

Goal 3: Simulated currency forecast for exchange rate EUR/CAD for 2025, assuming we work with only the data until the end of 2024 and did not know the actual data: Forecasting with SARIMAX, afterwards comparison with actual data for January, February, March and April 25

```
In [1]: #Import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import zipfile
import io
import requests
from statsmodels.tsa.statespace.sarimax import SARIMAX
import warnings

# Suppress warnings
warnings.filterwarnings("ignore")

# Downloading and preparing data:
# URL of the historical data from ECB
url = 'https://www.ecb.europa.eu/stats/eurofxref/eurofxref-hist.zip'

# Sending a GET request to the URL to fetch the zip file
response = requests.get(url)

# Unzipping the file in memory
with zipfile.ZipFile(io.BytesIO(response.content)) as z:

    # Opening the CSV file inside the zip file
    with z.open('eurofxref-hist.csv') as f:

        # Reading the CSV data into a pandas DataFrame
        df = pd.read_csv(f)

# Renaming the first column to 'Date' for clarity and converting the 'Date' column to datetime format
df.rename(columns={df.columns[0]: 'Date'}, inplace=True)
df['Date'] = pd.to_datetime(df['Date'], format='%Y-%m-%d')

# Setting the 'Date' column as the index of the DataFrame
df.set_index('Date', inplace=True)

# Filtering the data to keep only the EUR/CAD exchange rates and sorting the DataFrame by the date
df = df[['CAD']]
df = df.sort_index()

# Filtering data for the year 2024
df_train = df.loc['2024-01-01':'2024-12-31']

# Interpolating missing data based on the time index (linear interpolation)
df_train['CAD'] = df_train['CAD'].interpolate(method='time')

# Displaying the DataFrame for 2024
print("Exchange rate for EUR/CAD in 2024: ")
print(df_train)

# Adding a plot
# Creating a figure with specified size
plt.figure(figsize=(10, 6))

# Plotting the CAD/Euro exchange rate for the year 2024
plt.plot(df_train.index, df_train['CAD'], label='EUR/CAD 2024', color='blue')

# Adding a title to the plot
plt.title('EUR/CAD Exchange Rate in 2024')

# Adding labels to the x-axis and y-axis
plt.xlabel('Date')
plt.ylabel('EUR/CAD')

# Displaying gridlines
plt.grid(True)

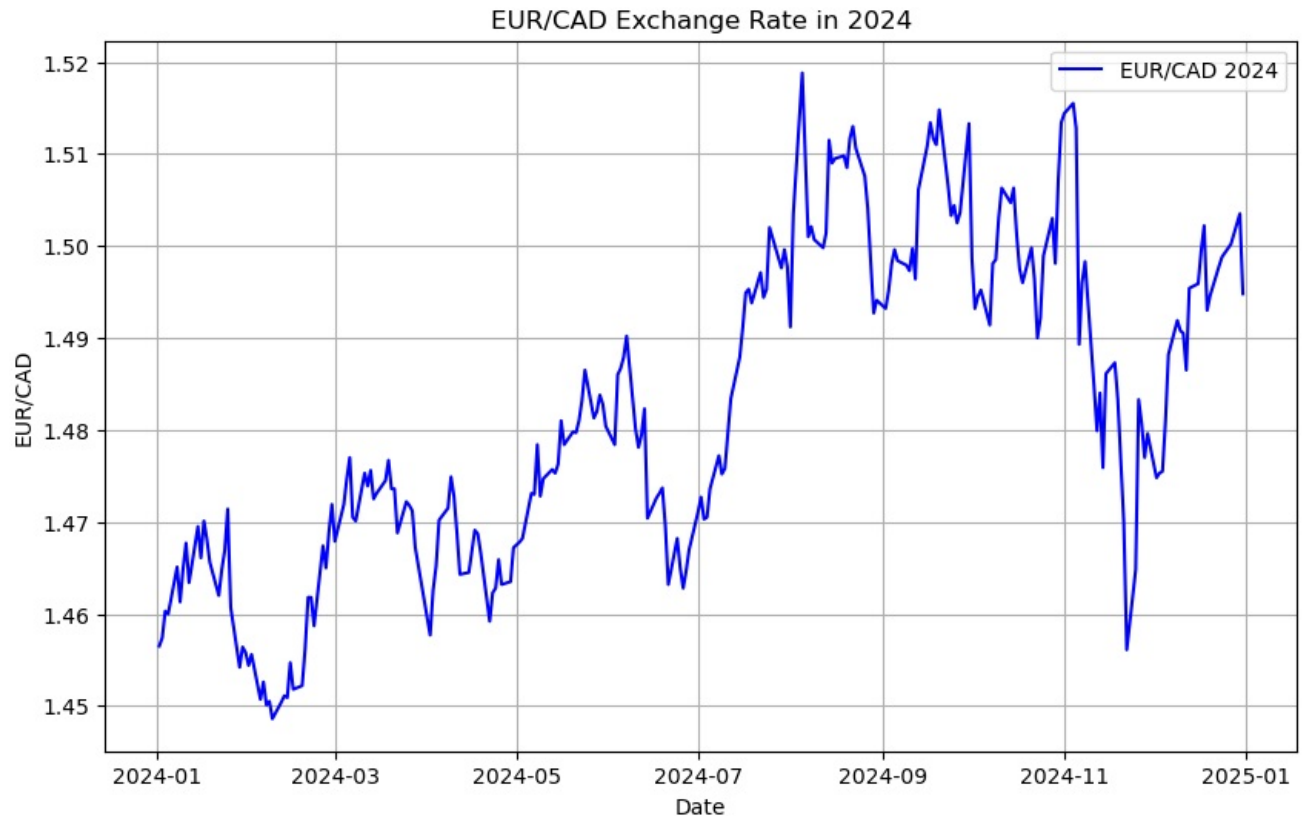
# Adding a legend to the plot
plt.legend()

# Displaying the plot
plt.show()
```

