DAFTAR PUSTAKA

Anzelia, N. P., Desmintari & Sugianto, n.d. Pengaruh Nilai Tukar, Jumlah Uang Beredar, dan Tingkat Inflasi Terhadap Pertumbuhan Ekonomi di Indonesia. *Jurnal Ilmu Ekonomi dan Sosial,* Volume 10, pp. 88-100.

Athey, S. & Imbens, G. W., 2019. Machine Learning Methods That Economists Should Know About. *Annual Review of Economics,* Volume 11.

Chauhan, N. K. & Singh, K., 2018. A Review on Conventional Machine Learning vs Deep Learning. *N. K. Chauhan and K. Singh, "A Review on Conventional Machine LearningInternational Conference on Computing, Power and Communication Technologies (GUCON),* pp. 347-352.

Elvierayani, R. R., 2017. Peramalan Nilai Tukar (Kurs) Rupiah Terhadap Dolar Tahun 2017 dengan Menggunakan Metode Arima Box-Jenkins. *Prosiding SI MaNIs (Seminar Nasional Integrasi Matematika dan Nilai Islami),* Volume 1, pp. 253-261.

Ensafia, Y., Amin, S. H., Zhang, G. & Shah, B., 2022. Time-series forecasting of seasonal items sales using machine learning –A comparative analysis. *International Journal of Information Management Data Insights 2.*

Ernita, D., 2021. ANALISIS PENGARUH INFLASI DAN PRODUK DOMESTIK REGIONAL BRUTO TERHADAP PENDAPATAN ASLI DAERAH KABUPATEN KERINCI. *Indonesian Journal of Business and Management.*

Fahmi, M., 2017. *Peramalan Kebutuhan Pelumas Castrol di PT. ASTRA Internasional Daihatsu Dengan Metode ARIMA Untuk Optimasi Persediaan.* Makassar: Universitas Islam Negeri Alauddin.

Gaudron, R. & Morgans, A. S., 2022. Thermoacoustic stability prediction using classification algorithms. *Data-Centric Engineering.*

Gikungu, S. W., Waititu, A. G. & Kihoro, J. M., 2015. Forecasting Inflation Rate in Kenya Using SARIMA Model. *American Journal of Theoretical and Applied Statistics,* 4(1), pp. 15-19.

Gooijer, J. G. D. & Hyndman, R. J., 2006. 25 Years of Time SeriesForecasting.International Journal of Forecasting. *International Journal of Forecasting,* Volume 22, pp. 443-473.

Hauzenberger, N., Huber, F. & Klieber, K., 2023. Real-time inflation forecasting using non-linear dimension. *International Journal of Forecasting,* pp. 901-921.

Hodijah, S. & Angelina, G. P., 2021. ANALISIS PENGARUH EKSPOR DAN IMPOR TERHADAP PERTUMBUHAN EKONOMIDIINDONESIA. *Jurnal Manajemen Terapan dan Keuangan.*

Hyndman, R. J. & Athanasopoulos, G., 2021. *Forecasting: Principles and Practice.* 3 ed. Australia: Monash University.

Id, I. d., 2021. *Machine Learning: Teori, studi kasus dan implementasi menggunakan phyton.* Riau: UR PRESS.

Jabat, D. E. B., Sipayung, L. Y. & Dakhi, K. R. S., 2024. Penerapan Algoritma Recurrent Neural Networks (RNN) Untuk Klasifikasi Ulos Batak Toba. *Seminar Nasional Inovasi Sains Teknologi Informasi Komputer,* pp. 545-550.

Jonathan D. Cryer, K.-S. C., 2008. *Time Series Analysis with Application in R.* New york: Springer.

Longo, L., Riccaboni, M. & Rungi, A., 2022. A neural network ensemble approach for GDP forecasting. *Journal of Economic Dynamics & Control.*

Lusiana, A. & Yuliarty, P., 2020. PENERAPAN METODE PERAMALAN (FORECASTING) PADA PERMINTAAN ATAP di PT X. *Jurnal Teknik Industri ITN Malang,* pp. 11-20.

Mahesh, B., 2020. Machine Learning Algorithms - A Review. *International Journal of Science and Research (IJSR),* pp. 381-386.

