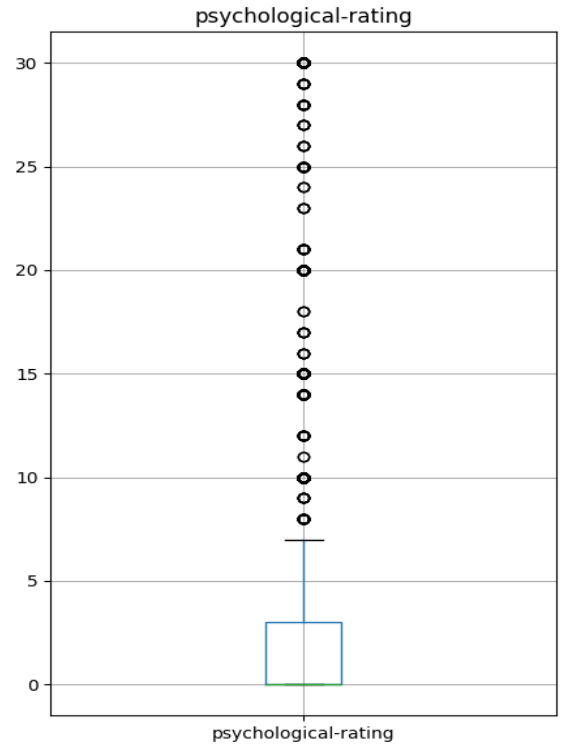
Tema2IA

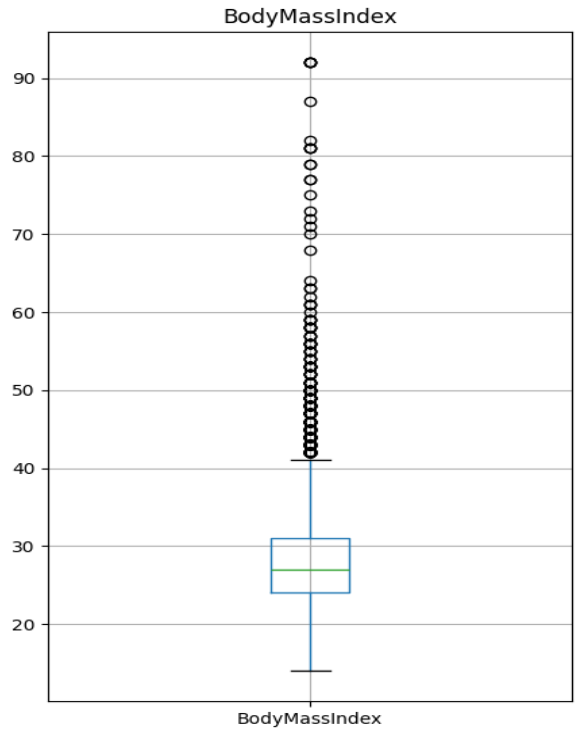
Cerinta 3.1:

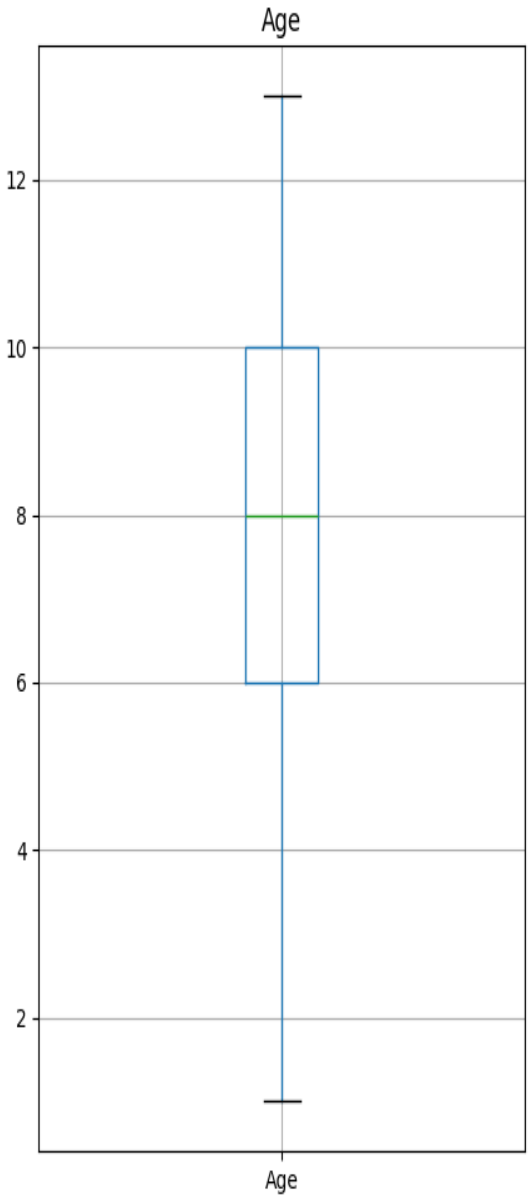
Diabet

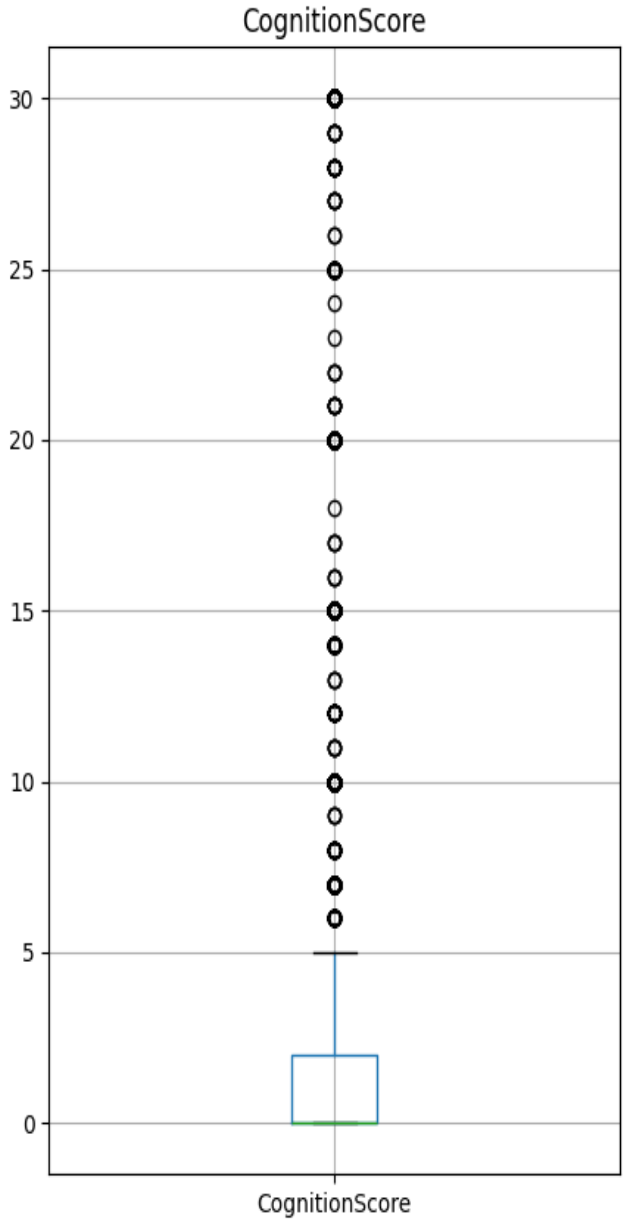
- **Atribute numerice:**

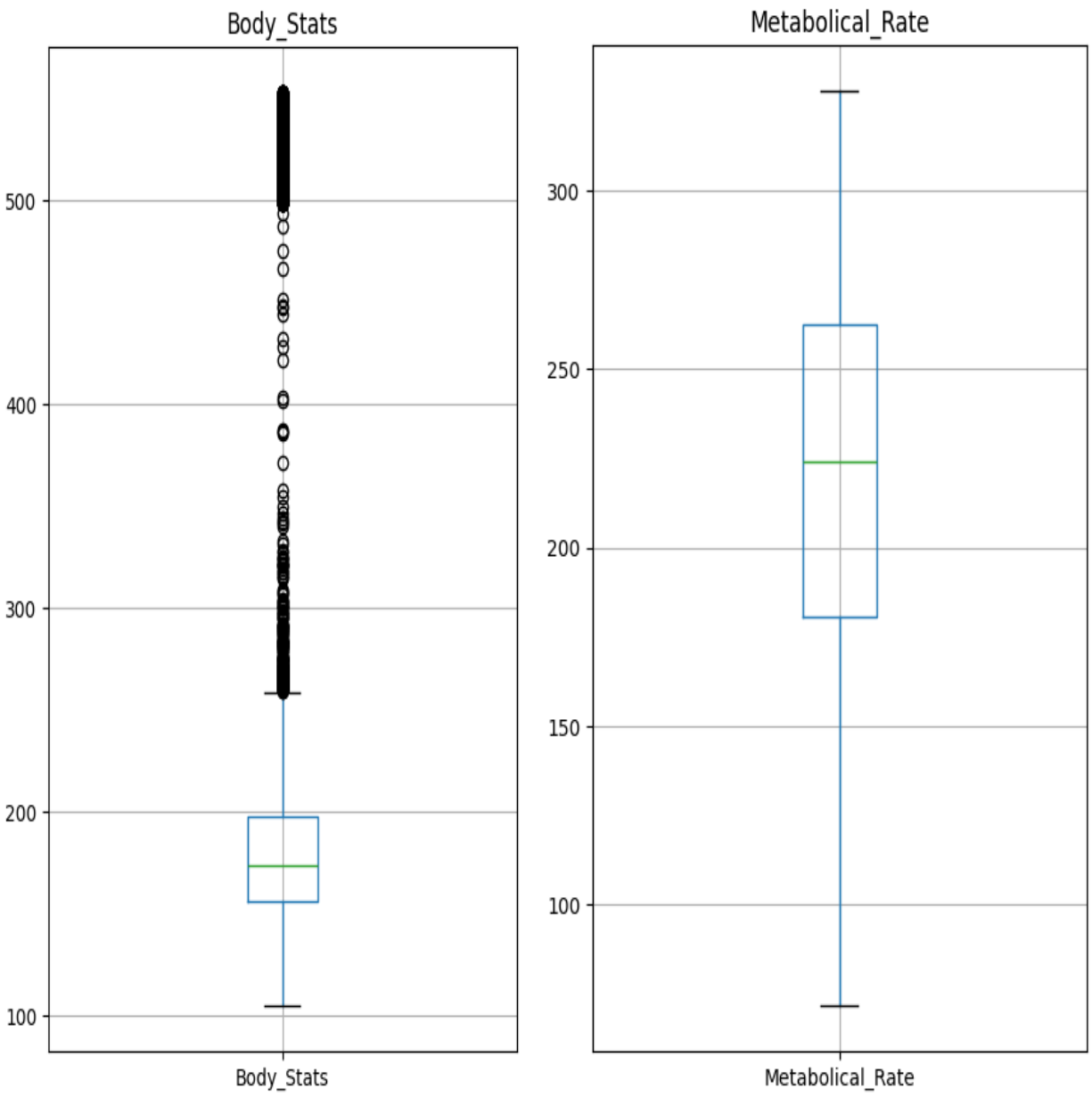
	psychological-rating	BodyMassIndex	Age	CognitionScore	Body_Stats	Metabolical_Rate
count	10000.000000	10000.000000	10000.0000	10000.000000	10000.000000	9000.000000
mean	4.365100	28.246500	8.0575	3.125300	194.960784	221.592499
std	8.891103	6.462563	3.0363	7.308607	82.438106	60.480951
min	0.000000	14.000000	1.0000	0.000000	105.063984	71.602207
25%	0.000000	24.000000	6.0000	0.000000	156.720671	180.542314
50%	0.000000	27.000000	8.0000	0.000000	174.042100	224.218817
75%	3.000000	31.000000	10.0000	2.000000	197.742249	262.688901
max	30.000000	92.000000	13.0000	30.000000	553.000000	327.936098





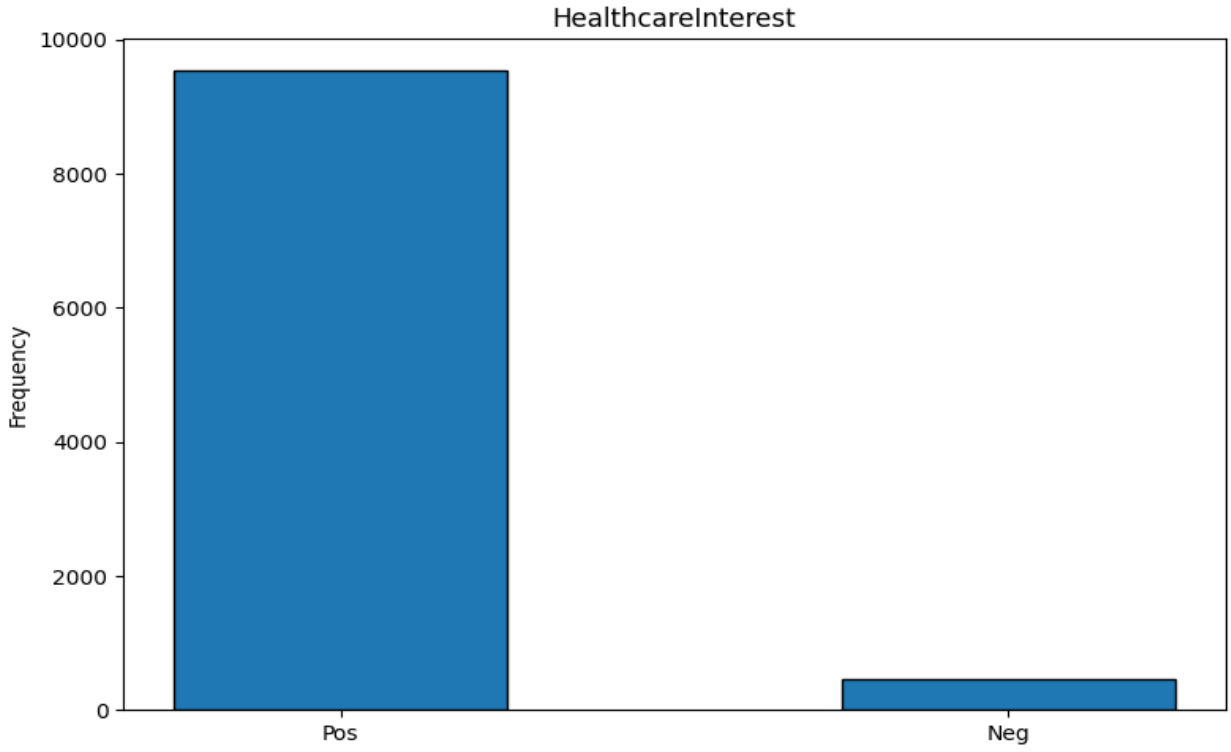

Age
12
10
8
6
4
2
Age

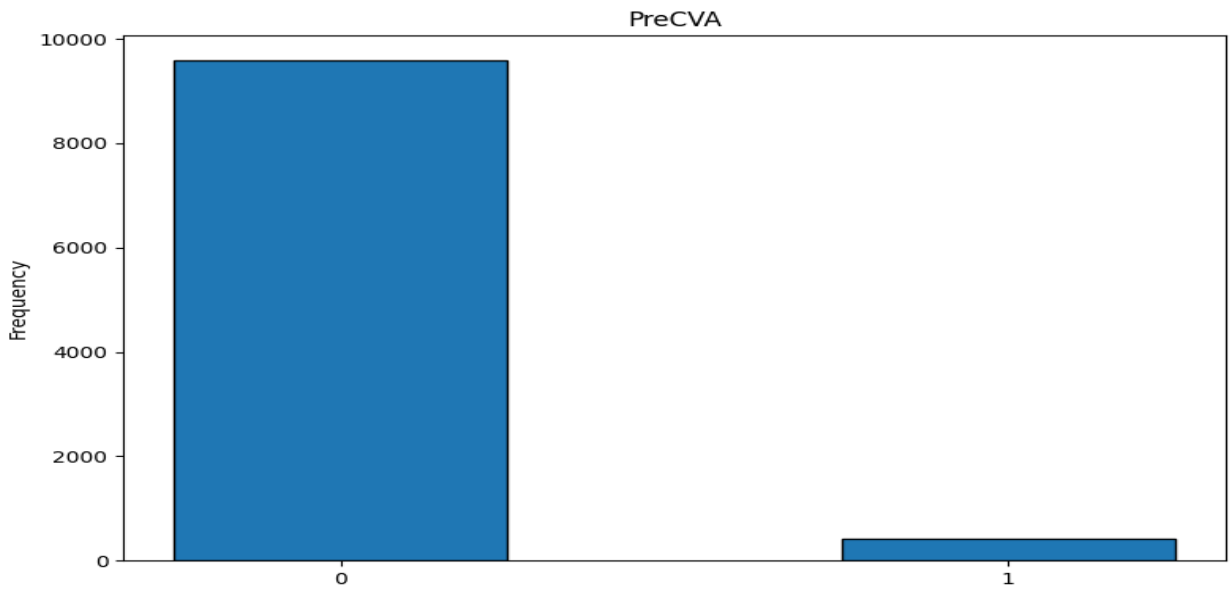

CognitionScore
30
25
20
15
10
5
0
CognitionScore

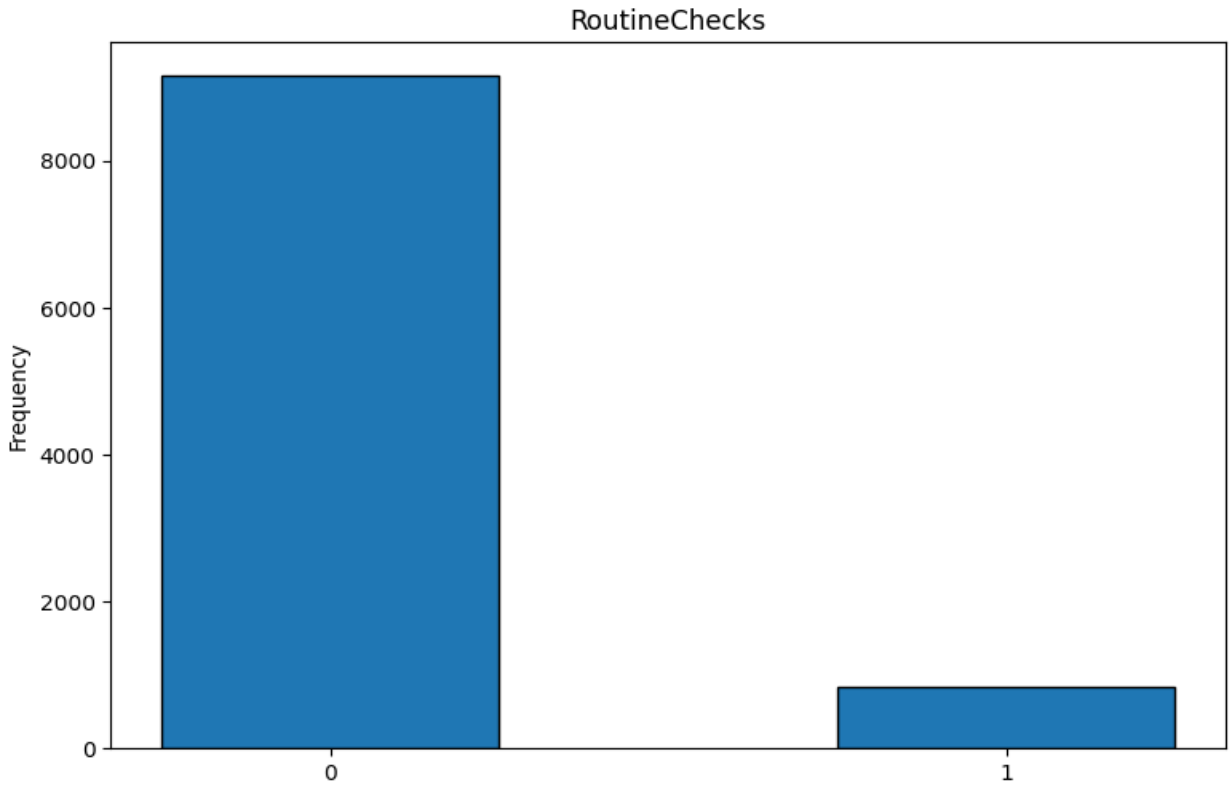

Body_Stats
500
400
300
200
100
Body_Stats
Metabolical_Rate
300
250
200
150
100
Metabolical_Rate