Exchange rate for EUR/CAD in 2024:

Date	CAD
2024-01-02	1.4565
2024-01-03	1.4574
2024-01-04	1.4603
2024-01-05	1.4600
2024-01-08	1.4651
...	...
2024-12-23	1.4978
2024-12-24	1.4988
2024-12-27	1.5002
2024-12-30	1.5035
2024-12-31	1.4948

[256 rows x 1 columns]



```
In [2]: # Fit SARIMAX model
# Define (p, d, q) and (P, D, Q, 52) (52 for the seasonal component, assuming weekly seasonality)
p, d, q = 1, 1, 1 # Example values for the model
P, D, Q = 1, 1, 1 # Seasonal values

# Create the model
model = SARIMAX(df_train['CAD'],
                order=(p, d, q),
                seasonal_order=(P, D, Q, 52), # Seasonal periods (52 for weekly)
                enforce_stationarity=False,
                enforce_invertibility=False)

# Fit the model
fitted = model.fit(dispatch=0)

# Forecast for 65 periods (business days)
n_periods = 65
predictions = fitted.predict(len(df_train), len(df_train) + n_periods - 1)

# Generate forecast data (future timestamps)
future_dates = pd.date_range(start=df_train.index[-1] + pd.Timedelta(days=1), periods=n_periods, freq='B')

# Forecast in tabular form
forecast_df = pd.DataFrame({
    'Date': future_dates,
    'Forecast_EUR_CAD': predictions
})
print("\nForecast for EUR/CAD (Jan-Apr 2025):")
print(forecast_df)

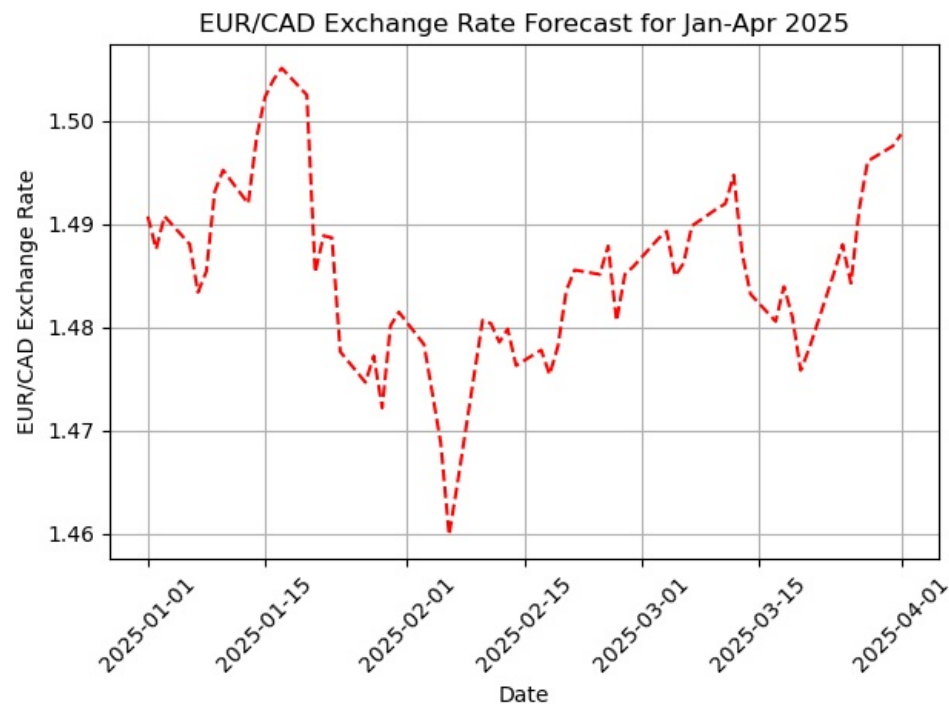
# Plot the forecasted values
plt.plot(forecast_df['Date'], forecast_df['Forecast_EUR_CAD'], label='Forecast (Jan-Apr 2025)', color='red', li
plt.xticks(rotation=45)
plt.title('EUR/CAD Exchange Rate Forecast for Jan-Apr 2025')
plt.xlabel('Date')
plt.ylabel('EUR/CAD Exchange Rate')
plt.grid(True)
```

