

Compararea modelelor de predictie aplicate pe un singur set de date

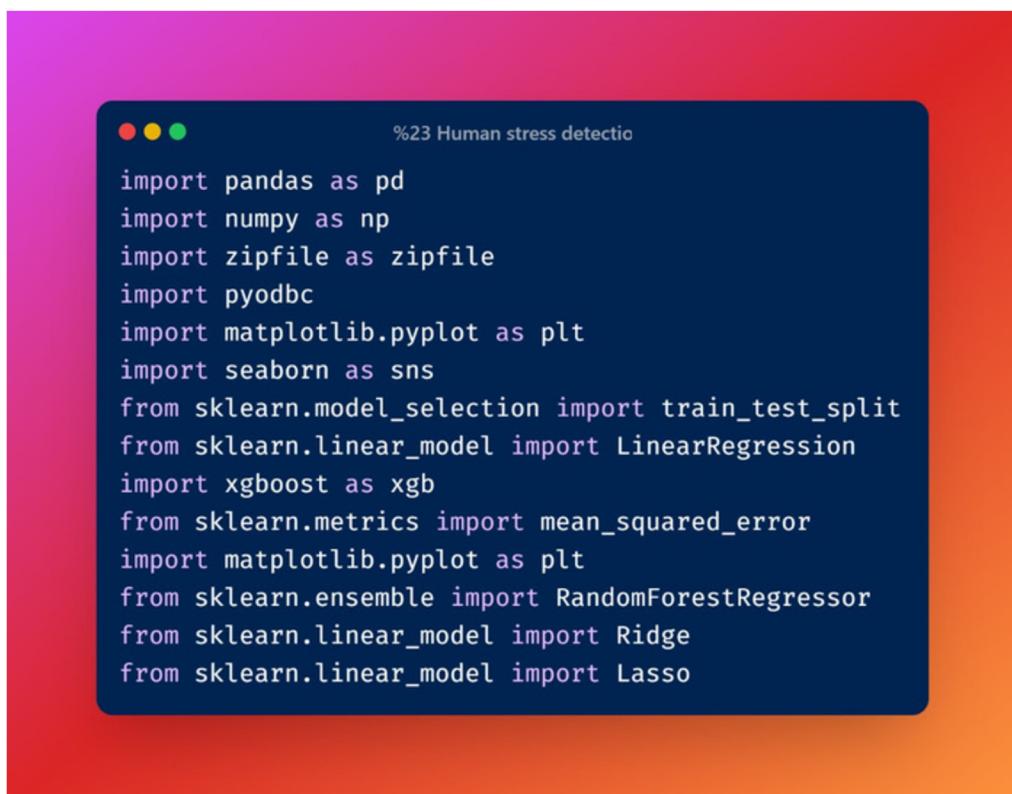
În acest proiect am vrut să arăt diferența dintre acuratețea a cinci modele de predicție diferite, acestea fiind :

- XGBoost**
- RandomForest**
- Regresia Ridge**
- Regresia Lasso**
- Regresia Lineară**

Proiectul este scris aproape în întregime între-un script de python și urmărește următoarea structură :

