Documentation: ARIMA and SARIMAX Training and Evaluation

This documentation provides an overview of the training and evaluation scripts for ARIMA and SARIMAX models. The scripts are designed to train predictive models on time series data, optimize model parameters, and evaluate the models' performance.

Training Script (arima_sarimax_training.py)

Key Functions and Components:

1. **Data Preprocessing**
    - **Function:** load_and_preprocess_data()
    - Reads time series data from a CSV file, converts date columns to datetime format, and sets period_end as the index for time-based operations.
2. **Finding Optimal Model Orders**
    - **Function:** find_best_order()
    - Utilizes the auto_arima function to automatically identify the best parameters for ARIMA and SARIMA models.
    - Outputs:
        - ARIMA order (p, d, q) for non-seasonal components.
        - SARIMA seasonal order (P, D, Q, m) for capturing periodic patterns (e.g., 12-month seasonality).
3. **Model Training**
    - **ARIMA:** train_arima() trains an ARIMA model using the specified (p, d, q) parameters.
    - **SARIMAX:** train_sarimax() trains a SARIMAX model with exogenous variables and seasonal components using specified orders.
    - Exogenous variables include:
        - demand_supply_ratio, price_drop_ratio, pending_to_sold_ratio, market_heat_index, price_change_vs_inventory, sales_change_vs_supply.
4. **Model Serialization**

- **Function:** save_model_to_pickle()
- Saves trained models to disk using Python's pickle for future evaluation and use.

5. **Main Execution**
    - Loads data, finds optimal parameters, trains models, and saves them as serialized files.

Outputs:

- arima_final.pkl: Trained ARIMA model.
- sarimax_final.pkl: Trained SARIMAX model.

Evaluation Script (arima_sarimax_eval.py)

Key Functions and Components:

1. **Loading Test Data**
    - **Function:** load_test_data()
    - Reads test data from a CSV file, processes date columns, and sets period_end as the index.
2. **Model Deserialization**
    - **Function:** load_model_from_pickle()
    - Loads previously trained models from serialized files for evaluation.
3. **Metrics Calculation**
    - **Function:** calculate_metrics()
    - Computes key evaluation metrics:
        - **Mean Absolute Error (MAE):** Average absolute difference between predictions and actual values.
        - **Mean Squared Error (MSE):** Average squared difference.
        - **Root Mean Squared Error (RMSE):** Square root of MSE, reflecting model accuracy.

        - **Mean Absolute Percentage Error (MAPE):** Average percentage error of predictions.

4. **Model Evaluation**
    - **ARIMA:** evaluate_arima() forecasts using the ARIMA model and evaluates its performance.
    - **SARIMAX:** evaluate_sarimax() generates predictions using the SARIMAX model, including exogenous variables, and evaluates performance.
5. **Main Execution**
    - Loads test data, deserializes models, performs evaluations, and calculates metrics.

Outputs:

- **Prediction Files:**
    - arima_predictions.csv: Contains ARIMA predictions.
    - sarimax_predictions.csv: Contains SARIMAX predictions.
- **Metrics:** Metrics for both models are printed and returned as dictionaries.

Workflow Summary

1. **Training Phase (arima_sarimax_training.py)**
    - Load and preprocess the training data.
    - Automatically determine optimal model parameters.
    - Train ARIMA and SARIMAX models.
    - Serialize trained models for later use.
2. **Evaluation Phase (arima_sarimax_eval.py)**
    - Load and preprocess the test data.
    - Deserialize trained models.
    - Generate predictions using ARIMA and SARIMAX.
    - Evaluate model performance using standard metrics.

Usage Notes

- The training and evaluation pipelines are designed for modularity, enabling updates or modifications (e.g., additional features or metrics) with minimal changes.
- The auto_arima function simplifies hyperparameter tuning, but manual adjustments may be necessary for specific datasets.
- Both ARIMA and SARIMAX models rely heavily on data preprocessing; ensure accurate time series indexing and clean data inputs.

This documentation provides a foundational understanding of the scripts, facilitating their effective use and adaptation.