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

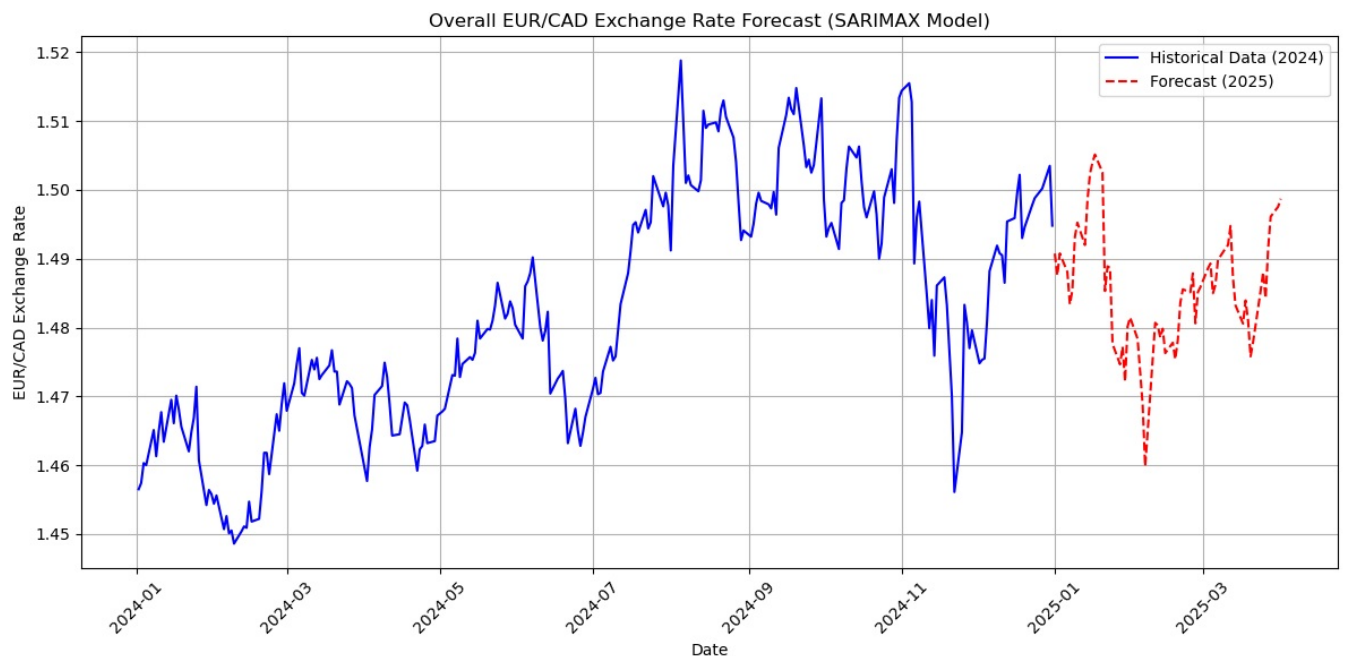
Forecast for EUR/CAD (Jan-Apr 2025):

