```
    crit_val_10 = result[4]['10%']

    if ((p_val < significance_level) & (adf_stat <
crit_val_1)):
        linecolor = 'green'
    elif ((p_val < significance_level) & (adf_stat <
crit_val_5)):
        linecolor = 'orange'
    elif ((p_val < significance_level) & (adf_stat <
crit_val_10)):
        linecolor = 'red'
    else:
        linecolor = 'purple'

    sns.lineplot(x=df['Minggu'], y=series, ax=ax,
color=linecolor)
    ax.set_title(f'ADF Statistics {adf_stat:0.3f}, p-value:
{p_val:0.3f}\nCritical values 1%: {crit_val_1:0.3f}, 5%:
{crit_val_5:0.3f}, 10%: {crit_val_10:0.3f}', fontsize=14)
    ax.set_ylabel(ylabel=title, fontsize=14)


# (df['perumahan: (Indonesia)'].values, 'Rainfall', ax[0, 0])

# f.delaxes(ax[0,0])
# plt.tight_layout()
# plt.show()
ts_diff = np.diff(df['perumahan: (Indonesia)'])

df['perumahan: (Indonesia)_diff'] = np.append([0], ts_diff)

f, ax = plt.subplots(nrows=1, ncols=1, figsize=(15, 6))
visualize_adfuller_results(df['perumahan: (Indonesia)_diff'],
'Keyword Setelah Differencing', ax)
from statsmodels.tsa.seasonal import seasonal_decompose

core_columns = [
    'perumahan: (Indonesia)'
]

for col in core_columns:
    decomp = seasonal_decompose(df[col], period=52,
model='additive', extrapolate_trend = 'freq')
    df[f'{col}_trend'] = decomp.trend
    df[f'{col}_seasonal'] = decomp.seasonal
```

```
f, ax = plt.subplots(ncols=2, nrows=4, sharex=True,
figsize=(16, 8))

for i, column in enumerate(['perumahan: (Indonesia)']):
    res = seasonal_decompose(df[column], period=52,
model='additive', extrapolate_trend = 'freq')

    ax[0, i].set_title(f'Decomposition of {column}',
fontsize=14)
    res.observed.plot(ax=ax[0, i], legend=False, color='blue')
    ax[0, i].set_ylabel('Observed', fontsize=14)

    res.trend.plot(ax=ax[1, i], legend=False,color='blue')
    ax[1, i].set_ylabel('Trend', fontsize=14)

    res.seasonal.plot(ax=ax[2, i], legend=False,color='blue')
    ax[2, i].set_ylabel('Seasonal', fontsize=14)

    res.trend.plot(ax=ax[3, i], legend=False,color='blue')
    ax[3, i].set_ylabel('Trend', fontsize=14)
for i in df.columns:
    print(i)
autocorrelation_plot(df['perumahan: (Indonesia)_diff'])
plt.show()
f, ax = plt.subplots(nrows=2, ncols=1, figsize=(16,8))

plot_acf(df['perumahan: (Indonesia)_diff'], lags=100,
ax=ax[0])
plot_pacf(df['perumahan: (Indonesia)_diff'], lags=100,
ax=ax[1])

plt.show()
N_SPLITS = 3

X = df['Minggu']
y = df['perumahan: (Indonesia)']

folds = TimeSeriesSplit(n_splits=N_SPLITS)
f, ax = plt.subplots(nrows=N_SPLITS, ncols=2, figsize=(16, 9))

for i, (train_index, valid_index) in
enumerate(folds.split(X)):
    X_train, X_valid = X[train_index], X[valid_index]
    y_train, y_valid = y[train_index], y[valid_index]
```

```
    sns.lineplot(
        x=X_train,
        y=y_train,
        ax=ax[i,0],
        color='dodgerblue',
        label='train'
    )
    sns.lineplot(
        x=X_train[len(X_train) - len(X_valid):(len(X_train) -
len(X_valid) + len(X_valid))],
        y=y_train[len(X_train) - len(X_valid):(len(X_train) -
len(X_valid) + len(X_valid))],
        ax=ax[i,1],
        color='dodgerblue',
        label='train'
    )

    for j in range(2):
        sns.lineplot(x= X_valid, y= y_valid, ax=ax[i, j],
color='darkorange', label='validation')
    ax[i, 0].set_title(f"Rolling Window with Adjusting
Training Size (Split {i+1})", fontsize=16)
