

**Analiza si Exploatarea Datelor**

**Examen**

**Profesor Maican Catalin**

**Membri:**

**Ivan Cosmin Denis**

**Iov Tudor**

**Olariu Oana**

**Manta Alexia**

**BRAȘOV, 2024**

*Universitatea Transilvania din Brașov*

*Facultatea de Științe Economice și Administrarea Afacerilor*

*Program de studii:Informatica Economica*

**Regression models to predict price/sales**

**Profesor Maican Catalin**

**Membri:**

**Ivan Cosmin Denis**

**Iov Tudor**

**Olariu Oana**

**Manta Alexia**

**BRAȘOV, 2024**

import statsmodels.api as sm

from statsmodels.stats.outliers\_influence import variance\_inflation\_factor

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestRegressor

from sklearn.feature\_selection import RFE

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import r2\_score

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from scipy.stats import pearsonr

#Citim setul de date

df = pd.read\_csv(“laptop\_price.csv", encoding='latin1')

#Aflam structura setului de date si informatii despre acesta

print(df.info())

print(df.describe())

#Stergem valorile nule

df.dropna(inplace=True)

#Verificam daca exista valori duplicate si in cazul in care exista le stergem(in caszul nostu avem 0 valori duplicate)

valori\_duplicate = df.duplicated()

print("Numar de valori duplicate:", valori\_duplicate.sum())

#Numarul total de inregistrari\*\*\*\*\*\*\*\*\*\*\*\*

numar\_inregistrari = df["laptop\_ID"]

print ("Numar inregistrari: ", numar\_inregistrari.count())

#Curatam coloana Product de explicatiile din paranteza

df['Product'] = df['Product'].str.split('(', n=1).str[0]

#Impartim ScreenResolution in doua coloane: PanelType si Resolution

df[['PanelType', 'Resolution']] = df['ScreenResolution'].str.extract(r'([A-Za-z\s/+|4K]+)?(\d+x\d+)?')

#Scoatem ScreenResolution

df = df.drop(["ScreenResolution"], axis=1)

#Inlocuim nan values cu valoarea cea mai frecventa care apare in coloana PanelType(mode)

df['PanelType'] = df['PanelType'].fillna(df['PanelType'].mode()[0])

#Impartim modelul de CPU si GHZ asociati acestuia

df[['Cpu', 'GHz']] = df['Cpu'].str.rsplit(" ", n=1, expand=True)

#Stergem sting-ul "GHz" din coloana GHz

df['GHz'] = df['GHz'].str.replace('GHz', '')

#Transformam in float coloana GHz

df['GHz'] = df['GHz'].astype(float)

#Stergem string-ul GB din coloana Ram

df['RamGB'] = df['Ram'].str.replace('GB', '')

#Transformam in int coloana Ram

df['RamGB'] = df['RamGB'].astype(int)

#Scoatem coloana Ram

df = df.drop(["Ram"], axis=1)

#Impartim producatorul de GPU si modelul asociat acestuia

df[['Gpu', 'Gpu\_Model']] = df['Gpu'].str.split(" ", n=1, expand=True)

#Stergem string-ul kg din coloana Weight

df['WeightKG'] = df['Weight'].str.replace('kg', '')

#Transformam in float coloana Weight

df['WeightKG'] = df['WeightKG'].astype(float)

#Scoatem coloana Weight

df = df.drop(["Weight"], axis=1)

#Impartim Memory in doua coloane: MemoryGB si MemoryBonus

df[['MemoryGB', 'MemoryBonus']] = df['Memory'].str.split("+", n=1, expand=True)

#Scoatem GB

df['MemoryGB'] = df['MemoryGB'].str.replace('GB', '')

#Inlocuim 1TB cu 1000GB

df['MemoryGB'] = df['MemoryGB'].str.replace('1TB', '1000')

#Inlocuim 2TB cu 2000GB

df['MemoryGB'] = df['MemoryGB'].str.replace('2TB', '2000')

#Inlocuim 1.0TB cu 1000GB

df['MemoryGB'] = df['MemoryGB'].str.replace('1.0TB', '1000')

#Scoatem GB

df['MemoryBonus'] = df['MemoryBonus'].str.replace('GB', '')

#Inlocuim 1TB cu 1000GB

df['MemoryBonus'] = df['MemoryBonus'].str.replace('1TB', '1000')

#Inlocuim 2TB cu 2000GB

df['MemoryBonus'] = df['MemoryBonus'].str.replace('2TB', '2000')

#Inlocuim 1.0TB cu 1000GB

df['MemoryBonus'] = df['MemoryBonus'].str.replace('1.0TB', '1000')

#Cautam doar partea numerica din MemoryGB

df['MemoryGB'] = df['MemoryGB'].str.extract('(\d+)')

#Transformam valorile in float

df['MemoryGB'] = pd.to\_numeric(df['MemoryGB'], errors='coerce')

#Cautam doar partea numerica din MemoryBonus

df['MemoryBonus'] = df['MemoryBonus'].str.extract('(\d+)')

#Transformam valorile in float

df['MemoryBonus'] = pd.to\_numeric(df['MemoryBonus'], errors='coerce')

#Înlocuim valorile NaN cu 0\*\*\*\*\*\*\*\*\*\*\*\*

df.fillna(0, inplace=True)

#Calcumam Memoria Totala

df['MemoryTotal'] = df['MemoryGB'] + df['MemoryBonus']

#Selectam string-urile care trebuie pastrate

important\_words = ['SSD', 'HDD',"Hybrid","Flash Storage"]

#Creem o expresie care sa contina toate combinatiile dintre toate cuvintele cheie

pattern = '|'.join(important\_words)

#Extrage substring-ul potivit din coloana Memory

df['Memory'] = df['Memory'].str.extractall(f'({pattern})').groupby(level=0).agg(lambda x: '+'.join(x))

###############################################################################

#medii / cvartile/deviatie standard / mediana pt var. numerice \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#Calculează media, deviația standard, mediana, min, max si cvartilele pentru coloana "laptop\_ID"

df\_laptop\_ID = pd.DataFrame({

'Mean\_laptop\_ID': [df['laptop\_ID'].mean()],

'Std\_laptop\_ID': [df['laptop\_ID'].std()],

'Min\_laptop\_ID': [df['laptop\_ID'].min()],

'Quantile\_25%\_laptop\_ID': [df['laptop\_ID'].quantile(0.25)],

'Quantile\_50%\_laptop\_ID': [df['laptop\_ID'].quantile(0.5)],

'Quantile\_75%\_laptop\_ID': [df['laptop\_ID'].quantile(0.75)],

'Max\_laptop\_ID': [df['laptop\_ID'].max()]

