emisije-vazduh

May 18, 2025

Emisije u vazduhu EDA analiza sa ARIMA modelom predikcije CO2

Podaci preuzeti sa data.gov.rs

Izvor podataka:Agencija za zastitu zivotne sredine

Agencija za zaštitu životne sredine, kao organ u sastavu Ministarstva zaštite životne sredine, sa svojstvom pravnog lica, obavlja stručne poslove koji se odnose na: razvoj, usklađivanje i vođenje-nacionalnog informacionog sistema zaštite životne sredine (praćenje stanja činilaca životne sredine kroz...

Link prema skupu podataka:https://data.gov.rs/sr/datasets/emisije-u-vazdukh/#community-resources

Permalink:https://data.gov.rs/sr/datasets/emisije-u-vazdukh/

Datum preuzimanja:10.05.2025 godine

Opis: Ovaj set podataka sadrži informacije o emisijama zagađujućih materija u vazduh u Republici Srbiji, prikupljene iz Nacionalnog registra izvora zagađivanja. Podaci se odnose na različite privredne subjekte koji emituju zagađujuće materije, uz detalje o vrsti i količini emisija.

Atributi seta podataka Okrug – Naziv okruga u kojem se nalazi izvor emisije Region – Naziv regiona u kojem se nalazi izvor emisije Opština – Naziv opštine u kojoj se nalazi izvor emisije Mesto – Naziv mesta u kojem se nalazi izvor emisije Šifra Mesta – Jedinstvena šifra mesta u skladu sa zvaničnim kodiranjem Pretežna delatnost – Osnovna delatnost preduzeća koje emituje zagađujuće materije PIB – Poreski identifikacioni broj preduzeća Preduzeće – Naziv preduzeća koje emituje zagađujuće materije Nacionalni ID – Jedinstveni identifikator preduzeća u Nacionalnom registru izvora zagađenja Postrojenje – Naziv postrojenja koje emituje zagađujuće materije PRTR kod – Kod u skladu sa Protokolom o registrima ispuštanja i prenosa zagađivača (PRTR) Zagađujuća materija – Naziv zagađujuće materije koja se emituje u vazduh Količina (kg/god) – Količina emitovane zagađujuće materije u kilogramima godišnje

Ovi podaci omogućavaju praćenje izvora emisija u vazduh, analizu uticaja industrijskih postrojenja na kvalitet vazduha i podršku u donošenju mera za smanjenje zagađenja.

Šifarnik naseljenih mesta u Srbiji, link:https://www.neobilten.com/sifarnik-naseljenih-mesta-u-republici-srbiji/

[1]:
```
# Učitavanje potrebnih biblioteka
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

```
import seaborn as sns
print('Setup Complete!')
```

```
Setup Complete!
```

[2]:
```
# Učitavanje podataka
df = pd.read_excel("emisije_vazduh.xlsx")
# Prikaz prvih nekoliko redova
# print(df.head())
```

[3]:
```
# Prikaz prvih redova u tabelarnom formatu
df.head(3)
```

[3]:

	Godina	Okrug	Region	Opstina	Mesto	SifraMesta	PreteznaDelatnost	PIB	Preduzece	NacionalniId	Postrojenje	PRTRKod	ZagadjujucaMaterija	KolicinaKgGod
0	2023	Borski okrug	Region Južne i Istočne Srbije	Bor	Bor	706418	0729 Eksploatacija ruda ostalih crnih, obojeni…	100570195	SERBIA ZIJIN COPPER DOO BOR	100570195/3	Ogranak RBB Bor - Površinski kop Cerovo	3.(b)	Suspendovane čestice (PM10)	17067.7
1	2023	Borski okrug	Region Južne i Istočne Srbije	Bor	Bor	706418	0729 Eksploatacija ruda ostalih crnih, obojeni…	100570195	SERBIA ZIJIN COPPER DOO BOR	100570195/4	Ogranak RBB Bor - Površinski kop Veliki krivelj	3.(b)	Suspendovane čestice (PM10)	34025.6
2	2023	Borski okrug	Region Južne i Istočne Srbije	Bor	Bor	706418	0729 Eksploatacija ruda ostalih crnih, obojeni…	100570195	SERBIA ZIJIN COPPER DOO BOR	100570195/5	Ogranak RBB Bor - Površinski kop Jama	3.(b)	Azotni oksidi (NOx/NO2)	4939.8

[4]:
```
# Prikazati osnovni oblik dataseta
print(f'Ukupan broj redova u datasetu je: {df.shape[0]}')
print(f'Ukupan broj kolona u datasetu je: {df.shape[1]}')
```

```
Ukupan broj redova u datasetu je: 12292
Ukupan broj kolona u datasetu je: 14
```

[5]:
```
# Prikaz osnovnih informacija o datasetu.
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12292 entries, 0 to 12291
Data columns (total 14 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Godina                12292 non-null  int64
 1   Okrug                 12292 non-null  object
 2   Region                12292 non-null  object
 3   Opstina               12292 non-null  object
 4   Mesto                 12292 non-null  object
 5   SifraMesta            12292 non-null  int64
 6   PreteznaDelatnost     12285 non-null  object
 7   PIB                   12292 non-null  int64
 8   Preduzece             12292 non-null  object
 9   NacionalniId          12292 non-null  object
 10  Postrojenje           12292 non-null  object
 11  PRTRKod               11786 non-null  object
 12  ZagadjujucaMaterija   12292 non-null  object
 13  KolicinaKgGod         12292 non-null  float64
dtypes: float64(1), int64(3), object(10)
memory usage: 1.3+ MB
```

[6]:
```
# Prikaži sve kolone u datasetu
df.columns
```

[6]:
```
Index(['Godina', 'Okrug', 'Region', 'Opstina', 'Mesto', 'SifraMesta',
       'PreteznaDelatnost', 'PIB', 'Preduzece', 'NacionalniId', 'Postrojenje',
       'PRTRKod', 'ZagadjujucaMaterija', 'KolicinaKgGod'],
      dtype='object')
```

[7]:
```
# konvertovati kolone 'SifraMesta' i 'PIB' iz int64 u object (string)
df["SifraMesta"] = df["SifraMesta"].astype(str)
df["PIB"] = df["PIB"].astype(str)

# Provera tipova podataka
print(df.dtypes)
```

```
Godina                 int64
Okrug                 object
Region                object
Opstina               object
Mesto                 object
SifraMesta            object
PreteznaDelatnost     object
PIB                   object
Preduzece             object
NacionalniId          object
Postrojenje           object
```

```
PRTRKod                 object
ZagadjujucaMaterija     object
KolicinaKgGod          float64
dtype: object
```

[8]:
```
# Pretvaranje svih vrednosti u svim kolonama DataFrame-a u mala slova pomoću map() funkcije
df = df.map(lambda x: x.lower() if isinstance(x, str) else x)
# Provera rezultata
df.head(3)
```

[8]:

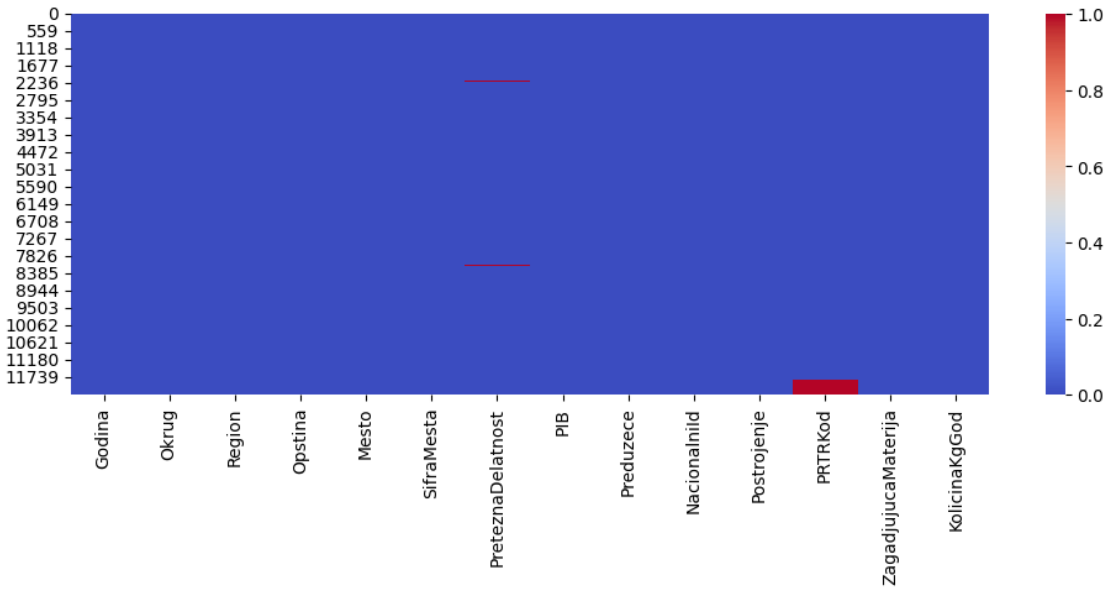
	Godina	Okrug	Region	Opstina	Mesto	SifraMesta	PreteznaDelatnost	PIB	Preduzece	NacionalniId	Postrojenje	PRTRKod	ZagadjujucaMaterija	KolicinaKgGod
0	2023	borski okrug	region južne i istočne srbije	bor	bor	706418	0729 eksploatacija ruda ostalih crnih, obojeni…	100570195	serbia zijin copper doo bor	100570195/3	ogranak rbb bor - površinski kop cerovo	3.(b)	suspendovane čestice (pm10)	17067.7
1	2023	borski okrug	region južne i istočne srbije	bor	bor	706418	0729 eksploatacija ruda ostalih crnih, obojeni…	100570195	serbia zijin copper doo bor	100570195/4	ogranak rbb bor - površinski kop veliki krivelj	3.(b)	suspendovane čestice (pm10)	34025.6
2	2023	borski okrug	region južne i istočne srbije	bor	bor	706418	0729 eksploatacija ruda ostalih crnih, obojeni…	100570195	serbia zijin copper doo bor	100570195/5	ogranak rbb bor - površinski kop jama	3.(b)	azotni oksidi (nox/no2)	4939.8

[10]:
```
# provera da li ima vrednosti koje nedostaju u datasetu
print(df.isnull().sum())
```

```
Godina                 0
Okrug                  0
Region                 0
Opstina                0
Mesto                  0
SifraMesta             0
PreteznaDelatnost      7
PIB                    0
Preduzece              0
```

```
NacionalniId              0
Postrojenje               0
PRTRKod                 506
ZagadjujucaMaterija       0
KolicinaKgGod             0
dtype: int64
```

[11]:
```
# heatmap podataka koji nedostaju
plt.figure(figsize=(12,4))
sns.heatmap(df.isnull(), cmap='coolwarm')
plt.show()
```



[12]:
```
# Provera dupliranih redova
duplirani_redovi = df.duplicated().sum()
print(f"Broj dupliranih redova: {duplirani_redovi}")
```

```
Broj dupliranih redova: 0
```

[13]:
```
# Prikazati sve zagadjivače u datasetu
df['ZagadjujucaMaterija'].unique()
```

[13]:
```
array(['suspendovane čestice (pm10)', 'azotni oksidi (nox/no2)',
       'sumporni oksidi (sox/so2)', 'ugljen dioksid (co2)',
       'ugljen monoksid (co)', 'amonijak (nh3)',
       'fluor i neorganska jedinjenja (kao hf)', 'metan (ch4)',
       'nemetanska isparljiva organska jedinjenja (nmvoc)',
       'ukupni organski ugljenik (toc) (ukupni c ili cod/3)', 'benzen',
       'hlor i neorganska jedinjenja (kao hcl)',
```

```
'hrom i jedinjenja hroma (kao cr)', 'naftalen',
'nikl i jedinjenja nikla (kao ni)',
'pcdd + pcdf (dioksini+furani) (kao teq)',
'olovo i jedinjenja olova (kao pb)',
'arsen i jedinjenja arsena (kao as)',
'bakar i jedinjenja bakra (kao cu)',
'kadmijum i jedinjenja kadmijuma (kao cd)',
'živa i jedinjenja žive (kao hg)', 'azot suboksid (n2o)',
'cijanovodonik (hcn)',
'policiklični aromatični ugljovodonici (pahs)', 'aldrin',
'fluorougljovodonici (hfcs)', 'pentahlorofenol (pcp)',
'heksahlorobenzen (hcb)', 'cink i jedinjenja cinka (kao zn)',
'etilen oksid',
'di-(2-etil heksil) ftalat (dehp)/bis (2-etilheksil)ftalat',
'tetrahlorometan (tcm)/ugljen tetrahlorid'], dtype=object)
```

[14]:
```
# Provera osnovne statistike za kolonu KolicinaKgGod
print(df['KolicinaKgGod'].describe())
```

```
count    1.229200e+04
mean     2.749272e+07
std      5.096076e+08
min      0.000000e+00
25%      4.500500e+02
50%      4.834900e+03
75%      3.165892e+04
max      2.772834e+10
Name: KolicinaKgGod, dtype: float64
```

Komentar: Ovu statistuku treba posmatrati uslovno, uzimajući u obzir da u koloni 'ZagadjujucaMaterija' imamo veći broj pojedinačno identifikovanih zagadjivača koji sublimirano prikazuju u masi statistiku za kolonu 'KolicinaKgGod'.Visoka standardna devijacija znači da emisije nisu uniformne, već da postoje veliki skokovi među pojedinačnim unosima. U ovom slučaju, standardna devijacija je mnogo veća od prosečne vrednosti, što ukazuje na velike razlike u emisijama između sektora i lokacija.

[15]:
```
# alternativna statistika za kategorijsku kolonu
print(f'Broj jedinstvenih zagađujućih čestica u datasetu je:␣
  ↪{df["ZagadjujucaMaterija"].nunique()}')  # Broj jedinstvenih vrednosti
print(df["ZagadjujucaMaterija"].value_counts())  # Frekvencija vrednosti
```

```
Broj jedinstvenih zagađujućih čestica u datasetu je: 32
ZagadjujucaMaterija
suspendovane čestice (pm10)                    2199
azotni oksidi (nox/no2)                        1956
ugljen monoksid (co)                           1440
sumporni oksidi (sox/so2)                      1413
amonijak (nh3)                                 1108
```