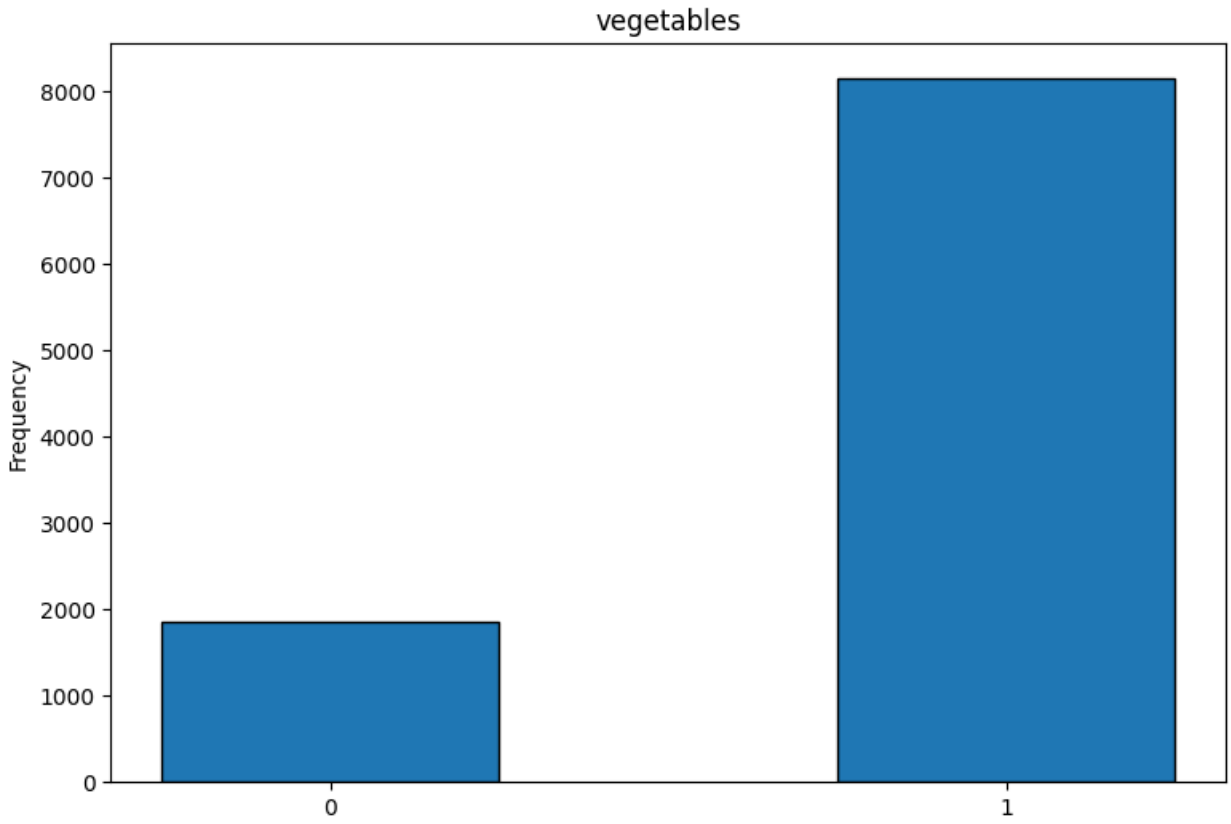
- **Atribute discrete:**

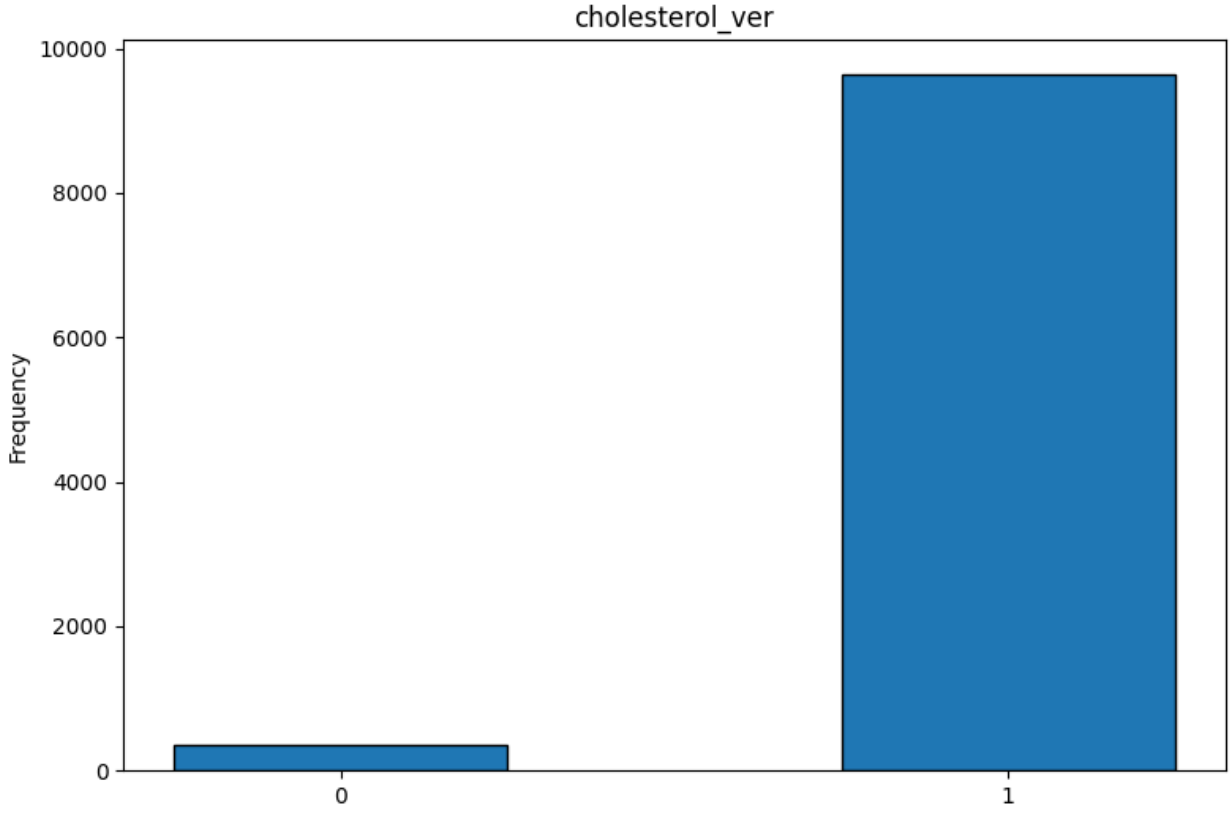
	Attribute	Non-missing Count	Unique Values Count
0	HealthcareInterest	10000	2
1	PreCVA	10000	2
2	RoutineChecks	10000	2
3	CompletedEduLvl	9000	6
4	alcoholAbuse	10000	2
5	cholesterol_ver	10000	2
6	vegetables	10000	2
7	HighBP	10000	2
8	Unprocessed_fructose	10000	2
9	Jogging	10000	2
10	IncreasedChol	10000	2
11	gender	10000	2
12	HealthScore	10000	5
13	myocardial_infarction	10000	2
14	SalaryBraket	10000	8
15	Cardio	10000	2
16	ImprovedAveragePulmonaryCapacity	10000	2
17	Smoker	10000	2