})

# Afișăm rezultatul

print("Laptop\_ID: ", df\_laptop\_ID)

#Calculează media, deviația standard, mediana, min, max si cvartilele pentru coloana "Inches"

df\_Inches = pd.DataFrame({

'Mean\_Inches': [df['Inches'].mean()],

'Std\_Inches': [df['Inches'].std()],

'Min\_Inches': [df['Inches'].min()],

'Quantile\_25%\_Inches': [df['Inches'].quantile(0.25)],

'Quantile\_50%\_Inches': [df['Inches'].quantile(0.5)],

'Quantile\_75%\_Inches': [df['Inches'].quantile(0.75)],

'Max\_Inches': [df['Inches'].max()]

})

# Afișăm rezultatul

print("Inches: ", df\_Inches)

#Calculează media, deviația standard, mediana, min, max si cvartilele pentru coloana "Price\_euros"

df\_Price\_euros = pd.DataFrame({

'Mean\_Price\_euros': [df['Price\_euros'].mean()],

'Std\_Price\_euros': [df['Price\_euros'].std()],

'Min\_Price\_euros': [df['Price\_euros'].min()],

'Quantile\_25%\_Price\_euros': [df['Price\_euros'].quantile(0.25)],

'Quantile\_50%\_Price\_euros': [df['Price\_euros'].quantile(0.5)],

'Quantile\_75%\_Price\_euros': [df['Price\_euros'].quantile(0.75)],

'Max\_Price\_euros': [df['Price\_euros'].max()]

})

# Afișăm rezultatul

print("Price\_euros: ", df\_Price\_euros)

#Calculează media, deviația standard, mediana, min, max si cvartilele pentru coloana "GHz"

df\_GHz = pd.DataFrame({

'Mean\_GHz': [df['GHz'].mean()],

'Std\_GHz': [df['GHz'].std()],

'Min\_GHz': [df['GHz'].min()],

'Quantile\_25%\_GHz': [df['GHz'].quantile(0.25)],

'Quantile\_50%\_GHz': [df['GHz'].quantile(0.5)],

'Quantile\_75%\_GHz': [df['GHz'].quantile(0.75)],

'Max\_GHz': [df['GHz'].max()]

})

# Afișăm rezultatul

print("GHz: ", df\_GHz)

#Calculează media, deviația standard, mediana, min, max si cvartilele pentru coloana "RamGB"

df\_RamGB = pd.DataFrame({

'Mean\_RamGB': [df['RamGB'].mean()],

'Std\_RamGB': [df['RamGB'].std()],

'Min\_RamGB': [df['RamGB'].min()],

'Quantile\_25%\_RamGB': [df['RamGB'].quantile(0.25)],

'Quantile\_50%\_RamGB': [df['RamGB'].quantile(0.5)],

'Quantile\_75%\_RamGB': [df['RamGB'].quantile(0.75)],

'Max\_RamGB': [df['RamGB'].max()]

})

# Afișăm rezultatul

print("RamGB: ", df\_RamGB)

#Calculează media, deviația standard, mediana, min, max si cvartilele pentru coloana "WeightKG"

df\_WeightKG = pd.DataFrame({

'Mean\_WeightKG': [df['WeightKG'].mean()],

'Std\_WeightKG': [df['WeightKG'].std()],

'Min\_WeightKG': [df['WeightKG'].min()],

'Quantile\_25%\_WeightKG': [df['WeightKG'].quantile(0.25)],

'Quantile\_50%\_WeightKG': [df['WeightKG'].quantile(0.5)],

'Quantile\_75%\_WeightKG': [df['WeightKG'].quantile(0.75)],

'Max\_WeightKG': [df['WeightKG'].max()]

})

# Afișăm rezultatul

print("WeightKG: ", df\_WeightKG)

#Calculează media, deviația standard, mediana, min, max si cvartilele pentru coloana "MemoryGB"

df\_MemoryGB = pd.DataFrame({

'Mean\_MemoryGB': [df['MemoryGB'].mean()],

'Std\_MemoryGB': [df['MemoryGB'].std()],

'Min\_MemoryGB': [df['MemoryGB'].min()],

'Quantile\_25%\_MemoryGB': [df['MemoryGB'].quantile(0.25)],

'Quantile\_50%\_MemoryGB': [df['MemoryGB'].quantile(0.5)],

'Quantile\_75%\_MemoryGB': [df['MemoryGB'].quantile(0.75)],

'Max\_MemoryGB': [df['MemoryGB'].max()]

})

# Afișăm rezultatul

print("MemoryGB: ", df\_MemoryGB)

#Calculează media, deviația standard, mediana, min, max si cvartilele pentru coloana "MemoryBonus"

df\_MemoryBonus = pd.DataFrame({

'Mean\_MemoryBonus': [df['MemoryBonus'].mean()],

'Std\_MemoryBonus': [df['MemoryBonus'].std()],

'Min\_MemoryBonus': [df['MemoryBonus'].min()],

'Quantile\_25%\_MemoryBonus': [df['MemoryBonus'].quantile(0.25)],

'Quantile\_50%\_MemoryBonus': [df['MemoryBonus'].quantile(0.5)],

'Quantile\_75%\_MemoryBonus': [df['MemoryBonus'].quantile(0.75)],

'Max\_MemoryBonus': [df['MemoryBonus'].max()]

})

# Afișăm rezultatul

print("MemoryBonus: ", df\_MemoryBonus)

#Calculează media, deviația standard, mediana, min, max si cvartilele pentru coloana "MemoryTotal"

df\_MemoryTotal = pd.DataFrame({

'Mean\_MemoryTotal': [df['MemoryTotal'].mean()],

'Std\_MemoryTotal': [df['MemoryTotal'].std()],

'Min\_MemoryTotal': [df['MemoryTotal'].min()],

'Quantile\_25%\_MemoryTotal': [df['MemoryTotal'].quantile(0.25)],

'Quantile\_50%\_MemoryTotal': [df['MemoryTotal'].quantile(0.5)],

'Quantile\_75%\_MemoryTotal': [df['MemoryTotal'].quantile(0.75)],

'Max\_MemoryTotal': [df['MemoryTotal'].max()]

})

# Afișăm rezultatul

print("MemoryTotal: ", df\_MemoryTotal)

###############################################################################

#Analiza exploratorie

#Cel mai popular panou

aparitie\_panou = df['PanelType'].value\_counts().idxmax()

print("Cel mai folosit Panou este: ", aparitie\_panou)

