SARIMA Forecasting Model Documentation

Overview

This Python script implements a SARIMA model with exogenous variables to forecast quarterly financial metrics, specifically net profit, using historical data that includes total revenue and total expenditure. The code is structured as a modular pipeline with two primary functions: `load_data` for data preprocessing and `run_sarima_model` for modeling, evaluation, and forecasting. It is designed to handle quarterly financial data from an Excel file, perform robust preprocessing, and return structured outputs for further analysis.

Purpose

- **Objective**: Forecast net profit for the next 8 quarters using a SARIMA model, incorporating exogenous variables (revenue, expenditure, and their interaction).
- **Use Case**: Financial analysis for companies, such as predicting future profitability based on historical quarterly results.
- **Key Features**:
  - Loads and validates Excel data with date, net profit, revenue, and expenditure columns.
  - Preprocesses data by parsing dates, handling missing values, and capping outliers.
  - Trains a SARIMA model with seasonal components and exogenous variables.
  - Evaluates model performance using MAE, RMSE, and MAPE.
  - Forecasts future values with 80% confidence intervals.
  - Returns structured results (training data, test data, forecasts, and metrics).

Code Structure

The code is organized into two main functions, with dependencies on standard Python libraries for data manipulation, statistical modeling, and forecasting.

Dependencies

- **pandas**: For data manipulation and DataFrame operations.
- **numpy**: For numerical computations (e.g., outlier capping).
- **sklearn.metrics**: For calculating mean absolute error (MAE) and mean squared error (MSE).
- **statsmodels.tsa.statespace.sarimax**: For SARIMA modeling.
- **statsmodels.tsa.holtwinters**: For Exponential Smoothing to forecast exogenous variables.

Functions

1. `load_data(df)`:
    - Purpose: Loads and preprocesses the input Excel data.
    - Input: A pandas DataFrame (`df`) from an Excel file.
    - Output: A tuple containing the processed DataFrame and an error message (if any).
2. `run_sarima_model(df)`:
    - Purpose: Trains the SARIMA model, evaluates performance, and forecasts future values.
    - Input: A preprocessed DataFrame from `load_data`.
    - Output: A dictionary with training data, test data, forecasts, and evaluation metrics.

Detailed Functionality

1. `load_data(df)`

This function validates and preprocesses the input data to ensure it meets the requirements for SARIMA modeling.

Input

- **df**: A pandas DataFrame loaded from an Excel file, expected to have:

- A date column (first column, e.g., "Mar 05 Q4").
- Columns named "Net profit/(loss) for the period", "Total Revenue", and "Total Expenditure".

Processing Steps

1. **Validation**:
   - Checks if the DataFrame is non-empty and has at least 4 columns.
   - Verifies the presence of required columns: "Net profit/(loss) for the period", "Total Revenue", "Total Expenditure".
   - Returns `None` and an error message if validation fails.
2. **Column Renaming**:
   - Renames the first column to `ds` (date).
   - Renames required columns to `y` (net profit), `revenue`, and `total_expenditure`.
3. **Date Parsing**:
   - Defines a nested function `parse_quarter` to convert date strings (e.g., "Mar 05") to datetime objects.
   - Supports quarters: "mar" (Q1), "jun" (Q2), "sep" (Q3), "dec" (Q4).
   - Handles year suffixes (e.g., "05" → 2005 if ≤ 50, else 1905).
   - Assigns end-of-quarter dates (e.g., "Mar 05" → 2005-03-31).
   - Drops rows with invalid dates and sorts by date.
4. **Output**:
   - Returns a tuple: `(processed_df, error_message)`.
     - `processed_df`: DataFrame with columns `ds`, `y`, `revenue`, `total_expenditure` (or `None` if errors occur).
     - `error_message`: `None` if successful, else a descriptive error string.

Error Handling

- Empty DataFrame or insufficient columns: Returns error message.
- Missing required columns: Lists missing columns in the error message.
- Invalid dates: Returns sample invalid values for debugging.

2. `run_sarima_model(df)`

This function performs feature engineering, model training, evaluation, and forecasting.

Input

- **df**: A preprocessed DataFrame from `load_data` with columns `ds`, `y`, `revenue`, `total_expenditure`.

