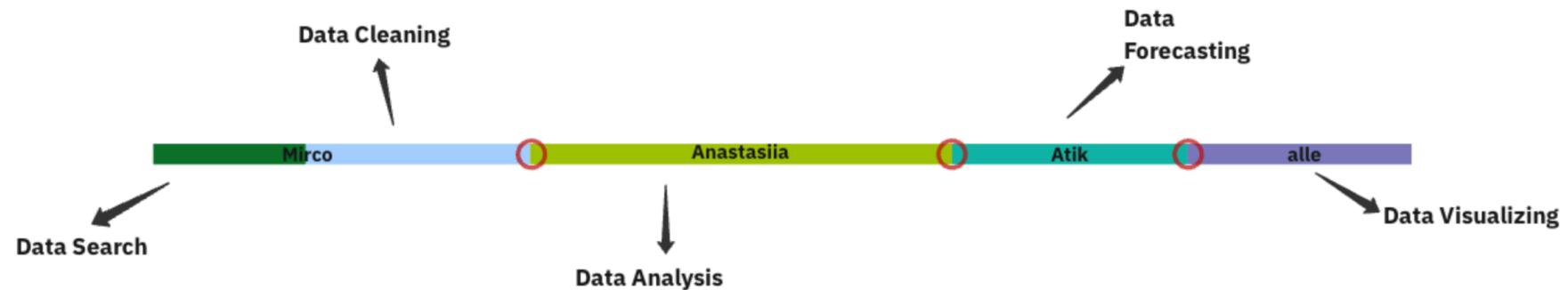


# Datenbasierte Entwicklung, Prognose und Visualisierung eines Nachhaltigkeitsindex für europäische Staaten anhand zentraler Umweltindikatoren.

## Introduction

Dieses Projekt umfasst eine vollständige Datenanalyse zur Bewertung der Nachhaltigkeit europäischer Länder.

Von der Datensuche über die Bereinigung und Normalisierung mit Python bis hin zur Entwicklung eines Green Index wurden zentrale Umwelt- und Energieindikatoren analysiert. Abschließend wurden die Ergebnisse in einer interaktiven Tableau-Karte visualisiert, die Vergleiche und Erkenntnisse zur grünen Leistung Europas ermöglicht.



## GitHub

Das Projekt wurde auf GitHub strukturiert, um eine klare Trennung zwischen den verschiedenen Phasen der Datenverarbeitung, Analyse und Visualisierung zu gewährleisten. Die wichtigsten Verzeichnisse sind:

1. **data\_raw** – Enthält die unbearbeiteten Ausgangsdaten.
2. **data\_subindex** – Einzelne Subindikatoren in Rohform (z. B. Energieverbrauch, CO<sub>2</sub>, Waldfläche).
3. **data\_final** – Zusammengeführte, bereinigte und normierte Datensätze zur weiteren Analyse.
4. **data\_for\_forecast** – Spezielle Dateien, vorbereitet für Zeitreihen- und Prognosemodelle.
5. **scripts** – Python-Skripte zur Datenbereinigung, Normalisierung und Analyse mit pandas und matplotlib.
6. **analysis** – Dokumentation und Zwischenergebnisse der Analyse.
7. **tableau** – Dateien für die finale Visualisierung im Tableau-Dashboard.

analysis	xlsx files	last week
data_final	analysis, cleaned files, created files for forecast, created ...	last week
data_for_forecast	analysis, cleaned files, created files for forecast, created ...	last week
data_raw	Raw File added: Fossil_consumption	3 weeks ago
data_subindex	xlsx files	last week
scripts	analysis, cleaned files, created files for forecast, created ...	last week
tableau	Add files via upload	27 minutes ago

## Data Search

Für unser Projekt haben wir verschiedene verlässliche Datensätze von renommierten Organisationen wie der IEA, FAO und Our World in Data verwendet.

Diese Quellen bieten aktuelle und gut dokumentierte Informationen über CO<sub>2</sub>-Emissionen, die Nutzung fossiler und erneuerbarer Energien sowie den Anteil von Waldflächen.

Ich habe die Daten analysiert, um Entwicklungen im Energieverbrauch und deren Auswirkungen auf die Umwelt besser zu verstehen.

Die Aktualität und Transparenz dieser Datenquellen waren entscheidend für die Aussagekraft meiner Ergebnisse.

### Annual CO<sub>2</sub> emissions

Annual total emissions of carbon dioxide (CO<sub>2</sub>), excluding land-use change, measured in tonnes.

