Tugas Kelompok “Mesin Pembelajar untuk Bisnis”

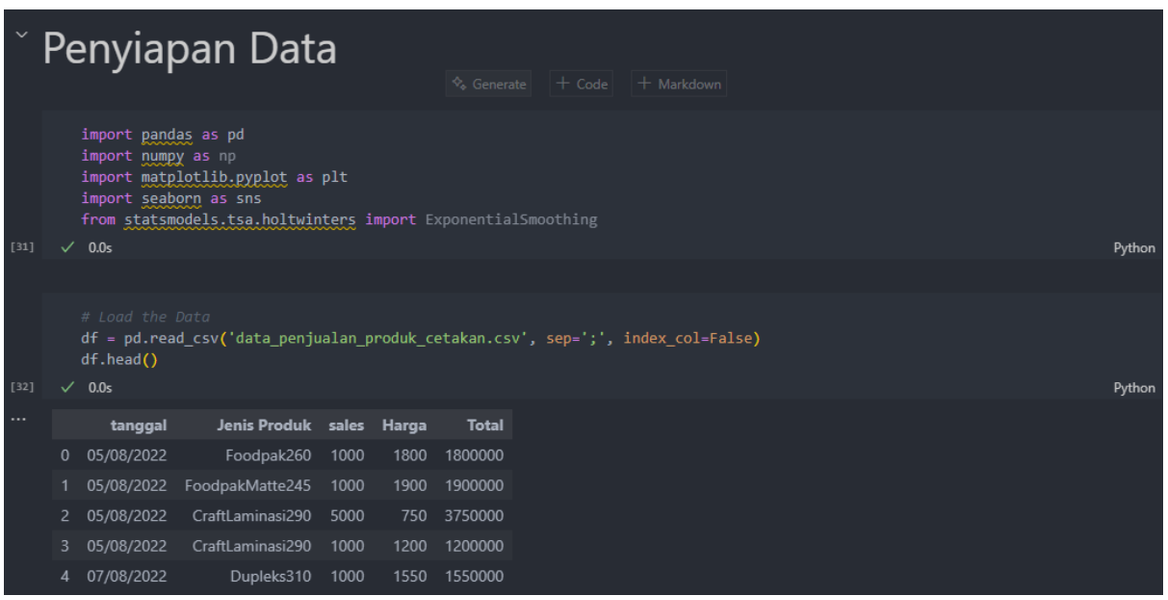
Anggota:

1. Wahyu Prasetya Adi – 23.01.85.0037
2. RR Fitri Damaryanti – 23.01.85.0038
3. Alfin Hilmy N – 23.01.85.0039
4. Randi Afif – 23.01.85.0035

Semester 2

Proyek Ujian Tengah Semester
"Analisis Time Series Penjualan Produk Cetakan menggunakan Metode ETS (Exponential Smoothing)”

A. Penyiapan Data



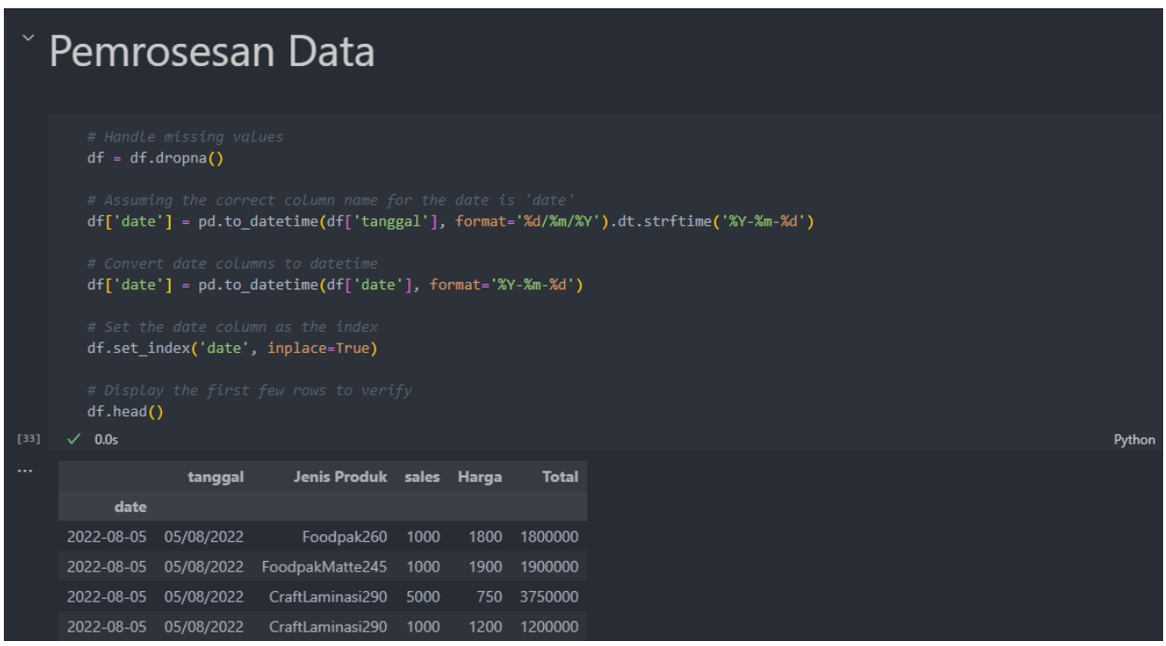
	tanggal	Jenis Produk	sales	Harga	Total
0	05/08/2022	Foodpak260	1000	1800	1800000
1	05/08/2022	FoodpakMatte245	1000	1900	1900000
2	05/08/2022	CraftLaminasi290	5000	750	3750000
3	05/08/2022	CraftLaminasi290	1000	1200	1200000
4	07/08/2022	Dupleks310	1000	1550	1550000

1. Import Library:
    - pandas (disingkat pd): digunakan untuk memanipulasi data, khususnya DataFrame.
    - numpy (disingkat np): digunakan untuk operasi numerik, meskipun tidak digunakan dalam kode ini.
    - matplotlib.pyplot (disingkat plt): digunakan untuk visualisasi data.
    - seaborn (disingkat sns): pustaka untuk visualisasi yang lebih estetis, meskipun belum digunakan di sini.
    - ExponentialSmoothing dari statsmodels.tsa.holtwinters: digunakan untuk menerapkan metode Exponential Smoothing pada data deret waktu (time series).
2. Memuat Data:

- df = pd.read_csv('data_penjualan_produk_cetakan.csv', sep=';', index_col=False): kode ini membaca file CSV dengan nama data_penjualan_produk_cetakan.csv menggunakan pemisah (sep) titik koma (;). Data disimpan dalam DataFrame df. Parameter index_col=False menunjukkan bahwa file CSV tidak memiliki kolom indeks.