McKinney, W., 2013. *Phyton for Data Analysis.* Sebastopol: O'Reilly.

Mullainathan, S. & Spiess, J., 2017. Machine Learning: An Applied Econometric Approach. *Journal of Economic Perspectives,* Volume 31, pp. 87-106.

Ospina, R., Gondim, J. A. M., Leiva, V. & Castro, a. C., 2023. An Overview of Forecast Analysis with ARIMA Models during the COVID-19 Pandemic: Methodology and Case Study in Brazil. *Mathematics.*

Panjaitan, P. D. et al., 2024. *Pengatar Ekonomi Makro.* Batam: CV.Rey Media Grafika.

Prayoga, A. H. & Voutama, A., 2024. PENGEMBANGAN APLIKASI BANK ACCOUNT FRAUD DETECTION DENGAN MENGGUNAKAN ALGORITMA XGBOOST. *Jurnal Mahasiswa Teknik Informatika.*

Richardson, A., a, T. v. F. M. & a, T. V., 2021. Nowcasting GDP using machine-learning algorithms: A real-time assessment. *International Journal of Forecasting,* pp. 941-948.

Rodríguez-Vargas, d., 2020. Forecasting Costa Rican inflation with machine learning methods. *Latin American Journal of Central Banking 1.*

SAPUTRI, A. R. A., 2019. Metode Penghalusan Eksponensial Holt-Winter dan Metode Autoregressive Integrated Moving Average (ARIMA). *Digital Repository Unila.*

Seto, S., Nita, Y. & Triana, L., 2016. *Manajemen Farmasi: Lingkungan Apotek, Farmasi Rumah Sakit, Indutri Farmasi, Pedagang Besar Farmas.* Surabaya: Airlangga University Press.

Silitonga, D., 2021. PENGARUH INFLASI TERHADAP PRODUK DOMESTIK BRUTO (PDB) INDONESIA PADA PERIODE TAHUN 2010-2020. *Jurnal Manajemen Bisnis,* Volume 24, pp. 111-122.

Supriatna, A., Susanti, D. & Hertini, E., 2017. Application of Holt exponential smoothing and ARIMA method for data population in West Java. *IOP Conference Series: Materials Science and Engineering.*

Tantika, H. N. & Nanang Supriadi, D. A., 2018. Metode Seasonal ARIMA untuk Meramalkan Produksi Kopi Dengan Indikator Curah Hujan Menggunakan Aplikasi R di Kabupaten Lampung Barat. *Jurnal Matematika,* Volume 17.

Truong, Q., Nguyen, M., Dang, H. & Mei, B., 2020. Housing Price Prediction via Improved Machine Learning. *Procedia Computer Science,* pp. 433-442.

Wei, W. W., 2006. Time Series Univariate and Multivariate Method. *Pearson Addison Wesley, Boston.*

