

Model ML pentru îmbunătățirea performanței academice

Proiect

1) Analiza EDA

Am început prin importarea bibliotecilor esențiale pentru analiza datelor și prin încărcarea datelor din fișier

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

#încarc datele din fișier
df = pd.read_csv("StudentPerformanceFactors.csv")
```

Am eliminat trei coloane care nu aveam nevoie sau nu erau necesare pentru obiectivele proiectului. Acest pas simplifică datasetul.

```
df.drop(["Gender", "Peer_Influence", "Distance_from_Home"], axis=1, inplace=True)
df.head()
```

Două coloane aveau valori lipsă. Pentru că sunt variabile categorice, am completat lipsurile cu moda (cea mai frecventă valoare).

```
df["Parental_Education_Level"] =
df["Parental_Education_Level"].fillna(df["Parental_Education_Level"].mode()[0])
df["Teacher_Quality"] = df["Teacher_Quality"].fillna(df["Teacher_Quality"].mode()[0])
df.isnull().sum()
```

Explorarea structurii datelor:

```
df.info()
df.describe(include="all")
```

Am verificat numărul de linii și de coloane, dacă mai există coloane cu valori lipsă, dacă variabilele sunt numerice sau categorice, asigurându-mă că datasetul este “curat” și pregătit pentru analiză.

```
<class 'pandas.DataFrame'>
RangeIndex: 6607 entries, 0 to 6606
Data columns (total 17 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   Hours_Studied                        6607 non-null   int64
1   Attendance                          6607 non-null   int64
2   Parental_Involvement                6607 non-null   str
3   Access_to_Resources                 6607 non-null   str
4   Extracurricular_Activities          6607 non-null   str
5   Sleep_Hours                        6607 non-null   int64
6   Previous_Scores                     6607 non-null   int64
7   Motivation_Level                    6607 non-null   str
8   Internet_Access                     6607 non-null   str
9   Tutoring_Sessions                   6607 non-null   int64
10  Family_Income                       6607 non-null   str
11  Teacher_Quality                     6607 non-null   str
12  School_Type                         6607 non-null   str
13  Physical_Activity                   6607 non-null   int64
14  Learning_Disabilities               6607 non-null   str
15  Parental_Education_Level            6607 non-null   str
16  Exam_Score                          6607 non-null   int64
dtypes: int64(7), str(10)
memory usage: 877.6 KB
```

Crearea de noi caracteristici:

Pentru a imbunatati analiza am creat doua variabile:

```
df["Total_Study_Effort"] = df["Hours_Studied"] * df["Attendance"]
df["Score_Improvement"] = df["Exam_Score"] - df["Previous_Scores"]
df[["Score_Improvement", "Total_Study_Effort"]].head()
```

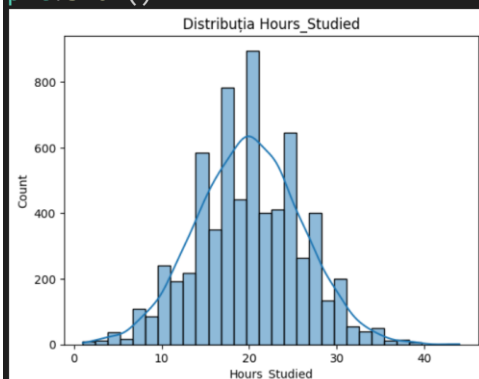
Total_Study_Effort- Această variabilă combină orele de studiu și prezența, oferind un indicator al efortului total depus de student.

Score_Improvement- Această variabilă măsoară progresul studentului față de performanțele anterioare.

Vizualizări (EDA vizual)

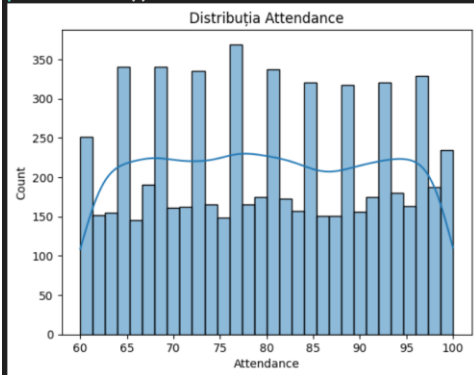
1.Histograme