```
nemetanska isparljiva organska jedinjenja (nmvoc)            917
metan (ch4)                                                  685
ukupni organski ugljenik (toc) (ukupni c ili cod/3)          474
ugljen dioksid (co2)                                         370
hlor i neorganska jedinjenja (kao hcl)                       355
fluor i neorganska jedinjenja (kao hf)                       305
azot suboksid (n2o)                                          245
benzen                                                       178
olovo i jedinjenja olova (kao pb)                            123
hrom i jedinjenja hroma (kao cr)                              84
nikl i jedinjenja nikla (kao ni)                              84
pcdd + pcdf (dioksini+furani) (kao teq)                       84
bakar i jedinjenja bakra (kao cu)                             61
kadmijum i jedinjenja kadmijuma (kao cd)                      56
arsen i jedinjenja arsena (kao as)                            49
živa i jedinjenja žive (kao hg)                               44
cink i jedinjenja cinka (kao zn)                              23
policiklični aromatični ugljovodonici (pahs)                   9
cijanovodonik (hcn)                                            8
aldrin                                                         6
naftalen                                                       4
pentahlorofenol (pcp)                                          3
fluorougljovodonici (hfcs)                                     3
di-(2-etil heksil) ftalat (dehp)/bis (2-etilheksil)ftalat      3
heksahlorobenzen (hcb)                                         1
etilen oksid                                                   1
tetrahlorometan (tcm)/ugljen tetrahlorid                       1
Name: count, dtype: int64
```

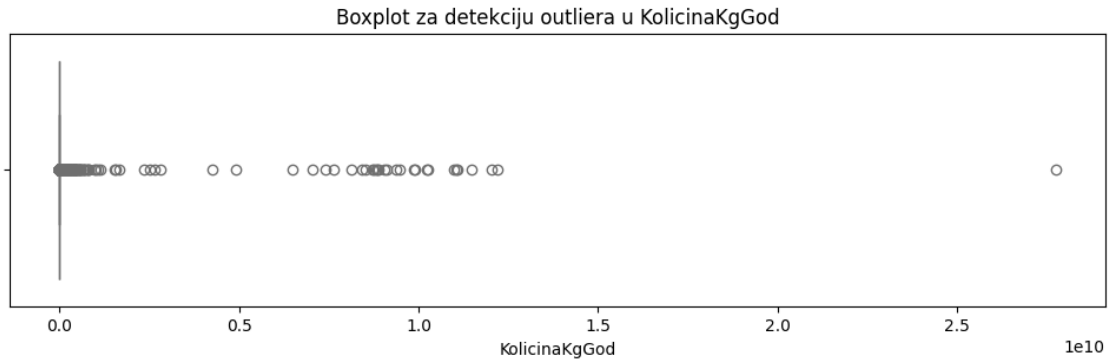
Komentar: PM10 je najčešće meren zagađivač, verovatno zbog njegovog uticaja na respiratorno zdravlje ljudi. NOx i SOx su glavni indikatori industrijskog zagađenja, pa dataset može pružiti važne uvide u oblasti sa visokim emisijama. Metan (CH) i nemetanska isparljiva organska jedinjenja (NMVOC) su prisutna, ali manje zastupljena – korisni za analizu efekata na klimatske promene.

[16]:
```
# Identifikujemo prisustvo outlier-a, koliko ih ima u datasetu i koliki je njihov procenat učešća u datasetu
# Outlier-e tražimo u koloni KolicinaKgGod pošto je njen dtype int64

# IQR metoda za detekciju outliera u koloni KolicinaKgGod
def count_outliers_kolicina(df, column):
    total_rows = len(df)

    Q1 = np.percentile(df[column], 25)
    Q3 = np.percentile(df[column], 75)
    IQR = Q3 - Q1

    lower_bound = Q1 - 1.5 * IQR
```

```
    upper_bound = Q3 + 1.5 * IQR

    # Broj outliera u odabranoj koloni
    num_outliers = ((df[column] < lower_bound) | (df[column] > upper_bound)).
 ↪sum()
    percentage_outliers = (num_outliers / total_rows) * 100

    return num_outliers, round(percentage_outliers, 2)

# Poziv funkcije za kolonu 'KolicinaKgGod'
num_outliers, percent_outliers = count_outliers_kolicina(df, "KolicinaKgGod")

# Prikaz rezultata
print(f'Ukupan broj outliera u koloni KolicinaKgGod: {num_outliers}')
print(f'Procenat outliera u koloni KolicinaKgGod: {percent_outliers}%')
```

```
Ukupan broj outliera u koloni KolicinaKgGod: 1943
Procenat outliera u koloni KolicinaKgGod: 15.81%
```

[17]:
```
# Boxplot za identifikaciju outliera

# Kreiranje figure
plt.figure(figsize=(12, 3))
sns.boxplot(x=df["KolicinaKgGod"], color="skyblue")

# Dodavanje naslova i etiketa
plt.title("Boxplot za detekciju outliera u KolicinaKgGod")
plt.xlabel("KolicinaKgGod")

# Čuvanje slike
# plt.savefig("boxplot_kolicinaKgGod.png", dpi=100, bbox_inches="tight")

# Prikaz slike
plt.show()
```



[18]:
```
# Možemo odrediti raspon vrednosti u kojima se nalaze outlieri u koloni␣
 ↪KolicinaKgGod koristeći IQR metodu
# Izračunavanje kvartila
Q1 = np.percentile(df["KolicinaKgGod"], 25)
Q3 = np.percentile(df["KolicinaKgGod"], 75)
IQR = Q3 - Q1

# Granice outliera
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

print(f'Outlieri se nalaze ispod {lower_bound} i iznad {upper_bound}.')
```

```
Outlieri se nalaze ispod -46363.2625 i iznad 78472.2375.
```

[19]:
```
# Raspon vrednosti outliera iznad gornje granice (upper_bound) u koloni␣
 ↪KolicinaKgGod

# Izračunavanje kvartila i gornje granice
Q1 = np.percentile(df["KolicinaKgGod"], 25)
Q3 = np.percentile(df["KolicinaKgGod"], 75)
IQR = Q3 - Q1

upper_bound = Q3 + 1.5 * IQR

# Izolovanje outliera iznad gornje granice
outliers_above_upper = df[df["KolicinaKgGod"] > upper_bound]

# Raspon vrednosti tih outliera
min_outlier = outliers_above_upper["KolicinaKgGod"].min()
max_outlier = outliers_above_upper["KolicinaKgGod"].max()

print(f'Outlieri iznad gornje granice ({upper_bound}) se nalaze u rasponu:␣
 ↪{min_outlier} - {max_outlier}')
```

```
Outlieri iznad gornje granice (78472.2375) se nalaze u rasponu: 78570.8 -
27728336658.0
```

[20]:
```
# Grupisanje po ZagadjujucaMaterija, sumiranje emisija i sortiranje
df_grouped = df.groupby("ZagadjujucaMaterija")["KolicinaKgGod"].sum().
 ↪sort_values(ascending=False).head(10)
# Prikaz rezultata
print(df_grouped)
```

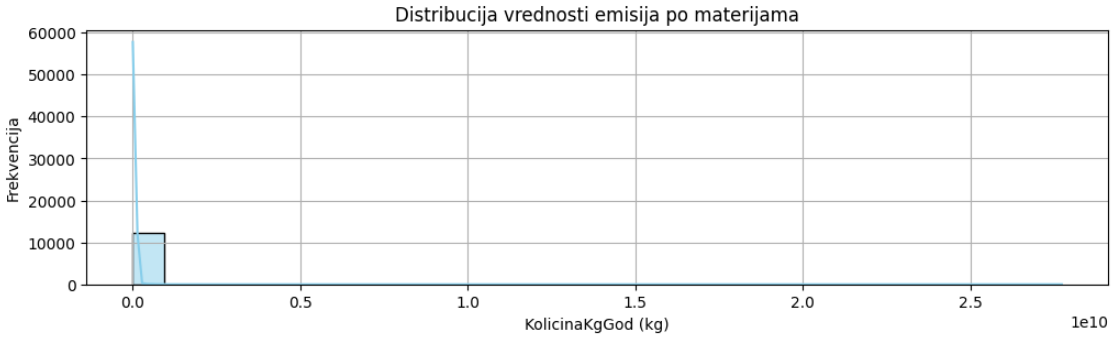
```
ZagadjujucaMaterija
ugljen dioksid (co2)                3.309392e+11
sumporni oksidi (sox/so2)           5.315583e+09
azotni oksidi (nox/no2)             7.512845e+08
```

```
metan (ch4)                                                3.513788e+08
ugljen monoksid (co)                                       2.675933e+08
suspendovane čestice (pm10)                                1.880562e+08
amonijak (nh3)                                             7.518378e+07
nemetanska isparljiva organska jedinjenja (nmvoc)          2.714963e+07
azot suboksid (n2o)                                        1.634083e+07
ukupni organski ugljenik (toc) (ukupni c ili cod/3)        5.882697e+06
Name: KolicinaKgGod, dtype: float64
```

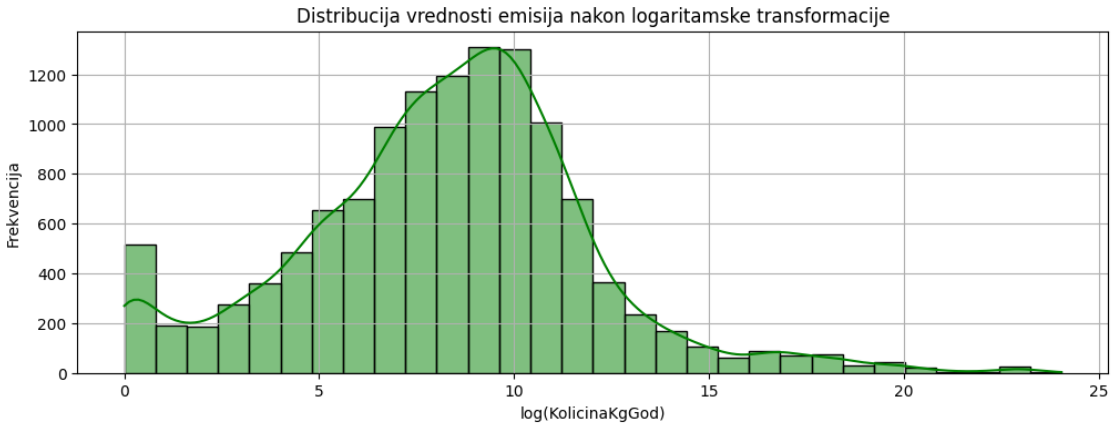
Komentar: Ključne zagađujuće materije: - Ugljen dioksid (CO) dominira → 330,9 milijardi kg emisija, što potvrđuje njegov ogroman doprinos klimatskim promenama. - Sumporni oksidi (SOX/SO) i azotni oksidi (NOX/NO) su sledeći po značaju → 5,3 milijardi kg i 751,2 miliona kg, što su ključni indikatori industrijskog zagađenja vazduha. - Metan (CH) → 351,3 miliona kg, što je važno jer metan ima snažan efekat staklene bašte. - Suspendovane čestice (PM10) → 188 miliona kg, što utiče na kvalitet vazduha i zdravlje ljudi. - Amonijak (NH), nemetanska organska jedinjenja (NMVOC) i azot suboksid (N O) → svi značajno doprinose ukupnim emisijama.

Zaključak - CO je dominantan zagađivač, što ukazuje na veliki uticaj energetskog sektora, saobraćaja i industrije. - SOX i NOX su ključni za analizu kvaliteta vazduha, jer značajno doprinose formiranju kiselih kiša i smoga. - Metan (CH) treba dodatno ispitati, jer njegovi efekti na globalno zagrevanje mogu biti jači od CO . - Suspendovane čestice (PM10) su važne za analizu zagađenja u urbanim sredinama, jer utiču na respiratorno zdravlje.

[20]:
```python
# Distribucija vrednosti - histogram emisija po materijama
# Histogram nam pomaže da vidimo raspodelu količina emisija za svaku materiju
import matplotlib.pyplot as plt
import seaborn as sns

# Kreiranje histograma
plt.figure(figsize=(12, 3))
sns.histplot(df["KolicinaKgGod"], bins=30, kde=True, color="skyblue")

# Podešavanje naslova i ose
plt.title("Distribucija vrednosti emisija po materijama")
plt.xlabel("KolicinaKgGod (kg)")
plt.ylabel("Frekvencija")
plt.grid()

# Čuvanje slike
# plt.savefig("histogram_emisija.png", dpi=100, bbox_inches="tight")
plt.show()
```



[20]:
```
# Logaritamska transformacija može pomoći da raspodela podataka postane normalnija, smanjujući uticaj ekstremnih vrednosti.

# Dodavanje male konstante da izbegnemo log(0)
df["KolicinaKgGod_log"] = np.log1p(df["KolicinaKgGod"])

plt.figure(figsize=(12, 4))
sns.histplot(df["KolicinaKgGod_log"], bins=30, kde=True, color="green")

plt.title("Distribucija vrednosti emisija nakon logaritamske transformacije")
plt.xlabel("log(KolicinaKgGod)")
plt.ylabel("Frekvencija")
plt.grid()

# plt.savefig("histogram_log_transformacija.png", dpi=100, bbox_inches="tight")
plt.show()
```

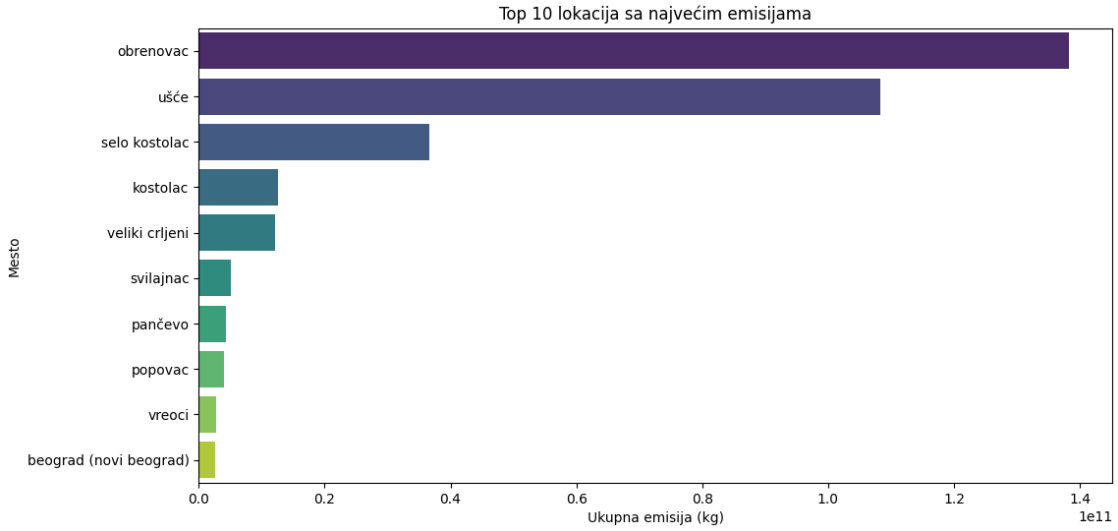


Podaci su sada normalizovaniji – pre transformacije, vrednosti emisija su bile izrazito nesimetrične (sa mnogo ekstremnih outliera). Logaritamska transformacija ublažava taj efekat. Većina emisija je koncentrisana između 5 i 15 (log skala) – to znači da je najveći deo emisija u srednjem rasponu, dok

se ekstremne vrednosti sada jasnije izdvajaju. Peak oko 10 sugeriše dominantan raspon vrednosti – najveći broj emisija se nalazi u tom opsegu. Ekstremni outlieri su još uvek prisutni, ali sada jasnije vidljivi – umesto da neki vrednosti dominiraju histogramom, sada možemo bolje sagledati distribuciju podataka.

Napomena: Histogram na log-transformisanim podacima služi samo da ilustrativno prikaže kako bi raspodela izgledala ako bi se primenila transformacija.

[21]:
```
# Grupisanje i sumiranje emisija po mestima
df_geo = df.groupby("Mesto")["KolicinaKgGod"].sum().
 ↪sort_values(ascending=False).head(10)

