Tema 1

Analiza exploratorie a datelor pentru oferte de vânzări auto
Mineritul datelor și analiza datelor (MDAD)

Raport tehnic

1. Citirea și încărcarea datelor din fișierul la dispoziție

Datele sunt citite din fisierul JSON folosind biblioteca python “json”: json.load(file). Mai multe detalii despre functia implementata sunt in Figura 9 din sectiunea Anexe.

2. Transformarea datelor

a. Din format JSON în formatul necesar pentru restul pipeline-ului

Formatul care faciliteaza manipularea si analiza datelor este cel oferit de biblioteca “pandas” a python. In cadrul temei am generat un dataframe pe baza datelor citite in format JSON: df = pd.DataFrame(json_data) cu care am prelucrat datele mai departe.

b. Descoperirea și corectarea erorilor care au apărut din procedura de colectare

Pentru identificarea problemelor aparute in procedura de colectare am afisat diferite informatii ale dataset-ului: dimensiunile datasetului, tipurile de date, valorile unice ale coloanelor, am iterat prin valorile coloanelor (vezi Figura 10 din sectiunea Anexe) si aditional am facut o cautarea manuala in dataset. Astfel am descoperit:

- Sunt 14206 intrari si 60 de coloane
- Tipuri de date si numarul coloanelor de acest tip:
  - object – 48
  - float64 – 11
  - int64 – 1
- Coloana “Numar locuri” este de tip float64, dar este mai corect sa fie int64
- Anumite coloane de tip object le-am convertit la float (extragand doar partea numerica) pentru ca exprimau date numerice avand aceeasi unitate de masura si este mai usor de manipulat si interpretat informatia oferita de acestea
- Unele coloane sunt in acelasi timp si lista de obiecte si obiect simplu, ex: “Audio si tehnologie”
- Majoritatea coloanelor contin valori NaN
- Coloana “Anul de fabricatie” era scrisa gresit si continea valori mai mari decat 2024
- Valoarea “Citroen” era scrisa gresit
- Coloana “VIN (serie sasiu)” nu contine date concludente, ci doar un alt string “Vezi VIN-ul (seria de sasiu)” care ascundea VIN-ul pe platforma sursa
- Anumite coloane contineau valori outliers. Coloana “Garantie dealer (inclusa in pret)”: 10000 luni, 41000 luni, “Pret”: 666666
- Anumite coloane au putine valori raportat la numarul total de intrari:

- Timp de incarcare: 6
- Masina de epoca: 30
- Numar de rate lunare ramase: 35
- Valoare rata lunara: 38
- Plata initiala (la predare): 40
- Tuning: 130
- Contract baterie: 190
- Autonomie: 260

In ceea ce priveste corectarea erorilor descoperite, am redenumit coloanele si valorile transcrise gresit si am facut conversia coloanelor in tipuri de date convenabile (vezi Figura 11 din sectiunea Anexe). Apoi, am completat valorile NaN ale fiecarei coloane cu valoarea "indisponibil" in cazul obiectelor, respectiv media valorilor coloanei respective in cazul datelor numerice.

- df[col] = df[col].apply(replace_nan_with, args=("indisponibil",))
- df[col].fillna(df[col].mean(), inplace=True)

Functia "replace_nan_with" este o metoda definita de mine care primeste un element si verifica daca acesta este NaN sau nu. Pentru datele noastre, am considerat ca aceasta asignare reflecta cat mai corect Realitatea (vezi Figura 12 din sectiunea Anexe).
Identificarea si normalizarea anomaliilor numerice le-am facut folosind metoda "Interquartile Range - IQR", pe care am adaptat-o putin, in functie de coloana pe care o aplic. Astfel, dupa mai multe teste, am ales:

- Q1 = df[column].quantile(0.20)
- Q3 = df[column].quantile(0.80)
- lower_bound = max(Q1 - 1.5 * IQR, min_value)
- upper_bound = min(Q3 + 1.5 * IQR, max_value)

Calculez in acest mod deoarece metoda clasica ar fi permis prezenta datelor eronate in set, de exemplu pret negativ, astfel pentru fiecare coloana in parte setez anumite limite logice. Un exemplu de utilizare al functiei definite:

- df = normalize_column(df, "pret", 0, 250000)

Pentru mai multe detalii referitoare la functia normalize_column, vezi Figura 13 din sectiunea Anexe.

c. Adăugarea sau eliminarea de coloane (acolo unde este cazul, de exemplu prin transformarea celor existente)

La aceasta sectiune am eliminat din dataset anumite coloane cu valori gresite sau coloane ce contineau un numar foarte mic de valori ce nu ar fi ajutat la obtinerea de informatie relevanta. Exemplu:

- df.drop("Timp de incarcare", axis=1, inplace=True)

3. Analiza datelor obținute

Dupa procesarea datelor, numarul de intrari s-a redus semnificativ, ramanand in jur de 6000. Am inceput analiza datelor prin trasarea unor grafice referitoare la date categorice, cele mai interesante coloane fiind: Marca, Combustibil, Transmisie, Norma de poluare.

