DAFTAR LAMPIRAN

Lampiran 1

Judul dataset: Data Emisi Karbon Indonesia dan Variabel Prediktor, 1990–2023

Jumlah observasi: 34 (deret waktu tahunan)

Variabel: 1 variabel target (CO_2 per kapita) dan 5 variabel prediktor

Keterangan Variabel dan Satuan

Variabel	Peran	Satuan	Sumber
Tahun	Indeks waktu	Tahun (YYYY)	—
CO_2 per Kapita	Target	ton CO_2 per kapita	World Bank Open Data (EN.ATM.CO2E.PC)
PDB per Kapita	Prediktor	USD konstan 2015	World Bank Open Data (NY.GDP.PCAP.KD)
Konsumsi Energi	Prediktor	TWh (energi primer)	Energy Institute: Statistical Review of World Energy
EBT	Prediktor	% (produksi listrik dari energi terbarukan)	World Bank Open Data (EG.ELC.RNEW.ZS)
Luas Hutan	Prediktor	Km^2	World Bank Open Data (AG.LND.FRST.K2)
Populasi	Prediktor	Jiwa	World Bank Open Data (SP.POP.TOTL)

Catatan: nilai EBT bersumber dari World Bank (EG.ELC.RNEW.ZS) dan telah divalidasi silang dengan referensi Energy Institute, dengan selisih di bawah 0,2%. Seluruh data merupakan data sekunder agregat nasional.

Tabel Data Lengkap (1990–2023)

No	Tahun	CO_2 per Kapita	PDB per Kapita	Konsumsi Energi	EBT	Luas Hutan	Populasi
1	1990	0,845	1.470,918	600,179	23,010	1.185.450,000	183.501.098
2	1991	0,936	1.544,996	652,938	22,205	1.168.185,000	186.778.238
3	1992	1,049	1.617,110	716,860	26,639	1.150.920,000	190.043.744
4	1993	1,113	1.693,107	763,843	25,142	1.133.655,000	193.305.168
5	1994	1,116	1.790,327	799,647	20,873	1.116.390,000	196.591.828
6	1995	1,113	1.905,542	870,284	19,412	1.099.125,000	199.888.057
7	1996	1,245	2.020,992	933,146	18,970	1.081.860,000	203.204.348
8	1997	1,355	2.081,842	1.012,695	12,157	1.064.595,000	206.536.095
9	1998	1,165	1.780,200	995,569	17,798	1.047.330,000	209.826.788
10	1999	1,371	1.767,515	1.086,986	17,054	1.030.065,000	213.004.668
11	2000	1,302	1.828,103	1.164,199	19,992	1.012.800,000	216.077.790
12	2001	1,447	1.868,592	1.246,685	20,902	1.011.179,200	219.097.902
13	2002	1,389	1.926,374	1.281,019	18,577	1.009.558,400	222.088.495
14	2003	1,508	1.991,918	1.391,678	16,884	1.007.937,600	225.048.008
15	2004	1,505	2.065,706	1.376,724	16,806	1.006.316,800	227.926.649
16	2005	1,506	2.155,448	1.416,892	17,052	1.004.696,000	230.871.650
17	2006	1,482	2.244,080	1.441,128	15,348	1.003.075,200	233.951.652
18	2007	1,636	2.355,153	1.546,681	16,368	1.001.454,400	237.062.337
19	2008	1,523	2.464,602	1.564,584	16,895	999.833,600	240.157.903
20	2009	1,640	2.546,220	1.607,271	16,449	998.212,800	243.220.028
21	2010	1,810	2.670,813	1.739,490	19,503	996.592,000	246.305.322
22	2011	2,007	2.799,625	1.837,594	15,875	987.329,400	249.470.032

23	2012	2,042	2.930,518	1.937,146	14,888	978.066,800	252.698.525
24	2013	1,911	3.055,242	1.824,511	15,765	968.804,200	255.852.467
25	2014	1,885	3.170,721	1.856,634	14,486	959.541,600	258.877.399
26	2015	2,059	3.288,223	1.896,461	13,795	950.279,000	261.799.249
27	2016	2,041	3.416,810	1.884,228	15,368	952.718,000	264.627.418
28	2017	2,083	3.553,520	2.023,711	16,247	939.498,000	267.346.658
29	2018	2,201	3.701,322	2.131,562	17,048	933.442,700	269.951.846
30	2019	2,399	3.850,903	2.253,939	16,260	927.387,300	272.489.381
31	2020	2,213	3.739,449	2.116,310	18,124	921.332,000	274.814.866
32	2021	2,239	3.850,689	2.196,681	18,167	915.276,633	276.758.053
33	2022	2,643	4.024,912	2.749,242	19,611	909.221,311	278.830.529
34	2023	2,608	4.192,652	2.785,135	18,597	903.165,959	281.190.067

Sumber: World Bank Open Data; Energy Institute — Statistical Review of World Energy. Data diakses dan dikompilasi untuk keperluan penelitian ini. Berkas dataset: https://docs.google.com/spreadsheets/d/1rtkDS-PVKVYKkZSeDEA0xxTnH_l6OOZm/edit?usp=sharing&ouid=106128372794553016544&rtpof=true&sd=true

Lampiran 2

Notebook ini memuat seluruh kode pengolahan data dan pengujian model penelitian. Setiap blok bersifat independen (mengunduh data dan mengimpor pustaka sendiri) sehingga dapat dijalankan terpisah dan tahan terhadap restart runtime. Blok 0–11 merupakan pengujian pada tahap sebelum ujian kelayakan; Blok 12 merupakan penambahan pada tahap revisi pasca-kelayakan.

Penunjuk "lihat Tabel X.X naskah" merujuk pada nomor tabel dalam naskah tesis.

Blok 0 — Setup & Verifikasi Data

Kode

```
import subprocess, sys
def _pip(p):
    try: __import__(p if p!="scikit-learn" else "sklearn")
    except ImportError: subprocess.run([sys.executable,"-m","pip","install","-q",p])
for _p in ["gdown","pandas","numpy","scikit-learn","xgboost"]: _pip(_p)

import gdown, pandas as pd, numpy as np, os
np.set_printoptions(suppress=True)
pd.set_option("display.float_format", lambda x: f"{x:.4f}")

FILE_ID = "1rtkDS-PVKVYKkZSeDEA0xxTnH_l6OOZm"
DATA = "Dataset_FINAL_fixed.xlsx"
if not os.path.exists(DATA):
    gdown.download(id=FILE_ID, output=DATA, quiet=True)

def load_data():
    d = pd.read_excel(DATA, skiprows=2, nrows=34)
    d.columns = ["Tahun","CO2_per_kapita","Luas_hutan","PDB_per_kapita",
                 "Populasi","Konsumsi_energi","EBT_persen"]
    d["Tahun"] = d["Tahun"].astype(int)
    return d

FEATURES = ["PDB_per_kapita","Konsumsi_energi","EBT_persen","Luas_hutan","Populasi"]
TARGET   = "CO2_per_kapita"
TEST_YEARS = list(range(2017,2024))

def splits(d):
    tr  = d[(d.Tahun>=1990)&(d.Tahun<=2011)]
    va  = d[(d.Tahun>=2012)&(d.Tahun<=2016)]
```

```
    tv  = d[(d.Tahun>=1990)&(d.Tahun<=2016)]
    te  = d[(d.Tahun>=2017)&(d.Tahun<=2023)]
    return tr, va, tv, te

def rmse(y,p):
    from sklearn.metrics import mean_squared_error
    return np.sqrt(mean_squared_error(y,p))

df = load_data()
print("Jumlah baris (n):", len(df))
print("Rentang tahun   :", df.Tahun.min(), "-", df.Tahun.max())
print("CO2 1990        :", df.iloc[0].CO2_per_kapita, "| PDB 1990:", round(df.iloc[0].PDB_
per_kapita,3))
print("\nAktual CO2 uji 2017-2023:", np.round(df[df.Tahun>=2017].CO2_per_kapita.values,3))
print("\nCuplikan data:")
print(df.head(3).to_string(index=False))
```