LAMPIRAN

Lampiran 1. *Coding* Program Inflasi

```
# Loading Library
import pandas as pd
import numpy as np
import os
import matplotlib.pyplot as plt
plt.style.use('fivethirtyeight')
import warnings
warnings.filterwarnings('ignore')
# establish path data
data_path = r'C:\JACQLIEN\INFLASI'
# import data
def load_data(file_name):


    # load data
    dataset = pd.read_excel(os.path.join(data_path, file_name))


    # unpivot from wide to long format
    dataset = dataset.melt(id_vars='Year', var_name='Month',
value_name='Rate')


    from pandas.tseries.offsets import MonthEnd


    # assign last day of month
    dataset['Date'] = pd.to_datetime(dataset[['Year',
'Month']].assign(DAY=1)) + MonthEnd(1)


    # order ascending data values
    dataset = dataset.sort_values(by='Date', ascending=True)


    # drop unnecessary columns
```

```
    dataset = dataset.drop(['Year', 'Month'], axis=1)

    # set date column as index
    dataset.set_index('Date', inplace=True)

    # drop NaN rows
    dataset.dropna(subset=['Rate'], inplace=True)

    return dataset

file_name='Inflasi Sulsel Umum yoy.xlsx'
df = load_data(file_name)
df.tail()
print(df)

# visualize target data
def plot_time_series(series):
    mean_rolling = series.rolling(window=12).mean()
    std_rolling = series.rolling(window=12).std()

    # plot inflation rates
    series.plot(figsize=(12, 5), label='Original')
    mean_rolling.plot(color='crimson', label='Rolling Mean')
    std_rolling.plot(color='black', label='Rolling Std')
    plt.title('Grafik'+' '+file_name)
    plt.grid(axis='y', alpha=0.5)
    plt.legend(loc='best')
    plt.show()

    from statsmodels.tsa.seasonal import seasonal_decompose
```

```
    # plot decomposition components
    decomp = seasonal_decompose(series, model='additive')
    fig, axes = plt.subplots(ncols=1, nrows=4, sharex=True,
figsize=(12, 5))
    fig.suptitle('Seasonal Decomposition')

    decomp.trend.plot(ax=axes[0], legend=False)
    axes[0].set_ylabel('Trend')

    decomp.seasonal.plot(ax=axes[1], legend=False)
    axes[1].set_ylabel('Seasonal')

    decomp.resid.plot(ax=axes[2], legend=False)
    axes[2].set_ylabel('Residual')

    decomp.observed.plot(ax=axes[3], legend=False)
    axes[3].set_ylabel('Original')
    plt.show()

plot_time_series(df['Rate'])
# ADF statistical test
def adf_test(series):
    from statsmodels.tsa.stattools import adfuller

    result = adfuller(series, regression='c', autolag='AIC')
    print('=======  Augmented  Dickey-Fuller  Test  Results
=======\n')
    print('1. ADF Test Statistic: {:.6f}'.format(result[0]))
    print('2. P-value: {:.6f}'.format(result[1]))
    print('3. Used Lags: {}'.format(result[2]))
    print('4. Used Observations: {}'.format(result[3]))
    print('5. Critical Values:')
```

```
    for key, value in result[4].items():
        print('\t{}: {:.6f}'.format(key, value))

    critical_value = result[4]['5%']
    if (result[1] <= 0.05) and (result[0] < critical_value):
        print('\nStrong evidence against the null hypothesis (H0), reject the null hypothesis.\
        Data has no unit root and is stationary.')
    else:
        print('\nWeak evidence against null hypothesis, time series has a unit root, indicating it is non-stationary.')
    return

# run function
adf_test(df['Rate'])
# ARIMA Method
from pmdarima.arima.utils import ndiffs
print("ADF Test: ", ndiffs(df['Rate'], test='adf'))
print("KPSS Test: ",ndiffs(df['Rate'], test='kpss'))
print("PP Test: ",ndiffs(df['Rate'], test='pp'))


# plotting ACF and PACF
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
fig = plt.figure(figsize=(8,7))
ax1 = fig.add_subplot(2,1,1)
fig = plot_acf(df, ax=ax1)
ax2 = fig.add_subplot(2,1,2)
fig = plot_pacf(df, ax=ax2)
plt.show()

# Original Series
```

```
fig, (ax1, ax2, ax3) = plt.subplots(3)

ax1.plot(df['Rate']);      ax1.set_title('Original      Series');
ax1.axes.xaxis.set_visible(False)

# 1st Differencing

ax2.plot(df['Rate'].diff());       ax2.set_title('1st       Order
Differencing'); ax2.axes.xaxis.set_visible(False)