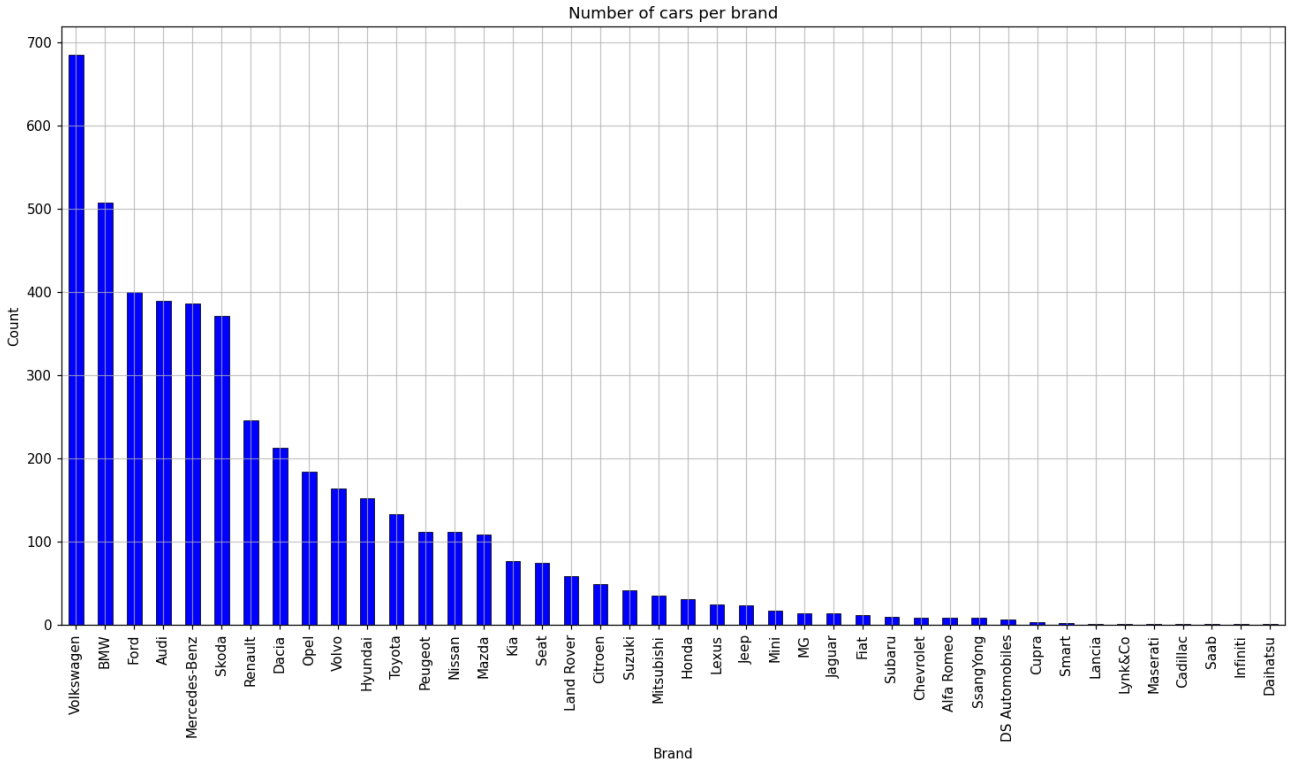


Figura 1: Numarul de masini per marca

In Figura 1 se observa destul de clar inclinatia ofertelor de vanzare spre anumite marci preferate, piata fiind predominata de 5 – 6 marci principale a caror numar de masini depaseste cu mult celelalte marci, insumate. Codul corespunzator acestui grafic se gaseste in Figura 14 din sectiunea Anexe.

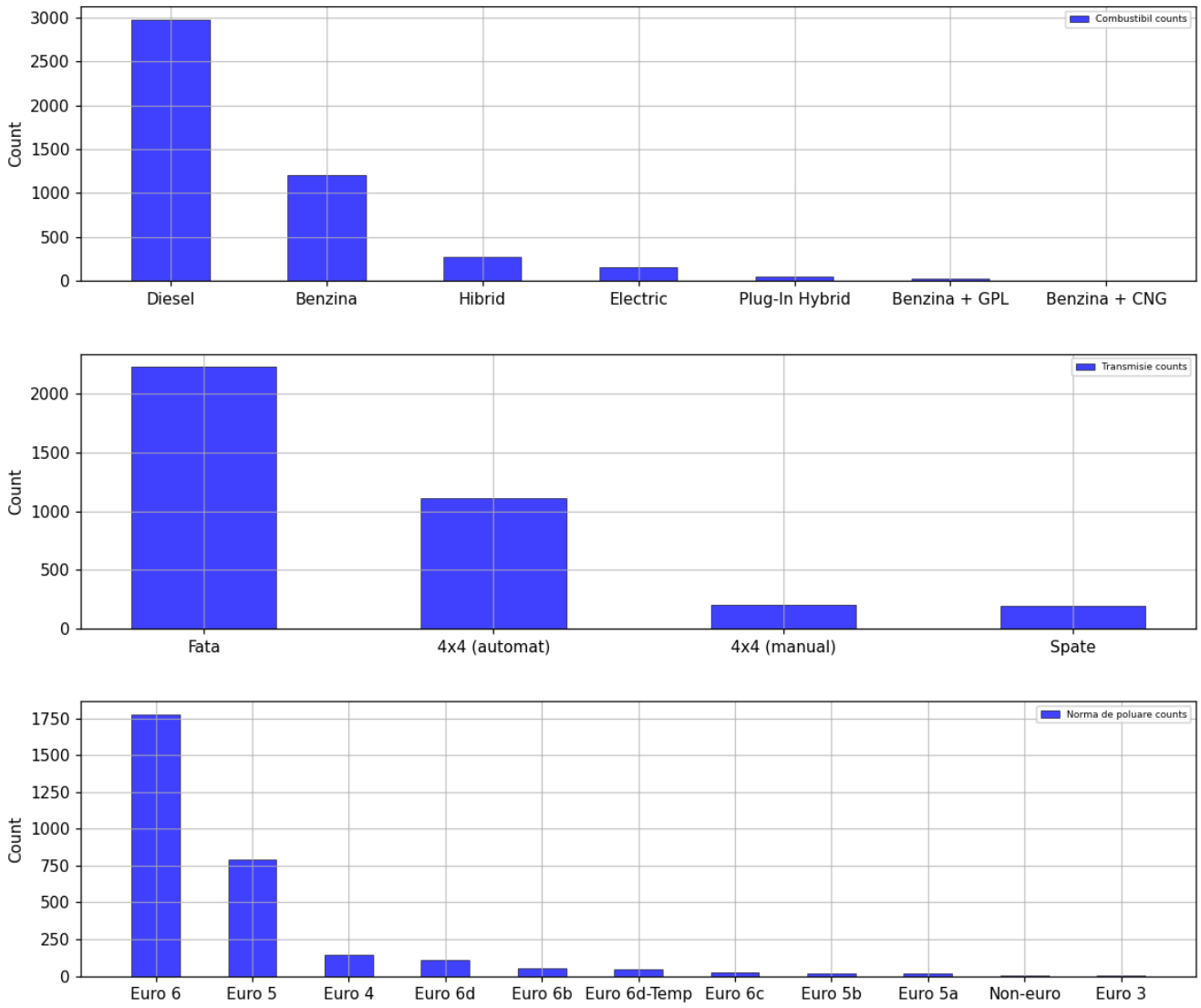


Figura 2: Numarul de masini per combustibil, transmisie si norma de poluare

Observam ca tendinta ofertei este pentru masini cu combustibil diesel, cu mult peste benzina sau alte sursa de energie. Pentru masinile ce folosesc alt combustibil decat diesel sau benzina, exista cel mai probabil un public nisa, cu preferinte foarte diferite in materie auto. Oamenii prefera transmisia fata, fiind probabil cea mai accesibila, totusi tractiunea spate care este cea mai periculoasa dintre toate ocupa un loc codas in oferta auto. O informatie foarte imbucuratoare vine din partea ultimului grafic al Figurii 2, din care reiese ca majoritatea masinilor puse la vanzare au norma de poluare Euro 6 si Euro 5, ceea ce inseamna ca sunt masini destul de noi ce vor fi puse in circulatie.
Din ambele figuri putem concluziona ca datele extrase si prelucrate au o calitate destul de buna, lipsite de anomalii, iar in urma procesarii acestora putem extrage informatie relevanta. Codul corespunzator acestor grafice se gaseste in Figura 15 din sectiunea Anexe.