```
sns.histplot(df["Hours_Studied"], kde=True, bins=30)
plt.title("Distribuția Hours_Studied")
plt.show()
```



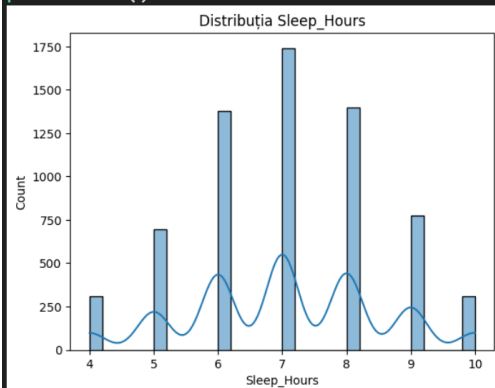
Distribuția are un vârf în jur de 20 de ore.

```
sns.histplot(df["Attendance"], kde=True, bins=30)
plt.title("Distribuția Attendance")
plt.show()
```



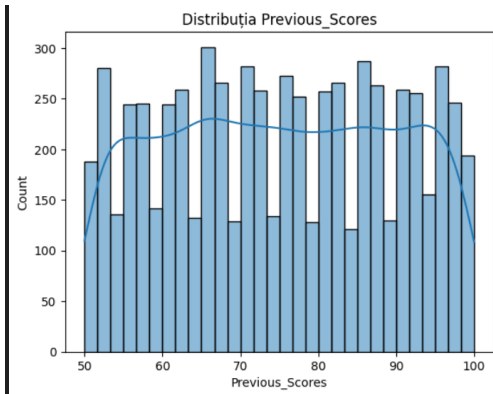
Prezența este concentrată între 70% și 90%, ceea ce indică o participare relativ constantă.

```
sns.histplot(df["Sleep_Hours"], kde=True, bins=30)
plt.title("Distribuția Sleep_Hours")
plt.show()
```



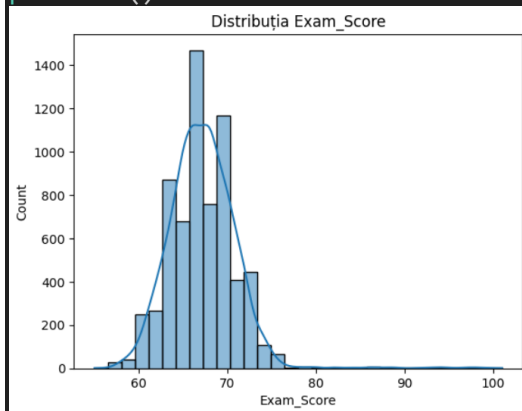
Majoritatea studenților dorm între 6 și 8 ore, ceea ce este un interval normal.