Source	Global Carbon Budget (2024) - with major processing by Our World in Data
Last updated	November 21, 2024
Next expected update	November 2025
Date range	1750–2023



### Forest area (% of land area)

Forest area is land under natural or planted stands of trees of at least 5 meters in situ, whether productive or not, and excludes tree stands in agricultural production systems (for example, in fruit plantations and agroforestry systems) and trees in urban parks and gardens.

ID	AGLNUOTRE2S
Source	Food and Agriculture Organization, electronic files and web site.
License	CC BY 4.0 BY

#### Dataset information

##### Share of renewable energy in gross final energy consumption by sector

This indicator measures the share of renewable energy consumption in gross final energy consumption by sector. Renewable energy includes biofuels, solar, wind, hydroelectric power, geothermal, and tidal.

<input checked="" type="checkbox"/> Prefer latest data	May 29, 2025
Source of data	IEA.org
Source dataset	IEA.org
Dataset license	World Bank Open Data License: Attribution-NonCommercial-ShareAlike
Last data update	17/05/2025 10:07:43 (10 hours ago)

### Fossil fuel energy consumption (% of total)

Fossil fuel comprises coal, oil, petroleum, and natural gas products.

ID	ESLUECQH4PZ5
Source	IEA Energy Statistics Data & Tables - IEA.org website
Notes	Includes geothermal sources of energy. Oil, natural gas, and renewables. Use and distribution of these fuels are subject to IEA terms and conditions.

### Other renewables (including geothermal and biomass)

Figures are based on gross generation and do not account for cross-border electricity supply.

Source	Energy Institute - Statistical Review of World Energy (2024) - with major processing by Our World in Data
Last updated	May 2024
Next expected update	June 2025



## Data Cleaning

### Ziele der Cleaning-Phase:

1. Nur relevante Daten behalten (z. B. europäische Länder)
2. Nur aktuelle Jahre verwenden (2019–2023)
3. Unnötige Spalten löschen
4. Lesbare Struktur herstellen

```
# List of European countries
european_countries = [
    "Albania", "Andorra", "Armenia", "Austria", "Azerbaijan", "Belarus", "Belgium", "Bosnia and Herzegovina", "Bulgaria",
    "Croatia", "Cyprus", "Czech Republic", "Denmark", "Estonia", "Finland", "France", "Georgia", "Germany", "Greece",
    "Hungary", "Iceland", "Ireland", "Italy", "Kazakhstan", "Kosovo", "Latvia", "Liechtenstein", "Lithuania", "Luxembourg",
    "Malta", "Moldova", "Monaco", "Montenegro", "Netherlands", "North Macedonia", "Norway", "Poland", "Portugal", "Romania",
    "Russia", "San Marino", "Serbia", "Slovakia", "Slovenia", "Spain", "Sweden", "Switzerland", "Turkey", "Ukraine", "United Kingdom", "Vatican"
]

# Filter for European countries
europe_df = df[df['country'].isin(european_countries)]

# Find the latest year in the dataset
latest_year = europe_df['year'].max()
print(f"\nLatest year in dataset: {latest_year}")

# Filter only the last 5 years
last_5_years = list(range(latest_year - 4, latest_year + 1))
recent_data = europe_df[europe_df['year'].isin(last_5_years)]
```

1. Die Rohdaten werden gefiltert, um nur die Länder und Jahre zu behalten, die wir wirklich brauchen.

2. „Cleaning“ bedeutet: Unordnung entfernen, Fokus auf Relevantes.

## Data Normalization

### Ziel:

1. Unterschiedliche Datenquellen sollen **gleich aufgebaut** sein  
→ später vergleichbar / zusammenfügbar
2. Einfache, einheitliche Spaltennamen und Datenformate

```
# Split the 'freq,unit,geo\\TIME_PERIOD' column by commas into three parts
split_cols = df['freq,unit,geo\\TIME_PERIOD'].str.split(',', expand=True)

# Assign the split columns directly to the DataFrame
df[['freq', 'unit', 'geo']] = split_cols

# Drop the original combined column as it's no longer needed
df.drop('freq,unit,geo\\TIME_PERIOD', axis=1, inplace=True)

# Melt the DataFrame to have one column for years and one column for values
df_melted = df.melt(id_vars=['freq', 'unit', 'geo'], var_name='year', value_name='value')
```

1. Eine komplexe Spalte wird aufgeteilt in lesbare Felder.

2. Anschließend werden die Daten so „umgebaut“, dass jedes Jahr eine eigene Zeile ist – das ist **long format**, und wird z. B. für Visualisierungen gebraucht.

