

Advanced Forecasting and MLOps Implementation Workshop

Hierarchical forecasting, trade-flow balancing and a monitored prototype pipeline

FA-0547 · 3 days

Data & Analytics · DevOps, Cloud & Infrastructure

Compare forecasting approaches, reconcile hierarchical data and build a guided forecasting-pipeline prototype with tracking and monitoring.

Introduction

This three-day advanced technical workshop explores forecasting for hierarchical trade and economic data. Participants compare statistical, machine-learning and deep-learning approaches, engineer features, examine forecast reconciliation and use iterative proportional fitting to investigate compatible trade-flow margins.

The practical work develops a guided prototype using Python, SQL Server, Azure DevOps and Power BI. Selected models and integration steps are implemented; the wider architecture, monitoring and deployment options are reviewed through demonstrations and discussion. Completing the workshop does not establish that a system is production-ready.

Exercises use illustrative or approved non-confidential datasets in a test environment. A production rollout would require further validation, operational review, security controls and organisation-specific integration beyond the three-day workshop.

Learning outcomes

The workshop teaches participants to:

- Compare forecasting approaches against a chosen horizon, baseline and operational constraints.

- Engineer lag, seasonal and external-indicator features and evaluate a selected model with time-aware validation.
- Use hierarchical reconciliation and distinguish it from margin-constrained trade-flow balancing with iterative proportional fitting.
- Assess external indicators without treating correlation or a predictive test as proof of causation.
- Assemble a prototype ingestion, modelling and forecasting workflow with traceable datasets, parameters and artifacts.
- Outline CI/CD, drift monitoring, alerting and maintenance for a VM-based forecasting service.
- Review prototype limitations and plan the additional work needed for production.

Prerequisites

- Solid working experience with Python (data manipulation, model building).
- Familiarity with time-series concepts (trend, seasonality, ARIMA/ETS, basic ML).
- Basic knowledge of databases (SQL Server) and ideally cloud/DevOps (Azure DevOps, CI/CD).
- Understanding of data engineering concepts (data ingestion, ETL/ELT).

Working Python skills for data manipulation and model development are essential. Participants need an approved training Python environment, sample CSV/SQL data, and access appropriate to the Azure DevOps and Power BI exercises. Real-time/cloud infrastructure topics are architecture discussions rather than a prerequisite to build a full platform.

Training outline

1. Day 1 — Forecasting, Hierarchies and Model Evaluation

Forecasting Fundamentals and Hierarchical Data

Overview of forecasting methodologies

- Traditional statistical methods (ARIMA, ETS, exponential smoothing)
- Machine-learning/time-series ML methods (Gradient Boosting, Random Forests)
- Survey deep-learning alternatives: LSTM, GRU and Transformer-based models; implement a selected approach rather than every architecture.

Selecting the right technique for mid-term horizons

- Definition of mid-term horizon in export/import/trade context
- Trade-offs: interpretability vs accuracy vs operational cost

Data requirements and quality considerations

- Data ingestion: sources, frequency, granularity, missing values

- Data quality: completeness, consistency, hierarchy issues
- Handling hierarchical time-series (multi-level) data structures — drill-down/roll-up issues; ensuring coherence across levels.

Performance metrics and evaluation frameworks

- Forecast accuracy metrics (MAE, RMSE, MAPE, sMAPE)
- Check aggregate consistency of hierarchical forecasts alongside predictive error.
- Baseline vs benchmark models; cross-validation for time-series

Introduction to hierarchical forecasting frameworks in Python

- HierarchicalForecast reconciliation examples: Bottom-Up, Top-Down and MinTrace; check the requirements of the chosen method.
- Use of sktime for grouped/hierarchical forecasting.

Models and Feature Engineering

Time series deep-learning approaches

- LSTM and GRU architectures: when to use, how to configure
- Transformer approaches for time-series forecasting: compare suitability, data needs and computational cost without assuming superiority.

Ensemble approaches: evaluate whether combinations improve on the baseline.

- Hybrid statistical + ML + deep learning ensembles
- Bagging, stacking, blending in forecasting context

Feature engineering for forecasting

- Deriving time features (lags, rolling windows, seasonal indicators)
- External factors: trade data, commodity flows, economic indicators
- Encoding hierarchical identifiers (levels, country, commodity)

Handling seasonality, trends, external factors

- Decomposition (trend/seasonality/residual)
- Incorporating external regressors (economic indicators, sentiment)
- Dealing with hierarchical trends and cross-level interactions

Practical hands-on: building a forecasting model in Python

- From raw/hierarchical CSV or SQL data to model input
- Training, tuning, validation, forecasting
- Reconciling hierarchical forecasts to ensure coherence

2. Day 2 – Trade-Flow Balancing and External Indicators

Iterative Proportional Fitting and Constrained Flow Balancing

Understanding the flow-match methodology and implementation challenges

- Iterative proportional fitting adjusts a non-negative multidimensional table towards specified target margins; distinguish this balancing task from forecasting or generative flow-matching models.
- Illustrative multi-level country/commodity trade flows: dimensions, compatible margins and hierarchy constraints.