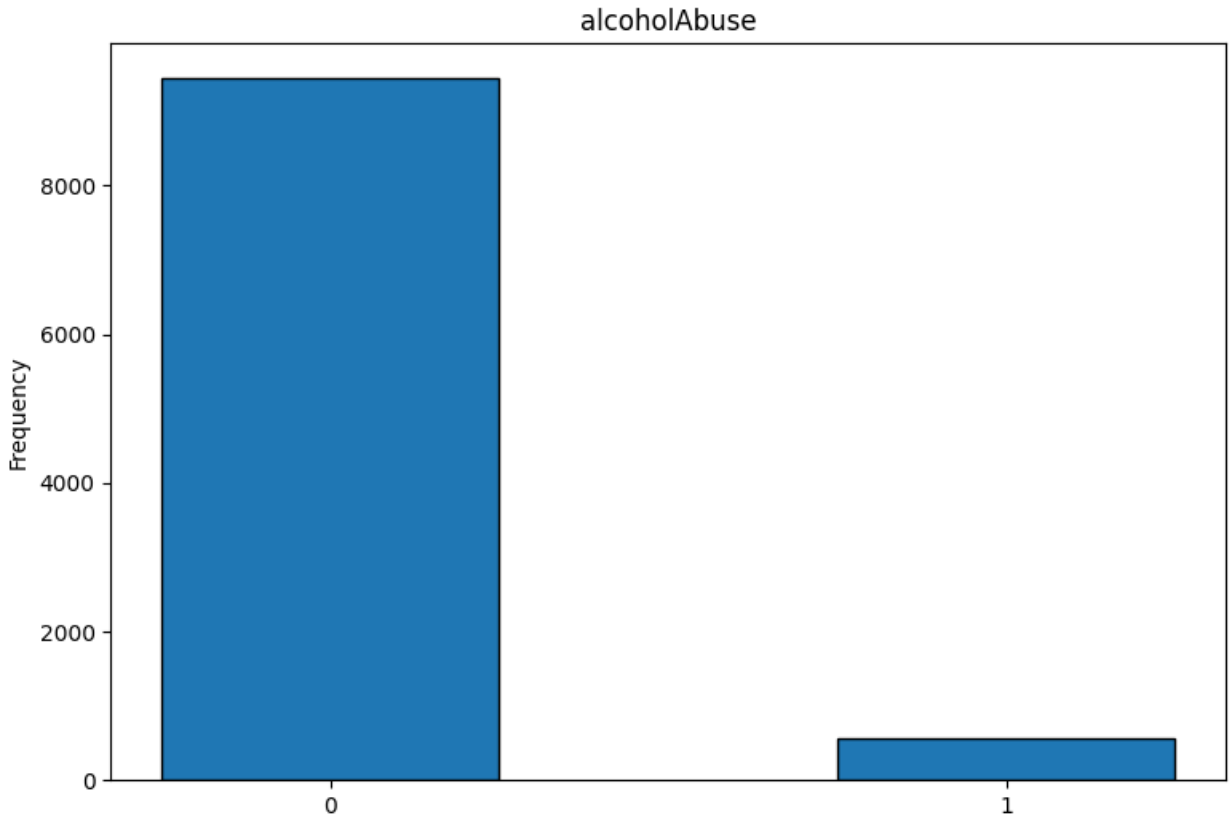


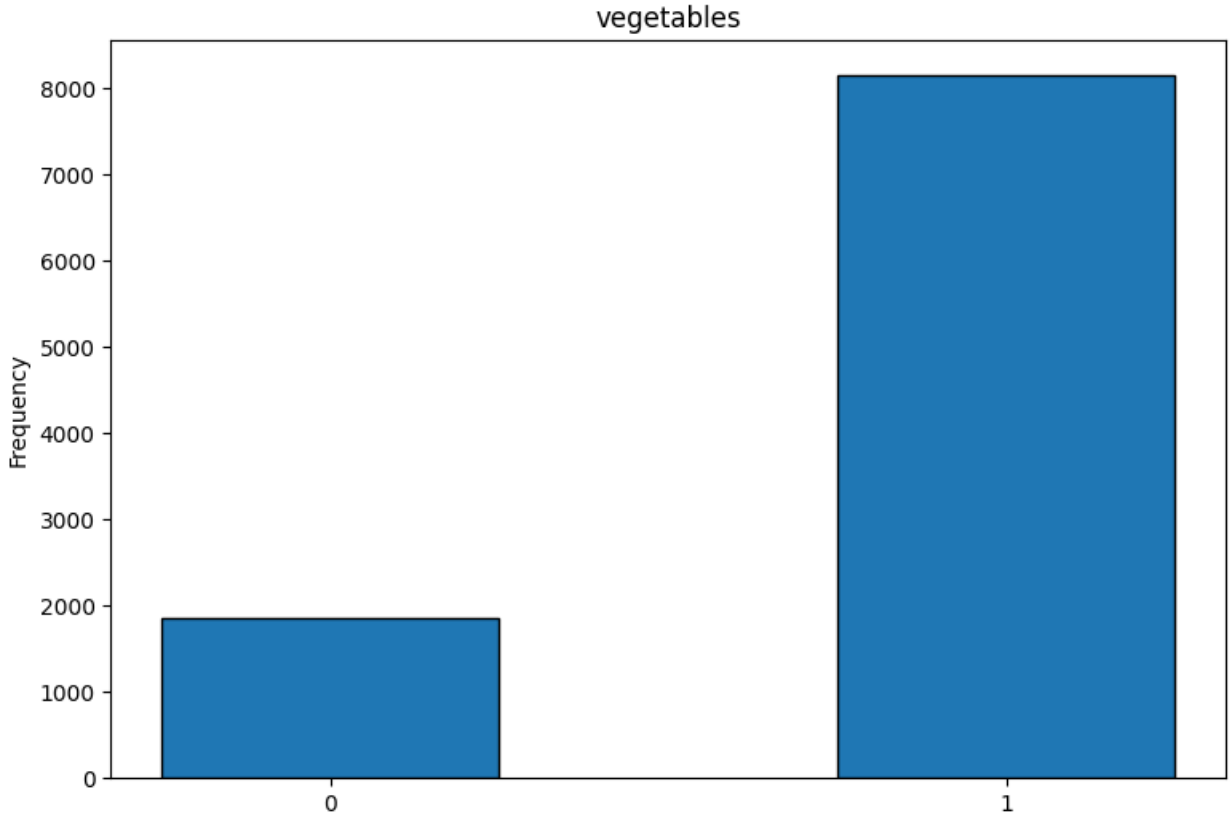

PreCVA
Frequency
10000
8000
6000
4000
2000
0
0
1

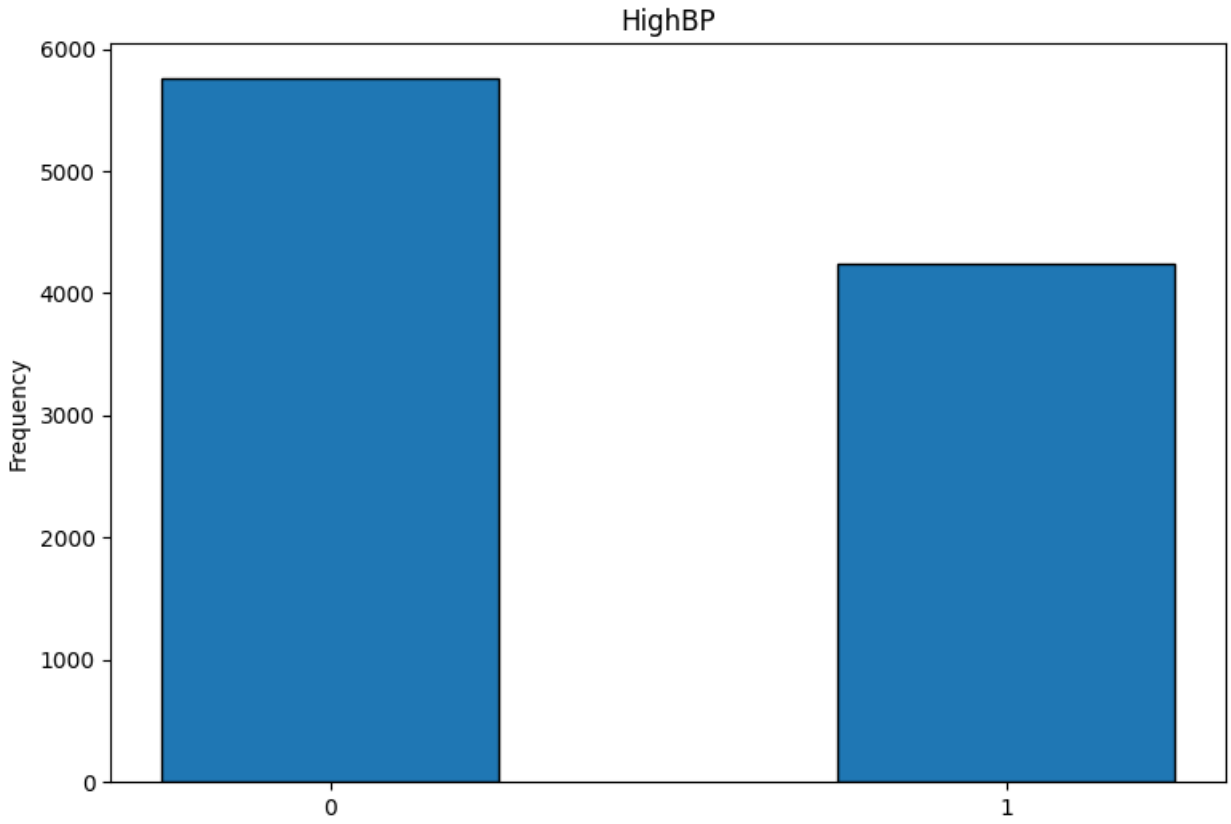

RoutineChecks
Frequency
8000
6000
4000
2000
0
0
1

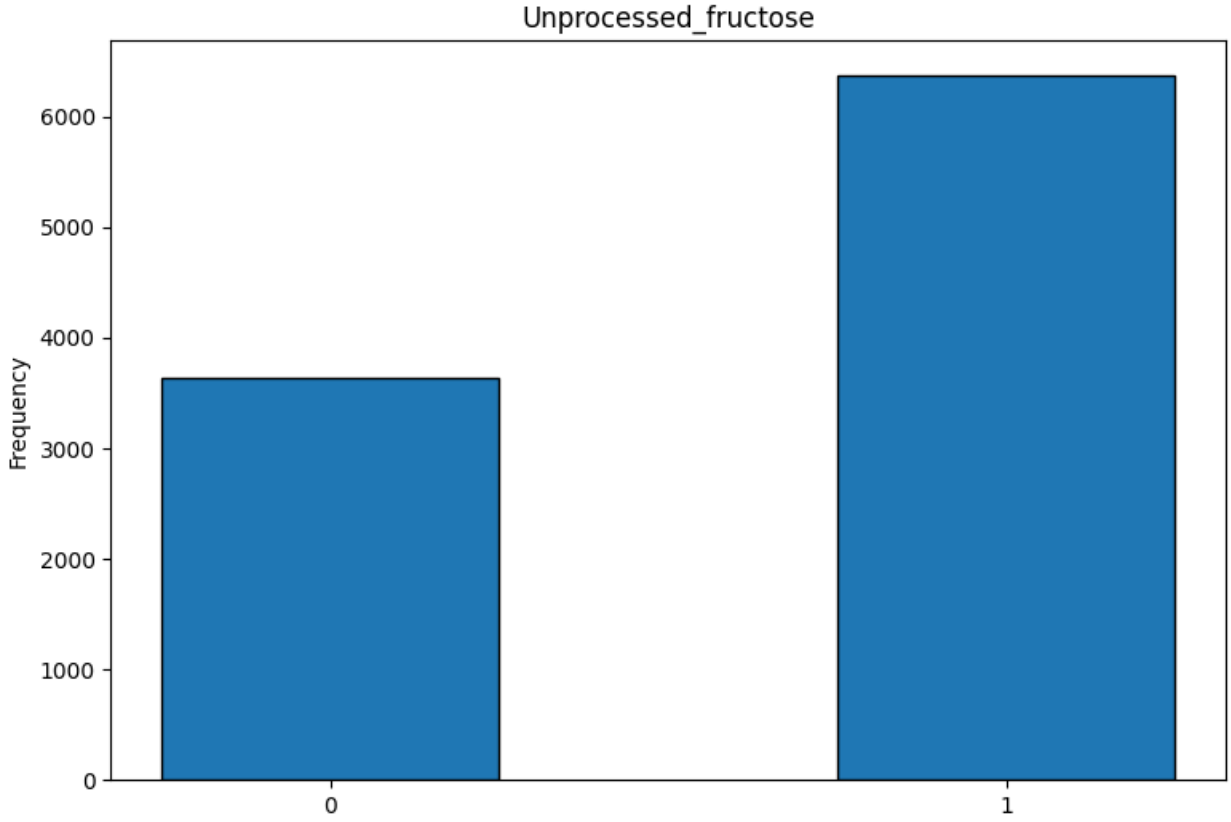

vegetables
Frequency
8000
7000
6000
5000
4000
3000
2000
1000
0
0
1

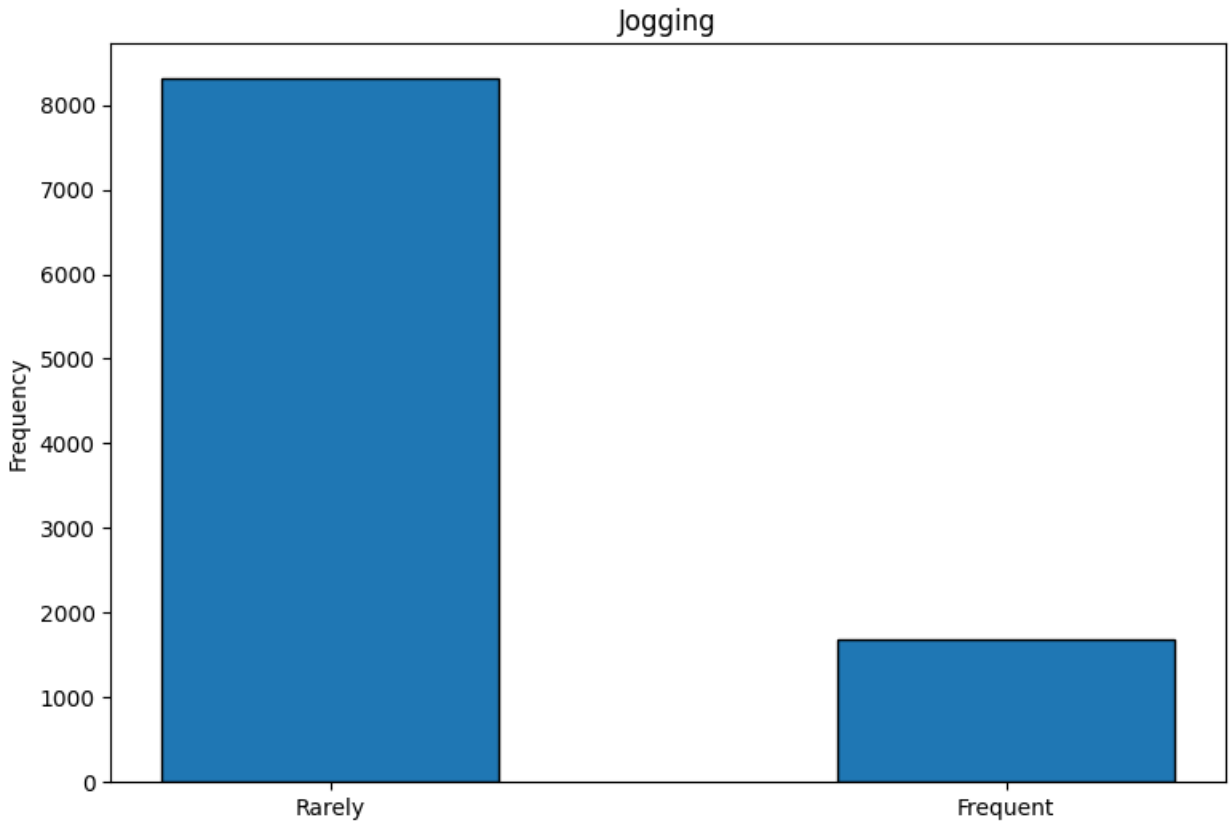

cholesterol_ver
Frequency
10000
8000
6000
4000
2000
0
0
1

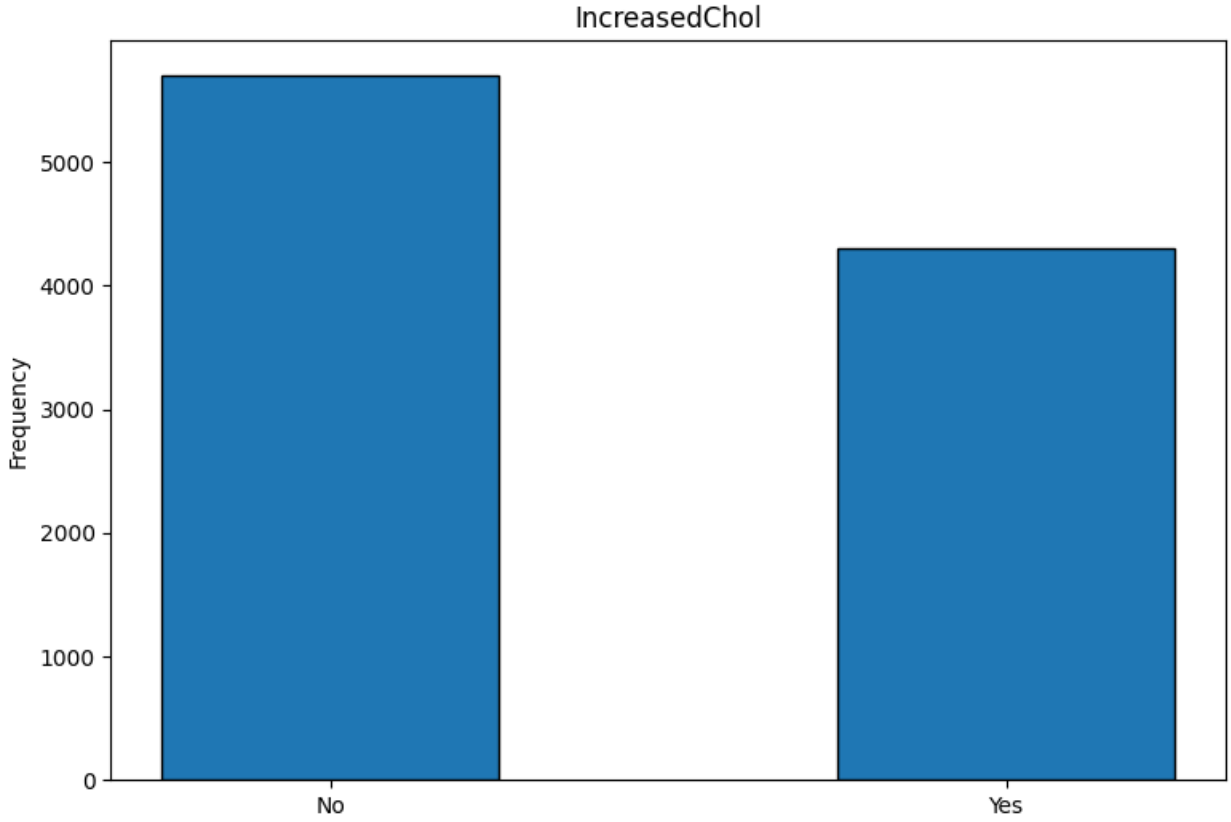

alcoholAbuse
Frequency
8000
6000
4000
2000
0
0
1

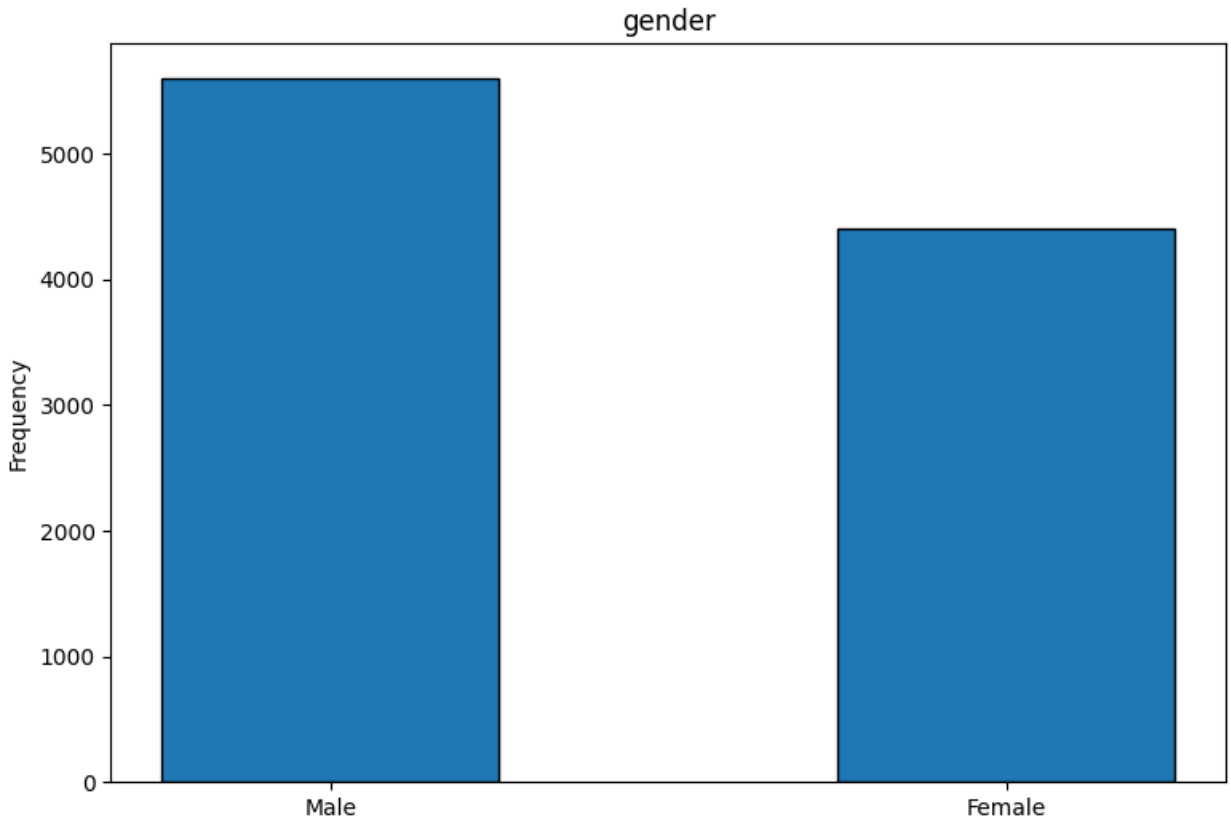

vegetables
Frequency
8000
7000
6000
5000
4000
3000
2000
1000
0
0
1

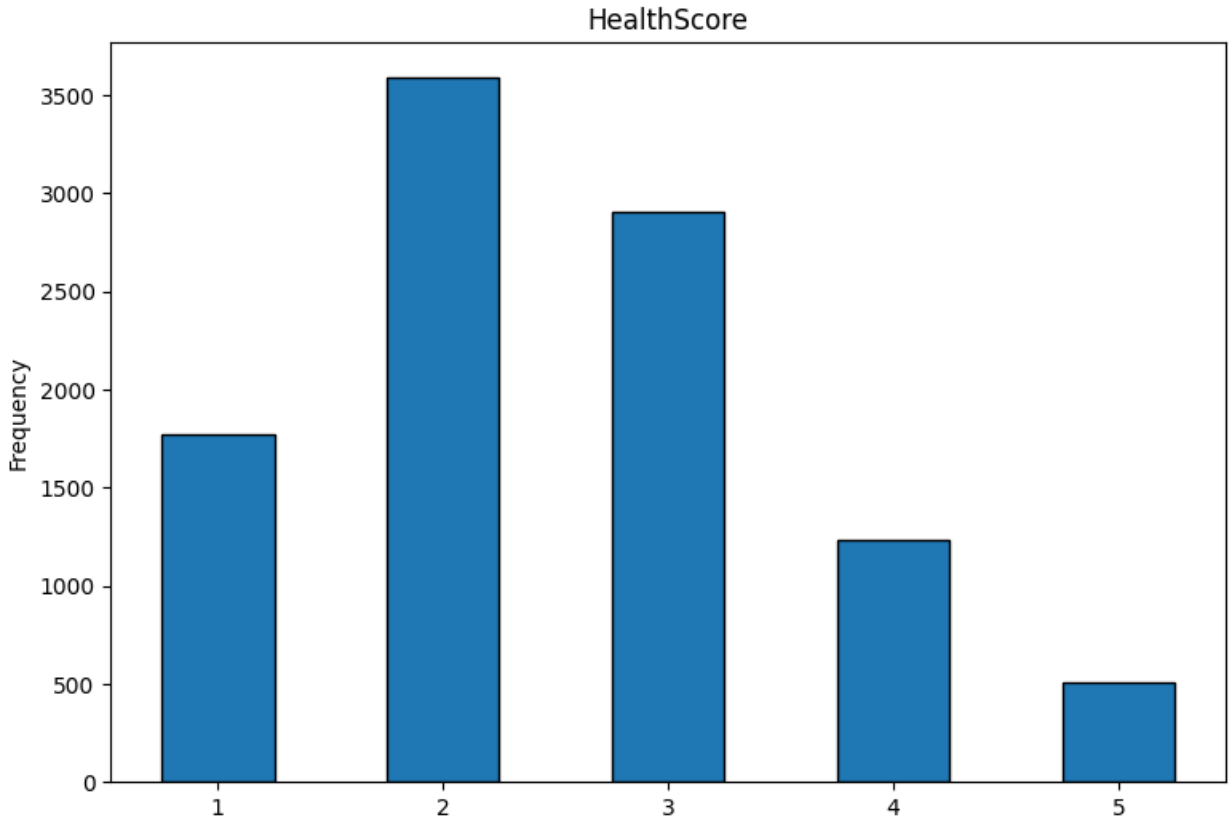

HighBP
Frequency
6000
5000
4000
3000
2000
1000
0
0
1

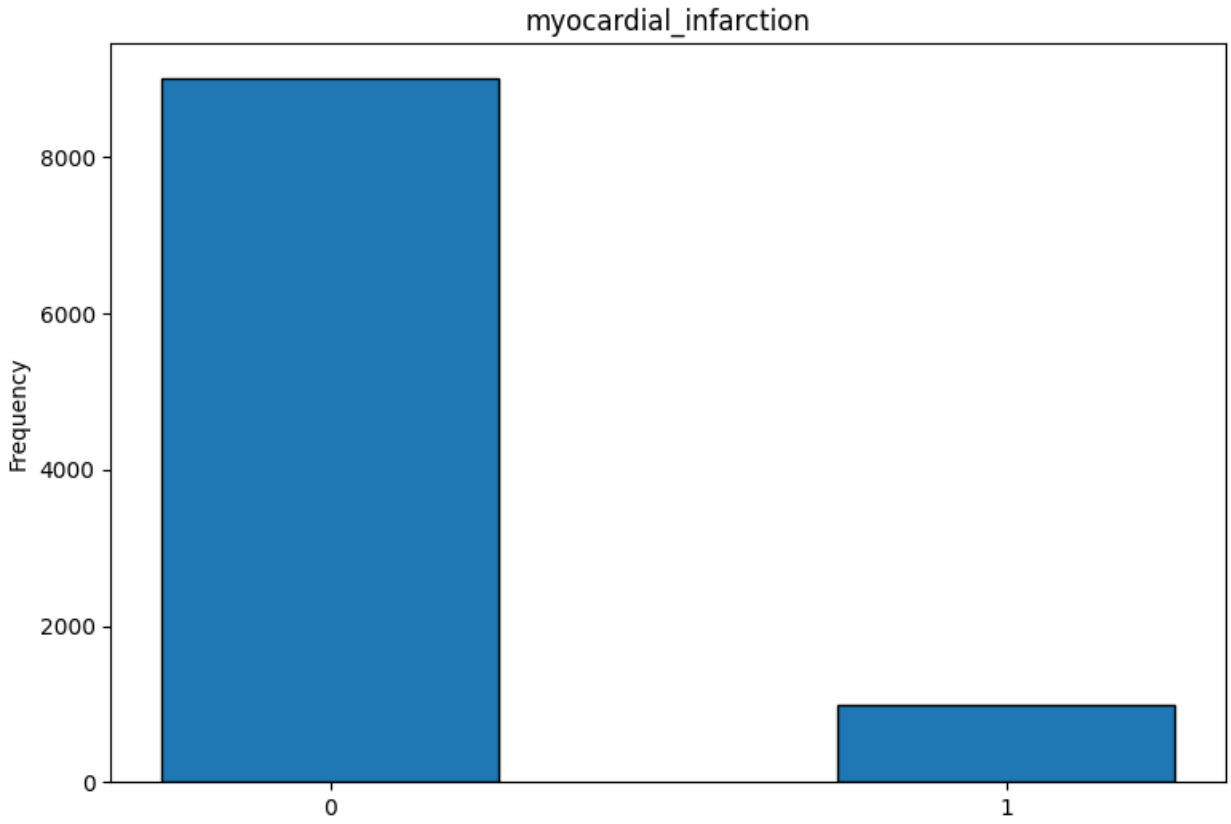

Unprocessed_fructose
Frequency
6000
5000
4000
3000
2000
1000
0
0
1

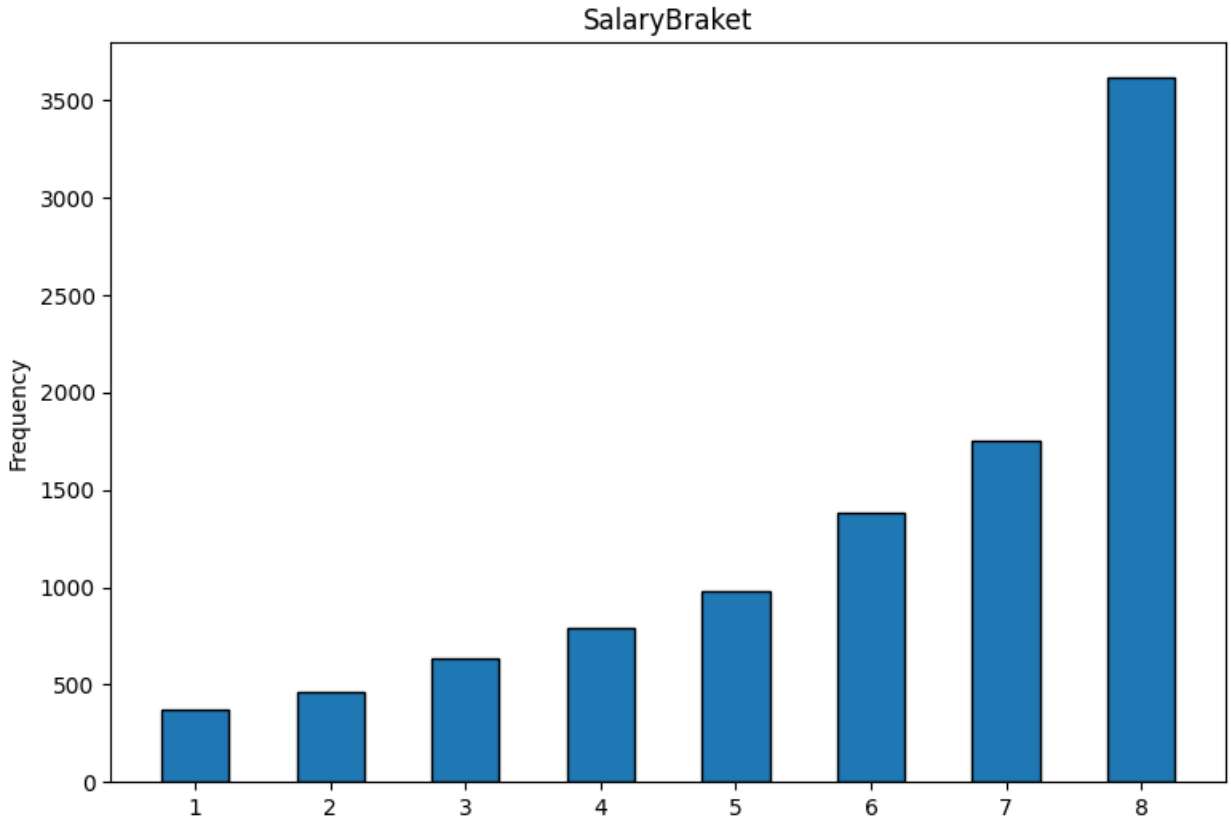

Jogging
Frequency
8000
7000
6000
5000
4000
3000
2000
1000
0
Rarely
Frequent

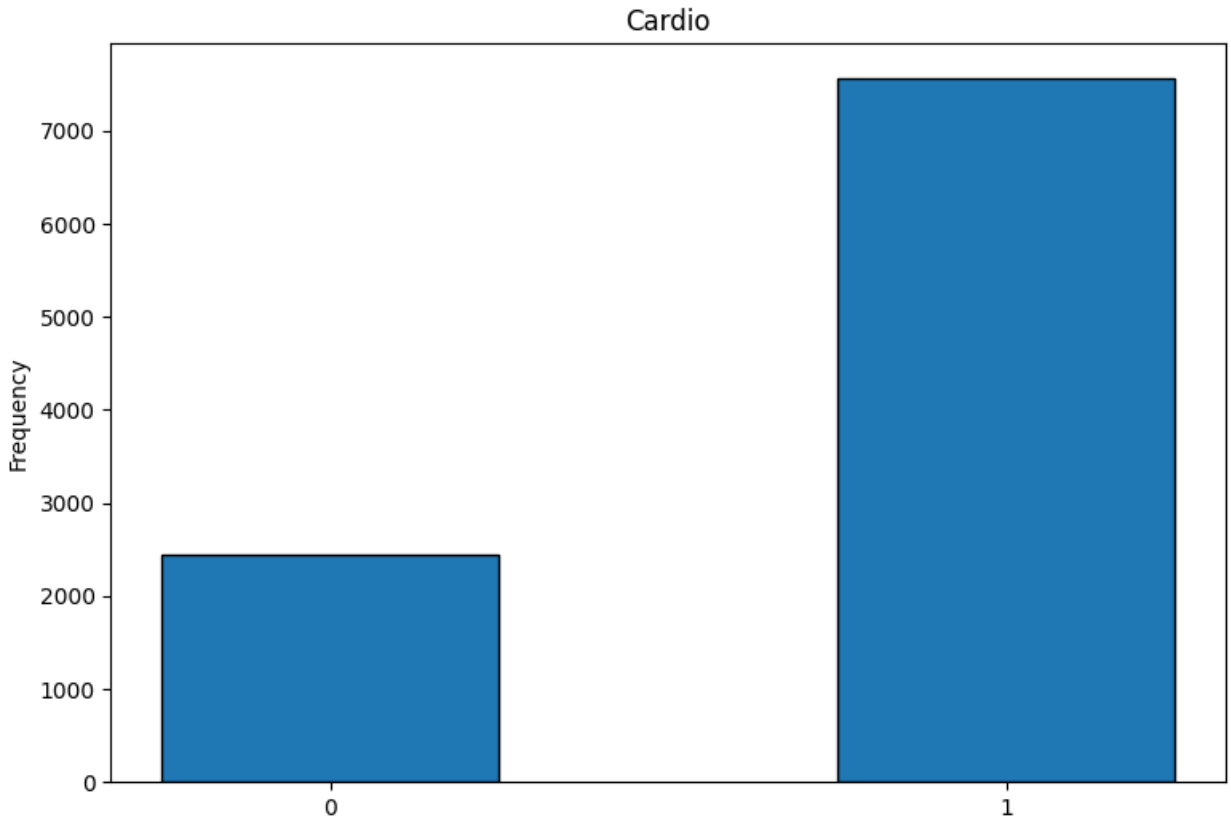

IncreasedChol
Frequency
5000
4000
3000
2000
1000
0
No
Yes

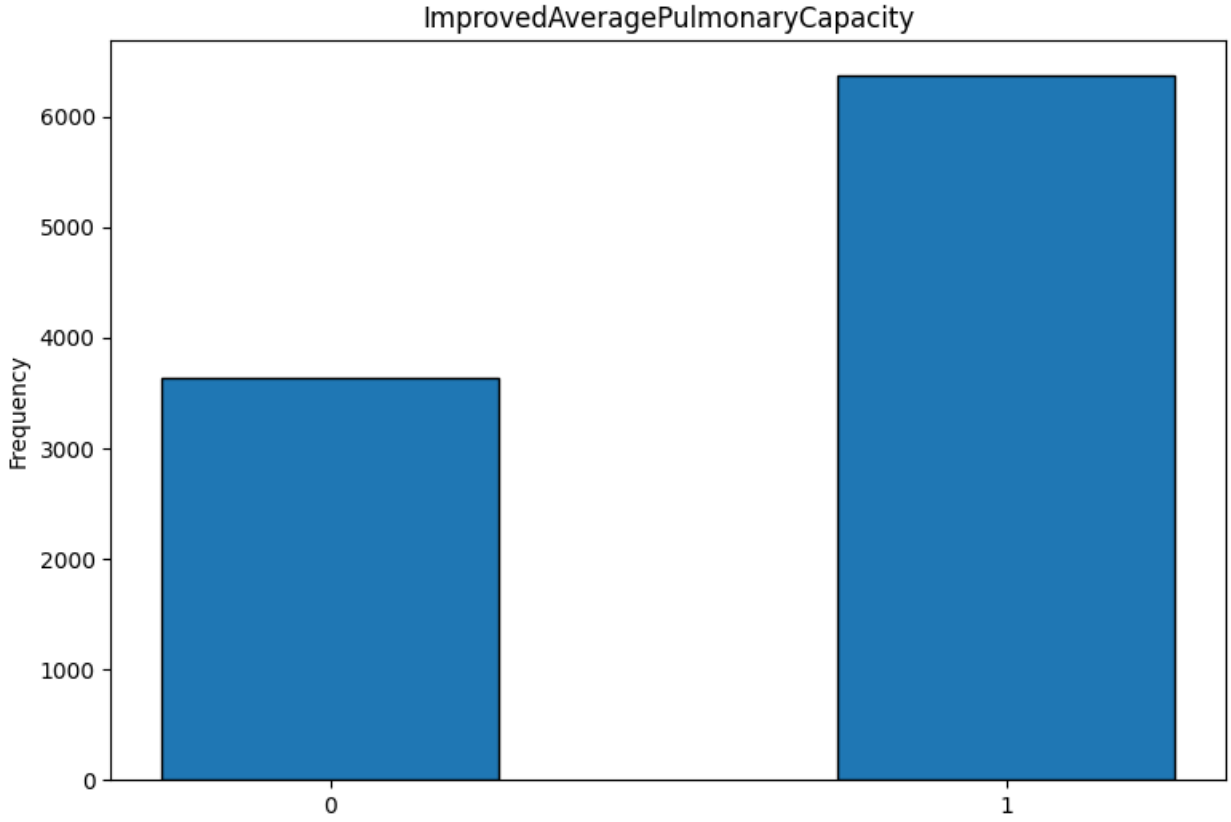

gender
Frequency
5000
4000
3000
2000
1000
0
Male
Female

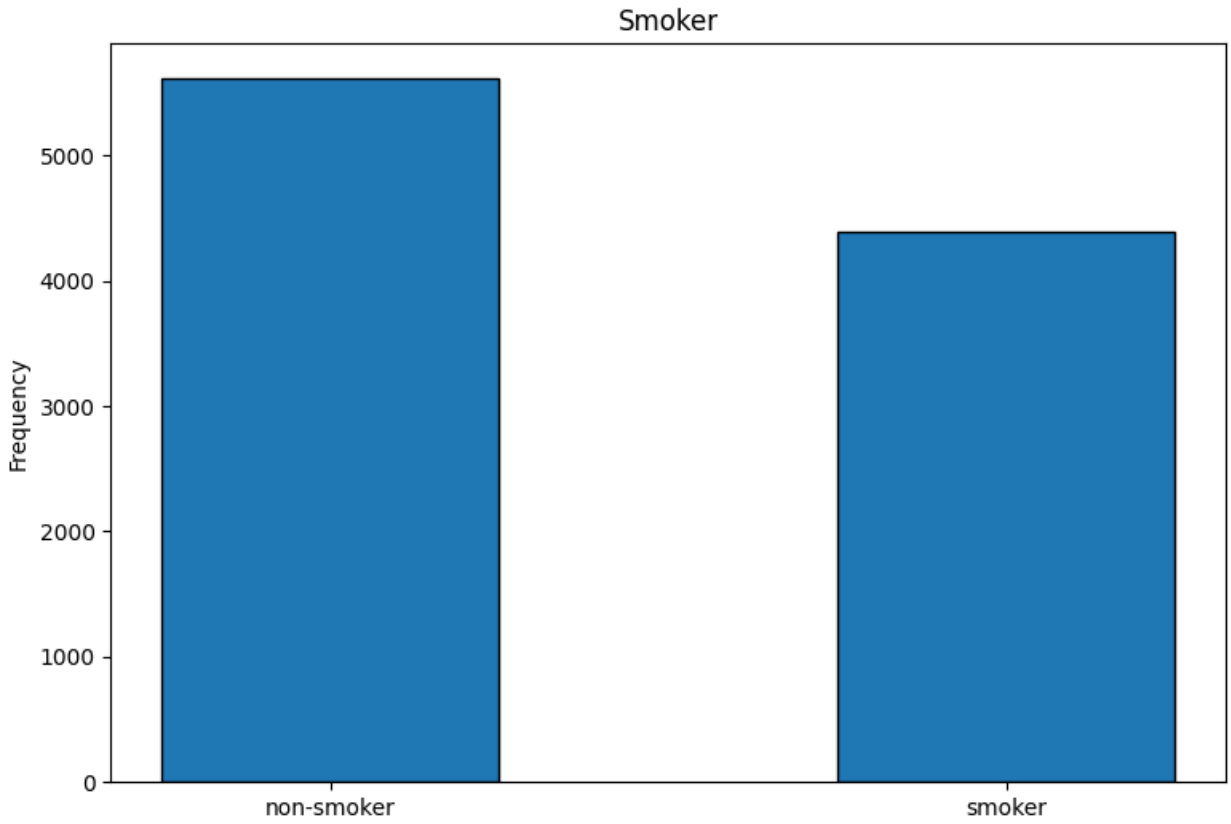

HealthScore
Frequency
3500
3000
2500
2000
1500
1000
500
0
1
2
3
4
5


myocardial_infarction
Frequency
8000
6000
4000
2000
0
0
1


SalaryBraket
Frequency
3500
3000
2500
2000
1500
1000
500
0
1
2
3
4
5
6
7
8


Cardio
Frequency
0
1000
2000
3000
4000
5000
6000
7000
0
1


ImprovedAveragePulmonaryCapacity
Frequency
0
1000
2000
3000
4000
5000
6000
0
1



Interpretare date:

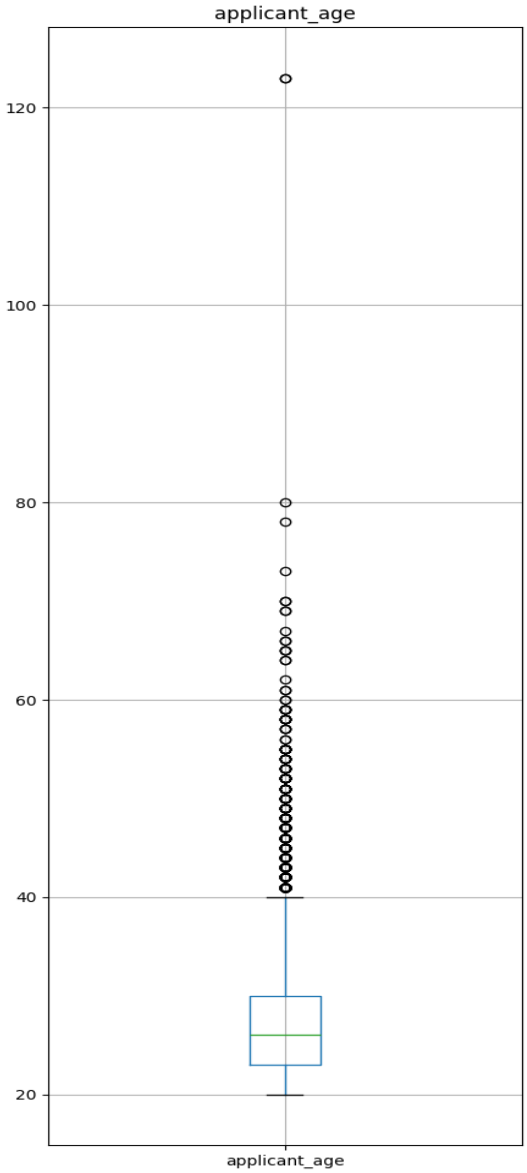
Atributele numerice din tabel indică o diversitate semnificativă într-o serie de măsurători relevante pentru pacienții diabetici. Datele furnizează o bază solidă pentru analiza ulterioară și modelarea cu algoritmi,oferind oportunități pentru identificarea de modele și tendințe semnificative în datele despre sănătate.Totusi exista unele valori destul de mari,care necesita o investigatie atenta.