## Final Csv

### Ziel:

1. Bereinigte, einheitlich strukturierte Datei für jede Quelle
2. Als .csv gespeichert → bereit für weitere Analyse oder Visualisierung

```
Save the cleaned DataFrame to a CSV
f_last_5_years.to_csv('data_final/european_forest_data.csv', index=False)
```

1. Die finale Tabelle wird als Datei exportiert – bereit für Diagramme, Karten, Präsentation usw.

## Green Index

- Ziel der Analyse war es, einen **Nachhaltigkeitsindex** zu erstellen, um das Nachhaltigkeitsniveau europäischer Länder zu bewerten.
- Es gab fünf Dateien und insgesamt acht Indikatoren, da eine der Dateien vier Indikatoren enthielt.
- Jede Datei wurde analysiert und für jeden Indikator einen Subindex erstellt, der dann zu einem Gesamtwert aggregiert wurde.

### Umweltindikatoren:

1. CO<sub>2</sub>-Ausstoß pro Kopf (Tonnen pro Person)
2. Waldfläche (Prozentanteil der gesamten Landesfläche)
3. Anteil fossiler Energie (Prozentanteil am gesamten Energieverbrauch)
4. Anteil erneuerbarer Energie (Prozentanteil am gesamten Energieverbrauch)
5. Erneuerbare Energieerzeugung (TWh pro Kopf), darunter:
  - a. Solarstromerzeugung
  - b. Windstromerzeugung
  - c. Wasserkrafterzeugung
  - d. Sonstige erneuerbare Erzeugung (Geothermie und Biomasse)

## Datenvorbereitung / Herausforderungen

### 1. Datenart

- Die Datei „5. Erneuerbare Energieerzeugung“ wurde zunächst in **Excel** bearbeitet, da sie nur absolute Werte enthalten hat.
- Es wurde eine neue Spalte mit der Bevölkerungszahl („Population“) hinzugefügt, und anschließend wurden die **Werte pro Kopf** berechnet.
- Zu dieser Datei wurden später auch vier weitere Dateien erstellt, da sie vier Indikatoren enthält (Solarstromerzeugung, Windstromerzeugung, Wasserkrafterzeugung und sonstige erneuerbare Erzeugung).

### 2. Spalten

- Die Spaltennamen wurden so angepasst, dass sie in allen Dateien übereinstimmen.
- Es wurden nur die benötigten Spalten ausgewählt und die Datensätze entsprechend sortiert.

	Country	Year	Code	Other renewables (TWh)	Solar generation (per capita - Wh)	Wind generation (TWh)	Hydro generation (TWh)
85	Germany	2019	DEU	50,32362	45,221	125,894	19,731
86	Germany	2020	DEU	51,16132	49,496	132,102	18,322
87	Germany	2021	DEU	50,304493	49,34	114,647	19,657415
88	Germany	2022	DEU	51,902596	60,304	124,816	17,624172
89	Germany	2023	DEU	49,491	61,216	142,103	19,639

```
df = pd.read_csv('../data_final/european_co2_last5years.csv')
# Rename columns
filtered_df = df.rename(columns={'country': 'Country', 'year': 'Year', 'population': 'Population',
                                 'co2': 'CO2 (Mt)', 'co2_per_capita': 'CO2 (per capita)'})
# Filter for only needed columns
relevant_columns = ['Country', 'Year', 'Population', 'CO2 (Mt)', 'CO2 (per capita)']
filtered_df = filtered_df[relevant_columns]
# Sort
filtered_df = filtered_df.sort_values(by='Country')
filtered_df.head()
```

```
# Find rows with null values
null_co2_rows = filtered_df[filtered_df['CO2 (Mt)'].isnull()]

# Display rows with null values
print("Rows with null 'CO2 (Mt)':")
print(null_co2_rows)

# Drop rows for Monaco, San Marino, Vatican
countries_to_drop = ['Monaco', 'San Marino', 'Vatican']
filtered_df_w_dropped_countries = filtered_df[~filtered_df['Country'].isin(countries_to_drop)]
```

```
# Rename countries to match the names from other datasets
country_mapping = {'Czechia': 'Czech Republic', 'Russia': 'Russian Federation'}

filtered_df['Country'] = filtered_df['Country'].replace(country_mapping)
```

### 3. NULL-Werte

- Länder mit NULL-Werten wurden identifiziert und danach gelöscht, da es hier ausschließlich die Mikrostaaten waren, die nur einen sehr geringen Einfluss auf die Analyse haben.

