DAFTAR PUSTAKA

Andriani, N., Wahyuningsih, S., & Siringoringo, M. (2022). Application of Double Exponential Smoothing Holt and Triple Exponential Smoothing Holt-Winter with Golden Section Optimization to Forecast Export Value of East Borneo Province. *Jurnal Matematika, Statistika dan Komputasi*, *18*(3), 475-483.

Andriani, Y., Silitonga, H., & Wanto, A. (2018). Analisis Jaringan Syaraf Tiruan untuk prediksi volume ekspor dan impor migas di Indonesia. *Register*, *4*(1), 30-40

Ardiansah, I., Adiarsa, I. F., Putri, S. H., & Pujianto, T. (2021). Penerapan Analisis Runtun Waktu pada Peramalan Penjualan Produk Organik menggunakan Metode Moving Average dan Exponential Smoothing. *Jurnal Teknik Pertanian Lampung*, *10*(4), 548-559.

Asana, I. M. D. P., Kurniadi, I. M. D., Dwipayani, S. A., & Atmaja, K. J. (2022). Sales Forecasting Applications For Retail Companies Using Double Exponential Smoothing And Golden Section Methods. *Jurnal Mantik*, *6*(2), 1603-1611.

Brata, A. S., Anhar, A., Lestari, W., Juliza, M., Rahmawati, S., & Nugroho, M. T. A. E. (2021). Average Based Length Fuzzy Time Series Data Seasonal untuk Prediksi Volume Impor Migas Indonesia. *Jurnal Ekonomi Manajemen dan Sekretari*, *6*(1), 15-21.

Febtrianti, I. D., Hani'ah, M., & Rosiani, U. D. (2021, November). Optimasi Double Exponential Smoothing menggunakan Metode Golden Section untuk Peramalan Penjualan Sparepart. In *Seminar Informatika Aplikatif Polinema* (pp. 42-45).

Gunawan, R., & Suripto, S. (2023). Determinan Impor Migas di Indonesia: Pendekatan VAR. *Journal of Macroeconomics and Social Development*, *1*(1), 1-13.

Hamidi, D. Z. (2023). BAB XI ANALISIS DERET WAKTU DAN RAMALAN. *STATISTIKA BISNIS*, 161.

Kinasih, S., Agoestanto, A., & Sugiman, S. (2018). Optimasi Parameter pada Model Exponential Smoothing Menggunakan Metode Golden Section untuk Pemilihan Model Terbaik dan Peramalan Jumlah Wisatawan Provinsi Jawa Tengah. *Unnes Journal of Mathematics*, *7*(1), 38-46.

Lawrence, K. D., Klimberg, R. K., & Lawrence, S. M. (2009). *Fundamentals of forecasting using excel*. Industrial Press Inc..

Makridakis, S., Wheelwright, S. C., & McGee, V. E. (1992). Metode dan Aplikasi Peramalan. Terjemahan Untung S, Andriyanto. Jakarta: PT Gelora Aksara Pratama; Erlangga.

. Forecasting oil tanker shipping market in crisis periods: smoothing model application. *The Asian Journal of Shipping* *s*, *37*(3), 239-244.

roni, O., Rahmawati, S. D., Febrianti, T., & Mahuda, I. (2021). eramalan Jumlah Penerimaan Mahasiswa Baru Dengan an Metode Regresi Linear Sederhana. *Jurnal Bayesian: Jurnal* *stika dan Ekonometrika*, *1*(1), 17-23.

Peixeiro, M. (2022). *Time series forecasting in python*. Simon and Schuster.

Permatasari, R., Mariani, S., & Sugiman, S. (2022). Pemodelan dan Peramalan Runtun Waktu Nonlinier dengan Metode Exponential Smooth Transition Autoregressive (ESTAR). *Indonesian Journal of Mathematics and Natural Sciences*, *45*(1), 20-29.

