**Time series analysis and forecasting in sales**

🧾 Dataset Overview

Time Period: Jan 2020 – Dec 2024 (60 months)

Total Records: 60 months

Columns Included:

Sales: Monthly sales figures

Advertising\_Spend: Marketing budget

Holiday: Binary indicator (1 = holiday month)

Promotion: Binary indicator (1 = promotions running)

---

📈 Trend and Seasonality (Decomposition)

The sales time series was decomposed using an additive model.

Components extracted:

Trend: Shows a steady upward trend in sales from 2020 to 2024.

Seasonality: Repeats every 12 months, with sales peaking Oct–Dec (holiday/promotion months).

Residuals: Minor variations around trend/seasonality, indicating low noise and predictable structure.

---

🧪 Stationarity Test (ADF Test)

Before Differencing:

ADF p-value > 0.05 → Non-stationary

Sales has a strong trend, failing the stationarity requirement.

After Differencing (1st order):

ADF p-value < 0.05 → Stationary

Differencing successfully removes trend, allowing ARIMA modeling.

---

📊 ACF & PACF Plots (for ARIMA Parameter Selection)

ACF shows slow decay, indicating AR(1) or higher.

PACF has a cut-off after lag 1, supporting use of ARIMA(1,1,1) or similar.

---

📉 ARIMA Model Fitting

Model: ARIMA(1,1,1)

The model was fit on the entire time series for initial forecasting.

Summary output includes:

Coefficient estimates for AR(1), MA(1)

Residual diagnostics (assumed adequate)

---

📅 Sales Forecast (Next 12 Months)

Forecasted for 12 future time periods

The forecast shows a continued upward trend in sales

Oct–Dec forecasted values are higher, consistent with seasonal spikes

---

🧪 Model Evaluation

Train/Test Split: Last 12 months used as test set

Forecasted vs Actual: Forecast closely tracks actual values

Mean Squared Error (MSE): Computed as a performance metric (value printed in output)

---

✅ Key Insights

1. Sales Growth: Steady, linear increase from 2020 to 2024

2. Seasonality: Strong year-end sales spikes (Q4) — likely due to holidays + promotions

3. Stationarity: Achieved through first-order differencing

4. Model Accuracy: ARIMA(1,1,1) provides a reasonable forecast with low MSE

5. Influencing Factors: Promotions and holidays align with observed sales surges

---

💡 Potential Enhancements

If you're looking to go further:

✅ Add exogenous variables (ARIMAX): Include Promotion and Advertising\_Spend

✅ Try SARIMA: Better handles seasonal cycles

✅ Implement cross-validation: For robust forecast validation

✅ Build a forecast dashboard (e.g., Streamlit, Power BI)

---

# Import necessary libraries

c

# Step 2: Plot the sales data

plt.figure(figsize=(10, 6))

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

plt.title('Sales Over Time')

plt.xlabel('Date')

plt.ylabel('Sales')

plt.legend()

plt.show()

# Step 3: Decompose the time series

decomposition = seasonal\_decompose(df['Sales'], model='additive', period=12)

# Plot the decomposed components

trend = decomposition.trend

seasonal = decomposition.seasonal

residual = decomposition.resid

plt.figure(figsize=(12, 8))

plt.subplot(411)

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

plt.legend(loc='best')

plt.subplot(412)

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

plt.legend(loc='best')

plt.subplot(413)

plt.plot(seasonal, label='Seasonality')

plt.legend(loc='best')

plt.subplot(414)

plt.plot(residual, label='Residuals')

plt.legend(loc='best')

plt.tight\_layout()

plt.show()

# Step 4: Check for stationarity using Augmented Dickey-Fuller test

result = adfuller(df['Sales'])

print('ADF Statistic:', result[0])

print('p-value:', result[1])

print('Critical Values:', result[4])

if result[1] > 0.05:

print("The series is not stationary")

else:

print("The series is stationary")

# Step 5: Apply differencing to achieve stationarity (if needed)

df['Sales\_diff'] = df['Sales'].diff().dropna()

# Check stationarity again

result = adfuller(df['Sales\_diff'].dropna())

print('ADF Statistic after differencing:', result[0])

print('p-value after differencing:', result[1])

print('Critical Values after differencing:', result[4])

if result[1] > 0.05:

print("The series is still not stationary")

else:

print("The series is now stationary")