3. Menampilkan Data Awal:
   - df.head(): menampilkan lima baris pertama dari data yang telah dimuat ke DataFrame untuk memberikan gambaran umum mengenai struktur dan isi data.
4. Struktur Data:
   - Data yang ditampilkan menunjukkan kolom-kolom seperti tanggal, Jenis Produk, sales, Harga, dan Total. Ini mencakup:
     - tanggal: Tanggal penjualan.
     - Jenis Produk: Nama produk yang dijual.
     - sales: Jumlah produk yang terjual.
     - Harga: Harga satuan dari produk.
     - Total: Total pendapatan dari penjualan produk (sales × Harga).

B. Pemrosesan Data



Berikut adalah penjelasan setiap bagian dari kode:

1. **Menghapus Nilai Kosong**:

   df = df.dropna()

   Baris ini menghapus baris yang memiliki nilai NaN (kosong) di DataFrame df. Hal ini memastikan bahwa semua data yang digunakan tidak memiliki data yang hilang, yang penting untuk mencegah error saat menganalisis data.

2. **Mengonversi Kolom tanggal ke Format datetime**:

   df['date'] = pd.to_datetime(df['tanggal'], format='%d/%m/%Y').dt.strftime('%Y-%m-%d')

   Di sini, kolom tanggal dikonversi ke format datetime dengan format awal '%d/%m/%Y' (dd/mm/yyyy). Setelah itu, data diformat ulang ke format '%Y-%m-%d' (yyyy-mm-dd) agar sesuai dengan standar datetime yang lebih mudah diproses dalam analisis data.

3. **Mengatur Kolom date Sebagai Indeks**:

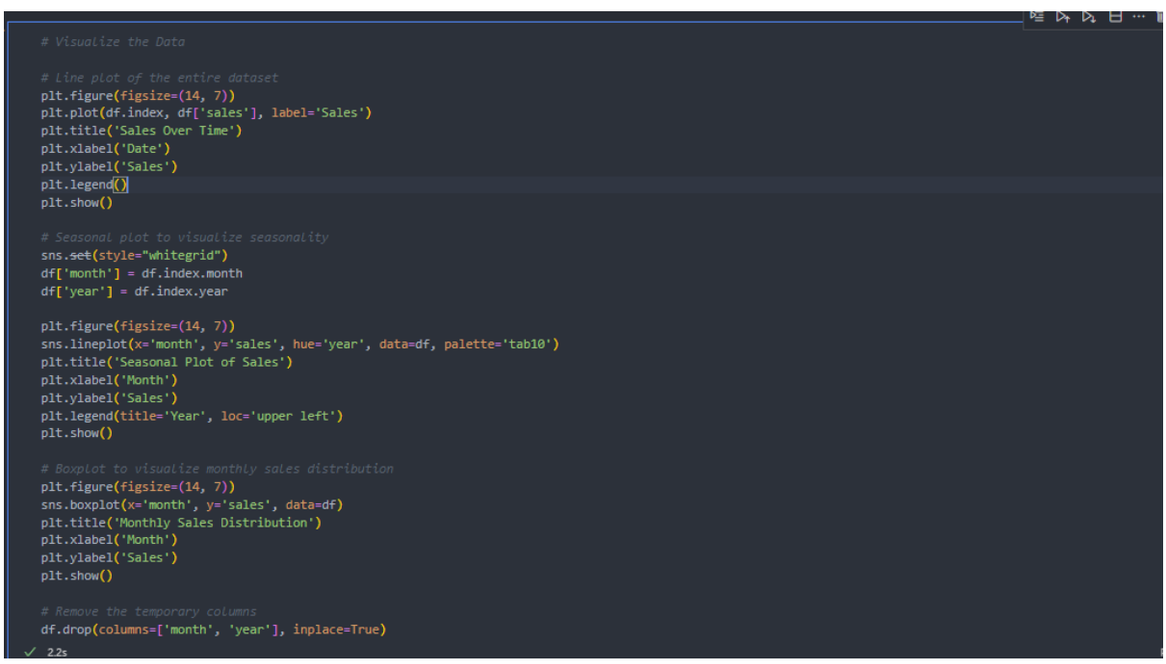
   df.set_index('date', inplace=True)

   Setelah kolom date berhasil dikonversi ke format datetime, kolom ini diatur sebagai indeks dari DataFrame. Menjadikan kolom tanggal sebagai indeks akan memudahkan dalam analisis deret waktu (time series), karena indeks datetime memudahkan penanganan dan pemrosesan data berdasarkan waktu.

4. **Menampilkan Data Awal untuk Verifikasi**:

   df.head()

   Bagian ini digunakan untuk menampilkan lima baris pertama dari DataFrame df setelah perubahan dilakukan. Ini dilakukan untuk memastikan bahwa kolom date sudah diatur sebagai indeks dan format datanya sesuai.



```
# Visualize the Data

# Line plot of the entire dataset
plt.figure(figsize=(14, 7))
plt.plot(df.index, df['sales'], label='Sales')
plt.title('Sales Over Time')
plt.xlabel('Date')
plt.ylabel('Sales')
plt.legend()
plt.show()

# Seasonal plot to visualize seasonality
sns.set(style="whitegrid")
df['month'] = df.index.month
df['year'] = df.index.year

plt.figure(figsize=(14, 7))
sns.lineplot(x='month', y='sales', hue='year', data=df, palette='tab10')
plt.title('Seasonal Plot of Sales')
plt.xlabel('Month')
plt.ylabel('Sales')
plt.legend(title='Year', loc='upper left')
plt.show()

# Boxplot to visualize monthly sales distribution
plt.figure(figsize=(14, 7))
sns.boxplot(x='month', y='sales', data=df)
plt.title('Monthly Sales Distribution')
plt.xlabel('Month')
plt.ylabel('Sales')
plt.show()

# Remove the temporary columns
df.drop(columns=['month', 'year'], inplace=True)
```

✓ 2.2s

Berikut penjelasan tiap bagian kode:

1. Line Plot untuk Seluruh Dataset

   plt.figure(figsize=(14, 7))

   plt.plot(df.index, df['sales'], label='Sales')

   plt.title('Sales Over Time')

   plt.xlabel('Date')

plt.ylabel('Sales')

plt.legend()

plt.show()

- Bagian ini membuat grafik garis (line plot) yang menampilkan data penjualan secara keseluruhan dari waktu ke waktu.
- Grafik ini membantu melihat tren umum dari penjualan, apakah cenderung naik, turun, atau memiliki pola tertentu seiring waktu.
- plt.plot() menggunakan index Date dari DataFrame sebagai sumbu X dan kolom sales sebagai sumbu Y.
- plt.title, plt.xlabel, dan plt.ylabel digunakan untuk memberikan judul dan label pada sumbu.