Primandari, A. H. (2016). Grey Double Exponential Smoothing dengan Optimasi Levenberg-Marquardt untuk Peramalan Volume Penumpang di Bandara Soekarno-Hatta. *Jurnal Derivat: Jurnal Matematika dan Pendidikan Matematika*, *3*(2), 25-39.

Purwanti, D., & Purwadi, J. (2019). Metode Brown's Double Exponential Smoothing dalam Peramalan Laju Inflasi di Indonesia. *Jurnal Ilmiah Matematika*, *6*(2), 54.

Putri, S., & Ibrahim, H. (2023). Peranan Perdagangan Internasional Terhadap Perekonomian Indonesia. *Jurnal Minfo Polgan*, *12*(2), 2424-2428.

Rusyida, W. Y. (2022). *Teknik Peramalan: Metode ARIMA dan Holt Winter*. Penerbit NEM.

Saputra, A. (2020). *Komputasi untuk teknik kimia menggunakan matlab*. Andri Saputra.

Saputra, N. D., Aziz, A., & Harjito, B. (2021). Perhitungan Kompleksitas Metode Golden Section dalam Optimasi Parameter Pemulusan Eksponensial Ganda Brown dan Holt. *Jurnal Algoritma*, *18*(2), 330-341.

Septiyanor, H., Syaripuddin, S., & Goejantoro, R. (2021). Perancangan Aplikasi Peramalan untuk Metode Exponential Smoothing Menggunakan Aplikasi Lazarus (Studi Kasus: Data Konsumsi Listrik Kota Samarinda). *ESTIMASI: Journal of Statistics and Its Application*, 57-70.

Syifahati, T., Triska, A., & Nahar, J. (2023). Forecasting the Indonesian Coffee Production and Consumption Using The Exponential Smoothing Methods with Modified Golden Section Search to Estimate the Smoothing Parameters. *Jurnal Matematika Integratif*, 41-54.

Triono, A., Budi, A. S., & Abdillah, R. (2023). Implementasi peretasan sandi vigenere chipher menggunakan bahasa pemrograman python. *JOCITIS-Journal Science Infomatica and Robotics*, *1*(1), 01-09.

Widitriani, NPS, Parwita, WGS, & Meinarni, NPS (2020). Sistem peramalan menggunakan single exponential smoothing dengan optimasi golden section. Dalam Jurnal Fisika: Seri Konferensi (Vol. 1516, No. 1, hal. 012008). Penerbitan IOP.

Wijaya, E., Nopiandri, K., & Habiburrokhman, H. (2017). Dinamika upaya melakukan ntara hukum perdagangan internasional dan hukum *Jurnal Hukum dan Peradilan*, *6*(3), 487-508.

& Rifai, N. A. K. (2022, August). Peramalan Produksi Mobil an Metode Double Exponential Smoothing dengan Algoritma tion. In *Bandung Conference Series: Statistics* (Vol. 2, No. 2, pp.

Yunita, T. (2020). Peramalan Jumlah Penggunaan Kuota Internet Menggunakan Metode Autoregressive Integrated Moving Average (ARIMA). *Journal of Mathematics: Theory and Applications*, 16-22.

Yuwida, N., & Wahyuningsih, N. (2012). Estimasi Parameter dan Dalam Pemulusan Eksponensial Ganda Dua Parameter Dengan Metode Modifikasi Golden Section. *Jurnal Sains dan Seni ITS*, *1*(1), A18-A22.

Zahrunnisa, A., Nafalana, R. D., Rosyada, I. A., & Widodo, E. (2021). Perbandingan Metode Exponential Smoothing Dan Arima Pada Peramalan Garis Kemiskinan Provinsi Jawa Tengah. *Jurnal Lebesgue: Jurnal Ilmiah Pendidikan Matematika, Matematika dan Statistika*, *2*(3), 300-314.