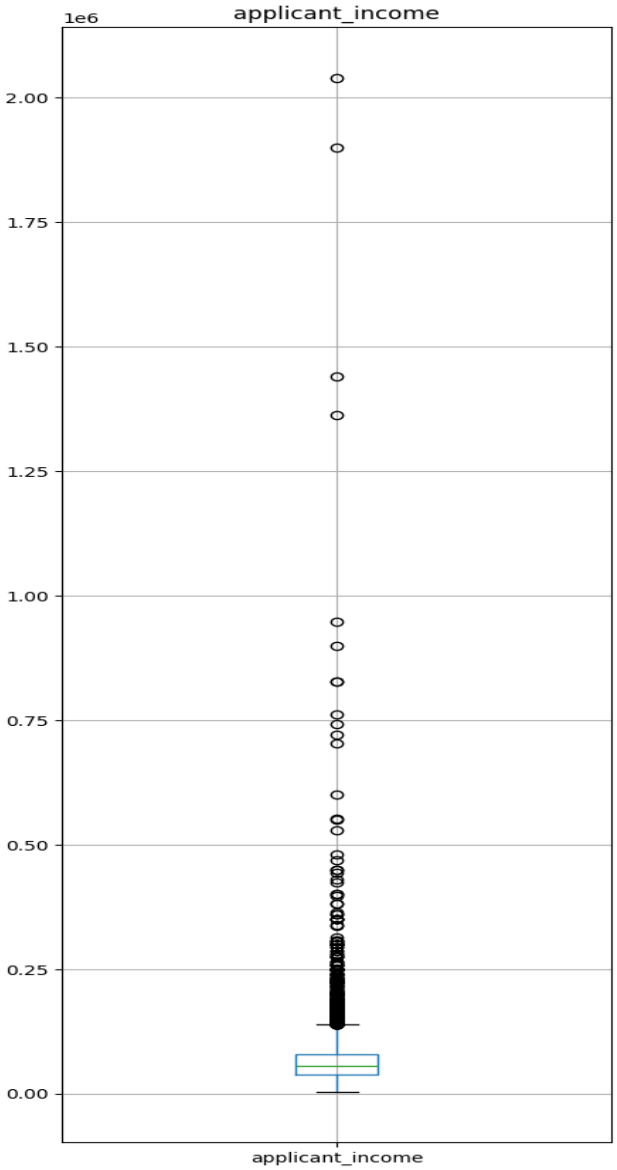
Tabelul si histrogramele pentru atributele discrete indică că majoritatea acestora sunt caracterizate de un număr mic de valori distincte. De asemenea, prezența unei frecvențe extreme la aceste valori sugerează o distribuție neuniformă a datelor, ceea ce poate afecta în mod semnificativ performanța și generalizarea algoritmilor de învățare automata.

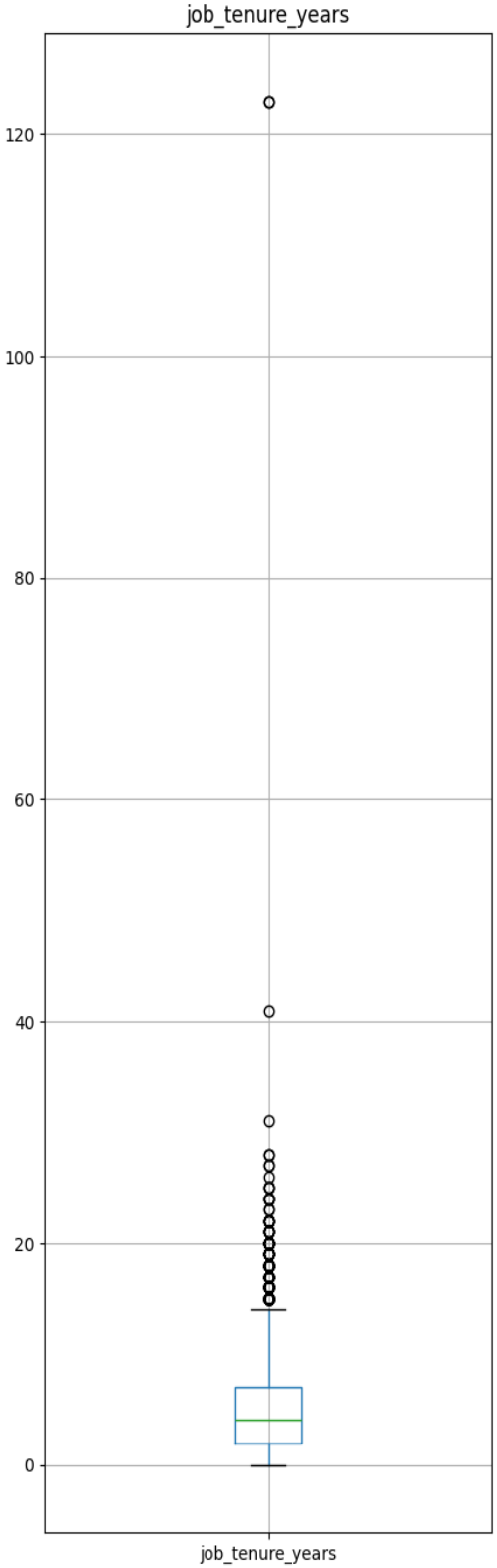
Risc de Creditare

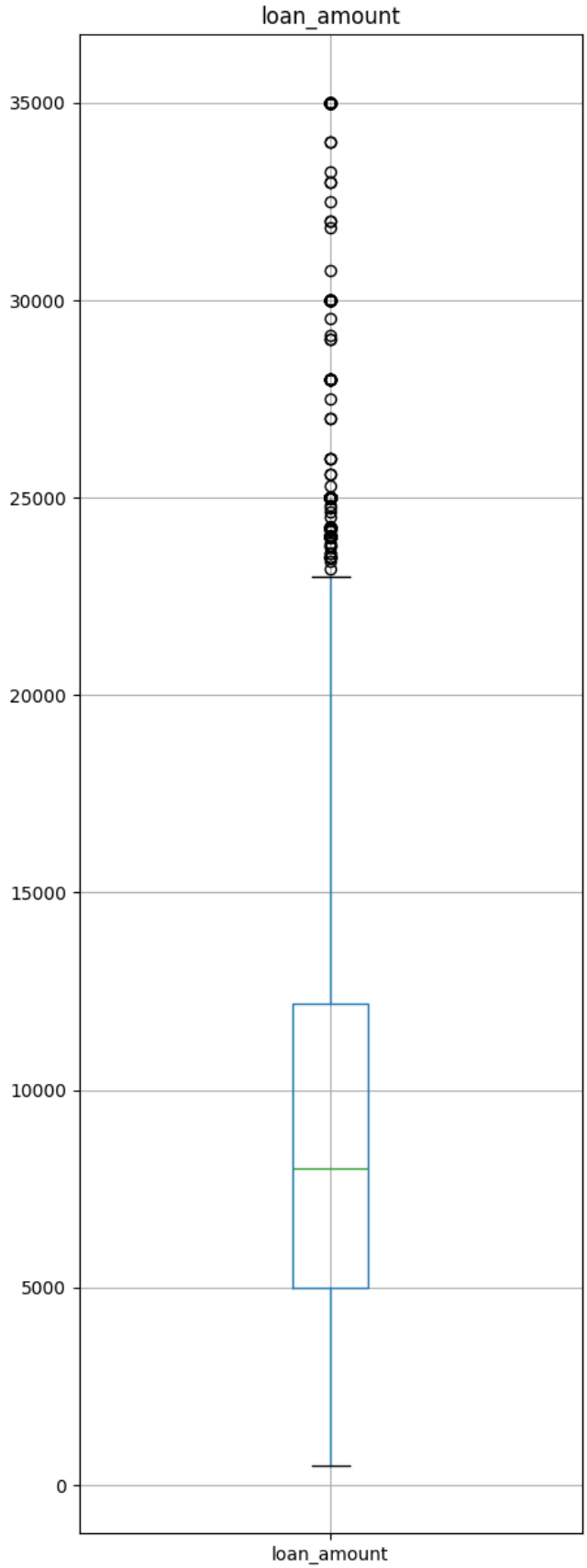
- **Atribute numerice:**

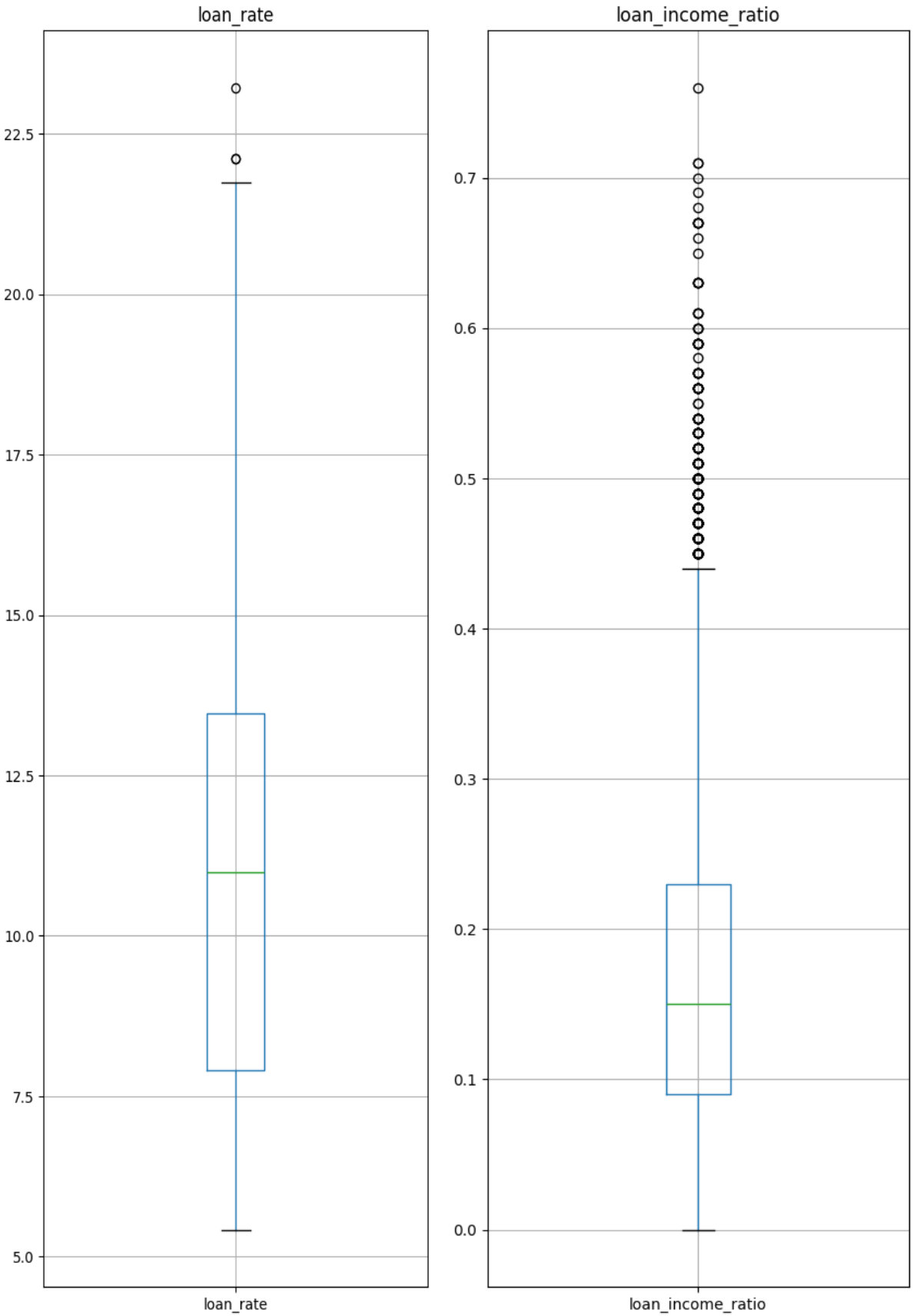
	applicant_age	applicant_income	job_tenure_years	loan_amount	loan_rate	loan_income_ratio	credit_history_length_years	credit_history_length_months
count	10000.000000	1.000000e+04	9736.000000	10000.000000	9060.000000	10000.000000	10000.000000	10000.000000
mean	27.745100	6.573421e+04	4.785744	9568.037500	11.007179	0.170130	5.811100	75.760700
std	6.360155	5.694439e+04	4.353122	6350.431581	3.266393	0.106814	4.050217	48.677362
min	20.000000	4.200000e+03	0.000000	500.000000	5.420000	0.000000	2.000000	25.000000
25%	23.000000	3.859500e+04	2.000000	5000.000000	7.900000	0.090000	3.000000	41.000000
50%	26.000000	5.500000e+04	4.000000	8000.000000	10.990000	0.150000	4.000000	57.000000
75%	30.000000	7.899700e+04	7.000000	12200.000000	13.470000	0.230000	8.000000	102.000000
max	123.000000	2.039784e+06	123.000000	35000.000000	23.220000	0.760000	30.000000	369.000000

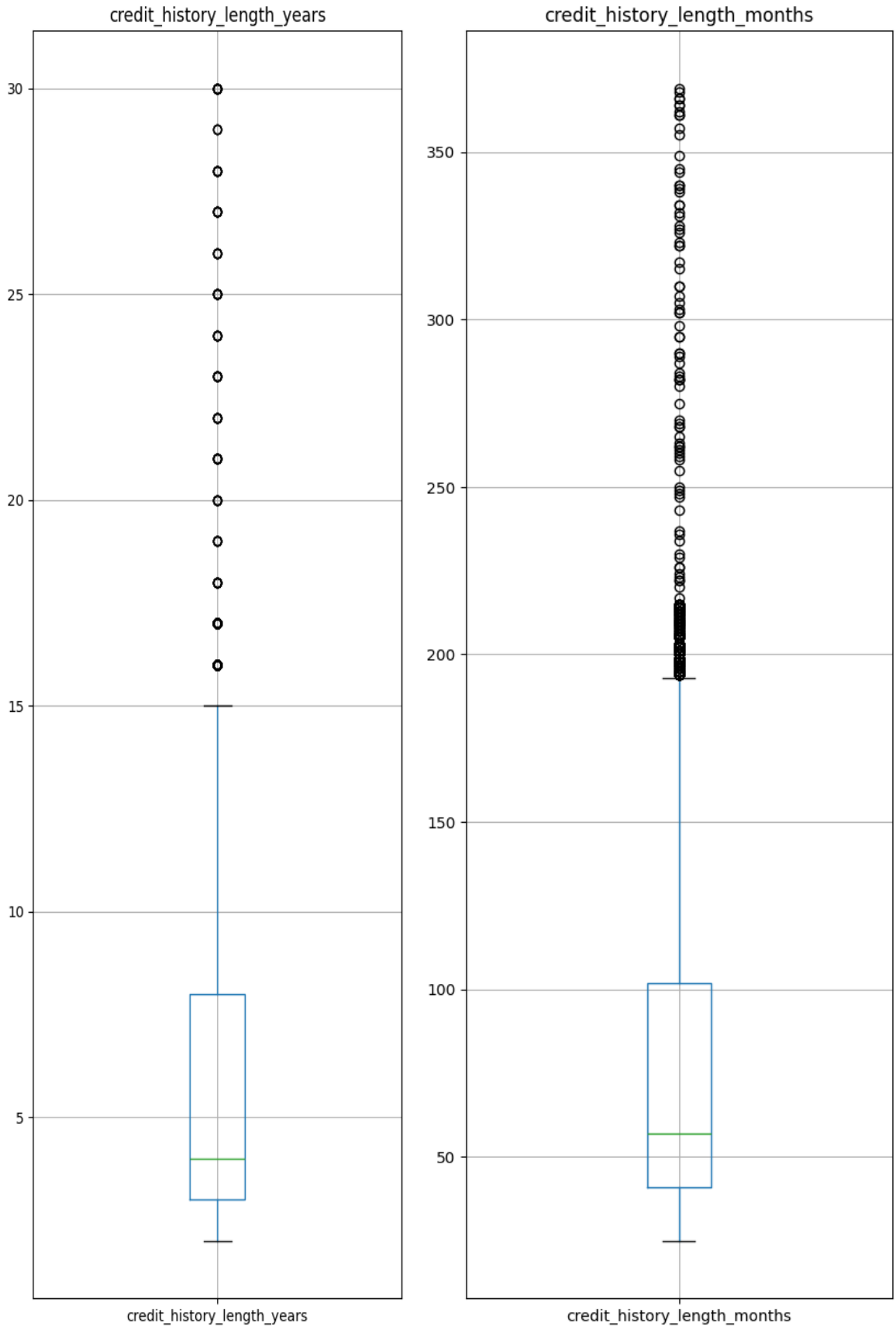





job_tenure_years
120
100
80
60
40
20
0
job_tenure_years

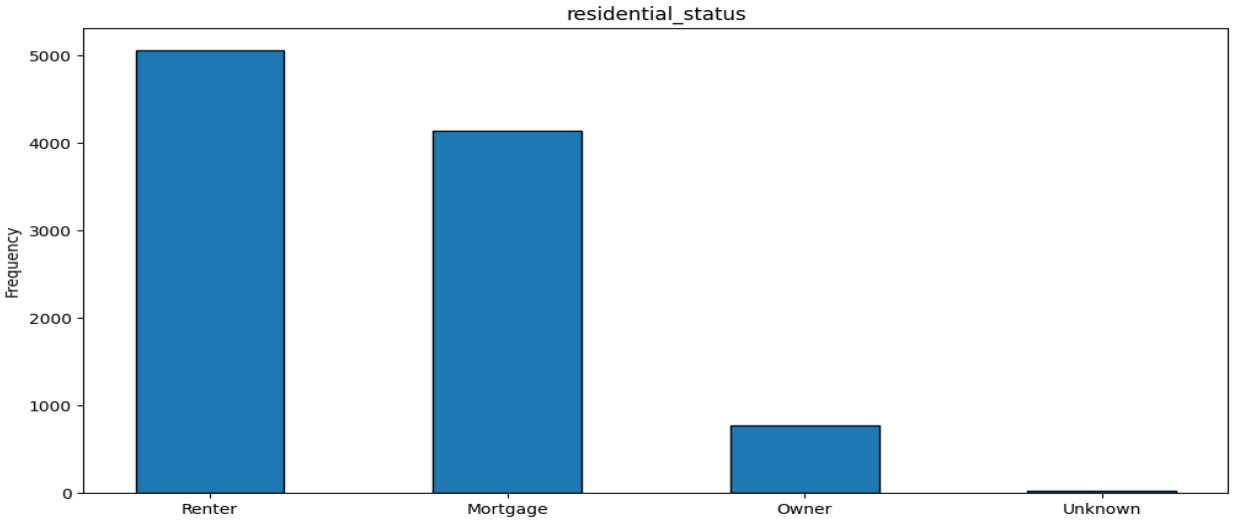

loan_amount
35000
30000
25000
20000
15000
10000
5000
0
loan_amount

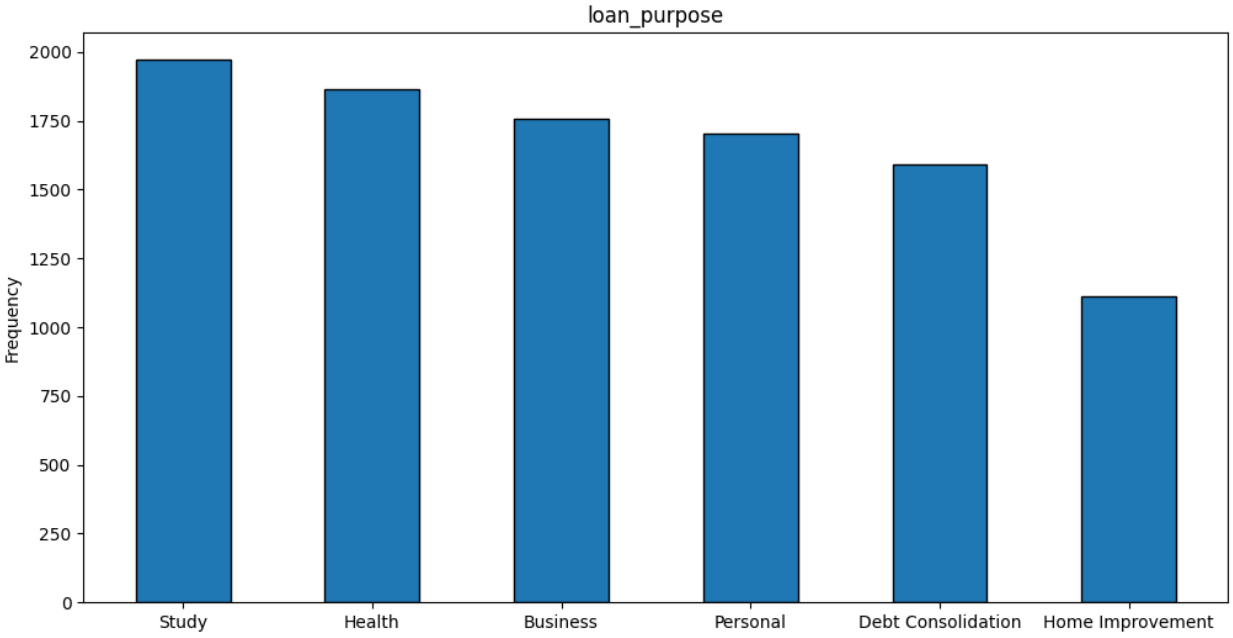

loan_rate
22.5
20.0
17.5
15.0
12.5
10.0
7.5
5.0
loan_rate
loan_income_ratio
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
loan_income_ratio