### 4. Länder Namen

- Länder Namen wurden standardisiert (z. B. Czechia → Czech Republic)

## Vorbereitung der Daten für Prognose

Nach der Bereinigung wurden separate Dateien für die Prognoseberechnungen vorbereitet. Dazu wurde:

- Die Tabelle wurde **pivotiert**, sodass jede Zeile ein Land darstellt und jede Spalte ein Jahr.
- Das Ergebnis wurde als CSV-Datei gespeichert, um sie in Prognosemodellen weiterzuverwenden.

```
# Prepare file for forecast
forecast_df = filtered_df_w_dropped_countries[['Country', 'Year', 'CO2 (per capita)']]
forecast_df = forecast_df.pivot(index='Country', columns='Year', values='CO2 (per capita)')
forecast_df = forecast_df.reset_index()

# Save file
forecast_df.to_csv('../data_for_forecast/co2_per_capita-clean.csv', index=False)
forecast_df.head()
```

	Country	Year	Population	CO2 (Mt)	CO2 (per capita)
0	Albania	2019	2885012.0	4.827	1.673
1	Albania	2020	2871951.0	4.711	1.640
2	Albania	2021	2849643.0	5.134	1.802
3	Albania	2022	2827615.0	5.173	1.830
4	Albania	2023	2811660.0	5.144	1.830

	Year	Country	2019	2020	2021	2022	2023
0	Albania	1.673	1.640	1.802	1.830	1.830	
1	Andorra	6.419	4.923	5.236	5.263	5.242	
2	Armenia	2.164	2.351	2.584	2.487	2.585	
3	Austria	7.651	6.969	7.333	6.783	6.416	
4	Azerbaijan	3.733	3.559	3.742	3.923	4.258	

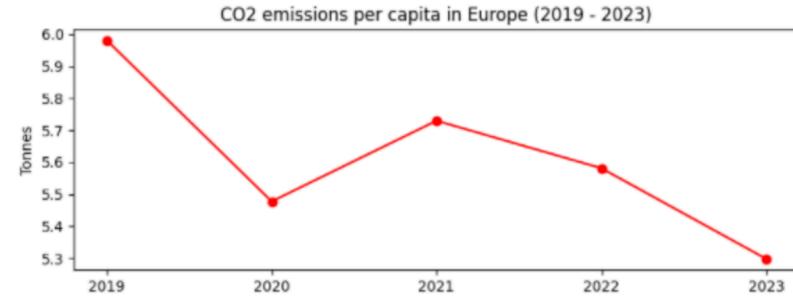


## Analyse der Daten mit Matplotlib

Für die Analyse wurden mehrere Diagramme mit matplotlib erstellt. Diese Diagramme dienten dazu, **zentrale Erkenntnisse** aus den Daten auf einen Blick sichtbar zu machen und länderspezifische Unterschiede hervorzuheben.

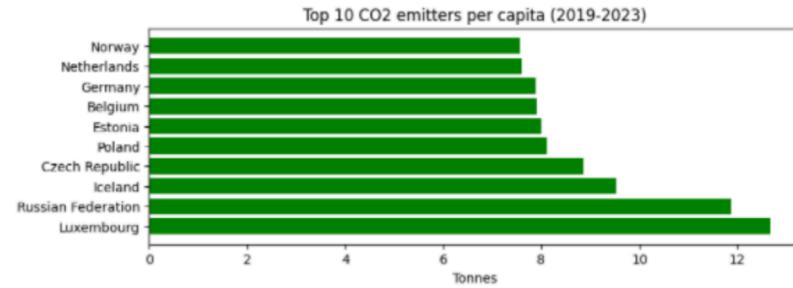
- Ein Liniendiagramm zeigt die Entwicklung der durchschnittlichen CO<sub>2</sub>-Emissionen pro Kopf. Dabei ist ein Rückgang zu erkennen, besonders im Jahr 2020, vermutlich wegen der COVID-19-Pandemie.

```
# TREND ANALYSIS OVER TIME – CO2 EMISSIONS PER CAPITA THE YEARS
yearly_co2_per_capita = filtered_df_w_dropped_countries.groupby('Year')['CO2 (per capita)'].mean()
plt.plot(yearly_co2_per_capita.index, yearly_co2_per_capita.values, marker='o', color='red')
plt.xticks([2019,2020,2021,2022,2023])
plt.title('CO2 emissions per capita in Europe (2019 – 2023)')
plt.ylabel('Tonnes')
plt.show()
```



- Ein weiteres Balkendiagramm zeigt die zehn Länder mit dem höchsten durchschnittlichen CO<sub>2</sub>-Ausstoß pro Kopf.

```
# TOP CO2 EMITTERS PER CAPITA (2019–2023)
avg_co2 = filtered_df_w_dropped_countries.groupby('Country')['CO2 (per capita)'].mean().reset_index()
top_countries = avg_co2[['Country', 'CO2 (per capita)']].sort_values(by='CO2 (per capita)', ascending=False)
plt.barh(top_countries['Country'][:10], top_countries['CO2 (per capita)'][:10], color='green')
plt.title('Top 10 CO2 emitters per capita (2019–2023)')
plt.xlabel('Tonnes')
plt.show()
```



## Skalierung (Min-Max-Normalisierung)

Um einen Subindex für jeden Umweltindikator zu erstellen, wurden die Werte zwischen 0 und 1 skaliert.