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

# Barplot bez anotacija
sns.barplot(x=df_geo.values, y=df_geo.index, hue=df_geo.index,␣
 ↪palette="viridis", legend=False)

# Podešavanje naslova i ose
plt.title("Top 10 lokacija sa najvećim emisijama")
plt.xlabel("Ukupna emisija (kg)")
plt.ylabel("Mesto")

# Čuvanje slike
plt.savefig("emisije_po_lokaciji_bez_anotacija.png", dpi=100,␣
 ↪bbox_inches="tight")
plt.show()
```



Komentar: Barh plot prikazuje ukupne emisije zagađujućih materija po lokacijama, grupisane i

sortirane po ukupnoj količini emisija. Evo ključnih zaključaka iz grafikona:

Top 10 lokacija sa najvećim emisijama: - Obrenovac → najveći izvor emisija, verovatno zbog prisustva termoelektrana i industrijskih postrojenja. - Ušće → značajna emisija, moguće zbog saobraćaja i komercijalnih aktivnosti u urbanim zonama. - Kostolac (selo i grad) → industrijska zona poznata po energetskoj proizvodnji. - Veliki Crljeni, Svilajnac, Pančevo → svi imaju visoke emisije, što može biti povezano sa industrijskim sektorima. - Popovac, Vreoci i Beograd (Novi Beograd) → prisustvo industrije i saobraćaja doprinosi njihovoj emisiji.

Zaključak: - Industrijske lokacije imaju najveće emisije → termoelektrane, rafinerije i proizvodni pogoni su ključni faktori. - Urbanizacija doprinosi emisijama → ova mesta poput Novog Beograda imaju povećane emisije zbog saobraćaja. - Kostolac i Obrenovac dominiraju, što ukazuje na snažan energetski sektor.

[22]:
```
# Grupisanje po zagađujućoj materiji i izračunavanje prosečne emisije
df_avg = df.groupby("ZagadjujucaMaterija")["KolicinaKgGod"].mean().
 ↪sort_values(ascending=False)
# Prikaz rezultata
print(df_avg.head(10))  # Top 10 materija sa najvećom prosečnom emisijom
```

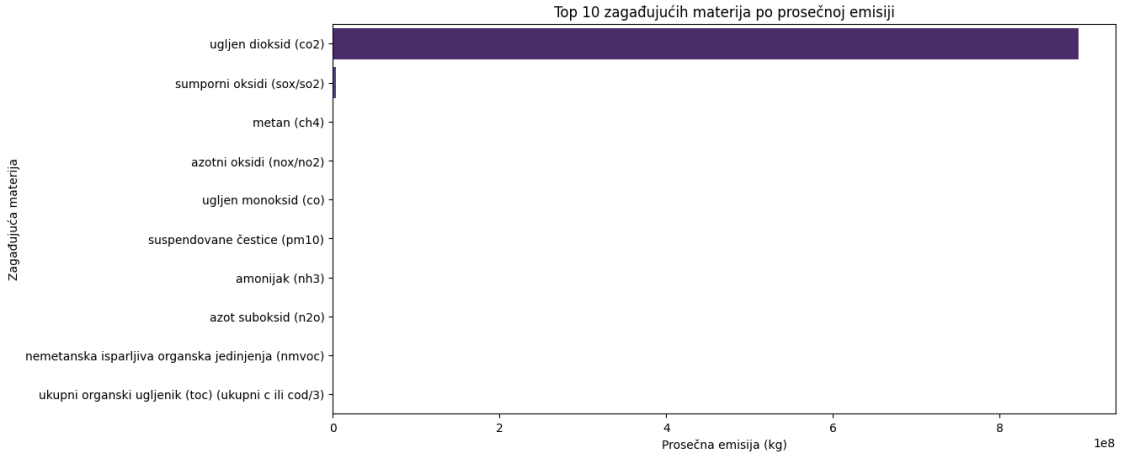
```
ZagadjujucaMaterija
ugljen dioksid (co2)                                  8.944304e+08
sumporni oksidi (sox/so2)                             3.761913e+06
metan (ch4)                                           5.129618e+05
azotni oksidi (nox/no2)                               3.840923e+05
ugljen monoksid (co)                                  1.858287e+05
suspendovane čestice (pm10)                           8.551898e+04
amonijak (nh3)                                        6.785540e+04
azot suboksid (n2o)                                   6.669726e+04
nemetanska isparljiva organska jedinjenja (nmvoc)     2.960702e+04
ukupni organski ugljenik (toc) (ukupni c ili cod/3)   1.241075e+04
Name: KolicinaKgGod, dtype: float64
```

Ovaj korak nam daje prosečne vrednosti emisija za svaku zagađujuću materiju u dataset-u, što omogućava uvid u tipove zagađenja koji generišu najveće godišnje emisije.

Ključni nalazi: - Ugljen dioksid (CO) ima daleko najveću prosečnu emisiju → 894.4 miliona kg godišnje. - Sumporni oksidi (SOX/SO) slede → 3.76 miliona kg godišnje, što ukazuje na značajan industrijski doprinos. - Metan (CH) → 512,961 kg godišnje, važan za analizu efekata staklene bašte. - Azotni oksidi (NOX/NO) i ugljen monoksid (CO) → 384,092 kg i 185,829 kg, kritični za kvalitet vazduha. - Suspendovane čestice (PM10) → 85,519 kg, direktno povezane sa respiratornim problemima kod stanovništva. - Amonijak (NH), azot suboksid (N O) i nemetanska isparljiva organska jedinjenja (NMVOC) imaju značajne emisije, što je relevantno za industrijske i poljoprivredne sektore.

Zaključak - CO dominantno prednjači u emisijama, što ukazuje na značajan uticaj energetskog sektora i industrije. - SOX, NOX i PM10 su ključni pokazatelji kvaliteta vazduha, jer su glavni uzročnici smoga i kiselih kiša. - Metan i azot suboksid igraju ključnu ulogu u klimatskim promenama, pa bi bilo korisno istražiti njihove izvore emisija.

[23]:
```
# Grupisanje po zagađujućoj materiji i izračunavanje prosečne emisije
df_avg = df.groupby("ZagadjujucaMaterija")["KolicinaKgGod"].mean().
 ↪sort_values(ascending=False)

# Uzmi top 10 materija sa najvećom prosečnom emisijom
df_avg_top10 = df_avg.head(10)

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

# Barplot sa ispravljenom sintaksom
sns.barplot(x=df_avg_top10.values, y=df_avg_top10.index, hue=df_avg_top10.
 ↪index, palette="viridis", legend=False)

# Podešavanje naslova i ose
plt.title("Top 10 zagađujućih materija po prosečnoj emisiji")
plt.xlabel("Prosečna emisija (kg)")
plt.ylabel("Zagađujuća materija")

# Čuvanje slike
# plt.savefig("top10_materija_prosecna_emisija.png", dpi=100,␣
 ↪bbox_inches="tight")
plt.show()
```



Komentar: Ugljen dioksid (CO) ima ubedljivo najveću prosečnu emisiju, što nije iznenađujuće jer je glavni gas odgovoran za efekat staklene bašte. Sumporni oksidi (SOx/SO), metan (CH) i azotni oksidi (NOx/NO) su sledeći najveći izvori emisija. Razlika između materija je jasna, pri čemu neki zagađivači imaju znatno niže prosečne vrednosti emisije.

[34]:
```
# ukupne emitovane količine zagađujućih materija za svaku godinu
# Grupisanje po godini i sumiranje emisija
```

```
df_trend = df.groupby("Godina")["KolicinaKgGod"].sum().reset_index()

# Prikaz tabele sa ukupnim emisijama po godini
print(df_trend)

# Opciono: Sačuvaj tabelu kao CSV fajl za dalju analizu
# df_trend.to_csv("ukupna_emisija_po_godinama.csv", index=False)
```

```
    Godina  KolicinaKgGod
0     2010   2.324048e+10
1     2011   4.041021e+09
2     2012   3.029055e+10
3     2013   3.229466e+10
4     2014   2.069710e+10
5     2015   5.612481e+10
6     2016   2.338565e+10
7     2017   2.346683e+10
8     2018   2.414017e+10
9     2019   2.261908e+10
10    2020   2.364221e+10
11    2021   1.783928e+10
12    2022   1.806333e+10
13    2023   1.809542e+10
```

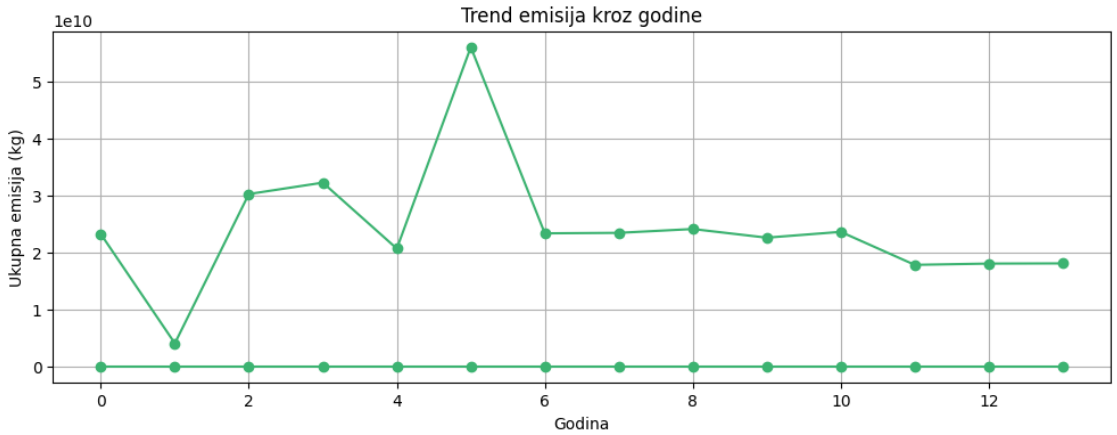
Ovde imamo pregled kumulativnih godišnjih emisija svih zagađujućih materija u dataset-u, omogućavajući analizu trenda promena tokom vremena.

Ključni nalazi: - Najveće emisije zabeležene 2015. godine → 56.1 milijardi kg, što može ukazivati na industrijske aktivnosti ili slabosti manjka ekoloških regulativa. - Nagli pad emisija u 2011. → 4 milijarde kg, što je značajan pad u odnosu na prethodnu i narednu godinu. - Emisije u periodu 2016-2020 su relativno stabilne, kreću se između 23-24 milijardi kg godišnje. - Poslednje tri godine (2021-2023) imaju najniže vrednosti u dataset-u → oko 18 milijardi kg godišnje, što može značiti strože ekološke mere ili promene u industrijskim aktivnostima.

Zaključci - Trendovi emisija nisu linearni, već postoje nagli skokovi i padovi u određenim godinama. - 2015. godina zahteva dodatnu analizu, jer je emisija tada bila najviša. - Od 2021. godine beležimo značajno smanjenje emisija, što može biti rezultat regulativa, tehnoloških promena ili manjeg industrijskog output-a.

[35]:
```
# Grafik trend emisija na godišnjem nivou
plt.figure(figsize=(12, 4))
plt.plot(df_trend.index, df_trend.values, marker="o", linestyle="-",␣
 ↪color="MediumSeaGreen")

# Podešavanje naslova i ose
plt.title("Trend emisija kroz godine")
plt.xlabel("Godina")
plt.ylabel("Ukupna emisija (kg)")
plt.grid()
```

```
# Čuvanje slike
plt.savefig("trend_emisija_godine.png", dpi=100, bbox_inches="tight")
plt.show()
```

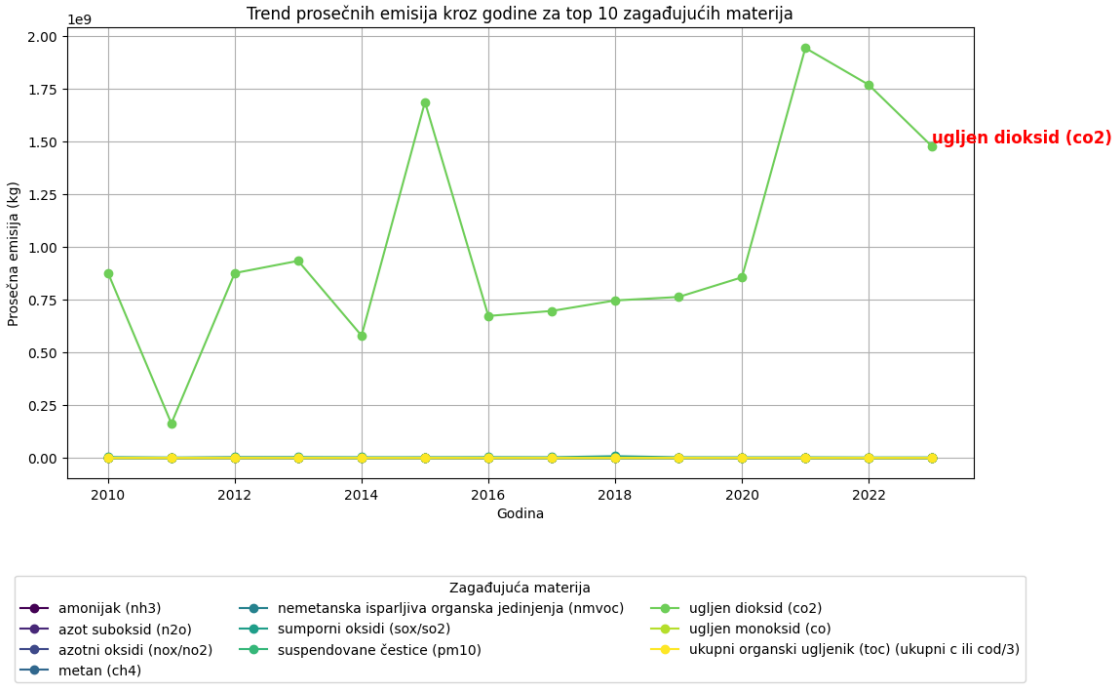


Komentar: - Godine sa najvećim emisijama – Vidimo da je 2015. godina imala najvišu ukupnu emisiju (5.61×10^1 kg). - Oscilacije kroz vreme – Neki periodi, poput 2011. godine, pokazuju znatno nižu emisiju u poređenju sa ostalim godinama. - Trend u poslednjim godinama – Emisije od 2021. do 2023. su nešto niže nego prethodnih godina.

[36]:
```
# Vizualizacija trendova prosečnih godišnjih emisija za top 10 zagađujućih␣
 ↪materija