2. Seasonal Plot untuk Visualisasi Musiman

sns.set(style="whitegrid")

df['month'] = df.index.month

df['year'] = df.index.year

plt.figure(figsize=(14, 7))

sns.lineplot(x='month', y='sales', hue='year', data=df, palette='tab10')

plt.title('Seasonal Plot of Sales')

plt.xlabel('Month')

plt.ylabel('Sales')

plt.legend(title='Year', loc='upper left')

plt.show()

- Pada bagian ini, kolom sementara month dan year ditambahkan ke df untuk mewakili bulan dan tahun dari setiap baris data.
- Grafik ini menunjukkan pola musiman dari penjualan setiap tahun, di mana setiap garis warna berbeda mewakili satu tahun.
- Plot ini berguna untuk melihat apakah ada pola khusus per bulan yang terjadi di setiap tahun, seperti peningkatan penjualan pada bulan tertentu.
- sns.lineplot() dengan hue='year' menambahkan warna berbeda untuk tiap tahun, membuat pola antar tahun lebih mudah dibandingkan.

3. Boxplot untuk Visualisasi Distribusi Penjualan Bulanan

plt.figure(figsize=(14, 7))

sns.boxplot(x='month', y='sales', data=df)

plt.title('Monthly Sales Distribution')

plt.xlabel('Month')

plt.ylabel('Sales')

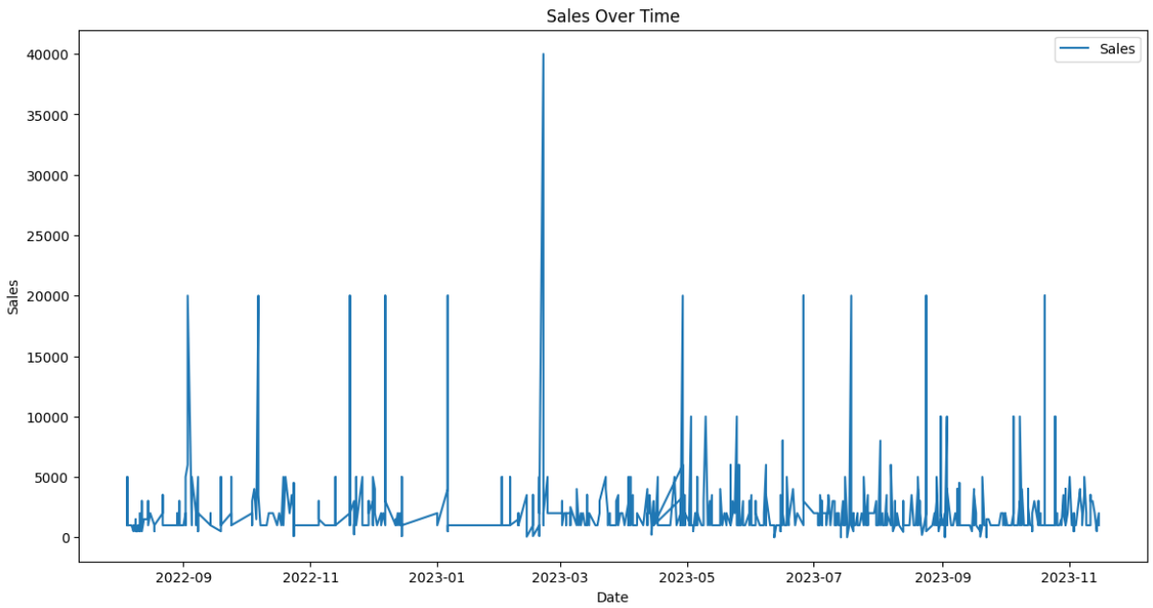
plt.show()

- Grafik boxplot ini memperlihatkan distribusi penjualan tiap bulan sepanjang dataset.
- Setiap box pada bulan tertentu menunjukkan median, rentang interkuartil (IQR), serta potensi outlier.
- Grafik ini sangat membantu untuk melihat apakah ada bulan tertentu dengan variasi penjualan yang lebih besar atau lebih kecil.

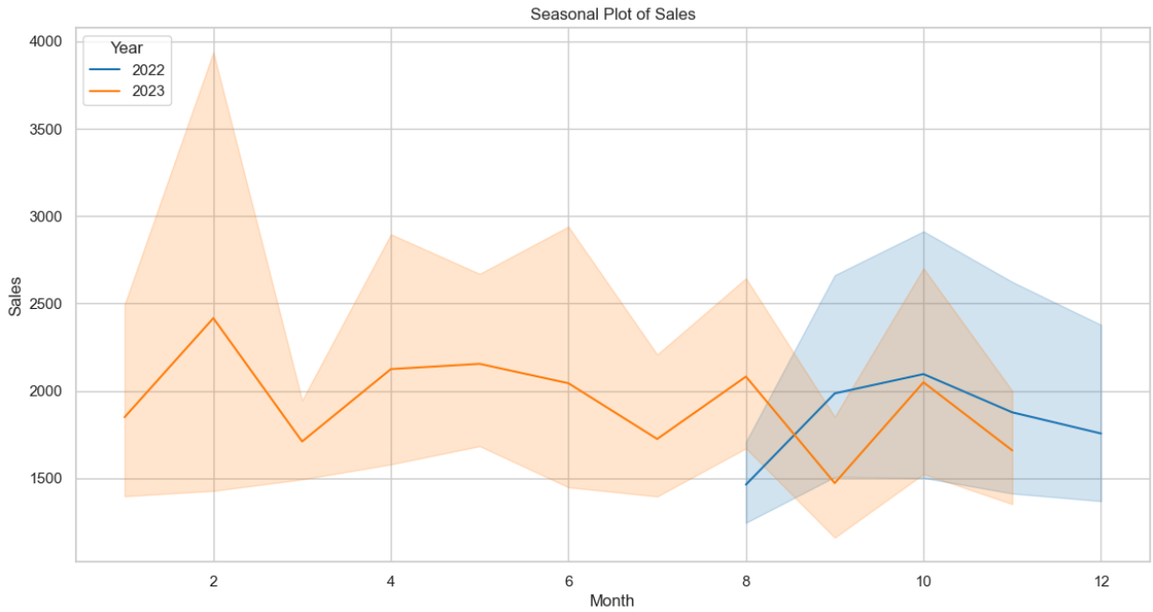
4. Menghapus Kolom Sementara

   df.drop(columns=['month', 'year'], inplace=True)

