Ciobanu Mihaela Lavinia
Master Ingineria Datelor, An II

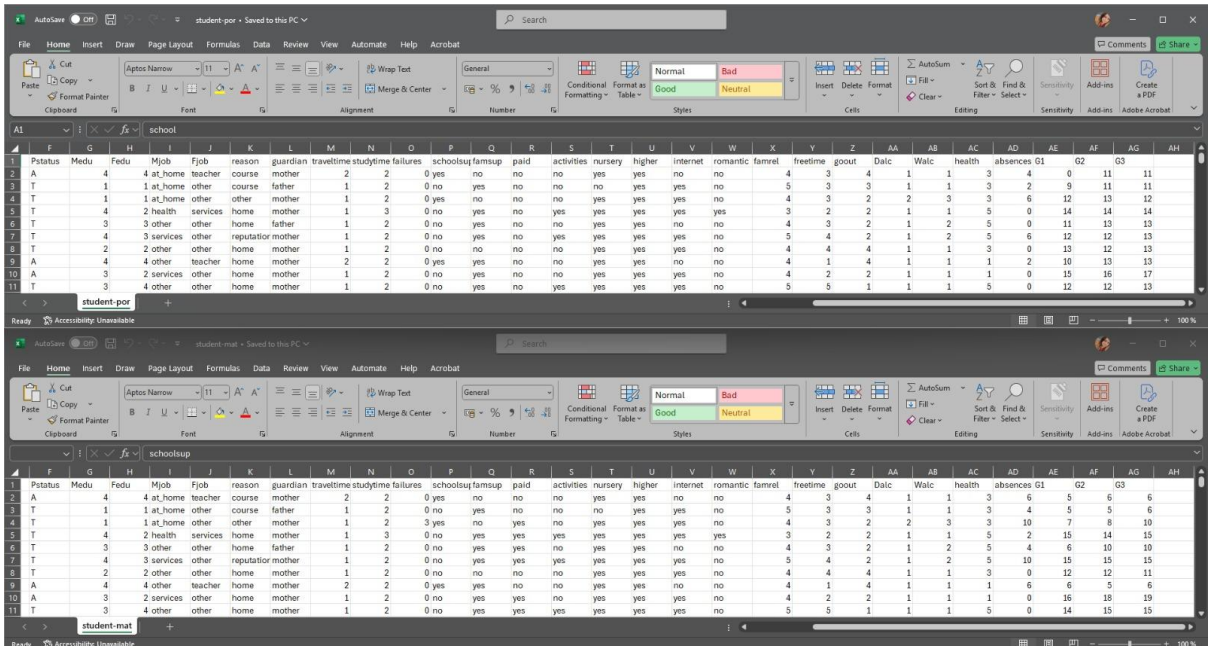
Raport Proiect „Student Performance”

Acest proiect are ca scop analiza setului de date Student Performance, folosind tehnici de preprocesare a datelor, clasificare, clustering si vizualizare. Prin aceste etape, se urmareste intelegerea factorilor care influenteaza performanta scolara si evaluarea comportamentului elevilor din perspective diferite.

1. Preprocesarea Datelor

Pentru aceasta tema am ales setul de date Student_Performance, bazat pe performanta elevilor, asupra caruia am aplicat mai multe operatii pentru a-l aduce intr-o forma potrivita pentru analiza.

Dupa descarcarea locala, am importat fisierul student-mat.csv, care contine informatii despre elevi, precum scoala, varsta, absentele si notele. Dupa incarcarea datelor, am verificat structura tabelului si tipurile de valori pentru a ma asigura ca informatiile sunt citite corect.

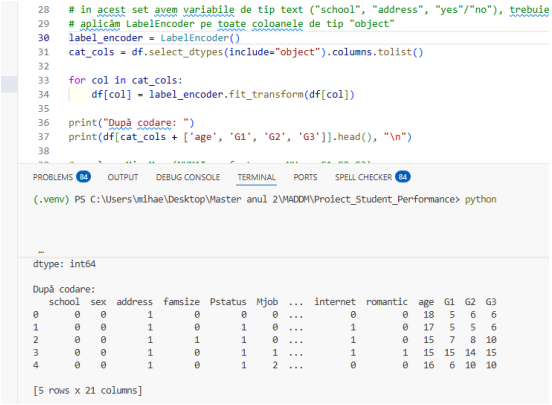


Am verificat daca exista valori lipsa, insa setul de date este complet. In cazul in care ar fi existat, valorile numerice ar fi putut fi completate cu media, iar valorile text cu moda, insa aceasta decizie ar fi fost luata doar daca distributia datelor ar fi permis acest lucru, deoarece aceste metode nu sunt potrivite in toate situatiile.

```
...
Verificarea valorilor lipsă:
school        0
sex           0
age           0
address       0
famsize       0
Pstatus       0
Medu          0
Fedu          0
Mjob          0
Fjob          0
reason        0
guardian      0
traveltime    0
studytime     0
```