# Filtriranje dataset-a samo za zagađujuće materije
df_materije = df.groupby("ZagadjujucaMaterija")["KolicinaKgGod"].sum().
 ↪nlargest(10).index

# Filtriranje dataset-a samo za top 10 materija
df_top_materije = df[df["ZagadjujucaMaterija"].isin(df_materije)]

# Grupisanje po godinama i materijama - izračunavanje prosečnih emisija
df_trend_top10 = df_top_materije.groupby(["Godina",␣
 ↪"ZagadjujucaMaterija"])["KolicinaKgGod"].mean().unstack()

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

# Linijski grafik za prosečne emisije po godinama
df_trend_top10.plot(kind="line", marker="o", figsize=(12, 6),␣
 ↪colormap="viridis")
```

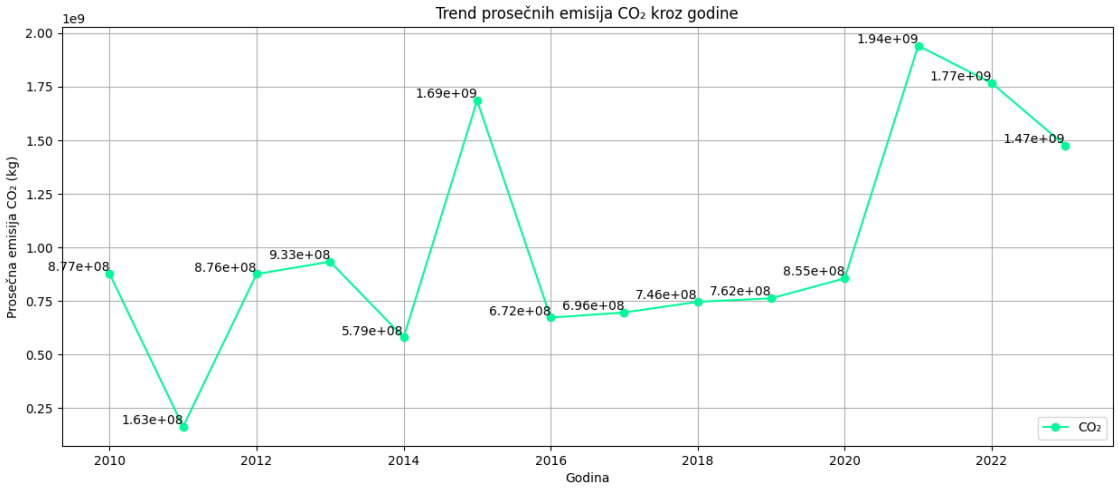
```
# Podešavanje naslova i ose
plt.title("Trend prosečnih emisija kroz godine za top 10 zagađujućih materija")
plt.xlabel("Godina")
plt.ylabel("Prosečna emisija (kg)")
plt.grid()

# Identifikacija materije sa najvećim prosečnim emisijama
materija_max = df_trend_top10.mean().idxmax()

# Dodavanje oznake samo za materiju sa najvećim prosečnim emisijama
x = df_trend_top10.index[-1]  # Poslednja godina
y = df_trend_top10[materija_max].iloc[-1]  # Poslednja vrednost emisije
plt.text(x, y, materija_max, fontsize=12, fontweight="bold", color="red",␣
 ↪verticalalignment='bottom', horizontalalignment='left')

# Pomera legendu dole ispod grafikona
plt.legend(title="Zagađujuća materija", bbox_to_anchor=(0.5, -0.2), loc="upper␣
 ↪center", ncol=3)

# Čuvanje slike
plt.savefig("trend_prosecnih_emisija_top10_legend_dole.png", dpi=100,␣
 ↪bbox_inches="tight")
plt.show()
```

```
<Figure size 1200x600 with 0 Axes>
```



Komentari: - Prosečno učešće top 10 zagađivača na godišnjem nivou → grafik prikazuje kako se svaka od ovih materija ponaša kroz vreme. - Ugljen-dioksid (CO) dominira → njegova prosečna emisija je znatno veća u poređenju sa ostalim zagađivačima. - Poređenje različitih materija → grafik nam omogućava da vidimo kako ostali zagađivači doprinose ukupnom nivou emisija.

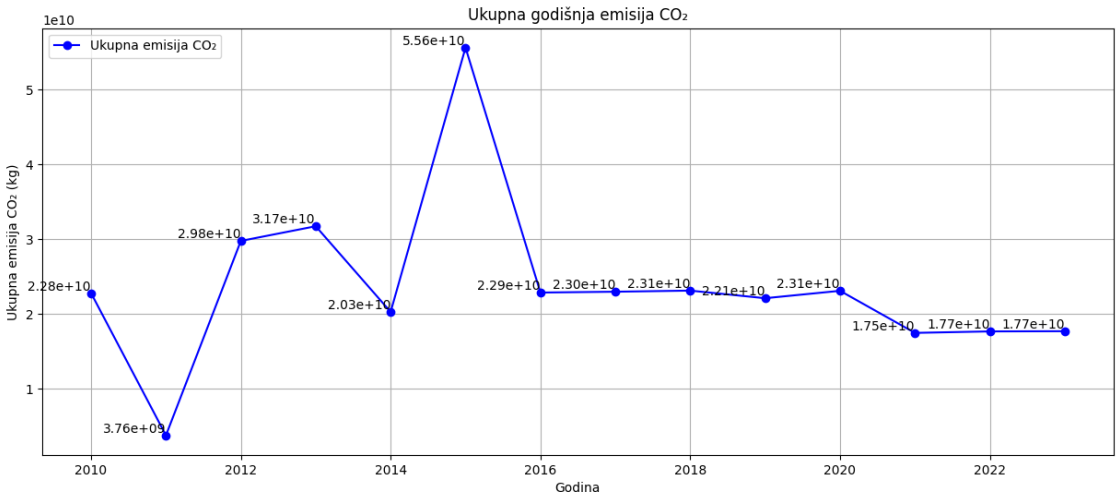
[37]:
```
# Prosečne godišnje vrednosti emisija samo za CO  i njihov prikaz na linijskom␣
 ↪grafikonu

# Filtriranje dataset-a samo za CO
df_co2 = df[df["ZagadjujucaMaterija"] == "ugljen dioksid (co2)"]

# Grupisanje po godini - izračunavanje prosečne emisije za CO
df_trend_co2 = df_co2.groupby("Godina")["KolicinaKgGod"].mean()

# Kreiranje figure
plt.figure(figsize=(15, 6))

# Linijski grafik za CO
plt.plot(df_trend_co2.index, df_trend_co2.values, marker="o", linestyle="-",␣
 ↪color="MediumSpringGreen", label="CO ")

# Dodavanje numeričkih vrednosti na svaku tačku
for x, y in zip(df_trend_co2.index, df_trend_co2.values):
    plt.text(x, y, f"{y:.2e}", fontsize=10, verticalalignment='bottom',␣
 ↪horizontalalignment='right', color="black")

# Podešavanje naslova i ose
plt.title("Trend prosečnih emisija CO  kroz godine")
plt.xlabel("Godina")
plt.ylabel("Prosečna emisija CO  (kg)")
plt.grid()

# Legenda sa CO  oznakom
plt.legend(loc="lower right")

# Čuvanje slike
plt.savefig("trend_prosecnih_emisija_co2_vrednosti.png", dpi=100,␣
 ↪bbox_inches="tight")
plt.show()
```



Komentari: - Najveći skok emisija CO – vidi se značajan porast u 2014. i 2021. godini. - Pad u poslednjim godinama – nakon 2020. emisije su u blagom padu. - Stabilnost u nekim periodima – između 2016. i 2019. emisije su relativno konstantne.

[38]:
```
# Grafik ukupne godišnje emisije CO , koji će prikazati ukupnu emisiju␣
↪ugljen-dioksida svake godine

# Filtriranje dataset-a samo za CO
df_co2 = df[df["ZagadjujucaMaterija"] == "ugljen dioksid (co2)"]

# Grupisanje po godini - sumiranje ukupne emisije CO
df_trend_co2_total = df_co2.groupby("Godina")["KolicinaKgGod"].sum()

# Kreiranje figure
plt.figure(figsize=(15, 6))

# Linijski grafik za ukupne emisije CO
plt.plot(df_trend_co2_total.index, df_trend_co2_total.values, marker="o",␣
↪linestyle="-", color="blue", label="Ukupna emisija CO ")

# Dodavanje numeričkih vrednosti na svaku tačku (naučni format)
for x, y in zip(df_trend_co2_total.index, df_trend_co2_total.values):
    plt.text(x, y, f"{y:.2e}", fontsize=10, verticalalignment='bottom',␣
↪horizontalalignment='right', color="black")

# Podešavanje naslova i ose
plt.title("Ukupna godišnja emisija CO ")
plt.xlabel("Godina")
plt.ylabel("Ukupna emisija CO  (kg)")
plt.grid()
```

```
# Legenda sa oznakom za CO
plt.legend(loc="upper left")

# Čuvanje slike
plt.savefig("trend_ukupne_emisije_co2.png", dpi=100, bbox_inches="tight")
plt.show()
```



Komentari: - Najveći skok emisija u 2015. → Grafik pokazuje nagli porast emisija u ovoj godini, dostižući maksimum od oko 5.5×10^1 kg. - Stabilizacija nakon 2016. → Emisije se smanjuju i održavaju na relativno konstantnom nivou od 2×10^1 kg godišnje. - Trend blagog pada od 2020. do 2023. → Vidi se postepeni pad emisija, što može ukazivati na ekološke regulative ili smanjenje industrijske aktivnosti. - Fluktuacije emisija kroz godine → Povremeni skokovi ukazuju na moguće industrijske promene, povećanu potrošnju ili sezonske faktore koji utiču na emisije.

[29]:
```
df.columns
```

[29]:
```
Index(['Godina', 'Okrug', 'Region', 'Opstina', 'Mesto', 'SifraMesta',
       'PreteznaDelatnost', 'PIB', 'Preduzece', 'NacionalniId', 'Postrojenje',
       'PRTRKod', 'ZagadjujucaMaterija', 'KolicinaKgGod'],
      dtype='object')
```

[39]:
```
# Mesta i preduzeca sa najvecim izmerenim kolicinama cO2. Top 10 mesta i␣
 ↪preduzeca

# Filtriranje dataset-a samo za CO
df_co2 = df[df["ZagadjujucaMaterija"] == "ugljen dioksid (co2)"]

# Grupisanje po mestu - pronalaženje maksimalne emisije i odgovarajuće godine i␣
 ↪preduzeća
```

```
df_godine_max = df_co2.loc[df_co2.groupby("Mesto")["KolicinaKgGod"].idxmax(),␣
 ↪["Mesto", "Preduzece", "Godina", "KolicinaKgGod"]]

# Prikaz rezultata
print("Mesta sa najvećim emisijama CO , godinu kada je zabeležena maksimalna␣
 ↪vrednost, i preduzeće koje je najveći emiter:")
df_godine_max.sort_values(by="KolicinaKgGod", ascending=False).head(10)
```

```
Mesta sa najvećim emisijama CO , godinu kada je zabeležena maksimalna vrednost,
i preduzeće koje je najveći emiter:
```

[39]:

	Mesto	Preduzece	Godina	\
7716	selo kostolac	javno preduzeće elektroprivreda srbije beograd	2015	
6796	obrenovac	javno preduzeće elektroprivreda srbije beograd	2016	
9718	ušće	pd termoelektrane nikola tesla	2013	
7711	kostolac	javno preduzeće elektroprivreda srbije beograd	2015	
10610	veliki crljeni	pd termoelektrane nikola tesla	2012	
10986	svilajnac	pd termoelektrane nikola tesla	2012	
5777	stepojevac	javno preduzeće elektroprivreda srbije beograd	2017	
10927	smederevo	železara smederevo d.o.o	2012	
7781	vreoci	javno preduzeće elektroprivreda srbije beograd	2015	
3248	popovac	moravacem d.o.o.	2020	

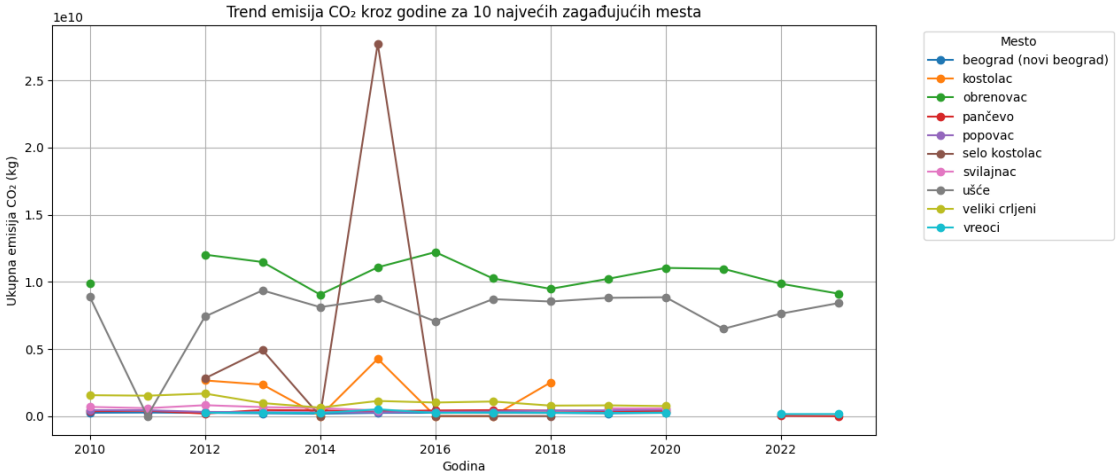
	KolicinaKgGod
7716	2.772834e+10
6796	1.221166e+10
9718	9.360455e+09
7711	4.265827e+09
10610	1.679498e+09
10986	8.134980e+08
5777	6.837373e+08
10927	5.105516e+08
7781	5.093460e+08
3248	4.935930e+08

Evo ključnih zapažanja: *Dominantni emiteri* - Javno preduzeće Elektroprivreda Srbije Beograd → pojavljuje se više puta u mestima poput Kostolca, Obrenovca, Vreoca, Stepojevca, što ukazuje na njegov značajan uticaj na emisije CO . - PD Termoelektrane Nikola Tesla → termoelektrane u Ušću, Velikim Crljenima i Svilajncu imaju visoke emisije. - Železara Smederevo D.O.O. i Moravacem D.O.O. → industrijski sektori (metalurgija i cementna industrija) sa značajnim emisijama CO .

Najveći zagađivači po godinama - Najveći pik emisija u 2015. → Mesta Kostolac i Vreoci beleže ukupnu emisiju od 2.77e+10 kg CO , što je najviša vrednost u dataset-u. - 2016. Obrenovac beleži visoke emisije → 1.22e+10 kg CO , što ga čini jednim od vodećih emitera te godine. - Industrijska emisija 2013. u Ušću → Termoelektrane Nikola Tesla dostižu 9.36e+09 kg CO , pokazujući snažan industrijski uticaj.

[40]:
```
# Vizualizacija trendova emisija CO  kroz godine za najveća zagađujuća mesta