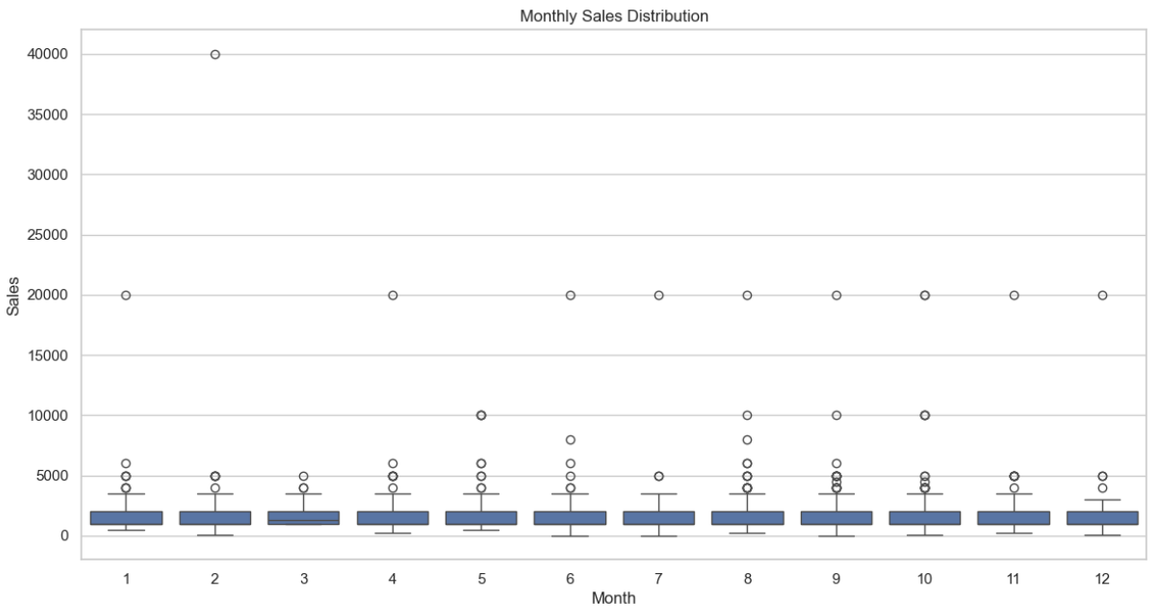
   - Kolom sementara month dan year dihapus dari df untuk membersihkan dataset, sehingga data kembali ke bentuk awalnya.



Bagian ini membuat grafik garis (line plot) yang menampilkan data penjualan secara keseluruhan dari waktu ke waktu. Grafik ini membantu melihat tren umum dari penjualan, apakah cenderung naik, turun, atau memiliki pola tertentu seiring waktu.



Pada bagian ini, kolom sementara month dan year ditambahkan ke df untuk mewakili bulan dan tahun dari setiap baris data. Grafik ini menunjukkan pola musiman dari penjualan setiap tahun, di mana setiap garis warna berbeda mewakili satu tahun. Plot ini berguna untuk melihat apakah ada pola khusus per bulan yang terjadi di setiap tahun, seperti peningkatan penjualan pada bulan tertentu.



Grafik boxplot ini memperlihatkan distribusi penjualan tiap bulan sepanjang dataset. Setiap box pada bulan tertentu menunjukkan median, rentang interkuartil (IQR), serta potensi outlier. Grafik ini sangat membantu untuk melihat apakah ada bulan tertentu dengan variasi penjualan yang lebih besar atau lebih kecil.

```python
from statsmodels.tsa.seasonal import seasonal_decompose

# Decompose the time series
result = seasonal_decompose(df['sales'], model='multiplicative', period=12)

# Plot the decomposed components
plt.figure(figsize=(14, 10))

plt.subplot(411)
plt.plot(result.observed, label='Observed')
plt.legend(loc='upper left')

plt.subplot(412)
plt.plot(result.trend, label='Trend')
plt.legend(loc='upper left')

plt.subplot(413)
plt.plot(result.seasonal, label='Seasonal')
plt.legend(loc='upper left')

plt.subplot(414)
plt.plot(result.resid, label='Residual')
plt.legend(loc='upper left')

plt.tight_layout()
plt.show()
```

✓ 2.2s Python

Kode di atas menggunakan fungsi seasonal_decompose dari modul statsmodels.tsa.seasonal untuk melakukan dekomposisi data time series, yaitu memisahkan data menjadi beberapa komponen: trend, seasonal (musiman), dan residual. Berikut adalah penjelasan setiap barisnya:

1. *from statsmodels.tsa.seasonal import seasonal_decompose*

   Ini mengimpor fungsi seasonal_decompose yang berguna untuk memisahkan data time series ke dalam komponen-komponen yang lebih mudah dianalisis.

2. *result = seasonal_decompose(df['sales'], model='multiplicative', period=12)*
   - Fungsi seasonal_decompose diterapkan pada data time series df['sales'], yang diasumsikan merupakan kolom berisi data penjualan.
   - Parameter model='multiplicative' mengindikasikan bahwa data akan didekomposisi secara multiplikatif, yang artinya komponen seasonal dan trend akan dikalikan untuk membentuk data asli.
   - period=12 berarti periode musiman adalah 12, yang biasanya menunjukkan data bulanan dengan pola tahunan.

3. Plot Decomposition

   Bagian ini akan menampilkan grafik dari komponen yang telah didekomposisi.
   - *plt.figure(figsize=(14, 10)):* Menentukan ukuran keseluruhan plot.
   - Plot Observed Data:

     plt.subplot(411)

     plt.plot(result.observed, label='Observed')

     plt.legend(loc='upper left')

     Menampilkan data asli (observed) pada subplot pertama (baris 1 dari 4).
   - Plot Trend Component:

     plt.subplot(412)

     plt.plot(result.trend, label='Trend')

     plt.legend(loc='upper left')

     Menampilkan komponen trend pada subplot kedua.

- Plot Seasonal Component:

  plt.subplot(413)

  plt.plot(result.seasonal, label='Seasonal')

  plt.legend(loc='upper left')

  Menampilkan komponen musiman (seasonal) pada subplot ketiga.