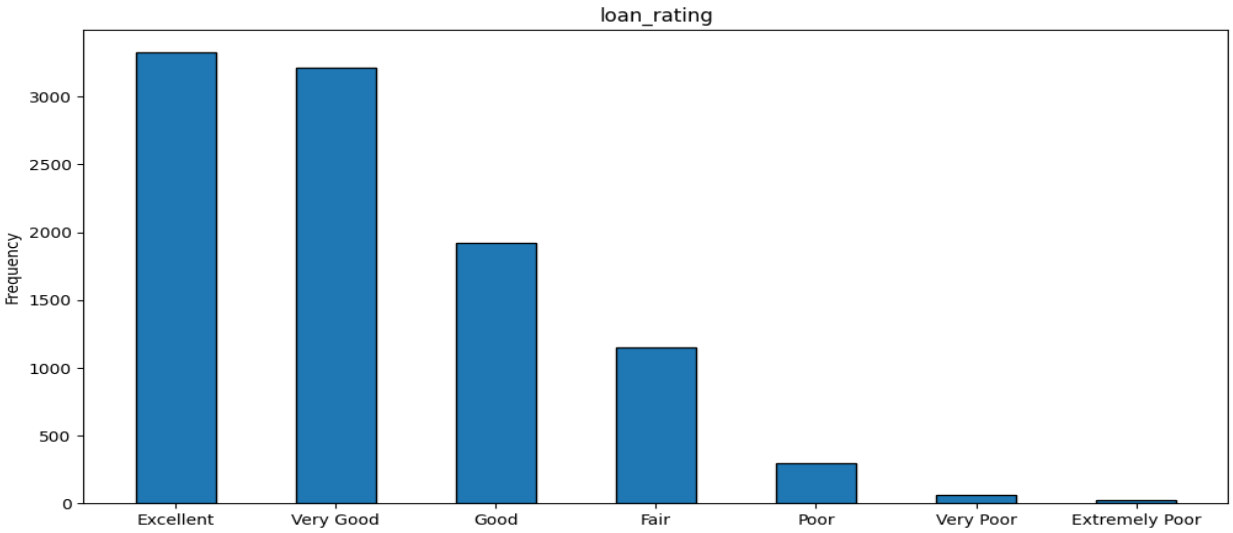

credit_history_length_years
30
25
20
15
10
5
credit_history_length_years
credit_history_length_months
350
300
250
200
150
100
50
credit_history_length_months

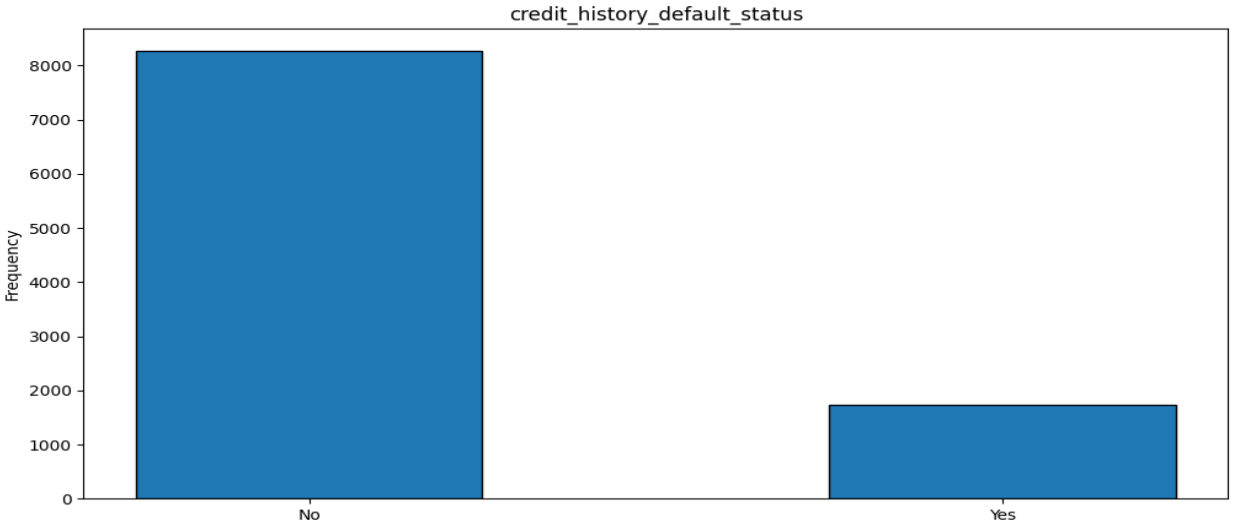
- **Discret atributes:**

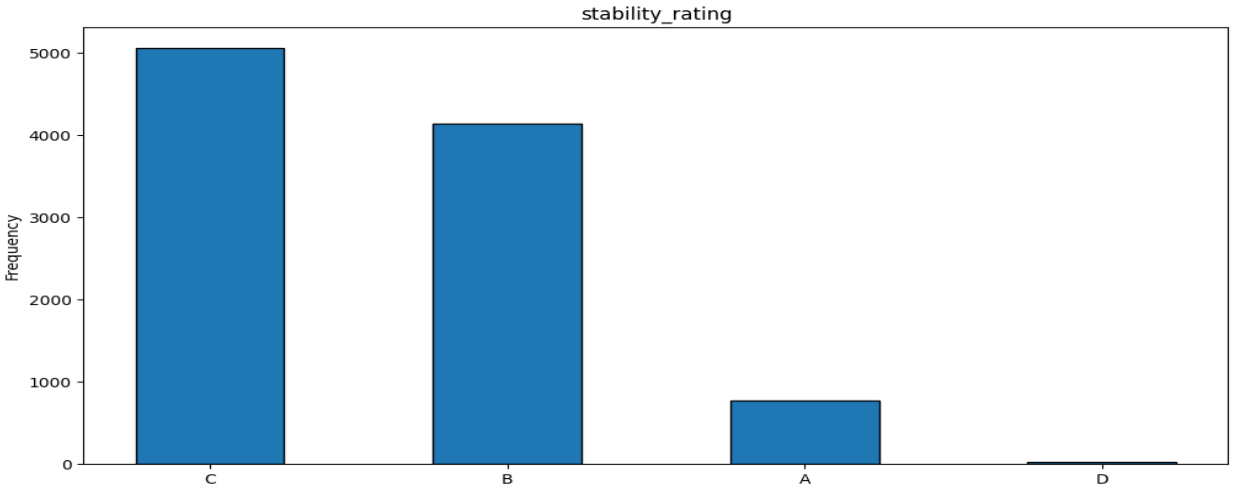
	Attribute	Non-missing Count	Unique Values Count
0	residential_status	10000	4
1	loan_purpose	10000	6
2	loan_rating	10000	7
3	credit_history_default_status	10000	2
4	stability_rating	10000	4






loan_rating
Frequency
0
500
1000
1500
2000
2500
3000
Excellent
Very Good
Good
Fair
Poor
Very Poor
Extremely Poor


credit_history_default_status
Frequency
0
1000
2000
3000
4000
5000
6000
7000
8000
No
Yes


stability_rating
Frequency
0
1000
2000
3000
4000
5000
C
B
A
D

Interpretare date:

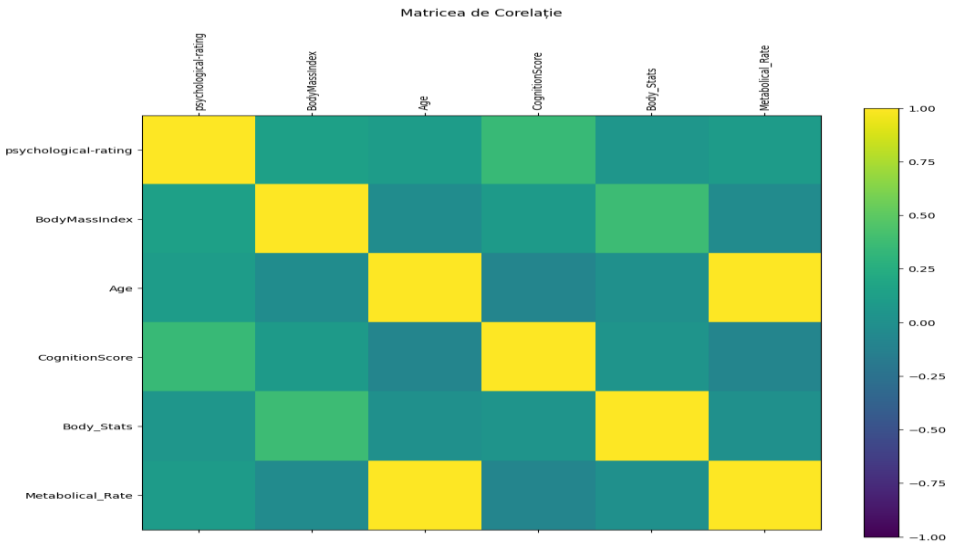
Tabelul si histrogramele pentru atributele numerice prezinta o variate și o gamă largă de valori pentru fiecare atribut. . Totuși, există unele aspecte care ar putea necesită investigație suplimentară, cum ar fi vârstele extreme, veniturile înalte și scăzute și lungimile neobișnuite ale istoricului de credit.

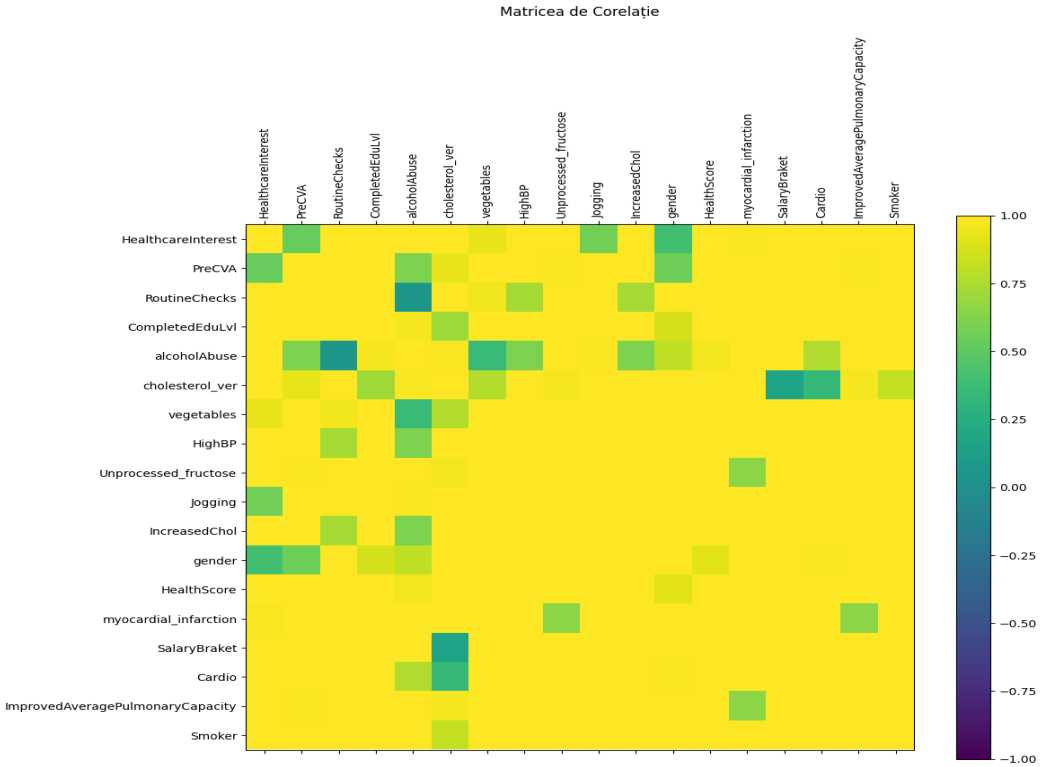
Atributele discrete par să fie mai diversificate și să ofere o perspectivă mai cuprinzătoare asupra evaluării riscului de credit în comparație cu setul precedent. . Cu toate acestea, numărul redus de valori unice pentru unele atribute și distribuția inegală a acestora ar putea să nu ofere o imagine completă sau echilibrată.

Cerinta 3.3

Diabet

Preprocesarea datelor:





- Pentru datele care lipsesc, am folosit SimpleImputer cu strategia “mean” pentru atributele numerice si strategia “most frequent” pentru atributele discrete.
- Pentru datele extreme am folosit recomandarea din anexa cu setarea quantilelor intre 0.25 si 0.75
- Pentru a standardiza toate atributele numerice, am folosit StandardScaler.

Evaluarea algortimilor:

- ## RandomForest

Sklearn:

Am folosit RandomizedSearchCV pentru a face tunning la hyperparametrii. Astfel, am obtinut o referinta despre cum ar trebui sa arate parametrii, facand, la final, doar niste mici ajustari.

RandomForestClassifier(n_estimators=100, max_depth=15,min_samples_leaf = 50,max_leaf_nodes=5, criterion='gini', class_weight='balanced')

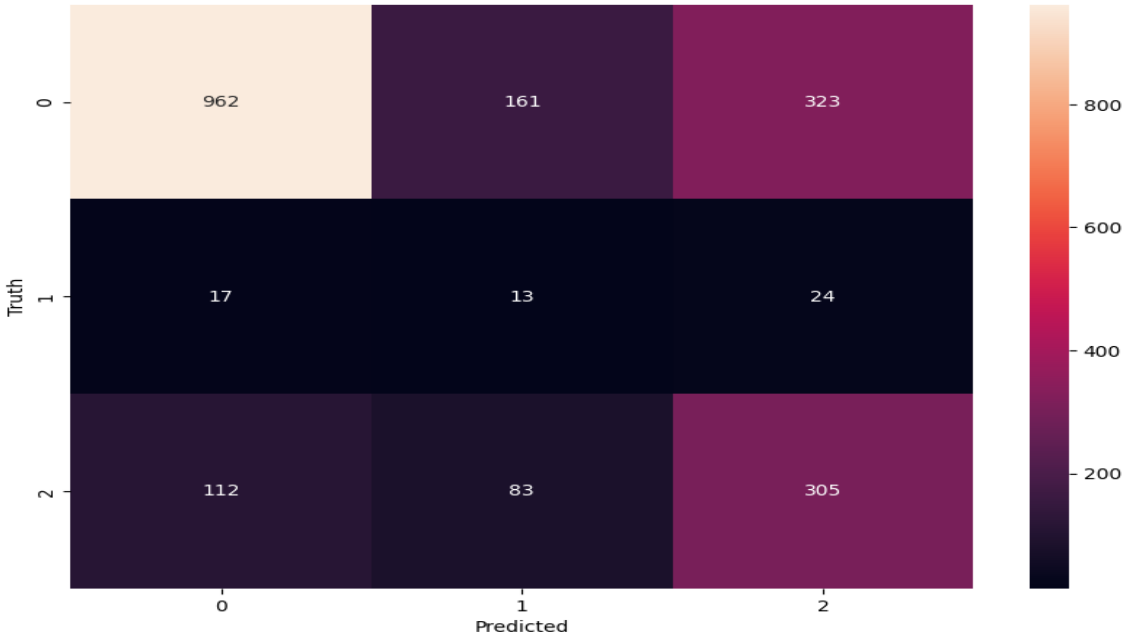
100 de arbori de decizie, fiecare având o adâncime maximă de 15, un minim de 50 de eșantioane necesare pentru a fi într-un nod frunză, și un maxim de 5 noduri frunză pe arbore. Arborii utilizează criteriul Gini pentru divizarea nodurilor, iar claselor sunt echilibrate pentru a aborda dezechilibrul de clasă în setul de date.

Cel mai important parametru setat a fost class_weight=’balanced’, deoarece,fara el precizia,recall,F1 -score ul si matricea de confuzie ar fi dat rezultate destul de slabe, intrucat clasele din dataset nu sunt balansate,majoritar fiind pentru clasa 0.

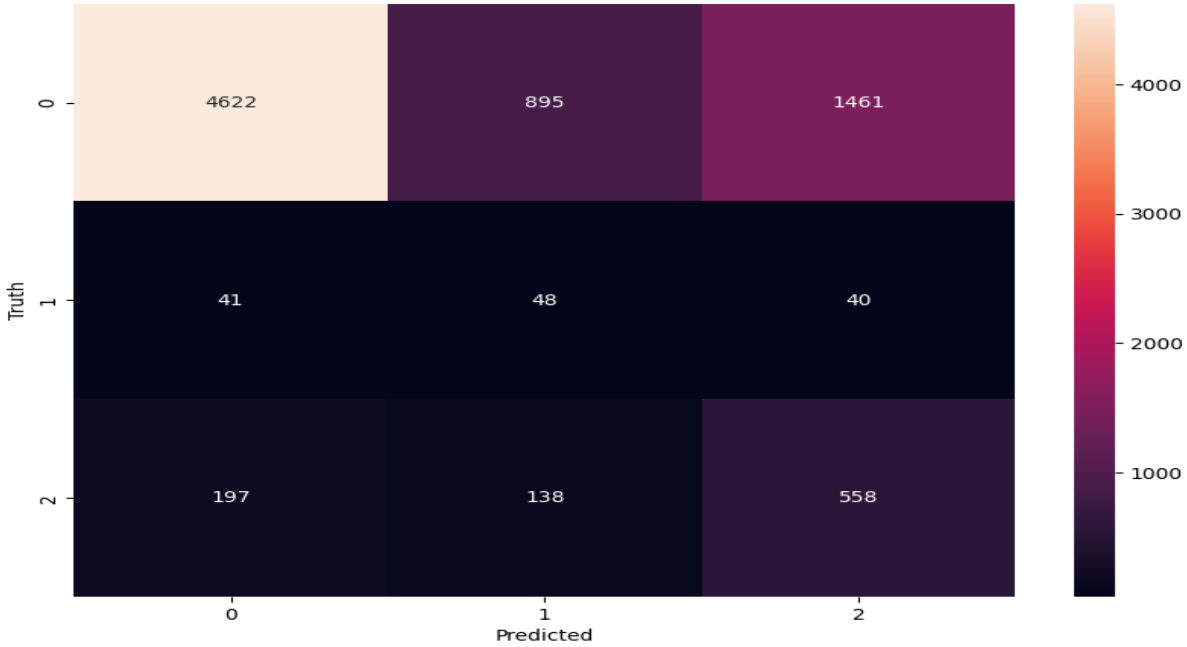
Restul, au fost aleși pentru a optimiza performanța modelului RandomForestClassifier

Astfel, s a obtinut:

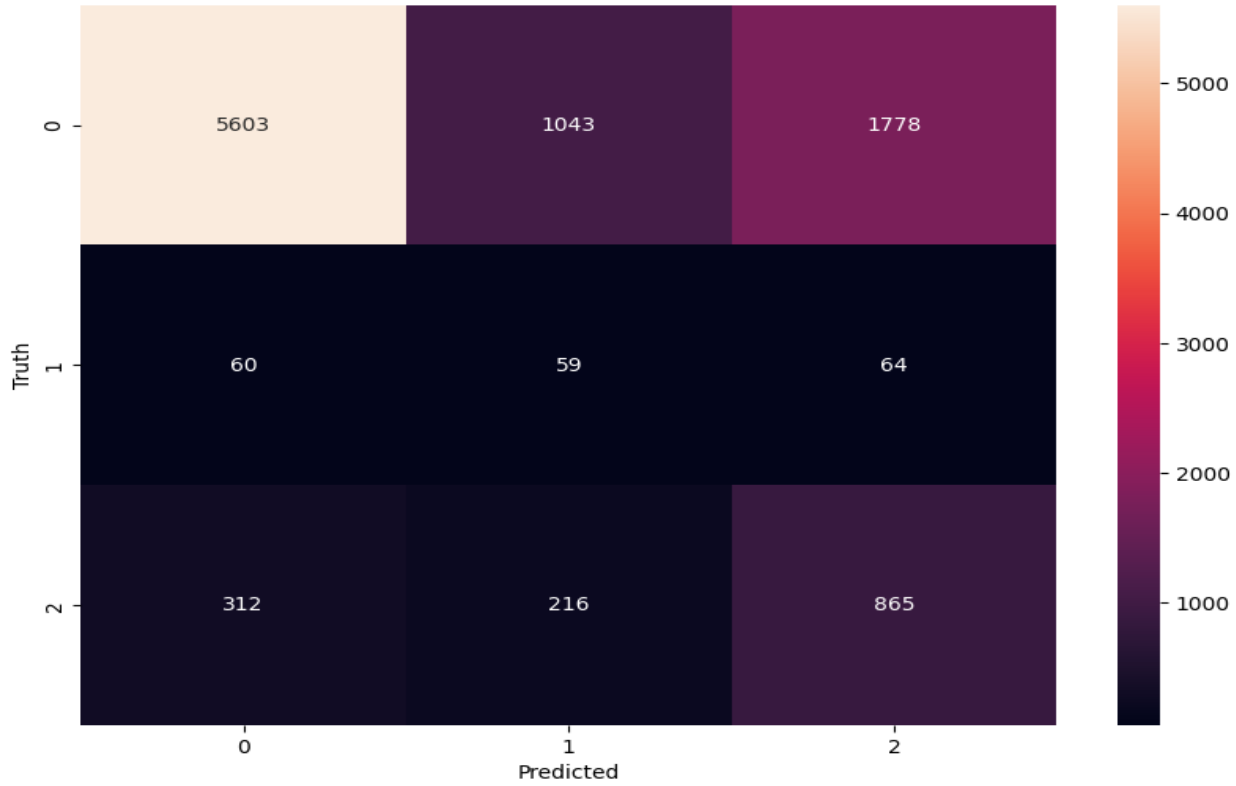
Test:



Train:



Full:



Se poate observa cum modelul obtine rezultate foarte bune pentru clasa 0,rezultate satisfacatoare pentru clasa 2 si rezultate proaste pentru 1,datorita dezechilibrului claselor din dataset.

