Summary Chapter # 10

The chapter provides an in-depth exploration of **Global Forecasting Models (GFMs)** for time series forecasting, emphasizing their advantages, drawbacks, and strategies to enhance their performance. It contrasts GFMs with **Local Forecasting Models (LFMs)** and introduces techniques to overcome common challenges in time series modeling. Below is a detailed summary:

Introduction to Global Forecasting Models

1. **What Are GFMs?**
    - Unlike LFMs, which train separate models for each time series, GFMs use a single model to forecast multiple time series simultaneously.
    - This approach assumes that all time series in a dataset share a common data-generating process or have related attributes, enabling shared learning.
2. **Why Use GFMs?**
    - **Efficiency**: GFMs reduce the computational and engineering complexity of managing multiple models, especially for large datasets.
    - **Accuracy**: They leverage the aggregated history of multiple time series, enabling better generalization and cross-learning.

Advantages of GFMs Over LFMs

1. **Improved Data Utilization**:
    - GFMs increase the "width" of datasets by using data from all series, enabling a data-driven approach even for time series with limited history.
    - LFMs, on the other hand, often rely on strong assumptions like seasonality or trend to compensate for small data sizes.
2. **Cross-Learning**:
    - GFMs share patterns across time series, benefiting those with limited data by learning from others with similar attributes.
    - For example, new products or households without historical data can still achieve accurate forecasts by borrowing patterns from related series.

3. **Multi-task Learning**:
    - Treats forecasting for each time series as a separate task but trains them together, enabling the model to learn shared features like seasonality.
    - Acts as a regularizer, preventing overfitting by focusing on commonalities across series.
4. **Engineering Simplicity**:
    - LFMs become impractical when dealing with thousands or millions of time series, requiring significant resources for training and managing.
    - GFMs streamline this process by training one model, reducing deployment, monitoring, and retraining overhead.

Drawbacks of GFMs

1. **Assumption of Common Data-Generating Process**:
    - GFMs assume relatedness among time series, which may not always hold true, leading to underfitting for highly diverse datasets.
2. **Struggles with Unrelated Series**:
    - For datasets with minimal similarity, GFMs might fail. However, some studies suggest meta-learning capabilities in GFMs may help even for unrelated series.
3. **Limited Interpretability**:
    - GFMs are less transparent than simpler LFMs, requiring advanced techniques for understanding their decisions.

Strategies to Improve GFMs

1. **Feature Engineering**:
    - **Lag Features**: Adding more lag terms (historical data points) enhances the model's memory of past events. Seasonal lags are particularly effective but must avoid overfitting.
    - **Rolling Features**: Use rolling averages to condense historical data into meaningful statistics like mean or maximum values.
    - **EWMA Features**: Incorporate exponentially weighted moving averages to introduce infinite memory into finite-memory models.

2. **Incorporating Meta-Features**:
    - Meta-features (e.g., product IDs, categories) distinguish between time series and improve forecasting for heterogeneous datasets.
    - **Encoding Techniques**:
        - **Ordinal Encoding**: Assigns numerical codes to categories but risks introducing unintended ordinal relationships.
        - **One-Hot Encoding**: Creates binary columns for each category but can become sparse with high-cardinality features.
        - **LightGBM's Native Encoding**: Gradient-boosting models like LightGBM and CatBoost handle categorical features directly and efficiently.
3. **Dataset Partitioning**:
    - Dividing datasets into smaller subsets improves model performance by focusing on more homogenous groups:
        - **Judgmental Partitioning**: Uses domain knowledge or meta-features for grouping (e.g., based on demographics).
        - **Algorithmic Partitioning**: Employs clustering techniques (e.g., t-SNE or Dynamic Time Warping) to identify groups based on similarity.
4. **Hyperparameter Tuning**:
    - GFMs allow for extensive hyperparameter optimization due to shorter training times.
    - Methods like **Bayesian Optimization** are computationally efficient, compared to grid or random search.

Performance Analysis

1. **Efficiency Gains**:
    - Training a GFM on 150 households took 57 seconds compared to 8 minutes for LFMs, achieving 777% faster training with an 8.78% decrease in Mean Absolute Error (MAE).
2. **Partitioning Results**:

- Algorithmic clustering outperformed judgmental methods and random partitioning, demonstrating the importance of grouping similar series.

3. **Hyperparameter Optimization**:
    - Bayesian optimization concentrated computational effort in promising regions, yielding better results than grid or random search.

Interpretability

1. **Challenges**:
    - GFMs are less inherently interpretable than simple statistical models like ARIMA.
2. **Post-Hoc Techniques**:
    - Methods like **Shapley Values**, **permutation importance**, and **LIME** explain model predictions by analyzing feature impacts.

Key Takeaways

1. **Superiority of GFMs**:
    - GFMs often outperform LFMs in terms of accuracy, scalability, and computational efficiency, particularly as the number of time series grows.
2. **Flexibility**:
    - Incorporating meta-features and advanced encoding allows GFMs to handle diverse datasets effectively.
3. **Future Directions**:
    - Transition to deep learning for time series forecasting, with GFMs serving as a robust baseline for machine learning-based approaches.

Conclusion

The chapter positions GFMs as a powerful paradigm in modern time series forecasting, offering practical solutions to common challenges. With proper tuning and enhancements, GFMs provide a scalable and accurate alternative to traditional forecasting methods, paving the way for more advanced approaches in the field.

NOTE: *The corresponding code for chapter 10 is added in the shared Github repository.*

References

- M. Joseph, Modern Time Series Forecasting with Python: Explore industry-ready time series forecasting using modern machine learning and deep learning. Packt Publishing Ltd, 2022.