Keluaran

```
Jumlah baris (n): 34
Rentang tahun   : 1990 - 2023
CO2 1990        : 0.845 | PDB 1990: 1470.918

Aktual CO2 uji 2017-2023: [2.083 2.201 2.399 2.213 2.239 2.643 2.608]
```

Cuplikan data (3 baris pertama)

Tahun	CO2_per_kapita	Luas_hutan	PDB_per_kapita	Populasi	Konsumsi_energi	EBT_persen
1990	0.8450	1185450.0000	1470.9180	183501098	600.1790	23.0100
1991	0.9360	1168185.0000	1544.9960	186778238	652.9380	22.2050
1992	1.0490	1150920.0000	1617.1100	190043744	716.8600	26.6390

Blok 1 — Pembagian Data Tiga-Lapis

→ mendukung Tabel 3.3 naskah

Kode

```
import subprocess, sys
def _pip(p):
    try: __import__(p if p!="scikit-learn" else "sklearn")
    except ImportError: subprocess.run([sys.executable,"-m","pip","install","-q",p])
for _p in ["gdown","pandas","numpy","scikit-learn","xgboost"]: _pip(_p)

import gdown, pandas as pd, numpy as np, os
np.set_printoptions(suppress=True)
pd.set_option("display.float_format", lambda x: f"{x:.4f}")

FILE_ID = "1rtkDS-PVKVYKkZSeDEA0xxTnH_l6OOZm"
DATA = "Dataset_FINAL_fixed.xlsx"
if not os.path.exists(DATA):
    gdown.download(id=FILE_ID, output=DATA, quiet=True)

def load_data():
    d = pd.read_excel(DATA, skiprows=2, nrows=34)
    d.columns = ["Tahun","CO2_per_kapita","Luas_hutan","PDB_per_kapita",
                 "Populasi","Konsumsi_energi","EBT_persen"]
    d["Tahun"] = d["Tahun"].astype(int)
    return d

FEATURES = ["PDB_per_kapita","Konsumsi_energi","EBT_persen","Luas_hutan","Populasi"]
TARGET   = "CO2_per_kapita"
TEST_YEARS = list(range(2017,2024))

def splits(d):
    tr  = d[(d.Tahun>=1990)&(d.Tahun<=2011)]
    va  = d[(d.Tahun>=2012)&(d.Tahun<=2016)]
    tv  = d[(d.Tahun>=1990)&(d.Tahun<=2016)]
    te  = d[(d.Tahun>=2017)&(d.Tahun<=2023)]
    return tr, va, tv, te
```

```
def rmse(y,p):
    from sklearn.metrics import mean_squared_error
    return np.sqrt(mean_squared_error(y,p))

df = load_data()
tr, va, tv, te = splits(df)
tabel = pd.DataFrame({
    "Lapis": ["Latih","Validasi","Latih+Validasi","Uji"],
    "Periode": ["1990-2011","2012-2016","1990-2016","2017-2023"],
    "n": [len(tr), len(va), len(tv), len(te)],
    "Rentang CO2": [f"{d.CO2_per_kapita.min():.3f} - {d.CO2_per_kapita.max():.3f}"
                    for d in [tr,va,tv,te]]
})
print(tabel.to_string(index=False))
print("\nCatatan: batas atas rentang latih+validasi =", round(tv.CO2_per_kapita.max(),4),
      "-> acuan penting untuk analisis ekstrapolasi.")
```

Keluaran

```
Catatan: batas atas rentang latih+validasi = 2.059 -
> acuan penting untuk analisis ekstrapolasi.
```

Pembagian data tiga-lapis

Lapis	Periode	n	Rentang CO2
Latih	1990-2011	22	0.845 - 2.007
Validasi	2012-2016	5	1.885 - 2.059
Latih+Validasi	1990-2016	27	0.845 - 2.059
Uji	2017-2023	7	2.083 - 2.643

Blok 2 — Multikolinearitas (VIF, data latih n=22)

→ lihat Tabel 4.2 naskah

Kode

```
import subprocess, sys
def _pip(p):
    try: __import__(p if p!="scikit-learn" else "sklearn")
    except ImportError: subprocess.run([sys.executable,"-m","pip","install","-q",p])
for _p in ["gdown","pandas","numpy","scikit-learn","xgboost"]: _pip(_p)

import gdown, pandas as pd, numpy as np,os
np.set_printoptions(suppress=True)
pd.set_option("display.float_format", lambda x: f"{x:.4f}")

FILE_ID = "1rtkDS-PVKVYKkZSeDEA0xxTnH_l6OOZm"
DATA = "Dataset_FINAL_fixed.xlsx"
if not os.path.exists(DATA):
    gdown.download(id=FILE_ID, output=DATA, quiet=True)

def load_data():
    d = pd.read_excel(DATA, skiprows=2, nrows=34)
    d.columns = ["Tahun","CO2_per_kapita","Luas_hutan","PDB_per_kapita",
                 "Populasi","Konsumsi_energi","EBT_persen"]
    d["Tahun"] = d["Tahun"].astype(int)
    return d

FEATURES = ["PDB_per_kapita","Konsumsi_energi","EBT_persen","Luas_hutan","Populasi"]
TARGET   = "CO2_per_kapita"
TEST_YEARS = list(range(2017,2024))

def splits(d):
    tr  = d[(d.Tahun>=1990)&(d.Tahun<=2011)]
    va  = d[(d.Tahun>=2012)&(d.Tahun<=2016)]
    tv  = d[(d.Tahun>=1990)&(d.Tahun<=2016)]
    te  = d[(d.Tahun>=2017)&(d.Tahun<=2023)]
    return tr, va, tv, te
```

```
def rmse(y,p):
    from sklearn.metrics import mean_squared_error
    return np.sqrt(mean_squared_error(y,p))

from statsmodels.stats.outliers_influence import variance_inflation_factor
from statsmodels.tools.tools import add_constant
df = load_data(); tr,_,_,_ = splits(df)
X = add_constant(tr[FEATURES])
vif = pd.DataFrame({
    "Fitur": FEATURES,
    "VIF": [variance_inflation_factor(X.values, i+1) for i in range(len(FEATURES))]
}).sort_values("VIF", ascending=False)
print("### T3 — VIF (data latih 1990-2011, n=22)")
print(vif.to_string(index=False))
```

VIF (data latih 1990-2011, n=22)

Fitur	VIF
Populasi	160.1330
Konsumsi_energi	138.4777
Luas_hutan	17.2221
PDB_per_kapita	12.8304
EBT_persen	2.6785

Blok 3 — XGBoost Baseline (5 fitur, hold-out)