	Date	Forecast_EUR_CAD
256	2025-01-01	1.490744
257	2025-01-02	1.487582
258	2025-01-03	1.490770
259	2025-01-06	1.488066
260	2025-01-07	1.483382
...
316	2025-03-26	1.484267
317	2025-03-27	1.491499
318	2025-03-28	1.496106
319	2025-03-31	1.497574
320	2025-04-01	1.498716

[65 rows x 2 columns]



```
In [3]: # Plot of historical data and forecast
plt.figure(figsize=(12, 6))
plt.plot(df_train.index, df_train['CAD'], label='Historical Data (2024)', color='blue')
plt.plot(future_dates, predictions, label='Forecast (2025)', color='red', linestyle='--')
plt.title('Overall EUR/CAD Exchange Rate Forecast (SARIMAX Model)')
plt.xlabel('Date')
plt.ylabel('EUR/CAD Exchange Rate')
plt.legend()
plt.xticks(rotation=45)
plt.grid(True)
plt.tight_layout()
plt.show()
```



```
In [4]: # Comparison: Getting actual data for Jan-Apr 2025
# Filter data for January to April 2025
df_2025_filtered = df.loc['2025-01-01':'2025-04-01']
df_2025_filtered['CAD'] = df_2025_filtered['CAD'].interpolate(method='time')

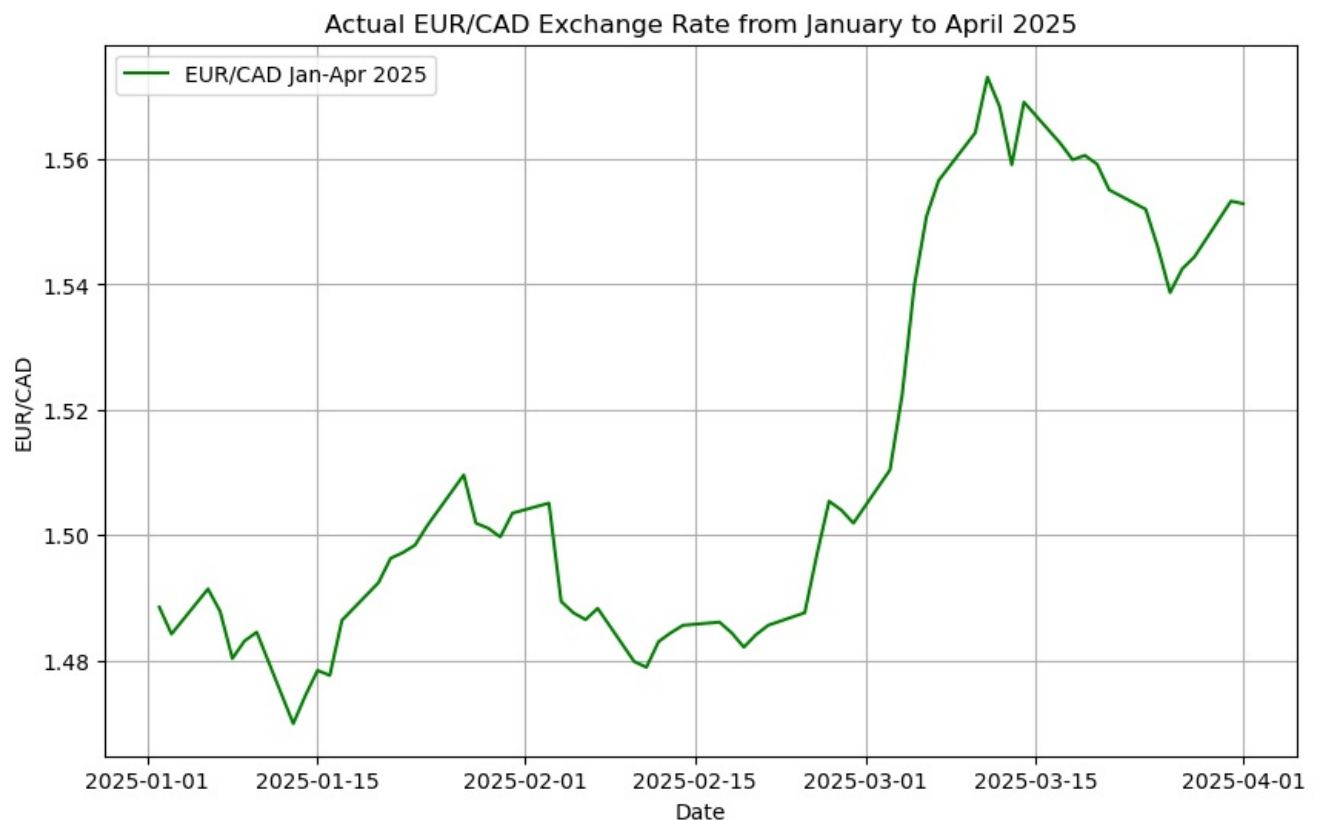
# Display the filtered data
print("\nEUR/CAD Exchange Rates from 1st Jan 2025 to 30th Apr 2025:")
print(df_2025_filtered)

# Add and adjust plot
plt.figure(figsize=(10, 6))
plt.plot(df_2025_filtered.index, df_2025_filtered['CAD'], label='EUR/CAD Jan-Apr 2025', color='green')
plt.title('Actual EUR/CAD Exchange Rate from January to April 2025')
plt.xlabel('Date')
plt.ylabel('EUR/CAD')
plt.grid(True)
plt.legend()
plt.show()
```