Ciobanu Mihaela Lavinia
Master Ingineria Datelor, An II

Setul contine coloane cu valori text, cum ar fi “school”, “address” sau raspunsurile “yes” si “no”. Pentru a le folosi in analiza, le-am transformat in valori numerice folosind **LabelEncoder**, astfel incat fiecare categorie sa aiba un cod propriu.



```
# in acest set avem variabile de tip text ("school", "address", "yes"/"no"), trebuie
# aplicăm LabelEncoder pe toate coloanele de tip "object"
label_encoder = LabelEncoder()
cat_cols = df.select_dtypes(include="object").columns.tolist()

for col in cat_cols:
    df[col] = label_encoder.fit_transform(df[col])

print("După codare: ")
print(df[cat_cols + ['age', 'G1', 'G2', 'G3']].head(), "\n")
```

```
PROBLEMS 84  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  SPELL CHECKER 84
(.venv) PS C:\Users\mihae\Desktop\Master anul 2\MADDM\Proiect_Student_Performance> python
…
dtype: int64

După codare:
   school  sex  address  famsize  Pstatus  Mjob  ...  internet  romantic  age  G1  G2  G3
0       0    0        1        0        0     0  ...         0         0   18   5   6   6
1       0    0        1        0        1     0  ...         1         0   17   5   5   6
2       0    0        1        1        1     0  ...         1         0   15   7   8  10
3       0    0        1        0        1     1  ...         1         1   15  15  14  15
4       0    0        1        0        1     2  ...         0         0   16   6  10  10

[5 rows x 21 columns]
```

Pentru coloanele text cu mai mult de doua valori (de exemplu Mjob, reason, guardian), am folosit **One-Hot Encoding**, pentru a evita introducerea unei ordini false intre valori.

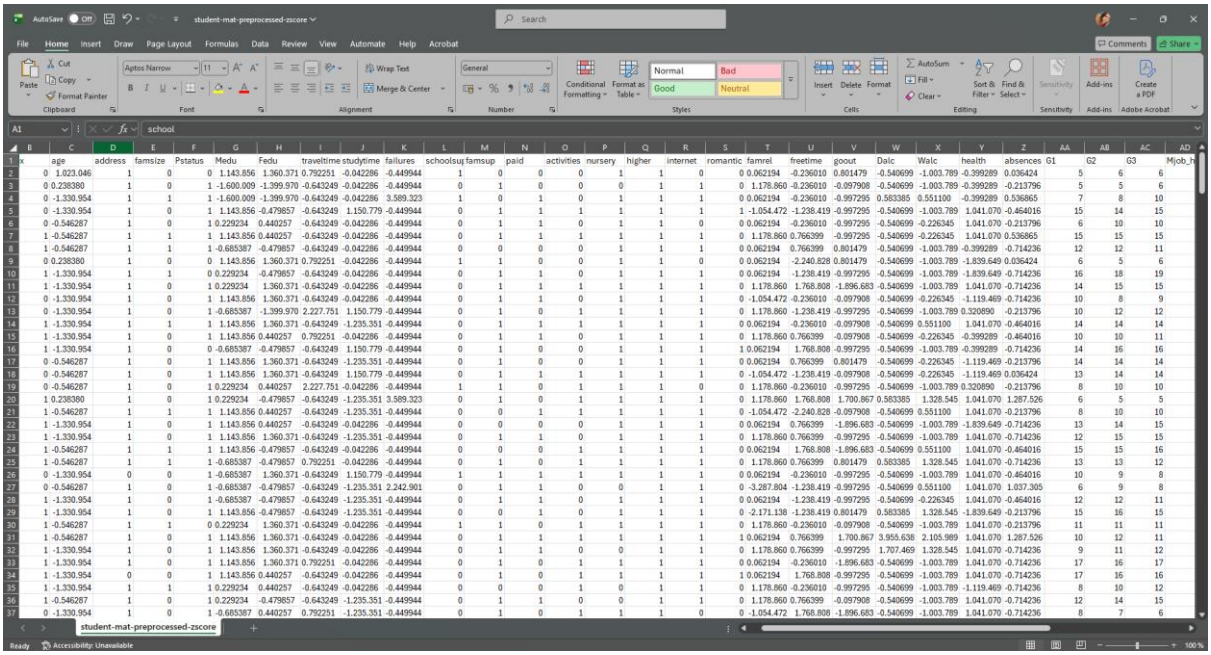
Apoi am aplicat scalarea datelor cu metoda **Min-Max**, pentru a aduce toate valorile numerice pe acelasi interval, intre 0 si 1. Am aplicat transformarea doar pe variabilele de intrare, fara a modifica notele G1, G2 si G3.

```
# scalare Min-Max (NUMAI pe features, NU pe G1,G2,G3)
target_cols = ['G1', 'G2', 'G3']
feat_cols = [c for c in df.columns if c not in target_cols]

scaler = MinMaxScaler()
df_scaled = df.copy()
df_scaled[feat_cols] = scaler.fit_transform(df_scaled[feat_cols])
df_scaled[feat_cols] = df_scaled[feat_cols].round(3)   # lizibil în Excel

print("După scalare: ")
print(df_scaled.head(), "\n")
```

