DAFTAR PUSTAKA

Arifin, B., Lanang, G., Tanaya, P., & Usman, A. (2021). Prosiding SAINTEK PERAMALAN HARGA KELAPA SAWIT DUNIA PADA TAHUN 2020-2024. *LPPM Universitas Mataram*, *3*.

Arum Pitaloka, R., & Rahmawati, R. (2019). *PERBANDINGAN METODE ARIMA BOX-JENKINS DENGAN ARIMA ENSEMBLE PADA PERAMALAN NILAI IMPOR PROVINSI JAWA TENGAH*. *8*(2), 194–207. http://ejournal3.undip.ac.id/index.php/gaussian

Asih Maruddani, D. I., Al Anisah, R., Studi Statistika, P., Matematika, J., & Prodi Statistika, A. (n.d.). *UJI STASIONERITAS DATA INFLASI DENGAN PHILLIPS-PERON TEST*.

Azmi, U., & Syaifudin, W. H. (2020). Peramalan Harga Komoditas Dengan Menggunakan Metode Arima-Garch. *Jurnal Varian*, *3*(2), 113–124. https://doi.org/10.30812/varian.v3i2.653

Akaike, H. (1974). A New Look at the Statistical Model Identification. *IEEE Transactions on Automatic Control, 19*(6), 716–723.

Azmi, U., & Syaifudin, W. H. (2020). Peramalan Harga Komoditas Dengan Menggunakan Metode Arima-Garch. *Jurnal Varian, 3*(2), 113–124.

Akaike, H. (1974). A New Look at the Statistical Model Identification. *IEEE Transactions on Automatic Control, 19*(6), 716–723.

Arifin, B., Lanang, G., Tanaya, P., & Usman, A. (2021). Prosiding SAINTEK Peramalan Harga Kelapa Sawit Dunia pada Tahun 2020–2024. *LPPM Universitas Mataram, 3*.

Arifin, B., Lanang, G., Tanaya, P., & Usman, A. (2021). *Prosiding SAINTEK Peramalan Harga Kelapa Sawit Dunia pada Tahun 2020–2024*. LPPM Universitas Mataram.

Azmi, U., & Syaifudin, W. H. (2020). Peramalan Harga Komoditas Dengan Menggunakan Metode Arima-Garch. *Jurnal Varian, 3*(2), 113–124.

Box, G. E. P., & Jenkins, G. M. (1970). *Time Series Analysis: Forecasting and Control*. San Francisco: Holden-Day.

Burnham, K. P., & Anderson, D. R. (2002). *Model Selection and Multimodel Inference: A Practical Information-Theoretic Approach*. Springer

Chatfield, C. (2003). *The Analysis of Time Series: An Introduction*. Chapman & Hall/CRC.

Dickey, D. A., & Fuller, W. A. (1979). Distribution of the Estimators for Autoregressive Time Series with a Unit Root. *Journal of the American Statistical Association, 74*(366a), 427–431.

Enders, W. (2004). *Applied Econometric Time Series*. Wiley.

Gupta, R., & Trivedi, P. K. (2021). Integration of ARIMA and GARCH Models for Volatile Commodity Prices. *Journal of Forecasting, 40*(5), 899–915.

Hyndman, R. J., & Athanasopoulos, G. (2018). *Forecasting: Principles and Practice*. OTexts.

Makridakis, S., Wheelwright, S. C., & Hyndman, R. J. (1998). *Forecasting Methods and Applications*. Wiley.

Nurman, S., Nusrang, M., & Sudarmin. (2022). Analysis of Rice Production Forecast in Maros District Using the Box-Jenkins Method with the ARIMA Model. *ARRUS Journal of Mathematics and Applied Science, 2*(1), 36–48.

of Time Series: An Introduction. Chapman & Hall/CRC. Nizar, M. A., Kebijakan, P., Makro, E., Fiskal, B. K., & Keuangan-Ri, K. (n.d.). DAMPAK FLUKTUASI HARGA MINYAK DUNIA TERHADAP PEREKONOMIAN INDONESIA The Impact of World Oil Prices Fluctuation on Indonesia's Economy. In *Buletin Ilmiah Litbang Perdagangan* (Vol. 6, Issue 2).