EUR/CAD Exchange Rates from 1st Jan 2025 to 30th Apr 2025:
CAD

```
Date
2025-01-02  1.4885
2025-01-03  1.4842
2025-01-06  1.4914
2025-01-07  1.4878
2025-01-08  1.4803
...
2025-03-26  1.5387
2025-03-27  1.5425
2025-03-28  1.5444
2025-03-31  1.5533
2025-04-01  1.5529
```

[64 rows x 1 columns]



```
In [5]: # Merge forecast with actuals
comparison_df = pd.merge(
    df_2025_filtered[['CAD']],
    forecast_df.set_index('Date'),
    left_index=True,
    right_index=True,
    how='inner'
)

# Calculate the absolute and relative differences
comparison_df['Abs_Diff'] = comparison_df['CAD'] - comparison_df['Forecast_EUR_CAD']
comparison_df['Rel_Diff (%)'] = 100 * comparison_df['Abs_Diff'] / comparison_df['CAD']

# Print the comparison table
print("\nComparison of Actual vs Forecasted EUR/CAD Exchange Rate (Jan-Apr 2025):")
print(comparison_df[['CAD', 'Forecast_EUR_CAD', 'Abs_Diff', 'Rel_Diff (%)']].round(4))

# Plot of actual vs forecast
```