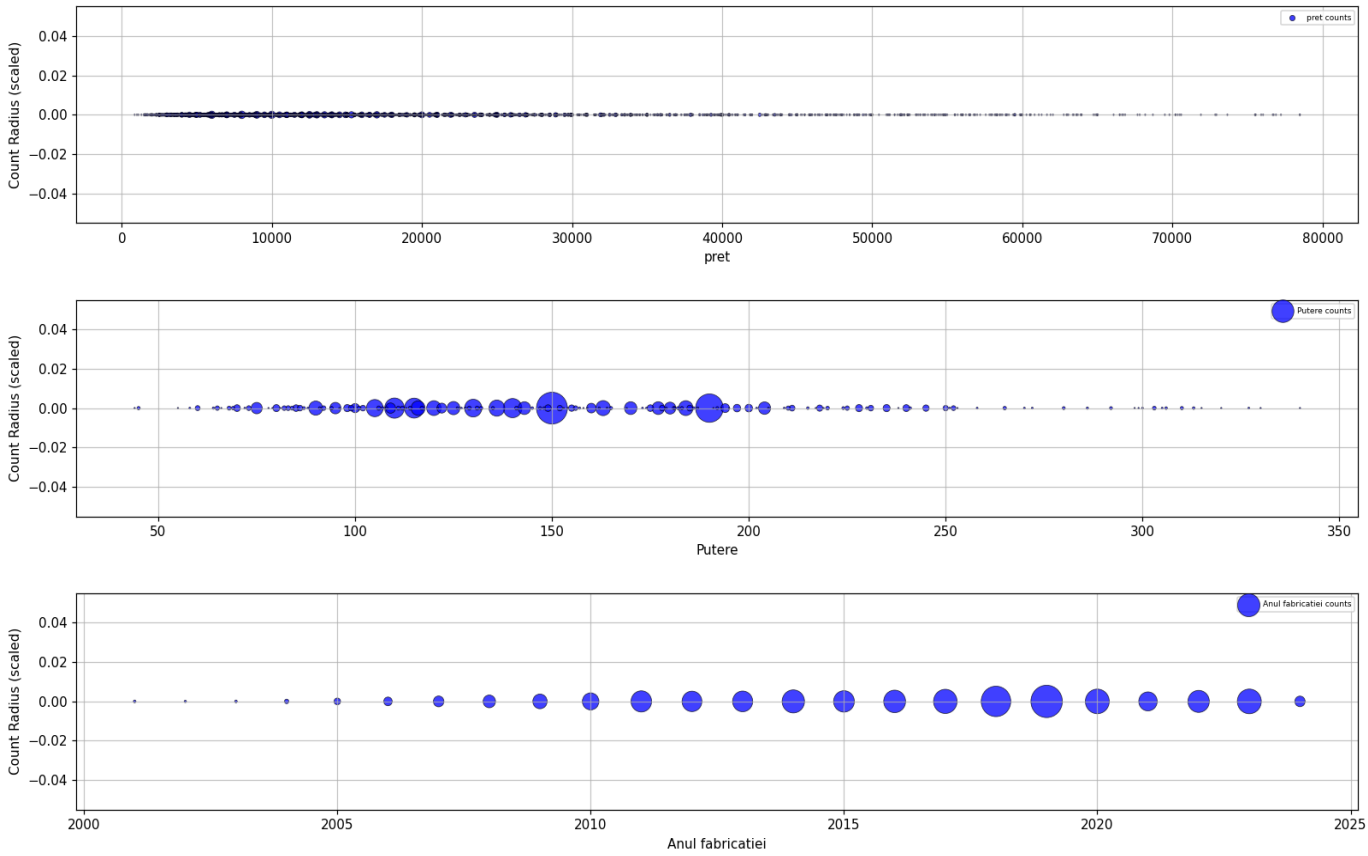


Figura 3: Numarul de masini cu un anumit pret, an de fabricatie si o anumita putere

Am ales sa reprezint grafic si date numerice, dar sub forma de “ball chart”. Aflam ca pretul de vanzare predominant se afla intr 6 – 7000 euro si 20.000 euro, dar sunt destule intrari care depasesc si 50.000 euro. In ceea ce priveste puterea masinilor, se opteaza in general pentru intervalul 100 – 200 cp, insa vedem anumite zone dominante precum: 150 si 180 cp, majoritatea masinilor avand aceasta putere fixa, indiferent de marca sau model. Cu toate acestea, exista un factor care poate induce acest interval de putere si anume combustibilul. Putem corela acest grafic cu cel referitor la combustibil din Figura 2, si cum masinile diesel predomina piata de vanzare rezulta veridicitatea legaturii dintre combustibil si gama de putere a unei masini. Vechimea predominanta pare sa fie intre 2 si 10 ani, fiind vorba despre masinile cu norma de poluare Euro 5 / 6. Cu toate acestea, pentru anul de fabricatie exista o distributie destul de liniara, cu mici exceptii in ceea ce priveste masinile noi sau cele foarte vechi. Codul corespunzator acestor grafice se gaseste in Figura 16 din sectiunea Anexe.