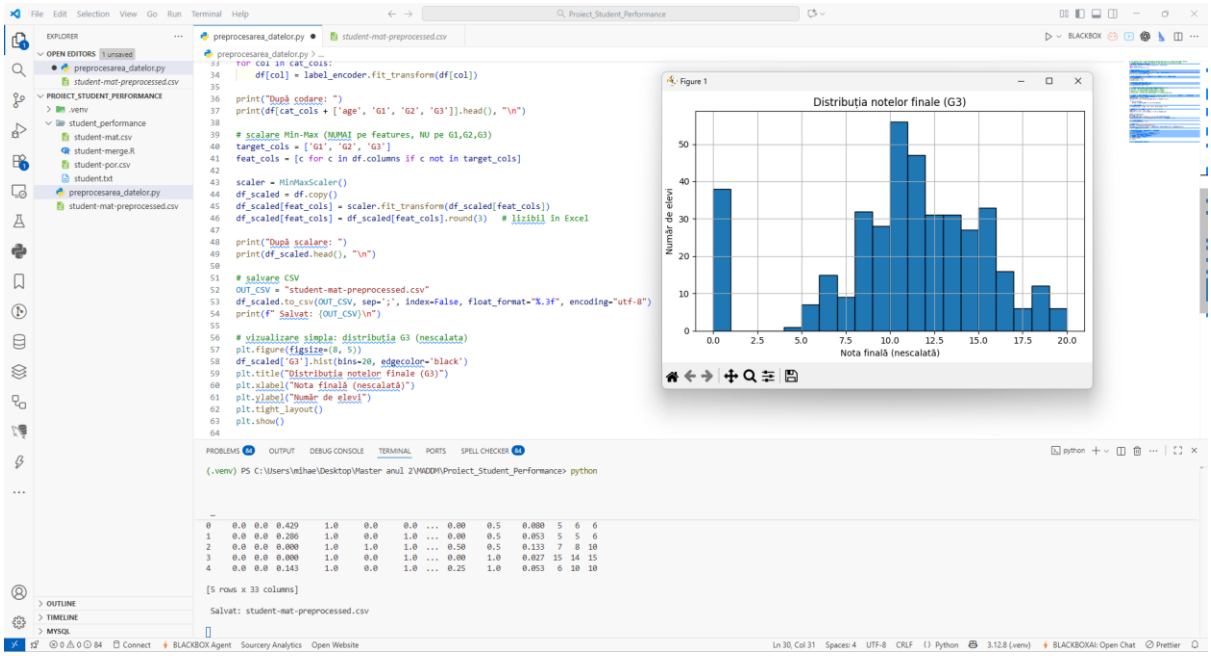
```
PROBLEMS 84  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  SPELL CHECKER 84
(.venv) PS C:\Users\mihae\Desktop\Master anul 2\MADDM\Proiect_Student_Performance> python
…

[5 rows x 21 columns]

După scalare:
   school  sex    age  address  famsize  Pstatus  ...  Walc  health  absences  G1  G2  G3
0     0.0  0.0  0.429      1.0      0.0      0.0  ...  0.00     0.5     0.080   5   6   6
1     0.0  0.0  0.286      1.0      0.0      1.0  ...  0.00     0.5     0.053   5   5   6
2     0.0  0.0  0.000      1.0      1.0      1.0  ...  0.50     0.5     0.133   7   8  10
3     0.0  0.0  0.000      1.0      0.0      1.0  ...  0.00     1.0     0.027  15  14  15
4     0.0  0.0  0.143      1.0      0.0      1.0  ...  0.25     1.0     0.053   6  10  10

[5 rows x 33 columns]

 Salvat: student-mat-preprocessed.csv
```

Ciobanu Mihaela Lavinia
Master Ingineria Datelor, An II

Pentru comparatie, am aplicat si scalarea **Z-score**, care aduce datele la media 0 si deviatia standard 1.