- Plot Residual Component:

  plt.subplot(414)

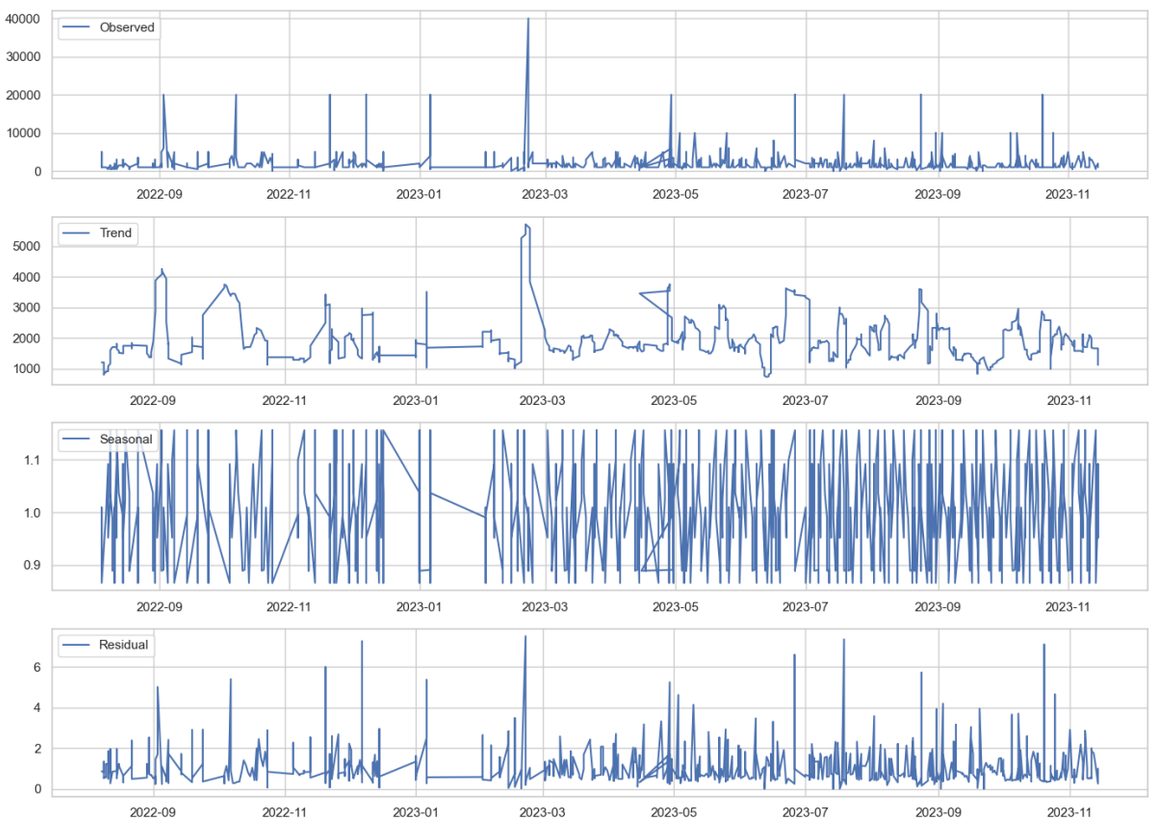
  plt.plot(result.resid, label='Residual')

  plt.legend(loc='upper left')

  Menampilkan komponen residual (sisa yang tidak dijelaskan oleh trend atau seasonal) pada subplot keempat.

4. *plt.tight_layout()* dan *plt.show()*
   - *plt.tight_layout()* memastikan subplot tidak saling tumpang tindih.
   - *plt.show()* menampilkan seluruh plot di layar.

Hasilnya adalah visualisasi dari data asli (observed) beserta komponen trend, seasonal, dan residual, yang memudahkan analisis pola dan perubahan musiman pada data penjualan dalam gambar berikut ini:



C. Pengembangan Model

Kode di atas menggunakan metode Exponential Smoothing atau Holt-Winters Exponential Smoothing dari library statsmodels untuk melakukan pemodelan time series. Pendekatan ini memprediksi nilai masa depan dengan menggabungkan komponen trend, seasonal (musiman), dan level dari data. Berikut penjelasan setiap barisnya:

1. from statsmodels.tsa.holtwinters import ExponentialSmoothing
    - Ini mengimpor kelas ExponentialSmoothing yang digunakan untuk mengaplikasikan metode eksponensial smoothing pada data time series.
    - Metode ini mampu menangani data yang memiliki trend dan komponen musiman.
2. *ets_model = ExponentialSmoothing(df['sales'], trend='add', seasonal='add', seasonal_periods=12)*
    - Baris ini membuat model ETS (Error, Trend, Seasonal) berdasarkan data penjualan (*df['sales']*).
    - Parameter:
        - trend='add': Menggunakan model trend aditif, yang artinya perubahan trend konstan dari waktu ke waktu.
        - seasonal='add': Menggunakan model musiman aditif, yaitu komponen musiman yang tetap (tidak berubah seiring waktu).
        - seasonal_periods=12: Mengatur periode musiman sebesar 12, yang cocok untuk data bulanan dengan siklus tahunan.
    - Dengan konfigurasi ini, model akan mengestimasi data penjualan dengan mempertimbangkan pola musiman dan trend aditif.
3. *ets_fit = ets_model.fit()*
    - Melakukan proses fitting (pencocokan model) pada data untuk menemukan parameter yang paling sesuai.
    - Setelah fitting, model ets_fit akan siap untuk membuat prediksi atau melakukan analisis pada data time series.
4. *print(ets_fit.summary())*
    - Menampilkan ringkasan hasil model, termasuk informasi tentang nilai parameter model, statistik error, dan komponen-komponen ETS.

- Ringkasan ini memberikan insight tentang kualitas model dan bagaimana komponen trend dan seasonal mempengaruhi hasil prediksi.