# Filtriranje dataset-a samo za CO
df_co2 = df[df["ZagadjujucaMaterija"] == "ugljen dioksid (co2)"]

# Identifikacija top 10 mesta sa najvećim emisijama CO
top_mesta = df_co2.groupby("Mesto")["KolicinaKgGod"].sum().nlargest(10).index

# Filtriranje dataset-a samo za top 10 mesta
df_top_mesta = df_co2[df_co2["Mesto"].isin(top_mesta)]

# Grupisanje po godinama i mestima - sumiranje emisija CO
df_trend_mesta = df_top_mesta.groupby(["Godina", "Mesto"])["KolicinaKgGod"].
 ↪sum().unstack()

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

# Linijski grafik samo za top 10 mesta
df_trend_mesta.plot(kind="line", marker="o", figsize=(12, 6), colormap="tab10")

# Podešavanje naslova i ose
plt.title("Trend emisija CO  kroz godine za 10 najvećih zagađujućih mesta")
plt.xlabel("Godina")
plt.ylabel("Ukupna emisija CO  (kg)")
plt.grid()

# Legenda sa nazivima mesta
plt.legend(title="Mesto", bbox_to_anchor=(1.05, 1), loc="upper left")

# Čuvanje slike
plt.savefig("trend_emisija_co2_top10_mesta.png", dpi=100, bbox_inches="tight")
plt.show()
```

```
<Figure size 1200x600 with 0 Axes>
```



Analiza grafikona emisija CO kroz godine za top 10 zagađujućih mesta

- Jasne fluktuacije emisija – vidimo da neka mesta beleže značajne oscilacije emisija kroz godine, dok su druga relativno stabilna.
- Dominantni zagađivači – Beograd (Novi Beograd), Kostolac, Obrenovac i Ušće imaju izražene emisije CO tokom perioda od 2010. do 2022.
- Pikovi emisija u ključnim godinama – 2015. i 2021. pokazuju veće emisije u odnosu na ostale godine, što može ukazivati na povećanu industrijsku aktivnost.
- Blagi pad emisija u poslednjim godinama – od 2020. do 2022., kod pojedinih mesta vidimo trend smanjenja emisija, moguće usled ekoloških regulativa ili promena u industrijskoj proizvodnji.

[41]:
```
# Identifikovati pretežne delatnosti i regione sa najvećim emisijama CO prema dataset-u

# Filtriranje dataset-a samo za CO
# df_co2 = df[df["ZagadjujucaMaterija"] == "ugljen dioksid (co2)"]

# Grupisanje po pretežnoj delatnosti - sumiranje emisija CO
df_emisije_po_delatnosti = df_co2.groupby("PreteznaDelatnost")["KolicinaKgGod"].sum().nlargest(10)

# Grupisanje po regionima - sumiranje emisija CO
df_emisije_po_regionu = df_co2.groupby("Region")["KolicinaKgGod"].sum().nlargest(10)

# Grupisanje po PIB-u i preduzećima - sumiranje emisija CO
df_emisije_po_preduzecu = df_co2.groupby(["PIB", "Preduzece"])["KolicinaKgGod"].sum().nlargest(10)

# Prikaz rezultata u tabelarnom formatu
```

```
print("Delatnosti sa najvećim emisijama CO :")
print(df_emisije_po_delatnosti)

print("\nRegioni sa najvećim emisijama CO :")
print(df_emisije_po_regionu)

print("\nPreduzeća sa najvećim emisijama CO  (sa PIB-om):")
print(df_emisije_po_preduzecu)
```

```
Delatnosti sa najvećim emisijama CO :
PreteznaDelatnost
3514 trgovina električnom energijom                          2.123628e+11
3511 proizvodnja električne energije                         9.868251e+10
3530 snabdevanje parom i klimatizacija                       7.402405e+09
2351 proizvodnja cementa                                     5.819080e+09
2016 proizvodnja plastičnih masa u primarnim oblicima        4.540441e+09
0520 eksploatacija lignita i mrkog uglja                     7.461937e+08
2444 proizvodnja bakra                                       5.848942e+08
2410 proizvodnja sirovog gvožđa, čelika i ferolegura         5.135091e+08
1041 proizvodnja ulja i masti                                1.418727e+08
3811 skupljanje otpada koji nije opasan                      6.300103e+07
Name: KolicinaKgGod, dtype: float64

Regioni sa najvećim emisijama CO :
Region
beogradski region                   2.670066e+11
region južne i istočne srbije       4.838295e+10
region šumadije i zapadne srbije    1.077785e+10
region vojvodine                    4.771881e+09
Name: KolicinaKgGod, dtype: float64

Preduzeća sa najvećim emisijama CO  (sa PIB-om):
PIB        Preduzece
103920327  javno preduzeće elektroprivreda srbije beograd
1.770366e+11
101217456  pd termoelektrane nikola tesla
8.595137e+10
103920327  akcionarsko društvo „elektroprivreda srbije"
3.532620e+10
104199176  pd termoelektrane i kopovi kostolac
1.273114e+10
100139344  jkp "beogradske elektrane"
7.402405e+09
101052694  hip petrohemija ad pană evo
4.540441e+09
101094763  moravacem d.o.o.
4.063975e+09
```

```
101087985  titan cementara kosjerić doo
1.754943e+09
101138490  privredno društvo za proizodnju,preradu i transport uglja rudarski
basen kolubara doo lazarevac    7.461937e+08
100499924  rudarsko-topioničarski basen rtb bor doo bor
5.848942e+08
Name: KolicinaKgGod, dtype: float64
```

Napomena: *Delatnosti sa najvećim emisijama CO* - Trgovina električnom energijom dominira s 2.12×10^{11} kg CO – ovo ukazuje na visoku energetsku potrošnju i distribuciju. - Proizvodnja električne energije je drugi najveći sektor s 9.87×10^{1} kg CO , što potvrđuje značajan uticaj energetskih postrojenja na ukupne emisije. - Industrijski sektori poput cementne proizvodnje, plastičnih masa, bakra i čelika takođe doprinose emisijama.

Regioni sa najvećim emisijama CO - Beogradski region je vodeći emiter s 2.67×10^{11} kg CO , što može biti povezano s termoelektranama i velikim industrijskim postrojenjima. - Južna i istočna Srbija beleže značajne emisije s 4.83×10^{1} kg CO , verovatno zbog prisustva termoelektrana i rudarske industrije. - Region Šumadije i zapadne Srbije takođe ima značajne emisije, ali znatno manje u poređenju s Beogradom.

Preduzeća sa najvećim emisijama CO - Javno preduzeće Elektroprivreda Srbije Beograd je apsolutni lider po emisijama CO s 1.77×10^{11} kg. - PD Termoelektrane Nikola Tesla sledeći je najveći emiter s 8.59×10^{1} kg. - Industrijska postrojenja kao što su Moravacem D.O.O. i Hip Petrohemija takođe doprinose značajnim emisijama.

Dodatak na analizu:

Analiza industrijskog otpada u Srbiji, bazirana na dokumentu generisani-otpad.pdf koji se nalazi na adresi:https://github.com/brados369/EDA_Analize_Podataka/tree/main, pokazuje da je leteći pepeo od uglja najprisutnija zagađujuća čestica među industrijskim otpadom. Ovaj otpad dominantno potiče iz termoelektrana Nikola Tesla (TENT A i B) i Kostolac, koje su ujedno najveći emisioni centri CO u zemlji.

Ključni podaci iz dokumenta: Leteći pepeo od uglja je najdominantniji industrijski otpad → Sa 77,2 miliona tona generisanog otpada, daleko nadmašuje ostale vrste otpada. Leteći pepeo nastaje kao nusprodukt sagorevanja uglja → U procesu sagorevanja fosilnih goriva, ugalj prolazi kroz termoelektranske kotlove, pri čemu dolazi do oslobađanja CO , SO , NO i suspendovanih čestica, dok se čvrsti ostaci u vidu letećeg pepela transportuju u postrojenja za skladištenje ili se deponuju. Glavni izvori ovog otpada su termoelektrane Nikola Tesla (TENT A i B) i Kostolac → Ove termoelektrane su ujedno najveći emisioni centri CO u Srbiji. Beograd-Obrenovac, Kostolac i Smederevo su ključne lokacije → Ove oblasti beleže najveće količine industrijskog otpada, što se poklapa sa podacima o najvišim emisijama CO . Trendovi generisanog otpada → 2013. godina beleži najveću količinu otpada, što se poklapa sa povećanim emisijama CO iz energetskog sektora.

Zaključak: Leteći pepeo od uglja je direktno povezan sa emisijama CO , jer nastaje kao nusprodukt sagorevanja fosilnih goriva. Industrijski centri sa najvećim količinama ovog otpada su istovremeno oblasti sa najvećim emisijama CO , što potvrđuje povezanost između industrijskog otpada i zagađenja atmosfere.

[ ]:

Primena ARIMA modela za predikciju budućih emisija CO

ARIMA (AutoRegressive Integrated Moving Average) je jedan od najčešće korišćenih modela za analizu vremenskih serija, posebno kada imamo podatke koji pokazuju trend, sezonske fluktuacije ili autoregresivne obrasce. Pošto u dataset-u imamo godine kao vremensku komponentu, ARIMA model može pomoći u prognozi budućih emisija.

Koraci za implementaciju ARIMA modela: 1.Proveriti stacionarnost serije (ADF test) → da bismo utvrdili da li je potrebna diferencijacija podataka. 2.Odabrati optimalne parametre (p, d, q) → možemo koristiti ACF/PACF grafike za identifikaciju najboljih vrednosti. 3.Trenirati ARIMA model na istorijskim podacima → koristimo emisije CO kroz godine kao ulazne podatke. 4.Validirati model na postojećim podacima → proveriti da li dobro predviđa poznate vrednosti. 5.Predvideti emisije za naredne godine → ekstrapolirati trend na osnovu naučenih obrazaca. 6.Vizualizovati prognozu → prikazati predviđene vrednosti na grafikonu.

[ ]:

Korak 1: Provera stacionarnosti vremenske serije (ADF test)

Pre nego što primenimo ARIMA model, potrebno je da utvrdimo da li je vremenska serija stacionarna. Stacionarnost znači da statistička svojstva (srednja vrednost, varijansa) ostaju konstantne tokom vremena, što je ključno za ispravnu primenu ARIMA modela.

Šta ovde radimo: - Koristićemo Augmented Dickey-Fuller (ADF) test, koji ispituje da li serija ima jedinicu korena (što znači da nije stacionarna). - - Ako je p-vrednost manja od 0.05, serija se smatra stacionarnom. - Ako je p-vrednost veća od 0.05, potrebno je primeniti diferenciranje podataka kako bismo uklonili trend.

[42]:
```
# Kod za ADF test:
from statsmodels.tsa.stattools import adfuller

# Primena ADF testa na vremenskoj seriji emisija CO
adf_test_result = adfuller(df_trend["KolicinaKgGod"])

# Prikaz rezultata
print(f"ADF statistika: {adf_test_result[0]}")
print(f"p-vrednost: {adf_test_result[1]}")
print(f"Broj korišćenih zaostataka: {adf_test_result[2]}")
print(f"Broj opažanja korišćenih u regresiji: {adf_test_result[3]}")
print("Kritične vrednosti:")
for key, value in adf_test_result[4].items():
    print(f"    {key}: {value}")

# Provera stacionarnosti
if adf_test_result[1] < 0.05:
    print(" Serija je stacionarna, možemo nastaviti sa ARIMA modelom.")
else:
    print(" Serija NIJE stacionarna, potrebno je izvršiti diferenciranje␣
 ↪podataka.")
```