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

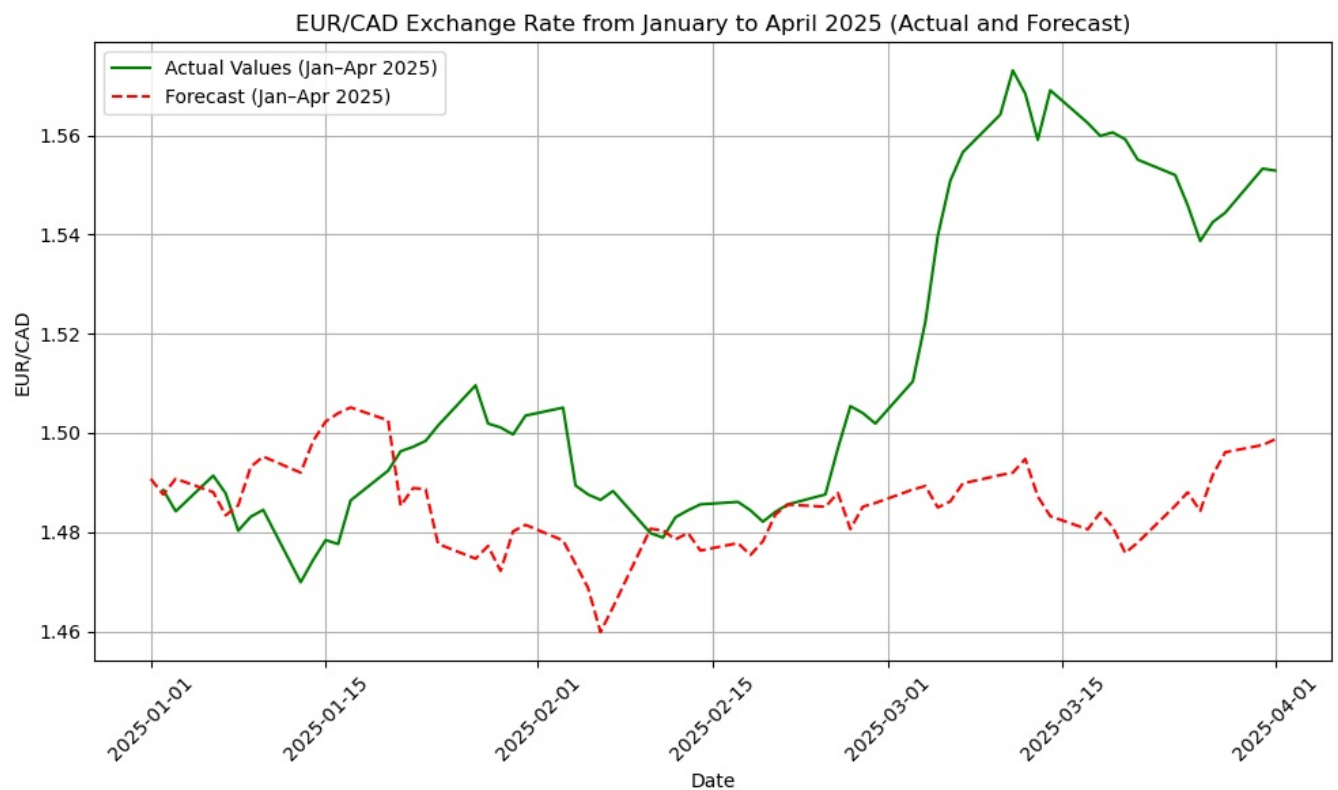
plt.plot(df_2025_filtered.index, df_2025_filtered['CAD'], label='Actual Values (Jan-Apr 2025)', color='green')
plt.plot(forecast_df['Date'], forecast_df['Forecast_EUR_CAD'], label='Forecast (Jan-Apr 2025)', color='red', li

plt.title('EUR/CAD Exchange Rate from January to April 2025 (Actual and Forecast)')
plt.xlabel('Date')
plt.ylabel('EUR/CAD')
plt.grid(True)
plt.legend()
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

