmi se afla in fisierul part2\_news\_popularity.ipynb.