# Step 6: Plot ACF and PACF to identify ARIMA parameters

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

plt.subplot(211)

plot\_acf(df['Sales\_diff'].dropna(), ax=plt.gca(), lags=40)

plt.subplot(212)

plot\_pacf(df['Sales\_diff'].dropna(), ax=plt.gca(), lags=40)

plt.tight\_layout()

plt.show()

# Step 7: Fit ARIMA model

# Replace (1, 1, 1) with appropriate (p, d, q) values based on ACF/PACF

model = ARIMA(df['Sales'], order=(1, 1, 1))

model\_fit = model.fit()

print(model\_fit.summary())

# Step 8: Forecast future sales

forecast\_steps = 12

forecast = model\_fit.forecast(steps=forecast\_steps)

# Plot the forecast

plt.figure(figsize=(10, 6))

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

plt.plot(pd.date\_range(df.index[-1], periods=forecast\_steps+1, closed='right'), forecast, label='Forecasted Sales')

plt.title('Sales Forecast')

plt.xlabel('Date')

plt.ylabel('Sales')

plt.legend()

plt.show()

# Step 9: Evaluate the model

# Split the data into train and test sets

train = df['Sales'][:-forecast\_steps]

test = df['Sales'][-forecast\_steps:]

# Fit the model on the training data

model = ARIMA(train, order=(1, 1, 1))

model\_fit = model.fit()

# Forecast on the test data

forecast = model\_fit.forecast(steps=forecast\_steps)

# Calculate Mean Squared Error (MSE)

mse = mean\_squared\_error(test, forecast)

print(f'Mean Squared Error: {mse}')

Data set

Date,Sales,Advertising\_Spend,Holiday,Promotion

2020-01-01,100,500,0,0

2020-02-01,120,600,0,0

2020-03-01,130,700,0,0

2020-04-01,110,550,0,0

2020-05-01,150,800,0,1

2020-06-01,160,850,0,1

2020-07-01,170,900,0,0

2020-08-01,180,950,0,0

2020-09-01,190,1000,0,0

2020-10-01,200,1050,1,1

2020-11-01,210,1100,1,1

2020-12-01,220,1200,1,1

2021-01-01,230,1250,0,0

2021-02-01,240,1300,0,0

2021-03-01,250,1350,0,0

2021-04-01,260,1400,0,0

2021-05-01,270,1450,0,1

2021-06-01,280,1500,0,1

2021-07-01,290,1550,0,0

2021-08-01,300,1600,0,0

2021-09-01,310,1650,0,0

2021-10-01,320,1700,1,1

2021-11-01,330,1750,1,1

2021-12-01,340,1800,1,1

2022-01-01,350,1850,0,0

2022-02-01,360,1900,0,0

2022-03-01,370,1950,0,0

2022-04-01,380,2000,0,0

2022-05-01,390,2050,0,1

2022-06-01,400,2100,0,1

2022-07-01,410,2150,0,0

2022-08-01,420,2200,0,0

2022-09-01,430,2250,0,0

2022-10-01,440,2300,1,1

2022-11-01,450,2350,1,1

2022-12-01,460,2400,1,1

2023-01-01,470,2450,0,0

2023-02-01,480,2500,0,0

2023-03-01,490,2550,0,0

2023-04-01,500,2600,0,0

2023-05-01,510,2650,0,1

2023-06-01,520,2700,0,1

2023-07-01,530,2750,0,0

2023-08-01,540,2800,0,0

2023-09-01,550,2850,0,0

2023-10-01,560,2900,1,1

2023-11-01,570,2950,1,1

2023-12-01,580,3000,1,1

2024-01-01,590,3050,0,0

2024-02-01,600,3100,0,0

2024-03-01,610,3150,0,0

2024-04-01,620,3200,0,0

2024-05-01,630,3250,0,1

2024-06-01,640,3300,0,1

2024-07-01,650,3350,0,0

2024-08-01,660,3400,0,0

2024-09-01,670,3450,0,0

2024-10-01,680,3500,1,1

2024-11-01,690,3550,1,1

2024-12-01,700,3600,1,1