Optimization techniques for flow-matching algorithms

- Specify constraints, target margins, convergence criteria and computational considerations.
- Python techniques: matrices, sparse representations, reconciliation methods.

Troubleshooting common flow-match issues

- Data sparsity, missing levels, misalignment of hierarchies
- Convergence issues, stability of reconciliation, scalability concerns

Hands-on: Implementing flow-match optimization

- Build a small IPF-based trade-flow balancing exercise in Python.
- Compare the balanced flow table with forecast outputs and check the aggregation constraints explicitly.
- Evaluate margin agreement, convergence and forecast error separately; improved coherence does not guarantee improved predictive accuracy.

Economic Indicators and Trade-Data Analysis

Identifying key economic indicators for forecasting models

- Macro indicators (GDP, PMI, Exchange rates, Commodity prices)
- Sentiment indicators, trade policy indicators, export/import volumes

Cross-correlation analysis between economic indicators and historical trade data

- Correlation matrices, lag analysis and Granger predictive tests; distinguish predictive association from causal conclusions.
- Feature engineering: creating lagged indicator features, interaction terms

Understanding market volatility patterns and their impact on forecasting

- Volatility bursts, regime changes, structural breaks in time-series
- Implications for forecasting; methods to detect and model volatility shifts

Integration of economic sentiment and market indicators into forecasting pipelines

- Design workflow to ingest indicator data (SQL/CSV), merge with trade time-series, feed into model

Illustrative scenarios: economic shifts, trade patterns and forecasting error.

- Use approved published or synthetic examples of disruptions; do not claim an unidentified client's results.
- Discuss how comparable patterns could be evaluated in a participant's own setting.

3. Day 3 — Prototype MLOps Pipeline and Operational Review

Tracking, Deployment and Monitoring

End-to-end prototype pipeline in a training stack: Python, SQL Server and Azure DevOps.

- Architecture: data ingestion → feature engineering → model training → forecasting → deployment → monitoring
- Connect hierarchical data and balancing/reconciliation steps; discuss feature management and experiment tracking.

Model versioning and experiment tracking

- Tracking hyper-parameters, model artifacts, dataset versions
- Tools and best practices: MLflow, Git, Azure DevOps pipelines

Automated retraining and model drift detection

- Scheduling retraining workflows; detecting performance degradation over time
- Handling concept drift, data drift in forecasting systems

Production deployment strategies

- Illustrative VM-based deployment with Azure DevOps; discuss container-free patterns where a target environment requires them.
- Run the Python forecasting workflow, store outputs in SQL Server and review REST-service/Power BI integration options; do not assume model execution inside SQL Server.

Data engineering & infrastructure

- Architecture overview of real-time ingestion and Azure or custom ETL; the guided prototype uses a manageable training dataset.
- Data validation and monitoring: schema checks, quality checks in pipeline
- Discuss scaling patterns for on-premises or VM deployments rather than building a full scalable platform in the workshop.
- Compare cloud-native patterns with environments that constrain containers.

Monitoring & maintenance

- Review performance alerts, dashboards and drift signals using prototype monitoring examples.
- Compare a challenger forecast with a baseline using suitable time-aware evaluation; discuss controlled operational experiments where appropriate.

- Alerting and incident response workflows: when forecast performance drops, when reconciliation fails
- Continuous improvement workflows: how to collect feedback, update models, versioning, maintain pipeline health

Guided Build, Test Deployment and Handoff

Assemble a guided forecasting pipeline prototype for illustrative hierarchical trade data.

- CSV/SQL ingestion, selected features and model, forecast generation and output storage.
- Integrate and test the selected balancing or reconciliation step.
- Develop economic indicator correlation models and integrate into forecasting pipeline
- Deploy or demonstrate the prototype in an approved test environment using the Python/Azure DevOps/SQL Server/Power BI stack.
- Create selected monitoring examples for forecast error, drift or margin/reconciliation failures.
- Troubleshooting scenarios: pipeline failures, drift detection alarms, hierarchy reconciliation issues
- Review the prototype, known limitations and further engineering required before production.