- Bei **positiven** Indikatoren wie dem Anteil an Waldfäche wurde die **Skala direkt verwendet**, da höhere Werte eine bessere Nachhaltigkeitsleistung anzeigen.

$$\text{Index} = \frac{x - x_{\min}}{x_{\max} - x_{\min}}$$

- Bei **negativen** Indikatoren wie dem CO<sub>2</sub>-Ausstoß wurde die **Skala umgekehrt**, sodass ein höherer Wert eine bessere Bewertung darstellt

$$\text{Index} = \frac{x_{\max} - x}{x_{\max} - x_{\min}}$$

- Das Ergebnis wurde zur Berechnung des Nachhaltigkeitsindex für jeden Umweltindikator als CSV-Datei gespeichert.

```
# Normalize the average values (0 = worst, 1 = best)
max_val = avg_forest_area['Forest area (avg)'].max()
min_val = avg_forest_area['Forest area (avg)'].min()
avg_forest_area['forest_area_subindex'] = (avg_forest_area['Forest area (avg)'] - min_val) / (max_val - min_val)

# Sort
avg_forest_area.sort_values(by='forest_area_subindex', ascending=False)
```

	Country	Forest area (avg)	forest_area_subindex
16	Finland	73.728040	1.000000
45	Sweden	68.699666	0.931798
34	Montenegro	61.486989	0.833970
44	Slovenia	61.421659	0.833084
15	Estonia	57.046271	0.773726
29	Latvia	54.841171	0.743831

```
# Normalize the average values (0 = worst, 1 = best)
max_val = avg_co2['CO2 (per capita)'].max()
min_val = avg_co2['CO2 (per capita)'].min()
avg_co2['co2_subindex'] = round((max_val - avg_co2['CO2 (per capita)']) / (max_val - min_val),5)

# Sort
avg_co2.sort_values(by='co2_subindex', ascending=False)
```

	Country	CO2 (per capita)	co2_subindex
0	Albania	1.7550	1.000000
29	Moldova	1.8512	0.99119
2	Armenia	2.4342	0.93777
16	Georgia	2.9422	0.89123
28	Malta	3.2416	0.86380
42	Sweden	3.6528	0.82613

```
avg_forest_area.to_csv('../data_subindex/subindex_forest_ares_2019-2022.csv', index=False)
```

## Berechnung des Nachhaltigkeitsindex

- Der Nachhaltigkeitsindex wurde in **Excel** berechnet, indem die Umweltindikatoren mit individuellen **Gewichtungen** kombiniert wurden.
- Die gewichteten Teilwerte wurden addiert, um den **Gesamtwert** – den Green Index – zu erhalten.
- Zur besseren Lesbarkeit wurden die Werte von einer 0-1-Skala in **0-100-Skala umgewandelt**.
- Nullwerte werden als schlechteste Ausprägung interpretiert.
- Die Daten wurden für die Visualisierung mit Tableau verwendet.
- Dieser Index könnte als positives oder negatives Kriterium dienen. Politik, Investoren und Organisationen könnten ihn zur strategischen Planung, Risikoanalyse und Bewertung nachhaltiger Entwicklungen nutzen.

Indicator	Weight
renewable_hydro_generation_subindex	9,00%
renewable_wind_generation_subindex	9,00%
renewable_solar_generation_subindex	9,00%
renewable_other_generation_subindex	9,00%
co2_subindex	17,00%
fossil_consumption_subindex	17,00%
renewable_energy_subindex	12,00%
forest_area_subindex	18,00%
<b>GREEN INDEX</b>	<b>100,00%</b>

Country	GREEN INDEX
Sweden	64,87
Finland	52,91
Norway	43,62
Iceland	43,09
Denmark	42,80
Latvia	42,53
Spain	41,70
Germany	40,06
Portugal	39,93

### 1. Zielsetzung

Ziel war es, die Entwicklung zentraler Nachhaltigkeitsindikatoren (CO<sub>2</sub>-Emissionen pro Kopf und Solarenergie-Erzeugung) pro Land zu analysieren und bis 2028 zu prognostizieren. Die Ergebnisse zeigen sowohl Top 5 Länder mit starkem Anstieg als auch die Länder mit sinkenden Werten

### 2. Datengrundlage

Die Daten stammen aus einer CSV-Datenbasis mit folgenden Informationen:

CO<sub>2</sub> pro Kopf (Tonnen) pro Land (2019–2023)

Solarenergie-Erzeugung (TWh) pro Land (2019–2023)

### Was ist der Prophet-Algorithmus?

Prophet ist ein Open-Source-Zeitreihenmodell, das von Meta (Facebook) entwickelt wurde, um zukünftige Werte auf Basis historischer Daten vorherzusagen.