LAMPIRAN

Lampiran 1. Data Volume Impor Migas Indonesia Periode Januari 2016-Desember 2022

Bulan	Data Volume Impor Minyak dan Gas (Migas) Indonesia (Ton)											
	2012	2013	2014	2015	2016	2017	2018	2019	2020	2021	2022	2023
Januari	3168.24	4219.16	3731.63	3606.31	3448.34	3717.95	3744.2	3383.26	3441.18	3515.4	3189.52	4340.4
Februari	3689.6	3742.7	3647.29	3517.87	3530.22	4912.03	3945.73	3110.15	3371.11	2512.06	3769.32	3403.1
Maret	3813.22	4089.34	4235.5	4296.99	4671.26	4614.01	4051.77	2696.88	3936.94	4108.83	3852.01	4180.1
April	3986.74	3941.8	3979.17	4411.94	3721.96	3376.84	4086.44	3836.75	3605.61	3752.94	4144.59	4244.2
Mei	3435.94	3890.68	4024.61	3575.25	4348.71	3933.55	4665.63	3673.87	2640.76	3627.45	3403.19	4713.1
Juni	3469.52	4041.1	3709.86	4500.06	4102.74	3662.75	3331.61	3210.05	2231.46	4008.89	3697.5	3498.4
Juli	3152.03	4629.74	4413.32	4079.6	3645.81	4170.23	4162.28	3268.02	3002.68	2777.65	4621.2	4996.5
Agustus	3722.94	3996.81	3731.56	4206.97	4624.52	4535.41	4844.54	3181.04	2781.86	3268.1	4283.34	3891
September	3553.76	3981.06	4106.69	4199.58	4546.33	4039.33	3566.66	3180.85	3208.94	2945.25	4172.1	4553
Oktober	3980.39	3781.33	4173.73	3860.72	3706.92	4494.29	4257.94	3349.64	3000.35	2577.75	4474.3	4308.7
		4221.77	4369.8	3691.52	4032.51	4185.6	4663.58	4098.37	2879.12	3968.27	3778.3	5004.2
		4518.56	4746.47	4362.44	3946.48	4728.09	3895.73	3937.58	3554.13	5063.74	4355.6	5011.4

Pusat Statistik Indonesia

Lampiran 2. *Source Code* Program Python untuk Plot Data

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

def lihat_tren(data):
    # Data x dari 0 sampai dengan panjang data
    x = list(range(len(data)))

    # Hitung garis regresi linier
    slope, intercept = np.polyfit(x, data, 1)
    trendline = intercept + (slope * np.array(x))

    # Plot data dan garis tren
    plt.figure(figsize=(7, 3))
    plt.plot(x, data, label='Data')
    plt.plot(x, trendline, color='grey', linestyle='--',
label='Garis Tren')
    plt.xlabel('Periode')
    plt.ylabel('Nilai Volume Impor Migas')
    plt.title('Tren Pola Data')
    plt.grid(True)
    plt.legend()

if __name__ == "__main__":
    # Membaca data dari file Excel
    nama_file = 'Data Impor Migas.xlsx'
    df = pd.read_excel(nama_file)

    # Mengambil kolom yang ingin dilihat tren polanya (misalnya
kolom 'nilai')
    kolom_data = 'Volume Impor Migas'
    data = df[kolom_data].tolist()
    # Memanggil fungsi untuk melihat tren pola data
    lihat_tren(data)
```

Lampiran 3 *Source Code* Program Uji Stasioner Data

```
import pandas as pd
from statsmodels.tsa.stattools import adfuller

# Langkah 1: Memuat data dari file Excel
data_path = 'Data Impor Migas.xlsx'
try:
    df = pd.read_excel(data_path)
except FileNotFoundError:
    print("File Excel tidak ditemukan. Pastikan path file sudah
benar.")