Implemenatare manuala:

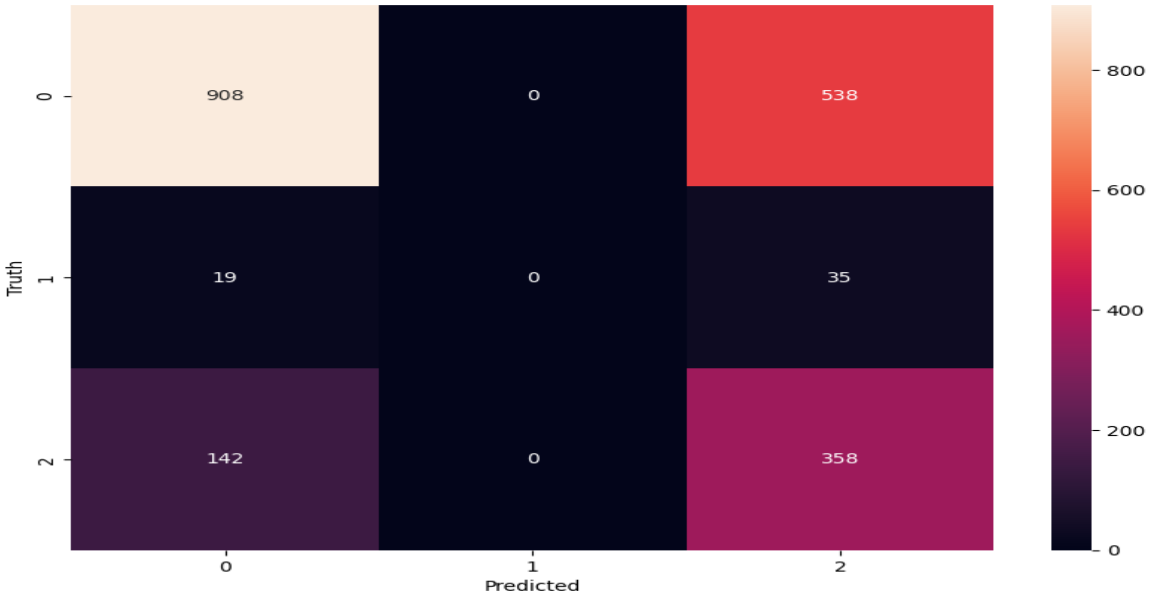
M am folosit de functia “Smote” pentru a face resampling pe datele din train care erau dezechilibrate pentru a putea face o comparatie echilibrata intra algoritmul implementat manual si cel cu sklearn.Deoarece timpul de rulare este destul de mare, nu am avut ocazia sa fac foarte mult tunning pe hyperparametrii,ajungand la solutia finala :

```
    (n_estimators=10,max_depth=5, min_samples_per_node=31)
```

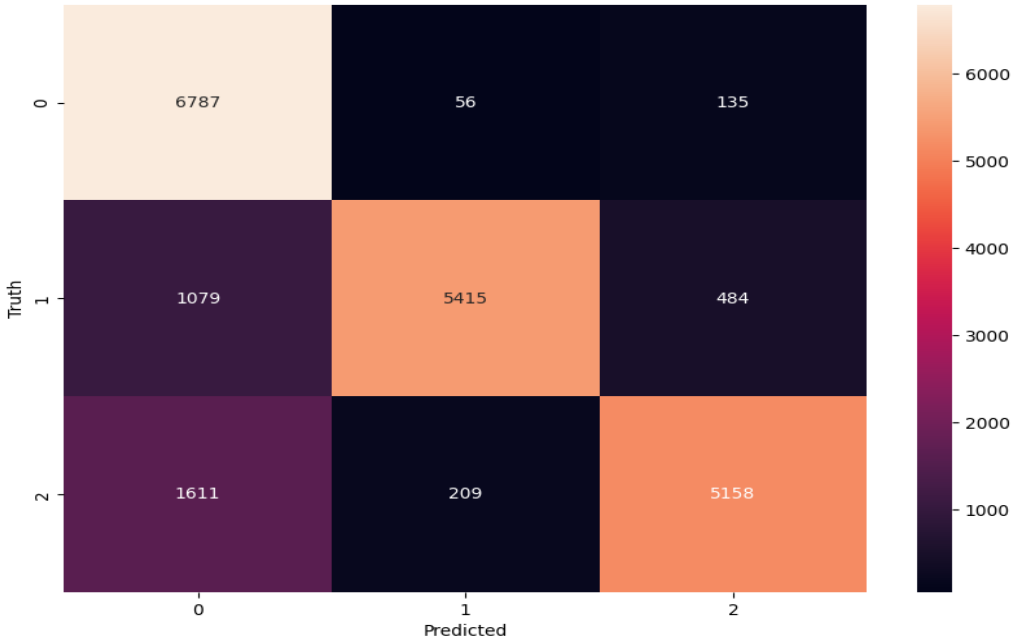
10 de arbori de decizie, fiecare având o adâncime maximă de 5, un minim de 31 de eșantioane necesare pentru a fi într-un nod frunză. Arborii utilizează criteriul Gini pentru divizarea nodurilor.

S a obtinut:

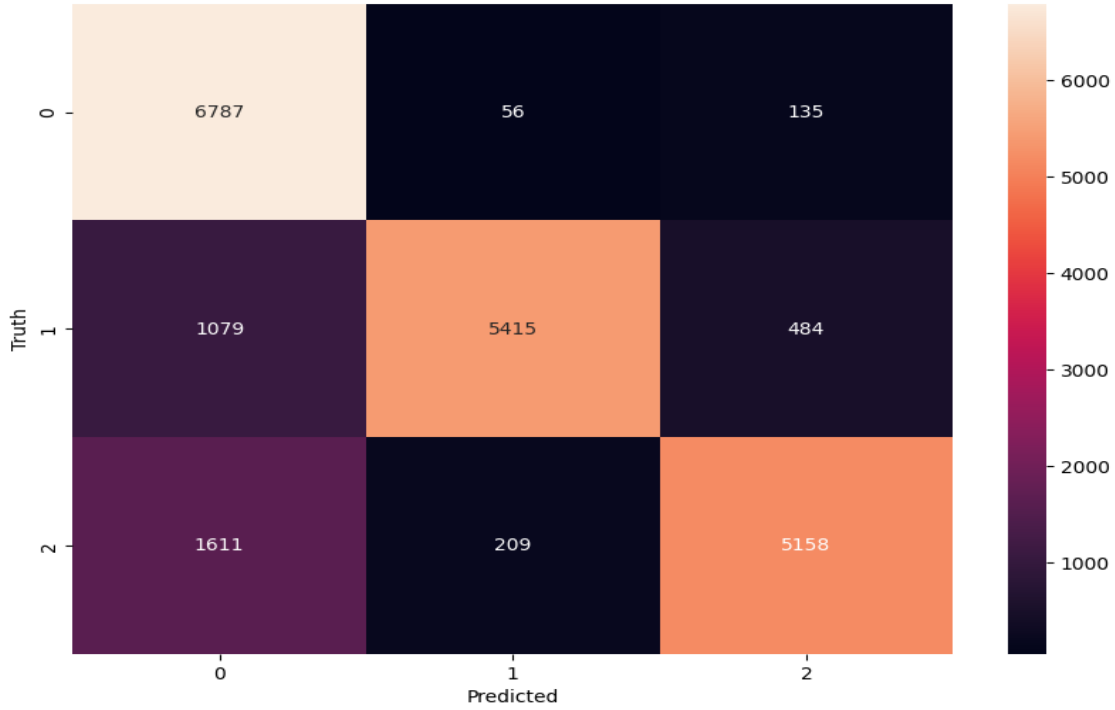
Test:



Train:



Full:



Se poate observa ca algoritmul prezice rezultate ok pentru clasele 0 si 2.Apar rezultate extrem de slabe pentru clasa 1,desi am incercat sa echilibrez clasele.

- **Mlp:**

Sklearn:

La fel ca si la RandomForest,m am folosit de RandomizedSearchCV pentru a seta hyperparametrii,obtinand:

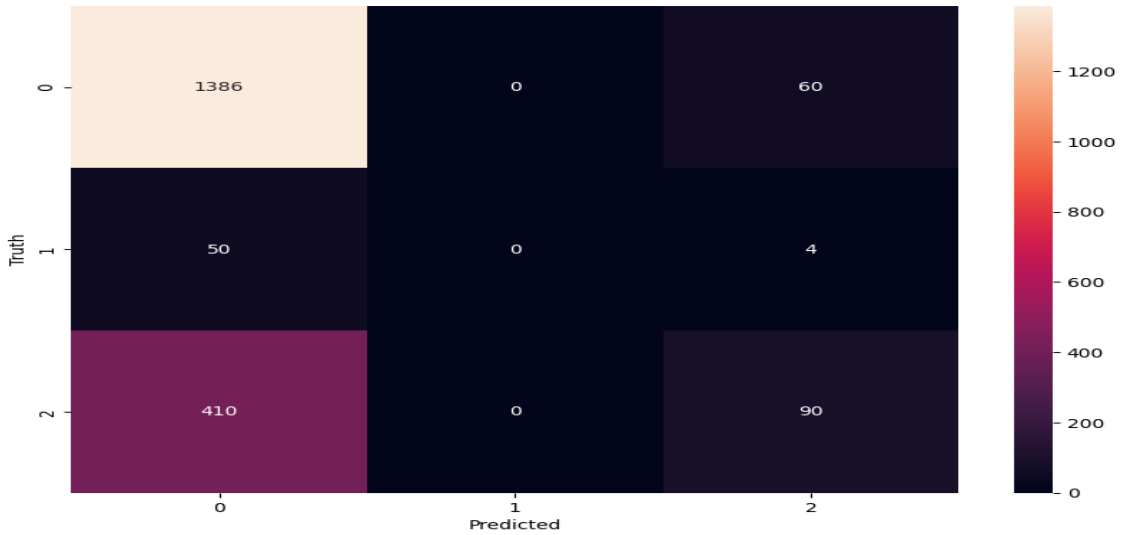
```
      (activation='relu', alpha=0.01, batch_size=32, hidden_layer_sizes=(256,),
learning_rate='adaptive', max_iter=500, solver='sgd')
```

Folosesc funcția de activare 'relu', un termen de regularizare L2 alpha de 0.01, un batch size de 32, un singur strat din rețea de dimensiune 256, un algoritm de învățare adaptivă, 500 de epoci de antrenare și un optimizator 'sgd' (Stochastic Gradient Descent).

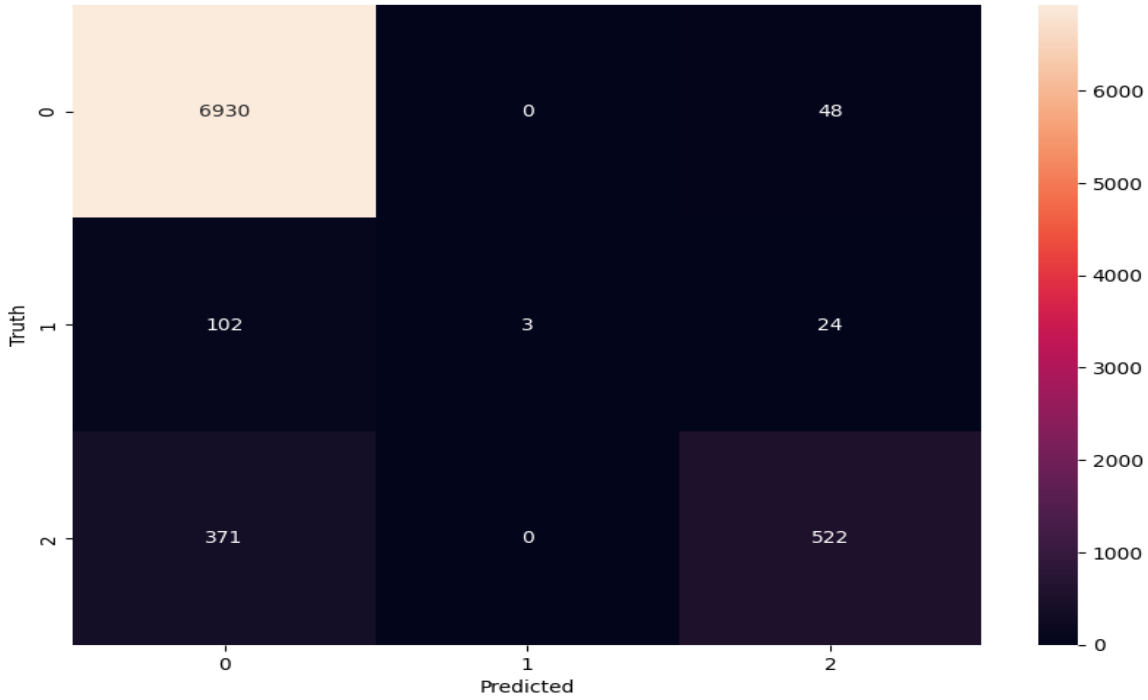
Am ales sa nu folosesc early_stopping, deoarece primeam un rezultat extrem de slab. Restul, au fost aleși pentru a optimiza performanța.

Astfel, s a obtinut:

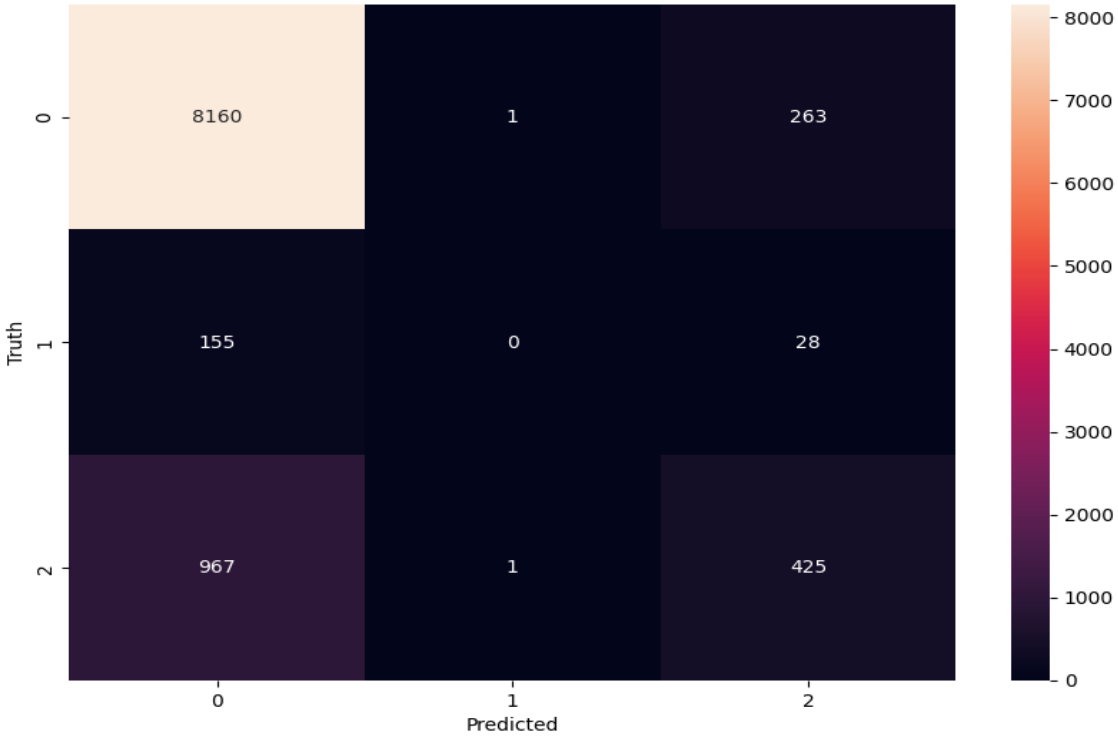
Test:



Train:



Full:



Se poate observa cum pe setul de test,rezulatetele sunt proaste,datorita prezentei dezechilibrului intre clase,majoritatea preziceriilor fiind doar pentru 0.

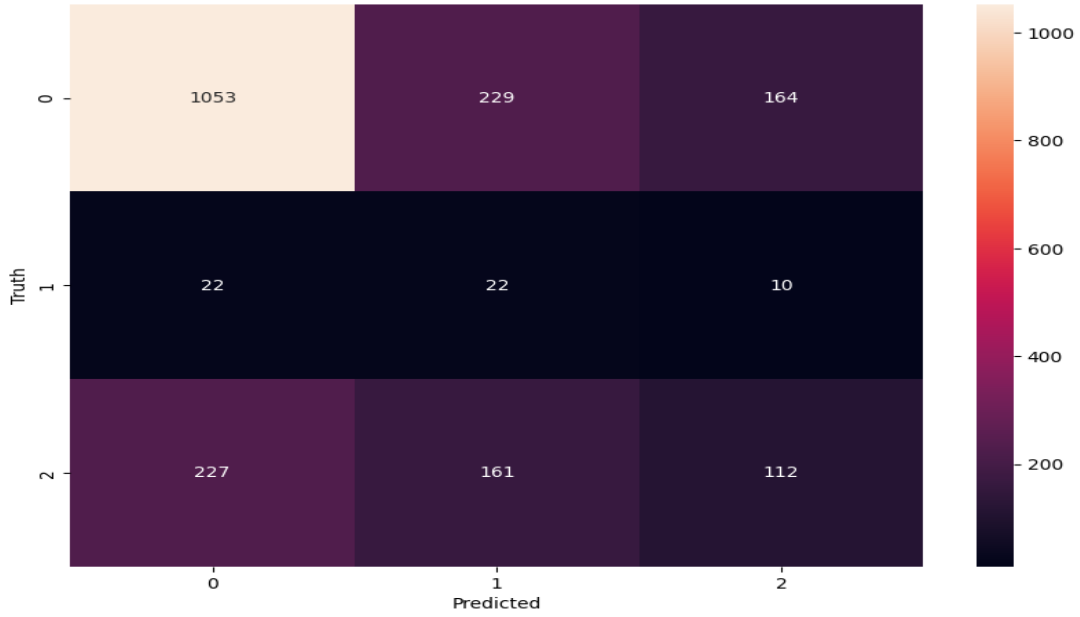
Pe celalalte seturi de date,se poate observa rezultate slabe pentru clasele 0 si 2.

Implementare manuala:

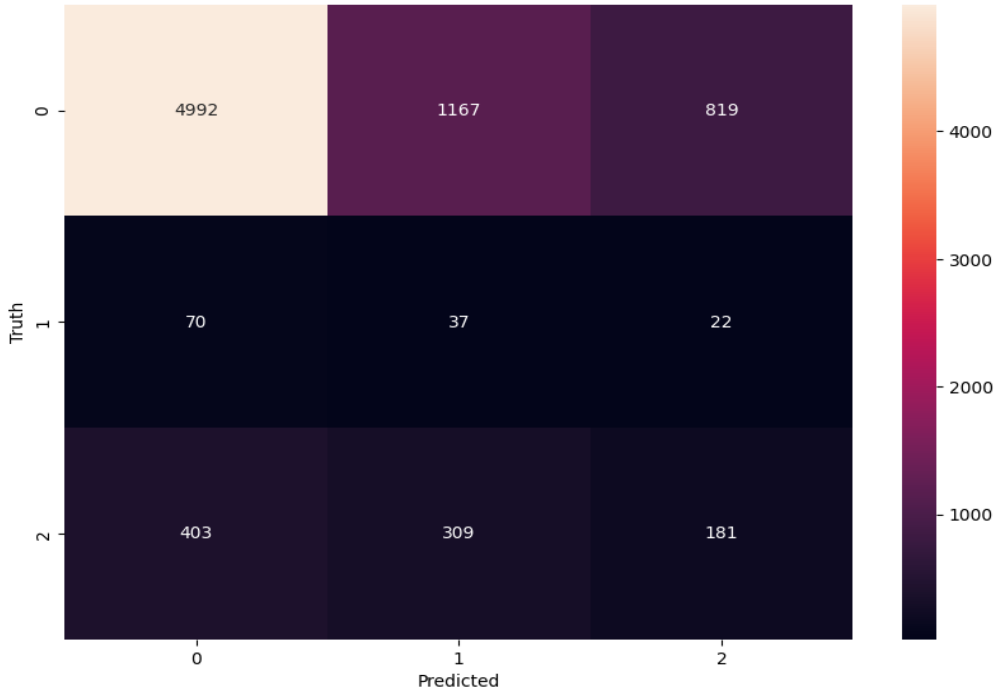
Pentru hyperparametrii, am pornit de la unii asemenatori cu modelul sklearn,unde i am mai retusat pentru a obtine o precizie mai buna. Am facut compromisul,de a alege un batch size mai mare care,imi da o acuratete mai mica pe teste si train,dar primesc un f1 score mai mare. Astfel:

Folosesc funcția de activare 'relu', un termen de regularizare L2 alpha de 0.01, un batch size de 512, doua straturi din rețea de dimensiune 128, 10 de epoci de antrenare și un optimizator 'sgd' (Stochastic Gradient Descent).

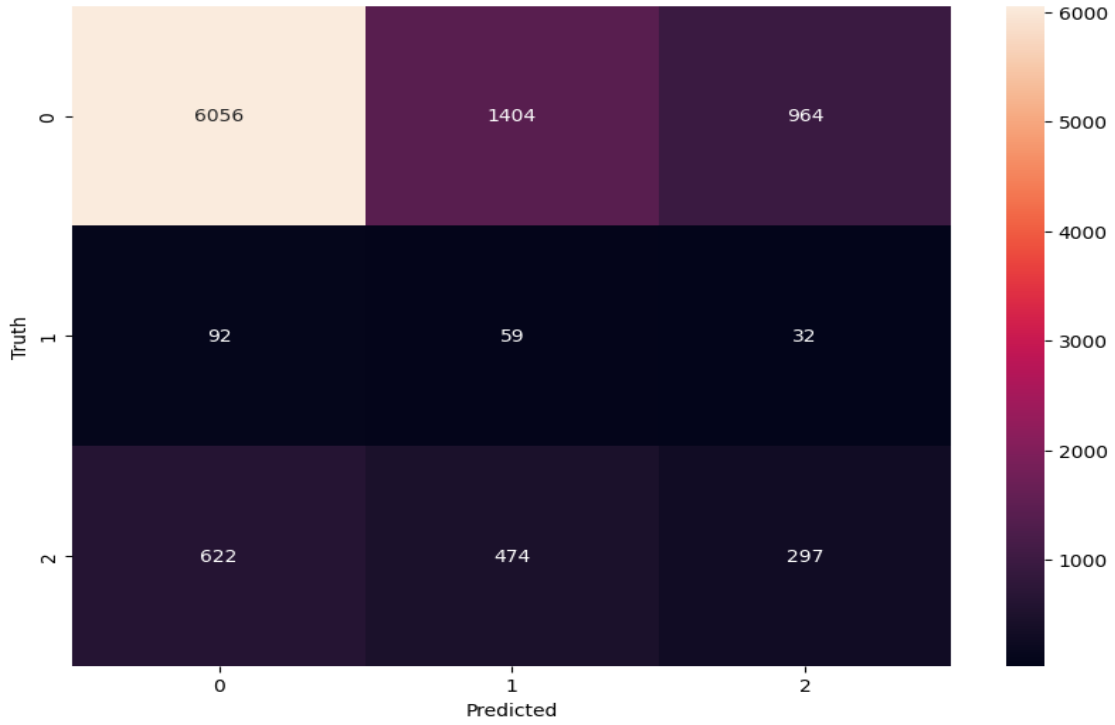
Test:



Train:



Full:



Se poate observa cum pe setul de test,rezulatetele sunt proaste,datorita prezentei dezechilibrului intre clase,majoritatea preziceriilor fiind doar pentru 0.

Pe celalalte seturi de date,se poate observa o imbunatatire, rezultate satisfacatoare pentru clasele 0 si 2.

Clasa 0 :

Train:

	ACC	PREC	F1	RECALL
RF_SK	0.66	0.66	0.78	0.95
RF_MA	0.83	0.72	0.82	0.97
MLP_SK	0**.93**	**0.94**	**0.96**	**0.99**
MLP_MA	0.82	0.88	0.90	0.92

Test:

	ACC	PREC	F1	RECALL
RF_SK	0.64	**0.89**	0.76	0.66
RF_MA	0.63	0.85	0.72	0.63
MLP_SK	**0.74**	0.76	**0.85**	**0.96**
MLP_MA	0.70	0.74	0.82	0.92

Full:

	ACC	PREC	F1	RECALL
RF_SK	0.66	0.67	0.78	0.94
RF_MA	0.63	0.63	0.75	0.92
MLP_SK	**0.86**	**0.88**	**0.92**	**0.97**
MLP_MA	0.80	0.85	0.89	0.92

Clasa 1 :

Train:

	ACC	PREC	F1	RECALL
RF_SK	0.66	0.36	0.05	0.08
RF_MA	0.83	**0.95**	**0.86**	0.**78**
MLP_SK	**0.93**	0.03	0.02	0.05
MLP_MA	0.82	0.02	0.03	0.05

Test:

	ACC	PREC	F1	RECALL
RF_SK	0.64	**0.04**	**0.19**	**0.07**
RF_MA	0.63	0	0	0
MLP_SK	**0.75**	0	0	0
MLP_MA	0.70	0.03	0.04	0.04

Full:

	ACC	PREC	F1	RECALL
RF_SK	0.66	**0.31**	**0.05**	**0.08**
RF_MA	0.58	0	0	0
MLP_SK	**0.86**	0.0	0.00	0.0
MLP_MA	0.80	0.02	0.02	0.03

Clasa 2 :

Train:

	ACC	PREC	F1	RECALL
RF_SK	0.66	0.65	0.38	0.27
RF_MA	0.83	**0.89**	**0.81**	**0.74**
MLP_SK	**0.93**	0.88	0.70	0.58
MLP_MA	0.82	0.30	0.21	0.16

Test:

	ACC	PREC	F1	RECALL
RF_SK	0.64	0.46	**0.53**	0.63
RF_MA	0.63	0.38	0.50	**0.72**
MLP_SK	**0.75**	**0.60**	0.32	0.22
MLP_MA	0.70	0.52	0.23	0.14

Full:

	ACC	PREC	F1	RECALL
RF_SK	0.66	0.64	**0.42**	**0.32**
RF_MA	0.63	**0.73**	0.36	0.24
MLP_SK	**0.86**	0.59	0.40	0.31
MLP_MA	0.80	0.33	0.20	0.14

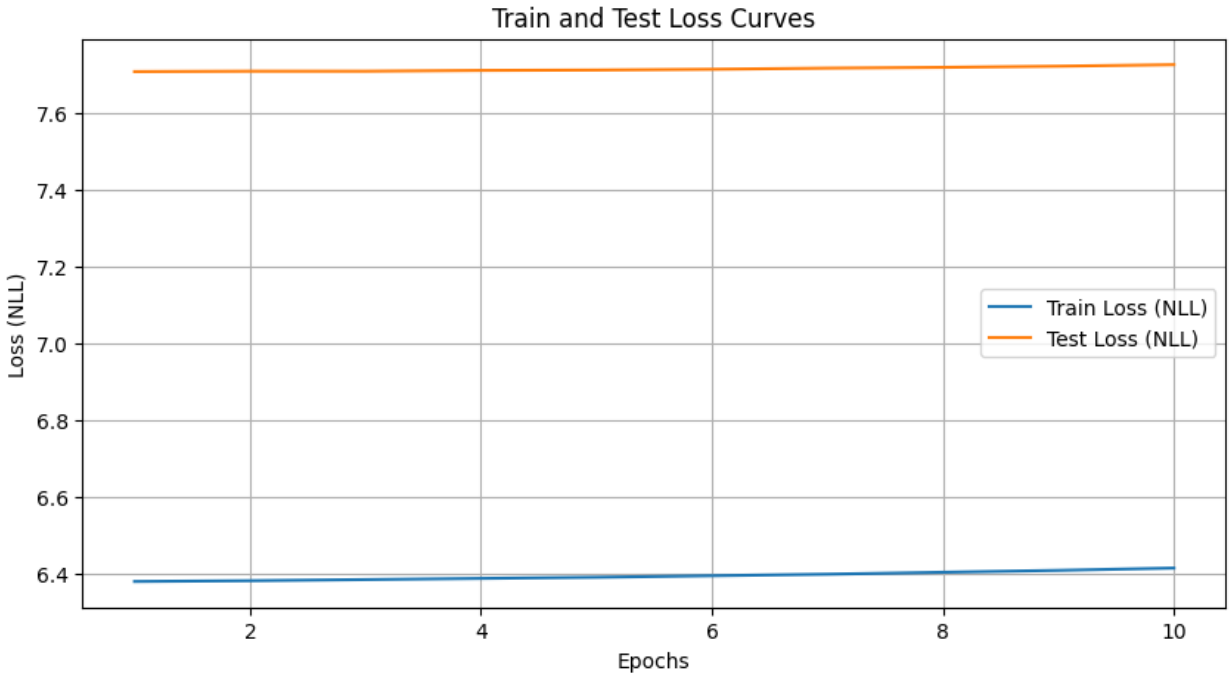
Concluzie:

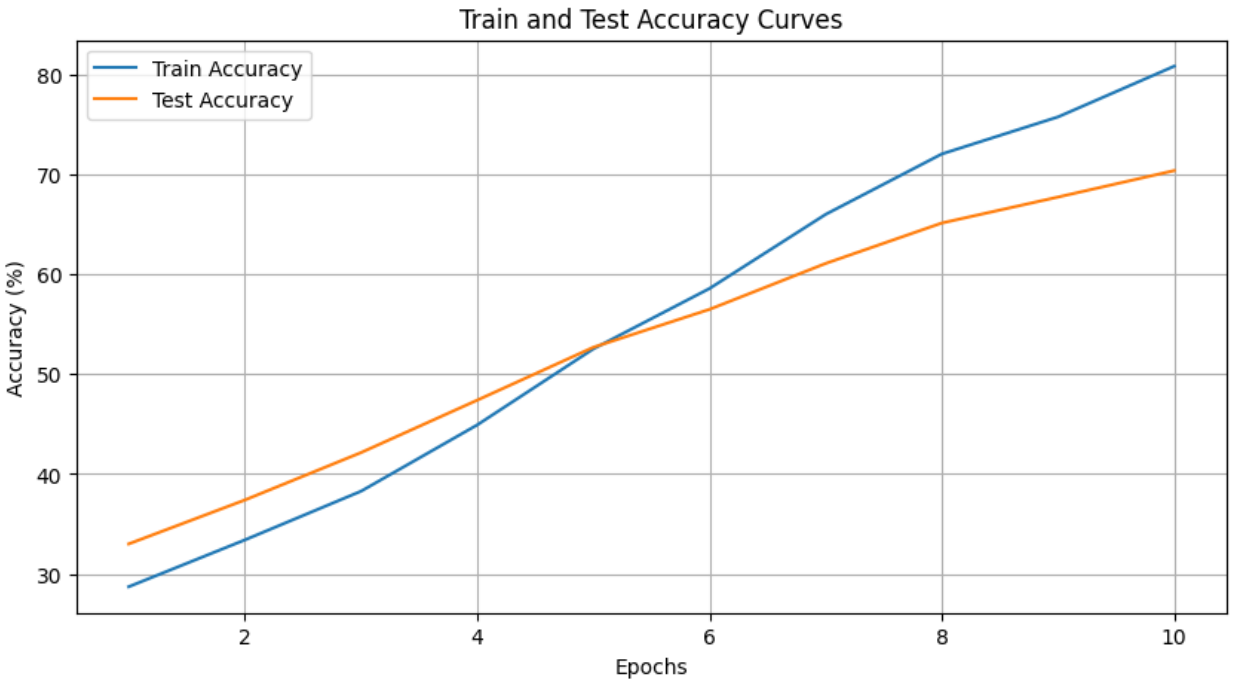
Prezenta unui dataset destul de dezechilibrat arata cum algoritimi se chinuie sa prezica correct,majoritatea fiind prezize pe clasa 0. Algoritmul RandomForest sklearn prezice cel mai correct, urmat de MTP cu implementarea manuala.

Prezenta hypermarametriilor este destul de importanta,fara ea, toate prezicerile ar fi fost doar pe clasa 0.

Algoritmul RandomForest sklearn este cel mai bun, deoarece este singurul care face o balansare a setului de data dezechilibrat,astfel reusind sa dea un f1 score decent si pentru restul claselor.

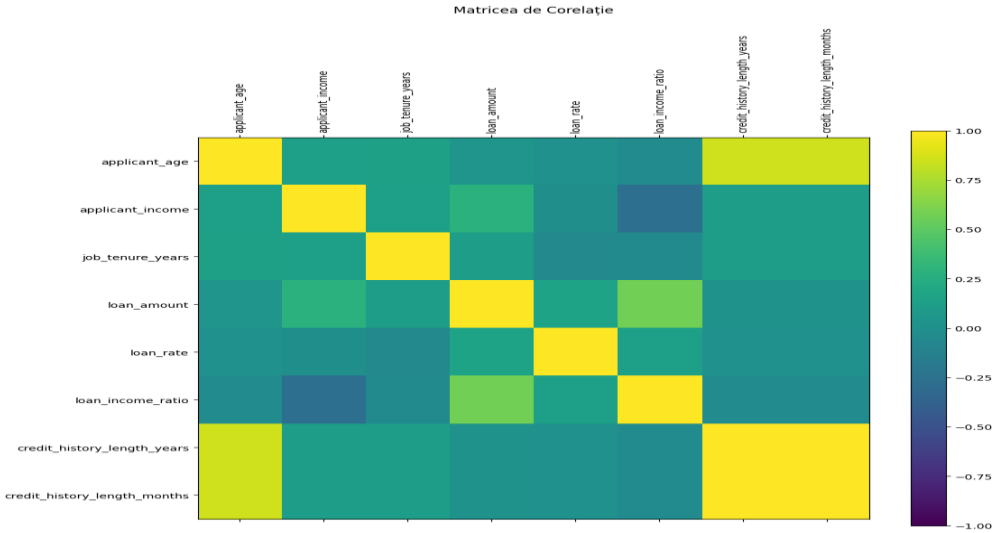
Grafice:

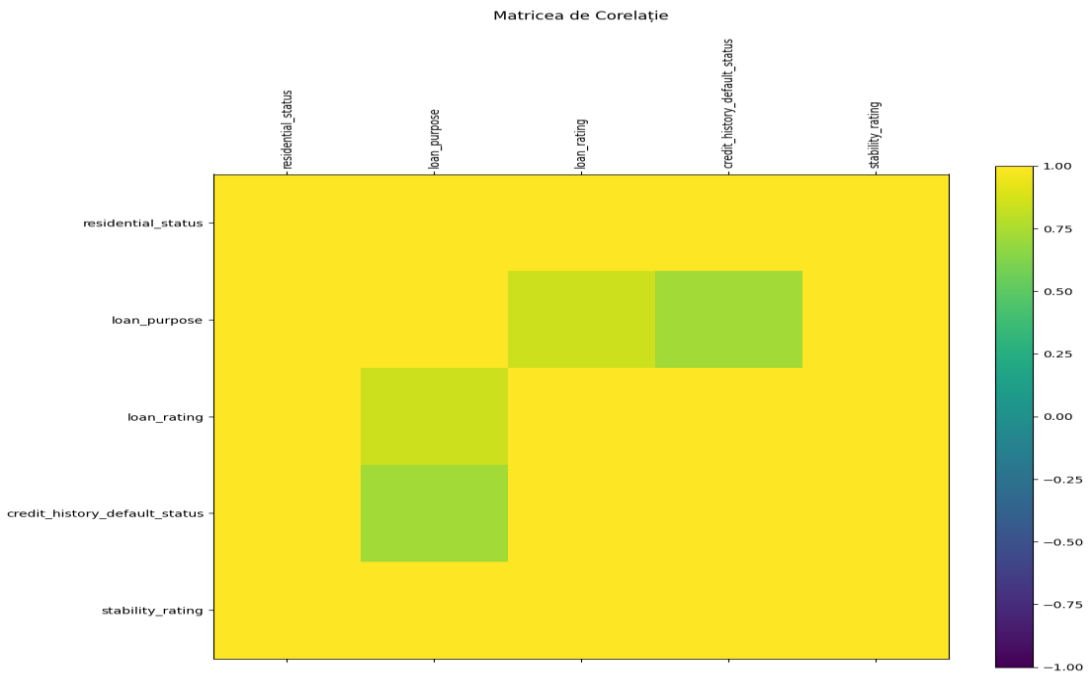




Risc de creditare

Preprocesarea datelor:





- Pentru datele care lipsesc, am folosit SimpleImputer cu strategia “mean” pentru atributele numerice si strategia “most frequent” pentru atributele discrete.
- Pentru datele extreme am folosit recomandarea din anexa cu setarea quantilelor intre 0.25 si 0.75
- Pentru a standardiza toate atributele numerice, am folosit StandardScaler.

Evaluarea algortimilor:

- **RandomForest**

Sklearn:

Am folosit RandomizedSearchCV pentru a face tunning la hyperparametrii. Astfel, am obtinut o referinta despre cum ar trebui sa arate parametrii, facand, la final, doar niste mici ajustari.

```
      RandomForestClassifier(n_estimators=200, max_depth=110,min_samples_leaf =
10,max_leaf_nodes=9, criterion='gini', class_weight='balanced')
```

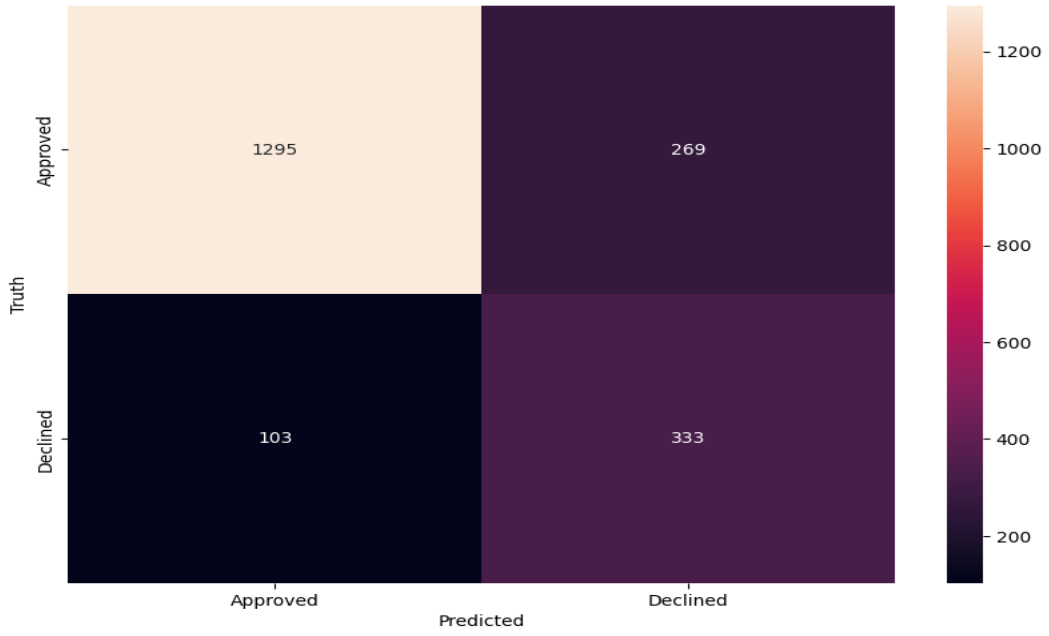
200 de arbori de decizie, fiecare având o adâncime maximă de 110, un minim de 10 de eșantioane necesare pentru a fi într-un nod frunză, și un maxim de 9 noduri frunză pe arbore. Arborii utilizează criteriul de impuritate Gini pentru divizarea nodurilor, iar claselor sunt echilibrate pentru a aborda dezechilibrul de clasă în setul de date.

Cel mai important parametru setat a fost class_weight='balanced', deoarece,fara el precizia,recall,F1 -score ul si matricea de confuzie ar fi dat rezultate destul de slabe, intrucat clasele din dataset nu sunt balansate,majoritar fiind pentru clasa 0.

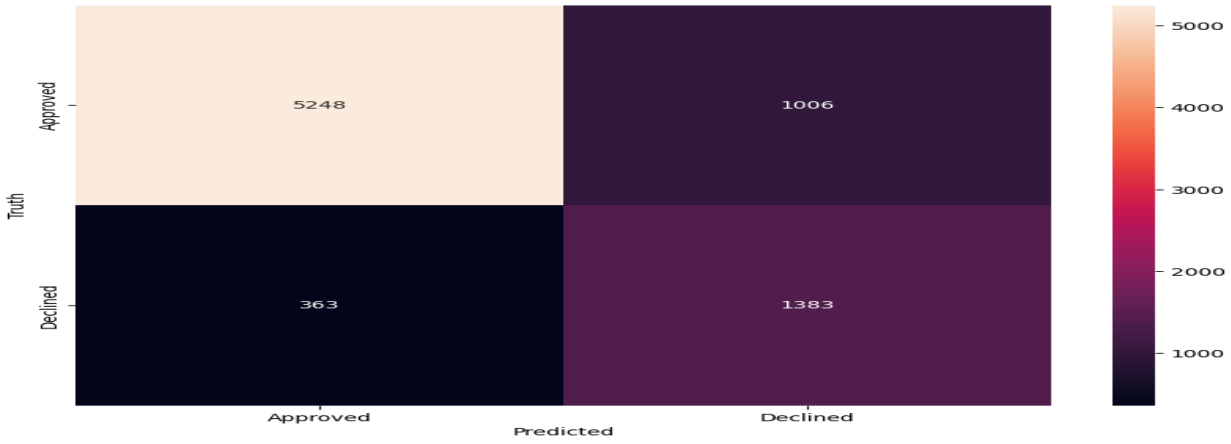
Restul, au fost aleși pentru a optimiza performanța modelului RandomForestClassifier

Astfel, s a obtinut:

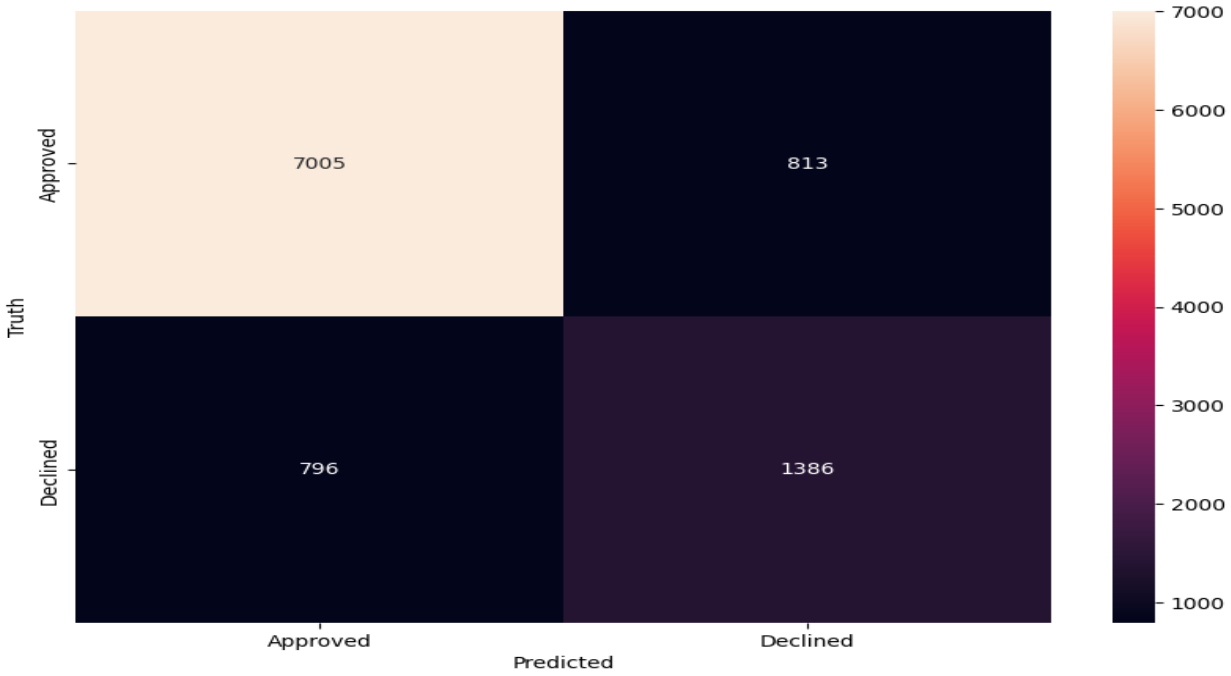
Test:



Train:



Full:



Se poate observa cum modelul obtine rezultate bune pentru clasa 0 si rezultate satisfacatoare pentru clasa 1.Un tunning mai detaliat al hperparametriilor nu ar fi adus imbunatatiri majore, datorita dezechilibrului claselor din dataset.

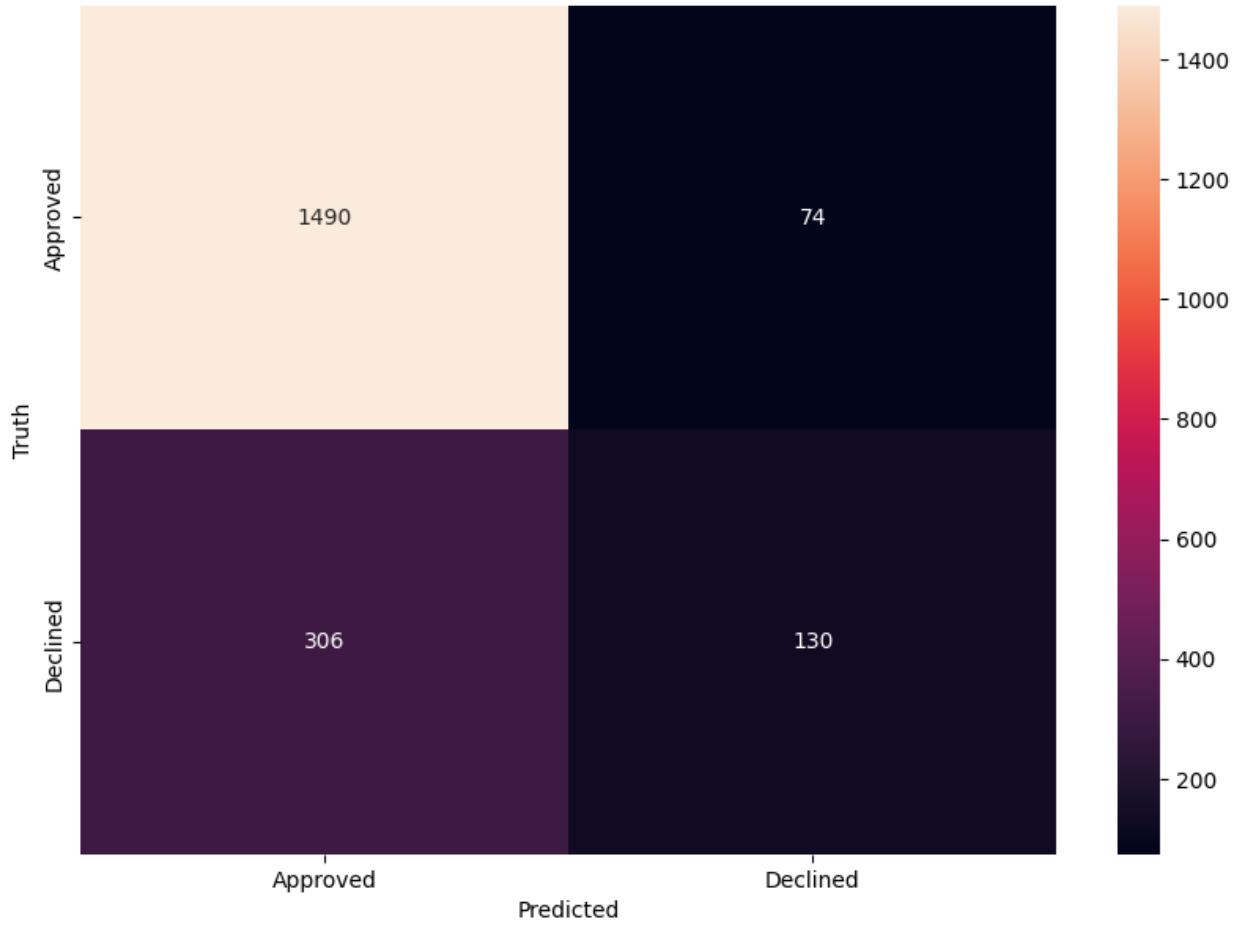
Implemenatare manuala:

M am folosit de functia “Smote” pentru a face resampling pe datele din train care erau dezechilibrate pentru a putea face o comparatie echilibrata intra algoritmul implementat manual si cel cu sklearn.Deoarece timpul de rulare este destul de mare, nu am avut ocazia sa fac foarte mult tunning pe hyperparametrii,ajungand la solutia finala :

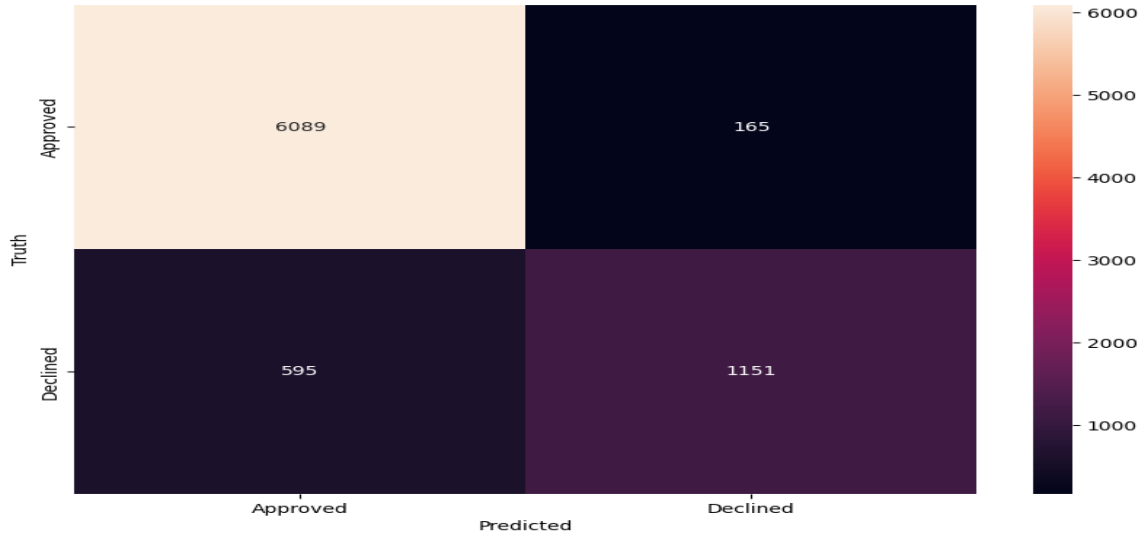
```
    (n_estimators=10,max_depth=5, min_samples_per_node=31)
```

10 de arbori de decizie, fiecare având o adâncime maximă de 5, un minim de 31 de eșantioane necesare pentru a fi într-un nod frunză. Arborii utilizează criteriul Gini pentru divizarea nodurilor.