    ax[i, 1].set_title(f"Rolling Window with Constant Training
Size (Split {i+1})", fontsize=16)

for i in range(N_SPLITS):
    ax[i, 0].set_xlim([date(2018, 1, 1), date(2023, 12, 30)])
    ax[i, 1].set_xlim([date(2018, 1, 1), date(2023, 12, 30)])

plt.tight_layout()
plt.show()
train_size = int(0.85 * len(df))
test_size = len(df) - train_size

univariate_df = df[['Minggu', 'perumahan:
(Indonesia)']].copy()

univariate_df.columns = ['ds', 'y']

train = univariate_df.iloc[:train_size, :]

x_train, y_train =
pd.DataFrame(univariate_df.iloc[:train_size, 0]),
pd.DataFrame(univariate_df.iloc[:train_size, 1])
```

```
x_valid, y_valid =
pd.DataFrame(univariate_df.iloc[train_size:, 0]),
pd.DataFrame(univariate_df.iloc[train_size:, 1])

print(len(train), len(x_valid))

!pip install prophet
from sklearn.metrics import mean_absolute_error,
mean_squared_error
import math
from prophet import Prophet
model = Prophet()
model.fit(train)

y_pred = model.predict(x_valid)

score_mae = mean_absolute_error(y_valid,
y_pred.tail(test_size)['yhat'])
score_rmse = math.sqrt(mean_squared_error(y_valid,
y_pred.tail(test_size)['yhat']))

print(f'RMSE: {score_rmse}, MAE: {score_mae}')
f, ax = plt.subplots(1)
f.set_figheight(6)
f.set_figwidth(12)

model.plot(y_pred, ax=ax)
sns.lineplot(x=x_valid['ds'], y=y_valid['y'], ax=ax,
color='green', label='Ground Truth')

ax.set_title(f'Prediction \n MAE:{score_mae},
RMSE:{score_rmse}', fontsize=14)
ax.set_xlabel(xlabel='Tanggal', fontsize=14)
ax.set_ylabel(ylabel='Jml Keyword', fontsize=14)

plt.show()
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.graphics.tsaplots import plot_predict

model = ARIMA(y_train, order=(15, 1, 35))

model_fit = model.fit()

forecast_steps = 39
```

```
forecast = model_fit.forecast(steps=forecast_steps)

conf_intv = model_fit.get_forecast(forecast_steps).conf_int()

stnd_err = model_fit.bse

score_mae = mean_absolute_error(y_valid, forecast)
score_rmse = math.sqrt(mean_squared_error(y_valid, forecast))

def mean_absolute_percentage_error(y_valid, y_pred):
    y_valid, y_pred = np.array(y_valid), np.array(y_pred)
    return np.mean(np.abs((y_valid - y_pred) / y_valid)) * 100

score_mape = mean_absolute_percentage_error(y_valid, forecast)

print(f'RMSE: {score_rmse}, MAE: {score_mae}, MAPE:
{score_mape}')

f, ax = plt.subplots(1)
f.set_figheight(6)
f.set_figwidth(12)

plot_predict(model_fit, 1, 250, dynamic=False, ax=ax)
sns.lineplot(x=x_valid.index, y=y_valid['y'], ax=ax,
color='green', label='Ground truth')

ax.set_title(f'Pengujian ARIMA \n MAPE:{score_mape},
RMSE:{score_rmse}', fontsize=14)
ax.set_xlabel(xlabel = 'Minggu', fontsize=14)
ax.set_ylabel(ylabel = 'perumahan: (Indonesia)', fontsize=14)


plt.show()
import matplotlib.pyplot as plt
import seaborn as sns


f, ax = plt.subplots(1)
f.set_figheight(4)
f.set_figwidth(15)

sns.lineplot(x=x_valid.index, y=forecast, ax=ax, color='blue',
label='Predicted')
sns.lineplot(x=x_valid.index, y=y_valid['y'], ax=ax,
color='orange', label='Ground Truth')
```

```
ax.set_xlabel(xlabel='Minggu', fontsize=14)
ax.set_ylabel(ylabel='perumahan: (Indonesia)', fontsize=14)

plt.legend()
plt.show()

import pandas as pd

prediction_df = pd.DataFrame({
    'Tanggal': y_valid.index,
    'Search Volume': forecast
})

prediction_df.set_index('Tanggal', inplace=True)

print(prediction_df)
```