#Cel mai popular Cpu

aparitie\_Cpu = df['Cpu'].value\_counts().idxmax()

print("Cel mai folosit Cpu este: ", aparitie\_Cpu)

#Cel mai popular producator de Gpu

aparitie\_Gpu = df['Gpu'].value\_counts().idxmax()

print("Cel mai folosit producator de Gpu este: ", aparitie\_Gpu)

#Cel mai popular Gpu

aparitie\_Gpu\_Model = df['Gpu\_Model'].value\_counts().idxmax()

print("Cel mai folosit Gpu este: ", aparitie\_Gpu\_Model)

#Cea mai populara rezolutie

aparitie\_resolution = df['Resolution'].value\_counts().idxmax()

print("Cea mai folosita rezolutie este: ", aparitie\_resolution)

#Cel mai popular produs (cel mai cumparat)

aparitie\_produs = df['Product'].value\_counts().idxmax()

print("Cel mai folosit produs este: ", aparitie\_produs)

#Cea mai populara companie

aparitie\_companie = df['Company'].value\_counts().idxmax()

print("Cea mai populara companie este: ", aparitie\_companie)

#Cel mai popular sistem de operare

aparitie\_opsys = df['OpSys'].value\_counts().idxmax()

print("Cel mai folosit sistem de operare este: ", aparitie\_opsys)

#Cea mai populara diagonala

aparitie\_inci = df['Inches'].value\_counts().idxmax()

print("Cea mai populara diagonala este: ", aparitie\_inci)

#Cel mai nepopular panou

aparitie\_panou\_nepopular = df['PanelType'].value\_counts().idxmin()

print("Cel mai nefolosit Panou este: ", aparitie\_panou\_nepopular)

#Cel mai nepopular Cpu

aparitie\_Cpu\_nepopular = df['Cpu'].value\_counts().idxmin()

print("Cel mai nefolosit Cpu este: ", aparitie\_Cpu\_nepopular)

#Cel mai nepopular producator de Gpu

aparitie\_Gpu\_nepopular = df['Gpu'].value\_counts().idxmin()

print("Cel mai nefolosit producator de Gpu este: ", aparitie\_Gpu\_nepopular)

#Cel mai nepopular Gpu

aparitie\_Gpu\_Model\_nepopular = df['Gpu\_Model'].value\_counts().idxmin()

print("Cel mai nefolosit Gpu este: ", aparitie\_Gpu\_Model\_nepopular)

#Cea mai nepopulara rezolutie

aparitie\_resolution\_nepopulara = df['Resolution'].value\_counts().idxmin()

print("Cea mai nefolosita rezolutie este: ", aparitie\_resolution\_nepopulara)

#Cel mai nepopular produs (cel mai cumparat)

aparitie\_produs\_nepopular = df['Product'].value\_counts().idxmin()

print("Cel mai nefolosit produs este: ", aparitie\_produs\_nepopular)

#Cea mai nepopulara companie

aparitie\_companie\_nepopulara = df['Company'].value\_counts().idxmin()

print("Cea mai nepopulara companie este: ", aparitie\_companie\_nepopulara)

#Cel mai nepopular sistem de operare

aparitie\_opsys\_nepopular = df['OpSys'].value\_counts().idxmin()

print("Cel mai nefolosit sistem de operare este: ", aparitie\_opsys\_nepopular)

#Cea mai nepopulara diagonala

aparitie\_inci\_nepopulara = df['Inches'].value\_counts().idxmin()

print("Cea mai nepopulara populara diagonala este: ", aparitie\_inci\_nepopulara)

#Pret laptop pe fiecare companie + Grafic

pret\_companie = df.groupby('Company')['Price\_euros'].mean()

print('Media de pret pentru fiecare companie: ',pret\_companie)

# Crearea unui grafic bar

plt.figure(figsize=(12, 6))

pret\_companie.plot(kind='bar', color='skyblue')

plt.title('Pretul mediu in functie de companie')

plt.xlabel('Companie')

plt.ylabel('Pret mediu (euro)')

plt.xticks(rotation=45, ha='right') # Rotirea etichetelor pentru a le face mai usor de citit

plt.show()

#Pret maxim laptop pe fiecare companie+ Grafic

pret\_companie\_max = df.groupby('Company')['Price\_euros'].max()

print('Pret maxim laptop pe fiecare companie: ',pret\_companie\_max)

# Crearea unui grafic bar

plt.figure(figsize=(12, 6))

pret\_companie\_max.plot(kind='bar', color='green')

plt.title('Pretul maxim in functie de companie')

plt.xlabel('Companie')

plt.ylabel('Pret maxim (euro)')

plt.xticks(rotation=45, ha='right')

plt.show()

#Pret minim laptop pe fiecare companie+ Grafic

pret\_companie\_min = df.groupby('Company')['Price\_euros'].min()

print('Pret minim laptop pe fiecare comapanie: ',pret\_companie\_min)

# Crearea unui grafic bar

plt.figure(figsize=(12, 6))

pret\_companie\_min.plot(kind='bar', color='red')

plt.title('Pretul minim in functie de companie')

plt.xlabel('Companie')

plt.ylabel('Pret minim (euro)')

plt.xticks(rotation=45, ha='right')

plt.show()

#Pret laptop pe fiecare tip de laptop + Grafic

pret\_type = df.groupby('TypeName')['Price\_euros'].mean()

print('Pret laptop pe fiecare tip de laptop:',pret\_type)

# Crearea unui grafic bar

plt.figure(figsize=(12, 6))

pret\_type.plot(kind='bar', color='orange')

plt.title('Pretul in functie de tipul laptopului')

plt.xlabel('Tipul laptopului')

plt.ylabel('Pret (euro)')

plt.xticks(rotation=45, ha='right')

plt.show()

#Pret minim laptop pe fiecare tip de laptop + Grafic

pret\_type\_min = df.groupby('TypeName')['Price\_euros'].min()

print('Pret minim laptop pe fiecare tip de laptop:',pret\_type\_min)

# Crearea unui grafic bar

plt.figure(figsize=(12, 6))

pret\_type\_min.plot(kind='bar', color='brown')

plt.title('Pretul minim in functie de tipul laptopului')

plt.xlabel('Tipul laptopului')

plt.ylabel('Pret minim (euro)')

plt.xticks(rotation=45, ha='right')

plt.show()

#Pret maxim laptop pe fiecare tip de laptop + Grafic

pret\_type\_max = df.groupby('TypeName')['Price\_euros'].max()

print('Pret maxim laptop pe fiecare tip de laptop:',pret\_type\_max)