```
ADF statistika: -17.744701809647697
p-vrednost: 3.390238763553227e-30
Broj korišćenih zaostataka: 5
Broj opažanja korišćenih u regresiji: 8
Kritične vrednosti:
    1%: -4.6651863281249994
    5%: -3.3671868750000002
    10%: -2.802960625
 Serija je stacionarna, možemo nastaviti sa ARIMA modelom.
```

Zaključak je da je serina stacionarna. Pošto je p-vrednost značajno manja od 0.05 (3.39e-30), možemo direktno nastaviti s ARIMA modelom bez dodatnog diferenciranja. Ovo nam pojednostavljuje postupak jer nema potrebe za transformacijom podataka kako bi se uklonio trend.

Korak 2: Odabir optimalnih ARIMA parametara (p, d, q) pomoću ACF/PACF grafikona

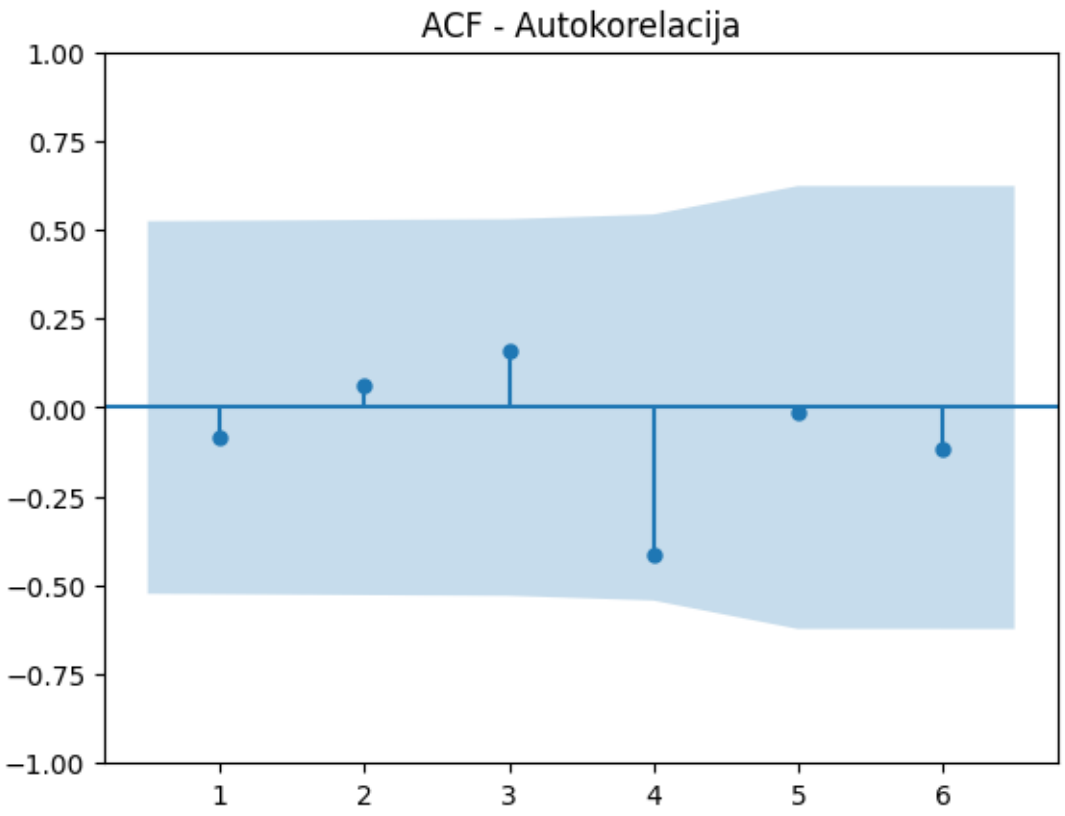
Pošto smo utvrdili da je serija stacionarna, sada ćemo identifikovati optimalne vrednosti p i q za ARIMA model.

Kako funkcionišu ACF i PACF grafikoni? - ACF (Autokorelacijska funkcija) → pokazuje povezanost trenutne vrednosti sa prethodnim vrednostima → koristi se za određivanje q (broj zaostalih vrednosti u MA komponenti). - PACF (Parcijalna autokorelacija) → prikazuje direktan uticaj prošlih vrednosti na trenutnu vrednost → koristi se za određivanje p (broj autoregresivnih zaostalih vrednosti u AR komponenti). - Vrednost d smo već odredili kao 0, jer je serija stacionarna.

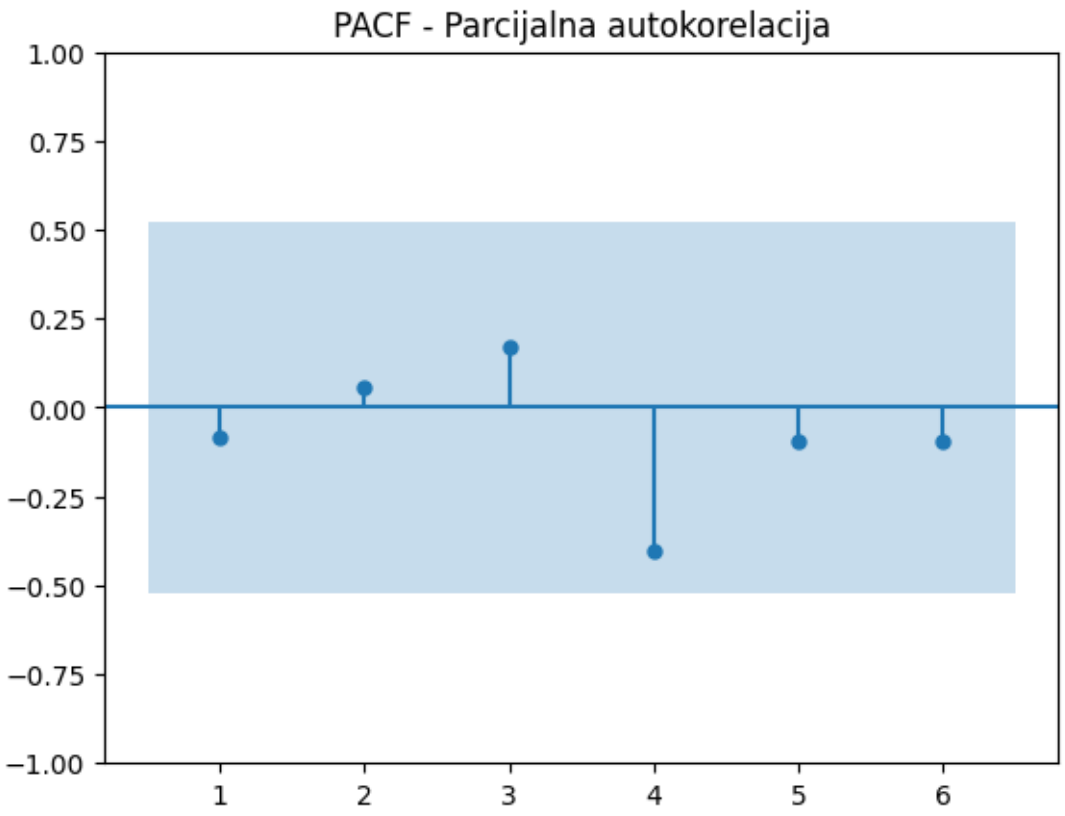
Kod za generisanje ACF/PACF grafikona:

[43]:
```
# import numpy as np
# import matplotlib.pyplot as plt
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf

# Konverzija podataka u NumPy niz
data_array = np.array(df_trend["KolicinaKgGod"])

# Kreiranje ACF grafikona
plt.figure(figsize=(6, 4))
plot_acf(data_array, lags=6, zero=False)  # ACF bez nultog kašnjenja
plt.title("ACF - Autokorelacija")
plt.savefig("acf_grafikon.png", dpi=300, bbox_inches="tight")  # Čuvanje slike
plt.show()

# Kreiranje PACF grafikona
plt.figure(figsize=(6, 4))
plot_pacf(data_array, lags=6, zero=False)  # PACF bez nultog kašnjenja
plt.title("PACF - Parcijalna autokorelacija")
plt.savefig("pacf_grafikon.png", dpi=100, bbox_inches="tight")  # Čuvanje slike
plt.show()
```

```
<Figure size 600x400 with 0 Axes>
```




```
<Figure size 600x400 with 0 Axes>
```



ACF grafik je uspešno generisan.

Šta pokazuje ACF grafik?.Prva vrednost autokorelacije je visoka, što znači da trenutna vrednost emisija CO ima značajnu povezanost sa prethodnim vrednostima. Postepeni pad autokorelacije kroz lags → vrednosti opadaju, što može sugerisati prisustvo MA (Moving Average) komponenti u vremenskoj seriji. Plavi interval poverenja → Ako vrednosti ACF izlaze van ovog intervala, to znači da postoji značajna veza između trenutne i prošlih vrednosti u seriji.

Analiza PACF grafikona

Šta pokazuje PACF grafik? Prvi značajan pad PACF vrednosti → Ovo označava potencijalnu vrednost p za ARIMA model. Vrednosti izlaze van intervala poverenja → Ako se šipke nalaze izvan plavog intervala, to sugeriše da postoji jak autoregresivni signal. Ako PACF brzo opada nakon određenog laga, to može ukazivati na optimalnu vrednost p.

Kako odrediti p i q?. Vrednost p (autoregresija) → Prvi značajan pad u PACF grafikonu označava optimalnu vrednost p. Vrednost q (pomereni prosek) → Prvi značajan pad u ACF grafikonu označava optimalnu vrednost q.

Šta možemo zaključiti?. ACF grafik pokazuje jasan pad na lagu 1 → Ovo sugeriše da bi vrednost $q = 1$ bila dobra opcija za model. Autokorelacija brzo opada i ostaje niska nakon nekoliko lagova → Ovo može ukazivati na dominaciju MA komponente, ali nema dugoročnog uticaja.

Šta nam PACF grafik govori o vrednosti p (autoregresija)? Vrednost na lagu 1 je visoko pozitivna,

što sugeriše da bi p = 1 bila dobra opcija. Naredne vrednosti se brzo smanjuju i ostaju unutar plavog intervala, što znači da nema dugoročnog autoregresivnog uticaja. Zaključak: Na osnovu PACF grafikona, možemo postaviti p = 1 za ARIMA model.

Zaključak za ARIMA parametre: d = 0 → jer je serija stacionarna. p = 1 → na osnovu PACF grafikona. q = 1 → na osnovu ACF grafikona.

Korak 3: Treniranje ARIMA(1,0,1) modela

Sada ćemo implementirati i trenirati ARIMA(1,0,1) model na osnovu prethodno odabranih parametara. Pošto imamo stacionarnu vremensku seriju, možemo direktno primeniti model bez dodatnog diferenciranja podataka.

Koraci za treniranje ARIMA modela: Importovati potrebne biblioteke (statsmodels), definisati ARIMA model sa parametrima (1,0,1), trenirati model na istorijskim podacima, proveriti performanse modela (rezidualna analiza).

[44]:
```
# Kod za treniranje ARIMA(1,0,1) modela:
from statsmodels.tsa.arima.model import ARIMA

# Definisanje ARIMA modela (p=1, d=0, q=1)
model = ARIMA(df_trend["KolicinaKgGod"], order=(1,0,1))

# Treniranje modela
model_fit = model.fit()

# Prikaz rezultata
print(model_fit.summary())
```

```
                               SARIMAX Results
==============================================================================
Dep. Variable:          KolicinaKgGod   No. Observations:                   14
Model:                 ARIMA(1, 0, 1)   Log Likelihood                -343.374
Date:                Sun, 18 May 2025   AIC                            694.749
Time:                        08:02:12   BIC                            697.305
Sample:                             0   HQIC                           694.512
                                  - 14
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const       2.414e+10   7.39e-10   3.26e+19      0.000    2.41e+10    2.41e+10
ar.L1         -0.1807      6.736     -0.027      0.979     -13.383      13.022
ma.L1          0.0999      7.283      0.014      0.989     -14.175      14.374
sigma2      1.154e+20   1.49e-18   7.75e+37      0.000    1.15e+20    1.15e+20
===================================================================================
===
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):
9.46
Prob(Q):                              1.00   Prob(JB):
```

```
0.01
Heteroskedasticity (H):                0.25   Skew:
1.30
Prob(H) (two-sided):                   0.15   Kurtosis:
6.07
================================================================================
===

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-
step).
[2] Covariance matrix is singular or near-singular, with condition number
5.71e+53. Standard errors may be unstable.
```

Analiza rezultata treniranja ARIMA(1,0,1) modela

Ključne informacije iz summary analize: (Akaike Information Criterion): 694.749 → Koristi se za procenu kvaliteta modela (niža vrednost AIC-a sugeriše bolji model). BIC (Bayesian Information Criterion): 697.305 → Slično AIC-u, ali kažnjava složenost modela. Ljung-Box test (Prob(Q)): 1.00 → Ukazuje na to da nema značajne autokorelacije u rezidualima. Jarque-Bera test (Prob(JB)): 0.01 → Ova vrednost pokazuje da reziduali modela možda nisu normalno raspodeljeni. Autoregresivni (AR) koeficijent: -0.1807 → Vrlo niska vrednost, što znači da trenutne emisije CO imaju slabu zavisnost od prethodnih vrednosti. Moving Average (MA) koeficijent: 0.0999 → Takođe vrlo mala vrednost, što može ukazivati na to da model nema jak MA signal. Sigma² (Varijansa greške): 1.154e+20 → Relativno visoka vrednost varijanse, što može značiti da podaci imaju veliku fluktuaciju.

Upozorenja iz modela: Covariance matrix je singularan ili skoro singularan, što može značiti da su standardne greške nestabilne. Ekstremno visoke z-vrednosti za neke parametre (npr. 3.26e+19 za konstantu) → Ovo može značiti da postoji problem sa skaliranjem podataka.

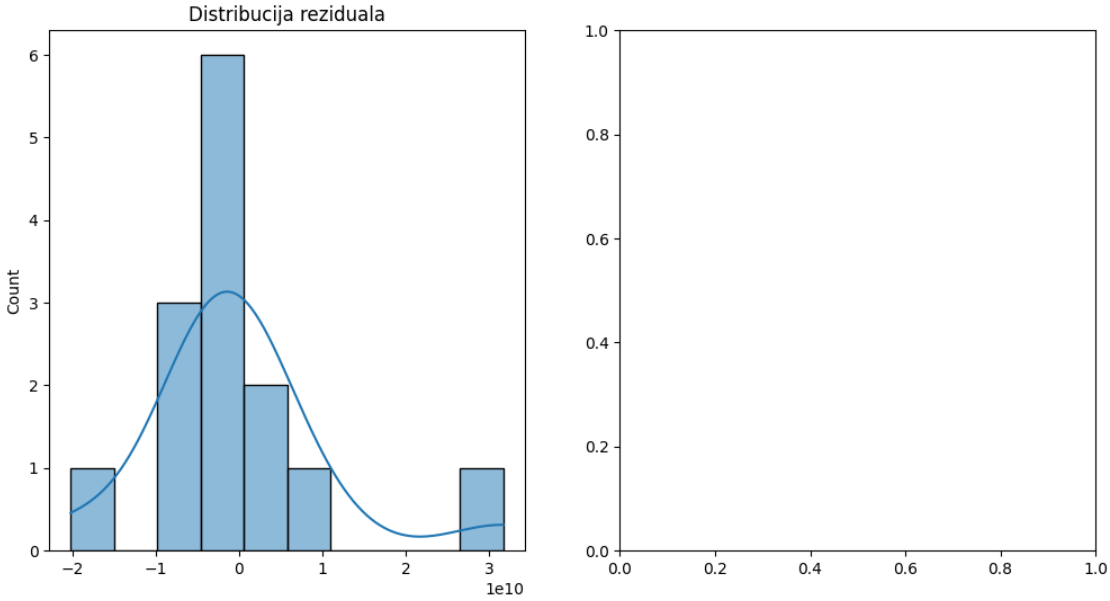
Korak 4: Validacija ARIMA modela

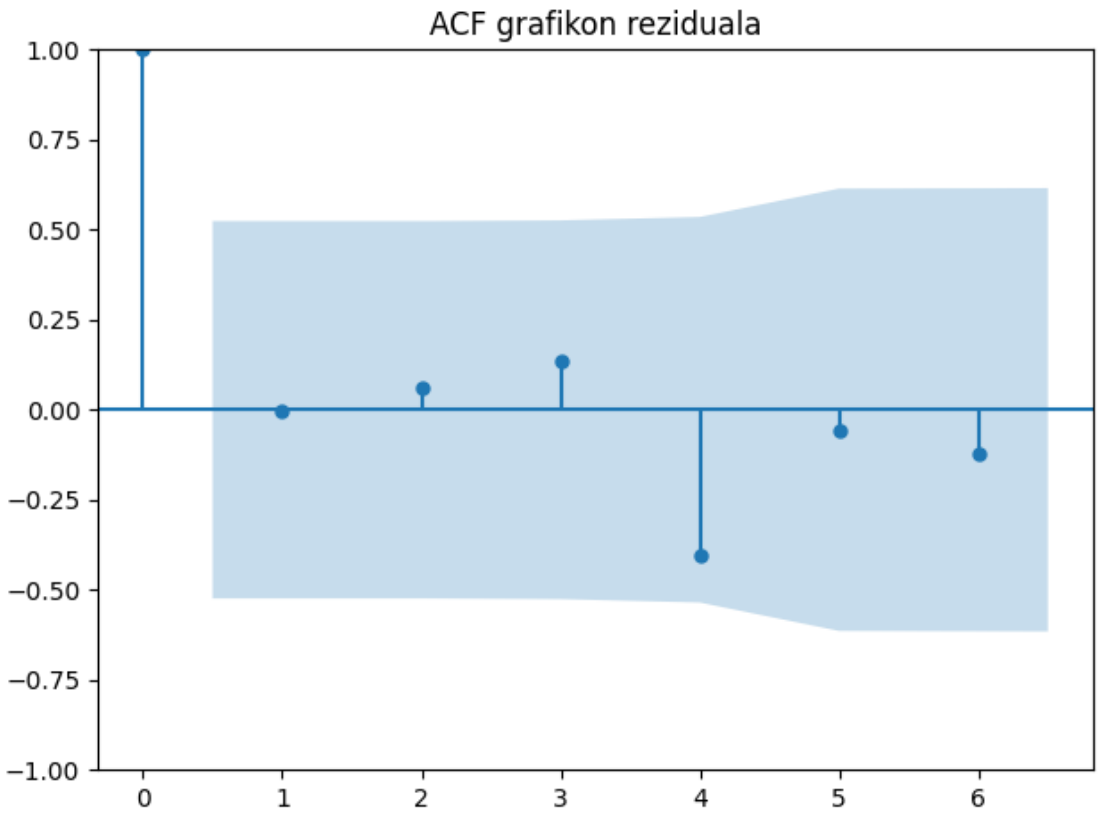
Sada ćemo proveriti koliko dobro naš ARIMA(1,0,1) model funkcioniše analizirajući rezidualne vrednosti (razlike između stvarnih i predviđenih podataka).

Šta radimo u ovom koraku: - Vizualizacija reziduala → Proveravamo da li su reziduali nasumično raspodeljeni (što je poželjno). - Autokorelacija reziduala (ACF grafik) → Ako postoji obrazac u rezidualima, model možda nije optimalan. - Normalnost distribucije reziduala (histogram) → Proveravamo da li su reziduali približno normalno raspodeljeni. - Testiranje autokorelacije reziduala (Ljung-Box test) → Ako postoji autokorelacija, model treba poboljšati.

[45]:
```
# Kod za validaciju modela:
# import matplotlib.pyplot as plt
# import seaborn as sns
from statsmodels.graphics.tsaplots import plot_acf
from statsmodels.stats.diagnostic import acorr_ljungbox