Processing Steps

1. **Feature Engineering**:
   - Creates an interaction term: `revenue_expenditure = revenue * total_expenditure`.
   - This is used as an exogenous variable in the SARIMA model.
2. **Missing Value Handling**:
   - Applies forward fill (`ffill`) followed by backward fill (`bfill`) to impute missing values in all columns.
3. **Outlier Capping**:
   - Computes the interquartile range (IQR) for `y` (net profit).
   - Caps `y` values outside `[Q1 - 1.5*IQR, Q3 + 1.5*IQR]` to mitigate the impact of extreme values.
4. **Train-Test Split**:
   - Splits the data, reserving the last 10 quarters for testing (`test_df`) and the rest for training (`train_df`).
   - Stores training and test dates (`dates_train`, `dates_test`) for potential use (though not used in this version).
5. **Exogenous Variables**:
   - Defines exogenous variables: `["revenue", "total_expenditure", "revenue_expenditure"]`.
   - Extracts these columns for training (`exog_train`) and testing (`exog_test`).
6. **SARIMA Model**:
   - Configures a SARIMA model with:
     - Order: `(1, 1, 1)` (ARIMA: 1 AR, 1 differencing, 1 MA).

    - Seasonal order: `(1, 1, 1, 4)` (seasonal AR, differencing, MA with quarterly periodicity).
    - Exogenous variables: `exog_train`.
    - Settings: `enforce_stationarity=False`, `enforce_invertibility=False` for flexibility.
  - Fits the model with `disp=False` to suppress convergence messages.
7. **Predictions**:
  - Generates predictions for the training set (`train_pred`) and test set (`test_pred`).
  - Stores predictions in `train_df["sarima_pred"]` and `test_df["sarima_pred"]`.
8. **Evaluation**:
  - Computes metrics on the test set:
    - **MAE**: Mean Absolute Error.
    - **RMSE**: Root Mean Squared Error.
    - **MAPE**: Mean Absolute Percentage Error (as a percentage).
  - Stores metrics in a dictionary: `{"mae": mae, "rmse": rmse, "mape": mape}`.
9. **Forecasting**:
  - Creates a `future` DataFrame with 8 quarterly dates starting after the last historical date (`df["ds"].iloc[-1]`).
  - Forecasts exogenous variables using Exponential Smoothing:
    - `revenue`: Fits a model with additive trend and seasonal components (period=4).
    - `total_expenditure`: Same as above.
    - `revenue_expenditure`: Computed as `revenue * total_expenditure`.
  - Generates SARIMA forecasts for 8 quarters (`sarima_forecast`) using `exog_future`.
  - Computes 80% confidence intervals using `get_forecast`.
10. **Output**:
  - Returns a dictionary:

- `train_df`: Training DataFrame with `ds`, `y`, `revenue`, `total_expenditure`, `revenue_expenditure`, `sarima_pred`.
- `test_df`: Test DataFrame with the same columns plus `sarima_pred`.
- `future_df`: Forecast DataFrame with `ds`, `yhat` (forecasted values), `yhat_lower`, `yhat_upper` (confidence intervals).
- `metrics`: Dictionary with `mae`, `rmse`, `mape`.

Error Handling

- Assumes input DataFrame is valid (from `load_data`).
- Handles missing values and outliers to prevent model failures.
- Uses robust SARIMA settings to avoid convergence issues.

Usage

Prerequisites

- **Python Version**: 3.6 or higher.
- **Libraries**: Install required packages:

```
pip install pandas numpy scikit-learn statsmodels
```

- **Input Data**: An Excel file with:
  - First column: Quarterly dates (e.g., "Mar 05 Q4").
  - Columns: "Net profit/(loss) for the period", "Total Revenue", "Total Expenditure".
  - Example format:

```
Quarterly Results | Net profit/(loss) for the period | Total Revenue | Total Expenditure
Mar 05 Q4        | 1000                             | 5000          | 4000
Jun 05 Q2        | 1100                             | 5200          | 4100
...
```

Example Usage

```
# Load Excel file
df = pd.read_excel("path/to/Infosys_Sorted_Quarterly_Data.xlsx", sheet_na
me="Sheet1")

# Preprocess data
processed_df, error = load_data(df)
if error:
    print(error)
    exit()

# Run SARIMA model
results = run_sarima_model(processed_df)

# Access results
print("Evaluation Metrics:")
for metric, value in results["metrics"].items():
    print(f"{metric.upper()}: {value:.2f}")

print("\nForecast for Next 8 Quarters:")
print(results["future_df"][["ds", "yhat"]].round(2))

# Optional: Save results to CSV
results["train_df"].to_csv("train_predictions.csv", index=False)
results["test_df"].to_csv("test_predictions.csv", index=False)
results["future_df"].to_csv("forecasts.csv", index=False)
```

Example Output

```
Evaluation Metrics:
MAE: 338.71
RMSE: 417.91
MAPE: 5.13

Forecast for Next 8 Quarters:
        ds    yhat
2025-03-31  6215.01
2025-06-30  5894.70
```

```
2025-09-30  6612.80
2025-12-31  6852.98
2026-03-31  6801.98
2026-06-30  6493.08
2026-09-30  7220.50
2026-12-31  7463.37
```