Pindyck, R. S., & Rubinfeld, D. L. (1998). *Econometric Models and Economic Forecasts*. McGraw-Hill.

Tsay, R. S. (2010). *Analysis of Financial Time Series*. Wiley.

LAMPIRAN

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from statsmodels.tsa.stattools import adfuller

from statsmodels.graphics.tsaplots import plot_acf, plot_pacf

from statsmodels.tsa.arima.model import ARIMA

from sklearn.metrics import mean_squared_error


# Path ke file

file_path =r"D:\tempat skripsi\data skripsi.xlsx"

data = pd.read_excel(file_path)

data = pd.read_excel(file_path, sheet_name='Annual Prices (Nominal)', skiprows=6)


# Pilih kolom 'Year' dan 'Palm oil ($/mt)'

data = data[['Year', 'Palm oil ($/mt)']].dropna()


# Set kolom 'Year' sebagai index dan ubah ke tipe datetime

data.set_index(pd.to_datetime(data['Year'], format='%Y'), inplace=True)

data.drop(columns=['Year'], inplace=True)
```

```
plt.figure(figsize=(12, 6))
plt.plot(data.index, data['Palm oil ($/mt)'], label='Harga Minyak Kelapa Sawit')
plt.title('Harga Minyak Kelapa Sawit per Tahun')
plt.xlabel('Tahun')
plt.ylabel('Harga ($/mt)')
plt.legend()
plt.grid(True)
plt.show()

def test_stationarity(timeseries):
    rolmean = timeseries.rolling(window=5).mean()
    rolstd = timeseries.rolling(window=5).std()

    plt.figure(figsize=(12, 6))
    plt.plot(timeseries, color='blue', label='Original')
    plt.plot(rolmean, color='red', label='Rolling Mean')
    plt.plot(rolstd, color='black', label='Rolling Std')
    plt.legend(loc='best')
    plt.title('Rolling Mean & Standard Deviation')
    plt.show()
```

```
    # Perform Dickey-Fuller test:
    print('Results of Dickey-Fuller Test:')
    dftest = adfuller(timeseries, autolag='AIC')
    dfoutput = pd.Series(dftest[0:4], index=['Test Statistic', 'p-value', '#Lags Used', 'Number of Observations Used'])
    for key, value in dftest[4].items():
        dfoutput[f'Critical Value ({key})'] = value
    print(dfoutput)

# Uji stasioneritas dengan de-trending menggunakan moving average
data_ma = data['Palm oil ($/mt)'].rolling(window=12).mean()
data_ma_diff = data['Palm oil ($/mt)'] - data_ma
data_ma_diff.dropna(inplace=True)
test_stationarity(data_ma_diff)

# Plot ACF dan PACF dari data de-trend
plt.figure(figsize=(12, 6))
plt.subplot(121)
plot_acf(data_ma_diff, lags=20, ax=plt.gca())
plt.subplot(122)
plot_pacf(data_ma_diff, lags=20, ax=plt.gca())
plt.show()
```

```
# Bangun model ARIMA
model = ARIMA(data['Palm oil ($/mt)'], order=(1, 1, 0))
arima_result = model.fit()
# Ringkasan model
print(arima_result.summary())

# Prediksi untuk periode 2017 hingga 2023
start_date = '2000'
end_date = '2023'
forecast = arima_result.get_prediction(start=pd.to_datetime(start_date), end=pd.to_datetime(end_date))
forecast_values = forecast.predicted_mean
forecast_conf_int = forecast.conf_int()