1. Procesarea Inițială a Datelor

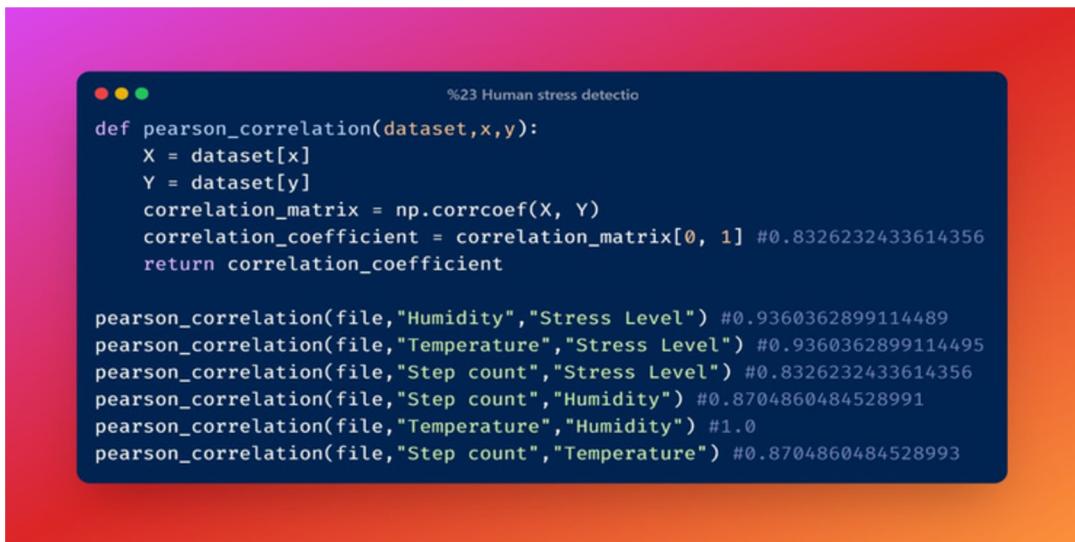
- Importarea bibliotecilor necesare;



```
● ● ● %23 Human stress detectio
import pandas as pd
import numpy as np
import zipfile as zipfile
import pyodbc
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
import xgboost as xgb
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import Ridge
from sklearn.linear_model import Lasso
```

- Extractia datelor dintr-un fisier zip
- Transformarea valorilor temperaturii din Fahrenheit in Celsius
- Verificarea datelor pentru a asigura ca nu exista valori NA si tipurile de date sunt corecte
- Crearea de grafice pentru a vizualiza distributia datelor si anomaliiile
- Calcularea coeficientilor de corelatie Pearson si vizualizarea lor folosind un heatmap

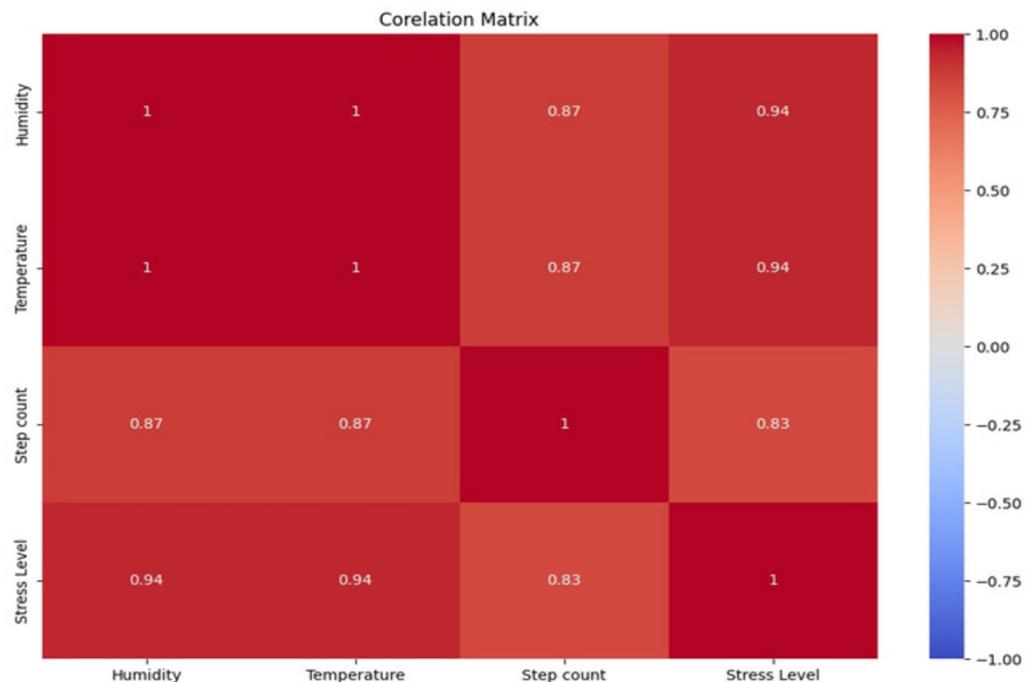
Functia pentru vizualizarea coeficientului de corelatie si coeficientul mai jos:



```
%23 Human stress detection
def pearson_correlation(dataset,x,y):
    X = dataset[x]
    Y = dataset[y]
    correlation_matrix = np.corrcoef(X, Y)
    correlation_coefficient = correlation_matrix[0, 1] #0.8326232433614356
    return correlation_coefficient

pearson_correlation(file,"Humidity","Stress Level") #0.9360362899114489
pearson_correlation(file,"Temperature","Stress Level") #0.9360362899114495
pearson_correlation(file,"Step count","Stress Level") #0.8326232433614356
pearson_correlation(file,"Step count","Humidity") #0.8704860484528991
pearson_correlation(file,"Temperature","Humidity") #1.0
pearson_correlation(file,"Step count","Temperature") #0.8704860484528993
```

