DATA SCIENCE & ML

Time Series Forecasting in Practice

Classical models, gradient boosting and deep learning for

Houssam Kodad

PDF · DATAFORGE BOOKS

Questions about this sample or the full edition: **support@dataforgebooks.com**

CONTENTS

Table of Contents

CHAPTER 01

Forecasting as a Business Problem

Forecasting is where clean statistics meets the messiness of real operations: promotions, holidays, stockouts, price changes, and the occasional structural break that makes last year's pattern useless. A forecast is never an academic exercise — it feeds a decision, and the decision is what we are really optimising. Keeping that in view is what separates forecasting that helps the business from forecasting that merely looks impressive.

This chapter grounds the whole book in that reality. We ask what a forecast is actually for, why accuracy and decision quality are not the same thing, and how to choose an error metric that reflects what your business genuinely cares about. We also preview the toolkit — classical models, gradient boosting, and modern deep learning — and the principle that decides between them: complexity must earn its place.

The throughline is humility about the future and rigour about evaluation. Anyone can fit a model that explains the past; the entire skill of forecasting is producing predictions that hold up out of sample and quantifying how much to trust them. We build that habit from the first page.

What a Forecast Is For

Nobody wants a forecast for its own sake. They want to decide how much stock to order, how many staff to roster, how much cash to hold. The forecast is an input to that decision, and its value is measured by how much better the decision becomes — not by how closely a line hugs the actuals in isolation. Start every forecasting problem from the decision it serves, and many apparent dilemmas resolve themselves.

This reframing changes what "good" means. A forecast that is slightly less accurate on average but never badly under-predicts demand may be far more valuable for an inventory decision where stockouts are catastrophic and overstock is merely annoying. The right forecast is the one that leads to the best decisions given the real costs of being wrong in each direction — which is rarely the one that minimises a symmetric error.

Accuracy vs Decisions

It is entirely possible to improve a forecast's accuracy and make the business worse off. If your errors become more symmetric while the cost of those errors is not — over- and under-prediction carry different consequences — then minimising symmetric error optimises precisely the wrong thing. The metric and the cost structure must agree, or the metric will quietly mislead you.

Good forecasting practice keeps the asymmetry of the decision in view at all times. Sometimes that means producing a quantile forecast rather than a single point, so the decision can be made at the right level of caution. Sometimes it means a custom loss that penalises the expensive direction more heavily. The model serves the decision; we never let the convenience of a standard metric reverse that relationship.

Decomposition: Trend, Seasonality, Residual

Almost every time series can be understood as the sum of a few components: a slow-moving trend, one or more seasonal cycles, and the residual noise that remains. Decomposing a series into these parts is the first thing to do with any new dataset, because it tells you what structure exists to be modelled and what is irreducible randomness you should not chase.

Seasonality is frequently the dominant signal — daily, weekly and yearly cycles in demand, traffic, and operations. Capturing it well, whether through classical seasonal models or engineered calendar features, accounts for a large fraction of achievable accuracy in most business series. We use decomposition both as an exploratory tool and as a guide to which modelling approach a series actually calls for.

```
# Quick decomposition to see what structure exists before modelling.
from statsmodels.tsa.seasonal import STL

res = STL(series, period=7).fit()        # weekly seasonality
trend, seasonal, resid = res.trend, res.seasonal, res.resid
# A flat residual means the model has captured the structure;
# patterns left in resid mean there is signal still on the table.
```

Classical Models That Still Win

It is fashionable to reach straight for deep learning, but classical methods remain the right answer for a large share of real forecasting problems, especially when you have one or a few series with clear seasonality and limited history. Exponential smoothing and ARIMA-family models are fast, interpretable, well understood, and frequently beat far more complex alternatives on the kind of data businesses actually have.

Their interpretability is a feature, not a consolation prize. When a stakeholder asks why the forecast moved, a model whose components you can read gives an answer; a black box does not. We treat classical models as strong, default baselines that any fancier approach must beat decisively before it earns the extra complexity and maintenance burden — a bar more candidates fail than their advocates expect.

Forecasting as Supervised Learning

The bridge from classical forecasting to modern machine learning is a simple reframing: treat the next value as a label and the recent past as features. Once a series is expressed as a supervised-learning table — lags, rolling statistics, calendar fields, external regressors — the entire arsenal of gradient boosting becomes available, and it shines especially when you must forecast many related series at once.

This reframing also lets a single model learn across series, borrowing strength from products or stores that behave similarly. The craft moves into feature construction and validation, and the dangers move with it: a lag or rolling feature computed without respect for time leaks the future and produces beautiful, useless backtests. We handle that risk explicitly, because it is the most common way forecasting projects fool themselves.

```
# Reframe a series as supervised rows: features must use only the past.
df = df.sort_values("date")
for lag in (1, 7, 14):
    df[f"lag_{lag}"] = df["y"].shift(lag)
df["roll_7"]   = df["y"].shift(1).rolling(7).mean()
df["dow"]      = df["date"].dt.dayofweek
df["is_month_start"] = df["date"].dt.is_month_start.astype(int)
```

Validating Forecasts Honestly

A forecast you cannot evaluate honestly is a forecast you cannot improve, and the ordinary cross-validation of machine learning is actively dangerous here. Shuffling rows lets the model train on the future to predict the past, producing scores that evaporate in production. Time series demand backtesting: train on the past, predict the following period, roll forward, and repeat — exactly mirroring how the model will be used.

Equally important is comparing against naive baselines. "Tomorrow equals today" and "this week equals last week" are deceptively strong, and a sophisticated model that cannot clearly beat them is not worth deploying. We make rolling-origin backtesting and honest baseline comparison non-negotiable habits, because they are the only defence against the very human tendency to believe a model that flatters us.

Uncertainty Is Part of the Forecast

A single number is a dangerously incomplete forecast. Decisions need a sense of the range of plausible outcomes — how bad could demand get, how high might it spike — and a point forecast hides exactly the information a good decision requires. Prediction intervals and quantile forecasts are not optional embellishments; they are often the most decision-relevant part of the output.

We treat uncertainty as a first-class deliverable throughout the book, whether it comes from a model's own intervals, from quantile regression, or from conformal methods that wrap any model with calibrated bounds. Communicating that uncertainty clearly to the people making decisions is the final, frequently neglected step that turns a forecast from a number into genuine support for a choice.