# Plot hasil prediksi
plt.figure(figsize=(12, 6))
plt.plot(data.index, data['Palm oil ($/mt)'], label='Harga Aktual')
plt.plot(forecast_values.index, forecast_values, color='orange', label='Prediksi ARIMA (2017-2023)')
plt.fill_between(forecast_values.index,
                 forecast_conf_int.iloc[:, 0],
                 forecast_conf_int.iloc[:, 1],
                 color='pink', alpha=0.3)
```

```
plt.title('Prediksi Harga Minyak Kelapa Sawit dengan Model ARIMA (2017-2023)')
plt.xlabel('Tahun')
plt.ylabel('Harga ($/mt)')
plt.legend()
plt.grid(True)
plt.show()
test_data = data.loc[start_date:end_date, 'Palm oil ($/mt)']

forecast = arima_result.get_prediction(start=start_date, end=end_date)
predicted_values = forecast.predicted_mean
mse = mean_squared_error(test_data, predicted_values)
print(f"Mean Squared Error (MSE): {mse}")

# Hitung RMSE (dengan akar dari MSE)
rmse = np.sqrt(mse)

print(f"Root Mean Squared Error (RMSE): {rmse}")

# Bangun model ARIMA
model = ARIMA(data['Palm oil ($/mt)'], order=(1, 1, 1))
arima_result = model.fit()
```

```
# Ringkasan model
print(arima_result.summary())


# Prediksi untuk periode 2017 hingga 2023
start_date = '2000'
end_date = '2023'
forecast = arima_result.get_prediction(start=pd.to_datetime(start_date), end=pd.to_datetime(end_date))
forecast_values = forecast.predicted_mean
forecast_conf_int = forecast.conf_int()


# Plot hasil prediksi
plt.figure(figsize=(12, 6))
plt.plot(data.index, data['Palm oil ($/mt)'], label='Harga Aktual')
plt.plot(forecast_values.index, forecast_values, color='orange', label='Prediksi ARIMA (2017-2023)')
plt.fill_between(forecast_values.index,
                 forecast_conf_int.iloc[:, 0],
                 forecast_conf_int.iloc[:, 1],
                 color='pink', alpha=0.3)
plt.title('Prediksi Harga Minyak Kelapa Sawit dengan Model ARIMA (2017-2023)')
plt.xlabel('Tahun')
```

```
plt.ylabel('Harga ($/mt)')

plt.legend()

plt.grid(True)

plt.show()


test_data = data.loc[start_date:end_date, 'Palm oil ($/mt)']


forecast = arima_result.get_prediction(start=start_date, end=end_date)

predicted_values = forecast.predicted_mean

mse = mean_squared_error(test_data, predicted_values)

print(f"Mean Squared Error (MSE): {mse}")


# Hitung RMSE (dengan akar dari MSE)

rmse = np.sqrt(mse)


print(f"Root Mean Squared Error (RMSE): {rmse}")


# Bangun model ARIMA

model = ARIMA(data['Palm oil ($/mt)'], order=(0, 0, 1))

arima_result = model.fit()


# Ringkasan model
```

```
print(arima_result.summary())


# Prediksi untuk periode 2017 hingga 2023
start_date = '2000'
end_date = '2023'
forecast = arima_result.get_prediction(start=pd.to_datetime(start_date), end=pd.to_datetime(end_date))
forecast_values = forecast.predicted_mean
forecast_conf_int = forecast.conf_int()


# Plot hasil prediksi
plt.figure(figsize=(12, 6))
plt.plot(data.index, data['Palm oil ($/mt)'], label='Harga Aktual')
plt.plot(forecast_values.index, forecast_values, color='orange', label='Prediksi ARIMA (2017-2023)')
plt.fill_between(forecast_values.index,
                 forecast_conf_int.iloc[:, 0],
                 forecast_conf_int.iloc[:, 1],
                 color='pink', alpha=0.3)
plt.title('Prediksi Harga Minyak Kelapa Sawit dengan Model ARIMA (2017-2023)')
plt.xlabel('Tahun')
plt.ylabel('Harga ($/mt)')
```