→ lihat Tabel 4.4 & 4.5 naskah

Kode

```
import subprocess, sys
def _pip(p):
    try: __import__(p if p!="scikit-learn" else "sklearn")
    except ImportError: subprocess.run([sys.executable,"-m","pip","install","-q",p])
for _p in ["gdown","pandas","numpy","scikit-learn","xgboost"]: _pip(_p)

import gdown, pandas as pd, numpy as np, os
np.set_printoptions(suppress=True)
pd.set_option("display.float_format", lambda x: f"{x:.4f}")

FILE_ID = "1rtkDS-PVKVYKkZSeDEA0xxTnH_l6OOZm"
DATA = "Dataset_FINAL_fixed.xlsx"
if not os.path.exists(DATA):
    gdown.download(id=FILE_ID, output=DATA, quiet=True)

def load_data():
    d = pd.read_excel(DATA, skiprows=2, nrows=34)
    d.columns = ["Tahun","CO2_per_kapita","Luas_hutan","PDB_per_kapita",
                 "Populasi","Konsumsi_energi","EBT_persen"]
    d["Tahun"] = d["Tahun"].astype(int)
    return d

FEATURES = ["PDB_per_kapita","Konsumsi_energi","EBT_persen","Luas_hutan","Populasi"]
TARGET   = "CO2_per_kapita"
TEST_YEARS = list(range(2017,2024))

def splits(d):
    tr  = d[(d.Tahun>=1990)&(d.Tahun<=2011)]
    va  = d[(d.Tahun>=2012)&(d.Tahun<=2016)]
    tv  = d[(d.Tahun>=1990)&(d.Tahun<=2016)]
    te  = d[(d.Tahun>=2017)&(d.Tahun<=2023)]
    return tr, va, tv, te

def rmse(y,p):
    from sklearn.metrics import mean_squared_error
    return np.sqrt(mean_squared_error(y,p))
```

```
from xgboost import XGBRegressor
from sklearn.metrics import r2_score
df = load_data(); tr,va,tv,te = splits(df)
Xtv, ytv = tv[FEATURES].values, tv[TARGET].values
Xte, yte = te[FEATURES].values, te[TARGET].values
m = XGBRegressor(random_state=42); m.fit(Xtv, ytv)
pred = m.predict(Xte)
print("XGBoost baseline (konfigurasi standar, refit train+val)")
print(f"  Test  RMSE = {rmse(yte,pred):.4f}")
print(f"  Test  R2   = {r2_score(yte,pred):+.4f}")
print(f"  in-sample R2 = {r2_score(ytv, m.predict(Xtv)):.6f}")
print(f"  Prediksi uji : {np.round(pred,3)}")
print(f"  Aktual   uji : {np.round(yte,3)}")
print(f"  Batas atas latih+val = {ytv.max():.4f}  (prediksi terkunci di sekitar nilai ini)
")
```

Keluaran

```
XGBoost baseline (konfigurasi standar, refit train+val)
  Test  RMSE = 0.3567
  Test  R2   = -2.1939
  in-sample R2 = 0.999994
  Prediksi uji : [2.045 2.045 2.045 2.045 2.045 2.045 2.045]
  Aktual   uji : [2.083 2.201 2.399 2.213 2.239 2.643 2.608]
  Batas atas latih+val = 2.0590  (prediksi terkunci di sekitar nilai ini)
```

Blok 4 — XGBoost + RFE (RFECV, estimator Random Forest)

→ lihat Tabel 4.4 & 4.5 naskah

Kode

```
import subprocess, sys
def _pip(p):
    try: __import__(p if p!="scikit-learn" else "sklearn")
    except ImportError: subprocess.run([sys.executable,"-m","pip","install","-q",p])
for _p in ["gdown","pandas","numpy","scikit-learn","xgboost"]: _pip(_p)

import gdown, pandas as pd, numpy as np, os
np.set_printoptions(suppress=True)
pd.set_option("display.float_format", lambda x: f"{x:.4f}")

FILE_ID = "1rtkDS-PVKVYKkZSeDEA0xxTnH_l6OOZm"
DATA = "Dataset_FINAL_fixed.xlsx"
if not os.path.exists(DATA):
    gdown.download(id=FILE_ID, output=DATA, quiet=True)

def load_data():
    d = pd.read_excel(DATA, skiprows=2, nrows=34)
    d.columns = ["Tahun","CO2_per_kapita","Luas_hutan","PDB_per_kapita",
                 "Populasi","Konsumsi_energi","EBT_persen"]
    d["Tahun"] = d["Tahun"].astype(int)
    return d

FEATURES = ["PDB_per_kapita","Konsumsi_energi","EBT_persen","Luas_hutan","Populasi"]
TARGET   = "CO2_per_kapita"
TEST_YEARS = list(range(2017,2024))

def splits(d):
    tr  = d[(d.Tahun>=1990)&(d.Tahun<=2011)]
    va  = d[(d.Tahun>=2012)&(d.Tahun<=2016)]
    tv  = d[(d.Tahun>=1990)&(d.Tahun<=2016)]
    te  = d[(d.Tahun>=2017)&(d.Tahun<=2023)]
    return tr, va, tv, te

def rmse(y,p):
    from sklearn.metrics import mean_squared_error
    return np.sqrt(mean_squared_error(y,p))

from xgboost import XGBRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.feature_selection import RFECV
```

```
from sklearn.model_selection import TimeSeriesSplit
from sklearn.metrics import r2_score
df = load_data(); tr,va,tv,te = splits(df)
Xtv, ytv = tv[FEATURES].values, tv[TARGET].values
Xte, yte = te[FEATURES].values, te[TARGET].values

sel = RFECV(RandomForestRegressor(random_state=42),
            step=1, cv=TimeSeriesSplit(n_splits=5),
            scoring="neg_root_mean_squared_error", min_features_to_select=1)
sel.fit(Xtv, ytv)
chosen = [f for f,keep in zip(FEATURES, sel.support_) if keep]
print("Fitur terpilih RFECV:", chosen)

m = XGBRegressor(random_state=42); m.fit(Xtv[:,sel.support_], ytv)
pred = m.predict(Xte[:,sel.support_])
print(f"\nXGBoost + RFE ({len(chosen)} fitur)")
print(f"  Test  RMSE = {rmse(yte,pred):.4f}")
print(f"  Test  R2   = {r2_score(yte,pred):+.4f}")
print(f"  in-sample R2 = {r2_score(ytv, m.predict(Xtv[:,sel.support_])):.6f}")
print(f"  Prediksi uji : {np.round(pred,3)}")
print(f"  Aktual   uji : {np.round(yte,3)}")
```

Keluaran

```
Fitur terpilih RFECV: ['Konsumsi_energi']

XGBoost + RFE (1 fitur)
  Test  RMSE = 0.3591
  Test  R2   = -2.2377
  in-sample R2 = 0.999982
  Prediksi uji : [2.042 2.042 2.042 2.042 2.042 2.042 2.042]
  Aktual   uji : [2.083 2.201 2.399 2.213 2.239 2.643 2.608]
```

Blok 5 — MLR Baseline (Regresi Linear Berganda, 5 fitur)