# Crearea unui grafic bar

plt.figure(figsize=(12, 6))

pret\_type\_max.plot(kind='bar', color='darkorange')

plt.title('Pretul maxim in functie de tipul laptopului')

plt.xlabel('Tipul laptopului')

plt.ylabel('Pret maxim (euro)')

plt.xticks(rotation=45, ha='right')

plt.show()

#Pret laptop pe fiecare tip de panou + Grafic

pret\_panel = df.groupby('PanelType')['Price\_euros'].mean()

print('Pret laptop pe fiecare tip de panou: ',pret\_panel)

# Crearea unui grafic bar

plt.figure(figsize=(12, 6))

pret\_panel.plot(kind='bar', color='yellow')

plt.title('Pretul in functie de tipul panoului')

plt.xlabel('Tipul panoului')

plt.ylabel('Pret (euro)')

plt.xticks(rotation=45, ha='right')

plt.show()

#Pret minim laptop pe fiecare tip de panou + Grafic

pret\_panel\_min = df.groupby('PanelType')['Price\_euros'].min()

print('Pret minim laptop pe fiecare tip de panou: ',pret\_panel\_min)

# Crearea unui grafic bar

plt.figure(figsize=(12, 6))

pret\_panel\_min.plot(kind='bar', color='skyblue')

plt.title('Pretul minim in functie de tipul panoului')

plt.xlabel('Tipul panoului')

plt.ylabel('Pret minim (euro)')

plt.xticks(rotation=45, ha='right')

plt.show()

#Pret maxim laptop pe fiecare tip de panou + Grafic

pret\_panel\_max = df.groupby('PanelType')['Price\_euros'].max()

print('Pret maxim laptop pe fiecare tip de panou: ',pret\_panel\_max)

# Crearea unui grafic bar

plt.figure(figsize=(12, 6))

pret\_panel\_max.plot(kind='bar', color='red')

plt.title('Pretul maxim in functie de tipul panoului')

plt.xlabel('Tipul panoului')

plt.ylabel('Pret maxim (euro)')

plt.xticks(rotation=45, ha='right')

plt.show()

#Pret laptop pe fiecare tip de rezolutie + Grafic

pret\_resolution = df.groupby('Resolution')['Price\_euros'].mean()

print('Pret laptop pe fiecare tip de rezolutie: ',pret\_resolution)

# Crearea unui grafic bar

plt.figure(figsize=(12, 6))

pret\_resolution.plot(kind='bar', color='black')

plt.title('Pretul in functie de tipul rezolutiei')

plt.xlabel('Tipul rezolutiei')

plt.ylabel('Pret (euro)')

plt.xticks(rotation=45, ha='right')

plt.show()

#Pret minim laptop pe fiecare tip de rezolutie + Grafic

pret\_resolution\_min = df.groupby('Resolution')['Price\_euros'].min()

print('Pret minim laptop pe fiecare tip de rezolutie: ',pret\_resolution\_min)

# Crearea unui grafic bar

plt.figure(figsize=(12, 6))

pret\_resolution\_min.plot(kind='bar', color='red')

plt.title('Pretul minim in functie de tipul rezolutiei')

plt.xlabel('Tipul rezolutiei')

plt.ylabel('Pret minim (euro)')

plt.xticks(rotation=45, ha='right')

plt.show()

#Pret maxim laptop pe fiecare tip de rezolutie + Grafic

pret\_resolution\_max = df.groupby('Resolution')['Price\_euros'].max()

print('Pret maxim laptop pe fiecare tip de rezolutie: ',pret\_resolution\_max)

# Crearea unui grafic bar

plt.figure(figsize=(12, 6))

pret\_resolution\_max.plot(kind='bar', color='grey')

plt.title('Pretul maxim in functie de tipul rezolutiei')

plt.xlabel('Tipul rezolutiei')

plt.ylabel('Pret maxim (euro)')

plt.xticks(rotation=45, ha='right')

plt.show()

#Pret laptop pe fiecare tip de Cpu + Grafic

pret\_Cpu = df.groupby('Cpu')['Price\_euros'].mean()

print('Pret laptop pe fiecare tip de Cpu: ',pret\_Cpu)

# Crearea unui grafic bar

plt.figure(figsize=(12, 6))

pret\_Cpu.plot(kind='bar', color='lightblue')

plt.title('Pretul in functie de tipul Cpu')

plt.xlabel('Tipul Cpu')

plt.ylabel('Pret (euro)')

plt.xticks(rotation=45, ha='right')

plt.show()

#Pret minim laptop pe fiecare tip de Cpu + Grafic

pret\_Cpu\_min = df.groupby('Cpu')['Price\_euros'].min()

print('Pret minim laptop pe fiecare tip de Cpu: ',pret\_Cpu\_min)

# Crearea unui grafic bar

plt.figure(figsize=(12, 6))

pret\_Cpu\_min.plot(kind='bar', color='darkblue')

plt.title('Pretul minim in functie de tipul Cpu')

plt.xlabel('Tipul Cpu')

plt.ylabel('Pret minim (euro)')

plt.xticks(rotation=45, ha='right')

plt.show()

#Pret maxim laptop pe fiecare tip de Cpu + Grafic

pret\_Cpu\_max = df.groupby('Cpu')['Price\_euros'].max()

print('Pret maxim laptop pe fiecare tip de Cpu: ',pret\_Cpu\_max)

# Crearea unui grafic bar

plt.figure(figsize=(12, 6))

pret\_Cpu\_max.plot(kind='bar', color='blue')

plt.title('Pretul maxim in functie de tipul Cpu')

plt.xlabel('Tipul Cpu')

plt.ylabel('Pret maxim (euro)')

plt.xticks(rotation=45, ha='right')

plt.show()

#Pret laptop pe fiecare tip de Gpu + Grafic

pret\_Gpu = df.groupby('Gpu')['Price\_euros'].mean()

print('Pret laptop pe fiecare tip de Gpu:',pret\_Gpu)

# Crearea unui grafic bar

plt.figure(figsize=(12, 6))

pret\_Gpu.plot(kind='bar', color='darkred')

plt.title('Pretul in functie de tipul Gpu')

plt.xlabel('Tipul Gpu')

plt.ylabel('Pret (euro)')

plt.xticks(rotation=45, ha='right')

plt.show()

#Pret minim laptop pe fiecare tip de Gpu + Grafic

pret\_Gpu\_min = df.groupby('Gpu')['Price\_euros'].min()

print('Pret minim laptop pe fiecare tip de Gpu:',pret\_Gpu\_min)