Vizualizarea Heatmap pentru corelatie:



- Căutarea anomaliei și eliminarea acestora

```
%23 Human stress detectio

def inner_quantile(file, variable):
    Q1 = file[variable].quantile(0.25)
    Q3 = file[variable].quantile(0.75)
    IQR = Q3-Q1
    outliers = file[(file[variable]<(Q1-1.5*IQR)) | (file[variable]>(Q3+1.5*IQR))]
    print(f"IQR = {IQR}, Lower={Q1 -1.5 * IQR}, Higher={Q3 +1.5 * IQR}")
    return outliers
inner_quantile(file,"Temperature")
inner_quantile(file,"Step count")
inner_quantile(file,"Humidity")
inner_quantile(file,"Stress Level")

# No outliers found with this method

# Z score method

def z_score_outliers(file, variable):
    mean = file[variable].mean()
    std = file[variable].std()
    file['z_score'] = (file[variable] - mean) / std
    outliers = file[(file['z_score'] > 3) | (file['z_score'] < -3)]
    print(f"Mean = {mean}, Std = {std}")
    return outliers

print(z_score_outliers(file, "Temperature"))
print(z_score_outliers(file, "Step count"))
print(z_score_outliers(file, "Humidity"))
print(z_score_outliers(file, "Stress Level"))
```

2.Crearea și Inserarea Datelor în Baza de Date:

- Conectarea la o bază de date locală

```
%23 Human stress detectio

conn = pyodbc.connect(
    'DRIVER={ODBC Driver 17 for SQL Server};'
    'SERVER=(localdb)\local;'
    'DATABASE=StressData;'
    'Trusted_Connection=yes;'
)
```

- **Crearea tabelelor StressMetrics și People**
- **Inserarea valorilor din fișierele CSV și Excel în tabelele respective**
- **Stabilirea unei relații de cheie străină între StressMetrics și People**

```
stressMetricsTable_Create_Query = '''  
CREATE TABLE StressMetrics (  
    ID int,  
    Humidity float,  
    Temperature float,  
    StepCount int,  
    StressLevel int  
)'''  
conn.execute(stressMetricsTable_Create_Query)  
conn.commit()  
query = "SELECT * FROM dbo.StressMetrics"  
df = pd.read_sql(query, conn)  
cursor = conn.cursor()  
cursor.execute('SET IDENTITY_INSERT dbo.StressMetrics ON')  
for index, row in file.iterrows():  
    cursor.execute('''  
        Insert INTO dbo.StressMetrics (ID,Humidity,Temperature,StepCount,StressLevel)  
        VALUES(?, ?, ?, ?, ?)  
    ''',row['id'],row["Humidity"],row["Temperature"],row["Step count"],row["Stress Level"])  
cursor.execute('SET IDENTITY_INSERT dbo.StressMetrics OFF')  
conn.commit()  
cursor.close()  
conn.close()
```

```
%23 Human stress detectio
cursor.execute('SET IDENTITY_INSERT dbo.StressMetrics ON')
cursor.execute(
    ...
ALTER TABLE dbo.StressMetrics
ADD CONSTRAINT FK_StressMetrics_People
FOREIGN KEY (PersonID) REFERENCES People(ID);
...
)
cursor.execute('SET IDENTITY_INSERT dbo.StressMetrics OFF')
conn.commit()
cursor.close()
```

3. Modele Predictive:

- **Pregatirea seturilor de date pentru antrenarea modelelor și alegem variabilele independente cu ajutorul cărora facem predicția ;**

```
%23 Human stress detectio
join_Query = '''
SELECT * FROM dbo.People p
JOIN dbo.StressMetrics s
ON p.ID = s.PersonID ;
'''

joined_df = pd.read_sql(join_Query,conn)
conn.close()

joined_df.describe()
# Creating a prediction on the stress levels based on the independent variables
x= joined_df[["Humidity","Temperature","StepCount"]]
y= joined_df[["StressLevel"]]

# Getting the test and train sets ready
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state=42)
```