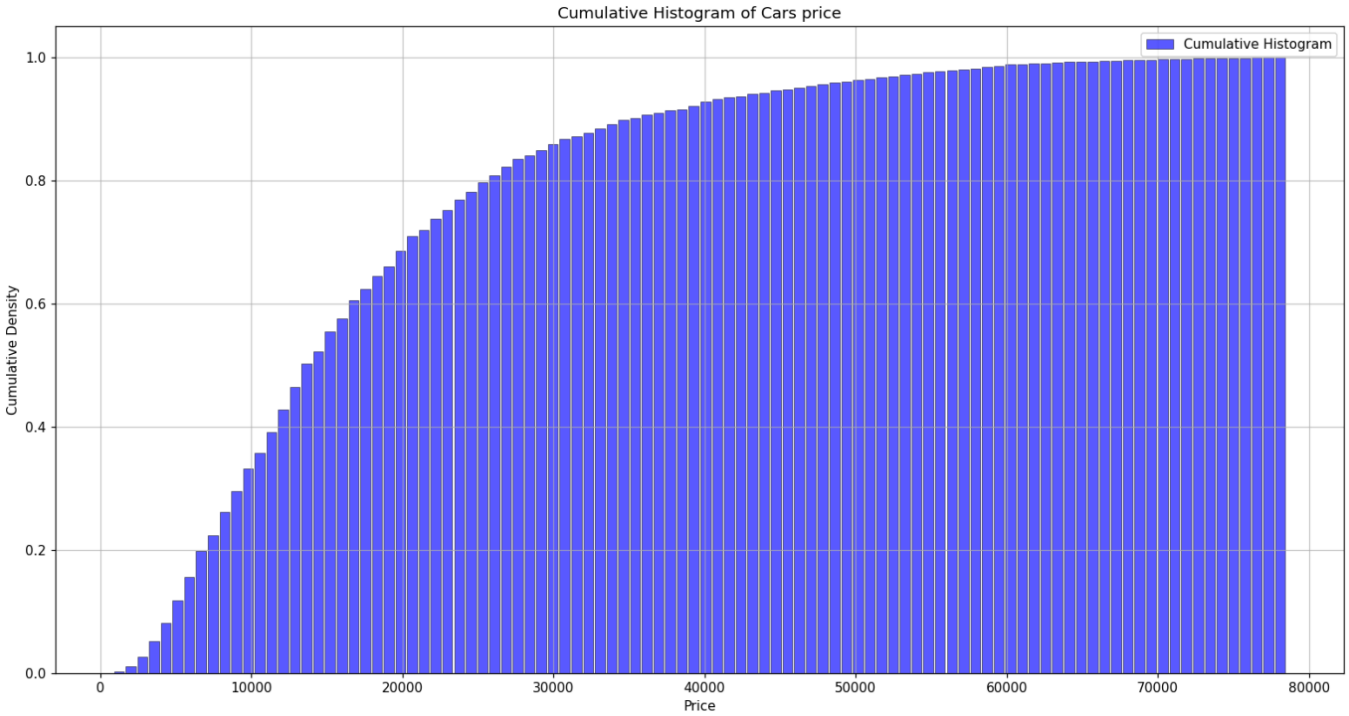


Figura 4: Histograma cumulativa a pretului masinilor

In Figura 4 am reprezentat grafic histograma cumulativa a pretului masinilor, am ales acest indicator pentru ca este unul dintre cele mai relevante in alegerea unei masini. Putem observa ca 80% dintre masini au preturi sub 25.000 euro, iar daca extindem pretul pana la 40.000 euro acoperim 90% din piata de vanzari auto. Aceasta informatie este foarte importanta pentru ca o companie de vanzari auto poate sa isi indrepte atentia catre segmentul cel mai popular si catre imbunatatirea altor segmente care sa conduca la o eficienta maxima a vanzarilor. Codul corespunzator acestui grafic se gaseste in Figura 17 din sectiunea Anexe.

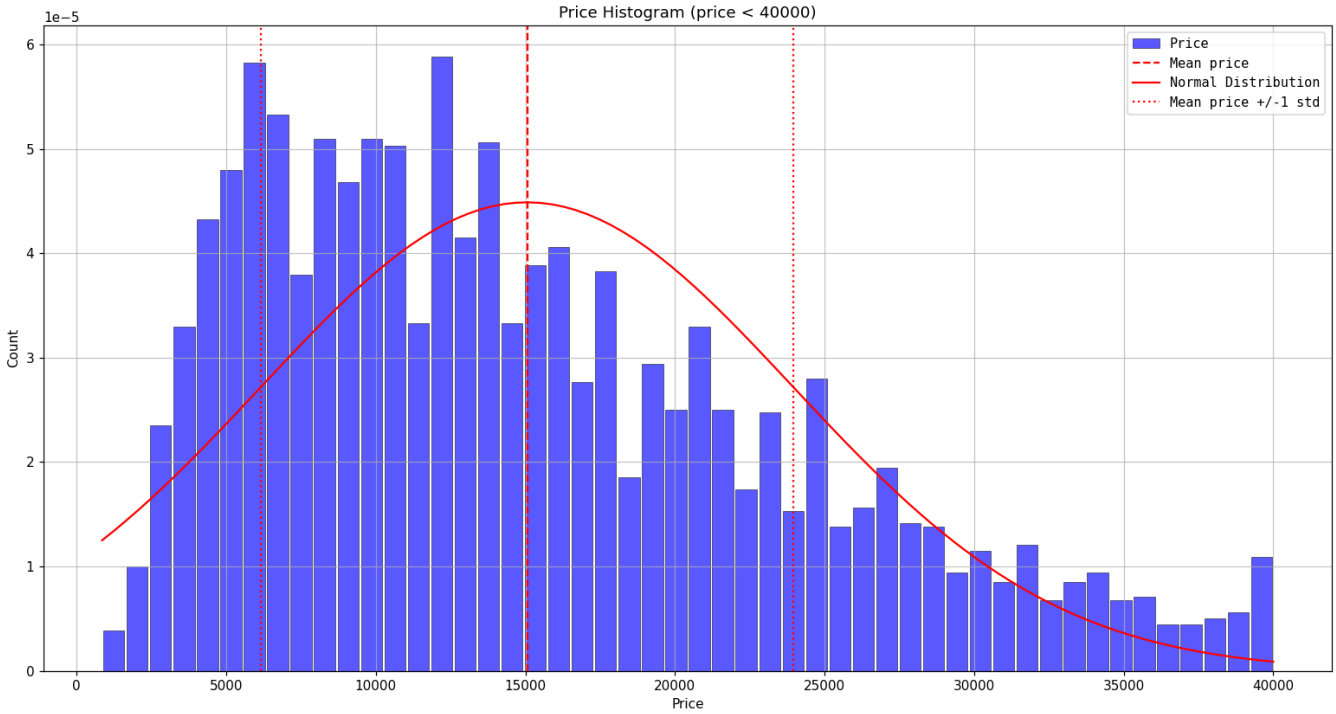
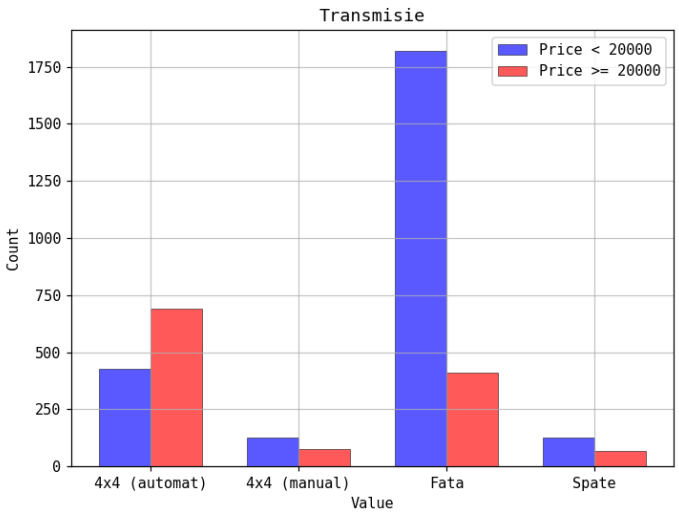
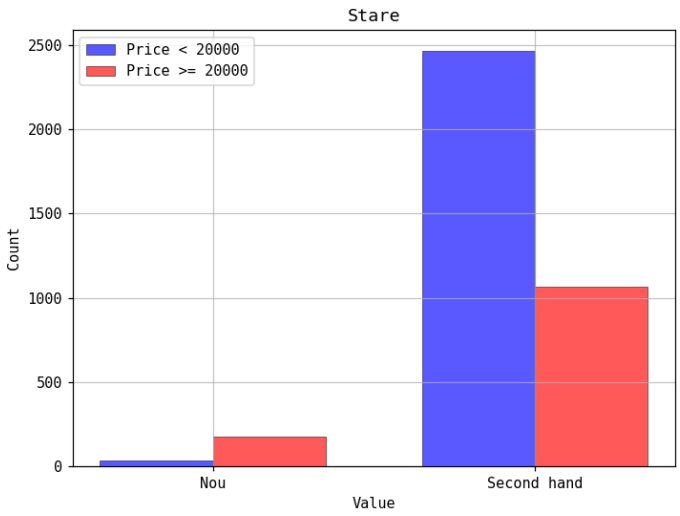