Er eignet sich besonders gut für:

kurze oder unregelmäßige Zeitreihen,

Daten mit Ausreißern oder fehlenden Jahren,

einfache und schnelle Anwendung auch ohne tiefes Statistikwissen.

$$y(t) = g(t) + s(t) + h(t) + \varepsilon(t)$$

Bedeutung der Bestandteile:

$g(t)$ : langfristiger Trend (z. B. Wachstum, Sättigung)

$s(t)$ : saisonale Schwankungen (nicht relevant bei Jahreswerten)

$h(t)$ : Effekte durch Ereignisse (z. B. politische Maßnahmen)

$\varepsilon(t)$ : zufällige Fehler oder Ausreißer

Da wir Jahresdaten ohne Saisonalität verwenden, vereinfacht sich das Modell zu:

$$y(t) = g(t) + \varepsilon(t)$$

Restriktionen des Modells (implizit):

Der Trend

$g(t)$  wird als piecewise-linear oder logistisch angenommen

Es gibt keine erklärenden Variablen außer der Zeit

Zukunftsdaten werden rein aus der Zeitreihe geschätzt (keine Korrelationen zwischen Ländern!)

#### Ablauf und Struktur

Schritt 1: Auswahl & Bereinigung der CSV-Daten

Schritt 2: Umstrukturierung in Prophet-kompatibles Format ( $ds$  = Datum,  $y$  = Zielwert)

Schritt 3: Modelltraining pro Land

Schritt 4: Vorhersage bis inkl. 2028

Schritt 5: Sortierung der Länder nach Veränderung

Schritt 6: Visualisierung als Tabelle mit Forecast-Werten oder als kompakte Liniendiagramme mit Zahlen (Sparklines)

```
df_prophet = df.T.reset_index()
df_prophet.columns = ['ds', 'y']
df_prophet['ds'] =
pd.to_datetime(df_prophet['ds'], format='%Y')
model =
Prophet()
model.fit(df_prophet)
future =
model.make_future_dataframe(periods=5,
freq='Y')
forecast = model.predict(future)
```

```
import pandas as pd
import matplotlib.pyplot as plt
from prophet import Prophet

# ♦ CSV-Datei laden
df = pd.read_csv("co2_per_capita-clean.csv")
df.columns = df.columns.str.strip()

# ♦ Letztes verfügbares Jahr ermitteln
letztes_jahr = df.columns[1:].astype(int).max()

# ♦ Länder mit gültigen Werten im letzten Jahr
df_last_year = df[['Country', str(letztes_jahr)]].copy()
df_last_year[str(letztes_jahr)] = pd.to_numeric(df_last_year[str(letztes_jahr)], errors='coerce')
df_last_year = df_last_year.dropna()

# ♦ Top 5 und Bottom 5 (# 0) auswählen
top5 = df_last_year.sort_values(by=str(letztes_jahr), ascending=False).head(5)['Country'].tolist()
low5 = df_last_year[df_last_year[str(letztes_jahr)] > 0].sort_values(by=str(letztes_jahr)).head(5)['Country'].tolist()

# ♦ Forecast-Funktion mit Prophet
def forecast_plot(laender_liste, titel):
    plt.figure(figsize=(12, 6))
```

```

for land in laender_liste:
    df_land = df[df['Country'] == land].copy()
    df_prophet = df_land.iloc[:, 1:].T.reset_index()
    df_prophet.columns = ['ds', 'y']
    df_prophet['ds'] = pd.to_datetime(df_prophet['ds'], format='%Y')
    df_prophet['y'] = pd.to_numeric(df_prophet['y'], errors='coerce')
    df_prophet = df_prophet.dropna()

    if len(df_prophet) < 3:
        continue

    model = Prophet()
    model.fit(df_prophet)
    future = model.make_future_dataframe(periods=5, freq='Y')
    forecast = model.predict(future)

    forecast_range = forecast[forecast['ds'].dt.year > letztes_jahr]
    plt.plot(forecast_range['ds'].dt.year, forecast_range['yhat'], label=land)

    for i in range(len(forecast_range)):
        x = forecast_range['ds'].dt.year.iloc[i]
        y = forecast_range['yhat'].iloc[i]
        plt.text(x, y, f"{y:.1f}", fontsize=8, ha='center')

plt.title(title)
plt.xlabel("Jahr")
plt.ylabel("CO2 pro Kopf (Tonnen)")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()

# * Forecast-Diagramme erzeugen
forecast_plot(top5, f"Forecast CO2 pro Kopf - Top 5 Länder ab {letztes_jahr + 1}")
forecast_plot(low5, f"Forecast CO2 pro Kopf - Niedrigste 5 Länder ab {letztes_jahr + 1}")