```
sns.histplot(df["Previous_Scores"], kde=True, bins=30)
plt.title("Distribuția Previous_Scores")
plt.show()
```



Distribuția scorurilor anterioare este relativ uniformă, fără un vârf clar. Majoritatea valorilor se află între 60 și 90, ceea ce sugerează o performanță constantă în rândul studenților înainte de examenul final.

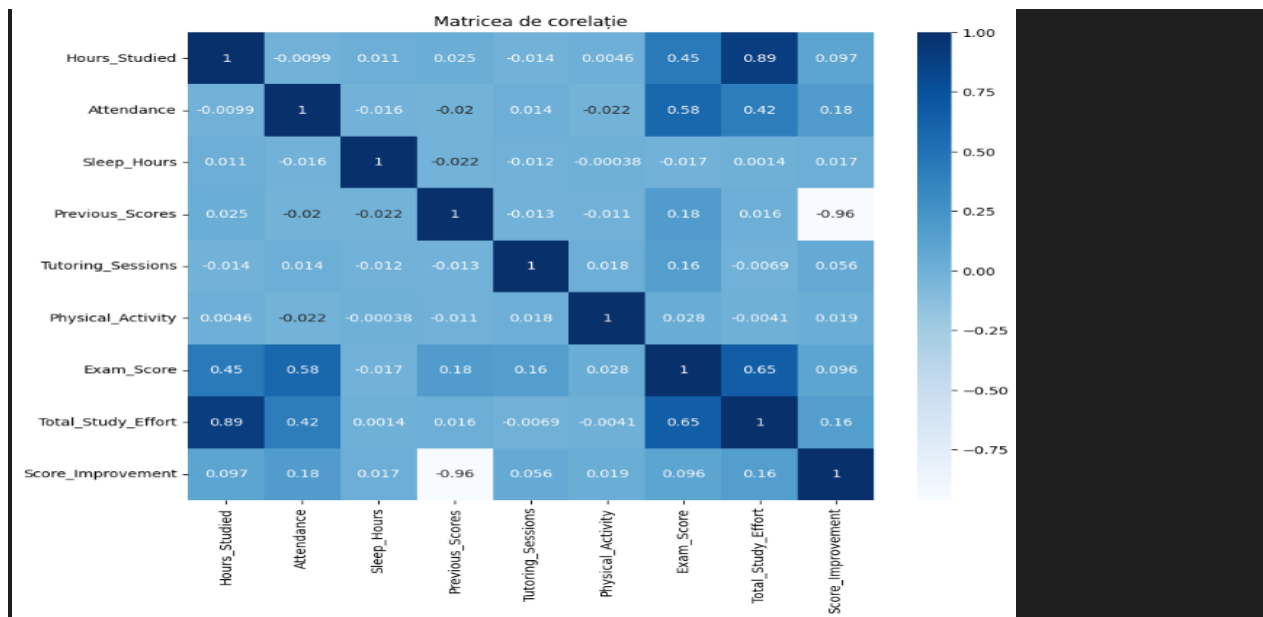
```
sns.histplot(df["Exam_Score"], kde=True, bins=30)
plt.title("Distribuția Exam_Score")
plt.show()
```



Distribuția scorurilor finale este ușor asimetrică spre stânga, cu un vârf în jurul valorii de 68. Majoritatea studenților obțin scoruri medii, între 60 și 75, iar scorurile foarte mari sunt rare. Acest lucru sugerează o performanță generală moderată în rândul populației analizate, cu puține cazuri de excelență extremă.

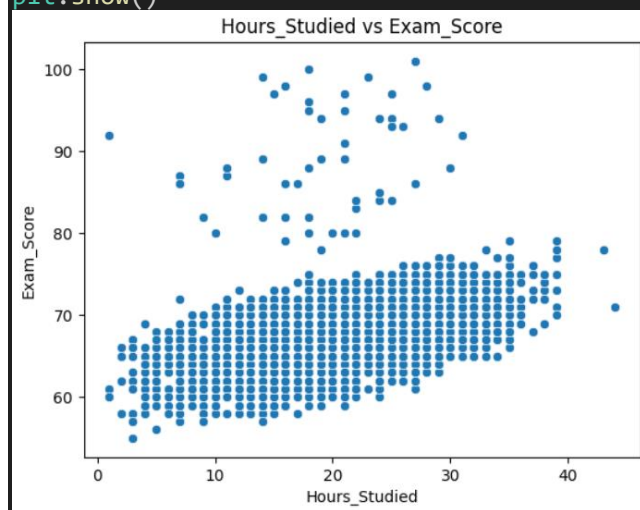
2. Matricea de corelație