→ lihat Tabel 4.4 & 4.5 naskah

Kode

```
import subprocess, sys
def _pip(p):
    try: __import__(p if p!="scikit-learn" else "sklearn")
    except ImportError: subprocess.run([sys.executable,"-m","pip","install","-q",p])
for _p in ["gdown","pandas","numpy","scikit-learn","xgboost"]: _pip(_p)

import gdown, pandas as pd, numpy as np, os
np.set_printoptions(suppress=True)
pd.set_option("display.float_format", lambda x: f"{x:.4f}")

FILE_ID = "1rtkDS-PVKVYKkZSeDEA0xxTnH_l6OOZm"
DATA = "Dataset_FINAL_fixed.xlsx"
if not os.path.exists(DATA):
    gdown.download(id=FILE_ID, output=DATA, quiet=True)

def load_data():
    d = pd.read_excel(DATA, skiprows=2, nrows=34)
    d.columns = ["Tahun","CO2_per_kapita","Luas_hutan","PDB_per_kapita",
                 "Populasi","Konsumsi_energi","EBT_persen"]
    d["Tahun"] = d["Tahun"].astype(int)
    return d

FEATURES = ["PDB_per_kapita","Konsumsi_energi","EBT_persen","Luas_hutan","Populasi"]
TARGET   = "CO2_per_kapita"
TEST_YEARS = list(range(2017,2024))

def splits(d):
    tr  = d[(d.Tahun>=1990)&(d.Tahun<=2011)]
    va  = d[(d.Tahun>=2012)&(d.Tahun<=2016)]
    tv  = d[(d.Tahun>=1990)&(d.Tahun<=2016)]
    te  = d[(d.Tahun>=2017)&(d.Tahun<=2023)]
    return tr, va, tv, te
```

```
def rmse(y,p):
    from sklearn.metrics import mean_squared_error
    return np.sqrt(mean_squared_error(y,p))

from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
df = load_data(); tr,va,tv,te = splits(df)
Xtv, ytv = tv[FEATURES].values, tv[TARGET].values
Xte, yte = te[FEATURES].values, te[TARGET].values
m = LinearRegression().fit(Xtv, ytv)
pred = m.predict(Xte)
print("MLR baseline (5 fitur)")
print(f"  Test  RMSE = {rmse(yte,pred):.4f}")
print(f"  Test  R2   = {r2_score(yte,pred):+.4f}")
print(f"  in-sample R2 = {r2_score(ytv, m.predict(Xtv)):.6f}")
print(f"  Prediksi uji : {np.round(pred,3)}")
print(f"  Aktual   uji : {np.round(yte,3)}")
print(f"  Prediksi 2022: {pred[te.Tahun.values==2022][0]:.3f}  (aktual {yte[te.Tahun.value
s==2022][0]:.3f})")
```

Keluaran

```
MLR baseline (5 fitur)
  Test  RMSE = 0.3200
  Test  R2   = -1.5712
  in-sample R2 = 0.981851
  Prediksi uji : [2.196 2.356 2.528 2.276 2.396 3.15  3.222]
  Aktual   uji : [2.083 2.201 2.399 2.213 2.239 2.643 2.608]
  Prediksi 2022: 3.150  (aktual 2.643)
```

Blok 6 — Rekayasa Fitur: Polynomial & PCA

→ lihat Tabel 4.8 naskah

Kode

```
import subprocess, sys
def _pip(p):
    try: __import__(p if p!="scikit-learn" else "sklearn")
    except ImportError: subprocess.run([sys.executable,"-m","pip","install","-q",p])
for _p in ["gdown","pandas","numpy","scikit-learn","xgboost"]: _pip(_p)

import gdown, pandas as pd, numpy as np, os
np.set_printoptions(suppress=True)
pd.set_option("display.float_format", lambda x: f"{x:.4f}")

FILE_ID = "1rtkDS-PVKVYKkZSeDEA0xxTnH_16OOZm"
DATA = "Dataset_FINAL_fixed.xlsx"
if not os.path.exists(DATA):
    gdown.download(id=FILE_ID, output=DATA, quiet=True)

def load_data():
    d = pd.read_excel(DATA, skiprows=2, nrows=34)
    d.columns = ["Tahun","CO2_per_kapita","Luas_hutan","PDB_per_kapita",
                 "Populasi","Konsumsi_energi","EBT_persen"]
    d["Tahun"] = d["Tahun"].astype(int)
    return d

FEATURES = ["PDB_per_kapita","Konsumsi_energi","EBT_persen","Luas_hutan","Populasi"]
TARGET   = "CO2_per_kapita"
TEST_YEARS = list(range(2017,2024))

def splits(d):
    tr  = d[(d.Tahun>=1990)&(d.Tahun<=2011)]
    va  = d[(d.Tahun>=2012)&(d.Tahun<=2016)]
    tv  = d[(d.Tahun>=1990)&(d.Tahun<=2016)]
    te  = d[(d.Tahun>=2017)&(d.Tahun<=2023)]
    return tr, va, tv, te

def rmse(y,p):
    from sklearn.metrics import mean_squared_error
    return np.sqrt(mean_squared_error(y,p))
```

```
from sklearn.preprocessing import PolynomialFeatures, StandardScaler
from sklearn.decomposition import PCA
from sklearn.pipeline import make_pipeline
from sklearn.linear_model import LinearRegression
from xgboost import XGBRegressor
from sklearn.metrics import r2_score
df = load_data(); tr,va,tv,te = splits(df)
Xtv, ytv = tv[FEATURES].values, tv[TARGET].values
Xte, yte = te[FEATURES].values, te[TARGET].values

def evalmodel(pipe):
    pipe.fit(Xtv,ytv); p=pipe.predict(Xte)
    return rmse(yte,p), r2_score(yte,p), np.round(p,3)

configs = {
 "XGB+Poly": make_pipeline(PolynomialFeatures(2, include_bias=False), XGBRegressor(random_
state=42)),
 "MLR+Poly": make_pipeline(PolynomialFeatures(2, include_bias=False), LinearRegression()),
 "XGB+PCA" : make_pipeline(StandardScaler(), PCA(n_components=3), XGBRegressor(random_stat
e=42)),
 "MLR+PCA" : make_pipeline(StandardScaler(), PCA(n_components=3), LinearRegression()),
}
rows=[]; permodel={}
for name,pipe in configs.items():
    r,r2,p=evalmodel(pipe); rows.append([name,r,r2]); permodel[name]=p

print("### Ringkas metrik rekayasa fitur (untuk T1)")
print(pd.DataFrame(rows, columns=["Model","Test RMSE","Test R2"]).to_string(index=False))
print("\n### T7 — Prediksi per-tahun rekayasa fitur vs aktual")
T7=pd.DataFrame({"Tahun":te.Tahun.values,"Aktual":np.round(yte,3), **permodel})
print(T7.to_string(index=False))
```

Ringkasan metrik rekayasa fitur

Model	Test RMSE	Test R2
XGB+Poly	0.3482	-2.0433
MLR+Poly	1.0120	-24.7135
XGB+PCA	0.3613	-2.2767
MLR+PCA	0.0848	0.8193

Prediksi per-tahun rekayasa fitur vs aktual

Tahun	Aktual	XGB+Poly	MLR+Poly	XGB+PCA	MLR+PCA
2017	2.0830	2.0560	2.5950	2.0400	2.1560
2018	2.2010	2.0560	3.2170	2.0400	2.2290
2019	2.3990	2.0560	3.9940	2.0400	2.3010
2020	2.2130	2.0560	2.3620	2.0400	2.2640
2021	2.2390	2.0560	2.8650	2.0400	2.3180
2022	2.6430	2.0560	3.1800	2.0400	2.4930
2023	2.6080	2.0560	4.2290	2.0400	2.5490