Ringkasannya, kode ini membangun model prediksi berdasarkan data penjualan yang memiliki trend dan pola musiman, dengan hasil ringkasan model yang mempermudah analisis kualitas prediksi.

```
                    ExponentialSmoothing Model Results
================================================================================
Dep. Variable:                  sales   No. Observations:                 1076
Model:          ExponentialSmoothing   SSE                 7091405920.942
Optimized:                      True   AIC                     16926.431
Trend:                      Additive   BIC                     17006.127
Seasonal:                   Additive   AICC                    16927.078
Seasonal Periods:                 12   Date:                Mon, 04 Nov 2024
Box-Cox:                       False   Time:                        21:14:51
Box-Cox Coeff.:                 None
================================================================================
                         coeff             code          optimized
--------------------------------------------------------------------------------
smoothing_level           0.0403571         alpha              True
smoothing_trend           0.0080714          beta              True
smoothing_seasonal        0.0342730         gamma              True
initial_level          1091.6667             l.0              True
initial_trend           -19.696970             b.0              True
initial_seasons.0       -327.25694             s.0              True
initial_seasons.1       -452.25694             s.1              True
initial_seasons.2        292.53472             s.2              True
initial_seasons.3        1032.1181             s.3              True
initial_seasons.4        21.701389             s.4              True
initial_seasons.5        125.86806             s.5              True
initial_seasons.6       -223.09028             s.6              True
...
initial_seasons.9       -421.00694             s.9              True
initial_seasons.10      -441.84028            s.10              True
initial_seasons.11      -322.04861            s.11              True
--------------------------------------------------------------------------------
```

Ringkasan output di atas merupakan hasil dari model Exponential Smoothing pada data time series. Berikut penjelasan dari setiap bagian:

Informasi Umum Model

- Dep. Variable: Variabel dependen yang diprediksi, yaitu sales dalam data ini.
- No. Observations: Jumlah observasi dalam dataset, yaitu 1076.
- Model: Model yang digunakan, yaitu ExponentialSmoothing.
- SSE (Sum of Squared Errors): Total kesalahan kuadrat antara nilai aktual dan prediksi, sebesar 7,091,405,920.942. Nilai ini mengukur seberapa baik model menyesuaikan data; semakin kecil, semakin baik.
- AIC (Akaike Information Criterion) dan BIC (Bayesian Information Criterion): Kriteria yang digunakan untuk menilai model, di mana nilai lebih kecil biasanya menunjukkan model yang lebih baik. Di sini, AIC sebesar 16,926.431 dan BIC 17,006.127.
- AICC: AIC yang disesuaikan untuk ukuran sampel, bernilai 16,927.078.
- Trend: Jenis trend yang digunakan, yaitu Additive.
- Seasonal: Jenis pola musiman, yaitu Additive.
- Seasonal Periods: Jumlah periode musiman, yaitu 12 (menunjukkan pola musiman tahunan pada data bulanan).
- Box-Cox: Transformasi Box-Cox tidak diterapkan (False).

Parameter Model

Berikut adalah parameter-parameter yang dipelajari atau dioptimalkan model:

- smoothing_level (α): Koefisien pelicinan level (alpha), yaitu 0.0403571. Ini menunjukkan pengaruh rata-rata jangka pendek pada nilai yang diprediksi.
- smoothing_trend (β): Koefisien pelicinan trend (beta), yaitu 0.0080714. Nilai ini menunjukkan seberapa cepat komponen trend diperbarui.
- smoothing_seasonal (γ): Koefisien pelicinan musiman (gamma), yaitu 0.0342730. Ini mengontrol pengaruh pola musiman dalam prediksi.
- initial_level: Nilai awal untuk level, yaitu 1091.6667.
- initial_trend: Nilai awal untuk trend, yaitu -19.696970. Nilai negatif menunjukkan penurunan pada awal periode data.
- initial_seasons: Nilai awal untuk setiap bulan dalam satu periode musiman. Ada 12 nilai, masing-masing untuk setiap bulan dalam pola musiman 12 bulan. Misalnya:
    - initial_seasons.0 (-327.25694) adalah penyesuaian untuk bulan pertama.
    - initial_seasons.3 (1032.1181) adalah penyesuaian untuk bulan keempat.
    - Nilai ini menunjukkan variasi musiman di masing-masing bulan dalam siklus.

Interpretasi

Model Exponential Smoothing ini menggunakan pendekatan aditif untuk komponen trend dan musiman, yang cocok untuk data dengan pola trend dan musiman yang stabil. Parameter smoothing (alpha, beta, gamma) dioptimalkan untuk menghasilkan prediksi terbaik. Setiap nilai musiman awal membantu model menangkap pola musiman yang spesifik untuk setiap bulan dalam periode tahunan.

```
# Forecast for the Next 2 Years

# Generate forecast for the next 2 years (24 months)
forecast = ets_fit.forecast(steps=24)

# Create a DataFrame to hold the forecasted values
forecast_df = pd.DataFrame(forecast.values, index=pd.date_range(start=df.index[-1] + pd.DateOffset(1), periods=24,
freq='M'), columns=['forecast'])

# Plot the forecasted values along with the historical data
plt.figure(figsize=(14, 7))
plt.plot(df.index, df['sales'], label='Historical Sales')
plt.plot(forecast_df.index, forecast, label='Forecasted Sales', color='red')
plt.title('Sales Forecast for the Next 2 Years')
plt.xlabel('Date')
plt.ylabel('Sales')
plt.legend()
plt.show()
```

32] ✓ 0.7s Python

Kode di atas digunakan untuk membuat prediksi penjualan untuk dua tahun ke depan (24 bulan) menggunakan model Exponential Smoothing yang telah di-fit sebelumnya. Berikut penjelasan rinci setiap langkahnya:

1. *forecast = ets_fit.forecast(steps=24)*
    - Menghasilkan prediksi untuk 24 langkah ke depan, atau 24 bulan (dua tahun).
    - ets_fit.forecast(steps=24) memanfaatkan model ets_fit yang telah di-fit untuk membuat prediksi penjualan pada periode mendatang.
    - Hasilnya adalah deret waktu berisi nilai prediksi untuk dua tahun ke depan.