# 2nd Differencing

ax3.plot(df['Rate'].diff().diff());   ax3.set_title('2nd   Order
Differencing')

plt.show()


from statsmodels.graphics.tsaplots import plot_acf

fig, (ax1, ax2, ax3) = plt.subplots(3, 1, figsize=(10, 8))


# Plot ACF untuk Data Asli

plot_acf(df['Rate'], ax=ax1, lags=20)

ax1.set_title('ACF of Original Series')


# Plot ACF untuk 1st Order Differencing

plot_acf(df['Rate'].diff().dropna(), ax=ax2, lags=20)

ax2.set_title('ACF of 1st Order Differencing')


# Plot ACF untuk 2nd Order Differencing

plot_acf(df['Rate'].diff().diff().dropna(), ax=ax3, lags=20)

ax3.set_title('ACF of 2nd Order Differencing')


plt.tight_layout()

plt.show()


from statsmodels.graphics.tsaplots import plot_pacf

fig, (ax1, ax2, ax3) = plt.subplots(3, 1, figsize=(10, 8))

# Plot PACF untuk Data Asli
```

```
plot_pacf(df['Rate'], ax=ax1, lags=20)
ax1.set_title('PACF of Original Series')

# Plot PACF untuk 1st Order Differencing
plot_pacf(df['Rate'].diff().dropna(), ax=ax2, lags=20)
ax2.set_title('PACF of 1st Order Differencing')

# Plot PACF untuk 2nd Order Differencing
plot_pacf(df['Rate'].diff().diff().dropna(), ax=ax3, lags=20)
ax3.set_title('PACF of 2nd Order Differencing')

plt.tight_layout()
plt.show()
# Prediksi Auto Arima
from pmdarima.arima import auto_arima

model_fit_auto = auto_arima(df['Rate'], start_p=1, start_q=1,
                      test='adf',
                      max_p=5, max_q=5,
                      m=4,
                      d=1,
                      seasonal=True,
                      start_P=0,
                      D=None,
                      trace=True,
                      error_action='ignore',
                      suppress_warnings=True,
                      stepwise=True)

model_fit_auto.summary()
```

```
forecast_arima_auto                                                    =
pd.concat([model_fit_auto.predict_in_sample(),
model_fit_auto.predict(n_periods=5,                typ='levels',
dynamic=False)])


print("last 5 data")

print(forecast_arima_auto.tail(5))

print("forecasted data")

print(forecast_arima_auto.tail(5))


# final plot

forecast_arima_auto.iloc[:].plot(legend=True,   label='Forecast',
figsize=(14,5), color='red')

df['Rate'].iloc[:].plot(legend=True, label='Actual')

plt.title('Proyeksi Auto ARIMA'+' '+file_name)

plt.show()

# Prediksi Arima Manual

from statsmodels.tsa.arima.model import ARIMA


# fitting the model

model_arima_manual = ARIMA(df['Rate'], order=(1, 1 ,1), freq='M')

model_fit_manual = model_arima_manual.fit()

print(model_fit_manual.summary())

# predict values

forecast_arima_manual     =     model_fit_manual.predict(start=0,
end=len(df) + 4, typ='levels', dynamic=False)

print(forecast_arima_manual.tail(5))


# final plot

forecast_arima_manual.iloc[:].plot(legend=True,
label='Forecast', figsize=(14,5), color='red')

df['Rate'].iloc[:].plot(legend=True, label='Actual')

plt.title('Proyeksi Manual'+' '+file_name)
```

```
plt.show()

# Model Evaluation

pred_arima_manual = model_fit_manual.predict(start=0, end=len(df)
- 1, typ='levels', dynamic=False)

pred_arima_auto = model_fit_auto.predict_in_sample()


model_fit_manual.plot_diagnostics(figsize=(9,9))

plt.show()


model_fit_auto.plot_diagnostics(figsize=(9,9))

plt.show()

from sklearn.metrics import mean_squared_error

from sklearn.metrics import mean_absolute_error

rmse_manual = mean_squared_error(pred_arima_manual, df['Rate'],
squared=False)

corr_manual = np.corrcoef(pred_arima_manual, df['Rate'])[0,1]

mae_manual = mean_absolute_error(pred_arima_manual, df['Rate'])


rmse_auto = mean_squared_error(pred_arima_auto, df['Rate'],
squared=False)

corr_auto = np.corrcoef(pred_arima_auto, df['Rate'])[0,1]

mae_auto = mean_absolute_error(pred_arima_auto, df['Rate'])


table_eval = pd.DataFrame([['rmse', rmse_manual, rmse_auto],
['corr', corr_manual, corr_auto],

                           ['mae', mae_manual, mae_auto]],

                          columns=['Type','Manual', 'Auto'])


print(table_eval)

if rmse_auto < rmse_manual:

    print('Pilih Hasil Model Auto Arima')

else:

    print('Pilih Hasil Model Manual Arima')
```