# Langkah 2: Memilih kolom atau deret waktu yang ingin diuji
ries = df['Volume Impor Migas']  # Ganti 'nama_kolom'
nama kolom yang sesuai dengan data deret waktu Anda

ah 3: Melakukan uji ADF
= adfuller(time_series, autolag='AIC')
```

```
# Langkah 4: Mencetak hasil uji ADF
print('Hasil Uji Augmented Dickey-Fuller:')
print('---------------------------------')
print('Nilai ADF:', result[0])
print('Nilai p-value:', result[1])
print(f'Jumlah lags: {result[2]}')
print('Nilai-nilai kritis:')
for key, value in result[4].items():
    print('\t', key, ':', value)
print('Tingkat signifikansi: 0.05')
print('---------------------------------')
print('Kesimpulan:')
if result[1] <= 0.05:
    print('P-value kurang dari atau sama dengan 0.05. Data
stasioner.')
else:
    print('P-value lebih besar dari 0.05. Data tidak stasioner.')
```

Lampiran 4. *Source Code* Program Python untuk Memuat Data Dari File Excel

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from tabulate import tabulate

# Load data dari file Excel
data_excel = pd.read_excel('Data Impor Migas.xlsx')
data_series = data_excel['Volume Impor Migas'].tolist()
```

Lampiran 5. *Source Code* Program Python untuk Pembagian Data

```
# Bagi data menjadi data training dan data testing (85% training,
15% testing)
train_size = int(len(data_series) * 0.85)
data_train, data_test = data_series[:train_size],
data_series[train_size:]

# Plot data
plt.figure(figsize=(8, 10))
plt.subplot(3, 1, 1)
plt.plot(data_train, label='Data Training', color='blue')
plt.plot(range(train_size, len(data_series)), data_test,
label='Data Testing', color='red')
plt.xlabel('Bulan')
plt.ylabel('Volume Impor Migas')
plt.title('Data Training and Testing')
plt.legend()
#plt.grid(True)
plt.show()
```