Figura 5: Histograma pretului pentru segmentul 0 – 90% din Figura 4

In Figura 5 intram in detalii in ceea ce priveste distributia pretului, in special pentru segmentul de interes. Observam ca pretul mediu al pietei auto este in jur de 15000 euro, iar deviatia standard, care reflecta variabilitatea preturilor in raport cu media, este cuprinsa intre aproximativ 6000 si 24000 euro. Aceasta variatie semnificativa arata ca preturile masinilor pot varia semnificativ de la caz la caz, in functie de diversi factori specifici pietei auto: marca, model, vechime, cilindree, stare si altele. Codul corespunzator acestui grafic se gaseste in Figura 18 din sectiunea Anexe.





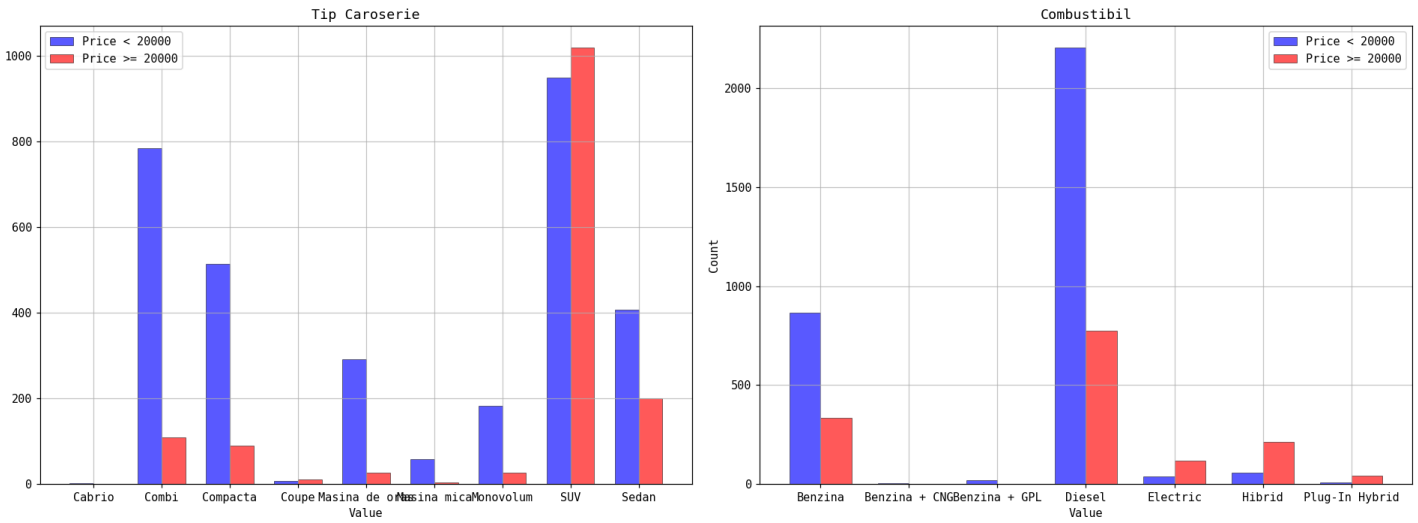


Figura 6: Date categorice impartite in doua bare in functie de pret

O alta perspectiva asupra pretului influentat de anumiti factori specifici masinilor este surprinsa in Figura 6 in care am realizat 4 grafice pentru categoriile: Transmisie, Stare, Tip Caroserie si Combustibil. Graficele sunt de tip “bar chart”, avem cate 2 bare pentru fiecare categorie: bara albastra semnifica numarul de aparitii ale categoriei in segmanetul de pret < 20.000 euro, iar bara rosie indica aparitiile categoriei in segmentul >= 20.000 euro. Aceasta partitionare ajuta la gasirea anumitor indicatori relevanti in ceea ce priveste pretul unui autovehicul, lucru pe care il vom analiza in randurile urmatoare.

In ceea ce priveste transmisia, pentru sume sub 20.000 euro primeaza transmisia fata, acestea fiind, in general, masinile obisnuite detinute de oamenii dintr-o clasa sociala medie, acelasi lucru reiesind si din graficul “Combustibil” in care masinile din aceasta categorie de pret sunt diesel, deci aparent mai economice. Maisnile cu pret peste 20.000 euro fac parte din gama SUV-urilor 4x4, care ofera siguranta sporita, insa combustibilul folosit de catre acestea tinde sa fie tot diesel. Combustibili precum “Electric” sau “Hibrid” sunt folositi in cele mai multe dintre cazuri de masinile de peste 20.000 euro, motiv pentru care alcatuiesc un segment inferior de unitati auto in comparatie cu combustibilii clasici, in care gasim si masini ieftine, atat noi cat si second-hand. Codul corespunzator acestor grafice se gaseste in Figura 19 din sectiunea Anexe.