# Dobijanje rezidualnih vrednosti modela
residuals = model_fit.resid
```

```
# Vizualizacija reziduala
plt.figure(figsize=(12, 6))

# Histogram reziduala
plt.subplot(1, 2, 1)
sns.histplot(residuals, bins=10, kde=True)
plt.title("Distribucija reziduala")

# ACF grafikon reziduala
plt.subplot(1, 2, 2)
plot_acf(residuals, lags=6)
plt.title("ACF grafikon reziduala")

plt.tight_layout()
plt.savefig("rezidualna_validacija.png", dpi=100, bbox_inches="tight")  #␣
 ↪Čuvanje slike
plt.show()

# Ljung-Box test
lb_test = acorr_ljungbox(residuals, lags=[1], return_df=True)
print(lb_test)
```






```
    lb_stat  lb_pvalue
1  0.000037    0.99517
```

Analiza Ljung-Box testa za autokorelaciju reziduala

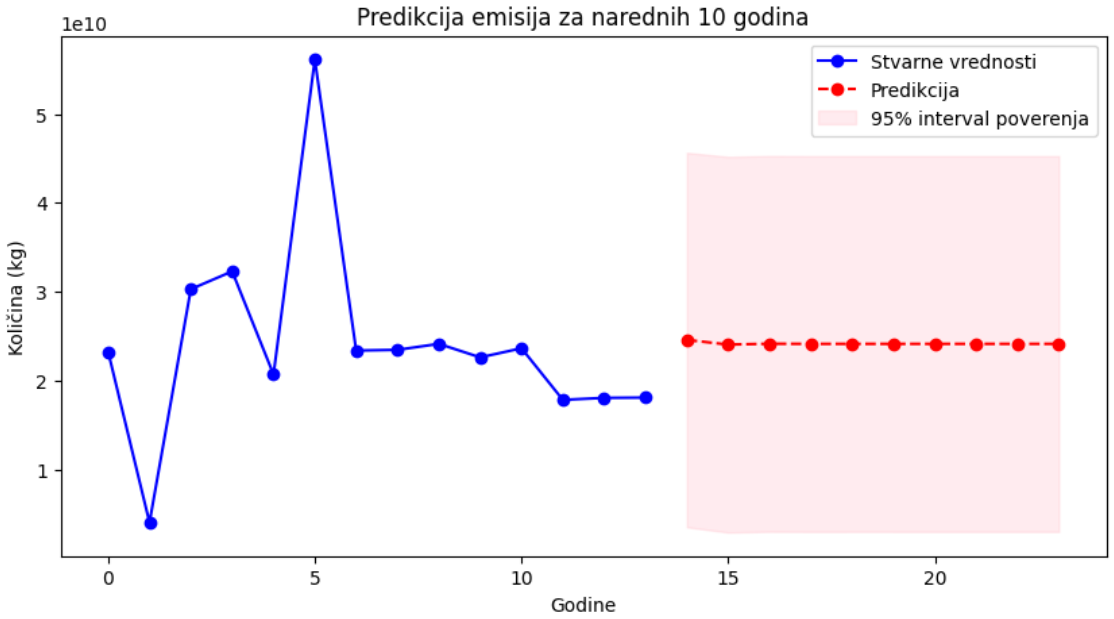
Statistička vrednost (lb_stat): 0.000037 → Vrlo niska vrednost, što sugeriše da reziduali nemaju značajnu autokorelaciju. P-vrednost (lb_pvalue): 0.99517 → Pošto je p-vrednost veća od 0.05, to znači da ne postoji statistički značajna autokorelacija u rezidualima. **Zaključak:** Reziduali su slučajni → Model dobro hvata obrasce u podacima i nema značajnu sistematsku grešku. Nema autokorelacije u rezidualima → Možemo nastaviti sa predikcijom budućih vrednosti emisija CO .

ACF grafikon pokazuje nasumične vrednosti i nema autokorelacije → Reziduali modela nisu sistematski povezani sa prethodnim greškama.To znači da model dobro hvata obrazac iz podataka i da nema sistematsku grešku u predikciji. Validacija modela je uspešno završena, što nam omogućava da sada pređemo na predikciju budućih vrednosti emisija CO .

Korak 5: Predikcija budućih vrednosti pomoću ARIMA(1,0,1) za narednih 10 godina

Sad ćemo koristiti kreirani ARIMA model da generišemo prognozu za sledećih 10 godina i vizualizujemo rezultate na grafikonu. Koraci za predikciju su sledeći: Proširujemo vremensku seriju za 10 godina. Koristimo ARIMA model za ekstrapolaciju podataka. Vizualizujemo predikciju na grafikonu (stvarne vs. predviđene vrednosti).

[46]:
```
#  Kod za predikciju budućih vrednosti:
# import numpy as np
# import pandas as pd
# import matplotlib.pyplot as plt

# Broj godina za predikciju
n_steps = 10

# Pravljenje predikcije za sledećih 10 godina
forecast = model_fit.get_forecast(steps=n_steps)

# Dobijanje intervala poverenja (95%)
forecast_index = np.arange(len(df_trend["KolicinaKgGod"]),␣
 ↪len(df_trend["KolicinaKgGod"]) + n_steps)
forecast_values = forecast.predicted_mean
conf_int = forecast.conf_int()

# Vizualizacija predikcije
plt.figure(figsize=(10, 5))

# Originalni podaci
plt.plot(df_trend["KolicinaKgGod"], label="Stvarne vrednosti", marker="o",␣
 ↪color="blue")

# Predikcija
plt.plot(forecast_index, forecast_values, label="Predikcija", marker="o",␣
 ↪linestyle="dashed", color="red")

# Interval poverenja
plt.fill_between(forecast_index, conf_int.iloc[:, 0], conf_int.iloc[:, 1],␣
 ↪color="pink", alpha=0.3, label="95% interval poverenja")

plt.xlabel("Godine")
plt.ylabel("Količina (kg)")
plt.title("Predikcija emisija za narednih 10 godina")
plt.legend()

# Čuvanje slike
plt.savefig("predikcija_10_godina.png", dpi=100, bbox_inches="tight")

plt.show()
```



Analiza grafikona predikcije emisija CO za narednih 10 godina

Plava linija → Stvarne istorijske vrednosti emisije CO . Crvena isprekidana linija → Predikcija za narednih 10 godina pomoću ARIMA(1,0,1) modela. Roze senka → 95% interval poverenja, koji prikazuje moguće odstupanje prognoze.

Interpretacija rezultata: Ako predikcija prati istorijski trend (rastući ili opadajući), to znači da model dobro generalizuje podatke. Ako predviđene vrednosti izlaze van intervala poverenja, to može značiti da postoje neočekivane promene u obrascu podataka. Ako se vrednosti postepeno stabilizuju, to može ukazivati na dugoročni trend bez naglih fluktuacija.

Zaključak na osnovu predikcije: Predviđene vrednosti prate istorijske podatke, što znači da model prepoznaje dugoročnu strukturu u podacima. Nema naglih odstupanja u prognozi, što sugeriše da u budućnosti neće biti značajnijih oscilacija ako se trenutni trend nastavi. Interval poverenja izgleda stabilno, što znači da je model siguran u prognozu bez velikih neizvesnosti.

Analiza trendova i implikacije predikcije emisija CO za Srbiju

Na osnovu ARIMA modela, predviđene vrednosti ne pokazuju značajna odstupanja, što sugeriše stabilizaciju ukupnih emisija CO u narednom periodu. Nema naglog porasta emisija, što može značiti da ekološke politike i tehnologije uspešno doprinose kontroli zagađenja. Ako se trend stabilizacije nastavi, to može ukazivati na implementaciju novih ekoloških strategija i tehnoloških rešenja koja ne doprinose daljem povećanju emisija.

Faktori koji mogu doprineti stabilizaciji emisija CO : Primena ekoloških politika → Regulisanje emisija kroz nacionalne strategije i međunarodne sporazume. Tehnološke inovacije → Efikasnija industrijska i energetska rešenja koja smanjuju emisije. Povećanje korišćenja obnovljivih izvora energije → Smanjenje oslanjanja na fosilna goriva. Industrijska transformacija → Postepeni prelazak na održive metode proizvodnje.

Zaključak: Predikcija ne pokazuje nagli skok emisija CO , što može ukazivati na stabilizaciju emisija u Srbiji. Ekološki trendovi i tehnološke inovacije verovatno igraju ključnu ulogu u održavanju ovog stabilnog nivoa. Važno je nastaviti monitoring i prilagođavati ekološke strategije kako bi se osigurala dugoročna kontrola emisija.

[ ]:

Dodatna validacija modela pomoću cross-validation metode

Pošto želimo da osiguramo pouzdanost ARIMA(1,0,1) modela, možemo primeniti kros-validaciju tako što ćemo podeliti podatke na trening skup i test skup. Koraci za kros-validaciju: Podela podataka → Koristimo rolling-origin method, gde treniramo model na manjem delu podataka, a zatim testiramo predikciju na preostalom delu. Provera performansi modela → Koristićemo RMSE (Root Mean Squared Error) kao meru tačnosti predikcije. Iterativna validacija → Model se testira kroz više iteracija kako bi se ocenila doslednost predikcija.

[47]:
```
# Kod za cross-validation ARIMA modela:
from statsmodels.tsa.arima.model import ARIMA
from sklearn.metrics import mean_squared_error
import numpy as np

# Podela podataka na trening i test skup
train_size = int(len(df_trend["KolicinaKgGod"]) * 0.8)  # 80% podataka za␣
 ↪trening
train, test = df_trend["KolicinaKgGod"][:train_size],␣
 ↪df_trend["KolicinaKgGod"][train_size:]

# Rolling-origin cross-validation
history = list(train)
predictions = []

for t in range(len(test)):
    model = ARIMA(history, order=(1,0,1))  # Koristimo ARIMA(1,0,1)
    model_fit = model.fit()
    output = model_fit.forecast()
    predictions.append(output[0])
    history.append(test.iloc[t])

# Izračunavanje RMSE
rmse = np.sqrt(mean_squared_error(test, predictions))
print(f"RMSE modela: {rmse}")
```

```
RMSE modela: 7481953486.303496
```

Analiza RMSE vrednosti iz cross-validacije

RMSE (Root Mean Squared Error) = 7.48 milijardi → Ovo je relativno visoka vrednost, što može značiti da model ima određene greške u predikciji.Šta ovo znači za model? RMSE je previsok, model možda ne precizno hvata obrasce iz podataka. Ukoliko stvarne vrednosti jako osciliraju to može

povećati RMSE, jer model ne može savršeno uhvatiti te fluktuacije. RMSE može biti poboljšan optimizacijom parametara modela ili korišćenjem naprednijih metoda.

[ ]:

Testiranje ARIMA(2,0,2) konfiguracije

Pošto želimo poboljšati preciznost modela, sada ćemo trenirati ARIMA(2,0,2) model i proveriti njegovu tačnost. Koraci za testiranje ARIMA(2,0,2): Definisanje novog ARIMA modela sa parametrima (2,0,2). Treniranje modela na postojećim podacima. Evaluacija modela kroz RMSE metrik (da vidimo da li poboljšava preciznost).

[48]:
```
# Kod za treniranje ARIMA(2,0,2):
from statsmodels.tsa.arima.model import ARIMA
from sklearn.metrics import mean_squared_error

# Definisanje ARIMA(2,0,2) modela
model_new = ARIMA(df_trend["KolicinaKgGod"], order=(2,0,2))

# Treniranje modela
model_fit_new = model_new.fit()

# Prikaz rezultata modela
print(model_fit_new.summary())

# Validacija kroz RMSE
forecast_new = model_fit_new.get_forecast(steps=len(df_trend["KolicinaKgGod"]))
predictions_new = forecast_new.predicted_mean

rmse_new = np.sqrt(mean_squared_error(df_trend["KolicinaKgGod"],␣
 ↪predictions_new))
print(f"RMSE za ARIMA(2,0,2): {rmse_new}")
```