Lampiran 6. *Source Code* Program Python untuk Perhitungan Parameter DES Brown Menggunakann Metode *Golden Section*

```
# Fungsi untuk menghitung MAPE
def calculate_MAPE(y_true, y_pred):
    if len(y_true) != len(y_pred):
        raise ValueError("Ukuran y_true dan y_pred harus sama.")

    y_true = np.array(y_true)
    y_pred = np.array(y_pred)

    # Pastikan tidak ada pembagian dengan nol
    if np.any(y_true == 0):
        raise ZeroDivisionError("Tidak boleh ada nilai nol dalam
y_true.")

    return np.mean(np.abs((y_true - y_pred) / y_true)) * 100

# Fungsi Double Exponential Smoothing
def double_exponential_smoothing_brown(alpha, series,
forecast_periods=1):
    n = len(series)
    Lt = [series[0]]
    Ltt = [series[0]]

    forecast = []

    # Hitung nilai Lt dan Ltt
    for i in range(1, n):
        Lt.append(alpha * series[i] + (1 - alpha) * Lt[i - 1])
        Ltt.append(alpha * Lt[i] + (1 - alpha) * Ltt[i - 1])

    # Hasilkan forecast
    for i in range(n):
        at = 2 * Lt[i] - Ltt[i]
        bt = (alpha / (1 - alpha)) * (Lt[i] - Ltt[i])
        forecast.append(at + bt)

    return forecast

# Optimasi alpha terbaik dengan Golden Section
def golden_section_optimization_brown(series, tolerance=1e-5,
max_iterations=100):
    phi = 0.61803398875
    a = 0
    d = 1
    iteration = 0
    optimal_alpha = (a + d) / 2
    table_data = []

    while iteration < max_iterations:
        iteration += 1

        # Hitung dua titik tengah
        b = phi * a + (1 - phi) * d
        c = a + d - b

        # Hitung nilai
```

```
        f_b = calculate_MAPE(series,
double_exponential_smoothing_brown(b, series))  # Hitung MAPE
        f_c = calculate_MAPE(series,
double_exponential_smoothing_brown(c, series))  # Hitung MAPE
        e = abs(d - a)
        table_data.append([iteration, a, b, c, d, f_b, f_c, e])

        # Update batas sesuai aturan golden section
        if f_b < f_c:
            d = c  # Atur batas atas
            c = b
            a = a
            b = phi * a + (1 - phi) * d

        else:
            a = b  # Atur batas bawah
            b = c
            d = d
            c = a + d - b

        if e < tolerance:
            optimal_alpha = (a + d) / 2
            break
        if iteration >= max_iterations:
            print("Iterasi mencapai batas maksimum")
            break

    headers = ["Iterasi", "a", "b", "c", "d", "f(b)", "f(c)", "|d-
a|"]
    print(tabulate(table_data, headers=headers, tablefmt="grid"))

    return optimal_alpha, iteration
```

```
# Lakukan optimisasi menggunakan metode Golden Section dengan satu
parameter (alpha) menggunakan data training
alpha_optimal, iteration =
golden_section_optimization_brown(data_train)

# Tampilkan parameter optimal
print("Nilai alpha optimal:", alpha_optimal)
```

```
# Buat peramalan menggunakan alpha optimal untuk data training
forecast_train = double_exponential_smoothing_brown(alpha_optimal,
data_train)
mape_train = calculate_MAPE(np.array(data_train),
np.array(forecast_train))
print(f'Nilai MAPE data train: {mape_train:.4f}%')

# Plot grafik data aktual, data training, data testing, dan
peramalan
plt.figure(figsize=(8, 10))
# Plot grafik data aktual, data training, dan peramalan training
plot(3, 1, 2)
t(data_series, label='Data Aktual')
(data_series[:len(data_train)], label='Data Aktual')
(range(len(data_train)), forecast_train, label='Peramalan
', linestyle='-', color='green')
le('Peramalan Data Training untuk DES Brown')
```

```
plt.xlabel('Bulan')
plt.ylabel('Volume Impor Migas')
plt.legend()

# Buat DataFrame untuk menampilkan hasil peramalan data testing
beserta periodenya
# Buat peramalan menggunakan alpha optimal untuk data testing
forecast_test = double_exponential_smoothing_brown(alpha_optimal,
data_test)
forecast_df = pd.DataFrame({'Periode': range(len(data_train),
len(data_series)), 'Data Testing': data_test, 'Peramalan Testing':
forecast_test})
mape_test = calculate_MAPE(np.array(data_test),
np.array(forecast_test))
print(f'Nilai MAPE data test: {mape_test:.4f}%')

# Tampilkan DataFrame
print("Hasil Peramalan Data Testing:")
print(forecast_df)

# Plot grafik data aktual, data training, data testing, dan
peramalan
plt.figure(figsize=(8, 10))
# Plot grafik data testing dan peramalan testing
plt.subplot(3, 1, 3)
plt.plot(forecast_df['Periode'], forecast_df['Data Testing'],
label='Data Testing', linestyle='-', color='blue')
plt.plot(forecast_df['Periode'], forecast_df['Peramalan Testing'],
label='Peramalan Testing', linestyle='-', color='green')
plt.title('Peramalan Data Testing')
plt.xlabel('Bulan')
plt.ylabel('Volume Impor Migas')
plt.legend()
plt.show()
```

Lampiran 7. *Source Code* Program Python Peramalan dengan DES Brown

```
# Tampilkan peramalan untuk beberapa periode ke depan
forecast_periods = 5
forecast_future =
double_exponential_smoothing_brown(alpha_optimal, data_series,
forecast_periods)
print("Peramalan untuk", forecast_periods, "periode ke depan:",
forecast_future[-forecast_periods:])
#mape
mape_future = calculate_MAPE(np.array(data_series),
np.array(forecast_future))
print(f'Nilai MAPE peramalan: {mape_future:.4f}%')

# Plot grafik data aktual, data training, data testing, dan
peramalan
plt.figure(figsize=(8, 10))
grafik peramalan masa depan
plot(3, 1, 3)
ndex_future = len(data_series) + 1
ex_future = start_index_future + forecast_periods
```

```
plt.plot(range(start_index_future, end_index_future),
forecast_future[-forecast_periods:], label='Peramalan Masa Depan',
linestyle='-', color='green')
plt.xticks(range(start_index_future, end_index_future, 1))
plt.title('Peramalan Masa Depan')
plt.xlabel('Bulan')
plt.ylabel('Volume Impor Migas')
plt.legend()

plt.tight_layout()
```