# Crearea unui grafic bar

plt.figure(figsize=(12, 6))

pret\_Gpu\_min.plot(kind='bar', color='pink')

plt.title('Pretul minim in functie de tipul Gpu')

plt.xlabel('Tipul Gpu')

plt.ylabel('Pret minim (euro)')

plt.xticks(rotation=45, ha='right')

plt.show()

#Pret maxim laptop pe fiecare tip de Gpu + Grafic

pret\_Gpu\_max = df.groupby('Gpu')['Price\_euros'].max()

print('Pret maxim laptop pe fiecare tip de Gpu:',pret\_Gpu\_max)

# Crearea unui grafic bar

plt.figure(figsize=(12, 6))

pret\_Gpu\_max.plot(kind='bar', color='red')

plt.title('Pretul maxim in functie de tipul Gpu')

plt.xlabel('Tipul Gpu')

plt.ylabel('Pret maxim (euro)')

plt.xticks(rotation=45, ha='right')

plt.show()

#Pret laptop pe fiecare tip de Ram + Grafic

pret\_Ram = df.groupby('RamGB')['Price\_euros'].mean()

print('Pret laptop pe fiecare tip de Ram: ',pret\_Ram)

# Crearea unui grafic bar

plt.figure(figsize=(12, 6))

pret\_Ram.plot(kind='bar', color='pink')

plt.title('Pretul in functie de tipul Ram')

plt.xlabel('Tipul Ram')

plt.ylabel('Pret (euro)')

plt.xticks(rotation=45, ha='right')

plt.show()

#Pret minim laptop pe fiecare tip de Ram + Grafic

pret\_Ram\_min = df.groupby('RamGB')['Price\_euros'].min()

print('Pret minim laptop pe fiecare tip de Ram: ',pret\_Ram\_min)

# Crearea unui grafic bar

plt.figure(figsize=(12, 6))

pret\_Ram\_min.plot(kind='bar', color='brown')

plt.title('Pretul minim in functie de tipul Ram')

plt.xlabel('Tipul Ram')

plt.ylabel('Pret minim (euro)')

plt.xticks(rotation=45, ha='right')

plt.show()

#Pret maxim laptop pe fiecare tip de Ram + Grafic

pret\_Ram\_max = df.groupby('RamGB')['Price\_euros'].max()

print('Pret maxim laptop pe fiecare tip de Ram: ',pret\_Ram\_max)

# Crearea unui grafic bar

plt.figure(figsize=(12, 6))

pret\_Ram\_max.plot(kind='bar', color='orange')

plt.title('Pretul maxim in functie de tipul Ram')

plt.xlabel('Tipul Ram')

plt.ylabel('Pret maxim (euro)')

plt.xticks(rotation=45, ha='right')

plt.show()

#Pret laptop pe fiecare in functie de Memorie + Grafic

pret\_memorytotal = df.groupby('MemoryTotal')['Price\_euros'].mean()

print('Pret laptop pe fiecare in functie de Memorie: ',pret\_memorytotal)

# Crearea unui grafic bar

plt.figure(figsize=(12, 6))

pret\_memorytotal.plot(kind='bar', color='salmon')

plt.title('Pretul in functie de memorie')

plt.xlabel('Memorie')

plt.ylabel('Pret (euro)')

plt.xticks(rotation=45, ha='right')

plt.show()

#Pret minim laptop pe fiecare in functie de Memorie + Grafic

pret\_memorytotal\_min = df.groupby('MemoryTotal')['Price\_euros'].min()

print('Pret minim laptop pe fiecare in functie de Memorie: ',pret\_memorytotal\_min)

# Crearea unui grafic bar

plt.figure(figsize=(12, 6))

pret\_memorytotal\_min.plot(kind='bar', color='purple')

plt.title('Pretul minim in functie de memorie')

plt.xlabel('Memorie')

plt.ylabel('Pret minim (euro)')

plt.xticks(rotation=45, ha='right')

plt.show()

#Pret maxim laptop pe fiecare in functie de Memorie + Grafic

pret\_memorytotal\_max = df.groupby('MemoryTotal')['Price\_euros'].max()

print('Pret maxim laptop pe fiecare in functie de Memorie: ',pret\_memorytotal\_max)

# Crearea unui grafic bar

plt.figure(figsize=(12, 6))

pret\_memorytotal\_max.plot(kind='bar', color='black')

plt.title('Pretul maxim in functie de memorie')

plt.xlabel('Memorie')

plt.ylabel('Pret maxim (euro)')

plt.xticks(rotation=45, ha='right')

plt.show()

#Pret laptop pt fiecare tip de memorie

pret\_memory = df.groupby('Memory')['Price\_euros'].mean()

print("Pret laptop pt fiecare tip de memorie:", pret\_memory)

# Crearea unui grafic bar

plt.figure(figsize=(12, 6))

pret\_memory.plot(kind='bar', color='grey')

plt.title('Pretul in functie de tipul de memorie')

plt.xlabel('Memorie')

plt.ylabel('Pret (euro)')

plt.xticks(rotation=45, ha='right')

plt.show()

#Pret minim laptop pt fiecare tip de memorie

pret\_memory\_min = df.groupby('Memory')['Price\_euros'].min()

print("Pret minim laptop pt fiecare tip de memorie:", pret\_memory\_min)

# Crearea unui grafic bar

plt.figure(figsize=(12, 6))

pret\_memory\_min.plot(kind='bar', color='brown')

plt.title('Pretul minim in functie de tipul de memorie')

plt.xlabel('Memorie')

plt.ylabel('Pret minim (euro)')

plt.xticks(rotation=45, ha='right')

plt.show()

#Pret maxim laptop pt fiecare tip de memorie

pret\_memory\_max = df.groupby('Memory')['Price\_euros'].max()

print("Pret laptop pt fiecare tip de memorie:", pret\_memory\_max)

# Crearea unui grafic bar

plt.figure(figsize=(12, 6))

pret\_memory\_max.plot(kind='bar', color='purple')

plt.title('Pretul maxim in functie de tipul de memorie')

plt.xlabel('Memorie')

plt.ylabel('Pret maxim (euro)')

plt.xticks(rotation=45, ha='right')

plt.show()

#Pret laptop pe fiecare in functie de Operating Sistem + Grafic

pret\_opsys = df.groupby('OpSys')['Price\_euros'].mean()

print("Pret laptop pe fiecare in functie de Operating Sistem:", pret\_opsys)