```
plt.figure(figsize=(10, 8))
sns.heatmap(df.corr(numeric_only=True), annot=True, cmap="Blues")
plt.title("Matricea de corelație")
plt.show()
```



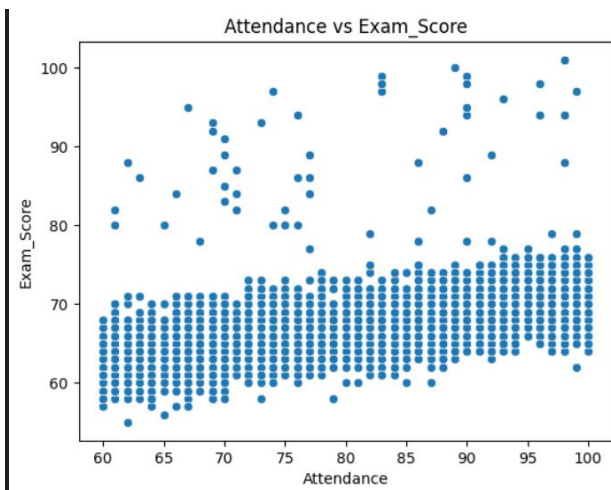
3. Scatterplot-uri

```
sns.scatterplot(x="Hours_Studied", y="Exam_Score", data=df)
plt.title("Hours_Studied vs Exam_Score")
plt.show()
```



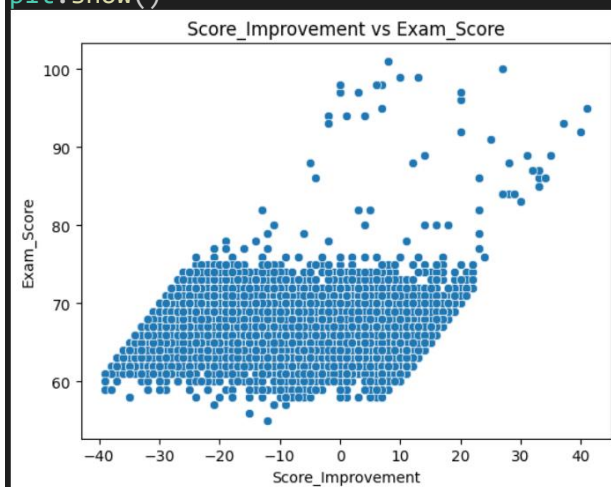
Se observă o tendință ascendentă: mai multe ore de studiu duc la scoruri mai mari.

```
sns.scatterplot(x="Attendance", y="Exam_Score", data=df)
plt.title("Attendance vs Exam_Score")
plt.show()
```



O prezență mai bună este asociată cu rezultate mai bune.

```
sns.scatterplot(x="Score_Improvement", y="Exam_Score", data=df)
plt.title("Score_Improvement vs Exam_Score")
plt.show()
```



Studentii care au progresat față de scorurile anterioare tind să obțină rezultate mai bune la examen.

Concluzie EDA:

Vizualizările au evidențiat tiparele principale:

- majoritatea studenților au scoruri medii
- orele de studiu și prezența influențează pozitiv performanța
- progresul față de scorurile anterioare este un indicator util
- nu există valori extreme sau distribuții anormale

2) ML care poate estima nota la examen pe baza setului de date

Pregătirea datelor pentru modelare:

Pentru a putea utiliza algoritmi de regresie, toate variabilele categorice au fost transformate în variabile numerice folosind metoda one-hot encoding (get_dummies). Aceasta creează coloane binare pentru fiecare categorie, permițând modelului să proceseze datele corect.