- **Calculăm eroarea medie patratică și coeficientul de determinare pentru a observa eficacitatea modelului înainte de a-l aplica pe setul de date;**
- **Vom crea un tabel nou pentru fiecare dintre predicțiile noastre la care vom adăuga un field nou de acuratețe prin care**
- **Calculam acuratețea modelului raportat la valorile din setul initial**

```

%23 Human stress detection
cursor = conn.cursor()
createXGboostTable_Query = '''
CREATE TABLE StressXGBoostPredictions (
ID int Primary Key Identity(1,1),
PersonID int,
Humidity float,
Temperature float,
StepCount int,
PredictedStressLevel float
)
...
conn.execute(createXGboostTable_Query)
conn.commit()

createForeignKey_Query = '''
ALTER TABLE dbo.StressXGBoostPredictions
ADD CONSTRAINT FK_StressXGBoostPredictions_People
FOREIGN KEY (PersonID) REFERENCES People(ID)
...
conn.execute(createForeignKey_Query)
cursor = conn.cursor()
conn.commit()
cursor.execute('SET IDENTITY_INSERT dbo.StressXGBoostPredictions ON')
conn.execute(createForeignKey_Query)
for index, row in xgboostPredictionDF.iterrows():
    cursor.execute('''
    INSERT INTO StressXGBoostPredictions(ID,PersonID,Humidity,Temperature,StepCount,PredictedStressLevel)
    VALUES (?,?,?,?,?,?)''', row["PersonID"],row["PersonID"],row["Humidity"],row["Temperature"],row["StepCount"],row["StressLevelPredicted"])
)
conn.commit()
cursor.execute('SET IDENTITY_INSERT dbo.StressXGBoostPredictions OFF')
addAccuracy_Query=''''
ALTER TABLE dbo.StressXGBoostPredictions
ADD Accuracy INT;
...

updateAccuracyValue_Query = '''
DECLARE @epsilon FLOAT = 0.0001;

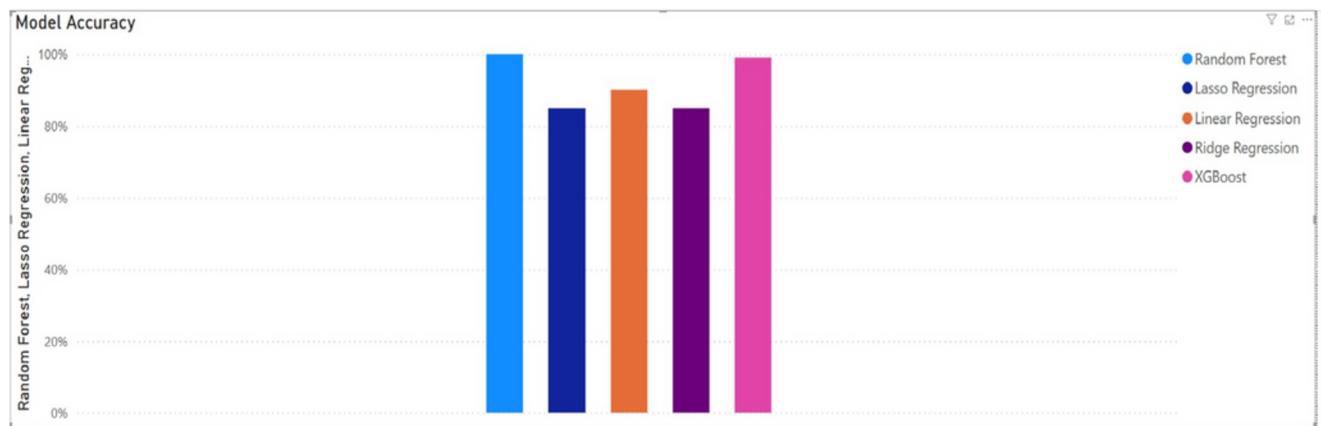
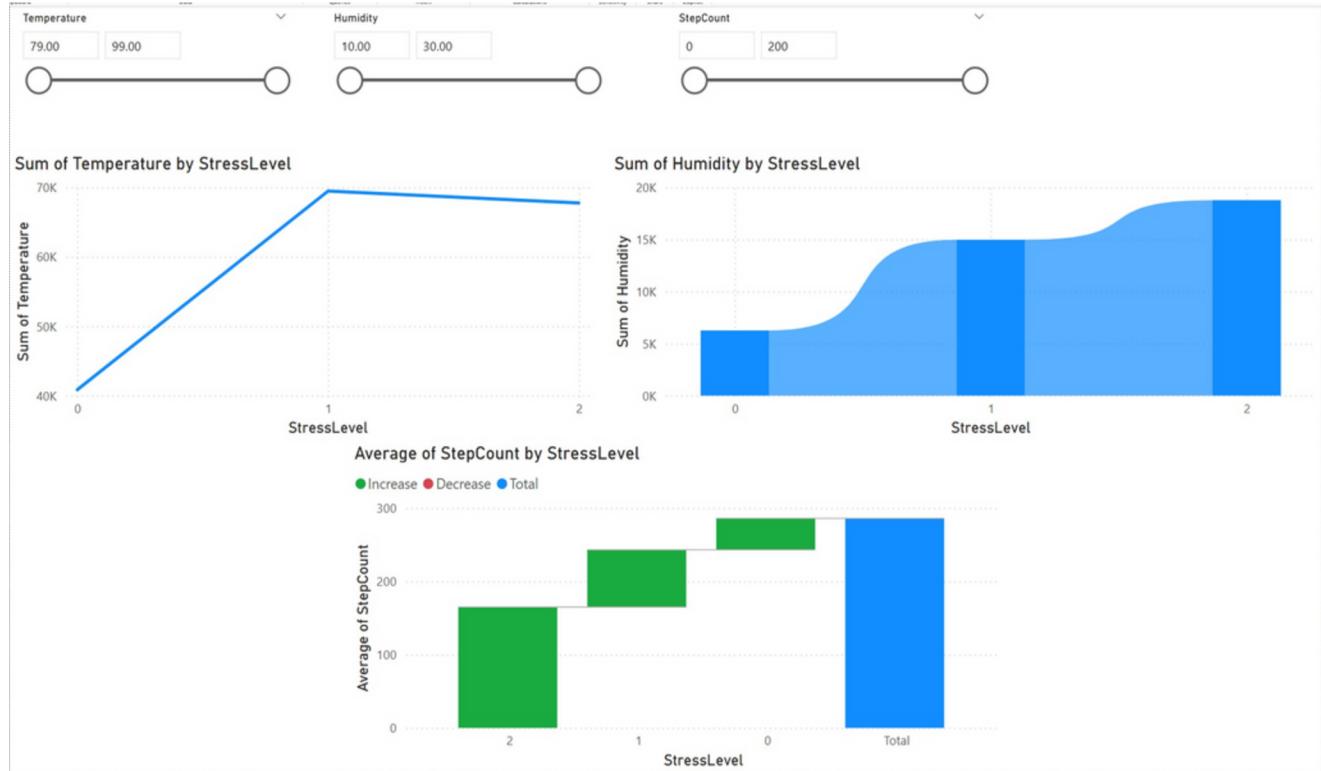
UPDATE p
SET p.Accuracy = CASE
        WHEN p.PredictedStressLevel = 0 AND s.StressLevel = 0 THEN 100
        ELSE CAST((p.PredictedStressLevel * 100.0 / (s.StressLevel + @epsilon)) AS INT)
END
FROM dbo.StressXGBoostPredictions p
JOIN dbo.StressMetrics s
ON p.PersonID = s.PersonID;
...

dropAccuracy_Query = '''
ALTER TABLE dbo.StressXGBoostPredictions
DROP COLUMN Accuracy, StressLevelPredicted
...
# drop columns just to modify values that are <0,01 to not get blanks
# on a accuracy column

conn.execute(addAccuracy_Query)
conn.execute(updateAccuracyValue_Query)
conn.commit()

```

4. Crearea de vizualizări în PowerBI



Rezultate :

Random forest - 99.2%

XGBoost - 99.0%

Regresia Lasso - 85%

Regresia Liniara - 90%

Regresia Ridge - 86%

Diagramă Entitate-Relație