Lampiran 8. *Source Code* Program Python untuk Perhitungan Parameter DES Holt menggunakan metode modifikasi *Golden Section.*

```
# Fungsi untuk melakukan peramalan menggunakan model Double
Exponential Smoothing
def double_exponential_smoothing(alpha, beta, series,
forecast_periods=1):
    n = len(series)
    level = [series[0]]
    trend = [series[1]-series[0]]
    forecast = [level[0] + trend[0]]

    for i in range(1, n):
        level.append(alpha * series[i] + (1 - alpha) * (level[i-1]
+ trend[i-1]))
        trend.append(beta * (level[i] - level[i-1]) + (1 - beta) *
trend[i-1])
        forecast.append(level[i] + trend[i])

    # Prediksi untuk beberapa periode ke depan
    for _ in range(forecast_periods):
        level.append(level[-1] + trend[-1])
        trend.append(beta * (level[-1] - level[-2]) + (1 - beta) *
trend[-1])
        forecast.append(level[-1] + trend[-1])

    return forecast

# Fungsi objektif untuk dioptimalkan
def calculate_MAPE(y_true, y_pred):
    # Ensure y_true and y_pred have the same length
    min_length = min(len(y_true), len(y_pred))
    y_true = np.array(y_true[:min_length])
    y_pred = np.array(y_pred[:min_length])

    # Calculate MAPE
    mape = np.mean(np.abs((y_true - y_pred) / y_true)) * 100
    return mape

#fungsi golden section
def golden_section_optimization_holt(series, tolerance=0.0001,
    rations=100):
    nisialisasi parameter
     = 0.618033989  # Golden ratio
     d1 = 0, 1
     d2 = 0, 1
    ration = 0
```

```
    mape_values = []
    mape_current = []  # Menyimpan nilai MAPE setiap iterasi
    table_data = [] # Menyimpan data untuk tabel

    while True:
        iteration += 1
        # Hitung dua titik tengah
        b1 = phi * a1 + (1 - phi) * d1
        c1 = a1 + d1 - b1
        b2 = phi * a2 + (1 - phi) * d2
        c2 = a2 + d2 - b2

        # Hitung nilai objektif pada dua titik tengah
        f_b1b2 = calculate_MAPE(series,
double_exponential_smoothing(b1, b2, series))
        f_b1c2 = calculate_MAPE(series,
double_exponential_smoothing(b1, c2, series))
        f_c1b2 = calculate_MAPE(series,
double_exponential_smoothing(c1, b2, series))
        f_c1c2 = calculate_MAPE(series,
double_exponential_smoothing(c1, c2, series))
        e1 = abs(d1 - a1)
        e2 = abs(d2 - a2)
        table_data.append([iteration, a1, b1, c1, d1, a2, b2, c2,
d2, f_b1b2, f_b1c2, f_c1b2, f_c1c2])

        # Update batas sesuai aturan Golden Section
        if f_b1b2 > f_b1c2 and f_b1b2 > f_c1b2 and f_b1b2 >
f_c1c2:
            a1 = b1
            b1 = c1
            c1 = a1 + d1 - b1
            a2 = b2
            b2 = c2
            c2 = a2 + d2 - b2

        elif f_b1c2 > f_b1b2 and f_b1c2 > f_c1b2 and f_b1c2 >
f_c1c2:
            a1 = b1
            b1 = c1
            c1 = a1 + d1 - b1
            d2 = c2
            c2 = b2
            b2 = phi * a2 + (1 - phi) * d2

        elif f_c1b2 > f_b1b2 and f_c1b2 > f_b1c2 and f_c1b2 >
f_c1c2:
            d1 = c1
            c1 = b1
            b1 = phi * a1 + (1 - phi) * d1
            a2 = b2
            c2 = a2 + d2 - b2
        else:
            d1 = c1
            c1 = b1
            b1 = phi * a1 + (1 - phi) * d1
            d2 = c2
            c2 = b2
            b2 = phi * a2 + (1 - phi) * d2
```

```
        e2 = abs(d2 - a1)

        #hitung mape
        mape_current = calculate_MAPE(series,
double_exponential_smoothing((a1 + d1) / 2, (a2 + d2) / 2,
series))
        mape_values.append(mape_current)

        # Cek kriteria berhenti
        if abs(d1 - a1) < tolerance:
            print("Konvergensi tercapai.")
            break
            #return (a1 + a2) / 2, (d1 + d2) / 2, iteration  #
Mengembalikan nilai optimal dan jumlah iterasi
        if iteration >= max_iterations:
            print("Iterasi mencapai batas maksimum")
            break
            #return (a1 + a2) / 2, (d1 + d2) / 2, iteration  #
Mengembalikan nilai optimal dan jumlah iterasi

    headers = ["Iteration", "a1", "b1", "c1", "d1", "a2", "b2",
"c2", "d2", "f_b1b2", "f_b1c2", "f_c1b2", "f_c1c2"]
    print(tabulate(table_data, headers=headers, tablefmt="grid"))

    return ((a1 + d1) / 2, (a2 + d2) / 2), iteration, mape_values

# Lakukan optimisasi menggunakan data training
(alpha_optimal, beta_optimal), iterations, mape_values =
golden_section_optimization_holt(data_train)

# Tampilkan parameter optimal untuk data training
print("Nilai alpha optimal:", alpha_optimal)
print("Nilai beta optimal:", beta_optimal)

# Buat peramalan menggunakan alpha dan beta optimal untuk data
training
forecast_train = double_exponential_smoothing(alpha_optimal,
beta_optimal, data_train)
mape_train = calculate_MAPE(np.array(data_train),
np.array(forecast_train))
print(f'Nilai MAPE data train: {mape_train:.4f}%')

# Plot grafik data aktual, data training, data testing, dan
peramalan
plt.figure(figsize=(8, 10))
# Plot grafik data aktual, data training, dan peramalan training
plt.subplot(3, 1, 2)
#plt.plot(data_series, label='Data Aktual')
plt.plot(data_series[:len(data_train)], label='Data Aktual')
plt.plot(range(len(data_train)), forecast_train[:-1],
label='Peramalan Training', linestyle='-', color='green')  #
Menggunakan[:-1] untuk memotong satu nilai dari peramalan
    le('Peramalan Data Training untuk DES Holt')
    bel('Bulan')
    bel('Volume Impor Migas')
    end()
```

```
forecast_test = double_exponential_smoothing(alpha_optimal,
beta_optimal, data_test)

# Pastikan panjang data test dan peramalan test sama
min_length = min(len(data_test), len(forecast_test))
data_test = data_test[:min_length]
forecast_test = forecast_test[:min_length]

forecast_df = pd.DataFrame({'Periode': range(len(data_train),
len(data_train) + min_length), 'Data Testing': data_test,
'Peramalan Testing': forecast_test})
mape_test = calculate_MAPE(np.array(data_test),
np.array(forecast_test))
print(f'Nilai MAPE data test: {mape_test:.4f}%')

# Tampilkan DataFrame
print("Hasil Peramalan Data Testing:")
print(forecast_df)

# Plot grafik data aktual, data training, data testing, dan
peramalan
plt.figure(figsize=(8, 10))
# Plot grafik data testing dan peramalan testing
plt.subplot(3, 1, 3)
plt.plot(forecast_df['Periode'], forecast_df['Data Testing'],
label='Data Testing', linestyle='-', color='blue')
plt.plot(forecast_df['Periode'], forecast_df['Peramalan Testing'],
label='Peramalan Testing', linestyle='-', color='green')
plt.title('Peramalan Data Testing')
plt.xlabel('Bulan')
plt.ylabel('Volume Impor Migas')
plt.legend()
plt.show()
```