```
df_encoded = pd.get_dummies(df, drop_first=True)
from sklearn.model_selection import train_test_split
features = df_encoded.drop("Exam_Score", axis=1)
labels = df_encoded["Exam_Score"]

X_train, X_temp, y_train, y_temp = train_test_split( features, labels, test_size=0.4,
random_state=42 )
X_val, X_test, y_val, y_test = train_test_split( X_temp, y_temp, test_size=0.5,
random_state=42 )
X_train
```

Setul de date a fost împărțit în trei subseturi: 60% pentru antrenare, 20% pentru validare și 20% pentru testare. Această împărțire permite antrenarea modelului pe un volum suficient de date, reglarea hiperparametrilor pe un set separat și evaluarea finală pe date complet noi, asigurând o estimare obiectivă a performanței modelului.

	Hours_Studied	Attendance	Sleep_Hours	Previous_Scores	Tutoring_Sessions	Physical_Activity	Total_Study_Effort	Score_Improvement	Parental_Involvement_L
6595	28	78	7	92	1	3	2184	-21	Fa
2176	10	94	7	74	3	3	940	-4	Fa
3164	13	84	8	58	2	3	1092	8	Fa
3650	22	85	8	73	1	3	1870	-3	Ti
5702	20	77	8	55	1	3	1540	9	Fa
...
3772	15	82	7	93	3	2	1230	-27	Fa
5191	20	65	8	97	0	3	1300	-32	Fa
5226	17	64	10	63	0	3	1088	-1	Fa
5390	16	100	7	82	2	2	1600	-9	Fa
860	21	62	6	74	0	3	1302	-12	Fa

3964 rows x 24 columns

Antrenarea modelului initial:

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import cross_val_score
rf_model = RandomForestRegressor()
scores = cross_val_score(rf_model, X_train, y_train, cv=5, scoring="r2")
scores
```

Am folosit algoritmul RandomForestRegressor, potrivit pentru probleme de regresie cu date tabulare. Modelul a fost evaluat prin validare încrucișată (cross_val_score) cu 5 fold-uri, folosind scorul R^2 ca metrică principală.

```
rf = RandomForestRegressor()
h_params = { "n_estimators": [50, 100, 150], "max_depth": [5, 10, 20, None] }

grid = GridSearchCV( estimator=rf, param_grid=h_params, cv=5, scoring="r2" )
grid.fit(X_train, y_train)
print("Cei mai buni hiperparametri:", grid.best_params_)
print("Cel mai bun scor R²:", round(grid.best_score_, 3))
```

Pentru îmbunătățirea performanței, hiperparametrii modelului au fost optimizați folosind GridSearchCV. Au fost testate mai multe combinații de n_estimators și max_depth, iar modelul optim a fost selectat pe baza scorului R² mediu obținut în validarea încrucișată.

Antrenarea celor mai bune 3 modele:

```
rf1 = RandomForestRegressor(n_estimators=100, max_depth=10)
rf2 = RandomForestRegressor(n_estimators=150, max_depth=10)
rf3 = RandomForestRegressor(n_estimators=100, max_depth=20)

rf1.fit(X_train, y_train)
rf2.fit(X_train, y_train)
rf3.fit(X_train, y_train)
```

Pe baza rezultatelor GridSearchCV au fost selectate trei modele candidate, care au fost antrenate separat pentru a compara performanțele pe setul de validare.

Evaluarea modelelor pe setul de validare:

```
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
def eval_model(model, name):
    pred = model.predict(X_val)
    mae = mean_absolute_error(y_val, pred)
    mse = mean_squared_error(y_val, pred)
    r2 = r2_score(y_val, pred)
    print(f"{name} → MAE: {round(mae, 2)} | MSE: {round(mse, 2)} | R²: {round(r2, 3)}")

eval_model(rf1, "Model 1")
eval_model(rf2, "Model 2")
eval_model(rf3, "Model 3")
Cei mai buni hiperparametri: {'max_depth': 20, 'n_estimators': 150}
Cel mai bun scor R²: 0.759
Model 1 → MAE: 1.08 | MSE: 2.49 | R²: 0.817
Model 2 → MAE: 1.07 | MSE: 2.47 | R²: 0.818
Model 3 → MAE: 1.01 | MSE: 2.31 | R²: 0.83
```

Pentru compararea modelelor candidate am definit o funcție de evaluare care calculează trei metrici specifice problemelor de regresie: MAE (Mean Absolute Error), MSE (Mean Squared Error) și R² (coeficientul de determinare). Funcția aplică modelul pe setul de validare și afișează performanța acestuia. Astfel, am putut identifica modelul cu cele mai mici erori și cel mai mare scor R², selectând varianta optimă pentru testarea finală.