```

```
df = pd.read_csv("co2_per_capita-clean.csv")
df.columns = df.columns.str.strip()

df_land = df[df['Country'] == land].copy()
df_prophet = df_land.iloc[:, 1: ].T.reset_index()
df_prophet.columns = ['ds', 'y']
df_prophet['ds'] = pd.to_datetime(df_prophet['ds'], format='%Y')
df_prophet['y'] = pd.to_numeric(df_prophet['y'], errors='coerce')

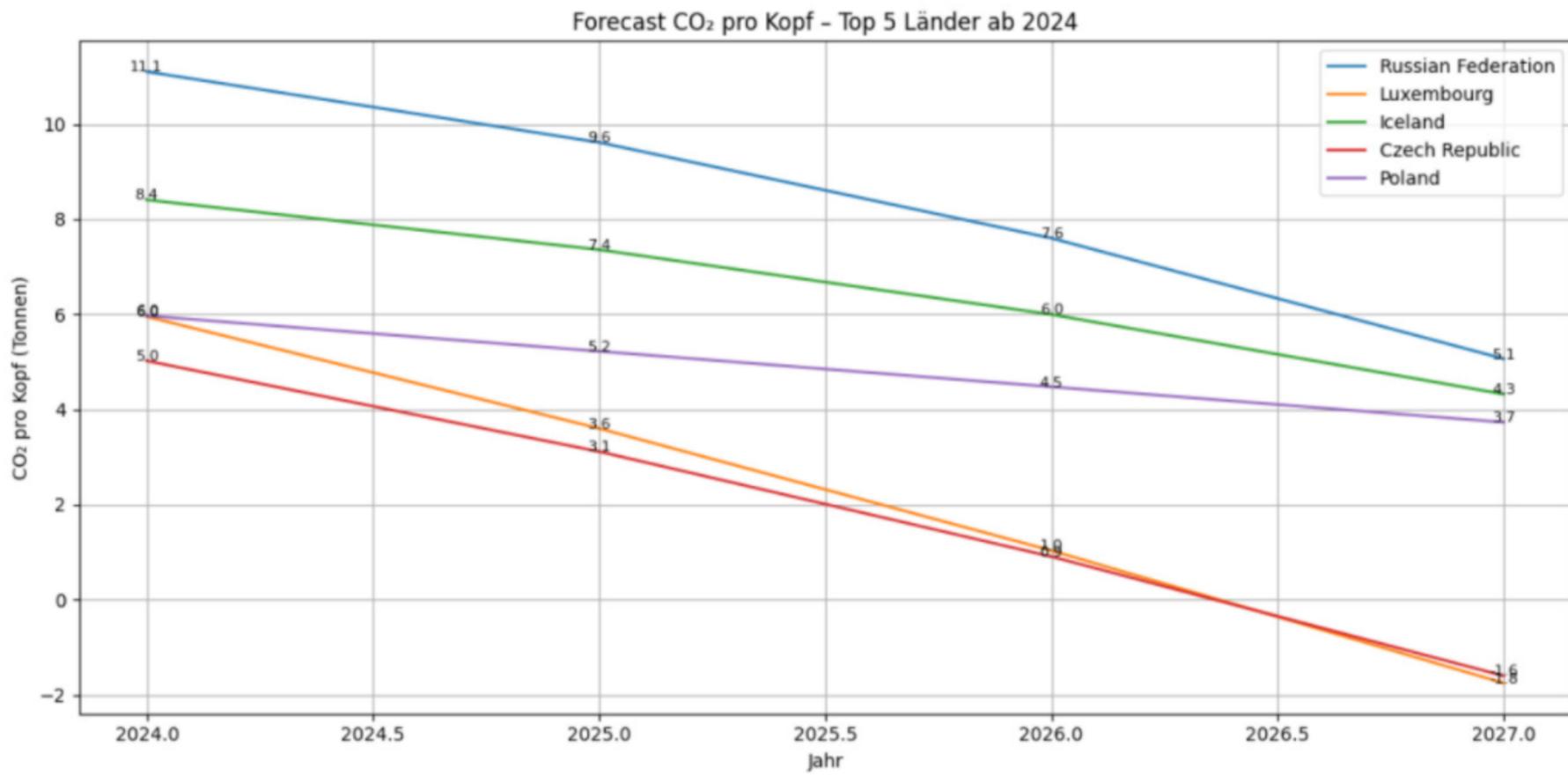
model = Prophet()
model.fit(df_prophet)

future = model.make_future_dataframe(periods=5, freq='Y')
forecast = model.predict(future)

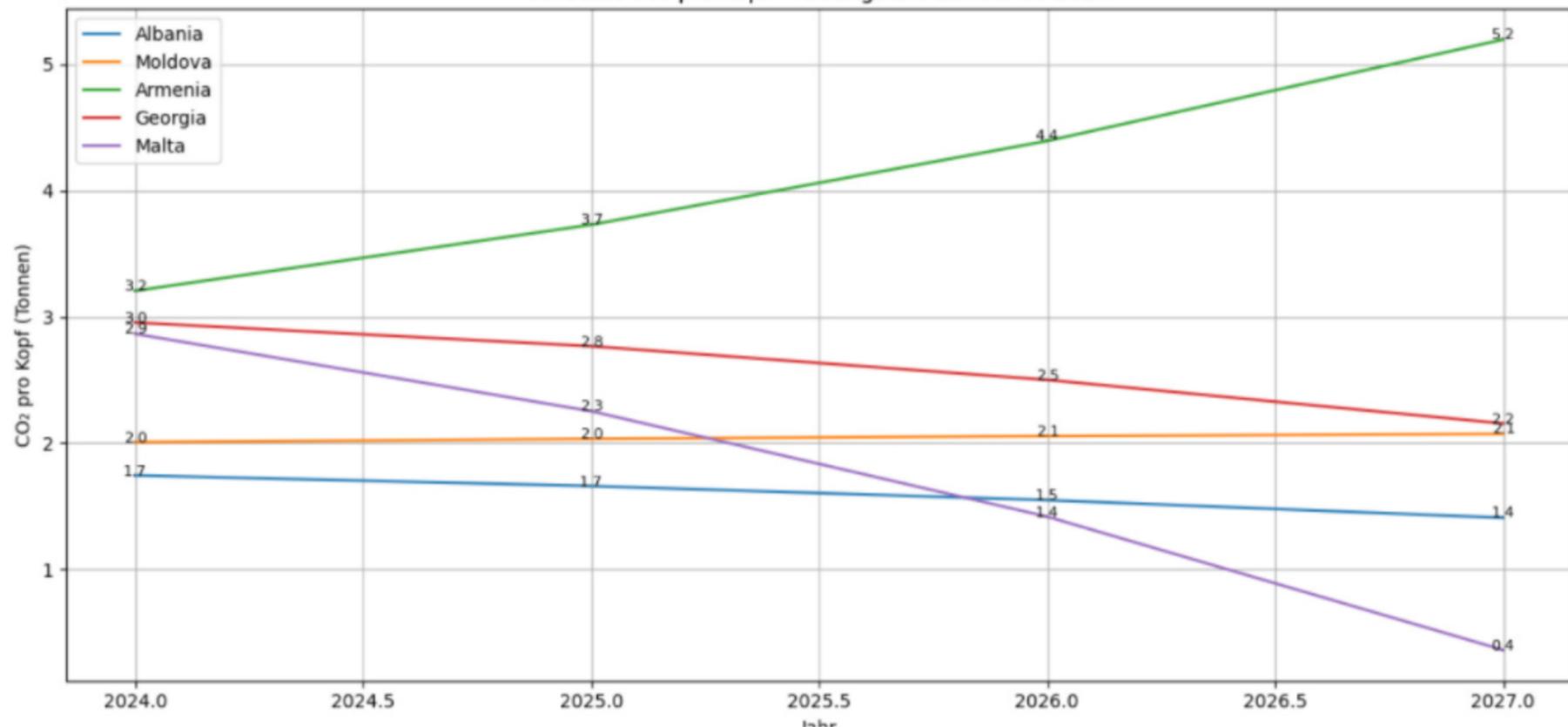
plt.plot(forecast['ds'], forecast['yhat'], ...)
plt.text(x, y, f"{y:.1f}", ...)

# Zahlen anzeigen
for i in range(-5, 0):
    x = forecast['ds'].iloc[i]
    y = forecast['yhat'].iloc[i]
    plt.text(x, y, f"{y:.1f}", fontsize=8, color='blue')

plt.title(f"● CO2 pro Kopf Forecast - {land}")
plt.xlabel("Jahr")
plt.ylabel("Tonnen CO2 / Kopf")
plt.grid(True)
plt.legend()
plt.tight_layout()
plt.show()
```



### Forecast CO<sub>2</sub> pro Kopf - Niedrigste 5 Länder ab 2024



\* Solarenergie-Erzeugung Forecast (Top 5 Länder, ab 2024)