```
plt.legend()

plt.grid(True)

plt.show()

test_data = data.loc[start_date:end_date, 'Palm oil ($/mt)']


forecast = arima_result.get_prediction(start=start_date, end=end_date)

predicted_values = forecast.predicted_mean

mse = mean_squared_error(test_data, predicted_values)

print(f"Mean Squared Error (MSE): {mse}")


# Hitung RMSE (dengan akar dari MSE)

rmse = np.sqrt(mse)


print(f"Root Mean Squared Error (RMSE): {rmse}")


# Bangun model ARIMA

model = ARIMA(data['Palm oil ($/mt)'], order=(2, 1, 0))

arima_result = model.fit()


# Ringkasan model

print(arima_result.summary())
```

```
# Prediksi untuk periode 2017 hingga 2023
start_date = '2000'
end_date = '2023'
forecast = arima_result.get_prediction(start=pd.to_datetime(start_date), end=pd.to_datetime(end_date))
forecast_values = forecast.predicted_mean
forecast_conf_int = forecast.conf_int()

# Plot hasil prediksi
plt.figure(figsize=(12, 6))
plt.plot(data.index, data['Palm oil ($/mt)'], label='Harga Aktual')
plt.plot(forecast_values.index, forecast_values, color='orange', label='Prediksi ARIMA (2017-2023)')
plt.fill_between(forecast_values.index,
                 forecast_conf_int.iloc[:, 0],
                 forecast_conf_int.iloc[:, 1],
                 color='pink', alpha=0.3)
plt.title('Prediksi Harga Minyak Kelapa Sawit dengan Model ARIMA (2017-2023)')
plt.xlabel('Tahun')
plt.ylabel('Harga ($/mt)')
plt.legend()
plt.grid(True)
```

```
plt.show()


test_data = data.loc[start_date:end_date, 'Palm oil ($/mt)']


forecast = arima_result.get_prediction(start=start_date, end=end_date)
predicted_values = forecast.predicted_mean
mse = mean_squared_error(test_data, predicted_values)
print(f"Mean Squared Error (MSE): {mse}")


# Hitung RMSE (dengan akar dari MSE)
rmse = np.sqrt(mse)


print(f"Root Mean Squared Error (RMSE): {rmse}")


# Bangun model ARIMA
model = ARIMA(data['Palm oil ($/mt)'], order=( 0, 1, 2))
arima_result = model.fit()


# Ringkasan model
print(arima_result.summary())


# Prediksi untuk periode 2017 hingga 2023
```

```
start_date = '2000'

end_date = '2023'

forecast = arima_result.get_prediction(start=pd.to_datetime(start_date), end=pd.to_datetime(end_date))

forecast_values = forecast.predicted_mean

forecast_conf_int = forecast.conf_int()


# Plot hasil prediksi

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

plt.plot(data.index, data['Palm oil ($/mt)'], label='Harga Aktual')

plt.plot(forecast_values.index, forecast_values, color='orange', label='Prediksi ARIMA (2017-2023)')

plt.fill_between(forecast_values.index,

                 forecast_conf_int.iloc[:, 0],

                 forecast_conf_int.iloc[:, 1],

                 color='pink', alpha=0.3)

plt.title('Prediksi Harga Minyak Kelapa Sawit dengan Model ARIMA (2017-2023)')

plt.xlabel('Tahun')

plt.ylabel('Harga ($/mt)')

plt.legend()

plt.grid(True)

plt.show()
```

```
# Pastikan test_data adalah subset dari dataset asli
test_data = data.loc[start_date:end_date, 'Palm oil ($/mt)']

forecast = arima_result.get_prediction(start=start_date, end=end_date)
predicted_values = forecast.predicted_mean
mse = mean_squared_error(test_data, predicted_values)
print(f"Mean Squared Error (MSE): {mse}")

# Hitung RMSE (dengan akar dari MSE)
rmse = np.sqrt(mse)

print(f"Root Mean Squared Error (RMSE): {rmse}")
```