Lampiran 2. *Coding* Program PDRB

```
import pandas as pd
import numpy as np
import os
import matplotlib.pyplot as plt
from pptx import Presentation
plt.style.use('fivethirtyeight')
import warnings
warnings.filterwarnings('ignore')
# establish path data
data_path = r'C:\JACQLIEN\PDRB'
file_name = 'Data PDRB Sulsel.xlsx'
df = pd.read_excel(os.path.join(data_path, file_name))
df['date'] = (
    pd.to_datetime(
        df['tanggal'].str.split(' ').apply(lambda x:
''.join(x[::-1]))
))
df['date'] = df['date'] + pd.offsets.QuarterEnd(0)
df = df.drop(['tanggal'], axis=1)
df.set_index('date', inplace=True)
df.tail()
def plot_time_series(series):
    mean_rolling = series.rolling(window=12).mean()
    std_rolling = series.rolling(window=12).std()

    # plot inflation rates
    series.plot(figsize=(12, 5), label='Original')
    mean_rolling.plot(color='crimson', label='Rolling Mean')
    std_rolling.plot(color='black', label='Rolling Std')
    plt.title('Grafik'+' '+file_name)
```

```
    plt.grid(axis='y', alpha=0.5)
    plt.legend(loc='best')
    plt.show()

    from statsmodels.tsa.seasonal import seasonal_decompose

    # plot decomposition components
    decomp = seasonal_decompose(series, model='additive')
    fig, axes = plt.subplots(ncols=1, nrows=4, sharex=True,
figsize=(12, 5))
    fig.suptitle('Seasonal Decomposition')

    decomp.trend.plot(ax=axes[0], legend=False)
    axes[0].set_ylabel('Trend')

    decomp.seasonal.plot(ax=axes[1], legend=False)
    axes[1].set_ylabel('Seasonal')

    decomp.resid.plot(ax=axes[2], legend=False)
    axes[2].set_ylabel('Residual')

    decomp.observed.plot(ax=axes[3], legend=False)
    axes[3].set_ylabel('Original')
    plt.show()
def adf_test(series):
    from statsmodels.tsa.stattools import adfuller

    result = adfuller(series, regression='c', autolag='AIC')
    print('======= Augmented Dickey-Fuller Test Results
=======\n')
    print('1. ADF Test Statistic: {:.6f}'.format(result[0]))
    print('2. P-value: {:.6f}'.format(result[1]))
```

```
    print('3. Used Lags: {}'.format(result[2]))
    print('4. Used Observations: {}'.format(result[3]))
    print('5. Critical Values:')
    for key, value in result[4].items():
        print('\t{}: {:.6f}'.format(key, value))


    critical_value = result[4]['5%']
    if (result[1] <= 0.05) and (result[0] < critical_value):
        print('\nStrong evidence against the null hypothesis
(H0), reject the null hypothesis.\
        Data has no unit root and is stationary.')
    else:
        print('\nWeak evidence against null hypothesis, time
series has a unit root, indicating it is non-stationary.')
    return
def define_pdq(column):
    from pmdarima.arima.utils import ndiffs
    from statsmodels.graphics.tsaplots import plot_acf,
plot_pacf
    import matplotlib.pyplot as plt


    # Calculating the number of differences required
    print("ADF Test: ", ndiffs(df[column], test='adf'))
    print("KPSS Test: ", ndiffs(df[column], test='kpss'))
    print("PP Test: ", ndiffs(df[column], test='pp'))


    # Plotting ACF and PACF for the original series
    fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(8, 7))
    plot_pacf(df[column], ax=ax1, title='PACF - Original
Series')
    plot_acf(df[column], ax=ax2, title='ACF - Original Series')
    plt.tight_layout()
    plt.show()
```

```
    # Plotting Original Series and its Differencing

    fig, (ax1, ax2, ax3) = plt.subplots(3, 1, figsize=(10, 8))

    ax1.plot(df[column]); ax1.set_title('Original Series');
ax1.axes.xaxis.set_visible(False)

    ax2.plot(df[column].diff()); ax2.set_title('1st Order
Differencing'); ax2.axes.xaxis.set_visible(False)

    ax3.plot(df[column].diff().diff()); ax3.set_title('2nd Order
Differencing')

    plt.tight_layout()

    plt.show()


    # Plotting ACF and PACF for the differenced series