Blok 7 — Model Pendukung: Ridge, SVR-linear, SVR-RBF

→ lihat Tabel 4.5 & 4.11 naskah

Kode

```
import subprocess, sys
def _pip(p):
    try: __import__(p if p!="scikit-learn" else "sklearn")
    except ImportError: subprocess.run([sys.executable,"-m","pip","install","-q",p])
for _p in ["gdown","pandas","numpy","scikit-learn","xgboost"]: _pip(_p)

import gdown, pandas as pd, numpy as np, os
np.set_printoptions(suppress=True)
```

```
pd.set_option("display.float_format", lambda x: f"{x:.4f}")

FILE_ID = "1rtkDS-PVKVYKkZSeDEA0xxTnH_l6OOZm"
DATA = "Dataset_FINAL_fixed.xlsx"
if not os.path.exists(DATA):
    gdown.download(id=FILE_ID, output=DATA, quiet=True)

def load_data():
    d = pd.read_excel(DATA, skiprows=2, nrows=34)
    d.columns = ["Tahun","CO2_per_kapita","Luas_hutan","PDB_per_kapita",
                 "Populasi","Konsumsi_energi","EBT_persen"]
    d["Tahun"] = d["Tahun"].astype(int)
    return d

FEATURES = ["PDB_per_kapita","Konsumsi_energi","EBT_persen","Luas_hutan","Populasi"]
TARGET   = "CO2_per_kapita"
TEST_YEARS = list(range(2017,2024))

def splits(d):
    tr  = d[(d.Tahun>=1990)&(d.Tahun<=2011)]
    va  = d[(d.Tahun>=2012)&(d.Tahun<=2016)]
    tv  = d[(d.Tahun>=1990)&(d.Tahun<=2016)]
    te  = d[(d.Tahun>=2017)&(d.Tahun<=2023)]
    return tr, va, tv, te

def rmse(y,p):
    from sklearn.metrics import mean_squared_error
    return np.sqrt(mean_squared_error(y,p))

from sklearn.linear_model import Ridge
from sklearn.svm import SVR
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import make_pipeline
from sklearn.metrics import r2_score
df = load_data(); tr,va,tv,te = splits(df)
Xtv, ytv = tv[FEATURES].values, tv[TARGET].values
Xte, yte = te[FEATURES].values, te[TARGET].values
GRIDV=[0.01,0.1,1,10,100]

def fit_eval(model):
    pipe=make_pipeline(StandardScaler(), model).fit(Xtv,ytv); p=pipe.predict(Xte)
    return rmse(yte,p), r2_score(yte,p), np.round(p,3)

# --- T10: rentang metrik lintas hyperparameter ---
def sweep(builder):
    rs=[]; r2s=[]
    for v in GRIDV:
        r,r2,_=fit_eval(builder(v)); rs.append(r); r2s.append(r2)
    return min(rs),max(rs),min(r2s),max(r2s)

sweeps={
 "Ridge (alpha)":     sweep(lambda v: Ridge(alpha=v)),
 "SVR-linear (C)":    sweep(lambda v: SVR(kernel="linear", C=v)),
 "SVR-RBF (C)":       sweep(lambda v: SVR(kernel="rbf", C=v)),
}
print("### T10 — Rentang metrik lintas hyperparameter (eksplorasi 5 nilai)")
print(pd.DataFrame([[k,f"{a:.3f}-
{b:.3f}",f"{c:+.3f}..{d:+.3f}"] for k,(a,b,c,d) in sweeps.items()],
      columns=["Model","Rentang RMSE","Rentang R2"]).to_string(index=False))

# --- T8: prediksi per-tahun pada konfigurasi acuan (alpha=1, C=1) ---
permodel={}
for name,mdl in [("Ridge (alpha=1)",Ridge(alpha=1)),
                 ("SVR-linear (C=1)",SVR(kernel="linear",C=1)),
                 ("SVR-RBF (C=1)",SVR(kernel="rbf",C=1))]:
    r,r2,p=fit_eval(mdl); permodel[name]=p
print("\n### T8 — Prediksi per-tahun model pendukung vs aktual")
T8=pd.DataFrame({"Tahun":te.Tahun.values,"Aktual":np.round(yte,3), **permodel})
print(T8.to_string(index=False))
print("\nCatatan: SVR-RBF (kernel lokal) menurun saat aktual naik -
> kehilangan acuan di luar")
```

```
print("rentang latih; SVR-
linear (kernel global) mengikuti tren. Kontrol kernel global vs lokal.")
```

Keluaran

```
Catatan: SVR-RBF (kernel lokal) menurun saat aktual naik -> kehilangan acuan di luar
rentang latih; SVR-linear (kernel global) mengikuti tren. Kontrol kernel global vs lokal.
```

Rentang metrik lintas hyperparameter (eksplorasi 5 nilai)

Model	Rentang RMSE	Rentang R2
Ridge (alpha)	0.064-0.501	-5.299..+0.896
SVR-linear (C)	0.047-0.260	-0.697..+0.945
SVR-RBF (C)	0.734-0.892	-18.966..-12.524

Prediksi per-tahun model pendukung vs aktual

Tahun	Aktual	Ridge (alpha=1)	SVR-linear (C=1)	SVR-RBF (C=1)
2017	2.0830	2.1350	2.1320	1.8820
2018	2.2010	2.2120	2.2150	1.7920
2019	2.3990	2.2950	2.3220	1.7220
2020	2.2130	2.2300	2.1980	1.7220
2021	2.2390	2.2890	2.2650	1.6680
2022	2.6430	2.5330	2.6230	1.5070
2023	2.6080	2.5880	2.6810	1.4910

Blok 8 — Walk-Forward (MLR & XGBoost)

→ lihat Tabel 4.10 naskah

Kode

```
import subprocess, sys
def _pip(p):
    try: __import__(p if p!="scikit-learn" else "sklearn")
    except ImportError: subprocess.run([sys.executable,"-m","pip","install","-q",p])
for _p in ["gdown","pandas","numpy","scikit-learn","xgboost"]: _pip(_p)

import gdown, pandas as pd, numpy as np, os
np.set_printoptions(suppress=True)
pd.set_option("display.float_format", lambda x: f"{x:.4f}")

FILE_ID = "1rtkDS-PVKVYKkZSeDEA0xxTnH_16OOZm"
DATA = "Dataset_FINAL_fixed.xlsx"
if not os.path.exists(DATA):
    gdown.download(id=FILE_ID, output=DATA, quiet=True)

def load_data():
    d = pd.read_excel(DATA, skiprows=2, nrows=34)
    d.columns = ["Tahun","CO2_per_kapita","Luas_hutan","PDB_per_kapita",
                 "Populasi","Konsumsi_energi","EBT_persen"]
    d["Tahun"] = d["Tahun"].astype(int)
    return d

FEATURES = ["PDB_per_kapita","Konsumsi_energi","EBT_persen","Luas_hutan","Populasi"]
TARGET   = "CO2_per_kapita"
TEST_YEARS = list(range(2017,2024))

def splits(d):
    tr  = d[(d.Tahun>=1990)&(d.Tahun<=2011)]
    va  = d[(d.Tahun>=2012)&(d.Tahun<=2016)]
    tv  = d[(d.Tahun>=1990)&(d.Tahun<=2016)]
    te  = d[(d.Tahun>=2017)&(d.Tahun<=2023)]
    return tr, va, tv, te

def rmse(y,p):
    from sklearn.metrics import mean_squared_error
    return np.sqrt(mean_squared_error(y,p))
```

```
from sklearn.linear_model import LinearRegression
from xgboost import XGBRegressor
from sklearn.metrics import r2_score
df = load_data()

def walk_forward(builder):
    preds=[]; acts=[]; yrs=[]
    for yr in TEST_YEARS:
        tr = df[df.Tahun < yr]
        row= df[df.Tahun==yr]
        m=builder(); m.fit(tr[FEATURES].values, tr[TARGET].values)
        preds.append(m.predict(row[FEATURES].values)[0])
        acts.append(row[TARGET].values[0]); yrs.append(yr)
    return np.array(yrs),np.array(acts),np.array(preds)

res={}
for name,builder in [("MLR",lambda:LinearRegression()),
                     ("XGBoost",lambda:XGBRegressor(random_state=42))]:
    yrs,acts,preds=walk_forward(builder); res[name]=preds
    print(f"{name} walk-
forward  RMSE={rmse(acts,preds):.4f}  R2={r2_score(acts,preds):+.4f}")

print("\n### T5 — Walk-forward prediksi per-tahun vs aktual")
T5=pd.DataFrame({"Tahun":yrs,"Aktual":np.round(acts,3),
                 "MLR":np.round(res['MLR'],3),"XGBoost":np.round(res['XGBoost'],3)})
print(T5.to_string(index=False))
print("\nCatatan: dengan data latih yang terus bertambah, XGBoost naik mengikuti tren")
print("(bukti bahwa yang menolong adalah perluasan RENTANG data latih, bukan skema per se)
.")
```

Keluaran

```
MLR walk-forward  RMSE=0.1400  R2=+0.5081
XGBoost walk-forward  RMSE=0.1295  R2=+0.5793

Catatan: dengan data latih yang terus bertambah, XGBoost naik mengikuti tren
(bukti bahwa yang menolong adalah perluasan RENTANG data latih, bukan skema per se).
```

Walk-forward prediksi per-tahun vs aktual

Tahun	Aktual	MLR	XGBoost
2017	2.0830	2.1960	2.0450
2018	2.2010	2.3060	2.0820
2019	2.3990	2.4170	2.2000
2020	2.2130	2.1930	2.2010
2021	2.2390	2.2990	2.2130
2022	2.6430	2.9320	2.3980
2023	2.6080	2.7680	2.6420

Blok 9 — Univariat Lag-1 (Regresi Linear, Naive, XGBoost)

→ lihat Tabel 4.9 naskah

Kode

```
import subprocess, sys
def _pip(p):
    try: __import__(p if p!="scikit-learn" else "sklearn")
    except ImportError: subprocess.run([sys.executable,"-m","pip","install","-q",p])
for _p in ["gdown","pandas","numpy","scikit-learn","xgboost"]: _pip(_p)

import gdown, pandas as pd, numpy as np, os
np.set_printoptions(suppress=True)
pd.set_option("display.float_format", lambda x: f"{x:.4f}")

FILE_ID = "1rtkDS-PVKVYKkZSeDEA0xxTnH_l6OOZm"
DATA = "Dataset_FINAL_fixed.xlsx"
if not os.path.exists(DATA):
```

```
    gdown.download(id=FILE_ID, output=DATA, quiet=True)

def load_data():
    d = pd.read_excel(DATA, skiprows=2, nrows=34)
    d.columns = ["Tahun","CO2_per_kapita","Luas_hutan","PDB_per_kapita",
                 "Populasi","Konsumsi_energi","EBT_persen"]
    d["Tahun"] = d["Tahun"].astype(int)
    return d

FEATURES = ["PDB_per_kapita","Konsumsi_energi","EBT_persen","Luas_hutan","Populasi"]
TARGET   = "CO2_per_kapita"
TEST_YEARS = list(range(2017,2024))

def splits(d):
    tr  = d[(d.Tahun>=1990)&(d.Tahun<=2011)]
    va  = d[(d.Tahun>=2012)&(d.Tahun<=2016)]
    tv  = d[(d.Tahun>=1990)&(d.Tahun<=2016)]
    te  = d[(d.Tahun>=2017)&(d.Tahun<=2023)]
    return tr, va, tv, te

def rmse(y,p):
    from sklearn.metrics import mean_squared_error
    return np.sqrt(mean_squared_error(y,p))

from sklearn.linear_model import LinearRegression
from xgboost import XGBRegressor
from sklearn.metrics import r2_score
df = load_data().sort_values("Tahun").reset_index(drop=True)
df["lag1"] = df[TARGET].shift(1)
d = df.dropna().reset_index(drop=True)
tv = d[d.Tahun<=2016]; te = d[d.Tahun>=2017]
Xtv,ytv = tv[["lag1"]].values, tv[TARGET].values
Xte,yte = te[["lag1"]].values, te[TARGET].values

reg = LinearRegression().fit(Xtv,ytv); p_reg=reg.predict(Xte)
p_naive = te["lag1"].values
xgb = XGBRegressor(random_state=42).fit(Xtv,ytv); p_xgb=xgb.predict(Xte)

for name,p in [("Regresi linear",p_reg),("Naive (t-1)",p_naive),("XGBoost",p_xgb)]:
    print(f"{name:16s} RMSE={rmse(yte,p):.4f}  R2={r2_score(yte,p):+.4f}")

print("\n### T6 — Univariat lag-1 prediksi per-tahun vs aktual")
T6=pd.DataFrame({"Tahun":te.Tahun.values,"Aktual":np.round(yte,3),
                 "Regresi linear":np.round(p_reg,3),"Naive":np.round(p_naive,3),
                 "XGBoost":np.round(p_xgb,3)})
print(T6.to_string(index=False))
```

Keluaran

```
Regresi linear   RMSE=0.1895  R2=+0.0985
Naive (t-1)      RMSE=0.1907  R2=+0.0868
XGBoost          RMSE=0.3612  R2=-2.2747
```

Univariat lag-1 prediksi per-tahun vs aktual

Tahun	Aktual	Regresi linear	Naive	XGBoost
2017	2.0830	2.0490	2.0410	2.0410
2018	2.2010	2.0880	2.0830	2.0400
2019	2.3990	2.1980	2.2010	2.0400
2020	2.2130	2.3830	2.3990	2.0400
2021	2.2390	2.2100	2.2130	2.0400
2022	2.6430	2.2340	2.2390	2.0400
2023	2.6080	2.6110	2.6430	2.0400

Blok 10 — Perbandingan Faktor Dominan: XGBoost importances vs RFE-RF ranking

→ *lihat Tabel 4.7 naskah*

Menjawab tujuan utama tesis (identifikasi faktor dominan). Membandingkan `feature_importances_`

XGBoost dengan peringkat penghapusan RFE (estimator Random Forest).

Angka blok ini baru dihasilkan — verifikasi silang dengan lingkungan Colab asli sebelum masuk naskah.

Kode

```
import subprocess, sys
def _pip(p):
    try: __import__(p if p!="scikit-learn" else "sklearn")
    except ImportError: subprocess.run([sys.executable,"-m","pip","install","-q",p])
for _p in ["gdown","pandas","numpy","scikit-learn","xgboost"]: _pip(_p)

import gdown, pandas as pd, numpy as np, os
np.set_printoptions(suppress=True)
pd.set_option("display.float_format", lambda x: f"{x:.4f}")

FILE_ID = "1rtkDS-PVKVYKkZSeDEA0xxTnH_l6OOZm"
DATA = "Dataset_FINAL_fixed.xlsx"
if not os.path.exists(DATA):
    gdown.download(id=FILE_ID, output=DATA, quiet=True)

def load_data():
    d = pd.read_excel(DATA, skiprows=2, nrows=34)
    d.columns = ["Tahun","CO2_per_kapita","Luas_hutan","PDB_per_kapita",
                 "Populasi","Konsumsi_energi","EBT_persen"]
    d["Tahun"] = d["Tahun"].astype(int)
    return d

FEATURES = ["PDB_per_kapita","Konsumsi_energi","EBT_persen","Luas_hutan","Populasi"]
TARGET   = "CO2_per_kapita"
TEST_YEARS = list(range(2017,2024))

def splits(d):
    tr  = d[(d.Tahun>=1990)&(d.Tahun<=2011)]
    va  = d[(d.Tahun>=2012)&(d.Tahun<=2016)]
    tv  = d[(d.Tahun>=1990)&(d.Tahun<=2016)]
    te  = d[(d.Tahun>=2017)&(d.Tahun<=2023)]
    return tr, va, tv, te

def rmse(y,p):
    from sklearn.metrics import mean_squared_error
    return np.sqrt(mean_squared_error(y,p))

from xgboost import XGBRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.feature_selection import RFE
df = load_data(); tr,va,tv,te = splits(df)
Xtv, ytv = tv[FEATURES].values, tv[TARGET].values

xgb = XGBRegressor(random_state=42).fit(Xtv,ytv)
imp = xgb.feature_importances_

rfe = RFE(RandomForestRegressor(random_state=42), n_features_to_select=1).fit(Xtv,ytv)
rank = rfe.ranking_   # 1 = paling penting (bertahan terakhir)

T9 = pd.DataFrame({
    "Fitur": FEATURES,
    "XGBoost importance": np.round(imp,4),
    "Peringkat XGBoost": pd.Series(imp).rank(ascending=False).astype(int).values,
    "Peringkat RFE-RF": rank
}).sort_values("Peringkat RFE-RF")
print("### T9 — Faktor dominan: XGBoost importances vs RFE-RF ranking")
print(T9.to_string(index=False))
```

Faktor dominan: XGBoost importances vs RFE-RF ranking

Fitur	XGBoost importance	Peringkat XGBoost	Peringkat RFE-RF
Konsumsi_energi	0.2936	2	1
Luas_hutan	0.0001	4	2
Populasi	0.0000	5	3
PDB_per_kapita	0.7002	1	4
EBT_persen	0.0061	3	5

Blok 11 — Rekap Metrik Seluruh Model

→ lihat Tabel 4.4 & 4.5 naskah

Blok konsolidasi. Menjalankan ulang seluruh model dan menyusun dua tabel utama:

Kode

```
import subprocess, sys
def _pip(p):
    try: __import__(p if p!="scikit-learn" else "sklearn")
    except ImportError: subprocess.run([sys.executable,"-m","pip","install","-q",p])
for _p in ["gdown","pandas","numpy","scikit-learn","xgboost"]: _pip(_p)

import gdown, pandas as pd, numpy as np, os
np.set_printoptions(suppress=True)
pd.set_option("display.float_format", lambda x: f"{x:.4f}")

FILE_ID = "1rtkDS-PVKVYKkZSeDEA0xxTnH_l6OOZm"
DATA = "Dataset_FINAL_fixed.xlsx"
if not os.path.exists(DATA):
    gdown.download(id=FILE_ID, output=DATA, quiet=True)

def load_data():
    d = pd.read_excel(DATA, skiprows=2, nrows=34)
    d.columns = ["Tahun","CO2_per_kapita","Luas_hutan","PDB_per_kapita",
                 "Populasi","Konsumsi_energi","EBT_persen"]
    d["Tahun"] = d["Tahun"].astype(int)
    return d

FEATURES = ["PDB_per_kapita","Konsumsi_energi","EBT_persen","Luas_hutan","Populasi"]
TARGET   = "CO2_per_kapita"
TEST_YEARS = list(range(2017,2024))

def splits(d):
    tr  = d[(d.Tahun>=1990)&(d.Tahun<=2011)]
    va  = d[(d.Tahun>=2012)&(d.Tahun<=2016)]
    tv  = d[(d.Tahun>=1990)&(d.Tahun<=2016)]
    te  = d[(d.Tahun>=2017)&(d.Tahun<=2023)]
    return tr, va, tv, te

def rmse(y,p):
    from sklearn.metrics import mean_squared_error
    return np.sqrt(mean_squared_error(y,p))

from sklearn.linear_model import LinearRegression, Ridge
from sklearn.svm import SVR
from sklearn.ensemble import RandomForestRegressor
from sklearn.feature_selection import RFECV
from sklearn.model_selection import TimeSeriesSplit
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import make_pipeline
from xgboost import XGBRegressor
from sklearn.metrics import r2_score
df = load_data(); tr,va,tv,te = splits(df)
Xtv, ytv = tv[FEATURES].values, tv[TARGET].values
Xte, yte = te[FEATURES].values, te[TARGET].values
```

```
def run(estimator_pipe, feats=FEATURES):
    Xtr=tv[feats].values; Xt=te[feats].values
    estimator_pipe.fit(Xtr,ytv); p=estimator_pipe.predict(Xt)
    ins=r2_score(ytv, estimator_pipe.predict(Xtr))
    return rmse(yte,p), r2_score(yte,p), ins, np.round(p,3)

models={}
models["XGBoost baseline"]=run(XGBRegressor(random_state=42))
models["XGBoost + RFE"]=run(XGBRegressor(random_state=42), ["Konsumsi_energi"])
models["MLR baseline"]=run(LinearRegression())
models["Ridge (alpha=1)"]=run(make_pipeline(StandardScaler(),Ridge(alpha=1)))
models["SVR-linear (C=1)"]=run(make_pipeline(StandardScaler(),SVR(kernel="linear",C=1)))
models["SVR-RBF (C=1)"]=run(make_pipeline(StandardScaler(),SVR(kernel="rbf",C=1)))

T1=pd.DataFrame([[k,v[0],v[1],v[2]] for k,v in models.items()],
    columns=["Model","Test RMSE","Test R2","in-sample R2"]).sort_values("Test RMSE")
print("### T1 — Rekap metrik seluruh model (urut RMSE)")
print(T1.to_string(index=False))

T2=pd.DataFrame({"Tahun":te.Tahun.values,"Aktual":np.round(yte,3)})
for k,v in models.items(): T2[k]=v[3]
print("\n### T2 — Prediksi per-tahun semua model vs aktual")
print(T2.to_string(index=False))
```

Rekap metrik seluruh model (urut RMSE)

Model	Test RMSE	Test R2	in-sample R2
SVR-linear (C=1)	0.0467	0.9453	0.9657
Ridge (alpha=1)	0.0643	0.8963	0.9654
MLR baseline	0.3200	-1.5712	0.9819
XGBoost baseline	0.3567	-2.1939	1.0000
XGBoost + RFE	0.3591	-2.2377	1.0000
SVR-RBF (C=1)	0.7339	-12.5237	0.9577

Prediksi per-tahun semua model vs aktual

Tahun	Aktual	XGBoost baseline	XGBoost + RFE	MLR baseline	Ridge (alpha=1)	SVR-linear (C=1)	SVR-RBF (C=1)
2017	2.0830	2.0450	2.0420	2.1960	2.1350	2.1320	1.8820
2018	2.2010	2.0450	2.0420	2.3560	2.2120	2.2150	1.7920
2019	2.3990	2.0450	2.0420	2.5280	2.2950	2.3220	1.7220
2020	2.2130	2.0450	2.0420	2.2760	2.2300	2.1980	1.7220
2021	2.2390	2.0450	2.0420	2.3960	2.2890	2.2650	1.6680
2022	2.6430	2.0450	2.0420	3.1500	2.5330	2.6230	1.5070
2023	2.6080	2.0450	2.0420	3.2220	2.5880	2.6810	1.4910

Blok 12 — Hyperparameter tuning (Tanpa penyetelan, *hold-out,5-fold CV, TimeSeriesSplit)*

→ *lihat Tabel 4.12 naskah*

Blok ini merupakan penambahan pada tahap revisi pasca-ujian kelayakan, menjawab arahan Penguji 2 untuk melakukan hyperparameter tuning. Penyetelan mengikuti prosedur grid search Wang et al. (2025) dan diuji pada tiga skema validasi (hold-out, 5-fold CV, TimeSeriesSplit) untuk memastikan hasil tidak bergantung pada pilihan skema.

```
import subprocess, sys
def _pip(p):
    try: __import__(p if p!="scikit-learn" else "sklearn")
    except ImportError: subprocess.run([sys.executable,"-m","pip","install","-
q",p])
```

```
for _p in ["gdown","pandas","numpy","scikit-learn","xgboost"]: _pip(_p)

import gdown, pandas as pd, numpy as np, os, warnings
warnings.filterwarnings("ignore")
from itertools import product
from xgboost import XGBRegressor
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import KFold, TimeSeriesSplit
np.set_printoptions(suppress=True)

FILE_ID = "1rtkDS-PVKVYKkZSeDEA0xxTnH_l6OOZm"; DATA = "Dataset_FINAL_fixed.xlsx"
if not os.path.exists(DATA): gdown.download(id=FILE_ID, output=DATA, quiet=True)

d = pd.read_excel(DATA, skiprows=2, nrows=34)
d.columns =
["Tahun","CO2_per_kapita","Luas_hutan","PDB_per_kapita","Populasi","Konsumsi_ene
rgi","EBT_persen"]
d["Tahun"] = d["Tahun"].astype(int)
FEATURES =
["PDB_per_kapita","Konsumsi_energi","EBT_persen","Luas_hutan","Populasi"]
TARGET   = "CO2_per_kapita"

tr = d[(d.Tahun>=1990)&(d.Tahun<=2011)]
va = d[(d.Tahun>=2012)&(d.Tahun<=2016)]
tv = d[(d.Tahun>=1990)&(d.Tahun<=2016)]
te = d[(d.Tahun>=2017)&(d.Tahun<=2023)]
def rmse(y,p): return np.sqrt(mean_squared_error(y,p))

GRID = list(product([0.01,0.1,0.2],[3,5,7],[50,100,200]))  # grid Wang: eta, d,
n

def best_holdout(feats):
    Xtr,ytr = tr[feats].values, tr[TARGET].values
    Xv,yv   = va[feats].values, va[TARGET].values
    best=None
    for eta,dep,n in GRID:

m=XGBRegressor(learning_rate=eta,max_depth=dep,n_estimators=n,random_state=42).f
it(Xtr,ytr)
        vr=rmse(yv,m.predict(Xv))
        if best is None or vr<best[0]: best=(vr,eta,dep,n)
    return best[1:]

def best_cv(feats, splitter):
    Xtv,ytv = tv[feats].values, tv[TARGET].values
    best=None
    for eta,dep,n in GRID:
        sc=[]
        for i,j in splitter.split(Xtv):

m=XGBRegressor(learning_rate=eta,max_depth=dep,n_estimators=n,random_state=42).f
it(Xtv[i],ytv[i])
            sc.append(rmse(ytv[j],m.predict(Xtv[j])))
        cv=np.mean(sc)
        if best is None or cv<best[0]: best=(cv,eta,dep,n)
    return best[1:]

def eval_cfg(feats,cfg):
    Xtv,ytv = tv[feats].values, tv[TARGET].values
    Xte,yte = te[feats].values, te[TARGET].values
    if cfg is None:
        m=XGBRegressor(random_state=42).fit(Xtv,ytv)
    else:
        eta,dep,n=cfg

m=XGBRegressor(learning_rate=eta,max_depth=dep,n_estimators=n,random_state=42).f
it(Xtv,ytv)
    p=m.predict(Xte)
    return rmse(yte,p), r2_score(yte,p)

rows=[]
for feats,label in [(FEATURES,"XGBoost baseline"),(["Konsumsi_energi"],"XGBoost
+ RFE")]:
```

```
    r,r2=eval_cfg(feats,None);
rows.append([label,"Tanpa penyetelan","–",r,r2])
    c=best_holdout(feats);              r,r2=eval_cfg(feats,c);
rows.append([label,"Hold-out",       f"{c[0]}; {c[1]}; {c[2]}",r,r2])
    c=best_cv(feats,KFold(5,shuffle=True,random_state=42));
r,r2=eval_cfg(feats,c); rows.append([label,"5-fold CV",f"{c[0]}; {c[1]};
{c[2]}",r,r2])
    c=best_cv(feats,TimeSeriesSplit(5)); r,r2=eval_cfg(feats,c);
rows.append([label,"TimeSeriesSplit",f"{c[0]}; {c[1]}; {c[2]}",r,r2])

T = pd.DataFrame(rows, columns=["Model","Skema Validasi","Konfigurasi (eta; d;
n)","RMSE Uji","R2 Uji"])
pd.set_option("display.float_format", lambda x: f"{x:.4f}")
print("### Tabel 4.12 – Penyetelan XGBoost (grid Wang) pada berbagai skema
validasi")
print(T.to_string(index=False))
print(f"\nRentang R2 uji lintas skema: {T['R2 Uji'].min():.4f} s.d. {T['R2
Uji'].max():.4f}")
```

Kinerja XGBoost Sebelum dan Sesudah Penyetelan Hyperparameter pada Berbagai Skema Validasi

Model	Skema Validasi	Konfigurasi (η; d; n)	RMSE uji	R^2 uji
XGBoost *baseline*	Tanpa Penyetelan	_	0,357	−2,194
XGBoost *baseline*	*Hold-out*	**η** *=0,1; depth=3; n=100*	0,352	−2,103
XGBoost *baseline*	5-fold CV	**η** *=0,1; depth=3; n=100*	0,352	−2,103
XGBoost *baseline*	*TimeSeriesSplit*	**η** =0,2; depth=7; n=50	0,358	−2,220
XGBoost + RFE	Tanpa Penyetelan	_	0,359	−2,238
XGBoost + RFE	*Hold-out*	**η** =0,2; depth=3; n=200	0,359	−2,231
XGBoost + RFE	5-fold CV	**η** =0,2; depth=3; n=50	0,369	−2,414
XGBoost + RFE	*TimeSeriesSplit*	**η** =0,2; depth=3; n=50	0,359	−2,237