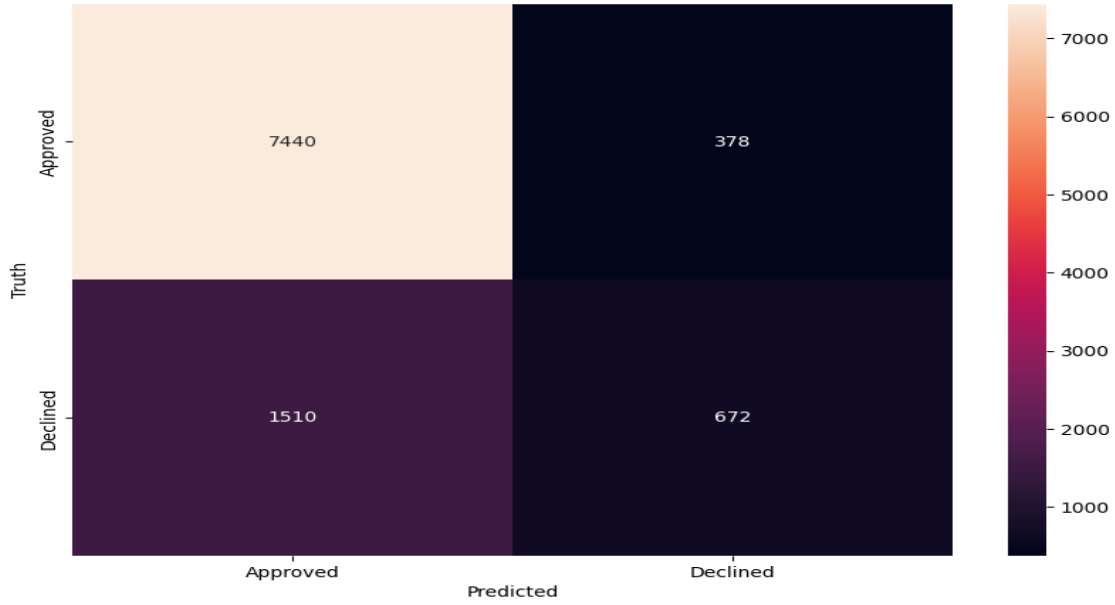
Test:



Train:



Full:



Se poate observa o imbunatatire fata de setul de date precedent, datorita incercarii echilibrului de date. Totusi, nu este suficient, algortimul nu a prezis foarte bine pentru clasa Declined

- **MLP:**

Sklearn:

La fel ca si la RandomForest,m am folosit de RandomizedSearchCV pentru a seta hyperparametrii,obtinand:

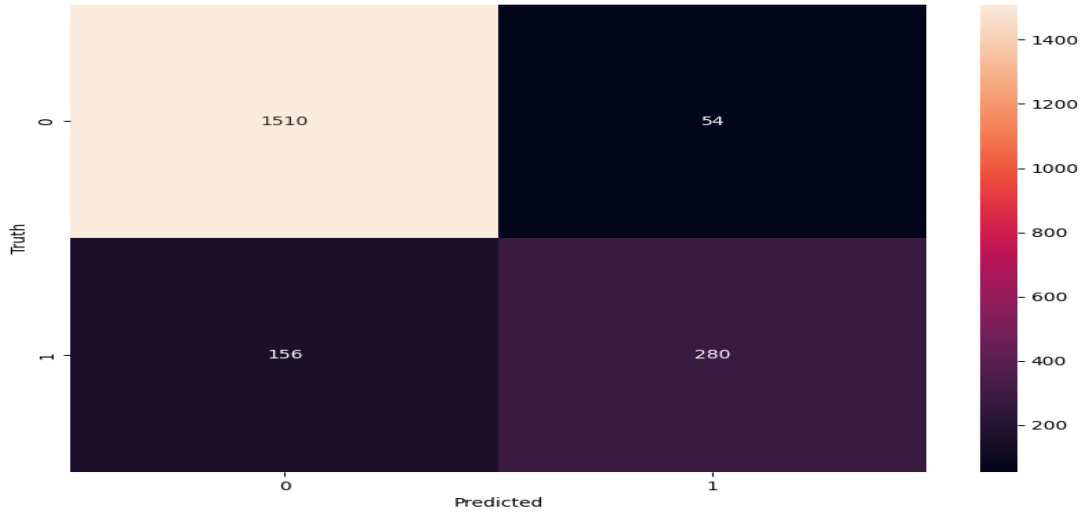
```
MLPClassifier(activation='relu', alpha=0.01, batch_size=128,
hidden_layer_sizes=(256,), learning_rate='adaptive', max_iter=500, solver='sgd')
```

Folosesc funcția de activare 'relu', un termen de regularizare L2 alpha de 0.01, un batch size de 128, un singur strat din rețea de dimensiune 256, un algoritm de învățare adaptivă, 500 de epoci de antrenare și un optimizator 'sgd' (Stochastic Gradient Descent).

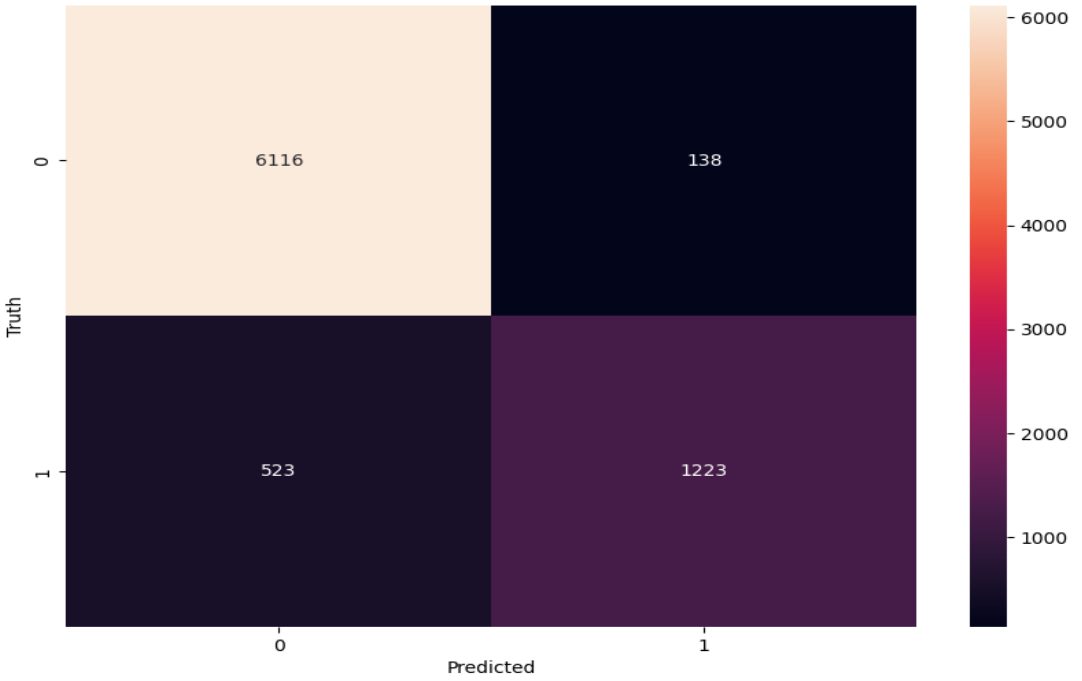
Am ales sa nu folosesc early_stopping, deoarece primeam un rezultat extrem de slab. Restul, au fost aleși pentru a optimiza performanța.

Astfel, s a obtinut:

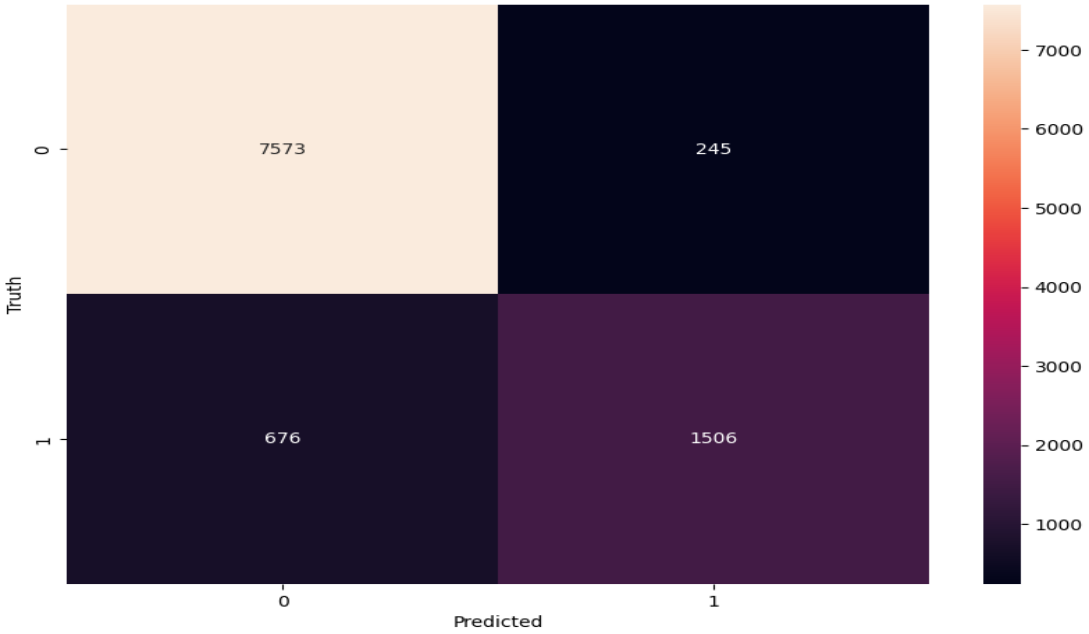
Test:



Train:



Full:



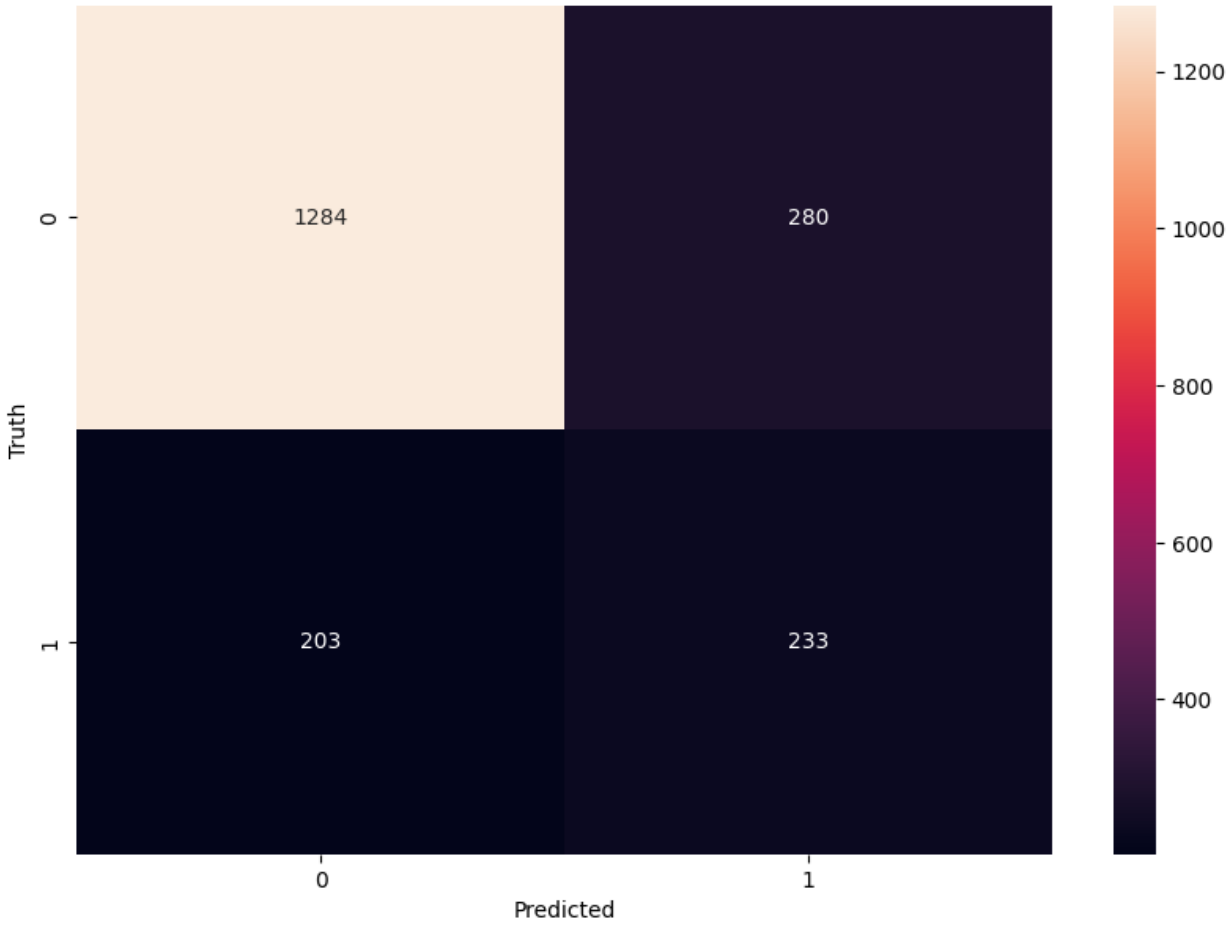
Se poate observa rezultate extrem de bune in ciuda dezechilibrului dintre clase.

Implementare manuala:

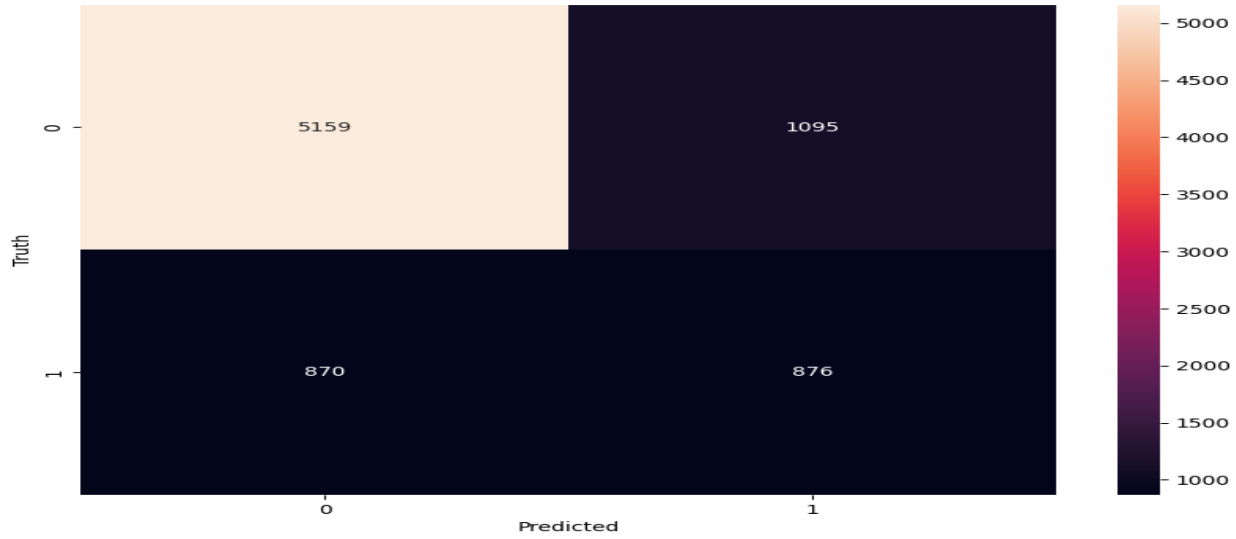
Pentru hyperparametrii, am pornit de la unii asemenatori cu modelul sklearn,unde i am mai retusat pentru a obtine o precizie mai buna. Am facut compromisul,de a alege un batch size mai mare care,imi da o acuratete mai mica pe teste si train,dar primesc un f1 score mai mare. Astfel:

Folosesc funcția de activare 'relu', un termen de regularizare L2 alpha de 0.01, un batch size de 512, doua straturi din rețea de dimensiune 128, 10 de epoci de antrenare și un optimizator 'sgd' (Stochastic Gradient Descent).

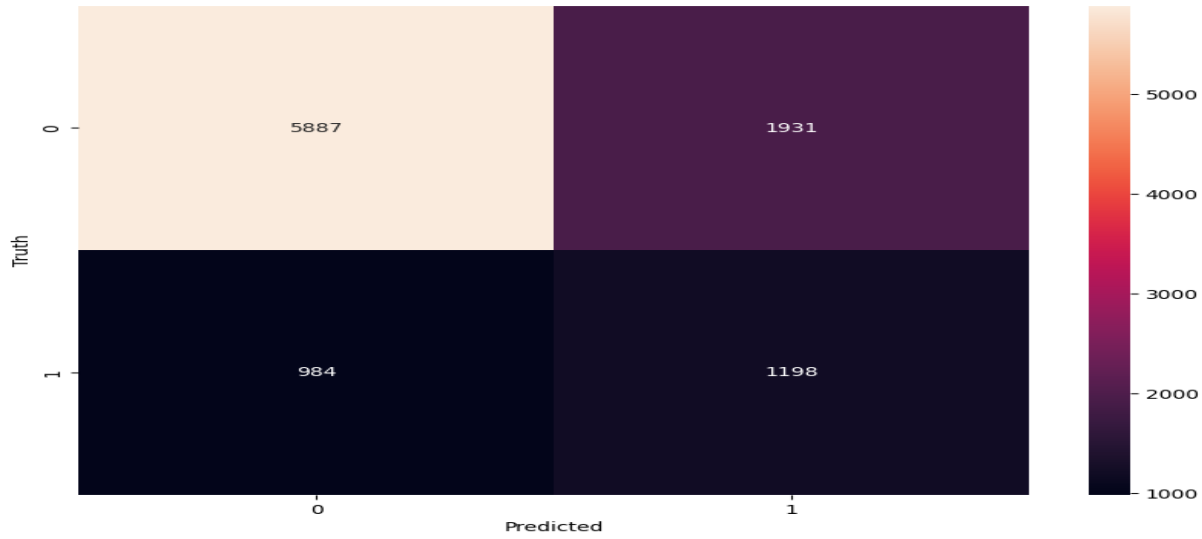
Test:



Train:



Full:



Algortimul a prezis ok clasele,afisand rezultate putin mai slabe fata de modelul MTP implementat cu sklearn.

Clasa Approved :

Train:

	ACC	PREC	F1	RECALL
RF_SK	0.83	0.84	0.89	**0.94**
RF_MA	0.91	0.97	0.94	0.91
MLP_SK	**0.92**	**0.98**	**0.95**	0.92
MLP_MA	0.71	0.86	0.80	0.76

Test:

	ACC	PREC	F1	RECALL
RF_SK	0.82	**0.93**	0.88	0.83
RF_MA	0.81	0.83	0.89	0.95
MLP_SK	**0.90**	0.91	**0.93**	**0.97**
MLP_MA	0.71	0.85	0.80	0.76

Full:

	ACC	PREC	F1	RECALL
RF_SK	0.83	0.84	0.88	**0.94**
RF_MA	0.81	0.95	0.89	0.83
MLP_SK	**0.91**	**0.97**	**0.94**	0.92
MLP_MA	0.71	0.86	0.80	0.75

Clasa Declined :

Train:

	ACC	PREC	F1	RECALL
RF_SK	0.83	**0.81**	0.68	0.58
RF_MA	0.91	0.66	0.75	0.87
MLP_SK	**0.92**	0.70	**0.79**	**0.90**
MLP_MA	0.71	0.39	0.45	0.55

Test:

	ACC	PREC	F1	RECALL
RF_SK	0.82	0.56	0.65	**0.77**
RF_MA	0.81	0.64	0.41	0.30
MLP_SK	**0.90**	**0.84**	**0.73**	0.64
MLP_MA	0.71	0.37	0.43	0.52

Full:

	ACC	PREC	F1	RECALL
RF_SK	0.83	**0.79**	0.67	0.58
RF_MA	0.81	0.31	0.42	0.64
MLP_SK	**0.91**	0.69	**0.77**	**0.86**
MLP_MA	0.71	0.38	0.45	0.55

Concluzie:

Putem observa diferențe semnificative în performanța algoritmilor pentru clasele "Approved" și "Declined" atât în seturile de antrenament, cât și în cele de testare.

Pentru clasa "Approved":

- Algoritmii Random Forest (RF) și (MLP) obțin rezultate similare în ceea ce privește acuratețea și alte metrici de evaluare atât în seturile de antrenament, cât și în cele de testare.

Pentru clasa "Declined":

- Modelul (MLP) pare să obțină cea mai bună performanță î n această clasă, cu acurateți și precizii mai mari decât cele ale Random Forest (RF).

Desi setul tot unul dezechilibrat este, acesta pare ceva mai ok decat cel precent si ,se poate obersva cum, MLP este mai performant. Deoarece tunning ul intre hyperparametrii nu difera foaret mult, putem trage concluzia ca caracteristicile și distribuția datelor din setul de date pot influența performanța algoritmilor. Este posibil ca anumite caracteristici să fie mai bine capturate de un algoritm decât de altul, ceea ce ar putea explica diferențele în performanța observată.

Grafice: