

TEMA 2 IA – ML Aplicat

Vlad Andra – 331CC

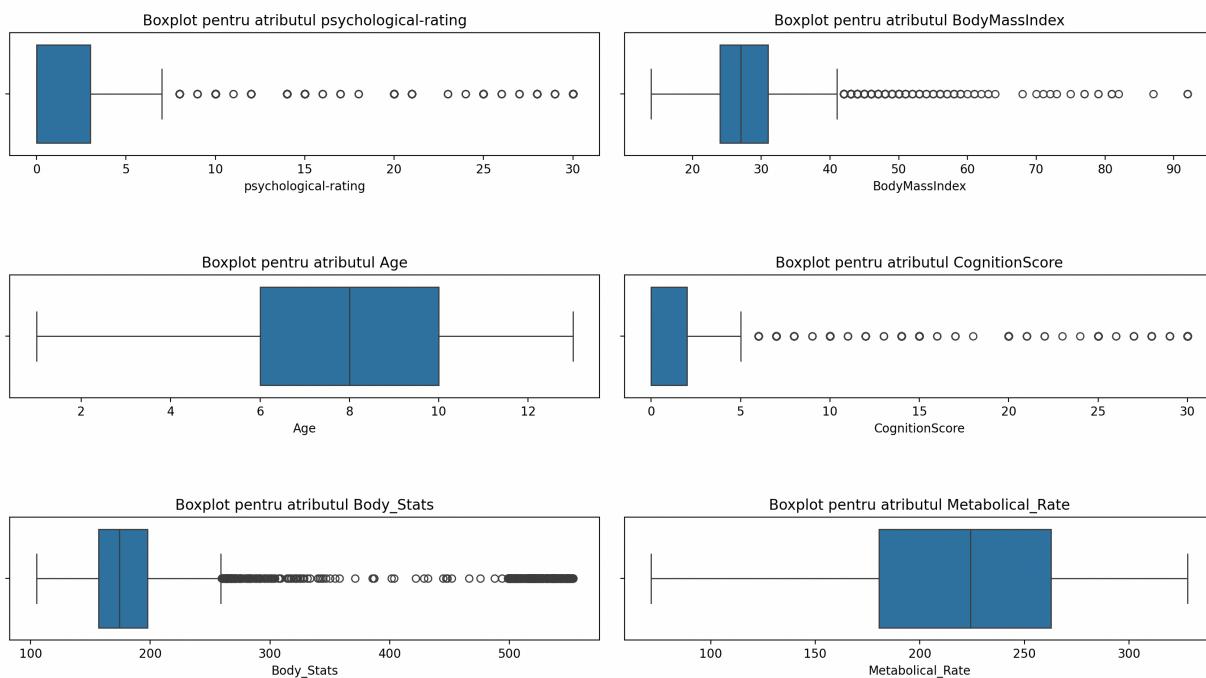
1. Introducere

În acest document voi prezenta un **raport de comparare** a celor două seturi de date în contextul aplicării modelului de învățare automată Random Forest pentru **prezicerea diabetului / riscului de aprobat a unui credit**. Singura parte de ML realizată de mine în această temă a fost evaluarea modelului Random Forest folosind biblioteca scikit-learn, cu **RandomForestClassifier**, aşadar focusul comparațiilor dintre rezultatele obținute va avea în vedere strict această parte a temei. Pentru **rularea temei**, se va folosi comanda **python3 tema_2.py <argument: diabet sau credit_risk>**.

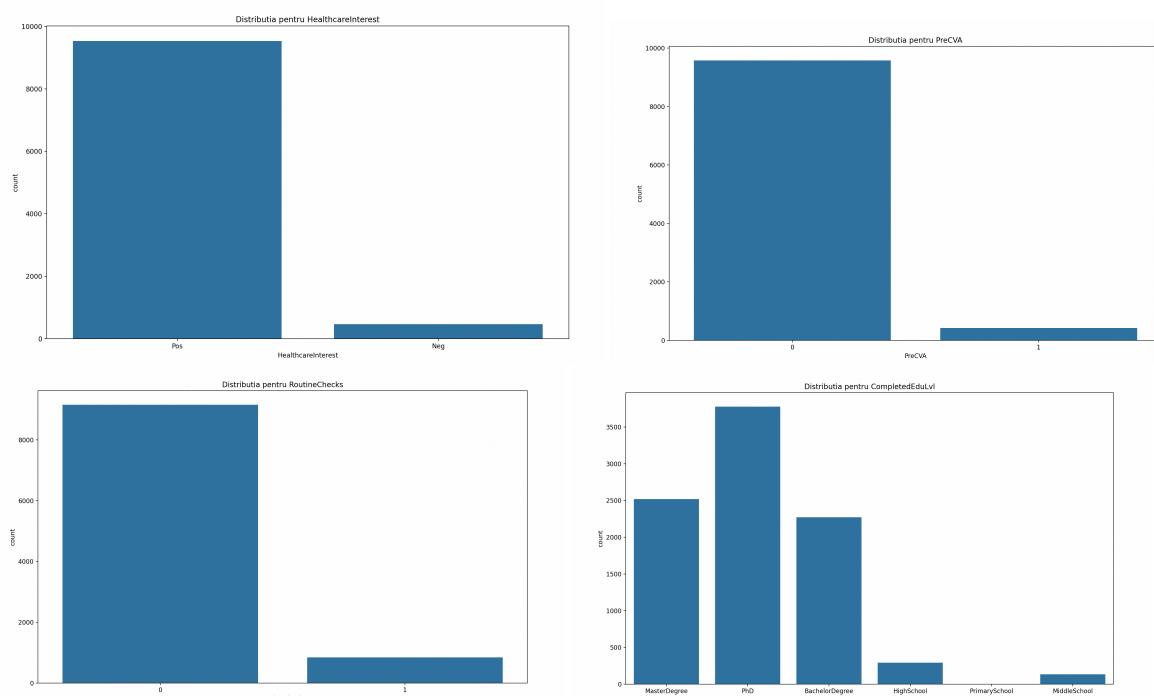
2. Interpretarea extragerii datelor

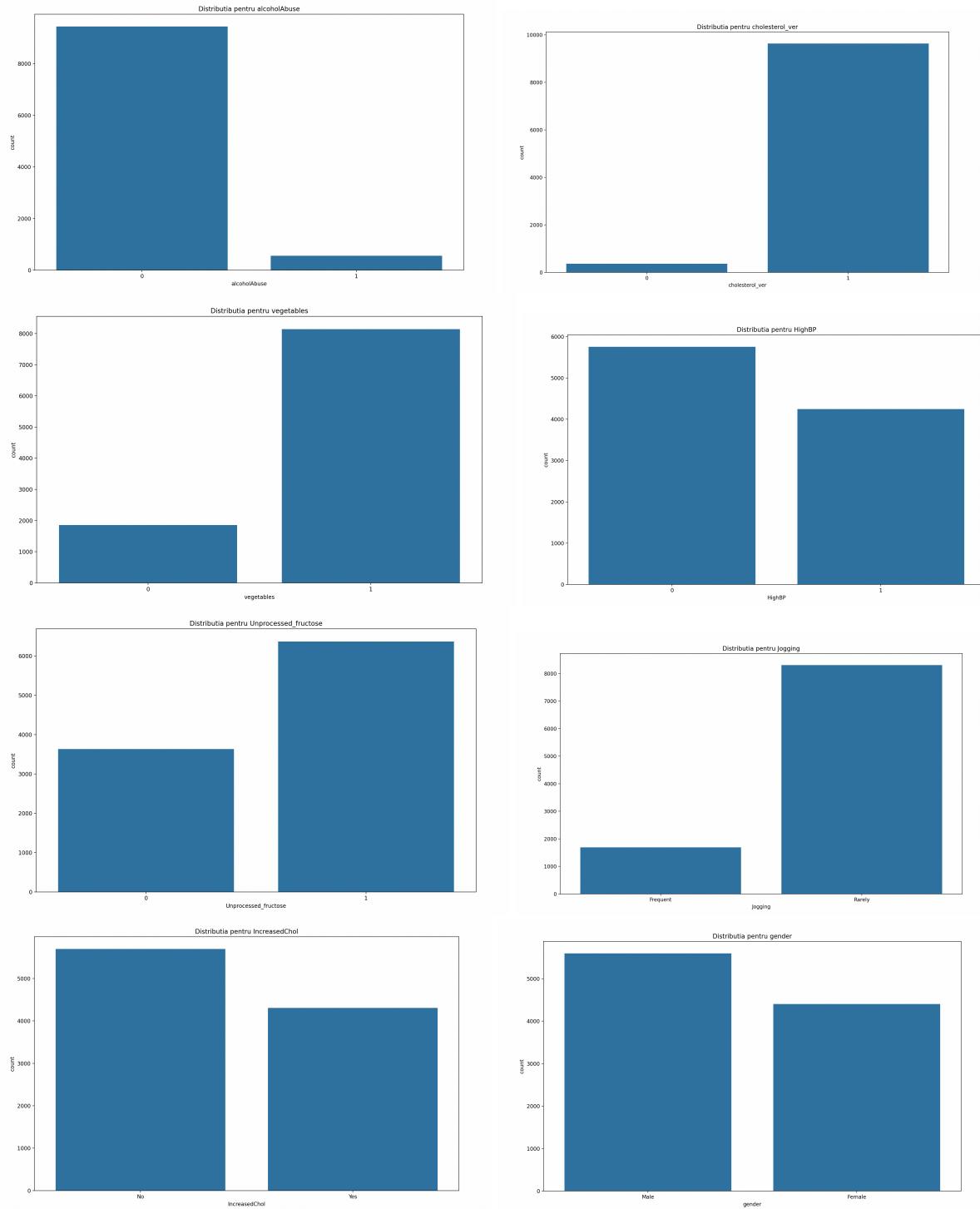
Pentru o analiză inițială a datelor, le vom observa mai întâi pe cele din [Diabet_full.csv](#) prin boxplot-uri pentru atributele numerice și histograme pentru atributele categorice.

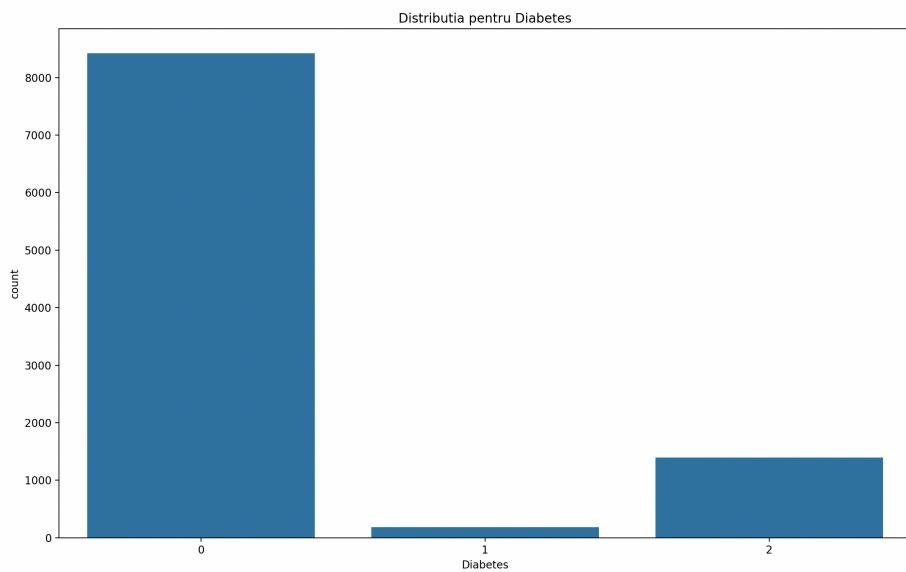
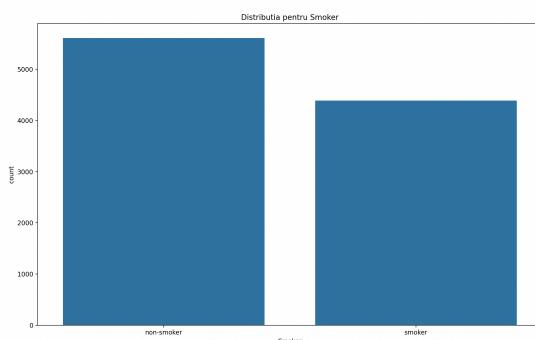
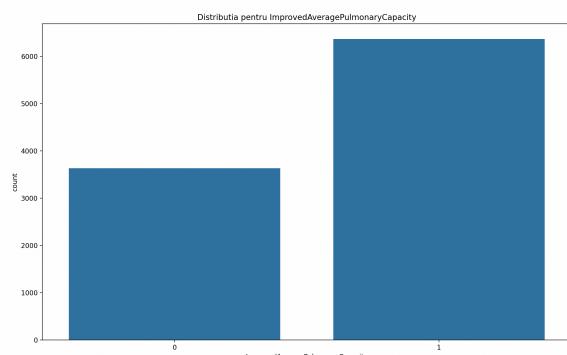
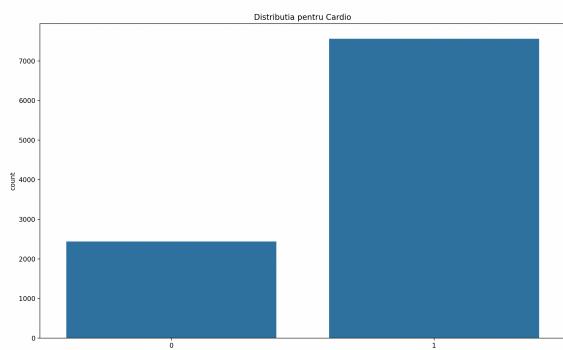
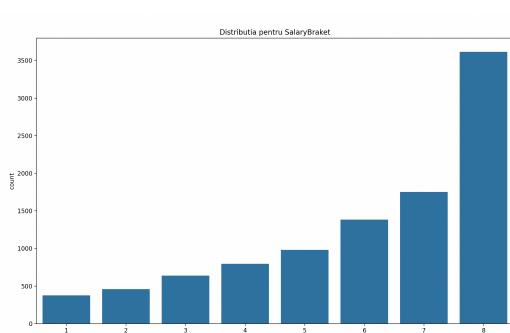
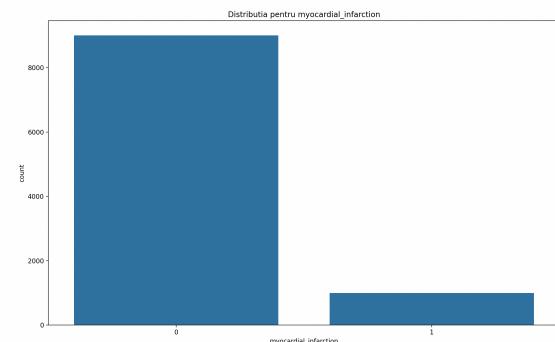
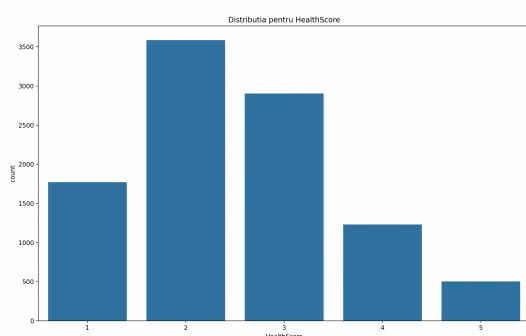
----- ANALIZA ATRIBUTE NUMERICE CONTINUE -----								
	count	mean	std	min	25%	50%	75%	max
psychological-rating	10000.0	4.365100	8.891103	0.000000	0.000000	0.000000	3.000000	30.000000
BodyMassIndex	10000.0	28.246500	6.462563	14.000000	24.000000	27.000000	31.000000	92.000000
Age	10000.0	8.057500	3.036300	1.000000	6.000000	8.000000	10.000000	13.000000
CognitionScore	10000.0	3.125300	7.308607	0.000000	0.000000	0.000000	2.000000	30.000000
Body_Stats	10000.0	194.960784	82.438106	105.063984	156.720671	174.042100	197.742249	553.000000
Metabolical_Rate	9000.0	221.592499	60.480951	71.602207	180.542314	224.218817	262.688901	327.936098



	count	unique
HealthcareInterest	10000	2
PreCVA	10000	2
RoutineChecks	10000	2
CompletedEduLvl	9000	6
alcoholAbuse	10000	2
cholesterol_ver	10000	2
vegetables	10000	2
HighBP	10000	2
Unprocessed_fructose	10000	2
Jogging	10000	2
IncreasedChol	10000	2
gender	10000	2
HealthScore	10000	5
myocardial_infarction	10000	2
SalaryBraket	10000	8
Cardio	10000	2
ImprovedAveragePulmonaryCapacity	10000	2
Smoker	10000	2
Diabetes	10000	3



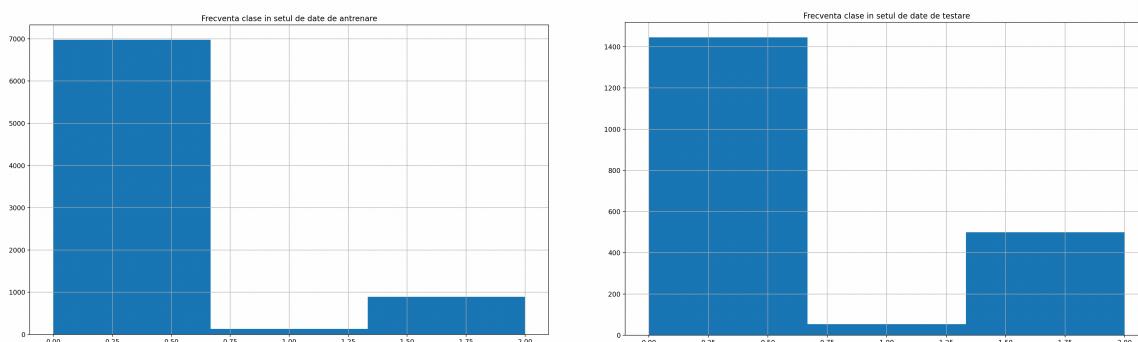




Încă din start putem observa că **datele noastre nu sunt bune**. Pentru **atributele numerice**, avem pentru attribute precum **psychological_rating** o medie de 4.365 și o deviație de 8.891, ceea ce ne sugerează că avem o distribuție neuniformă, valori extreme (multe de 0) ce trebuie ulterior imputate. Se vede că sunt multe attribute numerice cu valori extreme și că doar **Metabolical-Rate și Age** sunt mai echilibrate, însă unul de acesta va trebui eliminat ulterior pentru a reduce attributele redundante, deoarece se poate vedea din grafic că ele sunt foarte asemănătoare.

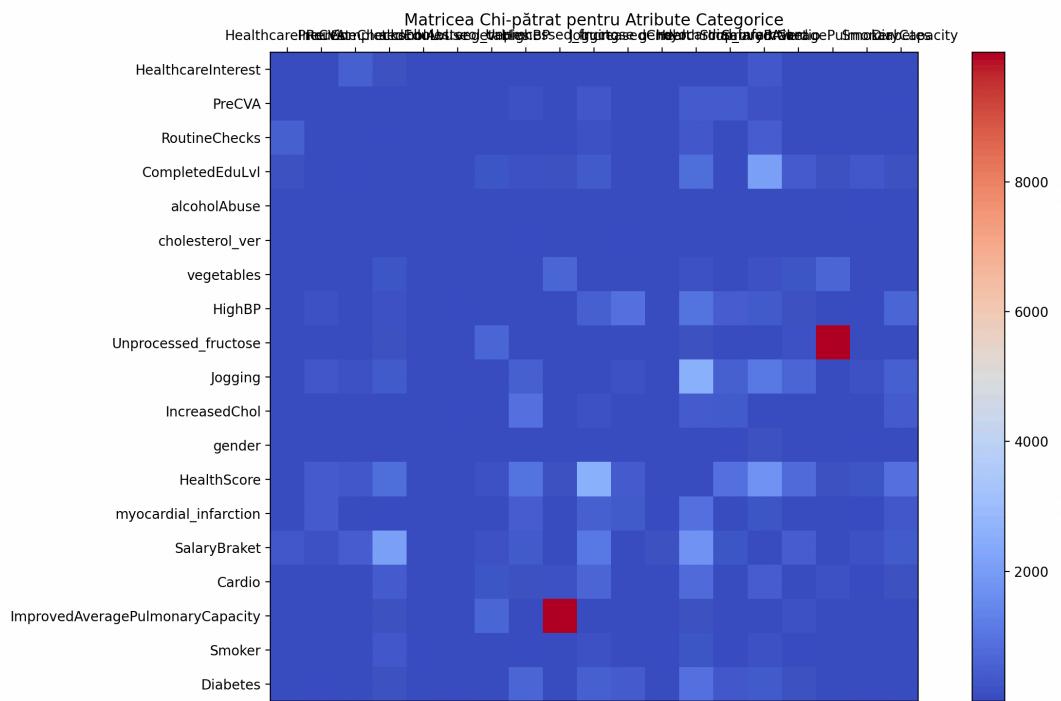
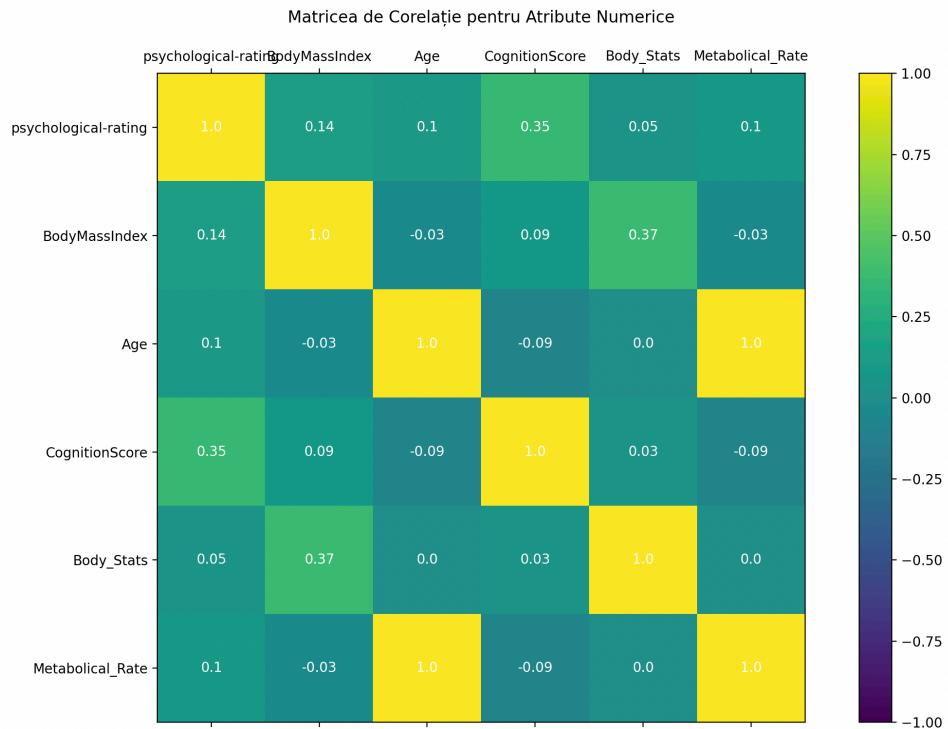
Pentru **atributele categorice**, se pot observa din start că sunt foarte multe corelate între ele, cum ar fi **Jogging** și **Cardio**. Acest lucru va crea probleme la eliminarea atributelor redundante în procesarea datelor.

În urma frecvenței claselor (etichetelor), în cazul nostru ‘Diabetes’, observăm că sunt 3 clase și că este foarte dezechilibrat, având foarte multe date despre cei fără diabet, aproape deloc despre prediabet și câteva despre diabet.



Statistică similară față de cele observate de noi putem deduce și prin plotarea matricelor de corelare. Pentru attributele numerice, `.corr()` folosește implicit criteriu Pearson. Pentru cele categorice am folosit testul statistic Chi Pătrat, însă rezultatele nu sunt unele care ne pot ajuta prea mult. Chi Pătrat testează independența între valorile categorice, așadar valorile foarte mici obținute marchează o **dependență puternică**. Se poate observa din plot că **majoritatea valorilor sunt extrem de mici**, așadar **multe câmpuri depind unele de altele și nu s-au extras cele mai relevante măsuri pentru datele despre diabet**. Codul pentru plotarea matricelor l-am preluat de aici:

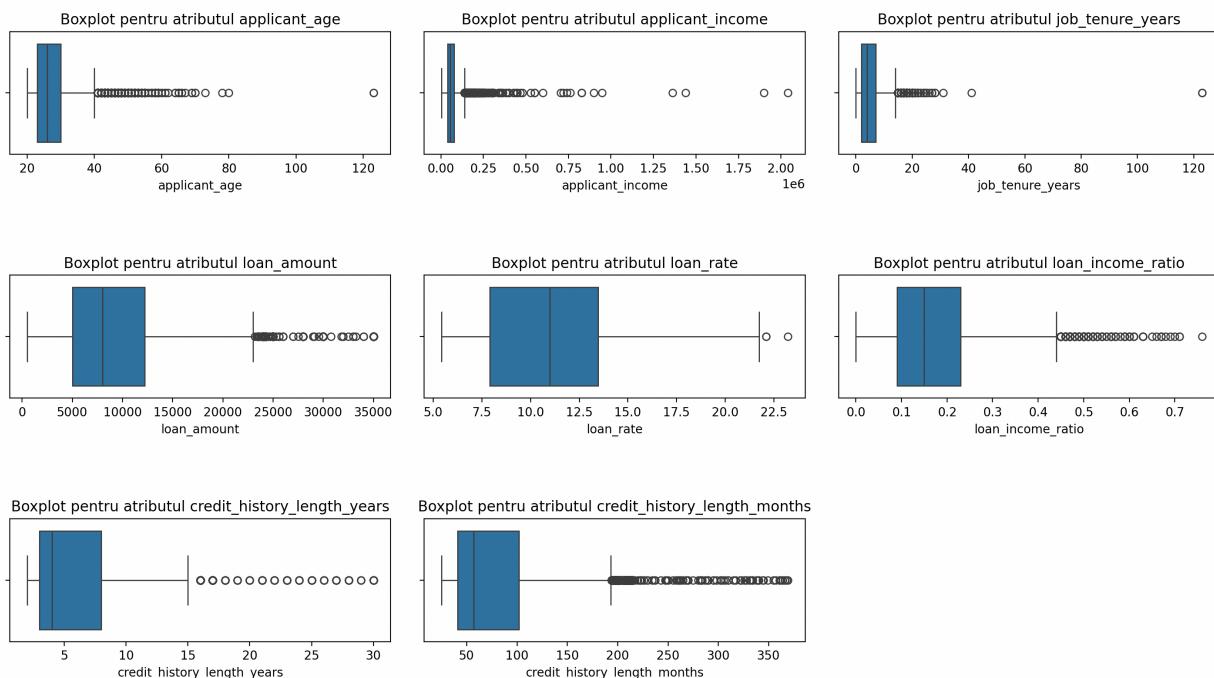
<https://www.andrewgurung.com/2018/12/23/matplotlib-correlation-matrix-plot/>



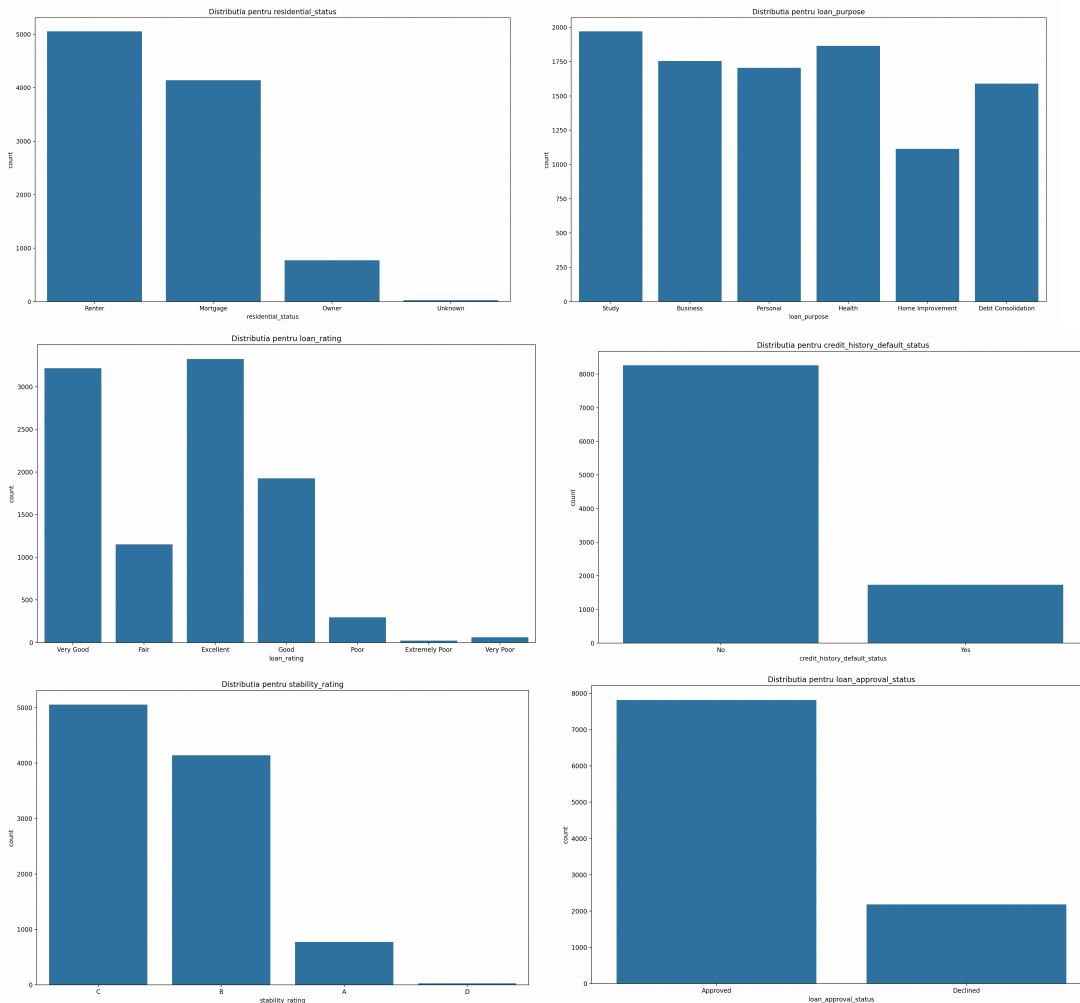
Următoarea analiză are în vedere [credit_risk_full.csv](#):

----- ANALIZA ATRIBUTE NUMERICE CONTINUE -----								
	count	mean	std	min	25%	50%	75%	max
applicant_age	10000.0	27.745100	6.360155	20.00	23.00	26.00	30.00	123.00
applicant_income	10000.0	65734.211300	56944.387081	4200.00	38595.00	55000.00	78997.00	2039784.00
job_tenure_years	9736.0	4.785744	4.353122	0.00	2.00	4.00	7.00	123.00
loan_amount	10000.0	9568.037500	6350.431581	500.00	5000.00	8000.00	12200.00	35000.00
loan_rate	9060.0	11.007179	3.266393	5.42	7.90	10.99	13.47	23.22
loan_income_ratio	10000.0	0.170130	0.106814	0.00	0.09	0.15	0.23	0.76
credit_history_length_years	10000.0	5.811100	4.050217	2.00	3.00	4.00	8.00	30.00
credit_history_length_months	10000.0	75.760700	48.677362	25.00	41.00	57.00	102.00	369.00
----- ANALIZA ATRIBUTE DISCRETE SAU ORDINALE -----								
	count	unique						
residential_status	10000	4						
loan_purpose	10000	6						
loan_rating	10000	7						
credit_history_default_status	10000	2						
stability_rating	10000	4						
loan_approval_status	10000	2						

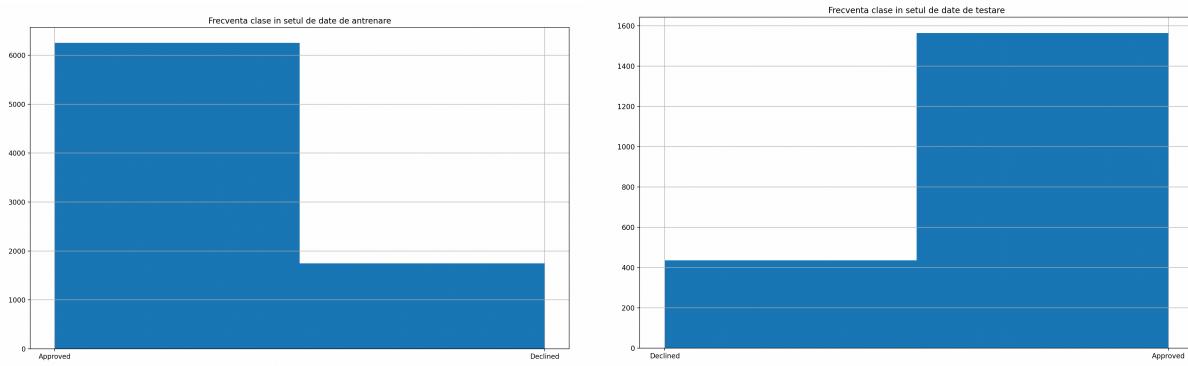
Putem observa de aici deja că multe attribute numerice au multe valori extreme, precum **applicant_age**, **applicant_income**, **job_tenure_years**, **loan_amount**, **loan_income_ratio**. Deja avem un indicator că pentru aceste date ar fi mai util să folosim un **scaler robust** pentru a le standardiza.



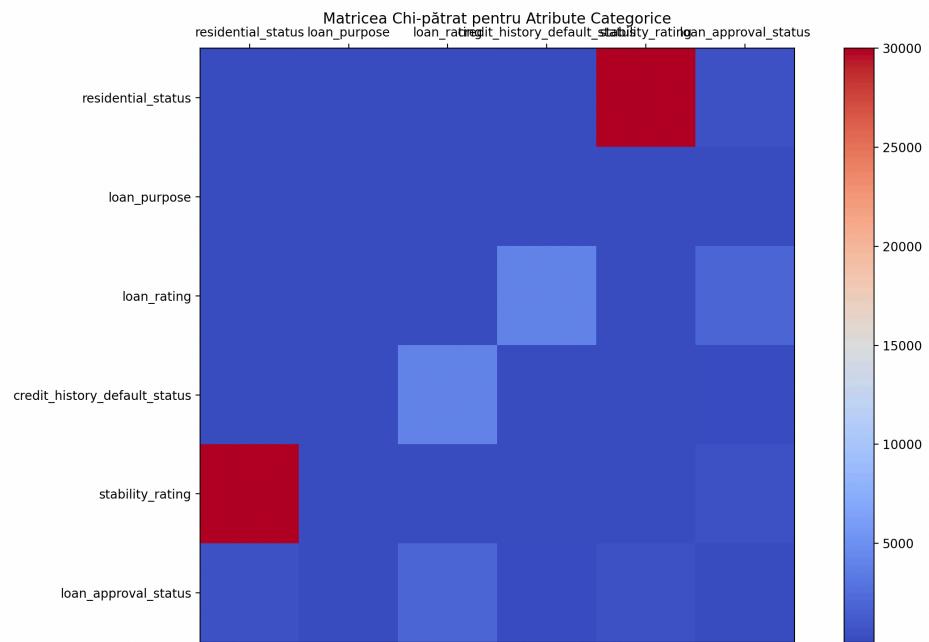
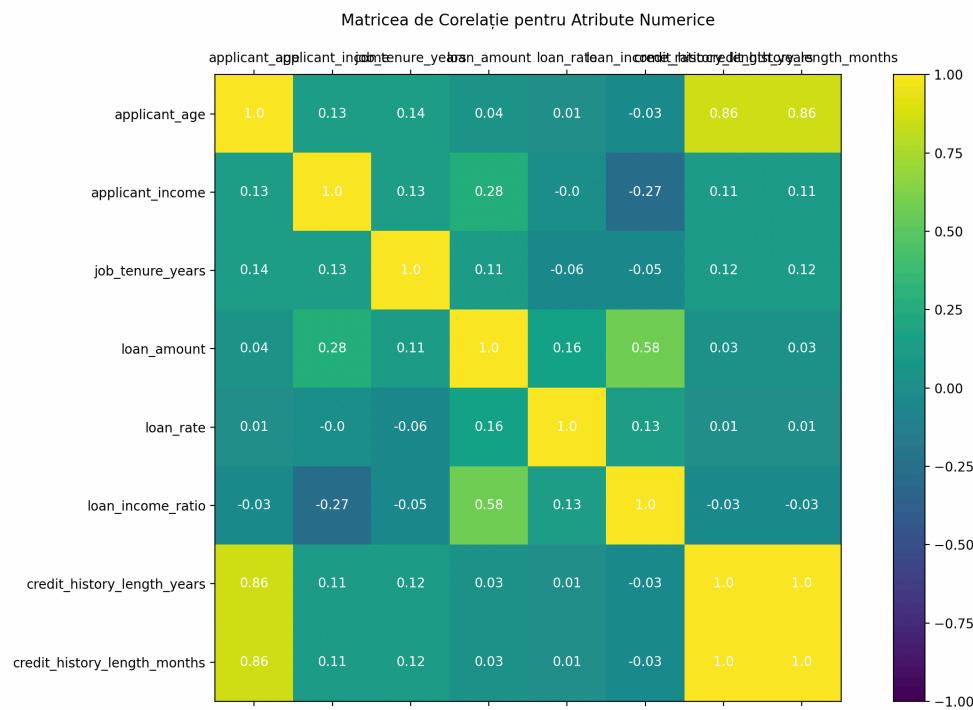
Se observă și corelarea unor attribute numerice precum **credit_history_length_years** și **applicant_age** sau **credit_history_length_years** și **credit_history_length_months**. Acestea se vor vedea mai clar în matricea de corelare.



Deși și aici există atribute categorice puternic corelate, acestea sunt mai puține, mai utile pentru problema noastră și aşadar putem estima deja că vor duce la rezultate mai bune decât cele din Diabet, după eliminarea atributelor redundante.



Aici avem două clase, ‘Approved’ or ‘Declined’. Deși încă dezechilibrate, tot sunt mai bine proporționate decât la dataset-ul cu Diabet.



Pentru matricele de corelație, pentru testul Chi Pătrat, se pot observa rezultate similare cu cele de la dataset-ul cu Diabet. De aceea, în procesarea datelor, vom folosi aceleași tehnici de eliminare a atributelor redundante.

3. Interpretarea procesării datelor

Deși am implementat această funcție conform cerinței, aleg să **NU elimin valorile extreme**, deoarece acestea sunt numeroase, iar eu fac doar algoritmul RandomForest. RandomForest este cunoscut pentru robustețea față de valorile extreme, deoarece fiecare arbore de decizie e construit pe un subset diferit al datelor, iar valoarea finală e o medie a predicțiilor individuale a arborilor. Acest lucru tinde să atenueze efectul valorilor extreme, aşadar prefer să nu le elimin, fiind foarte prezente atât în ‘Diabet’, cât și în ‘credit_risk’.

Am ales să **imputez valorile lipsă** cu SimpleImputer cu strategia ‘most_frequent’ pentru atributele categorice și IterativeImputer pentru atributele numerice.

SimpleImputer **alege** în cazul nostru **cea mai frecventă valoare pentru a imputa valoarea lipsă**, menținând astfel distribuția originală a datelor. IterativeImputer **estimează valorile lipsă, bazându-se pe relația dintre toate caracteristicile din dataset**. Așadar, în teorie, acesta ar trebui să fie mai precis decât un SimpleImputer care folosește media spre exemplu, însă în realitate obțin cam aceleași rezultate.

Pentru datasetul ‘Diabet’, **standardizez datele** cu StandardScaler, care **scalează datele astfel încât fiecare caracteristică să aibă o medie 0 și o deviație standard 1**. Pentru datasetul ‘credit_risk’ folosesc însă ceva mai util pentru datele cu multe variații extreme, RobustScaler, care funcționează mai bine pe acest tip de date. MinMaxScaler nu este o opțiune, deoarece este mult mai sensibil la valori extreme.