    fig, ax = plt.subplots(3, 2, figsize=(12, 10))

    plot_acf(df[column], ax=ax[0, 0], title='ACF - Original
Series')

    plot_pacf(df[column], ax=ax[0, 1], title='PACF - Original
Series')

    plot_acf(df[column].diff().dropna(), ax=ax[1, 0], title='ACF
- 1st Order Differencing')

    plot_pacf(df[column].diff().dropna(), ax=ax[1, 1],
title='PACF - 1st Order Differencing')

    plot_acf(df[column].diff().diff().dropna(), ax=ax[2, 0],
title='ACF - 2nd Order Differencing')

    plot_pacf(df[column].diff().diff().dropna(), ax=ax[2, 1],
title='PACF - 2nd Order Differencing')

    plt.tight_layout()

    plt.show()


# Analisis

nama_kolom = 'yoy'


plot_time_series(df[nama_kolom])

adf_test(df[nama_kolom])

define_pdq(nama_kolom)
```

```
# (p, d, q) = ( 1, 1 ,1) -> from graph

# Prediksi Auto Arima

from pmdarima.arima import auto_arima


model_fit_auto = auto_arima(df[nama_kolom], start_p=1,
start_q=1,

                        test='adf',

                        max_p=5, max_q=5,

                        m=4,

                        d=1,

                        seasonal=True,

                        start_P=0,

                        D=None,

                        trace=True,

                        error_action='ignore',

                        suppress_warnings=True,

                        stepwise=True)


model_fit_auto.summary()

forecast_arima_auto =
pd.concat([model_fit_auto.predict_in_sample(),
model_fit_auto.predict(n_periods=5, typ='levels',
dynamic=False)])


print("last 5 data")

print(forecast_arima_auto.tail(5))

print("forecasted data")

print(forecast_arima_auto.tail(5))


# final plot

forecast_arima_auto.iloc[:].plot(legend=True, label='Forecast',
figsize=(14,5), color='red')

df[nama_kolom].iloc[:].plot(legend=True, label='Actual')
```

```
plt.title('Proyeksi Auto ARIMA'+' '+file_name)

plt.show()

# Prediksi Arima Manual

from statsmodels.tsa.arima.model import ARIMA


# fitting the model

model_arima_manual = ARIMA(df[nama_kolom], order=(1, 1 ,1),
freq='Q')

model_fit_manual = model_arima_manual.fit()

print(model_fit_manual.summary())


# predict values

forecast_arima_manual = model_fit_manual.predict(start=0,
end=len(df) + 4, typ='levels', dynamic=False)

print(forecast_arima_manual.tail(5))


# final plot

forecast_arima_manual.iloc[:].plot(legend=True,
label='Forecast', figsize=(14,5), color='red')

df[nama_kolom].iloc[:].plot(legend=True, label='Actual')

plt.title('Proyeksi Manual ARIMA'+' '+file_name)

plt.show()


# Model Evaluation

pred_arima_manual = model_fit_manual.predict(start=0,
end=len(df) - 1, typ='levels', dynamic=False)

pred_arima_auto = model_fit_auto.predict_in_sample()


model_fit_manual.plot_diagnostics(figsize=(9,9))

plt.show()


model_fit_auto.plot_diagnostics(figsize=(9,9))

plt.show()
```

```
from sklearn.metrics import mean_squared_error

from sklearn.metrics import mean_absolute_error

rmse_manual = mean_squared_error(pred_arima_manual,
df[nama_kolom], squared=False)

corr_manual = np.corrcoef(pred_arima_manual,
df[nama_kolom])[0,1]

mae_manual = mean_absolute_error(pred_arima_manual,
df[nama_kolom])


rmse_auto = mean_squared_error(pred_arima_auto, df[nama_kolom],
squared=False)

corr_auto = np.corrcoef(pred_arima_auto, df[nama_kolom])[0,1]

mae_auto = mean_absolute_error(pred_arima_auto, df[nama_kolom])


table_eval = pd.DataFrame([['rmse', rmse_manual, rmse_auto],
['corr', corr_manual, corr_auto],
                           ['mae', mae_manual, mae_auto]],
                          columns=['Type','Manual', 'Auto'])


print(table_eval)

if rmse_auto < rmse_manual:

    print('Pilih Hasil Model Auto Arima')

else:

    print('Pilih Hasil Model Manual Arima')
```

Lampiran 3. Diskusi bersama ahli ekonom Bank Indonesia KPw Sulawesi Selatan