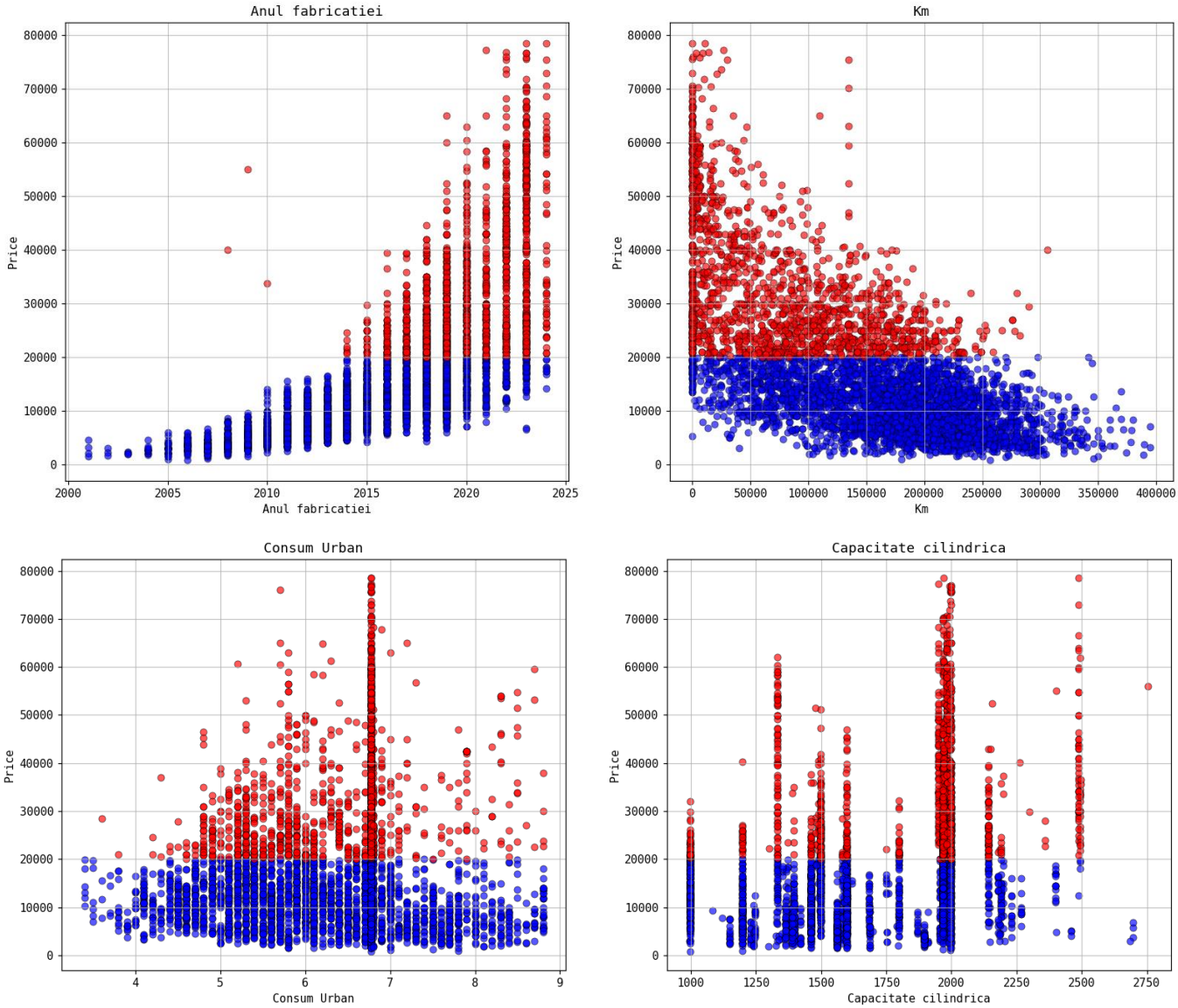


Figura 7: Diagrame de dispersie pentru Anul fabricatiei, Km, Consum Urban, Capacitate cilindrica

Figura 7 cuprinde patru diagrame de dispersie pentru unii dintre cei mai importanti factori luati in considerare la o masina, raportati la pretul de vanzare. Cerculetele sunt colorate in doua culori, cele albastre indica pretul sub 20.000 euro, iar cele rosii >= 20.000. In diagrama “Anul fabricatiei” se observa destul de clar influenta anului de fabricatie al masinii in pretul de vanzare, majoritatea masinilor mai noi de 2017 avand un pret de peste 20.000 euro, iar masinile vechi fiind mult mai ieftine cu cateva exceptii. Sunt prezente 3 anomalii: masini cu un an de fabricatie sub 2010, dar cu un pret de peste 30.000 euro. Desi ar putea fi considerate greseli de continut al datelor, aceste cazuri reprezinta unitati auto cu anumite specificatii deosebite care fac ca pretul acestora sa fie peste media obisnuita. Acelasi lucru se observa si in graficul kilometrilor, majoritatea masinilor puse la vanzare au intre 100.000 si 300.000 km si au un pret de sub 20.000 de euro, cauzat de uzura avansata a acestora. Cu toate acestea, si aici sunt prezente

cateva intrari cu un numar foarte mare de km dar cu un pret ridicat in comparatie cu media, motivul fiind acelasi ca in cazul anului de fabricatie. Graficele referitoare la consumul urban sau capacitate cilindrica difera de primele doua prin faptul ca exista anumite concentrari ale datelor care nu depind de pret, ci mai degraba de categoriile de pe axa orizontala. Observam ca sunt anumite capacitati cilindrice in jurul carora sunt dispersate valorile, ceea ce insemna ca indiferent de categoria autovehiculului, marca, model, pret si alte caracteristici, capacitatea cilindrica este comuna si impartita pe anumite segmente principale. Ex: 1400cm3, 1600cm3, 2000cm3, 2500cm3. Aceeasi idee se intampla si in cazul consumului urban, unde exista o concentrare foarte mare in zona consumului de 6.8 l/100km.

La o privire mai atenta, ar putea fi vorba despre o corelatie si o relatie de cauzalitate intre aceste date. Masini cat mai noi insemana kilometrii cat mai putini si preturi ridicate. Dar oare taria prezentei intrarilor in jurul capacitatii cilindrice de 2000cm3 induce un consum de aproximativ 6.8l/100km? Nu exista un raspuns bine definit la aceasta intrebare, dar cert este ca putem obtine informatii valoroase in urma analizei acestor grafice. Codul corespunzator acestor grafice se gaseste in Figura 20 din sectiunea Anexe.