2. *forecast_df = pd.DataFrame(forecast.values, index=pd.date_range(start=df.index[-1] + pd.DateOffset(1), periods=24, freq='M'), columns=['forecast'])*
    - Membuat DataFrame baru forecast_df untuk menyimpan hasil prediksi dalam bentuk yang lebih terstruktur.
    - forecast.values adalah nilai prediksi yang dihasilkan sebelumnya.
    - index=pd.date_range(...) membuat indeks tanggal baru untuk 24 bulan ke depan, dimulai dari bulan setelah tanggal terakhir dalam data historis df. Dengan freq='M', rentang waktu akan mengikuti format bulanan.
    - columns=['forecast'] memberi nama kolom sebagai forecast untuk menyimpan data prediksi.
3. Plotting the Historical and Forecasted Data

   Kode di bawah ini digunakan untuk memvisualisasikan hasil prediksi beserta data historis agar lebih mudah dianalisis.
    - *plt.figure(figsize=(14, 7))*: Mengatur ukuran grafik.
    - Plot Data Historis:

      plt.plot(df.index, df['sales'], label='Historical Sales')

      Menampilkan data historis sales dari DataFrame df sebagai garis waktu.
    - Plot Data Prediksi:

      plt.plot(forecast_df.index, forecast, label='Forecasted Sales', color='red')

      Menampilkan nilai prediksi forecast untuk dua tahun ke depan, dengan warna merah untuk membedakannya dari data historis.
    - Label dan Judul:

      plt.title('Sales Forecast for the Next 2 Years')

      plt.xlabel('Date')

      plt.ylabel('Sales')

      Memberikan judul grafik dan label untuk sumbu X (Date) dan Y (Sales) agar grafik lebih mudah dipahami.
    - Legend dan Show Plot:

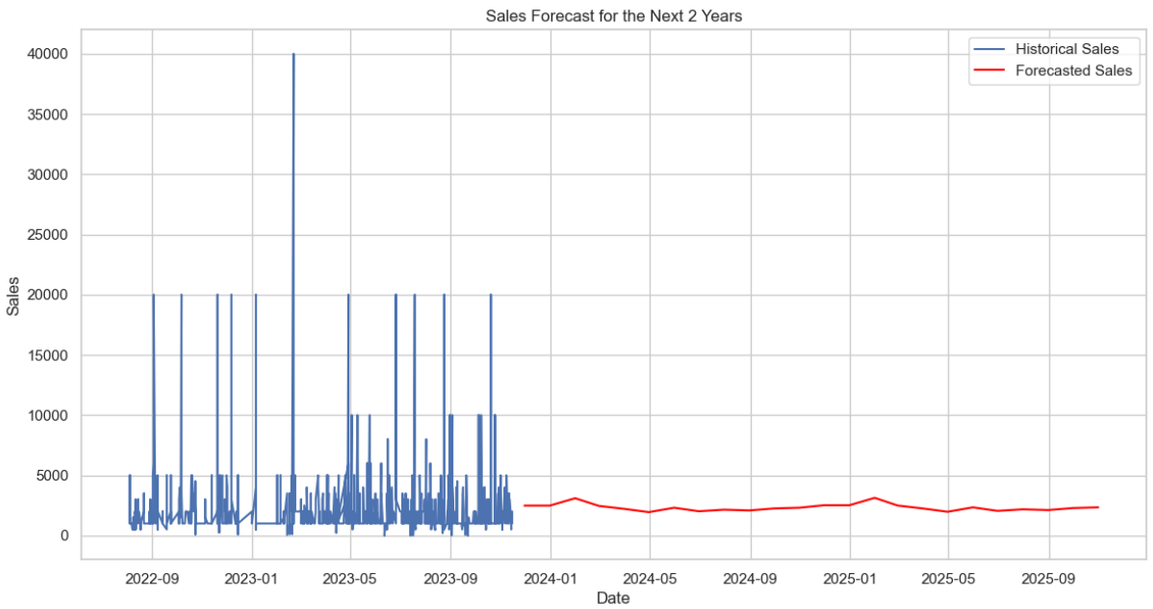
      plt.legend()

      plt.show()

      Menambahkan legenda untuk membedakan data historis dan data prediksi, serta menampilkan grafik.

Kesimpulan

Plot ini akan menunjukkan grafik dari data historis penjualan bersama prediksi untuk dua tahun ke depan. Visualisasi berikut ini membantu memahami pola prediksi dan memberikan gambaran tren masa depan berdasarkan model yang telah dilatih:



D. Evaluasi Model

Evaluasi Model

```
from sklearn.metrics import mean_absolute_error, mean_squared_error
import numpy as np

# Split the data into training and test sets
train_size = int(len(df) * 0.8)
train, test = df.iloc[:train_size], df.iloc[train_size:]

# Fit the model on the training data
ets_model = ExponentialSmoothing(train['sales'], trend='add', seasonal='add', seasonal_periods=12)
ets_fit = ets_model.fit()

# Generate forecast on the test data
test_forecast = ets_fit.forecast(steps=len(test))

# Calculate evaluation metrics
mae = mean_absolute_error(test['sales'], test_forecast)
mse = mean_squared_error(test['sales'], test_forecast)
rmse = np.sqrt(mse)

print(f'Mean Absolute Error (MAE): {mae}')
print(f'Mean Squared Error (MSE): {mse}')
print(f'Root Mean Squared Error (RMSE): {rmse}')

# Plot the forecasted values against the actual values
plt.figure(figsize=(14, 7))
plt.plot(train.index, train['sales'], label='Training Data')
plt.plot(test.index, test['sales'], label='Test Data')
plt.plot(test.index, test_forecast, label='Forecasted Sales', color='red')
plt.title('Sales Forecast vs Actual Sales')
plt.xlabel('Date')
plt.ylabel('Sales')
plt.legend()
plt.show()
```

Kode di atas melakukan pembagian data time series menjadi data training dan testing, memodelkan data training menggunakan Exponential Smoothing, memprediksi data testing, serta mengevaluasi akurasi prediksi menggunakan metrik error. Berikut penjelasan dari tiap bagian:

1. *from sklearn.metrics import mean_absolute_error, mean_squared_error*
    - Mengimpor fungsi *mean_absolute_error (MAE) dan mean_squared_error (MSE)* dari sklearn untuk mengukur kualitas prediksi model.
2. *train_size = int(len(df) * 0.8)*
    - Menentukan ukuran data training sebesar 80% dari total data. Ini akan menjadi batas untuk memisahkan data training dan testing.
3. *train, test = df.iloc[:train_size], df.iloc[train_size:]*
    - Memisahkan data menjadi:

        - train: Berisi 80% data awal sebagai data training.
        - test: Berisi 20% data sisanya sebagai data testing.
4. *ets_model = ExponentialSmoothing(train['sales'], trend='add', seasonal='add', seasonal_periods=12)*
    - Membangun model Exponential Smoothing berdasarkan data *train['sales']* dengan konfigurasi trend dan seasonal aditif serta periode musiman 12.
5. *ets_fit = ets_model.fit()*
    - Melakukan fitting model pada data training untuk mendapatkan parameter yang optimal.
6. *test_forecast = ets_fit.forecast(steps=len(test))*
    - Menggunakan model yang telah dilatih untuk membuat prediksi pada data test dengan jumlah langkah sesuai panjang data testing.