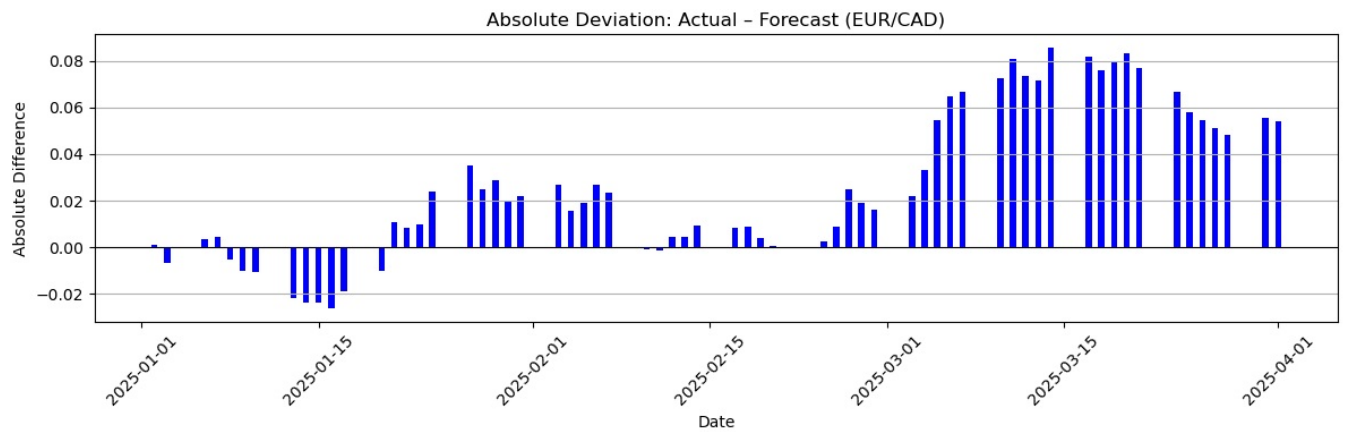
Comparison of Actual vs Forecasted EUR/CAD Exchange Rate (Jan-Apr 2025):

	CAD	Forecast_EUR_CAD	Abs_Diff	Rel_Diff (%)
Date				
2025-01-02	1.4885	1.4876	0.0009	0.0617
2025-01-03	1.4842	1.4908	-0.0066	-0.4426
2025-01-06	1.4914	1.4881	0.0033	0.2235
2025-01-07	1.4878	1.4834	0.0044	0.2969
2025-01-08	1.4803	1.4854	-0.0051	-0.3453
...
2025-03-26	1.5387	1.4843	0.0544	3.5376
2025-03-27	1.5425	1.4915	0.0510	3.3064
2025-03-28	1.5444	1.4961	0.0483	3.1270
2025-03-31	1.5533	1.4976	0.0557	3.5876
2025-04-01	1.5529	1.4987	0.0542	3.4892

[64 rows x 4 columns]



```
In [6]: # Plotting deviation as bar chart
plt.figure(figsize=(12, 4))
plt.bar(comparison_df.index, comparison_df['Abs_Diff'], width=0.5, color='blue')
plt.axhline(0, color='black', linewidth=0.8)
plt.title('Absolute Deviation: Actual - Forecast (EUR/CAD)')
plt.xlabel('Date')
plt.ylabel('Absolute Difference')
plt.grid(True, axis='y')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



```
In [7]: # Model summary
print("These are the SARIMAX results:")
print(fitted.summary())
```

These are the SARIMAX results:

```

=====
                        SARIMAX Results
=====
Dep. Variable:          CAD      No. Observations:          256
Model:                SARIMAX(1, 1, 1)x(1, 1, 1, 52)    Log Likelihood        -3050.916
Date:                  Sat, 10 May 2025                AIC             6111.832
Time:                  09:06:03                        BIC             6126.852
Sample:                0                               HQIC            6117.935
Covariance Type:      opg
=====

```

	coef	std err	z	P> z	[0.025	0.975]
ar.L1	0.3956	-0	-inf	0.000	0.396	0.396
ma.L1	-0.5359	-0	inf	0.000	-0.536	-0.536
ar.S.L52	-0.3060	3.42e-34	-8.94e+32	0.000	-0.306	-0.306
ma.S.L52	-4.063e+12	2.91e-30	-1.4e+42	0.000	-4.06e+12	-4.06e+12
sigma2	5.88e-09	9.84e-10	5.977	0.000	3.95e-09	7.81e-09

```

=====
Ljung-Box (L1) (Q):          0.38    Jarque-Bera (JB):          8.56
Prob(Q):                    0.54    Prob(JB):              0.01
Heteroskedasticity (H):      1.78    Skew:                  -0.46
Prob(H) (two-sided):         0.04    Kurtosis:              3.73
=====

```

Warnings:

[1] Covariance matrix calculated using the outer product of gradients (complex-step).
 [2] Covariance matrix is singular or near-singular, with condition number inf. Standard errors may be unstable.

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js