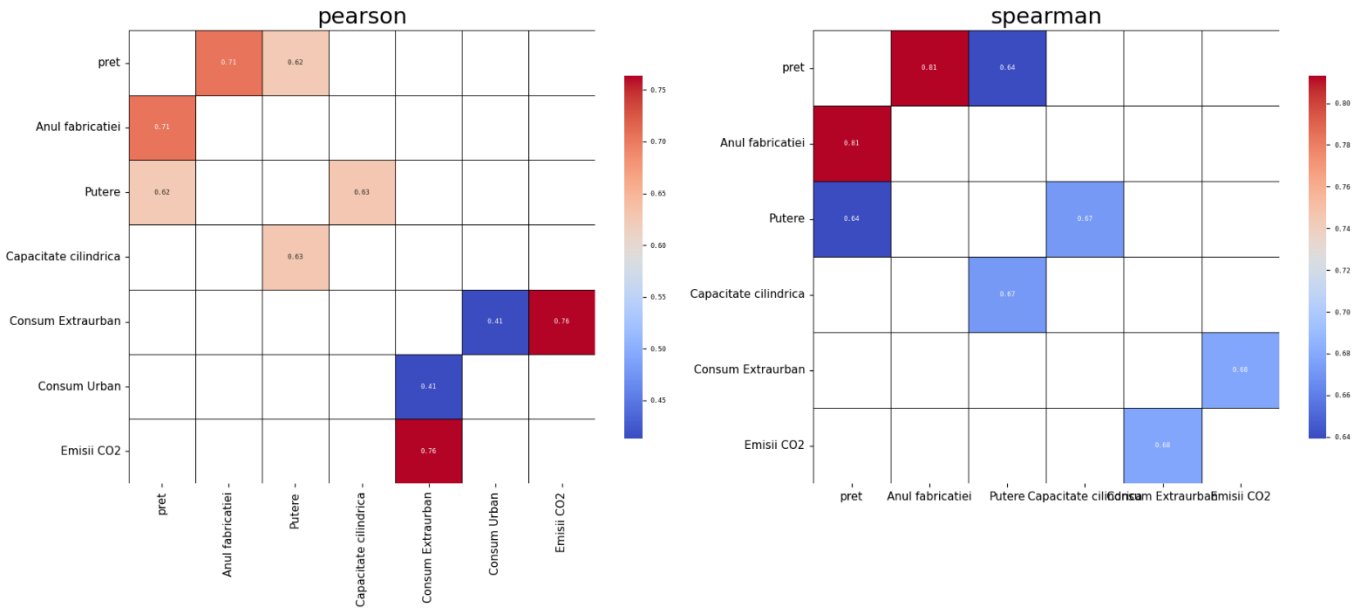


Figura 8: Matricele de corelatie Pearson si Spearman

Exista insa anumiti indici care ofera o perspectiva mult mai buna asupra corelatiei dintre dintre variabile numerice, si anume coeficientul Pearson si coeficientul Spearman. Acesti indici exprima taria variabilitatii comune ale valorilor numerice si ofera o forma cuantificabila a corelatie acestora. Pentru dataset-ul curent, am pastrat doar valorile de corelatie mai mari de 0.4 deoarece sunt mai relevante. Culorile alese evidentiaza, de asemenea, masura corelatiei, albastru inseamna corelatie slaba, iar rosu semnifica corelatie puternica. Putem observa ca datasetul nostru are date care sunt corelate, precum Anul fabricatiei – pret, Consum Extraurban – Emisii CO2, Putere – pret, Capacitate cilindrica – Putere. Codul corespunzator acestor grafice se gaseste in Figura 21 din sectiunea Anexe.

4. Anexe

```
def read_data(path):
    with open(path, 'r') as file:
        data = json.load(file)
    return data
```

Figura 9: citire fisier JSON

```
    print(df.shape)
    print(df.dtypes)
    print(df.dtypes.value_counts())

    for col in df.select_dtypes('object').columns:
        if not isinstance(df[col].iloc[0], list):
            print(col, df[col].unique())

            for el in df[col]:
                if isinstance(el, list):
                    print(col, set(el))
                else:
                    print(el)
        else:
            print(col, df[col].unique())

        print(col, df[col].unique())

    for lista in df["Audio si tehnologie"]:
        if isinstance(lista, list):
            print(set(lista))
        else:
            print(lista)
```

Figura 10: descoperirea erorilor din dataset

```
df.rename(columns={'Anul fabricaÈ›iei': 'Anul fabricatiei'}, inplace=True)
    df['Marca'] = df['Marca'].replace('CitroÃ«n', 'Citroen')
    df['Valoare rata lunara'] = df['Valoare rata
lunara'].str.extract('(\d+)').astype(float)
    df['Plata initiala (la predare)'] = df['Plata initiala (la
predare)'].str.extract('(\d+)').astype(float)
```

```
    df['Valoare reziduala'] = df['Valoare
reziduala'].str.extract('(\d+)').astype(float)
    df['Consum Mixt'] = df['Consum Mixt'].str.extract('(\d+)').astype(float)
    df['Garantie dealer (inclusa in pret)'] = df['Garantie dealer (inclusa in
pret)'].str.extract('(\d+)').astype(float)
```

Figura 11: redenumire si reconversie de tip

```
# iterez prin variabilele te dip object si inlocuiesc valorile NAN cu
"indisponibil" sau ["indisponibil"], dupa caz
    for col in df.select_dtypes('object').columns:
        if isinstance(df[col], list) or isinstance(df[col].iloc[0], list):
            df[col] = df[col].apply(replace_nan_with, args=(["indisponibil"],))
        else:
            df[col] = df[col].apply(replace_nan_with, args=("indisponibil",))



    # iterez prin variabilele te dip float64 si int64 si inlocuiesc valorile NAN
cu media valorilor pe coloana respectiva
    for col in df.columns:
        if df[col].dtype == 'float64' or df[col].dtype == 'int64':
            df[col].fillna(df[col].mean(), inplace=True)
```

Figura 12: inlocuire NaN cu valori convenabile

```
def normalize_column(df, column, min_value, max_value):
    Q1 = df[column].quantile(0.20)
    Q3 = df[column].quantile(0.80)

    IQR = Q3 - Q1

    lower_bound = max(Q1 - 1.5 * IQR, min_value)
    upper_bound = min(Q3 + 1.5 * IQR, max_value)

    return df[(df[column] >= lower_bound) & (df[column] <= upper_bound)]
```

Figura 13: functia de normalizare a coloanelor numerice

```
cars_brands = df["Marca"].value_counts()
    cars_brands.plot(kind='bar', color='blue', edgecolor='black', linewidth=0.5,
width=0.5)
    plt.grid(alpha=0.75)
    plt.title('Number of cars per brand')
    plt.xlabel('Brand')
```

```
    plt.ylabel('Count')
    plt.show()
```

Figura 14: realizare grafic din Figura 1

```
cols = [ "Combustibil", "Transmisie", "Norma de poluare"]
    fig, ax = plt.subplots(3, 1, figsize=(8, 16), dpi=100)
    plt.rcParams['font.size'] = 6
    for idx, col in enumerate(cols):
        filtered_df = df[df[col] != "indisponibil"]
        counts = filtered_df[col].value_counts()
        ax[idx].bar(counts.index, counts.values, alpha=0.75, label=f'{col}
counts', color='blue', edgecolor='black', linewidth=0.5, width=0.5)
        ax[idx].set_ylabel('Count')
        ax[idx].grid(alpha=0.75)
        ax[idx].legend()
    plt.tight_layout()
    plt.show()
```