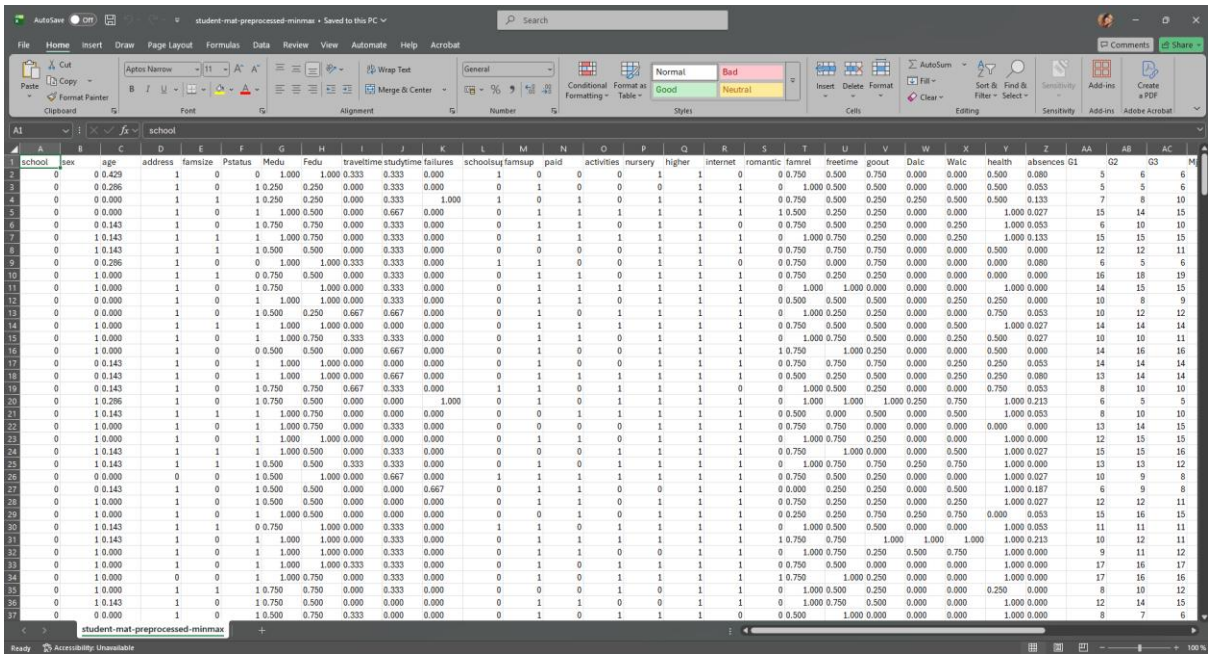


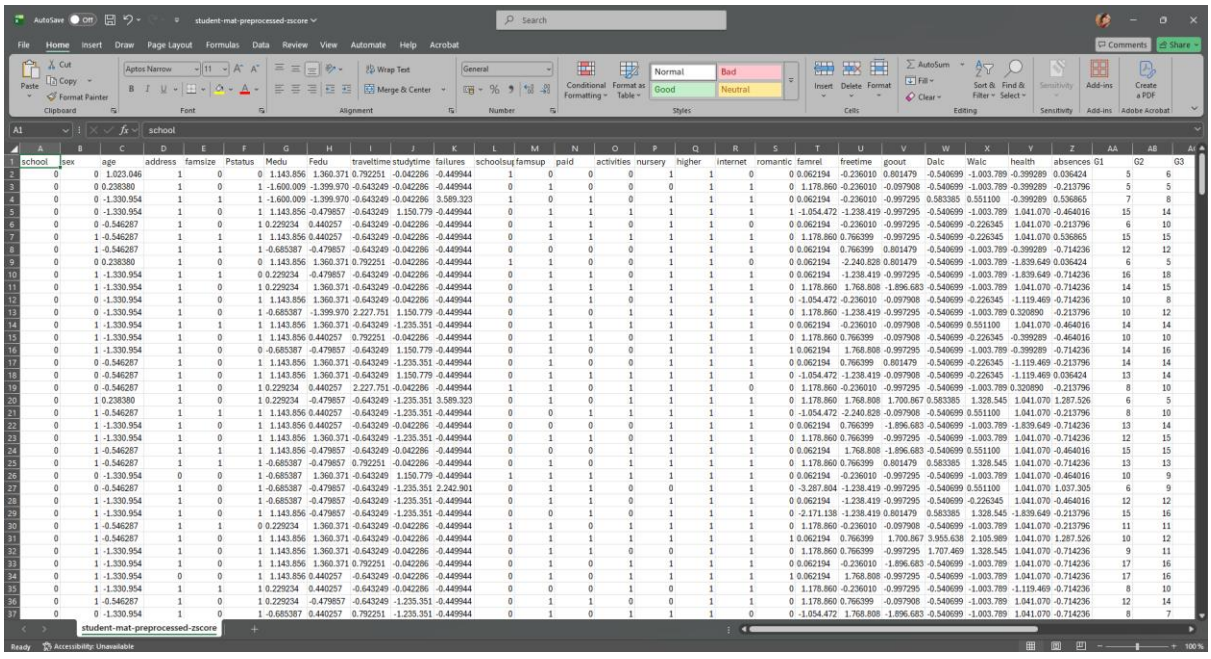
Pentru o vizualizare rapida, am realizat o histograma a coloanei G3, care arata ca majoritatea elevilor au note intre 8 si 15.



Ciobanu Mihaela Lavinia
Master Ingineria Datelor, An II

In final, am salvat datele preprocesate in fisierul student-mat-preprocessed-minmax.csv si student-mat-preprocessed-zscore.csv.





In urma aplicarii celor doua metode de scalare, se observa ca Min-Max pastreaza valorile intr-un interval restrans intre 0 si 1, fiind usor de interpretat vizual, in timp ce Z-score produce valori pozitive si negative, centrate in jurul mediei 0.

Z-score poate fi mai potrivit pentru algoritmi sensibili la distributia datelor, in timp ce Min-Max este mai usor de interpretat vizual.

Ciobanu Mihaela Lavinia
Master Ingineria Datelor, An II

2. Clasificarea

Pentru aceasta parte am continuat sa lucrez pe setul de date preprocesat mai sus, Student_Performance, bazat pe performanta elevilor, asupra caruia am aplicat un algoritm de clasificare.

Am ales algoritmul **Random Forest Classifier,** deoarece ofera o precizie buna si permite observarea influentei parametrilor diferiti asupra predictiei.

Am creat o noua coloana numita target, care indica daca un elev este promovat (1 = nota finala (G3) >= 10) sau nepromovat (0 = G3 < 10).

```
# configurare strategie de cross-validation si metricile de evaluare
cv    # rulam 10-fold cross-validation pentru fiecare configuratie
me    for params in configs:
          model = RandomForestClassifier(random_state=42, **params)
# (     print(f"\nConfiguratie: {params}")
co        res = {}
          for m in metrics:
              scores = cross_val_score(model, X, y, cv=cv, scoring=m)
              mean_score = scores.mean()
              print(f"{m:>9}: {mean_score:.3f}")
              res[m] = mean_score
          results.append({'config': params, **res})
]
```

Pentru evaluare am folosit metoda **10-fold cross-validation**, prin care setul de date este impartit in zece parti pentru antrenare si testare succesiva. Am analizat performanta modelului folosind mai multe metrici: **accuracy, precision, recall si F1-score**.

Am modificat parametrii n_estimators (numărul de arbori) și max_depth (adâncimea maximă a arborilor) pentru a observa efectul asupra performanței modelului. În general, creșterea numărului de arbori a dus la rezultate mai stabile, însă după un anumit prag îmbunătățirile au fost minore.

Pentru datele scalate cu **Min-Max**, cea mai bună configurație a fost n_estimators = 100 și max_depth = 10, cu un scor **F1** mediu de aproximativ **0.82**.

Pentru datele scalate cu **Z-score**, performanțele au fost ușor mai scăzute, scorul **F1** maxim fiind de aproximativ **0.81**.

```
PS C:\Users\mihae\Desktop\master_an2\MADDM\Proiect_Student_Performance> .venv\Scripts\activate
(.venv) PS C:\Users\mihae\Desktop\master_an2\MADDM\Proiect_Student_Performance> python clasificare.py
Evaluare RandomForest pentru fisierul: student-mat-preprocessed-minmax.csv

Configuratie: {'n_estimators': 50, 'max_depth': 5}
 accuracy: 0.694
precision: 0.706
   recall: 0.932
       f1: 0.803

Configuratie: {'n_estimators': 100, 'max_depth': 10}
 accuracy: 0.724
precision: 0.736
   recall: 0.921
       f1: 0.817

Configuratie: {'n_estimators': 200, 'max_depth': None}
 accuracy: 0.717
precision: 0.733
   recall: 0.910
       f1: 0.811

Rezumat configuratii pentru: student-mat-preprocessed-minmax.csv
{'config': {'n_estimators': 50, 'max_depth': 5}, 'accuracy': np.float64(0.6935256410256411), 'precision': np.float64(0.7063813416116049), 'recall': np.float64(0.9319088319088319), 'f1': np.float64(0.8030195395041811)}
{'config': {'n_estimators': 100, 'max_depth': 10}, 'accuracy': np.float64(0.7241666666666667), 'precision': np.float64(0.7362551119097609), 'recall': np.float64(0.9207977207977208), 'f1': np.float64(0.817426814407975)}
{'config': {'n_estimators': 200, 'max_depth': None}, 'accuracy': np.float64(0.7166666666666666), 'precision': np.float64(0.733438820298402 8), 'recall': np.float64(0.9095441595441596), 'f1': np.float64(0.8114620648553617)}

[OK] Rezultate salvate in 'rezultate_clasificare_rf_cv10_student-mat-preprocessed-minmax.csv.csv'
```

```
Evaluare RandomForest pentru fisierul: student-mat-preprocessed-zscore.csv

Configuratie: {'n_estimators': 50, 'max_depth': 5}
       f1: 0.810

Configuratie: {'n_estimators': 100, 'max_depth': 10}
 accuracy: 0.706
precision: 0.723
   recall: 0.913
       f1: 0.807

Configuratie: {'n_estimators': 200, 'max_depth': None}
 accuracy: 0.699
precision: 0.715
   recall: 0.917
       f1: 0.803

Rezumat configuratii pentru: student-mat-preprocessed-zscore.csv
{'config': {'n_estimators': 50, 'max_depth': 5}, 'accuracy': np.float64(0.7061538461538461), 'precision': np.float64(0.7149027312011058), 'recall': np.float64(0.9354700854700855), 'f1': np.float64(0.8100239378128171)}
{'config': {'n_estimators': 100, 'max_depth': 10}, 'accuracy': np.float64(0.7062820512820512), 'precision': np.float64(0.7232473830196791), 'recall': np.float64(0.9131054131054132), 'f1': np.float64(0.8068443498658853)}
{'config': {'n_estimators': 200, 'max_depth': None}, 'accuracy': np.float64(0.6987820512820513), 'precision': np.float64(0.7151239283592224), 'recall': np.float64(0.917094017094017), 'f1': np.float64(0.8033247206488662)}

[OK] Rezultate salvate in 'rezultate_clasificare_rf_cv10_student-mat-preprocessed-zscore.csv.csv'
(.venv) PS C:\Users\mihae\Desktop\master_an2\MADDM\Proiect_Student_Performance>
```

În concluzie, algoritmul Random Forest Classifier s-a dovedit potrivit pentru problema de clasificare a promovării elevilor, oferind rezultate stabile în cadrul validării 10-fold.

Compararea celor două metode de scalare a arătat că Min-Max a oferit rezultate ușor mai bune decât Z-score în acest caz, însă diferențele nu sunt foarte mari (0.01).

Eliminarea notelor G1 și G2 a permis o evaluare corectă a modelului, fără informații care influențează direct nota finală.

3. Clustering

3.1 Rezumat Articol „K-means clustering algorithm and Python implementation”

Articolul "K-means clustering algorithm and Python implementation" prezinta algoritmul K-means, un algoritm clasic de invatare nesupravegheata folosit pentru gruparea automata a datelor in clustere. Principiul de baza este ca elementele din acelasi cluster trebuie sa fie similare, iar distanta, de obicei cea Euclidiana, este utilizata pentru a masura aceasta apropiere.

Autorul descrie propria implementarea a algoritmului, realizata printr-un API si cod in Python, folosind Java pentru generarea datelor experimentale. Procedura urmeaza pasii standard: alegerea aleatorie a K centroizi, atribuirea fiecarui obiect la cel mai apropiat centru si recalcularea centroizilor pe baza mediei, repetand procesul pana la convergenta.

Experimentele au fost efectuate pe un set de date cu un milion de inregistrari, iar rezultatele obtinute au fost similare cu cele generate de implementarea K-means din Apache Spark. Articolul subliniaza si limitarile algoritmului, precum necesitatea stabilirii valorii K si sensibilitatea la valori anormale.

3.2 Aplicarea algoritmului K-means pe setul Student Performance (Python, sklearn)

Am aplicat algoritmul **K-means** pe setul Student Performance preprocesat la inceput, folosind variantele scalate cu **Min-Max** si **Z-score**.

Datele au fost incarcate din fisiere CSV, iar coloanele G1, G2 si G3 au fost eliminate, deoarece K-means este un algoritm nesupravegheat.

Modelul a fost antrenat cu K = 2, 3 si 4, folosind implementarea KMeans din scikit-learn, cu initializare aleatorie a centroizilor. Pentru fiecare valoare K au fost calculate metricele **WCSS** si **Silhouette Score**.

Cele mai bune rezultate au fost obtinute pentru datele scalate cu **Min-Max**, in cazul **K = 4**, unde **Silhouette Score** a fost aproximativ **0.30**, ceea ce arata ca studentii pot fi impartiti mai clar in patru grupuri.

Pentru varianta scalata cu **Z-score**, cea mai buna configuratie a fost **K = 2**, dar cu o separare mai slaba intre grupuri, unde **Silhouette Score** a fost aproximativ **0.15.**

Pentru evaluarea modelului am ales **WCSS** deoarece este metrica folosita direct de algoritmul K-means pentru a forma grupuri cat mai compacte.

Am utilizat si **Silhouette Score** pentru a verifica cat de bine sunt separate grupurile intre ele.

Rezultatele arata ca studentii pot fi impartiti in mai multe categorii, in functie de nivelul de implicare scolara, cum ar fi prezenta la cursuri, timpul de studiu si suportul familial. Chiar daca diferentele nu sunt foarte mari, modelul ajuta la intelegerea modului in care studentii se grupeaza in functie de comportamentul lor scolar.

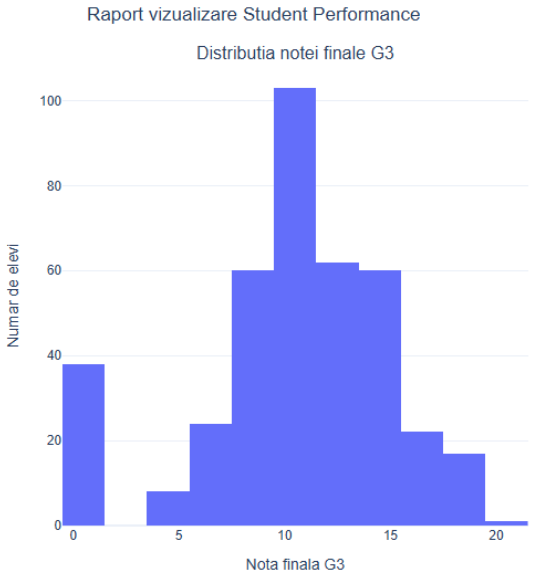
```
PS C:\Users\mihae\Desktop\master_an2\MADDM\Proiect_Student_Performance> .venv\Scripts\activate
(.venv) PS C:\Users\mihae\Desktop\master_an2\MADDM\Proiect_Student_Performance> python kmeans_clustering.py

Clustering K-means pentru fisierul: student-mat-preprocessed-minmax.csv
Dimensiune set de date: (395, 39)

Rulare K-means cu K = 2
Silhouette score: 0.2610633291079305
WCSS: 49486287.903445095

Rulare K-means cu K = 3
Silhouette score: 0.26005901931818287
WCSS: 40159874.09994477

Rulare K-means cu K = 4
Silhouette score: 0.29596813929123594
WCSS: 33104090.517095327
Rezultate metrice salvate in: rezultate_clustering_kmeans_student-mat-preprocessed-minmax.csv.csv
Clustere finale salvate in: clustere_kmeans_K4_student-mat-preprocessed-minmax.csv

Clustering K-means pentru fisierul: student-mat-preprocessed-zscore.csv
Dimensiune set de date: (395, 39)

Rulare K-means cu K = 2
Silhouette score: 0.15302502011299593
WCSS: 4510302412718668.0

Rulare K-means cu K = 3
Silhouette score: 0.1045560419200294
WCSS: 4126518451452694.0

Rulare K-means cu K = 4
Silhouette score: 0.09622254738234111
WCSS: 3885829183780169.0
Rezultate metrice salvate in: rezultate_clustering_kmeans_student-mat-preprocessed-zscore.csv.csv
Clustere finale salvate in: clustere_kmeans_K2_student-mat-preprocessed-zscore.csv
```

Ciobanu Mihaela Lavinia
Master Ingineria Datelor, An II

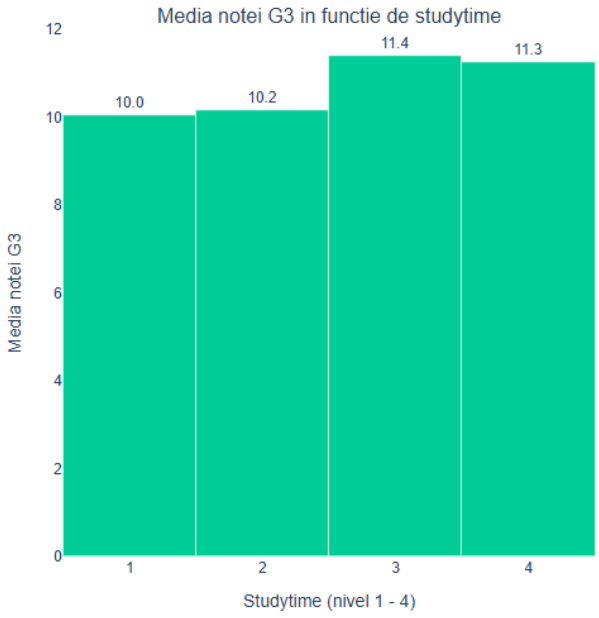
4. Vizualizarea Datelor

Setul de date Student Performance contine informatii despre elevi si nota finala G3.

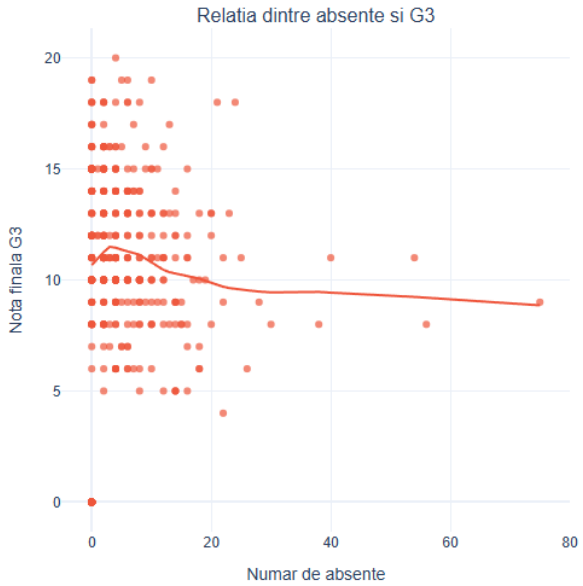
In primul grafic, histograma notei G3 arata o distributie concentrata in zona valorilor medii, cu cei mai multi elevi avand note intre aproximativ 8 si 14. Se observa si un numar vizibil de elevi cu nota finala G3 egala cu 0, care corespund cazurilor de nepromovare sau neprezentare la evaluare si nu reprezinta erori in setul de date.



Al doilea grafic, prezinta media notei G3 pe fiecare nivel de studytime, evidentiaza o crestere a mediei notei finale pe masura ce creste timpul de studiu. Elevii din grupele cu studytime 3 si 4 obtin, in medie, rezultate mai bune decat cei din grupele 1 si 2, ceea ce sugereaza ca timpul de studiu are un impact pozitiv asupra performantei scolare.



Al treilea grafic surprinde relatia dintre numarul de absente si nota finala G3, impreuna cu o curba de trend. Curba indica o relatie slaba si neliniara intre absente si performanta, fara a sugera o legatura cauzala directa. In ansamblu, graficele arata ca timpul de studiu influenteaza mai clar rezultatele finale decat numarul de absente.



5. Concluzii

In concluzie, analiza setului de date Student Performance a aratat ca timpul de studiu este un factor mai relevant pentru performanta scolara decat absentele.

Algoritmul Random Forest a oferit rezultate stabile pentru clasificarea promovarii elevilor, iar clustering-ul K-means a permis identificarea unor grupuri de elevi cu comportamente similare.

Vizualizarea datelor a completat analiza, oferind o interpretare clara si intuitiva a rezultatelor obtinute.