Pentru **eliminarea atributelor redundante**, logica prin care am obținut rezultatele cele mai bune pentru ambele dataset-uri este următoarea: pentru atributele numerice, eliminăm **al doilea atribut numeric** dintre perechile de atrbute ce obțin **valori de peste 0.75** în matricea de corelație realizată cu **Pearson**; pentru atributele categorice, eliminăm **dintre primele 5 perechi cu cel mai mic scor în urma Chi Pătrat**, doar **al doilea atribut categoric**. Am ales un număr echilibrat de atrbute ce ar trebui eliminate, considerând că în urma algoritmilor am aflat că majoritatea sunt redundante, mai ales la ‘Diabet’. Alegerea unui numar limitat de eliminare a atributelor **elimina overfitting-ul**, iar **eliminarea multor atrbute nu ar ajuta modelul** deoarece am pierde informații relevante.

4. Interpretarea rezultatelor

Ambel dataset-uri folosesc funcția **train_si_eval_model_forest**, pentru care am folosit în mare parte codul din laboratorul ML-RandomForest. Ce face funcția este să antreneze și să evaluateze modelul de clasificare Random Forest pe un set de date. Funcția are 3 etape: preprocesarea datelor, antrenarea modelului și evaluarea performanței.

Separăm atrbutul target (‘Diabetes’, respectiv ‘loan_approval_status’) de restul atrbutelor pentru seturile de antrenament și test. X_train și X_test conțin doar atrbutele,

iar `Y_train` și `Y_test` conțin atributul target. Folosim `get_dummies` pentru a converti variabilele categorice în variabile numerice folosind one-hot encoding, după care ne asigurăm că `X_train` și `X_test` au aceleași coloane după one-hot encoding, completând cu 0 unde e necesar cu ajutorul funcției `reindex`. După aceea, creăm un model `RandomForestClassifier` cu 50 de arbori de decizie și un seed fix și antrenăm modelul pe datele de antrenament. Facem predicțiile pe setul de test cu funcția `model_predict(X_test)` și evaluăm performanța modelului.

Pentru [datele despre diabet](#), avem următoarele rezultate:

Number of examples per class in train set:				
Diabetes				
0	6978			
2	893			
1	129			
Name: count, dtype: int64				
Accuracy: 0.736				
Classification Report:				
	precision	recall	f1-score	support
0	0.74	0.99	0.84	1446
1	0.00	0.00	0.00	54
2	0.71	0.08	0.15	500
accuracy			0.74	2000
macro avg	0.48	0.36	0.33	2000
weighted avg	0.71	0.74	0.65	2000

Din căte putem observa după **weighted avg f1-score**, rezultatele nu sunt prea bune. Motivul principal este dezechilibrul semnificativ dintre clase. Prezicerea pentru clasa 1 este inexistentă, clasa reprezentantă pentru prediabet a fost complet ignorată. Pentru clasa 2 (diabet), avem o prezicere mare, însă un recall mic. Asta se traduce astfel: **dacă modelul prezice că un pacient are diabet, are dreptate de cele mai multe ori, însă modelul ratează cele mai multe dintre cazurile de diabet și nu le detectează**. Așadar, putem analiza scorul overall pentru **f1-score**, unde vedem că modelul obține rezultate foarte bune pentru 0 = fără diabet, foarte mici pentru 2 = diabet și inexistent pentru 1 = prediabet. Modelul funcționează, așadar, foarte prost. **Ar trebui o ponderare a claselor în vreun mod pentru echilibrarea claselor și folosirea unor atrbute mai bune pentru a obține niște rezultate utile**.

Pentru [datele despre riscul unui credit](#), avem următoarele rezultate:

```
Dataset: Credit Risk
Train set size: 8000
Test set size: 2000
Number of classes: 2
Number of features: 9

Number of examples per class in train set:
loan_approval_status
Approved    6254
Declined    1746
Name: count, dtype: int64

Accuracy: 0.919
Classification Report:
precision    recall    f1-score   support
Approved      0.92      0.98      0.95      1564
Declined      0.91      0.70      0.79       436

accuracy           0.92      0.92      0.92      2000
macro avg        0.91      0.84      0.87      2000
weighted avg     0.92      0.92      0.92      2000
```

Aici putem să observăm niște **rezultate bune**. Deși clasele sunt încă dezechilibrate, avem totuși date mai bune, cu atribute care ne ajută să determinăm target-ul. Însă, se poate vedea și la valoarea ‘recall’ pentru ‘Decline’ că modelul ratează totuși unele valori. Cu toate acestea, rezultatele sunt satisfăcătoare și putem observa că am realizat o eliminare bună a atributelor redundante, iar RobustScaler a standardizat eficient valorile extreme pentru a crește acuratețea modelului.

5. Comparație între cele două seturi de date

Pe parcursul documentului, am realizat tot felul de comparații între cele două seturi de date. Aici voi centraliza diferențele principale:

- ‘Diabet’ are 3 clase (0, 1, 2), ‘credit_risk’ are 2 clase (‘Approved’ și ‘Declined’)
- Datele din ‘Diabet’ conțin valori lipsă, datele din ‘credit_risk’ conțin și ele câteva valori lipsă și foarte multe valori extreme, însă valorile extreme nu ne afectează foarte mult la RandomForest după standardizare.
- Clasele de la ‘Diabet’ sunt mai dezechilibrate decât cele de la ‘credit_risk’. ‘credit_risk’ are un dezechilibru moderat, dar cu suficiente exemple în clasa minoritară ‘Declined’ astfel încât modelul să poată învăța.
- Avem o relație mai bună între atrbute și target (‘loan_approval_status’) pentru ‘credit_risk’.
- Ambele clase au mult prea multe atrbute categorice redundante în urma testului Chi Pătrat.
- ‘Diabet’ conține mult mai multe atrbute decât ‘credit_risk’, fiind o problemă mai complexă, însă multe din acestea pot fi eliminate.
- Pentru ‘Diabet’, acuratețea obținută de Random Forest este una îngălătoare din cauza dezechilibrului dintre clase, în timp ce pentru ‘credit_risk’ denotă într-adevăr o performanță bună a modelului.
- Random Forest se descurcă mult mai bine cu prezicerea datelor din ‘credit_risk’ decât cu cele din ‘Diabet’.