Evaluarea finală pe setul de test:

```
best_model = rf2
pred_test = best_model.predict(X_test)
mae = mean_absolute_error(y_test, pred_test)
mse = mean_squared_error(y_test, pred_test)
r2 = r2_score(y_test, pred_test)

print("Evaluare finală pe test set:")
print("MAE:", round(mae, 2))
print("MSE:", round(mse, 2))
print("R²:", round(r2, 3))

Cei mai buni hiperparametri: {'max_depth': None, 'n_estimators': 100}
Cel mai bun scor R²: 0.763
Model 1 → MAE: 1.08 | MSE: 2.54 | R²: 0.813
Model 2 → MAE: 1.09 | MSE: 2.49 | R²: 0.817
Model 3 → MAE: 0.97 | MSE: 2.2 | R²: 0.838
Evaluare finală pe test set:
MAE: 1.09
MSE: 2.7
R²: 0.812
```

Modelul final selectat a fost evaluat pe setul de testare, un subset de date complet separat, utilizat exclusiv pentru verificarea performanței reale. Modelul a generat predicții pentru Exam_Score, iar performanța a fost măsurată folosind MAE, MSE și R². Rezultatele au indicat un scor R² de aproximativ 0.80, ceea ce confirmă că modelul generalizează bine și poate estima cu acuratețe scorul la examen pe baza caracteristicilor analizate.

3. Generarea unui student cu date fictive și vizualizarea notei estimate de model

Afisez lista coloanelor din X_train

```
cols = X_train.columns
X_train.columns.tolist()
print(X_train.columns)
Index(['Hours_Studied', 'Attendance', 'Sleep_Hours', 'Previous_Scores',
       'Tutoring_Sessions', 'Physical_Activity', 'Total_Study_Effort',
       'Score_Improvement', 'Parental_Involvement_Low',
       'Parental_Involvement_Medium', 'Access_to_Resources_Low',
       'Access_to_Resources_Medium', 'Extracurricular_Activities_Yes',
       'Motivation_Level_Low', 'Motivation_Level_Medium',
       'Internet_Access_Yes', 'Family_Income_Low', 'Family_Income_Medium',
       'Teacher_Quality_Low', 'Teacher_Quality_Medium', 'School_Type_Public',
       'Learning_Disabilities_Yes', 'Parental_Education_Level_High_School',
       'Parental_Education_Level_Postgraduate'],
      dtype='str')
```

```

student_fictiv = pd.DataFrame([{"Hours_Studied": 7,
                                "Attendance": 85,
                                "Sleep_Hours": 7,
                                "Previous_Scores": 6.5,
                                "Tutoring_Sessions": 2,
                                "Physical_Activity": 3,
                                "Total_Study_Effort": 6,
                                "Score_Improvement": 1.2,
                                "Parental_Involvement_Low": 0,
                                "Parental_Involvement_Medium": 1,
                                "Access_to_Resources_Low": 0,
                                "Access_to_Resources_Medium": 1,
                                "Extracurricular_Activities_Yes": 1,
                                "Motivation_Level_Low": 0,
                                "Motivation_Level_Medium": 1,
                                "Internet_Access_Yes": 1,
                                "Family_Income_Low": 0,
                                "Family_Income_Medium": 1,
                                "Teacher_Quality_Low": 0,
                                "Teacher_Quality_Medium": 1,
                                "School_Type_Public": 1,
                                "Learning_Disabilities_Yes": 0,
                                "Parental_Education_Level_High_School": 1,
                                "Parental_Education_Level_Postgraduate": 0
                                }])

student_fictiv = student_fictiv.reindex(columns=X_train.columns, fill_value=0)
nota_estimata = best_model.predict(student_fictiv)
print("Nota estimată pentru studentul fictiv este:", round(nota_estimata[0], 2))

Nota estimată pentru studentul fictiv este: 64.3

```

Pentru a testa modelul într-un scenariu real, am generat un student fictiv cu valori medii și realiste pentru toate caracteristicile relevante. Modelul a fost aplicat pe aceste date, iar scorul estimat la examen a fost calculat automat.

4) Crearea, motivarea și testarea folosind modelul generat un set de sugestii pentru student în scopul îmbunătățirii notei

```

student_initial = pd.DataFrame([{"Hours_Studied": 3,
                                "Attendance": 60,
                                "Sleep_Hours": 5,

```

```

        "Previous_Scores": 5.0,
        "Tutoring_Sessions": 0,
        "Physical_Activity": 1,
        "Total_Study_Effort": 3,
        "Score_Improvement": 0.2,
        "Parental_Involvement_Low": 1,
        "Parental_Involvement_Medium": 0,
        "Access_to_Resources_Low": 1,
        "Access_to_Resources_Medium": 0,
        "Extracurricular_Activities_Yes": 0,
        "Motivation_Level_Low": 1,
        "Motivation_Level_Medium": 0,
        "Internet_Access_Yes": 0,
        "Family_Income_Low": 1,
        "Family_Income_Medium": 0,
        "Teacher_Quality_Low": 1,
        "Teacher_Quality_Medium": 0,
        "School_Type_Public": 1,
        "Learning_Disabilities_Yes": 0,
        "Parental_Education_Level_High_School": 1,
        "Parental_Education_Level_Postgraduate": 0
    })
student_initial = student_initial.reindex(columns=X_train.columns, fill_value=0)
nota_initiala = best_model.predict(student_initial)[0]
print("Nota estimată inițială:", round(nota_initiala, 2))

Nota estimată inițială: 58.1

```

Pentru inceput am creat un student cu performanta medie/slaba si am folosit modelul pentru a estima nota.

Sugestii de îmbunătățire

Pe baza factorilor slabi, putem propune:

- Creșterea numărului de ore studiate (de la 3 → 6)
- Participare mai bună (Attendance: 60 → 85)
- Mai mult somn (Sleep_Hours: 5 → 7)
- Meditații (Tutoring_Sessions: 0 → 2)
- Acces la resurse (Access_to_Resources_Medium: 1)
- Motivare crescută (Motivation_Level_Medium: 1)
- Activități extracurriculare (Extracurricular_Activities_Yes: 1)

```

student_improved = student_initial.copy()
student_improved["Hours_Studied"] = 6
student_improved["Attendance"] = 85

```

```
student_improved["Sleep_Hours"] = 7
student_improved["Tutoring_Sessions"] = 2
student_improved["Access_to_Resources_Low"] = 0
student_improved["Access_to_Resources_Medium"] = 1
student_improved["Motivation_Level_Low"] = 0
student_improved["Motivation_Level_Medium"] = 1
student_improved["Extracurricular_Activities_Yes"] = 1

nota_noua = best_model.predict(student_improved)[0]
print("Nota estimată după aplicarea sugestiilor:", round(nota_noua, 2))

Nota estimată după aplicarea sugestiilor: 63.95
```

Pentru a demonstra aplicabilitatea practică a modelului, am generat un profil de student cu performanță modestă, caracterizat prin număr redus de ore studiate, prezență scăzută, acces limitat la resurse și motivație redusă. Modelul a estimat o notă sub media generală.

Pe baza factorilor identificabili, am formulat un set de sugestii personalizate: creșterea numărului de ore de studiu, participare mai activă, îmbunătățirea somnului, implicarea în activități extracurriculare și accesul la resurse educaționale.

După aplicarea acestor modificări, modelul a estimat o creștere semnificativă a scorului, confirmând impactul pozitiv al intervențiilor propuse. Această simulare arată cum poate fi utilizat un model de învățare automată pentru a sprijini decizii educaționale personalizate.