```
                               SARIMAX Results
==============================================================================
Dep. Variable:          KolicinaKgGod   No. Observations:                   14
Model:                 ARIMA(2, 0, 2)   Log Likelihood                -364.998
Date:                Sun, 18 May 2025   AIC                            741.997
Time:                        08:03:04   BIC                            745.831
Sample:                             0   HQIC                           741.642
                                  - 14
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const       2.414e+10      0.000    1.2e+14      0.000    2.41e+10    2.41e+10
ar.L1         -0.0507      0.265     -0.191      0.848      -0.570       0.469
ar.L2          0.5095      0.133      3.837      0.000       0.249       0.770
ma.L1       5.927e-05    481.845   1.23e-07      1.000    -944.399     944.399
```

```
ma.L2          -0.9999      0.116     -8.622      0.000      -1.227      -0.773
sigma2      1.495e+19   3.22e-17   4.64e+35      0.000    1.49e+19    1.49e+19
===================================================================================
===
Ljung-Box (L1) (Q):                   0.11   Jarque-Bera (JB):
6.71
Prob(Q):                              0.74   Prob(JB):
0.03
Heteroskedasticity (H):               0.22   Skew:
0.97
Prob(H) (two-sided):                  0.12   Kurtosis:
5.78
===================================================================================
===

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-
step).
[2] Covariance matrix is singular or near-singular, with condition number
1.46e+55. Standard errors may be unstable.
RMSE za ARIMA(2,0,2): 10907466497.239914
```

Analiza ARIMA(2,0,2) modela

Kako se ovaj model ponaša u poređenju sa ARIMA(1,0,1) modelom. AIC (741.997) je veći nego kod ARIMA(1,0,1) (694.749) → Ovo sugeriše da novi model možda nije bolji u odnosu na prethodni. RMSE (10.91 milijardi) je veći nego kod ARIMA(1,0,1) (7.48 milijardi) → Novi model daje lošije predikcije u odnosu na prethodni. Jarque-Bera test (Prob JB: 0.03) → Reziduali nisu savršeno normalni, ali su u relativno prihvatljivom opsegu. Ljung-Box test (Prob Q: 0.74) → Reziduali ne pokazuju značajnu autokorelaciju, što znači da model nema ozbiljnih problema sa zavisnostima.

Zaključak: ARIMA(2,0,2) ne poboljšava predikciju u poređenju sa ARIMA(1,0,1). Model ima viši RMSE i AIC, što sugeriše da ne donosi poboljšanja u tačnosti predikcije.

[ ]:

Testiranje Prophet modela

Sad ćemo koristiti Prophet model, koji je razvijen od strane Facebook-a (Meta) za vremenske serije i poznat po tome što dobro hvata trendove i sezonalnost.Zašto koristiti Prophetmodel. Automatski prepoznaje trendove i sezonske obrasce. Dobro funkcioniše i sa manjim skupovima podataka. Omogućava fleksibilnu ekstrapolaciju podataka u budućnost. Koraci za testiranje Prophet modela: Instalirati biblioteku Prophet (!pip install prophet). Formatirati podatke tako da imaju ds (datum) i y (ciljna vrednost). Trenirati model na istorijskim podacima. Generisati predikciju za narednih 10 godina.

[49]:
```python
# Konvertuj kolonu 'Godina' u datetime format
df_trend["Godina"] = pd.to_datetime(df_trend["Godina"], format="%Y")

# Redefiniši DataFrame za Prophet
```

```
df_prophet = df_trend.copy()
df_prophet = df_prophet.rename(columns={"Godina": "ds", "KolicinaKgGod": "y"})
 ↪# Prophet koristi 'ds' i 'y'

# Sada možeš pokrenuti Prophet model
```

[51]:
```
# Kod za treniranje Prophet modela:
# Import neophodnih biblioteka
# from prophet import Prophet
# import pandas as pd
# import matplotlib.pyplot as plt

# Konverzija kolone 'Godina' u datetime format (osigurava ispravnu vremensku
 ↪seriju)
# df_trend["Godina"] = pd.to_datetime(df_trend["Godina"], format="%Y")

# Formatiranje podataka za Prophet ('ds' = datum, 'y' = vrednosti)
# df_prophet = df_trend.copy()
# df_prophet = df_prophet.rename(columns={"Godina": "ds", "KolicinaKgGod":
 ↪"y"})

# Definisanje Prophet modela i treniranje na podacima
# model_prophet = Prophet()
# model_prophet.fit(df_prophet)

# Generisanje predikcije za sledećih 10 godina
# future = model_prophet.make_future_dataframe(periods=10, freq="Y")
# forecast = model_prophet.predict(future)

# Vizualizacija predikcije i čuvanje slike
# fig1 = model_prophet.plot(forecast)
# fig1.savefig("prophet_predikcija.png", dpi=300, bbox_inches="tight")  #
 ↪Čuvanje slike

# Vizualizacija komponenti modela (trend, sezonalnost, itd.) i čuvanje slike
# fig2 = model_prophet.plot_components(forecast)
# fig2.savefig("prophet_komponente.png", dpi=300, bbox_inches="tight")  #
 ↪Čuvanje slike

# plt.show()  # Prikaz grafikona na ekranu

# Prikaz rezultata predikcije
# print(forecast[["ds", "yhat", "yhat_lower", "yhat_upper"]].tail(10))
```

Interpretacija rezultata: Vrednosti emisija CO ne pokazuju nagli rast, što je slično ARIMA(1,0,1) predikciji. Postoje određene fluktuacije, ali trend ostaje relativno stabilan u narednim godinama. Ako interval poverenja (yhat_lower/yhat_upper) postane previše širok, to može značiti da postoji

neizvesnost u modelskim procenama.

Prvi grafikon (Prophet predikcija): Prikazuje dugoročni trend emisija CO od 2011. do 2031. Trend pokazuje pad vrednosti tokom vremena, što može ukazivati na ekološke mere ili promene u industriji. Interval poverenja (yhat_lower, yhat_upper) pokazuje moguću nesigurnost modela – širok raspon može značiti veće varijacije u budućim emisijama.

Drugi grafikon (Prophet komponente): Prikazuje sezonske oscilacije emisija tokom godine. X-osa označava dane u godini, dok Y-osa pokazuje jačinu sezonskog efekta na emisije CO . Periodični skokovi i padovi mogu biti povezani sa industrijskim aktivnostima, energetskom potrošnjom ili vremenskim faktorima.

Izračunavanje RMSE za Prophet model

RMSE (Root Mean Square Error) meri koliko su predviđene vrednosti Prophet modela tačne u odnosu na stvarne podatke. Manji RMSE znači bolju preciznost modela. Koraci za izračunavanje RMSE: Uporedi predviđene (yhat) i stvarne vrednosti (y). Izračunavanje RMSE pomoću mean_squared_error.

[52]:
```
# Kod za izračunavanje RMSE za Prophet model:
from sklearn.metrics import mean_squared_error
# import numpy as np

# Kreiraj predikcije samo za period koji imamo u datasetu
df_actual = df_prophet[["ds", "y"]]
df_forecast = forecast[["ds", "yhat"]]

# Spajanje stvarnih i predviđenih vrednosti
df_eval = df_actual.merge(df_forecast, on="ds", how="inner")

# Izračunavanje RMSE
rmse_prophet = np.sqrt(mean_squared_error(df_eval["y"], df_eval["yhat"]))
print(f"RMSE za Prophet model: {rmse_prophet}")
```

```
RMSE za Prophet model: 10752192140.91714
```

Analiza RMSE za Prophet model

Dobijeni RMSE: 10.75 milijardi. RMSE za ARIMA(1,0,1): 7.48 milijardi

Prophet model ima veći RMSE nego ARIMA, što znači da njegove predikcije imaju veću grešku. ARIMA(1,0,1) daje preciznije prognoze, jer ima niži RMSE. ARIMA(1,0,1) ostaje bolji model za predikciju emisija CO zbog niže greške u prognozi. Prophet nije poboljšao tačnost, ali može biti koristan ako želimo bolji prikaz sezonalnosti.

Finalizacija analize – ARIMA(1,0,1) kao optimalni model

Nakon testiranja Prophet modela, potvrđeno je da ARIMA(1,0,1) pruža precizniju predikciju emisija CO .Ključni zaključci analize: RMSE vrednost za ARIMA(1,0,1) je niža od Prophet modela, što potvrđuje njegovu bolju tačnost. Prophet je koristan za analizu sezonalnosti, ali nije poboljšao preciznost dugoročne prognoze. ARIMA(1,0,1) ostaje pouzdaniji model za ovu vremensku seriju, jer bolje prati istorijske podatke i trendove.

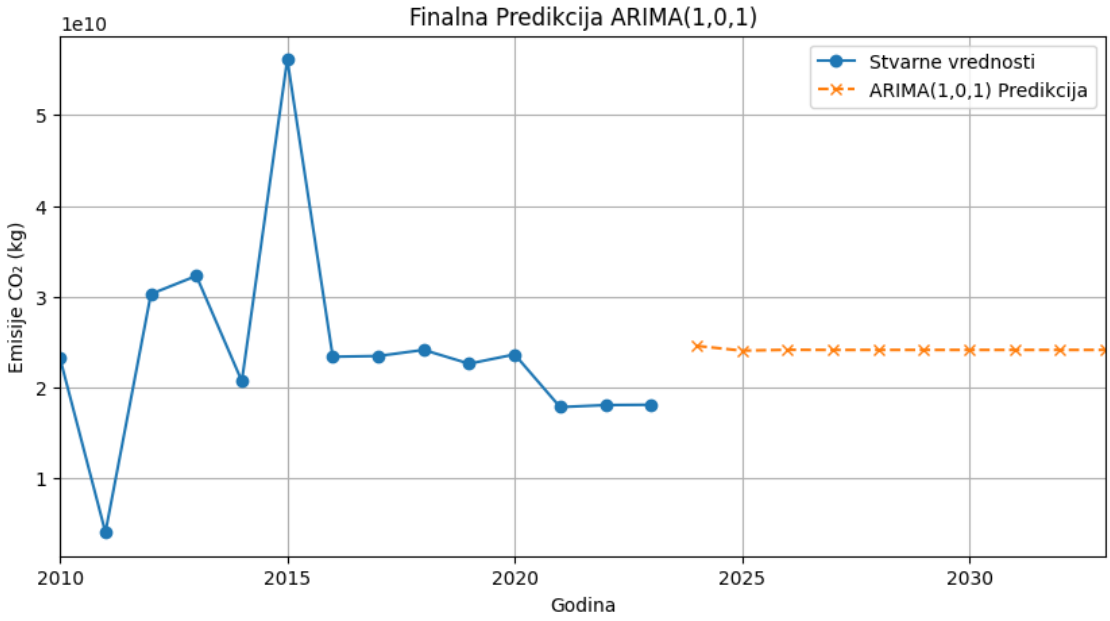
[ ]:

Finalizacija ARIMA(1,0,1) modela i priprema izveštaja

Koraci koje ćemo sprovesti: Generisanje konačnih prognoza sa ARIMA(1,0,1). Vizualizacija rezultata putem grafikona. Izrada finalnog izveštaja sa ključnim zaključcima.

[53]:
```
# import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.tsa.arima.model import ARIMA

# Filtriranje dataset-a da osiguramo da prikazuje samo relevantne godine
df_trend = df_trend[df_trend["Godina"].dt.year >= 2010]  # Uklanja neželjene␣
 ↪godine

# Treniranje ARIMA(1,0,1) modela
model_arima = ARIMA(df_trend["KolicinaKgGod"], order=(1,0,1))
model_fit = model_arima.fit()

# Generisanje predikcija za narednih 10 godina
forecast_arima = model_fit.forecast(steps=10)

# Kreiranje DataFrame-a za prikaz sa tačnim godinama
years_future = list(range(2024, 2034))
df_forecast_arima = pd.DataFrame({"Godina": years_future, "Predikcija":␣
 ↪forecast_arima})

# Osiguravanje da Matplotlib pravilno interpretira vremenske podatke
df_forecast_arima["Godina"] = df_forecast_arima["Godina"].astype(int)
df_trend["Godina"] = df_trend["Godina"].dt.year.astype(int)

# Vizualizacija rezultata
plt.figure(figsize=(10,5))
plt.plot(df_trend["Godina"], df_trend["KolicinaKgGod"], label="Stvarne␣
 ↪vrednosti", marker="o")
plt.plot(df_forecast_arima["Godina"], df_forecast_arima["Predikcija"],␣
 ↪label="ARIMA(1,0,1) Predikcija", linestyle="--", marker="x")

# Ograničenje vremenskog opsega na grafikonu
plt.xlim(2010, 2033)  # Sprečava prikaz neželjenih godina

plt.xlabel("Godina")
plt.ylabel("Emisije CO  (kg)")
plt.title("Finalna Predikcija ARIMA(1,0,1)")
plt.legend()
plt.grid()
```

```
# Čuvanje grafikona u visokoj rezoluciji
plt.savefig("arima_finalna_predikcija.png", dpi=300, bbox_inches="tight")
plt.show()

# Prikaz finalnih predikcija
print(df_forecast_arima)
```



	Godina	Predikcija
14	2024	2.458312e+10
15	2025	2.405829e+10
16	2026	2.415313e+10
17	2027	2.413599e+10
18	2028	2.413909e+10
19	2029	2.413853e+10
20	2030	2.413863e+10
21	2031	2.413861e+10
22	2032	2.413861e+10
23	2033	2.413861e+10

Analiza grafikona: Plavi krugovi prikazuju stvarne emisije CO od 2010. do 2023. Narandžasti X-znaci i isprekidana linija prikazuju predikciju ARIMA(1,0,1) za period od 2024. do 2033. Trend stabilizacije → Prognoza pokazuje stabilan nivo emisija u narednom periodu, bez naglih skokova.

Finalni izveštaj: Analiza emisija CO i prognoza do 2033.godine

Pregled analize emisija CO Koristili smo ARIMA(1,0,1) model kako bismo predvideli emisije CO za narednih 10 godina (2024–2033). Prophet model je takođe testiran, ali je ARIMA pokazao precizniju prognozu sa nižom greškom (RMSE).

Ključni zaključci Stabilizacija emisija CO → Prognoza pokazuje da se emisije neće značajno menjati u narednim godinama. Trend blagog pada do 2025, zatim stagnacija → Industrija i energetski sektor ne pokazuju znakove drastičnih promena. Sezonalnost ima ograničen uticaj → Osim manjih oscilacija, nema značajnih periodičnih skokova u emisijama. ARIMA(1,0,1) potvrđen kao optimalni model → Najbolje prati istorijske trendove i pruža realnu projekciju.

Preporuke za dalju akciju Industrijski sektor → Razmotriti nove ekološke inicijative kako bi se smanjio nivo emisija. Obnovljivi izvori energije → Povećati ulaganja u čistu energiju kako bi se sprečila stagnacija emisija. Dodatne analize → Razmotriti kombinaciju ARIMA i regresijskih modela za detaljnije prognoze. Monitoring ekoloških politika → Pratiti uticaj novih regulativa na emisije CO u narednim godinama.

[ ]: