SARIMA Model Documentation for Infosys Quarterly Net Profit Forecasting

---

1. Overview

The SARIMA model is a time-series forecasting method that extends the ARIMA model by incorporating seasonality. It is used here to predict the net profit (`Net profit/(loss) for the period`, in Rs. Cr.) of Infosys, leveraging both the target variable and exogenous variables such as `Total Revenue` and `Total Expenditure`. The model is implemented using Python libraries, primarily `statsmodels` for SARIMA and `pandas` for data manipulation.

---

2. Dataset Description

The dataset contains quarterly financial data for Infosys from March 2005 to December 2024. Key columns used in the model include:

- **Quarterly Results of Infosys (in Rs. Cr.)**: Renamed to `ds` (date column, e.g., "Mar 05 Q4").
- **Net profit/(loss) for the period**: Renamed to `y` (target variable, net profit in Rs. Cr.).
- **Total Revenue**: Renamed to `revenue` (total revenue in Rs. Cr.).
- **Total Expenditure**: Renamed to `total_expenditure` (total expenditure in Rs. Cr.).

Additional features are engineered to enhance model performance, including lagged values, rolling statistics, and interactions between revenue and expenditure.

---

3. Model Implementation

The implementation involves data preprocessing, feature engineering, model training, evaluation, and forecasting. Below is a step-by-step breakdown of the process, including the functions and parameters used.

3.1. Libraries and Functions Used

The following Python libraries and their key functions are used:

- **pandas** (`pd`):
    - `pd.read_excel()`: Loads the Excel dataset.
    - `pd.to_datetime()`: Converts string dates to datetime objects.
    - `DataFrame.rename()`: Renames columns for consistency.
    - `DataFrame.shift()`: Creates lagged features.
    - `DataFrame.rolling()`: Computes rolling mean and standard deviation.
    - `DataFrame.ffill()`/`bfill()`: Handles missing values.
    - `DataFrame.clip()`: Caps outliers.
- **numpy** (`np`):
    - `np.arange()`: Creates a trend feature.
    - `np.percentile()`: Calculates outlier bounds.
    - `np.sqrt()`: Computes RMSE.
- **sklearn.metrics**:
    - `mean_absolute_error()`: Calculates MAE.
    - `mean_squared_error()`: Calculates MSE.
- **statsmodels.tsa.statespace.sarimax**:
    - `SARIMAX()`: Fits the SARIMA model with exogenous variables.
    - `SARIMAX.fit()`: Trains the model.
    - `SARIMAX.predict()`: Generates in-sample predictions.
    - `SARIMAX.forecast()`: Forecasts future values.
- **statsmodels.tsa.holtwinters**:
    - `ExponentialSmoothing()`: Forecasts exogenous variables (`revenue` and `total_expenditure`).
    - `ExponentialSmoothing.fit()`: Fits the exponential smoothing model.
    - `ExponentialSmoothing.forecast()`: Predicts future exogenous values.
- **matplotlib.pyplot** (`plt`):
    - `plt.plot()`: Plots time series and forecasts.

3.2. Data Preprocessing

3.2.1. Date Parsing

The `ds` column (e.g., "Mar 05 Q4") is parsed into datetime objects using a custom function `parse_quarter`:

```
def parse_quarter(date_str):
    month_map = {"Mar": ("03", "31"), "Jun": ("06", "30"), "Sep": ("09", "30"), "Dec": ("12", "31")}
    parts = date_str.split()
    year = "20" + parts[1] if int(parts[1]) < 25 else "19" + parts[1]
    month, day = month_map[parts[0]]
    return pd.to_datetime(f"{year}-{month}-{day}")
```

This function maps quarters to the last day of the corresponding month (e.g., "Mar 05" → 2005-03-31).

3.2.2. Feature Engineering

The following features are created:

- **Trend**: `trend = np.arange(len(df))` (sequential index for trend modeling).
- **Lagged Values**: `lag_1`, `lag_2`, `lag_3` (net profit shifted by 1, 2, and 3 quarters).
- **Rolling Statistics**:
    - `rolling_mean`: 4-quarter rolling mean of `y`.
    - `rolling_std`: 4-quarter rolling standard deviation of `y`.
- **Interaction Term**: `revenue_expenditure = revenue * total_expenditure`.

3.2.3. Handling Missing Values

Missing values are filled using forward-fill (`ffill()`) and backward-fill (`bfill()`):

```
df = df.ffill().bfill()
```

3.2.4. Outlier Detection and Capping

Outliers in `y` are detected using the interquartile range (IQR) method:

- Q1, Q3: 25th and 75th percentiles.
- IQR: Q3 - Q1.
- Bounds: `[Q1 - 1.5 * IQR, Q3 + 1.5 * IQR]`.
- Values outside these bounds are clipped to the bounds.

```
q1, q3 = np.percentile(df["y"], [25, 75])
iqr = q3 - q1
lower_bound, upper_bound = q1 - 1.5 * iqr, q3 + 1.5 * iqr
df["y"] = df["y"].clip(lower_bound, upper_bound)
```

3.2.5. Train-Test Split

The dataset is split into training (first `n-10` quarters) and test (last 10 quarters) sets:

```
train_size = len(df) - 10
train_df = df.iloc[:train_size].copy()
test_df = df.iloc[train_size:].copy()
```

3.3. SARIMA Model Configuration

The SARIMA model is configured with the following parameters:

- **Order**: `(p, d, q) = (1, 1, 1)`
  - `p = 1`: One autoregressive term.
  - `d = 1`: First differencing to make the series stationary.
  - `q = 1`: One moving average term.
- **Seasonal Order**: `(P, D, Q, s) = (1, 1, 1, 4)`
  - `P = 1`: One seasonal autoregressive term.
  - `D = 1`: Seasonal differencing.
  - `Q = 1`: One seasonal moving average term.
  - `s = 4`: Quarterly seasonality (4 quarters per year).
- **Exogenous Variables**: `revenue`, `total_expenditure`, `revenue_expenditure`.
- **Model Settings**:
  - `enforce_stationarity=False`: Allows non-stationary components.
  - `enforce_invertibility=False`: Allows non-invertible moving average components.

The model is trained using:

```
sarima_model = SARIMAX(
 Train_df["y"],
exog=exog_train,
order=(1, 1, 1),
seasonal_order=(1, 1, 1, 4),
enforce_stationarity=False,
enforce_invertibility=False
).fit(disp=False)
```

3.4. Predictions and Evaluation

3.4.1. In-Sample and Out-of-Sample Predictions

- **Training Predictions**: Generated for the training period using `sarima_model.predict()`.

```
train_pred = sarima_model.predict(start=0, end=len(train_df)-1, exog=exog_train)
```

- **Test Predictions**: Generated for the test period.

```
test_pred = sarima_model.predict(start=len(train_df), end=len(df)-1, exog=exog_test)
```

3.4.2. Evaluation Metrics

The model's performance on the test set is evaluated using:

- **Mean Absolute Error (MAE)**: Average absolute difference between predicted and actual values.
- **Root Mean Squared Error (RMSE)**: Square root of the average squared differences.
- **Mean Absolute Percentage Error (MAPE)**: Average percentage error.

```
mae = mean_absolute_error(test_df["y"], test_df["sarima_pred"])
mse = mean_squared_error(test_df["y"], test_df["sarima_pred"])
rmse = np.sqrt(mse)
mape = np.mean(np.abs((test_df["y"] - test_df["sarima_pred"]) / test_df["y"])) * 100
```

3.5. Forecasting Future Values

To forecast the next 8 quarters, the following steps are taken:

3.5.1. Future Dates

A DataFrame `future` is created with dates for the next 8 quarters starting from the last date in the dataset (2024-12-31):

```
future = pd.DataFrame()
future["ds"] = pd.date_range(start=df["ds"].iloc[-1], periods=9, freq="QE")[1:]  # Starts
2025-03-31
```

This creates a sequence of dates: 2025-03-31, 2025-06-30, 2025-09-30, 2025-12-31, 2026-03-31, 2026-06-30, 2026-09-30, and 2026-12-31. The `pd.date_range` function with `freq="QE"` ensures quarterly frequency, aligning with the dataset's structure.

3.5.2. Forecasting Exogenous Variables

The exogenous variables (`revenue`, `total_expenditure`) are critical inputs to the SARIMA model, as they influence the net profit predictions. Since these variables are not available for future quarters, they are forecasted using the Holt-Winters Exponential Smoothing method, which is well-suited for time series with trends and seasonality. The model is configured with:

- **Additive Trend**: Captures the linear growth in revenue and expenditure over time.
- **Additive Seasonality**: Accounts for quarterly seasonal patterns (e.g., higher revenue in certain quarters).
- **Seasonal Periods**: Set to 4, reflecting the quarterly cycle.

The implementation is as follows:

```
revenue_model = ExponentialSmoothing(df["revenue"], trend="add", seasonal="add", seasonal_periods=4).fit()
expenditure_model = ExponentialSmoothing(df["total_expenditure"], trend="add", seasonal="add", seasonal_periods=4).fit()
future["revenue"] = revenue_model.forecast(8).values
future["total_expenditure"] = expenditure_model.forecast(8).values
future["revenue_expenditure"] = future["revenue"] * future["total_expenditure"]
```

Detailed Explanation:

- **Why Exponential Smoothing?**: This method smooths historical data to predict future values, giving more weight to recent observations. It is effective for financial data like revenue and expenditure, which exhibit both long-term growth (trend) and recurring quarterly patterns (seasonality).
- **Model Fitting**: The `ExponentialSmoothing.fit()` function optimizes parameters (smoothing levels for level, trend, and seasonality) to minimize prediction errors based on historical data from 2005 to 2024.
- **Forecasting**: The `forecast(8)` method generates predictions for the next 8 quarters. For example, if historical revenue shows a steady increase with peaks in Q4 (December), the model will project this pattern forward.
- **Interaction Term**: The `revenue_expenditure` variable is calculated as the product of forecasted `revenue` and `total_expenditure`, capturing their combined effect on net profit (e.g., high revenue with controlled expenditure may lead to higher profits).
- **Practical Example**: For Q1 2025, the model might predict `revenue` as 36,500 Rs. Cr. based on a historical upward trend and Q1 seasonal dip, and `total_expenditure` as

30,000 Rs. Cr., reflecting cost patterns. The interaction term is then 36,500 * 30,000 = 1,095,000,000 Rs. Cr.

This step ensures that the SARIMA model has reliable inputs for future predictions, maintaining consistency with historical trends and seasonal behaviors.

3.5.3. SARIMA Forecasting

The SARIMA model forecasts net profit for the next 8 quarters using the forecasted exogenous variables (`revenue`, `total_expenditure`, `revenue_expenditure`). The implementation is:

```
exog_future = future[["revenue", "total_expenditure", "revenue_expenditure"]]
sarima_forecast = sarima_model.forecast(steps=8, exog=exog_future)
future["yhat"] = sarima_forecast.values
```

Detailed Explanation:

- **Role of SARIMA**: The SARIMA model combines historical net profit data with exogenous variables to predict future values. It accounts for:
    1. **Autoregressive Terms**: How past net profits influence future values (e.g., a high profit in Q4 2024 may suggest strong performance in Q1 2025).
    2. **Moving Average Terms**: The impact of past forecast errors on current predictions.
    3. **Seasonal Patterns**: Quarterly cycles (e.g., profits may peak in Q4 due to year-end activities).
    4. **Exogenous Variables**: The influence of revenue, expenditure, and their interaction on profits.
- **Forecasting Process**: The `forecast(steps=8)` method generates predictions for 8 quarters by:
    1. Using the trained SARIMA model parameters (fitted on training data up to Q1 2023).
    2. Incorporating the forecasted exogenous variables from 3.5.2.
    3. Iteratively predicting each quarter, using previous predictions as inputs for subsequent quarters.
- **Exogenous Variables Integration**: The model adjusts predictions based on `revenue`, `total_expenditure`, and `revenue_expenditure`. For instance, higher forecasted revenue in Q4 2025 may increase predicted profits, while rising expenditure may reduce them.
- **Practical Example**: For Q1 2025, the model uses the last observed net profit (6358 Rs. Cr. from Q3 2024), historical seasonal patterns (e.g., Q1 profits relative to Q4), and exogenous inputs (e.g., revenue = 36,500 Rs. Cr.). It might predict a net profit of 7017.5 Rs. Cr., reflecting a balance of seasonal trends and exogenous impacts.
- **Why 8 Quarters?**: Forecasting 8 quarters (2 years) provides a medium-term outlook, balancing reliability (closer quarters are more accurate) with strategic planning needs.

This step produces the final net profit forecasts, which are stored in the `yhat` column of the `future` DataFrame.

3.6. Visualization

A time series plot is generated, showing:

- Training data (`y` for training period).
- Test data (`y` for test period).
- Test predictions (`sarima_pred`).
- Future forecasts (`yhat`).

---

4. Example: Forecasting the Next 8 Quarters

4.1. Input Data

The last observed data point is for Q3 2024 (2024-12-31):

- `y` (net profit): 6358 Rs. Cr.
- `revenue`: 35916 Rs. Cr.
- `total_expenditure`: 29558 Rs. Cr.
- `revenue_expenditure`: 35916 * 29558 = 1,061,515,428 Rs. Cr.

4.2. Forecasting Exogenous Variables

Using `ExponentialSmoothing`, the model predicts `revenue` and `total_expenditure` for Q1 2025 to Q4 2026. For Q1 2025 (2025-03-31), assume:

- `revenue`: 36,500 Rs. Cr. (based on historical growth and Q1 seasonal patterns).
- `total_expenditure`: 30,000 Rs. Cr. (reflecting cost trends and seasonality).
- `revenue_expenditure`: 36,500 * 30,000 = 1,095,000,000 Rs. Cr.

These values are generated by the Holt-Winters model, which analyzes historical data to project trends and seasonal fluctuations.

4.3. SARIMA Forecast

The SARIMA model uses the trained parameters and forecasted exogenous variables to predict net profit for each quarter. The process involves:

- Using historical net profit data (e.g., 6358 Rs. Cr. in Q3 2024) and recent trends.
- Incorporating the seasonal pattern (quarterly, s=4, e.g., Q1 may have different profit levels than Q4).
- Adjusting predictions based on `revenue`, `total_expenditure`, and their interaction (e.g., high revenue with controlled expenditure boosts profits).

For Q1 2025, the model combines these factors to predict a net profit, such as 7017.5 Rs. Cr., and continues this process for subsequent quarters.

4.4. Forecast Output

The forecasted values for the next 8 quarters (example output):

Date	Forecasted Net Profit (Rs. Cr.)
2025-03-31	7017.50
2025-06-30	7150.20
2025-09-30	7300.80
2025-12-31	7450.10
2026-03-31	7600.90
2026-06-30	7750.30
2026-09-30	7900.70
2026-12-31	8050.40