Figura 15: realizare grafic din Figura 2

```
cols = ["pret", "Putere", "Anul fabricatiei"]
    fig, ax = plt.subplots(3, 1, figsize=(8, 16), dpi=100)
    plt.rcParams['font.size'] = 6
    for idx, col in enumerate(cols):
        counts = df[col].value_counts()
        ax[idx].scatter(counts.index, np.zeros(len(counts)), alpha=0.75,
label=f'{col} counts', color='blue', edgecolor='black', linewidth=0.5,
s=counts.values)
        ax[idx].set_ylabel('Count Radius (scaled)')
        ax[idx].set_xlabel(col)
        ax[idx].grid(alpha=0.75)
        ax[idx].legend()
    plt.tight_layout()
    plt.show()
```

Figura 16: realizare grafic din Figura 3

```
fig, ax = plt.subplots(1, 1, figsize=(10, 5), dpi=100)
    plt.hist(df['pret'], bins=100, cumulative=True, density=True, alpha=0.65,
label='Cumulative Histogram', color='blue', edgecolor='black', linewidth=0.5,
rwidth=0.85)
    plt.title('Cumulative Histogram of Cars price')
    plt.ylabel('Cumulative Density')
```

```
    plt.xlabel('Price')
    plt.grid(alpha=0.75)
    plt.legend()
    plt.show()
```

Figura 17: realizare grafic din Figura 4

```
truncated_df = df[df['pret'] < 40000]
    fig, ax = plt.subplots(1, 1, figsize=(10, 5), dpi=100)
    plt.rcParams['font.family'] = 'monospace'
    plt.hist(truncated_df['pret'], bins=50, alpha=0.65, color='blue',
edgecolor='black', linewidth=0.5, label='Price', rwidth=0.9, density=True)
    plt.axvline(truncated_df['pret'].mean(), color='red', linestyle='--',
label='Mean price')
    x = np.linspace(truncated_df['pret'].min(), truncated_df['pret'].max(), 100)
    y = stats.norm.pdf(x, truncated_df['pret'].mean(),
truncated_df['pret'].std())
    plt.plot(x, y, color='red', linestyle='-', label='Normal Distribution')
    plt.axvline(truncated_df['pret'].mean() + truncated_df['pret'].std(),
color='red', linestyle='dotted', label='Mean price +/-1 std')
    plt.axvline(truncated_df['pret'].mean() - truncated_df['pret'].std(),
color='red', linestyle='dotted')
    plt.title('Price Histogram (price < 40000)')
    plt.xlabel('Price')
    plt.ylabel('Count')
    plt.grid(alpha=0.75)
    plt.legend()
    plt.show()
```

Figura 18: realizare grafic din Figura 5

```
plt.rcParams['font.family'] = 'monospace'
    low_prices = df[df['pret'] < 20000 ]
    high_prices = df[df['pret'] >= 20000]
    categories = ["Transmisie", "Stare", "Tip Caroserie", "Combustibil"]
    fig, ax = plt.subplots(1, len(categories), figsize=(20, 5), dpi=100)
    for i, category in enumerate(categories):
        low_counts = low_prices[category].value_counts()
        high_counts = high_prices[category].value_counts()
        low_counts_dict = low_counts.to_dict()
        high_counts_dict = high_counts.to_dict()
        for key in low_counts_dict.keys():
            if key not in high_counts_dict:
                high_counts_dict[key] = 0
```

```
        for key in high_counts_dict.keys():
            if key not in low_counts_dict:
                low_counts_dict[key] = 0
        # sort the values by the key
        low_counts_dict = dict(sorted(low_counts_dict.items()))
        high_counts_dict = dict(sorted(high_counts_dict.items()))
        # get the values of the counts
        low_counts_values = list(low_counts_dict.values())
        high_counts_values = list(high_counts_dict.values())
        # get the keys of the counts
        low_counts_keys = list(low_counts_dict.keys())
        high_counts_keys = list(high_counts_dict.keys())
        ind = np.arange(len(low_counts_keys))
        width = 0.35
        ax[i].bar(ind, low_counts_values, width, alpha=0.65, color='blue',
edgecolor='black', linewidth=0.5, label='Price < 20000')
        ax[i].bar(ind + width, high_counts_values, width, alpha=0.65,
color='red', edgecolor='black', linewidth=0.5, label='Price >= 20000')
        ax[i].set_xticks(ind + width / 2)
        ax[i].set_xticklabels(low_counts_keys)
        ax[i].set_title(category)
        ax[i].set_xlabel('Value')
        ax[i].set_ylabel('Count')
        ax[i].grid(alpha=0.75)
        ax[i].legend()
    plt.tight_layout()
    plt.show()
```

Figura 19: realizare grafic din Figura 6

```
 plt.rcParams['font.family'] = 'monospace'
    low_prices = df[df['pret'] < 20000]
    high_prices = df[df['pret'] >= 20000]
    numerical_categories = ['Consum Urban', 'Capacitate cilindrica', 'Anul
fabricatiei', 'Km']
    fig, ax = plt.subplots(1, len(numerical_categories), figsize=(20, 5),
dpi=100)
    for i, variable in enumerate(numerical_categories):
        ax[i].scatter(low_prices[variable], low_prices['pret'], alpha=0.65,
color='blue', edgecolor='black', linewidth=0.5, label='Price < 20000')
        ax[i].scatter(high_prices[variable], high_prices['pret'], alpha=0.65,
color='red', edgecolor='black', linewidth=0.5, label='Price >= 20000')
        ax[i].set_title(variable)
        ax[i].set_xlabel(variable)
        ax[i].set_ylabel('Price')
```

```
        ax[i].grid(alpha=0.75)
    plt.show()
```

Figura 20: realizare grafic din Figura 7

```
fig, ax = plt.subplots(1, 2, figsize=(30, 10), dpi=100)
    plt.rcParams['font.family'] = 'monospace'
    plt.rcParams['font.size'] = 6
    numeric_columns = df.select_dtypes(include=['float64', 'int64']).columns
    numeric_df = df[numeric_columns]
    for i, corr_type in enumerate(['pearson', 'spearman']):
        corr_df = numeric_df.corr(corr_type)
        corr_df = corr_df - np.diag(np.diag(corr_df))
        corr_df = corr_df[corr_df > 0.4]
        corr_df = corr_df.dropna(axis=0, how='all')
        corr_df = corr_df.dropna(axis=1, how='all')
        sns.heatmap(corr_df, cmap='coolwarm', annot=True, fmt='.2f',
linewidths=0.5, linecolor='black', square=False, cbar=True,
cbar_kws={'orientation': 'vertical', 'shrink': 0.8, 'pad': 0.05}, ax=ax[i],
mask=corr_df.isnull())
        ax[i].set_title(corr_type, fontsize=20)
    plt.tight_layout()
    plt.show()
```

Figura 21: realizare grafic din Figura 8