# Crearea unui grafic bar

plt.figure(figsize=(12, 6))

pret\_opsys.plot(kind='bar', color='red')

plt.title('Pretul in functie de sistemul de operare')

plt.xlabel('Sistem Operare')

plt.ylabel('Pret (euro)')

plt.xticks(rotation=45, ha='right')

plt.show()

#Pret minim laptop pe fiecare in functie de Operating Sistem + Grafic

pret\_opsys\_min = df.groupby('OpSys')['Price\_euros'].min()

print("Pret minim laptop pe fiecare in functie de Operating Sistem:", pret\_opsys\_min)

# Crearea unui grafic bar

plt.figure(figsize=(12, 6))

pret\_opsys\_min.plot(kind='bar', color='grey')

plt.title('Pretul minim in functie de sistemul de operare')

plt.xlabel('Sistem Operare')

plt.ylabel('Pret minim (euro)')

plt.xticks(rotation=45, ha='right')

plt.show()

#Pret maxim laptop pe fiecare in functie de Operating Sistem + Grafic

pret\_opsys\_max = df.groupby('OpSys')['Price\_euros'].max()

print("Pret maxim laptop pe fiecare in functie de Operating Sistem:", pret\_opsys\_max)

# Crearea unui grafic bar

plt.figure(figsize=(12, 6))

pret\_opsys\_max.plot(kind='bar', color='red')

plt.title('Pretul maxim in functie de sistemul de operare')

plt.xlabel('Sistem Operare')

plt.ylabel('Pret maxim (euro)')

plt.xticks(rotation=45, ha='right')

plt.show()

#Pret laptop pe fiecare in functie de greutate + Grafic

pret\_weight = df.groupby('WeightKG')['Price\_euros'].mean()

print("Pret laptop pe fiecare in functie de greutate:", pret\_weight)

# Crearea unui grafic bar

plt.figure(figsize=(12, 6))

plt.scatter(pret\_weight.index, pret\_weight.values, color='purple')

plt.title('Pretul in functie de greutate')

plt.xlabel('Greutate')

plt.ylabel('Pret (euro)')

plt.xticks(rotation=45, ha='right')

plt.show()

#Pret minim laptop pe fiecare in functie de greutate + Grafic

pret\_weight\_min = df.groupby('WeightKG')['Price\_euros'].min()

print("Pret minim laptop pe fiecare in functie de greutate:", pret\_weight\_min)

# Crearea unui grafic bar

plt.figure(figsize=(12, 6))

plt.scatter(pret\_weight\_min.index, pret\_weight\_min.values, color='brown')

plt.title('Pretul minim in functie de greutate')

plt.xlabel('Greutate')

plt.ylabel('Pret minim (euro)')

plt.xticks(rotation=45, ha='right')

plt.show()

#Pret maxim laptop pe fiecare in functie de greutate + Grafic

pret\_weight\_max = df.groupby('WeightKG')['Price\_euros'].max()

print("Pret maxim laptop pe fiecare in functie de greutate:", pret\_weight\_max)

# Crearea unui grafic bar

plt.figure(figsize=(12, 6))

plt.scatter(pret\_weight\_max.index, pret\_weight\_max.values, color='black')

plt.title('Pretul maxim in functie de greutate')

plt.xlabel('Greutate')

plt.ylabel('Pret maxim (euro)')

plt.xticks(rotation=45, ha='right')

plt.show()

#Cate laptopuri pt fiecare companie

laptop\_model = df.groupby('Company')['Product'].count()

print("Cate laptopuri pt fiecare companie:", laptop\_model)

# Crearea unui grafic bar

plt.figure(figsize=(12, 6))

laptop\_model.plot(kind='bar', color='orange')

plt.title('Laptopuri pentru fiecare companie')

plt.xlabel('Companie')

plt.ylabel('Numar')

plt.xticks(rotation=45, ha='right')

plt.show()

#Cel mai cumparat produs pentru fiecare companie

cumparat = df.groupby(['Company', 'Product']).size().reset\_index(name='Number')

print("Cel mai cumparat produs pentru fiecare companie: ", cumparat)

#Crearea unui grafic bar

plt.figure(figsize=(12, 6))

cumparat.plot(kind='bar', color='purple')

plt.title('Cel mai cumparat produs')

plt.xlabel('Produs')

plt.ylabel('Numar')

plt.xticks(rotation=45, ha='right')

plt.show()

#Sistem operare in functie de Ram + Grafic

op\_ram = df.groupby('OpSys')['RamGB'].mean().round()

print("Cat ram e nevoie pentru fiecare sistem de operare: ", op\_ram)

#Crearea unui grafic bar

plt.figure(figsize=(12, 6))

op\_ram.plot(kind='bar', color='skyblue')

plt.title('Ram pentru fiecare sistem de operare')

plt.xlabel('Sistem Operare')

plt.ylabel('Ram')

plt.xticks(rotation=45, ha='right')

plt.show()

# Afișarea celor mai scumpe 5 laptopuri

laptop5 = df.nlargest(5, 'Price\_euros')[['Product', 'Price\_euros']]

print("Top 5 cele mai scumpe laptopuri:", laptop5)

# Crearea unui grafic bar pentru cele mai scumpe laptopuri

plt.figure(figsize=(12, 6))

plt.scatter(laptop5['Product'], laptop5['Price\_euros'], color='salmon')

plt.title('Cele mai scumpe 5 laptopuri')

plt.xlabel('Modelul laptopului')

plt.ylabel('Pret (euro)')

plt.xticks(rotation=45, ha='right')

plt.show()

# Afișarea celor mai ieftine 5 laptopuri

laptop5s = df.nsmallest(5, 'Price\_euros')[['Product', 'Price\_euros']]

print("Top 5 cele mai ieftine laptopuri:", laptop5s)

# Crearea unui grafic bar pentru cele mai scumpe laptopuri

plt.figure(figsize=(12, 6))

plt.scatter(laptop5s['Product'], laptop5s['Price\_euros'], color='purple')

plt.title('Cele mai ieftine 5 laptopuri')

plt.xlabel('Modelul laptopului')

plt.ylabel('Pret (euro)')

plt.xticks(rotation=45, ha='right')

plt.show()

# Afișarea top 5 laptopuri cu cea mai multa memorie

laptop5m = df.nlargest(5, 'MemoryTotal')[['Product', 'MemoryTotal']]

print("Top 5 laptopuri cu cea mai multa memorie:", laptop5m)

# Crearea unui grafic

plt.figure(figsize=(12, 6))

plt.scatter(laptop5m['Product'], laptop5m['MemoryTotal'], color='green')

plt.title('5 laptopuri cea mai multa memorie')

plt.xlabel('Modelul laptopului')

plt.ylabel('Memorie')

plt.xticks(rotation=45, ha='right')

plt.show()

# Afișarea top 5 laptopuri cu cea mai putina memorie

laptop5ms = df.nsmallest(5, 'MemoryTotal')[['Product', 'MemoryTotal']]

print("Top 5 laptopuri cu cea mai putina memorie:", laptop5ms)

# Crearea unui grafic

plt.figure(figsize=(12, 6))

plt.scatter(laptop5ms['Product'], laptop5ms['MemoryTotal'], color='orange')

plt.title('5 laptopuri cea mai putina memorie')

plt.xlabel('Modelul laptopului')

plt.ylabel('Memorie')

plt.xticks(rotation=45, ha='right')

plt.show()

# Afișarea top 5 laptopuri cu cel mai mult RAM

laptop5r = df.nlargest(5, 'RamGB')[['Product', 'RamGB']]

print("Top 5 laptopuri cu cel mai mult Ram:", laptop5r)

# Crearea unui grafic

plt.figure(figsize=(12, 6))

plt.scatter(laptop5r['Product'], laptop5r['RamGB'], color='skyblue')

plt.title('5 laptopuri cel mai mult RAM')

plt.xlabel('Modelul laptopului')

plt.ylabel('Ram')

plt.xticks(rotation=45, ha='right')

plt.show()

# Afișarea top 5 laptopuri cu cel mia putin RAM

laptop5rs = df.nsmallest(5, 'RamGB')[['Product', 'RamGB']]

print("Top 5 laptopuri cu cel mai putin Ram:", laptop5rs)

# Crearea unui grafic

plt.figure(figsize=(12, 6))

plt.scatter(laptop5rs['Product'], laptop5rs['RamGB'], color='purple')

plt.title('5 laptopuri cel mai putina RAM')

plt.xlabel('Modelul laptopului')

plt.ylabel('Ram')

plt.xticks(rotation=45, ha='right')

plt.show()

# Afișarea top 3 cele mai grele laptopuri

laptop3w = df.nlargest(3, 'WeightKG')[['Product', 'WeightKG']]

print("Top 3 cele mai grele laptopuri:", laptop3w)

# Crearea unui grafic

plt.figure(figsize=(12, 6))

plt.scatter(laptop3w['Product'], laptop3w['WeightKG'], color='red')

plt.title('3 laptopuri cele mai grele laptopuri')

plt.xlabel('Modelul laptopului')

plt.ylabel('Greutate')

plt.xticks(rotation=45, ha='right')

plt.show()

# Afișarea top 3 laptopuri cu cea mai mare diagonala

laptop5d = df.nlargest(3, 'Inches')[['Product', 'Inches']]

print("Top 5 laptopuri cu cea mai mare diagonala:", laptop5d)

# Crearea unui grafic

plt.figure(figsize=(12, 6))

plt.scatter(laptop5d['Product'], laptop5d['Inches'], color='purple')

plt.title('5 laptopuri cel mai mare diagonala')

plt.xlabel('Modelul laptopului')

plt.ylabel('Diagonala')

plt.xticks(rotation=45, ha='right')

plt.show()

###############################################################################

#Determinam coloanele categoriale

coloane\_categoriale = df.select\_dtypes(include=["object"]).columns.drop(['Company', 'Product', 'Cpu', 'Gpu', 'Memory', 'PanelType', 'Resolution', 'Gpu\_Model'])

#Transformam coloanele folosind One Hot Encoding

df\_encoded = pd.get\_dummies(df, columns=coloane\_categoriale, drop\_first=True, dtype=int)

################################################################################ Pana aici am curatat df-ul\*

# Selectează caracteristicile pentru regresie

features = df\_encoded.drop(['Company', 'Product', 'Cpu', 'Gpu', 'Memory', 'PanelType', 'Resolution', 'Gpu\_Model','Price\_euros'], axis=1)

# Selectează variabila țintă

target = df\_encoded['Price\_euros']

# Împarte datele în seturi de antrenare și de testare

x\_train, x\_test, y\_train, y\_test = train\_test\_split(features, target, test\_size=0.2, random\_state=0)

###############################################################################

#Corelatie df

correlation\_matrix = df.corr(numeric\_only=True)

plt.figure(figsize=(10, 8))

sns.heatmap(correlation\_matrix, annot=True, cmap='coolwarm', fmt=".2f")

plt.title('Correlation Matrix')

plt.show()

#Corelatie df\_encoded

correlation\_matrix\_encoded = df\_encoded.corr(numeric\_only=True)

plt.figure(figsize=(10, 8))

sns.heatmap(correlation\_matrix\_encoded, annot=True, cmap='coolwarm', fmt=".2f")

plt.title('Correlation Matrix')

plt.show()

###############################################################################

#Coeficient Pearson

# Lista pentru a stoca rezultatele

correlation\_results = []

# Calculați și adăugați coeficientul de corelație Pearson în listă

for feature in features.columns:

correlation, \_ = pearsonr(features[feature], target)

correlation\_results.append({'Feature': feature, 'Pearson Correlation': correlation})

# Creare DataFrame din lista de rezultate

correlation\_df = pd.DataFrame(correlation\_results)

# Afișare DataFrame cu rezultatele

print(correlation\_df)

###############################################################################

#Regresie Liniara

model = LinearRegression()

model.fit(x\_train, y\_train)

y\_prezis = model.predict(x\_test)

r\_patrat = model.score(x\_train, y\_train)

scorul\_r2 = r2\_score(y\_test, y\_prezis)

print(f'Coeficientul de determinare (R^2): {scorul\_r2}')

#Valorile prezise vs. cele reale

plt.figure(figsize=(10, 6))

sns.scatterplot(x=y\_test, y=y\_prezis)

plt.xlabel('Preturi Reale')

plt.ylabel('Preturi Prezise')

plt.title('Preturi Reale vs. Preturi Prezise')

plt.show()

###############################################################################

#Regresie OLS

#Adauga o coloană constanta pentru termenul liber (intercept)

x\_train = sm.add\_constant(x\_train)

x\_test = sm.add\_constant(x\_test)

#Initializeaza si antreneaza modelul de regresie liniara folosind OLS

ols\_model = sm.OLS(y\_train, x\_train).fit()

#Realizeaza predictii pe setul de testare

y\_pred2 = ols\_model.predict(x\_test)

#Evaluează performanta modelului

r2rOLS = r2\_score(y\_test, y\_pred2)

print(ols\_model.summary())

print(f'Coeficientul de determinare (R^2): {r2rOLS}')

# Vizualizează valorile prezise vs. cele reale

plt.figure(figsize=(10, 6))

sns.scatterplot(x=y\_test, y=y\_pred2)

plt.xlabel('Prețuri Reale')

plt.ylabel('Prețuri Prezise')

plt.title('Prețuri Reale vs. Prețuri Prezise (Regresie OLS)')

plt.show()

###############################################################################

#Regresie Random Forest

# Inițializați și antrenați Random Forest Regressor

rf\_model\_laptop = RandomForestRegressor(n\_estimators=100, random\_state=0)

rf\_model\_laptop.fit(x\_train, y\_train)

# Make predictions on the test set

y\_pred = rf\_model\_laptop.predict(x\_test)

#Evaluam performanta folosind metrici precum coeficientul R^2

r2 = r2\_score(y\_test, y\_pred)

print(f'Coeficientul de determinare R^2: {r2}')

# Creează un grafic scatter pentru regresia Random Forest

plt.figure(figsize=(10, 6))

plt.scatter(y\_test, y\_pred, alpha=0.5)

plt.title('Regresie Random Forest: Valori Reale vs. Prezise')

plt.xlabel('Valori Reale')

plt.ylabel('Valori Prezise')

plt.show()

###############################################################################

#Ca la curs pt multicoliniaritate

#Importanta

feature\_importance = rf\_model\_laptop.feature\_importances\_

coeficienti = ols\_model.params

p\_values = ols\_model.pvalues

rez\_regresie = pd.DataFrame({

"Coeficient": coeficienti,

"p-values": p\_values

})

r\_squared = ols\_model.rsquared

###############################################################################

#VIF - pt multicoliniaritate

x\_with\_const = sm.add\_constant(features)

vif\_data = pd.DataFrame()

vif\_data["Variable"] = x\_with\_const.columns

vif\_data["VIF"] = [variance\_inflation\_factor(x\_with\_const.values, i) for i in range(x\_with\_const.shape[1])]

print(vif\_data)

###############################################################################

#Optimizare set de date dupa rulare VIF

features2 = df\_encoded.drop(['Company', 'Product', 'Cpu', 'Gpu', 'Memory', 'PanelType', 'Resolution', 'Gpu\_Model','Price\_euros','MemoryGB','MemoryBonus','OpSys\_Linux','OpSys\_Chrome OS','OpSys\_No OS','OpSys\_Windows 10','OpSys\_Windows 7'], axis=1)

#VIF - pt multicoliniaritate

x\_with\_const = sm.add\_constant(features2)

vif\_data2 = pd.DataFrame()

vif\_data2["Variable"] = x\_with\_const.columns

vif\_data2["VIF"] = [variance\_inflation\_factor(x\_with\_const.values, i) for i in range(x\_with\_const.shape[1])]

print(vif\_data2)

#RFE

rfe = RFE(estimator = model, n\_features\_to\_select=10)

rf = rfe.fit(features2, target)

atribute\_selectate = pd.DataFrame({

"atribut": features2.columns,

"selectate": rfe.support\_,

"ranking": rfe.ranking\_

})

###############################################################################

#Regresie pe modelul optimizat

atribute\_selectate\_pt\_reg = atribute\_selectate[atribute\_selectate["selectate"]]["atribut"]

features\_selected= features2[atribute\_selectate\_pt\_reg]

x\_tr\_sel, x\_test\_sel, y\_tr\_sel, y\_test\_sel = train\_test\_split(features\_selected, target, test\_size = 0.2, random\_state=0)

reg\_lin\_sel = LinearRegression()

reg\_lin\_sel.fit(x\_tr\_sel, y\_tr\_sel)

y\_prezis\_sel = reg\_lin\_sel.predict(x\_test\_sel)

r2\_scor\_selectat = r2\_score(y\_test\_sel, y\_prezis\_sel)

print(r2\_scor\_selectat)

#Valorile prezise vs. cele reale

plt.figure(figsize=(10, 6))

sns.scatterplot(x=y\_test\_sel, y=y\_prezis\_sel)

plt.xlabel('Preturi Reale')

plt.ylabel('Preturi Prezise')

plt.title('Preturi Reale vs. Preturi Prezise Selectate')

plt.show()

###############################################################################

#Regresie OLS selectat

#Adauga o coloană constanta pentru termenul liber (intercept) selectat

x\_tr\_sel = sm.add\_constant(x\_tr\_sel)

x\_test\_sel = sm.add\_constant(x\_test\_sel)

#Initializeaza si antreneaza modelul de regresie liniara folosind OLS selectat

ols\_model\_selectat = sm.OLS(y\_tr\_sel, x\_tr\_sel).fit()

#Realizeaza predictii pe setul de testare selectat

y\_pred2\_selectat = ols\_model\_selectat.predict(x\_test\_sel)

#Evaluează performanta modelului selectat

r2rOLS\_selectat = r2\_score(y\_test\_sel, y\_pred2)

print(ols\_model\_selectat.summary())

print(f'Coeficientul de determinare (R^2): {r2rOLS}')

# Vizualizează valorile prezise vs. cele reale

plt.figure(figsize=(10, 6))

sns.scatterplot(x=y\_test\_sel, y=y\_pred2\_selectat)

plt.xlabel('Prețuri Reale')

plt.ylabel('Prețuri Prezise')

plt.title('Prețuri Reale vs. Prețuri Prezise (Regresie OLS) Selectate')

plt.show()

###############################################################################

#Regresie Random Forest

# Inițializați și antrenați Random Forest Regressor selectat

rf\_model\_laptop\_select = RandomForestRegressor(n\_estimators=100, random\_state=0)

rf\_model\_laptop\_select.fit(x\_tr\_sel, y\_tr\_sel)

# Make predictions on the test set selected

y\_pred\_select = rf\_model\_laptop\_select.predict(x\_test\_sel)

#Evaluam performanta folosind metrici precum coeficientul R^2 selectat

r2\_select = r2\_score(y\_test\_sel, y\_pred\_select)

print(f'Coeficientul de determinare R^2: {r2\_select}')

# Creează un grafic scatter pentru regresia Random Forest

plt.figure(figsize=(10, 6))

plt.scatter(y\_test\_sel, y\_pred\_select, alpha=0.5)

plt.title('Regresie Random Forest: Valori Reale vs. Prezise Selectate')

plt.xlabel('Valori Reale')

plt.ylabel('Valori Prezise')

plt.show()