Evaluasi Model

Setelah model menghasilkan prediksi, evaluasi dilakukan dengan menghitung metrik error berikut:

- MAE (Mean Absolute Error):
  mae = mean_absolute_error(test['sales'], test_forecast)
  Mengukur rata-rata kesalahan absolut antara nilai aktual dan prediksi. Semakin kecil MAE, semakin baik model dalam memprediksi data testing.
- MSE (Mean Squared Error):
  mse = mean_squared_error(test['sales'], test_forecast)
  Mengukur rata-rata kesalahan kuadrat, memberikan penalti lebih besar pada kesalahan yang besar.
- RMSE (Root Mean Squared Error):
  rmse = np.sqrt(mse)
  Akar kuadrat dari MSE, mengembalikan nilai ke skala aslinya. RMSE juga menunjukkan seberapa besar perbedaan antara nilai prediksi dan aktual.

Visualisasi Prediksi

Plot berikut menampilkan data training, data test, dan hasil prediksi model pada data test.

- *plt.figure(figsize=(14, 7))*: Menentukan ukuran grafik.
- Plot Data Training:
  plt.plot(train.index, train['sales'], label='Training Data')
  Menampilkan data training untuk menunjukkan periode yang digunakan untuk melatih model.
- Plot Data Test:
  plt.plot(test.index, test['sales'], label='Test Data')

Menampilkan data test untuk menunjukkan perbandingan antara prediksi dan data aktual.

- Plot Data Prediksi:
  plt.plot(test.index, test_forecast, label='Forecasted Sales', color='red')
  Menampilkan prediksi yang dihasilkan model untuk data test.
- Label, Judul, dan Legend:
  plt.title('Sales Forecast vs Actual Sales')
  plt.xlabel('Date')
  plt.ylabel('Sales')
  plt.legend()
  plt.show()
  Menyediakan judul, label sumbu, dan legenda untuk memperjelas grafik.

ExponentialSmoothing Model Results

Dep. Variable:	sales	No. Observations:	1076
Model:	ExponentialSmoothing	SSE	7091405920.942
Optimized:	True	AIC	16926.431
Trend:	Additive	BIC	17006.127
Seasonal:	Additive	AICC	16927.078
Seasonal Periods:	12	Date:	Mon, 04 Nov 2024
Box-Cox:	False	Time:	21:14:51
Box-Cox Coeff.:	None		

	coeff	code	optimized
smoothing_level	0.0403571	alpha	True
smoothing_trend	0.0080714	beta	True
smoothing_seasonal	0.0342730	gamma	True
initial_level	1091.6667	l.0	True
initial_trend	-19.696970	b.0	True
initial_seasons.0	-327.25694	s.0	True
initial_seasons.1	-452.25694	s.1	True
initial_seasons.2	292.53472	s.2	True
initial_seasons.3	1032.1181	s.3	True
initial_seasons.4	21.701389	s.4	True
initial_seasons.5	125.86806	s.5	True
initial_seasons.6	-223.09028	s.6	True
...			
initial_seasons.9	-421.00694	s.9	True
initial_seasons.10	-441.84028	s.10	True
initial_seasons.11	-322.04861	s.11	True

Hasil model Exponential Smoothing yang ditampilkan memberikan informasi penting tentang model yang telah dibangun untuk memprediksi variabel penjualan. Berikut adalah penjelasan dari setiap bagian dalam hasil tersebut:

Informasi Umum

- Dep. Variable:
  - sales: Variabel dependen yang diprediksi oleh model, yaitu penjualan.

- No. Observations:
    - 1076: Jumlah total pengamatan dalam dataset yang digunakan untuk fitting model.
- Model:
    - ExponentialSmoothing: Jenis model yang digunakan, dalam hal ini adalah model smoothing eksponensial.
- SSE (Sum of Squared Errors):
    - 7091405920.942: Jumlah kesalahan kuadrat yang menunjukkan seberapa baik model fit dengan data. Semakin kecil nilai SSE, semakin baik model.
- Optimized:
    - True: Menunjukkan bahwa parameter model dioptimalkan selama proses fitting.

Kriteria Model

- AIC (Akaike Information Criterion):
    - 16926.431: Kriteria informasi yang digunakan untuk memilih model terbaik. Semakin kecil nilai AIC, semakin baik model.
- BIC (Bayesian Information Criterion):
    - 17006.127: Kriteria serupa dengan AIC tetapi memberikan penalti lebih besar untuk jumlah parameter. Nilai yang lebih kecil menunjukkan model yang lebih baik.
- AICC:
    - 16927.078: Versi AIC yang disesuaikan untuk sampel kecil.
- Seasonal Periods:
    - 12: Menunjukkan bahwa model mempertimbangkan pola musiman dengan periode 12 (biasanya bulanan).
- Date dan Time:
    - Menunjukkan waktu saat model di-fit.

Parameter Model

- coeff: Menunjukkan nilai koefisien untuk parameter model:
    - smoothing_level (alpha):
        - 0.0403571: Tingkat smoothing untuk level data. Ini adalah bobot yang diberikan untuk pengamatan terbaru dalam menentukan level saat ini.
    - smoothing_trend (beta):
        - 0.0080714: Tingkat smoothing untuk komponen tren. Ini adalah bobot untuk memperhitungkan tren dalam data.
    - smoothing_seasonal (gamma):
        - 0.0342730: Tingkat smoothing untuk komponen musiman. Ini menunjukkan seberapa banyak pengaruh musiman dalam peramalan.

- initial_level (l.0):
    - 1091.6667: Nilai awal untuk level data.
- initial_trend (b.0):
    - -19.696970: Nilai awal untuk tren, menunjukkan arah dan besaran tren awal.
- initial_seasons.0 hingga initial_seasons.11:
    - Ini adalah nilai awal untuk komponen musiman, masing-masing untuk periode musiman 0 hingga 11. Nilai-nilai ini menunjukkan seberapa besar kontribusi setiap musim terhadap prediksi:
        - Contoh:
            - initial_seasons.0: -327.25694 menunjukkan kontribusi musiman untuk bulan pertama (misalnya Januari) adalah negatif, sementara initial_seasons.3: 1032.1181 menunjukkan kontribusi musiman untuk bulan keempat (misalnya April) adalah positif dan cukup besar.

Hasil model ini memberikan gambaran yang jelas tentang bagaimana model Exponential Smoothing dibangun dan parameter-parameter kunci yang berkontribusi terhadap prediksi. Gambar berikut ini dapat membantu dalam memahami kekuatan model dan seberapa baik ia dapat *forecasting* data penjualan berdasarkan komponen yang ada: