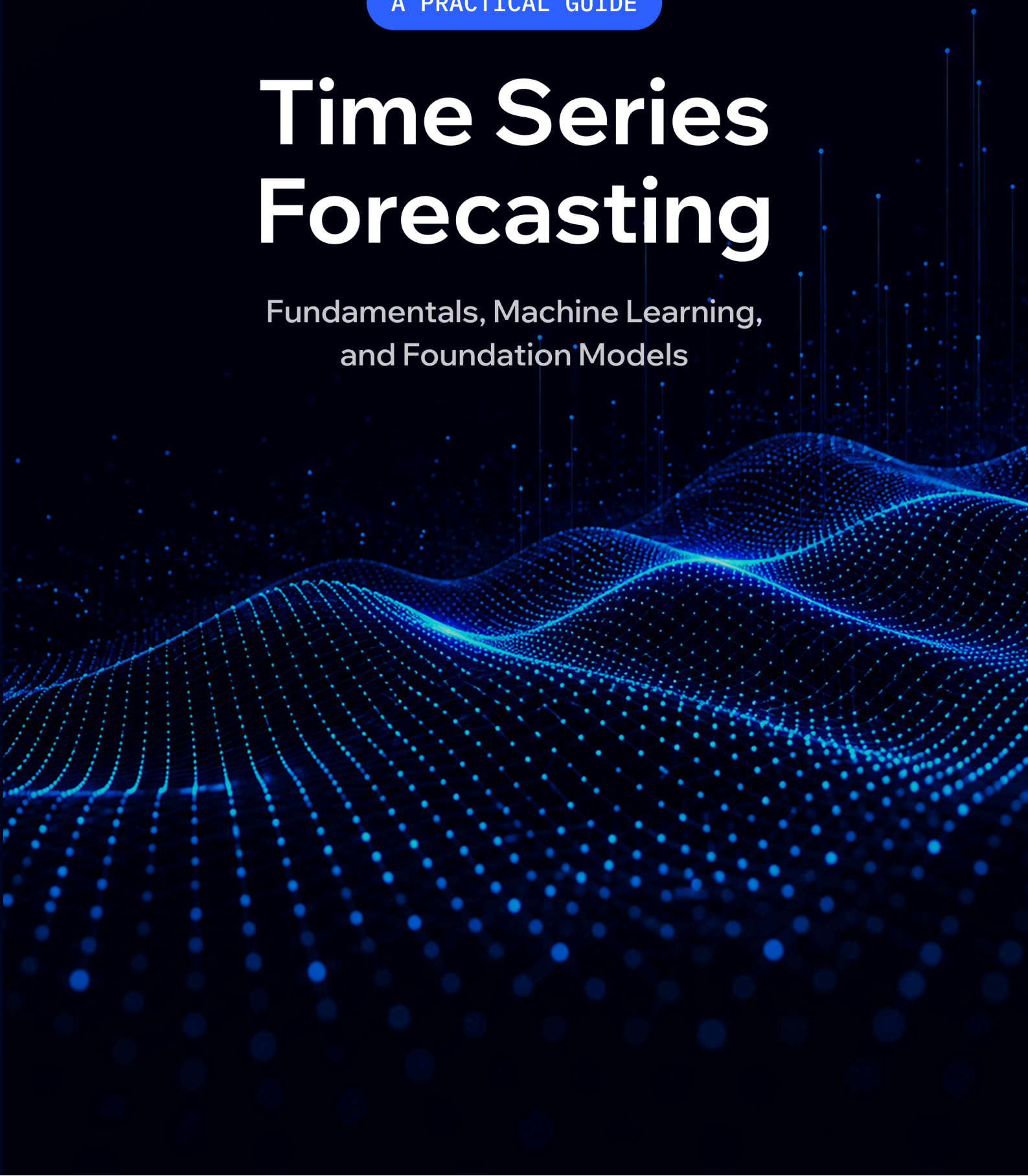
marvik.

A PRACTICAL GUIDE

Time Series Forecasting

Fundamentals, Machine Learning, and Foundation Models



marvik.

Built for the Now

Marvik builds production AI systems for complex operating environments.

AI-native and forward-deployed since 2018, Marvik helps leading organizations move beyond pilots and demos, turning high-value AI opportunities into systems engineered around real data, real workflows, and measurable business outcomes.

Our team of more than 250 AI and data specialists has delivered over 300 production deployments for global technology leaders, Fortune 500 enterprises, and category-defining platforms. We provide end-to-end services, from initial strategy to full deployment.

Most AI initiatives fail not because the model is impossible, but because the real environment is harder than the demo: fragmented data, legacy platforms, governance constraints, and no clear path to adoption. Marvik is built specifically to bridge that gap, combining technical expertise in agentic systems, predictive analytics, generative AI, robotics & automation, computer vision, and data engineering with the engineering discipline to ship and own what we build.

Strategic partnerships with NVIDIA, Anthropic, OpenAI, Oracle, Microsoft, AWS, Google Cloud, and ElevenLabs give us direct access to advanced tooling and enterprise-grade infrastructure, translating into faster deployment cycles and production-ready foundations from day one.

Learn more at **www.marvik.ai**

Executive Summary

Time Series Forecasting: A Practical Guide

Forecasting works best as a portfolio of tools matched to the shape of each problem, not as a single default model. This guide covers the full landscape, from classical statistics to the foundation models that ran through two complete generations between 2023 and 2026, with one goal: helping a team pick the right tool for the problem in front of them, and recognise when it is the wrong one.

The book opens with a road map: a decision table and a decision tree that route the reader from their actual situation (how many series, how much history, external variables or not, zeros or not, explainability required or not) to the chapter that answers it. A reader on a deadline can start and end there.

Part one: **The foundations**
Decomposition, stationarity and diagnostics: the work that determines whether anything built afterwards can be trusted. Then the classical tools (ARIMA/SARIMA), the boosting family (XGBoost, LightGBM, CatBoost) for problems where price, promotions or weather drive the outcome, and Global Models, which train a single model across thousands of SKUs by sharing signal between series. Two chapters cover the discipline that makes any of it real: backtesting instead of one lucky split, and uncertainty quantification (classical intervals, Bayesian, and conformal prediction), because a forecast paired with an interval is a decision, while a bare number is a guess.

Part two: **the frontier, tested rather than announced.**
Foundation models forecast a series they have never seen, with no training at all. The guide covers both generations: TimeGPT, Chronos and MOIRAI first, then Chronos-2, Moirai 2.0, TimesFM 2.5 and Sundial, which add multivariate support and covariates while getting significantly smaller and faster. Rather than take those claims on trust, the guide benchmarks them head to head against its own SARIMA and boosting baselines, and reports every result, including the uncomfortable one, where the boosting model finished last. Not because tree models are weak, but because it was handed a thin feature set. That is the chapter's real lesson: a foundation model asks nothing of you, while a boosting model is only ever as good as the features you build for it.

It then argues the other side, at length: the case that in domains with strong, well-defined seasonality a simple statistical method can match or beat a foundation model; the argument that these models learn statistical shape but not cause, which caps how far they truly generalise; the dependence of zero-shot quality on the pre-training domain; and explainability, where at the level of an individual prediction they remain a black box, which rules them out whenever a stakeholder needs a reason rather than a number.

The operating recommendation that follows is a modest one: before investing in feature engineering, run a foundation model as the reference to beat. If a model trained on your own data cannot beat it, that work was never necessary. If it can, the foundation model has already done its job: it is the baseline that justifies the effort.

Executive Summary

The closing chapters

Handle what no model solves automatically (intermittent demand, cold start, anomaly detection) and the methodological mistakes that, in professional practice, cost more than the technical ones.

The takeaway for technical and business readers alike: whichever model is state of the art today will be replaced within a year or two. What decides whether a forecast can be trusted (diagnosing the series before modelling it, validating with backtesting instead of one lucky split, and attaching an interval to every number) does not depend on which model happens to be fashionable.

Table of contents

Road Map 9

1 The anatomy of a time series 11

1.1 Introduction 11

1.2 Our case study: airline passengers 11

1.3 Deconstructing the series 12

1.3.1 The dilemma: additive or multiplicative? 12

1.4 Residual analysis 13

1.5 Conclusion 13

2 Stationarity and exploratory analysis 14

2.1 Introduction: the problem of comparing apples and oranges 14

2.2 The diagnosis: what do the data tell us? 14

2.3 The practical treatment 15

2.3.1 Step 1: from absolute numbers to percentages 15

2.3.2 Step 2: the growth rate 16

2.4 The formal test: Augmented Dickey-Fuller test (ADF) 17

2.4.1 How does it work? 17

2.4.2 Results on our data 17

2.5 Conclusion 18

3 The classical arsenal: ARIMA 19

3.1 Introduction 19

3.2 The building blocks: AR and MA 19

3.2.1 AR: the past predicts the future 19

3.2.2 MA: shocks ripple through time 19

3.2.3 Why both matter 20

3.3 Quick identification reference 20

3.4 ARIMA: handling trend 20

3.5 SARIMA: adding seasonality 21

3.6 ARIMAX: adding external variables 22

3.7 The practical shortcut: `auto_arima` 22

3.8 Diagnostic checking 22

3.9 Conclusion 23

4 Time series as supervised regression 24
4.1 Introduction: breaking the paradigm 24
4.2 The case study: a large retailer 24
4.3 Feature engineering: teaching time to the machine 24
4.3.1 1. Calendar features 24
4.3.2 2. Lags 25
4.3.3 3. Rolling windows 25
4.3.4 The resulting dataset 25
4.4 The boosting family: not just XGBoost 25
4.5 Advanced feature engineering 26
4.6 Multi-step forecasting strategies 26
4.7 The duel: XGBoost vs ARIMA 27
4.7.1 Why compare? 27
4.7.2 How do we measure success? 27
4.7.3 Visual results 27
4.7.4 Results analysis 28
4.7.5 Which features were most important? 28
4.7.6 Beyond feature importance: SHAP 28
4.8 Hyperparameter essentials 29
4.9 Selection guide: which model should I use? 29
4.10 Conclusion 29

5 Validation and error metrics 30
5.1 Introduction: the danger of the “lucky split” 30
5.2 The solution: backtesting 30
5.3 Error metrics: how do we measure success? 30
5.3.1 1. MAE (mean absolute error) 31
5.3.2 2. RMSE (root mean squared error) 31
5.3.3 3. MAPE (mean absolute percentage error) 32
5.3.4 4. WMAPE (weighted MAPE) 32
5.4 Error diagnostics: from residuals to features 32
5.4.1 How to read this chart? 33
5.4.2 From residuals to features: capitalizing on errors 33
5.5 Conclusion 33

6 Navigating uncertainty 34
6.1 Introduction: the weather forecast 34
6.2 Classical confidence intervals 34
6.2.1 What is a confidence interval? 34
6.2.2 How are they calculated? 34

6.2.3 The assumptions 35
6.2.4 Interpretation on the logarithmic scale 35
6.2.5 The counterfactual: intervals on raw vs transformed data 36
6.3 Bayesian forecasting: the complete distribution 36
6.3.1 What is the Bayesian approach? 36
6.3.2 Results 37
6.3.3 Advantages of the Bayesian approach 37
6.3.4 Limitations 37
6.4 Conformal prediction: the modern approach 38
6.4.1 The idea in 3 steps 38
6.4.2 Numerical example 38
6.4.3 The exchangeability assumption 39
6.4.4 Results: Conformal vs parametric 39
6.5 Comparison and selection guide 40
6.6 Conclusion 41

7 Global models 42
7.1 Introduction: the 10,000-product problem 42
7.2 The solution: global models 42
7.2.1 The data format: Panel 42
7.3 The experiment: local vs global 43
7.3.1 Experiment design 43
7.3.2 Speed 43
7.4 Accuracy results 43
7.4.1 The cross-learning phenomenon 43
7.5 When global, when local, when hybrid 44
7.6 Evaluation beyond MAE 45
7.7 Clustering series before training 45
7.8 Ensembling: the simplest win 46
7.9 Retraining in production 46
7.10 Conclusion 46

8 Foundation Models: theory 47
8.1 Introduction: beyond training 47
8.2 The cold start problem 47
8.2.1 When does it occur? 47
8.2.2 Why do traditional methods fail? 47
8.2.3 Modern strategies 48
8.3 Zero-shot forecasting: predicting without training 48
8.3.1 Our experiment 48
8.4 Transformers for time series 49
8.4.1 What is attention? 49

8.4.2 Key components of a transformer for time series 49
8.5 Foundation Models 50
8.5.1 How are they trained? 50
8.5.2 The Foundation Model ecosystem 50
8.5.3 Field trends 51
8.6 Conclusion 51

9 Foundation Models in practice 52
9.1 Introduction 52
9.2 Case 1: TimeGPT (Nixtla) 52
9.2.1 Characteristics 52
9.2.2 Results 53
9.2.3 Limitations 53
9.3 Case 2: Chronos (Amazon) 53
9.3.1 Characteristics 53
9.4 Case 3: MOIRAI (Salesforce) 54
9.4.1 Characteristics 54
9.5 The consolidated benchmark 55
9.5.1 Interpretation of results 55
9.6 The other side of the coin: limitations and criticisms 56
9.6.1 Criticism 1: they do not always outperform simple models 56
9.6.2 Criticism 2: the generalization problem 56
9.6.3 Criticism 3: zero-shot depends on the pre-training domain 56
9.6.4 Criticism 4: covariates and exogenous variables 56
9.7 When Foundation Model, when not? 57
9.8 Conclusion 57

10 Real-world challenges 58
10.1 Introduction 58
10.2 Intermittent demand: the problem of zeros 58
10.2.1 The real problem 58
10.2.2 The counterfactual: why do the models from Ch 3-4 fail? 58
10.2.3 The solution: Croston-SBA method 59
10.2.4 A note on metrics 61
10.3 Cold start: the phantom product 61
10.3.1 The problem 61
10.3.2 Strategy 1: similarity-based transfer 62
10.3.3 Strategy 2: global model with cross-learning 62
10.3.4 Strategy 3: zero-shot with Foundation Models 62
10.3.5 Which one to choose? 63
10.4 Anomaly detection: finding the unusual 64
10.4.1 Why does it matter? 64

10.4.2 The counterfactual: the cost of ignoring anomalies 64
10.4.3 Three detection methods 64
10.5 Common mistakes in forecasting practice 65
10.5.1 Data handling errors 65
10.5.2 Model selection errors 66
10.5.3 Errors in communicating results 66
10.6 Final reflections 66
Next Steps 67
Bibliography 68

CHAPTER 0

Road map

This section is a shortcut: if you already have a concrete problem and want to know where to start, use it as a quick reference. If you prefer to go through the full guide, skip it and come back when you need it.

Before choosing a model: if your data has missing values, anomalies, or irregular frequency, start with Chapter 2 (diagnostics) and the anomaly section in Chapter 10. No model compensates for dirty data.

Which problem, which tool?

Situation	Exog.?	Recommended method	Chapter
1 series, >2 seasonal cycles, no external variables	No	SARIMA	3
1 series, with price/promos/weather or other external variables	Yes	XGBoost / Boosting	4
1 series, limited data (<1 seasonal cycle)	—	Foundation Models	8–9
10–50 similar series	Yes/No	XGBoost (individual) or Global Model	4, 7
50+ series	Yes/No	Global Model	7
Intermittent demand (many zeros)	—	Croston	10
New product with no history	—	Transfer / Global / Foundation	7,8,9,10
Need confidence intervals	—	Ch 6 (applies to any model)	6
Anomaly detection	—	Isolation Forest / Z-Score	10

Quick decision tree

Do you have exogenous variables (price, weather, promotions, calendar)?

- **Yes** → XGBoost/Boosting (Ch 4). If you also have 50+ series → Global Model (Ch 7).
- **No** → How much data do you have per series?
  - More than 2 seasonal cycles → SARIMA (Ch 3) or XGBoost (Ch 4)
    Less than 1 cycle → Foundation Models (Ch 8–9) or Cold Start (Ch 10)

How many series?

- **1–10:** individual model per series
- **10–50:** if they are similar → Global (Ch 7); if very different → individual
- **50+:** Global Model (Ch 7)

Special cases:

- Many zeros? → Croston (Ch 10)
- Need uncertainty? → Ch 6 (applies to any model)
- Long horizon (>3 cycles)? → Uncertainty dominates; prioritize Ch 6

A note on explainability

If your stakeholder needs to understand why the model predicts what it predicts:

- **SARIMA:** directly interpretable coefficients ("the effect of last month is 0.7").
- **XGBoost:** explainable via SHAP ("price was the most important variable").
- **Foundation Models:** black box; hard to explain at the individual level.

What you will learn in each chapter

1. Anatomy of a Time Series — Trend, seasonality, residuals. How to decompose and understand any series.

2. Series Diagnostics — Stationarity tests (ADF), ACF/PACF, transformations. The diagnostic toolkit.

3. Classical Models (ARIMA) — AR, MA, ARIMA, SARIMA. When and how to use them for univariate series.

4. Machine Learning (Boosting) — XGBoost, LightGBM, CatBoost. Feature engineering and exogenous variables.

5. Validation — Temporal splits, backtesting, metrics. How to evaluate honestly.

6. Uncertainty — Prediction intervals, conformal prediction, Bayesian approaches.

7. Global Models — One model for many series. Cross-learning and scaling.

8. Foundation Models I — Chronos, MOIRAI. Pre-trained models for zero-shot forecasting.

9. Foundation Models II — Critical evaluation, benchmarks, and practical limitations.

10. Real-World Challenges — Intermittent demand, cold start, anomalies, and a comprehensive decision matrix.

For a detailed evaluation of when Foundation Models add value over alternatives, see Chapter 9. For special problems (intermittent demand, cold start, anomalies), see Chapter 10.

CHAPTER 1

The anatomy of a time series

1.1 Introduction

Welcome to the fascinating world of time series. Unlike cross-sectional data where the order of observations does not matter, in a time series **the order is fundamental**. The past dictates, to a large extent, the present.

A time series is a sequence of data points indexed in chronological order. Mathematically, we denote it as:

$$\{Y_t\}_{t=1}^{T} = \{y_1, y_2, \dots, y_T\}$$

1.2 Our case study: airline passengers

For this first chapter, we will analyze the classic Box & Jenkins dataset: **Monthly Airline Passenger Numbers 1949-1960**. This dataset contains the monthly number of passengers of a U.S. airline.

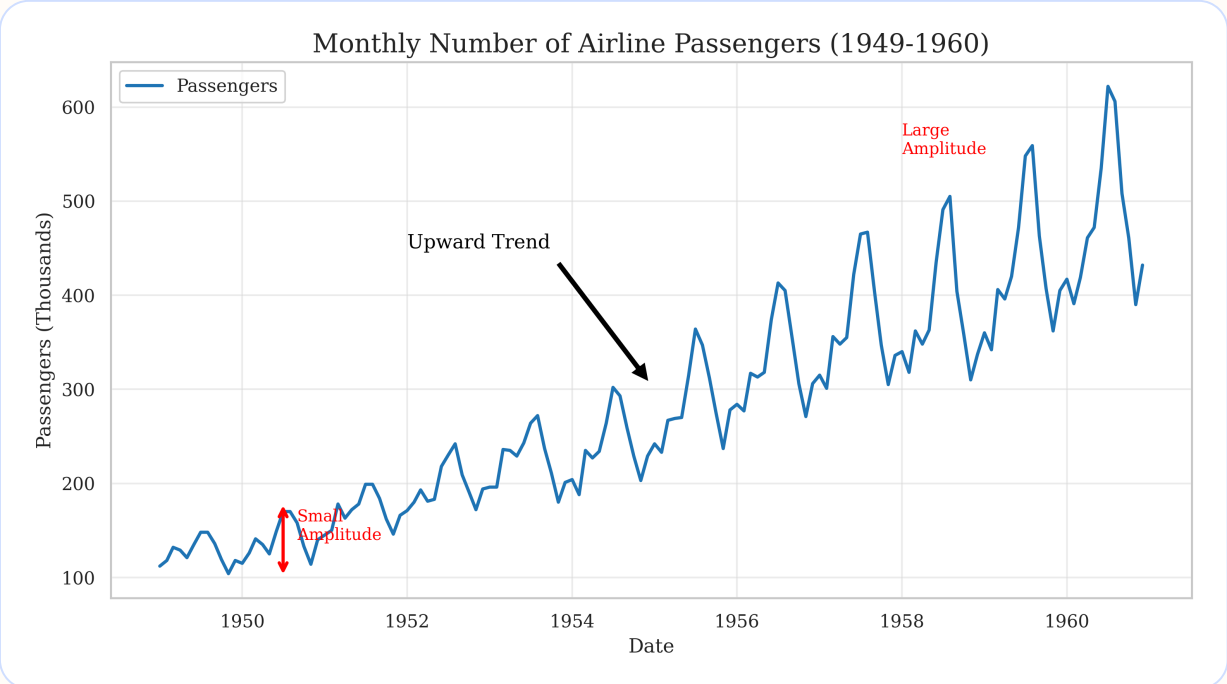


Figure 1.1: Monthly evolution of passengers (1949-1960)

When observing Figure 1.1, a trained economist immediately notices three things:

1. Positive Trend: The number of passengers grows consistently over the decade.

2. Strong Seasonality: There is a cyclical pattern that repeats every 12 months (peaks in summer, troughs in winter).

3. Heteroscedasticity (Non-constant Variance): Note how the seasonal oscillations become wider as the series rises. The difference between summer and winter of 1960 is much larger than that of 1950.

1.3 Deconstructing the series

We can think of a time series as the sum (or product) of three invisible components:

- **Trend** (T_t)**:** The long-term direction.
- **Seasonality** (S_t)**:** The short-term repetitive pattern.
- **Residual** (R_t)**:** The random or irregular noise that the model does not explain.

1.3.1 The dilemma: additive or multiplicative?

This is the first crucial modeling decision.

Additive Model: Assumes that the magnitude of seasonality is constant, regardless of the level of the series.

$$Y_t = T_t + S_t + R_t$$

Multiplicative Model: Assumes that seasonality grows proportionally to the level of the series.

$$Y_t = T_t \times S_t \times R_t$$

Let us see what happens when we try to force both models onto our data.

We apply multiplicative decomposition to our passenger series. Figure 1.2 shows the result: four panels revealing the hidden "layers" of the data.

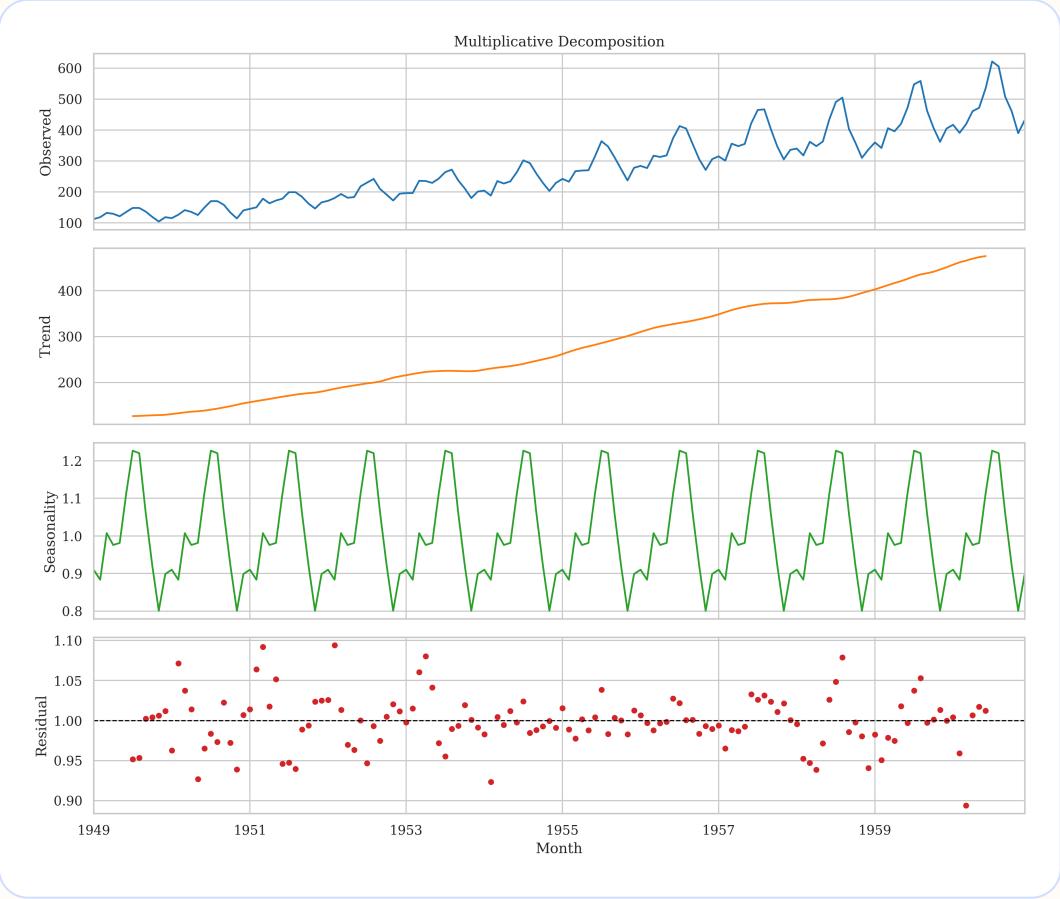


Figure 1.2: Multiplicative Decomposition. From top to bottom: (1) the original series, (2) the extracted trend —long-term direction—, (3) the seasonal component —pattern repeating every 12 months, with peaks in summer and troughs in winter—, and (4) the residuals —oscillating around 1.0 with stable variance, indicating the model captured the data structure—.

But how do we know the multiplicative decomposition is the correct one and not the additive? The answer lies in the residuals: what is "left over" after extracting trend and seasonality. If the model is good, the residuals should be **white noise** (pure randomness, no patterns).

1.4 Residual analysis

Let us compare side by side the residuals of both models. The difference is revealing.

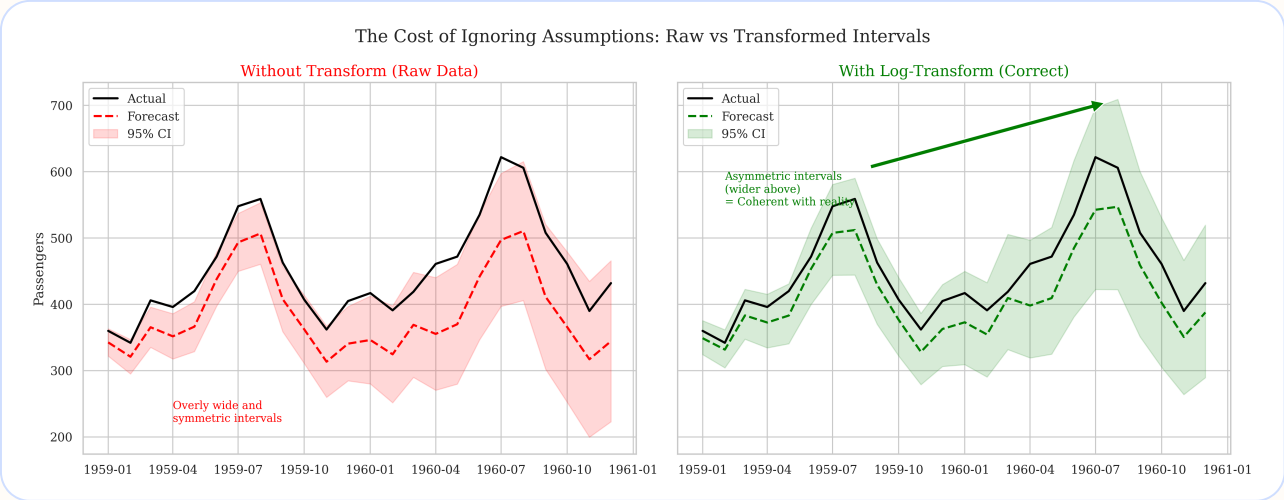


Figure 1.3: Direct comparison of residuals. **Left (Additive):** errors form a fan that grows over time —from ±15 passengers in 1950 to ±60 in 1960, a 3x increase. Arrows highlight this expansion. **Right (Multiplicative):** errors remain within a stable band of ±5% throughout the entire decade. Arrows confirm that the band is the same at the beginning and end of the series.

In the left panel of Figure 1.3 (Additive Model), we observe a **fan pattern**: errors are small at the beginning (±15 passengers) and enormous at the end (±60 passengers). This indicates that the additive model failed to capture the data structure. Why? Because it assumed seasonality always has the same absolute magnitude (e.g., "in summer, passengers always increase by 50"), but in reality the increase is proportional to the level ("in summer, passengers always increase by 30%"). As the series grows from 100 to 500 passengers, the error grows proportionally.

In the right panel (Multiplicative Model), residuals oscillate within a stable band of ±5% throughout the entire series. Both in 1950 and 1960, the arrows confirm that variance does not change. This looks much more like random noise: the multiplicative model "understood" that seasonality grows proportionally to the level.

1.5 Conclusion

For economic and business time series (sales, demand, traffic), where growth is usually percentage-based (exponential) and volatility increases with size, **the multiplicative model is generally the correct choice**.

In the next chapter, we will prepare the ground for prediction by introducing the concept of **Stationarity** — the fundamental diagnostic step before we can build forecasting models in Chapter 3.

CHAPTER 2

Stationarity and exploratory analysis

2.1 Introduction: the problem of comparing apples and oranges

Imagine you want to predict a teenager's height based on their height from previous years. It is difficult, because the teenager is in the middle of a "growth spurt." Their average height changes every year.

In contrast, predicting an adult's height is easier: their average height is constant. If one day they measure a bit taller, it is probably because they are wearing thick-soled shoes (noise), but they will return to their mean.

In time series, we call this state of a "stable adult" **Stationarity**.

- **Non-Stationary (Teenager):** The mean and variance change over time. (E.g., Airline passengers in the 1950s).
- **Stationary (Adult):** The mean and variance are constant. The statistical patterns do not depend on when we look at the series.

Our goal in this chapter is to turn our "teenager" series into an "adult" series so that we can model it.

2.2 The diagnosis: what do the data tell us?

Before applying any treatment, we need a diagnosis. The main instrument is the **Correlogram (ACF)**: the Autocorrelation Function.

The ACF tells us how much the series resembles itself in the past. Each bar represents the correlation between the series and its version lagged by k periods ("lag k"):

- **Lag 1:** Does this month's value resemble last month's?
- **Lag 12:** Does this month's value resemble that of a year ago?

If the series has a trend, the ACF decays very slowly ("the distant past still matters a lot"). If the series is stationary, the ACF drops to zero quickly ("only the recent past matters").

In the top panel (ACF), the blue bars decrease very slowly. This confirms what we saw in Chapter 1: the series has a strong trend. Today's number of passengers depends on that of 1 month ago, 2 months ago, 5 months ago, even 12 months ago. The series has "infinite memory," which is a classic symptom of non-stationarity.

In the bottom panel (PACF), the **Partial Autocorrelation** provides complementary information. While the ACF measures the total correlation between Y_t and Y_{t-k} (including the indirect influence of all intermediate lags), the PACF isolates the direct correlation of each lag, removing the effect of intermediate ones. In our case, the PACF shows a very high bar at lag 1 and then drops abruptly. This indicates that the primary relationship is with the immediately preceding month: once we know Y_{t-1}, lags 2, 3, etc. contribute little additional information on

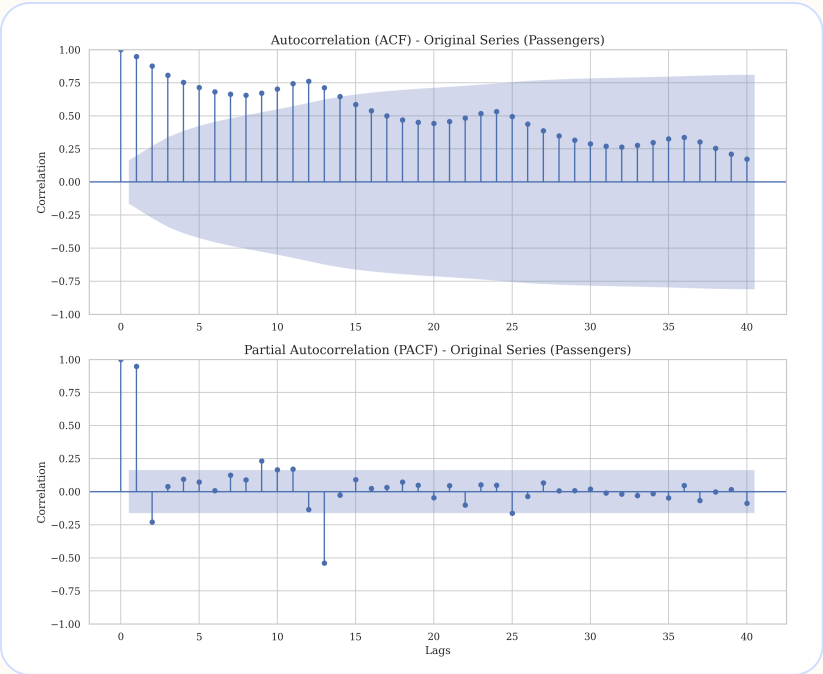


Figure 2.1: ACF of the original Passengers series. **Top (ACF):** The bars decay very slowly, indicating a strong trend. Bars extending beyond the shaded blue zone are "statistically significant." **Bottom (PACF):** The Partial Autocorrelation shows the direct correlation with each lag, removing the influence of intermediate lags.

This pattern (slowly decaying ACF + PACF that cuts off at lag 1) is the classic fingerprint of an autoregressive process of order 1 with a trend. You will also notice a significant negative spike around lag 13 — this is the seasonal fingerprint we will address in Step 2.

2.3 The practical treatment

2.3.1 Step 1: from absolute numbers to percentages

In Chapter 1, we saw that the variance of the series grew with its level (heteroscedasticity). In economics, when something grows exponentially, the standard tool is the **Natural Logarithm**.

Why does it work? Because the logarithm converts exponential growth (where changes are proportional to the level) into linear growth (where changes are constant). In other words, it transforms an accelerating curve into a straighter line. Let us see with a numerical example:

- If passengers go from 100 to 110 (10% growth): $\log(110) - \log(100) = 4.70 - 4.61 = 0.095$
- If passengers go from 400 to 440 (10% growth): $\log(440) - \log(400) = 6.09 - 5.99 = 0.095$

In both cases, although the absolute numbers are very different (a difference of 10 vs. 40), the logarithmic change is identical (0.095). The logarithm "normalizes" the differences: 10% is 10%, regardless of whether we start from 100 or 400.

Visually, the transformed series (Figure 2.2) no longer shows the "fan" effect from Chapter 1: the seasonal oscillations now have a uniform amplitude across the entire series, whether in 1950 or 1960. We have cured the heteroscedasticity.

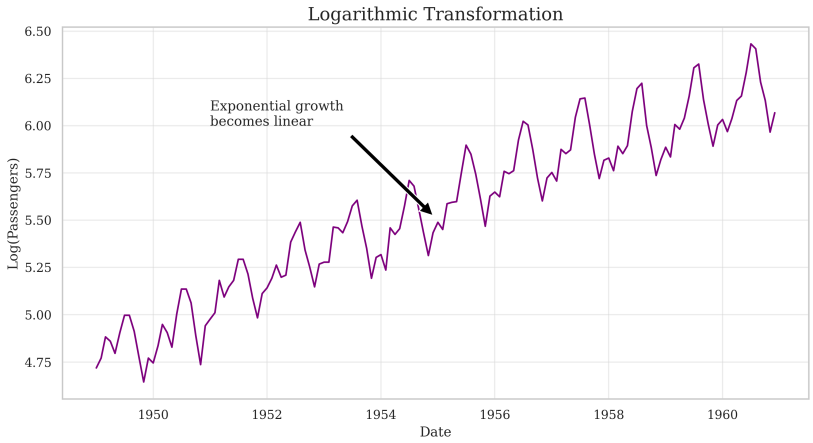


Figure 2.2: Logarithmic Transformation. The exponential curve "flattens" into a straighter line. Note that the seasonal oscillations now have a more uniform amplitude (compare with Figure 1.1 from Chapter 1).

2.3.2 Step 2: the growth rate

We have stabilized the variance, but the series still rises: we still have a trend. To remove it, instead of predicting the level of the series, we will predict the **change** between one period and the previous one.

Mathematically:

$$\Delta \log(Y_t) = \log(Y_t) - \log(Y_{t-1}) \approx \text{Percentage Growth Rate}$$

Numerical example: If $\log(Y_t) = 5.50$ and $\log(Y_{t-1}) = 5.45$, then $\Delta = 0.05$, which is interpreted as approximately 5% growth. This is something any CEO understands: "Last month we grew by 5%."

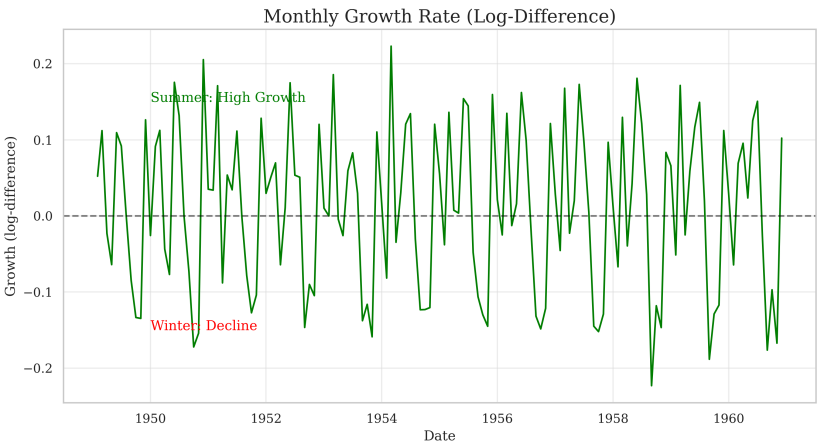


Figure 2.3: Monthly Growth Rate (first difference of the logarithm). The series now oscillates around zero: positive peaks correspond to summer months (high demand) and negative troughs to winter. There is no longer an upward trend.

Eureka! In Figure 2.3, the trend has disappeared. The series goes up and down around 0%.
In summer we grow a lot (positive peaks) and in winter we decline (negative peaks). But we still see a repetitive pattern: the seasonality.

2.4 The formal test: Augmented Dickey-Fuller test (ADF)

So far, our diagnosis has been visual ("it looks stationary"). But in science we need a formal verdict. The **Augmented Dickey-Fuller Test (ADF)** is the standard for this.

2.4.1 How does it work?

The test poses two hypotheses:

- **H_0 (Null Hypothesis):** The series has a unit root, meaning it is NOT stationary. There is a trend or unstable behavior.
- **H_1 (Alternative Hypothesis):** The series IS stationary.

The test computes a **p-value**: a number between 0 and 1 that indicates the probability of observing these data if H_0 were true. The decision rule is:

- If $p < 0.05$: We reject H_0 → The series is stationary.
- If $p \geq 0.05$: We cannot reject H_0 → The series is not stationary.

2.4.2 Results on our data

We apply the ADF test to each version of the series:

1. Original Series (Raw Passengers): $p = 0.99$. Very far from 0.05. The series is clearly non-stationary. The test confirms what we were seeing: there is a trend.

2. Growth Rate (Log + Differencing): $p = 0.07$. Better, but we still do not pass the 0.05 threshold. Why? Because we removed the trend, but the seasonality (the 12-month annual cycle) is still present and the test detects it as residual non-stationarity.

3. Seasonally Adjusted Growth (Log + Diff + Seasonal Diff): $p = 0.0002$. Now we are talking! By applying a second differencing (this time with lag 12 to remove the annual cycle), the series becomes completely stationary.

2.5 Conclusion

We have learned that to model economic time series, we almost never use the raw data. The transformation path is:

$$\text{Passengers} \xrightarrow{\text{Log}} \log(\text{Passengers}) \xrightarrow{\Delta_1} \text{Growth Rate} \xrightarrow{\Delta_{12}} \text{Stationary Series}$$

Each step solves a specific problem:

- **Logarithm:** Stabilizes the variance (heteroscedasticity).
- **Differencing (Δ_1):** Removes the trend.

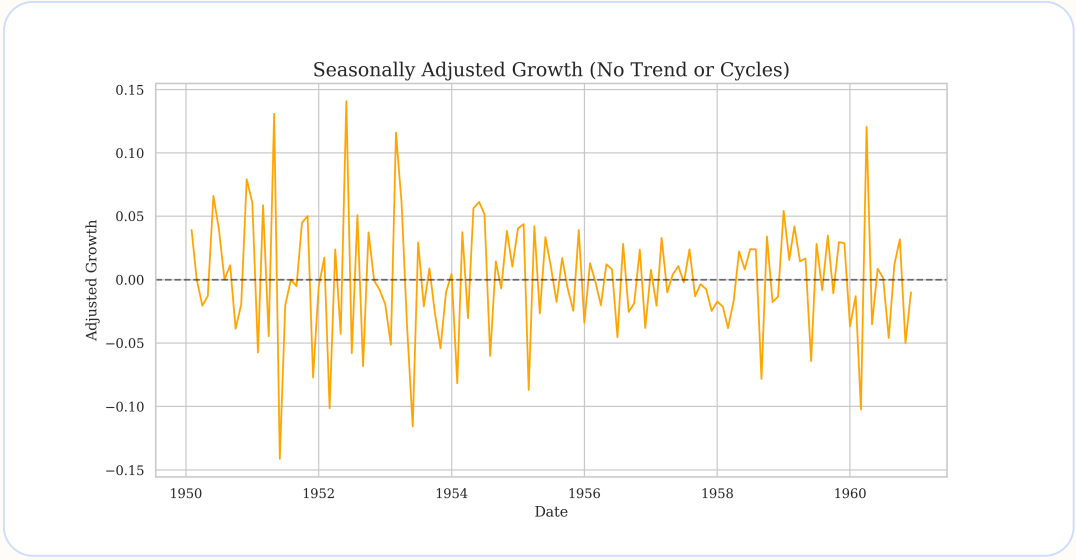


Figure 2.4: Final series after Log + Differencing + Seasonal Differencing (lag 12). This is pure stationary "noise": no trend, no predictable cycles. This is what we need to feed an ARIMA model in Chapter 3.

- **Seasonal Differencing (Δ_{12}):** Removes the annual cycle.

Now our series is stable and predictable. In the next chapter, we will use **ARIMA** to model this transformed series and generate predictions for the future.

CHAPTER 3

The classical arsenal: ARIMA

3.1 Introduction

This chapter covers the most fundamental family of models in time series: the **Box-Jenkins models**, culminating in SARIMA. These models are transparent structures built on the idea that the future is a linear combination of the past.

We will use the classic AirPassengers dataset (monthly airline passengers, 1949–1960) throughout this chapter. The goal is practical: understand when and how to use these models, and their limitations. For the full mathematical details (lag operator notation, formal stationarity/invertibility conditions, Yule-Walker equations), see the specialized literature (e.g., Hyndman & Athanasopoulos, *Forecasting: Principles and Practice*).

3.2 The building blocks: AR and MA

Every ARIMA model is built from two simple mechanisms:

3.2.1 AR: the past predicts the future

An **Autoregressive** model of order p predicts today's value as a linear combination of its own past values:

$$X_t = c + \phi_1 X_{t-1} + \phi_2 X_{t-2} + \cdots + \phi_p X_{t-p} + \epsilon_t \tag{3.1}$$

The coefficients $\phi_1, \ldots, \phi_p$ capture how strongly the series "remembers" its recent past. For the model to be stable (stationary), the influence of the past must fade over time rather than amplify. Consider AR(1): if $\phi_1 = 0.8$, a shock of 10 units fades: 10 → 8 → 6.4 → 5.1, until it disappears. But if $\phi_1 = 1.1$, the same shock amplifies: 10 → 11 → 12.1 → 13.3, and the series drifts away from its mean indefinitely. For AR(1), the condition is $|\phi_1| < 1$. For higher-order models, the condition involves the roots of the characteristic polynomial (in practice, the software verifies this automatically).

ACF/PACF signature: The PACF cuts off at lag p; the ACF decays gradually.

3.2.2 MA: shocks ripple through time

A **Moving Average** model of order q predicts today's value based on past random shocks:

$$X_t = \mu + \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \cdots + \theta_q \epsilon_{t-q} \tag{3.2}$$

Unlike AR, the memory of an MA process is strictly limited: a shock at time t has zero effect after q periods.

ACF/PACF signature: The ACF cuts off at lag q; the PACF decays gradually.

3.2.3 Why both matter

Real series typically have both persistence (AR) and shock effects (MA). Combining them gives ARMA(p, q), which can explain complex dynamics with fewer parameters than either component alone. When both ACF and PACF decay gradually without clear cutoffs, an ARMA model is likely needed — in practice, use AIC/BIC or `auto_arima` to find the right (p, q).

3.3 Quick identification reference

ACF/PACF signature	Model	How to determine order
PACF cuts off at lag p, ACF decays	AR(p)	p = PACF cutoff lag
ACF cuts off at lag q, PACF decays	MA(q)	q = ACF cutoff lag
Both decay gradually	ARMA(p, q)	Use AIC/BIC grid search
Spikes at seasonal lag s	Add seasonal terms $(P, D, Q)_s$	

Table 3.1: Model identification via ACF and PACF patterns.

3.4 ARIMA: handling trend

AR and MA assume the series is stationary — its mean does not change over time. Real-world data usually has a trend.

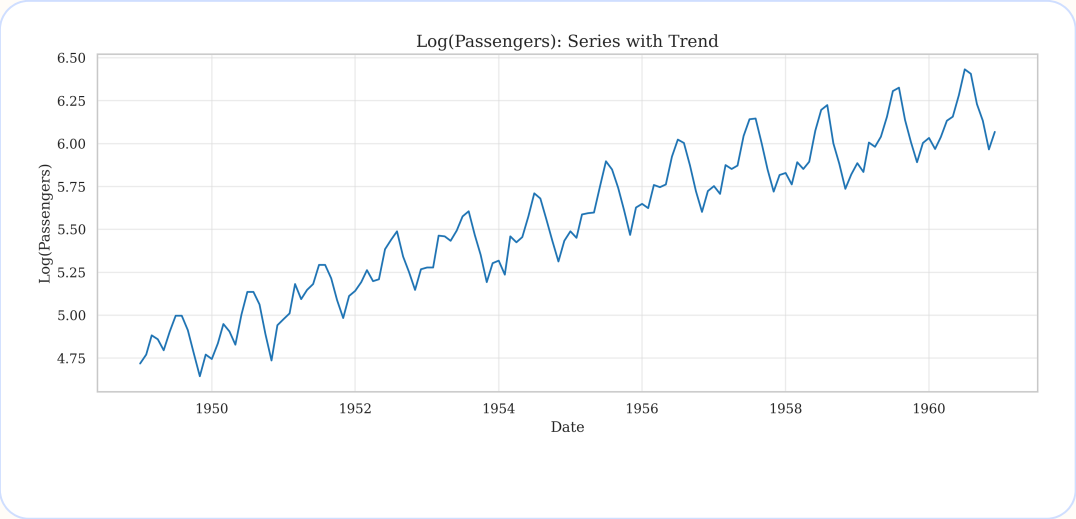


Figure 3.1: Log-passengers with an approximately linear trend. This series is non-stationary — applying ARMA directly would fail.

The solution is **differencing**: instead of modeling X_t, we model the change $\nabla X_t = X_t - X_{t-1}$. If the trend is linear, the first difference will be stationary.

An **ARIMA(p, d, q)** model applies ARMA to the series differenced d times. In practice, d = 1 handles linear trends and d = 2 handles quadratic trends (rare).

3.5 SARIMA: adding seasonality

Airline passengers peak in summer and dip in winter — a cycle that repeats every s = 12 months. ARIMA alone cannot capture this.

Seasonal differencing removes the seasonal cycle:

$$\nabla_s X_t = X_t - X_{t-s} \tag{3.3}$$

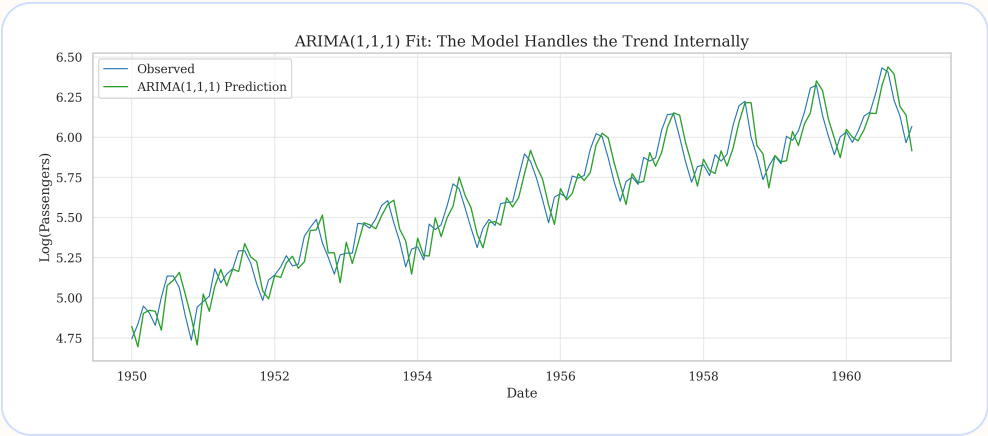


Figure 3.2: ARIMA(1,1,1) fit. The model handles the trend by differencing internally.

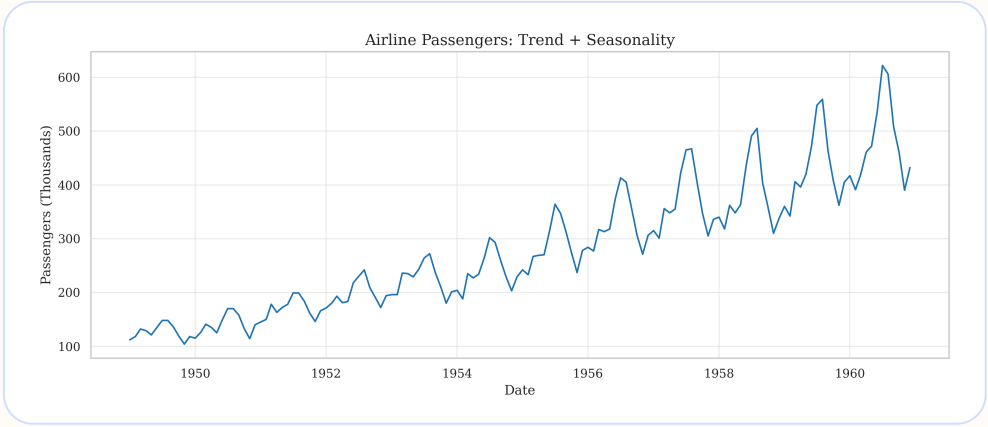


Figure 3.3: Data with trend and seasonality. Note the repeating wave pattern every 12 months.

A **SARIMA(p, d, q)(P, D, Q)$_s$** model combines:

- **Non-seasonal parts (p, d, q):** short-term dynamics (lags 1, 2, …).
- **Seasonal parts (P, D, Q)$_s$:** patterns repeating every s periods (lags s, 2s, …).

Seasonal identification: A spike in the ACF at lag s suggests a Seasonal MA term; a spike in the PACF at lag s suggests a Seasonal AR term.

3.6 ARIMAX: adding external variables

So far, we have used only the series' own past. An **ARIMAX** model adds external regressors (price, promotions, temperature):

$$\mathrm{Yt} = \beta 0 + \beta 1 \mathrm{Z1,t} + \cdots + \beta \mathrm{k} \mathrm{Zk,t} + \mathrm{ut}, \quad \text{where ut} \sim \mathrm{ARIMA} \tag{3.4}$$

Key caveat: To forecast Y_{T+h}, you need future values of Z_{T+h}. If those are unknown, you must forecast them first — adding uncertainty. For problems with many exogenous variables, the ML approach in Chapter 4 (XGBoost/Boosting) is often more practical.

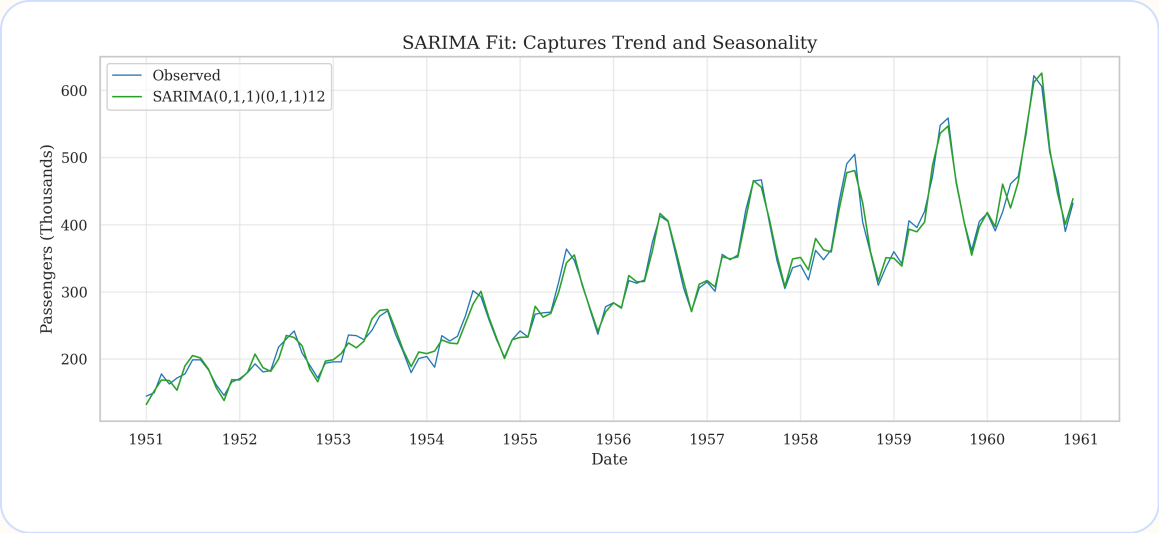


Figure 3.4: SARIMA(0, 1, 1)(0, 1, 1)$_{12}$ fit (the classic "Airline Model"). Captures both the upward trend and seasonal oscillations.

3.7 The practical shortcut: `auto_arima`

Manually inspecting ACF/PACF to choose (p, d, q)(P, D, Q)$_s$ works but is slow and subjective. In practice, `pmdarima.auto_arima()` automates this by searching over candidate models and minimizing AIC. It is a reliable starting point that often matches or outperforms manual identification.

3.8 Diagnostic checking

After fitting, verify that the model has extracted all the information from the data. If it has, the residuals should look like **white noise**: random, uncorrelated, and approximately normal.

Ljung-Box test: Tests whether the residuals have significant autocorrelation. If the p-value is high (> 0.05), the residuals are consistent with white noise and the model is valid.

Model selection: When multiple models pass diagnostic checks, choose using Information Criteria:

$$\mathrm{AIC} = -2\log L + 2k, \qquad \mathrm{BIC} = -2\log L + k\log(T) \tag{3.5}$$

Lower values indicate a better trade-off between fit and complexity. When in doubt, prefer the simpler model (parsimony).

3.9 Conclusion

2. MA: Captures shock propagation (effects of past errors).
The SARIMA framework builds from four components:

1. AR: Captures memory (persistence from past values).
3. I: Handles trend (via differencing).
4. S: Handles seasonality (via seasonal differencing).
SARIMA remains a strong baseline against which all other models should be compared. Its main advantages are interpretability and efficiency on univariate series with clear seasonal patterns.

Practical limitations:

- Forecasts degrade rapidly beyond 2–3 seasonal cycles.
- Handling many exogenous variables is awkward — prefer XGBoost (Chapter 4).
- One model per series — does not scale to hundreds of series (see Global Models, Chapter 7).

For the full mathematical treatment of the concepts introduced in this chapter, see the specialized literature (e.g., Hyndman & Athanasopoulos, *Forecasting: Principles and Practice*).

CHAPTER 4

Time series as supervised regression

4.1 Introduction: breaking the paradigm

In the previous chapters, we treated the time series as a sacred sequence where t depends on t − 1. Machine Learning models (such as XGBoost, Random Forest, or Neural Networks) do not understand sequences. For them, each row is independent.

How do we then use these powerful algorithms? **We transform the problem.** We convert a time series into a classic regression table (X → Y).

4.2 The case study: a large retailer

For this challenge, we need more complex data than airline passengers. We have simulated the daily sales of a large retailer department over 5 years.

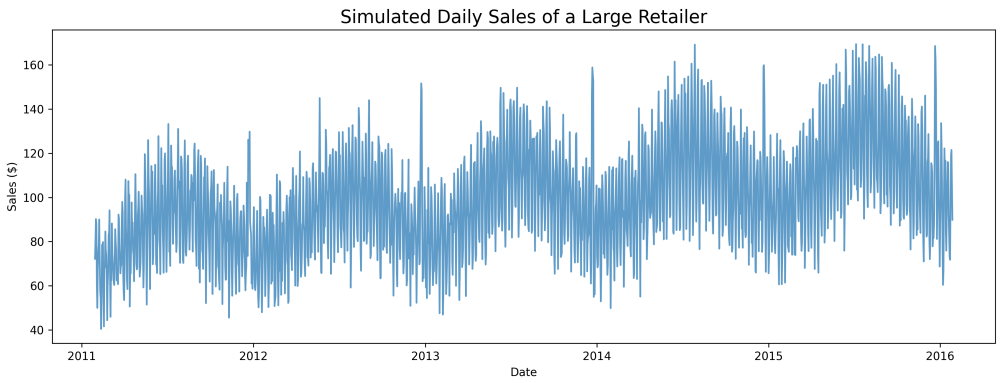


Figure 4.1: Simulated daily sales of a large retailer. Note the weekly seasonality (sawtooth pattern), annual seasonality (waves), and the Christmas spikes.

4.3 Feature engineering: teaching time to the machine

If we only give XGBoost the date, it will not learn anything useful. We need to create columns ("features") that explain the temporal context.

4.3.1 1. Calendar features

We extract explicit information from the date:

- **Day of the week:** Is it Monday or Saturday? (Crucial for retail).
- **Month:** Is it December? (Christmas).
- **Weekend:** Is it Saturday or Sunday?

4.3.2 2. Lags

We create columns with past values:

- `lag_1`: How much did I sell yesterday?
- `lag_7`: How much did I sell on the same day last week?
- `lag_28`: How much did I sell 4 weeks ago? (captures monthly cycles).
- `lag_365`: How much did I sell on this day last year?

4.3.3 3. Rolling windows

They capture the recent trend:

- `rolling_mean_7`: Average sales over the last week.
- `rolling_std_7`: Sales variability over the last week.

4.3.4 The resulting dataset

After applying this process, each row in our dataset is no longer just a date and a sale. It is a complete "snapshot" of the temporal context:

Sales	dow	month	lag_1	lag_7	lag_365	roll_mean_7	weekend
132.5	3	6	118.2	130.1	125.8	127.4	0
155.8	4	6	132.5	148.3	150.2	129.1	0
178.3	5	6	155.8	170.6	172.1	133.5	1
165.1	6	6	178.3	160.2	158.7	138.2	1
120.4	0	6	165.1	122.8	118.5	141.7	0

Table 4.1: Example of the dataset with temporal features. Each row is a "snapshot" that XGBoost can understand: it knows what day of the week it is, how much was sold yesterday, a week ago, and a year ago.

Now XGBoost can learn rules like: *"If it is Saturday (dow=5) and yesterday's sales were high (lag_1 > 150), then today's sales will be high"*.

4.4 The boosting family: not just XGBoost

Throughout this chapter we use XGBoost, but it is one member of a larger family. The three main gradient boosting libraries for time series are:

A notable detail: CatBoost uses **ordered boosting**, which respects the temporal order of data during training. This makes it naturally suited for time series, as it avoids a subtle form of data leakage present in standard boosting when applied to sequential data.

In practice, start with LightGBM (fastest iteration cycle) and switch to CatBoost if you have many categorical features that benefit from native encoding.

4.5 Advanced feature engineering

The basic features (calendar, lags, rolling windows) from the previous section cover most cases. But for complex problems, three additional techniques can make a significant difference:

Model	Key strength	Best use case
XGBoost	Robust, well-documented, large community	General-purpose default
LightGBM	Fastest training, leaf-wise growth, low memory	Large datasets, global models (Ch 7). Was the backbone of winning solutions in the M5 Kaggle competition
CatBoost	Native categorical handling, ordered boosting	Datasets with many categorical features (product, store, region)

Table 4.2: Gradient boosting family comparison. All three accept the same feature engineering; the difference is in training speed and how they handle specific data types.

Fourier features capture complex seasonality. When a series has multiple overlapping cycles (daily + weekly + annual), encoding them as sine/cosine pairs at different frequencies lets the model learn each cycle independently:

$$\sin\left(\frac{2\pi k \cdot t}{T}\right), \cos\left(\frac{2\pi k \cdot t}{T}\right) \quad \text{for } k = 1, 2, \ldots$$

This is particularly useful for hourly data with daily and weekly patterns, where calendar features alone are too coarse.

Target encoding for categorical variables replaces a category (e.g., "Store A") with the historical mean of the target for that category. This gives the model a numerical "baseline" for each group. **Caution:** always compute the encoding using only past data to avoid leakage.

Interaction features combine two features into one: `Monday_Electronics` captures the fact that Monday sales patterns differ across product categories. These interactions are things that tree models can discover on their own, but providing them explicitly speeds up learning and improves accuracy on smaller datasets.

A recent insight from the literature: **feature engineering matters more than model architecture**. A well-engineered LightGBM consistently outperforms a poorly featured deep learning model, even for intermittent demand.

4.6 Multi-step forecasting strategies

When you need to forecast $h > 1$ steps ahead, you face a design choice:

- **Recursive:** Train one model for h = 1. To predict h = 2, feed the h = 1 prediction back as a lag. Simple, but errors accumulate with each step.
- **Direct:** Train h separate models, one per horizon. No error accumulation, but more models to maintain and no information sharing between horizons.
- **Hybrid (Rectify):** Combine both: use recursive predictions as features for a direct model. Recent research shows hybrid strategies outperform either alone in the majority of cases.

Libraries like `skforecast` support all three strategies out of the box, making it easy to experiment and choose the best one for your data.

4.7 The duel: XGBoost vs ARIMA

4.7.1 Why compare?

In Chapters 2 and 3, we invested a lot of effort in transforming the series (logarithms, differencing) and configuring SARIMA. Was all that theory worth it, or can a "brute force" ML algorithm with good features do the same (or better)?

This question is fundamental in practice: in a company, the Data Scientist's time is limited. If XGBoost with manual features achieves results comparable to SARIMA without needing to understand stationarity, many teams will prefer the ML route.

We trained two models on the same data:

1. **XGBoost:** With all the features from Table 4.1.
2. **SARIMA:** Configured to capture the weekly cycle (7 days), extending the seasonal logic from Chapter 3 to daily data.

4.7.2 How do we measure success?

To declare a winner, we need an impartial referee. We will use the **MAE (Mean Absolute Error)**:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

It is simply the average of our errors in absolute value. If I sell 100 and predicted 110, my error is 10. The MAE tells us, on average, how many units we are off each day. It is easy to interpret for any manager.

4.7.3 Visual results

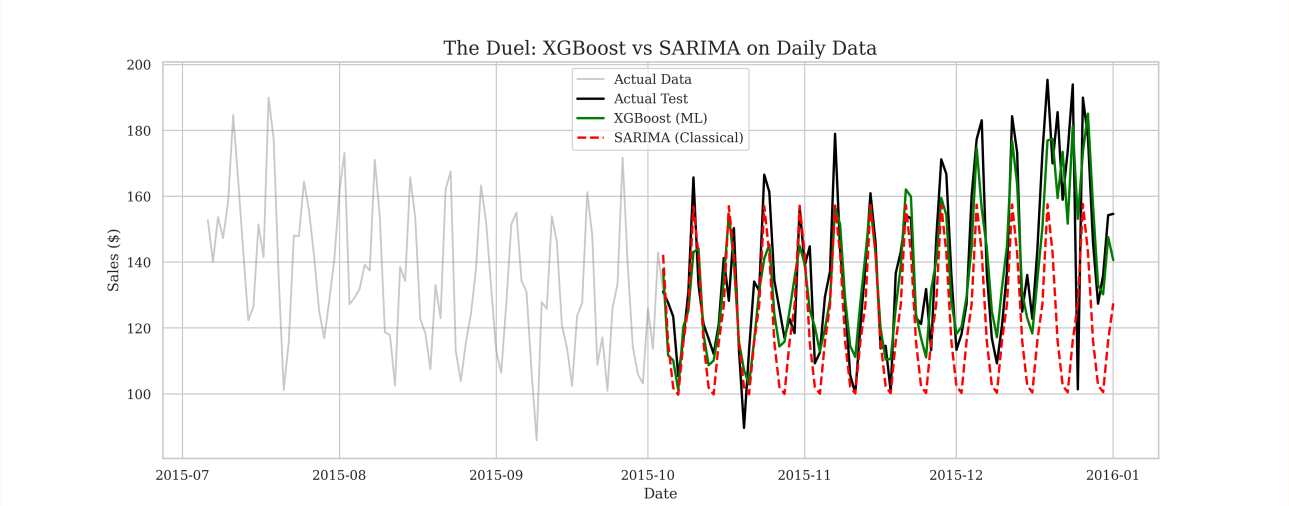


Figure 4.2: Forecast Comparison over the last 3 months. The black line is the actual data, green is XGBoost, and red is SARIMA. Pay special attention to the late December (Christmas) zone.

4.7.4 Results analysis

Let us examine Figure 4.2 in detail:

- **XGBoost (Green):** It fits the actual data very well throughout the entire test period. It captures the pre-Christmas shopping spike (December 21–24) and the drop on the 25th (store closed). Why? Primarily thanks to `lag_365`, which lets it "remember" last year's Christmas pattern, combined with calendar features like `month` and `day_of_year` that provide additional context.
- **SARIMA (Red):** It correctly captures the weekly rhythm (rises and falls every 7 days), but **completely fails at Christmas**. It keeps repeating the average weekly pattern without understanding that December is special. The classical model is "blind" to special events unless we use advanced techniques such as Dynamic Regression.

Error Metrics (MAE):

- XGBoost: **10.29** (on average, it is off by $10.29 per day)
- SARIMA: **18.19** (nearly double the error)

XGBoost cuts the error in half. The difference is not marginal: in a retail context with thousands of products, this translates into millions of dollars less in wasted inventory.

4.7.5 Which features were most important?

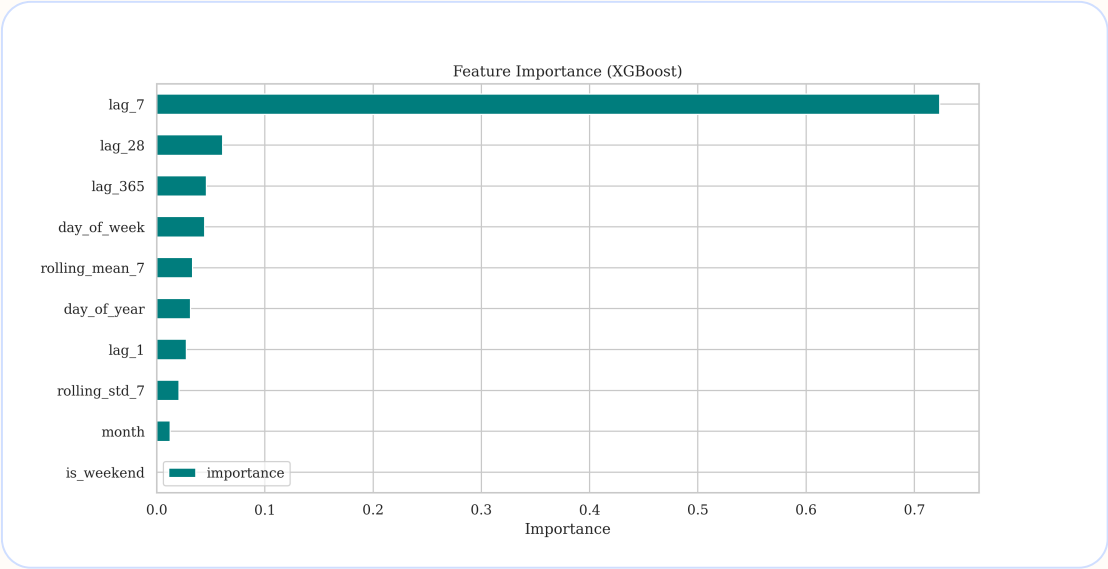


Figure 4.3: Importance of each feature for XGBoost. The weekly lag (lag_7) dominates overwhelmingly, followed by lag_28 and lag_365. The weekly cycle is the strongest signal in this retail data.

4.7.6 Beyond feature importance: SHAP

The built-in feature importance (Figure 4.3) tells us which features matter globally. But **SHAP (SHapley Additive exPlanations)** goes further: it explains how each feature contributed to a specific prediction. For example, SHAP can show that for December 24th, `lag_365` pushed the prediction up by 40 units because last year's Christmas Eve was also high. This level of explainability is essential when a stakeholder asks "why did the model predict this?"

4.8 Hyperparameter essentials

Three hyperparameters dominate the performance of any boosting model:

- `learning_rate` (0.01–0.3): How much each tree corrects the previous one. Lower values learn more slowly but generalize better.
- `n_estimators` (100–3000): Number of trees. More trees with a low learning rate is generally better, but increases training time.
- `max_depth` (3–8): Depth of each tree. Deeper trees capture more complex interactions but risk overfitting.

Rule of thumb: Start with `learning_rate=0.05`, `max_depth=5`, and `n_estimators=1000` with early stopping. The early stopping mechanism monitors validation error and stops adding trees when performance plateaus, preventing overfitting automatically. For rigorous hyperparameter tuning, see the validation techniques in Chapter 5.

4.9 Selection guide: which model should I use?

This is the million-dollar question. Here is a practical guide to help you decide.

Characteristic	Use ARIMA / ETS	Use ML (XGBoost)
Frequency	Low (Monthly, Quarterly)	High (Daily, Hourly)
Seasonality	Simple (Annual only)	Complex (Multiple cycles)
External Variables	Hard to include	Very easy (Price, Weather)
Number of Series	Few (< 100)	Many (> 1000)
Horizon	Short Term	Short and Medium Term
Explainability	High (Coefficients)	Medium (Feature Importance)

Table 4.3: Guide for choosing between Classical Statistics and Machine Learning.

4.10 Conclusion

We have shown that for high-frequency data (daily) with multiple patterns (weekly + annual) and special events (Christmas), Machine Learning outperforms classical methods.

But the key is not the algorithm itself, but the **Feature Engineering**: the ability to translate our business knowledge (calendars, lags, rolling windows) into columns that the model can understand. Without good features, XGBoost will be just as blind as ARIMA when facing Christmas.

In the next chapter, we will learn to properly validate these models with **Backtesting**, to make sure we are not fooling ourselves.

CHAPTER 5

Validation and error metrics

5.1 Introduction: the danger of the "lucky split"

Imagine you are a student and the professor tells you: "The final exam will cover topic 4". You study only topic 4 and score a perfect 10. Do you really know the subject? Probably not. You got lucky.

In Time Series, doing a single Train/Test split (as we did in the previous chapters) is risky.

- If your Test falls on a quiet month (e.g., July), your model will look like a genius.
- If your Test falls on a crisis (e.g., Pandemic or Christmas), your model will look useless.

In our experiment, we saw that the error (MAE) changed drastically:

- Evaluating in July: MAE = **8.89**
- Evaluating in December: MAE = **18.42**

More than double the error just by changing the evaluation date.

5.2 The solution: backtesting

To avoid fooling ourselves, we must evaluate the model multiple times at different points in the past. We use an **Expanding Window** technique:

1. We train with data up to March → We predict April.
2. We train with data up to April → We predict May.
3. We train with data up to May → We predict June.

At the end, we average the errors across the 5 folds. This average is a much more honest estimate of how the model will perform in the real future.

5.3 Error metrics: how do we measure success?

We already have the predictions, but how do we measure how "close" we were? There are several metrics, each with its advantages and limitations. Let us go through them with a concrete example.

Suppose that over 5 days the actual sales were [100, 150, 80, 0, 200] and our model predicted [110, 140, 95, 10, 180].

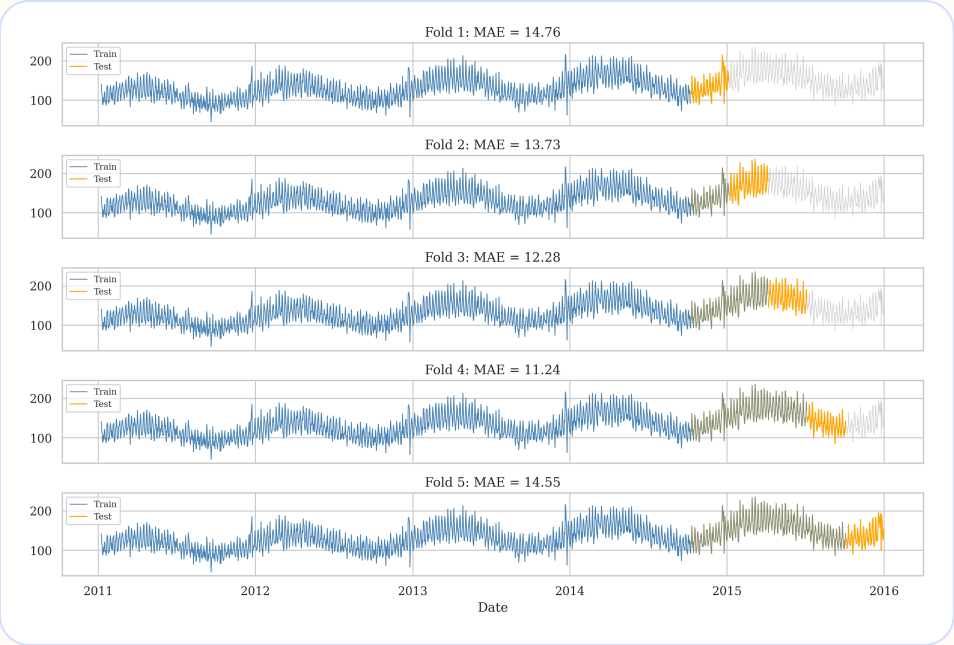


Figure 5.1: Backtesting with 5 Folds. Each panel is an independent evaluation. The model must achieve a good result in all of them, not just one.

5.3.1 1. MAE (mean absolute error)

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

Let us calculate step by step with our example:

$$\text{MAE} = \frac{|100-110| + |150-140| + |80-95| + |0-10| + |200-180|}{5} = \frac{10+10+15+10+20}{5} = \frac{65}{5} = 13$$

Interpretation: "On average, we are off by $13 each day". It is the most intuitive metric: any manager understands it without needing formula explanations. It is robust because it does not disproportionately penalize large errors.

5.3.2 2. RMSE (root mean squared error)

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

With the same example:

$$\text{RMSE} = \sqrt{\frac{10^2+10^2+15^2+10^2+20^2}{5}} = \sqrt{\frac{100+100+225+100+400}{5}} = \sqrt{185} = 13.6$$

Interpretation: Note that the RMSE (13.6) is greater than the MAE (13.0). It always will be, because the RMSE penalizes large errors by squaring them. The error of 20 on day 5 "weighs" more in the RMSE than in the MAE. Use RMSE when a large error is catastrophic (e.g., if under-predicting means shutting down a production line).

5.3.3 3. MAPE (mean absolute percentage error)

$$\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \times 100$$

With our example:

$$\text{MAPE} = \frac{\frac{10}{100} + \frac{10}{150} + \frac{15}{80} + \frac{10}{0} + \frac{20}{200}}{5} \times 100$$

Problem: On day 4 the actual sale was 0. The division $\frac{10}{0} = \infty$. The MAPE blows up to infinity. This happens constantly in retail with slow-moving products (as we will see in Chapter 10). For this reason, **never use MAPE if your series contains zeros**.

5.3.4 4. WMAPE (weighted MAPE)

$$\text{WMAPE} = \frac{\sum |y_i - \hat{y}_i|}{\sum y_i} \times 100$$

With our example (excluding the zero problem):

$$\text{WMAPE} = \frac{10 + 10 + 15 + 10 + 20}{100 + 150 + 80 + 0 + 200} \times 100 = \frac{65}{530} \times 100 = 12.3\%$$

Interpretation: "Our total error amounts to 12.3% of total sales". The advantage of WMAPE is that it weights by volume: an error of 10 on a product that sells 200 "weighs" less than an error of 10 on a product that sells 20. It is the favorite metric in the Retail sector.

5.4 Error diagnostics: from residuals to features

It is not only important how much we are off, but when and why. The model's residuals (errors) tell a story.

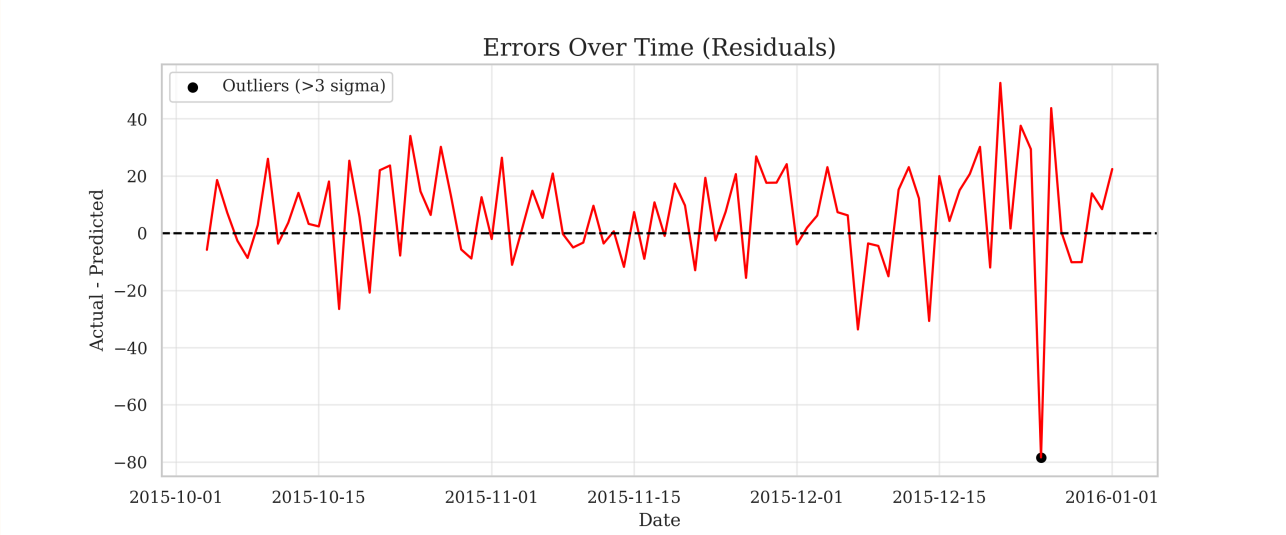


Figure 5.2: Residuals over time. The red line mostly oscillates around zero, but note the growing positive bias toward December — the model systematically under-predicts the Christmas season. The black dot is an "Outlier": an unusually large error that exceeds 3σ.

5.4.1 How to read this chart?

- **If the residuals oscillate around zero with no patterns:** The model is capturing the signal well. The errors are random "noise".
- **If there are visible patterns** (e.g., errors always positive on Fridays): The model is systematically failing at something. There is information it did not capture.
- **If there are black dots (outliers):** These are days where the error was abnormally large. They typically correspond to events the model does not know about.

5.4.2 From residuals to features: capitalizing on errors

Outliers are not just "annoying noise". They are an opportunity to improve the model.

The process is iterative:

1. **Identify the outliers:** On which days did we have enormous errors?
2. **Investigate the cause:** Was there a promotion? A holiday? A transport strike? Bad weather?
3. **Create a feature:** If we discover that the large errors coincide with promotions, we create a column `is_promotion` (0 or 1) and add it to the model from Chapter 4.
4. **Re-train and re-evaluate:** The model now "knows" that promotions exist. The outliers should decrease.

This cycle of error → investigation → feature → improvement is the heart of a Data Scientist's daily work in forecasting. A model is never "finished"; it can always be improved by incorporating more business knowledge.

5.5 Conclusion

Now we have a robustly validated model (Backtesting) and we know exactly how much to trust it (Metrics).

Final Results from our Backtesting:

- Average MAE: **13.31**
- Average MAPE: **8.87%**

We are ready to deploy this model. But a point prediction alone can be dangerously misleading. In the next chapter, we will learn to quantify the uncertainty of our predictions with **Confidence Intervals, Bayesian Forecasting, and Conformal Prediction**.

CHAPTER 6

Navigating uncertainty

6.1 Introduction: the weather forecast

When a meteorologist appears on television, they don't say "Tomorrow it will be 25 degrees". They say: "Tomorrow it will be between 23 and 27 degrees, with an 80% probability".

Why? Because they know that predicting the exact future is impossible. The honest thing is to communicate how much uncertainty their prediction has.

In previous chapters, we have built models that give us a point estimate: "January sales will be 450". But in real life, an operations manager needs to know: Could they be 400? Could they be 550? How much stock should I order so I neither run out nor overbuy?

In this chapter, we will explore three ways to quantify that uncertainty.

6.2 Classical confidence intervals

6.2.1 What is a confidence interval?

A confidence interval (CI) is a range around the point prediction that indicates where we expect the true value to fall with a certain probability. For example:

- **80% CI:** We expect the true value to fall within this range 80% of the time.
- **95% CI:** A wider range, with greater "certainty".

The relationship between confidence and width is direct: the more certainty you want, the wider the interval will be. A 99% CI is so wide that it almost always gets it right, but it is of little practical use ("sales will be between 0 and 1 million").

6.2.2 How are they calculated?

For an ARIMA model, the confidence interval at the 100(1 − α)% level is calculated as:

$$\hat{Y}_{t+h} \pm z_{\alpha/2} \cdot \sigma_h$$

where:

- $\hat{Y}_{t+h}$ is the point prediction at horizon h.
- $z_{\alpha/2}$ is the quantile of the standard Normal distribution (1.96 for 95%, 1.28 for 80%).
- σ_h is the standard deviation of the prediction error at horizon h.

Numerical example: If our model predicts 450 passengers for the next month ($\hat{Y} = 450$), with a prediction error standard deviation of $\sigma = 30$, the 95% CI would be:

$$450 \pm 1.96 \times 30 = 450 \pm 58.8 = [391.2, 508.8]$$

That is, we expect between 391 and 509 passengers with 95% confidence.

6.2.3 The assumptions

This elegant formula has conditions that must be met:

1. **Normality:** The model errors must follow a Normal distribution (Gaussian bell curve). If the errors are asymmetric or have heavy tails, the intervals will be incorrect.

2. **Homoscedasticity:** The variance of the errors must be constant over time. If the errors are larger in certain periods (as we saw in Chapter 1 with heteroscedasticity), the intervals do not adapt.

3. **Independence:** The errors must not be correlated with each other. If today's error predicts tomorrow's, the intervals underestimate the uncertainty.

Each model calculates its intervals differently. ARIMA assumes normal errors. XGBoost does not generate native intervals (it needs additional techniques such as Conformal Prediction, which we will see later). Bayesian models generate complete distributions without assuming normality.

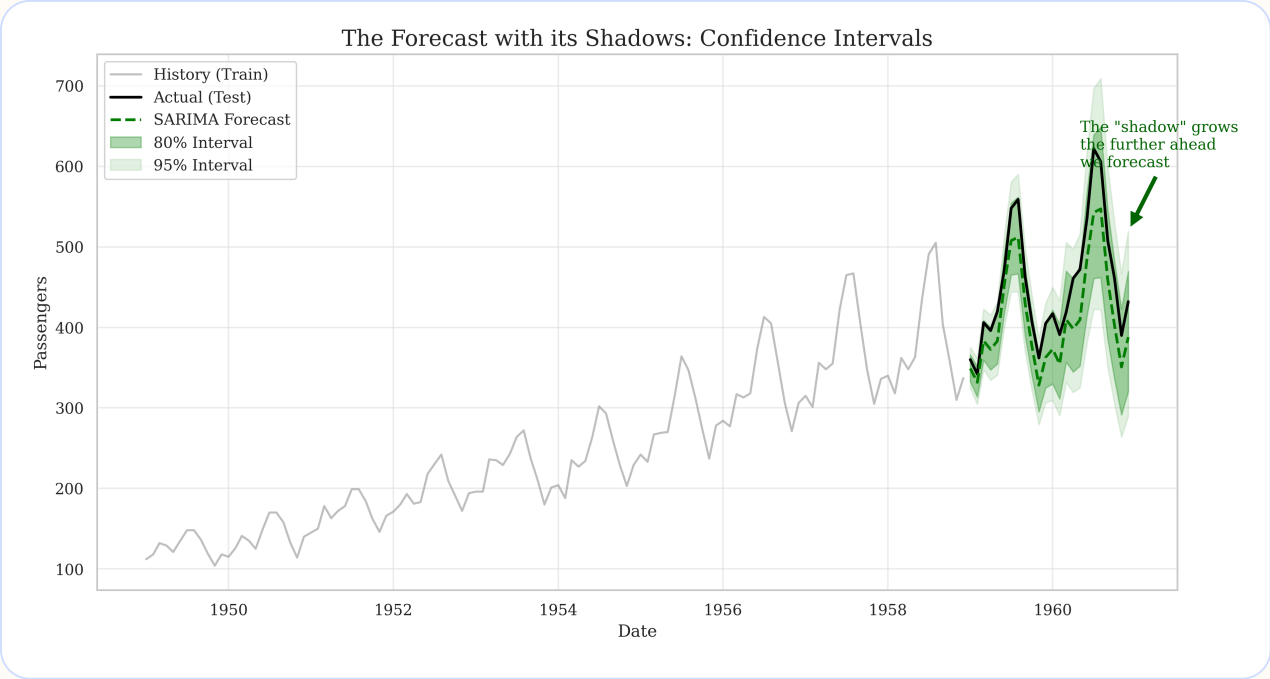


Figure 6.1: SARIMA Prediction with 80% and 95% Confidence Intervals. Note how the bands widen the further out we predict: we are more certain about next month than about next year.

6.2.4 Interpretation on the logarithmic scale

This has an important consequence: the intervals become **asymmetric**. If on the log scale the interval is [5.8, 6.2], transforming back we get $[e^{5.8}, e^{6.2}] = [330, 493]$. The upper limit is farther from the prediction than the lower one. This is consistent with reality: an airline can grow significantly in a good year, but it will hardly drop to zero.

In Chapter 2, we transformed the data with logarithms. This means the SARIMA model works internally on the logarithmic scale. To obtain the intervals in actual passengers, we apply the inverse function (e^x) to the interval limits.

6.2.5 The counterfactual: intervals on raw vs transformed data

What happens if we ignore the assumptions? Let's compare the intervals of a SARIMA on untransformed data vs log-transformed data:

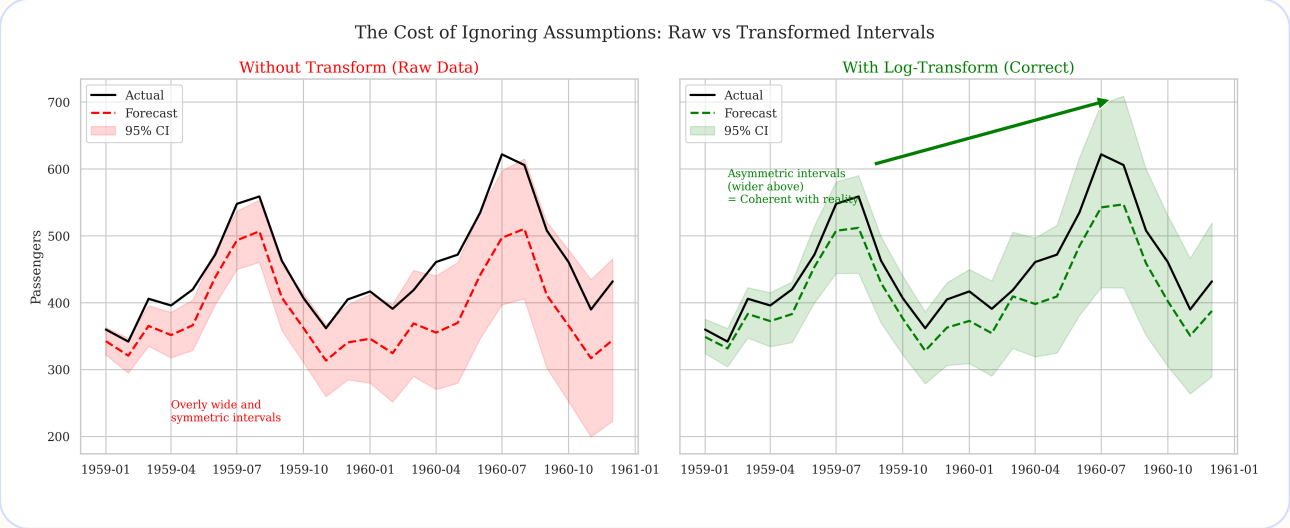


Figure 6.2: Left: Untransformed. The intervals are symmetric and overly wide, failing to capture the actual data pattern. **Right:** With log-transform. The intervals are asymmetric and consistent with reality.

The results are revealing:

- **Raw Data:** The 95% CI only contains 54% of the actual values. The intervals are symmetric and poorly calibrated: the non-constant variance makes them too wide in some segments and too narrow in others.
- **Log-Transform:** The 95% CI contains 100% of the actual values. By transforming, the assumptions are better satisfied and the intervals are reliable.

Lesson: Classical intervals are only as good as the assumptions that support them. Without the logarithmic transformation from Chapter 2, the intervals provide incorrect information.

6.3 Bayesian forecasting: the complete distribution

6.3.1 What is the Bayesian approach?

Imagine you want to predict next month's sales.

- **The Classical Approach:** "Sales will be 450" (a single number, with an interval around it).
- **The Bayesian Approach:** "There is a 60% probability they will be between 420-480, a 25% probability they will be between 380-420, and a 15% probability they will exceed 480" (a complete probability distribution).

The Bayesian approach does not generate a single forecast, but **hundreds of possible scenarios** of the future. Each scenario represents a "possible world" where the model parameters are slightly different, reflecting our uncertainty about them.

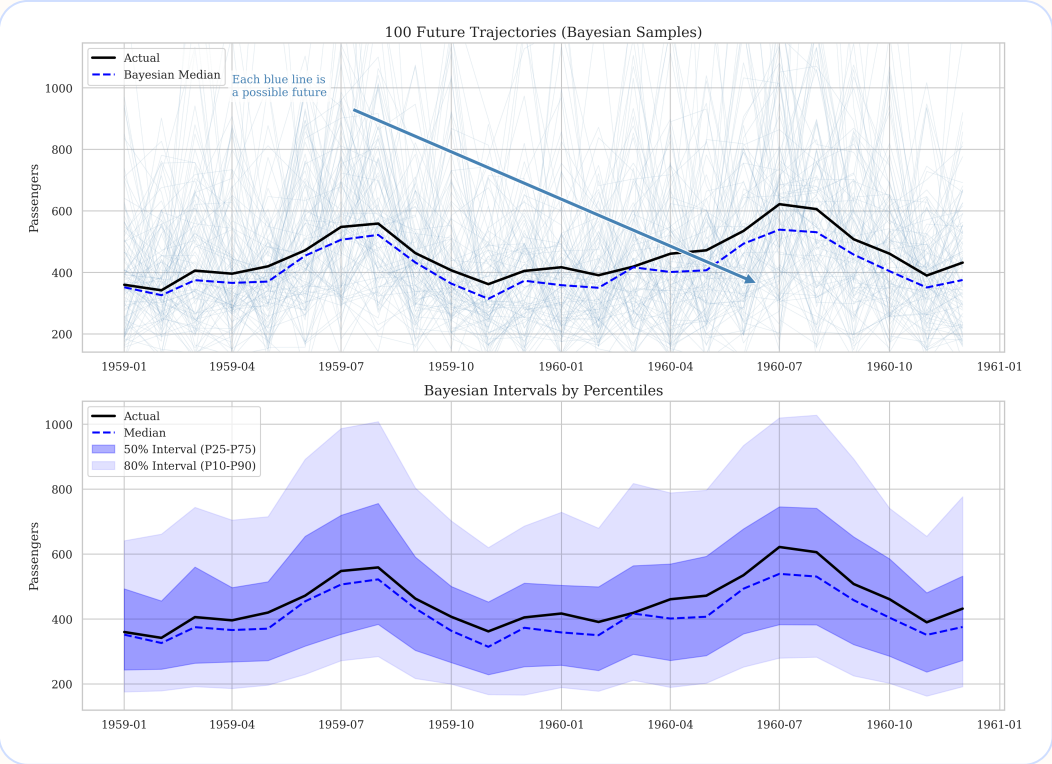


Figure 6.3: Top: 100 possible future trajectories (blue lines). The spread of these lines visualizes the uncertainty. **Bottom:** Percentile-based intervals derived from all 500 samples.

6.3.2 Results

In Figure 6.3, the top panel shows 100 overlaid trajectories. Note how they form a "fan" that widens into the future: there is more consensus about the first months than about the last ones. The bottom panel summarizes all samples into percentile-based intervals (P25-P75 for 50%, P10-P90 for 80%).

6.3.3 Advantages of the Bayesian approach

- **Complete distribution:** You don't just have the mean and an interval; you have the entire shape of the uncertainty. This is valuable when decisions depend on extreme scenarios (e.g., "What is the probability that sales fall below 300?").
- **Incorporates prior knowledge (Priors):** You can tell the model "Sales never drop below 100" and it takes that into account. This is useful when you have little data but a lot of domain knowledge.
- **More flexible intervals:** It does not necessarily assume normality. The shape of the intervals adapts to the actual distribution of errors.

6.3.4 Limitations

- **Computational cost:** Generating hundreds or thousands of samples is slower than calculating an analytical interval. For high-frequency series (minute-level) with thousands of products, it can be prohibitive.
  Choice of Priors: If your "initial beliefs" are incorrect (e.g., you assume sales never exceed 500 and they actually do), the model can produce biased results. In practice, "weak" (uninformative) priors are typically used to let the data speak.
  When NOT to use it: If you have a single monthly series with plenty of historical data and the normality assumptions hold, a classical SARIMA with analytical intervals will be faster and equally accurate.

6.4 Conformal prediction: the modern approach

What if we could generate prediction intervals that work with **any model** (XGBoost, Neural Networks, anything) without assuming anything at all about the distribution of errors?

That is exactly what **Conformal Prediction** does.

6.4.1 The idea in 3 steps

1. Split the data into 3: Train (to train the model), Calibration (to measure errors), and Test (to evaluate).

2. Calculate absolute errors on Calibration: We predict on the Calibration set and measure $|y_i - \hat{y}_i|$ for each point.

3. Take the quantile: If we want 90% coverage, we take the 90th percentile of those absolute errors. That number is the "width" q of the band.

The final prediction for a new point is:

$$[\hat{Y} - q, \hat{Y} + q]$$

where $\hat{Y}$ is the model's point prediction and q is the quantile calculated in step 3.

6.4.2 Numerical example

Suppose our XGBoost predicted 10 points in the Calibration set, and the absolute errors were:

$$|e| = [5, 8, 12, 3, 7, 15, 6, 9, 4, 11]$$

Sorted from smallest to largest: [3, 4, 5, 6, 7, 8, 9, 11, 12, 15]. For 90% coverage, we take the 90th percentile, which is the value at position $\lceil 0.90 \times 10 \rceil = 9$: **q = 12**.

Now, if the model predicts $\hat{Y} = 150$ for a new point, the 90% conformal interval is:

$$[150 - 12, 150 + 12] = [138, 162]$$

We assume nothing about the distribution of errors. We simply use the actual observed errors.

6.4.3 The exchangeability assumption

The only assumption of Conformal Prediction is that the calibration data and the future data come from the same process. In practice, this means:

- **Works well when:** Business conditions during calibration are similar to those in the future. For example, if you calibrate with 2023 data and predict 2024, and there are no structural changes (new regulations, pandemics, etc.), the calibration errors are representative.
- **Fails when:** There is a structural change between calibration and the future. For example, if you calibrate with pre-pandemic data (2019) and predict during the pandemic (2020), the calibration errors will dramatically underestimate the actual errors, and the intervals will be too narrow.

In summary: Conformal works if "the future resembles the recent past". If there is a break, no uncertainty method (neither classical nor Bayesian) will be reliable.

6.4.4 Results: Conformal vs parametric

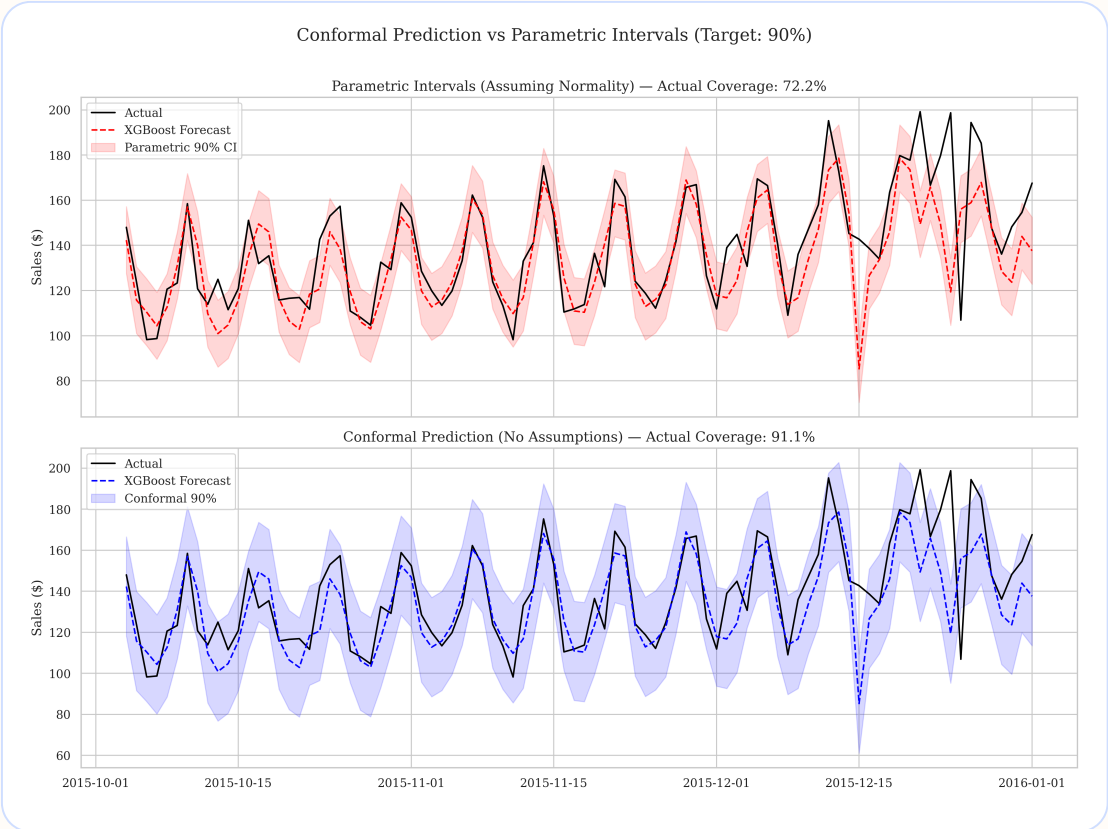
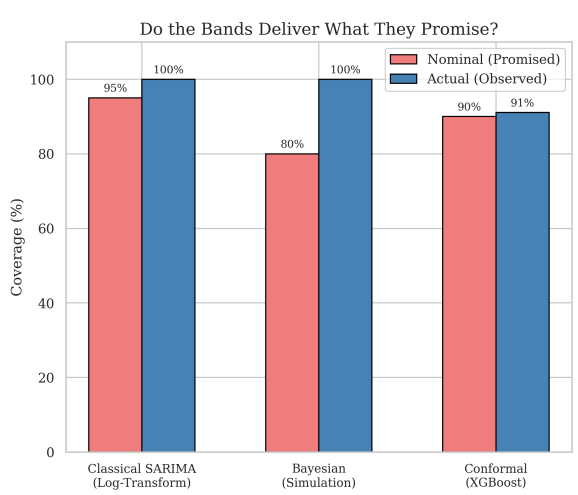


Figure 6.4: Top: Parametric intervals (assuming normality) achieve only 72% coverage when the target was 90%. **Bottom:** Conformal Prediction achieves 91%, without assuming any distribution.

The parametric intervals fail because XGBoost's errors are not normal. Conformal Prediction hits the target because it makes no assumptions: it measures real errors and uses their empirical distribution.

6.5 Comparison and selection guide



Quick Guide: When to Use Each Approach?

Method	Assumptions	Complexity	Best For
Classical SARIMA	Yes (Normality)	Low	Simple univariate series
Bayesian	Partial (Priors)	High	When you need the full distribution
Conformal	No	Low	Any model (ML, DL, etc.)

Figure 6.5: Actual vs nominal coverage of the three methods. Classical Intervals (log-transform) achieve perfect coverage; the Bayesian approach significantly exceeds the target (conservative intervals); Conformal Prediction meets the theoretical guarantee without distributional assumptions.

Method	Assumptions	Complexity	Compatible Models	Problem Type
Classical Intervals	Normality, homoscedasticity	Low	ARIMA, ETS	Univariate monthly/quarterly series with stable behavior.
Bayesian	Priors (initial beliefs)	High	Prophet, structural models	When you need the full distribution or you have substantial domain knowledge and little data.
Conformal	Only exchangeability	Low	Any (XGBoost, Neural Networks, etc.)	Daily/hourly series with ML models. Ideal when the error distribution is unknown.

Table 6.1: Selection guide: each method has its ideal context. There is no universal "best".

Practical rules:

- If you use **ARIMA or ETS** with verified assumptions (residual normality, constant variance) → The model's native prediction intervals are sufficient and computationally efficient.
- If you need the **complete distribution** of possible futures, or you want to incorporate expert knowledge (priors) → Bayesian approach. Particularly useful when there is little historical data but substantial domain knowledge.
- If you use **ML models** (XGBoost, LightGBM, Neural Networks) that do not natively generate intervals → Conformal Prediction is the most practical option. But keep in mind that basic split-conformal assumes the calibration errors are representative of the future; for series with strong autocorrelation, there are adaptive variants (ACI) that improve coverage.
- If there are **recent structural changes** (pandemic, new regulation, strategy shift) → No uncertainty method will be reliable with old data. The priority is to recalibrate with recent data, regardless of the chosen approach.

Technical note: In forecasting, what we calculate are **prediction intervals** (the range where we expect future observations to fall), not confidence intervals in the strict sense (a range for a population parameter). The distinction is subtle but relevant: prediction intervals are always wider because they include both the model's uncertainty and the inherent variability of the data.

6.6 Conclusion

A prediction without uncertainty is like a map without a scale:
technically correct, but dangerously incomplete.

1. **Classical Intervals** are elegant but fragile: they depend on the assumptions being met.
2. **Bayesian Forecasting** is the richest conceptually, but also the most costly.
3. **Conformal Prediction** is the most practical for ML models: it works without distributional assumptions.

In the next chapter, we will scale our models to thousands of series simultaneously with **Global Models**.

CHAPTER 7

Global models

7.1 Introduction: the 10,000-product problem

Imagine you are the Data Scientist at a large retailer. You have 10,000 products in your catalog. So far, we have learned how to train a model for one series (univariate). If we follow that strategy, we would have to train, maintain, and monitor 10,000 different models.

This has two serious problems:

1. **Computational Cost:** Training 10,000 models every night is slow and expensive.
2. **Wasted Information:** The model that predicts "Coca-Cola" knows nothing about what the "Pepsi" model learned. If both are soft drinks, they should share patterns (e.g., sales go up in summer).

7.2 The solution: global models

A **Global Model** is a single model trained on the data from all products at once. It is like a general practitioner who has seen thousands of patients, rather than a doctor who has only seen one patient in their entire life.

7.2.1 The data format: Panel

To train a Global Model, we need all products in a single table. This is called **Panel or Long Format**. Each row has a `Product_ID` column that identifies which product it belongs to:

Product_ID	Date	Sales	dow	lag_1	lag_7	roll_mean_7
A_BestSeller	2015-12-01	135.2	1	128.4	132.1	130.5
A_BestSeller	2015-12-02	142.8	2	135.2	138.7	131.2
B_Trending	2015-12-01	155.3	1	148.9	145.2	147.6
B_Trending	2015-12-02	162.1	2	155.3	150.8	149.3
D_Seasonal	2015-12-01	28.5	1	25.1	30.2	27.8

Table 7.1: Panel Format. All products in a single table. The features (dow, lags, rolling mean) are calculated within each product using `groupby`, to avoid mixing information across series.

The model receives this complete table and learns patterns that are common to all products ("Saturdays sell more") as well as patterns specific to each product (through the numerically encoded `Product_ID` column).

7.3 The experiment: local vs global

7.3.1 Experiment design

We simulated 4 products with very different personalities to test the hypothesis that a single model can handle all this diversity:

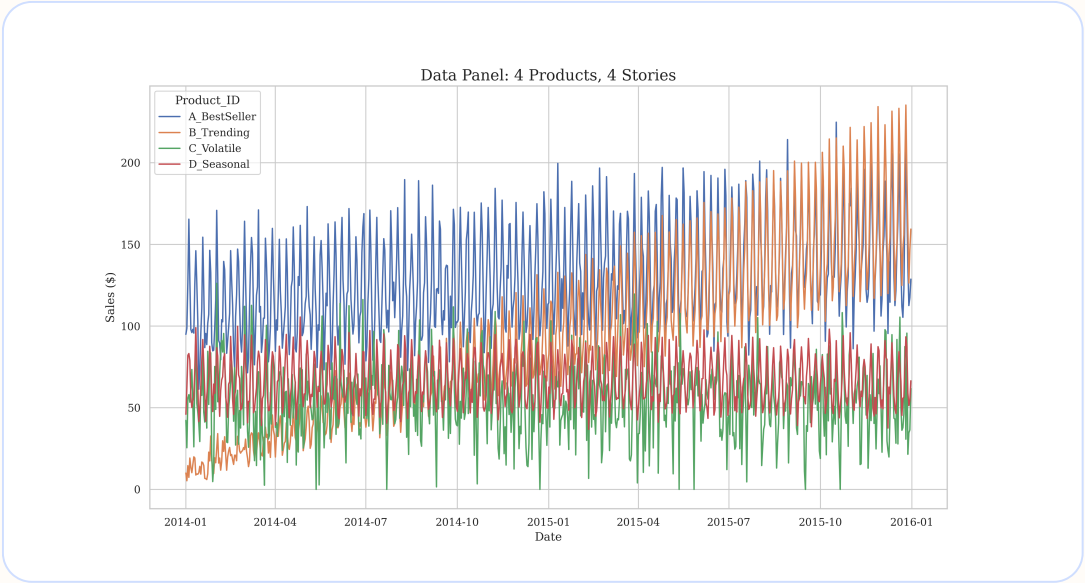


Figure 7.1: Panel Data Visualization. Four products with very different behaviors: A (blue) has high and stable sales, B (orange) grows over time, C (green) is volatile, and D (red) has pronounced seasonal spikes.

- **A (Best Seller):** High sales (~100-150/day) and stable. Low variability.
- **B (Trending):** New product that starts with low sales (~10/day) and grows to ~150/day.
- **C (Volatile):** Medium sales (~50/day) but with a lot of noise. Hard to predict.
- **D (Seasonal):** Low sales (~30/day) with pronounced weekend spikes (up to 80/day).

We trained two approaches and compared them:

1. Local Approach: 4 independent XGBoost models, each trained only on its own product's data. Each model only "knows" its own history.

2. Global Approach: 1 single XGBoost model trained on the complete Panel table (data from all 4 products together). This model "sees" all histories simultaneously.

7.3.2 Speed

The Global model was trained in **0.18 seconds**, while the 4 local models took **0.70 seconds** in total. With only 4 products, the difference seems minor. But extrapolating: with 10,000 products, the local approach would take ~30 minutes while the Global would take ~45 seconds. With 100,000 products, the difference is hours vs minutes.

7.4 Accuracy results

7.4.1 The cross-learning phenomenon

The most interesting result is Product D: the Global Model improved over the Local thanks to **Cross-Learning**. Product D has a very pronounced weekly seasonality. By training together,

Product	Local MAE	Global MAE	Interpretation
A (Best Seller)	12.06	12.12	Technical tie. Both models handle stable, high-volume products well.
B (Trending)	7.02	7.17	Local wins slightly: B has a unique growth pattern that the Global "dilutes" when mixing with others.
C (Volatile)	18.28	19.62	Local wins: with so much noise, the Global model's cross-learning adds no value and slightly hurts.
D (Seasonal)	5.81	5.44	Global improves: it has more data to learn the weekly pattern from all products.

Table 7.2: Error comparison (MAE). The Global clearly wins on D (Seasonal) thanks to cross-learning of the weekly pattern. For the others, the Local approach matches or slightly outperforms.

the Global model learned that "Saturdays sell more" from a larger pool of data, and applied that knowledge more confidently to D's predictions.

For the other products, the Local approach matched or slightly outperformed the Global. This is expected with only 4 products: Cross-Learning shines when there are hundreds or thousands of series sharing common patterns. With few products, each Local model has enough data on its own. When does each approach win? We formalize this in the next section.

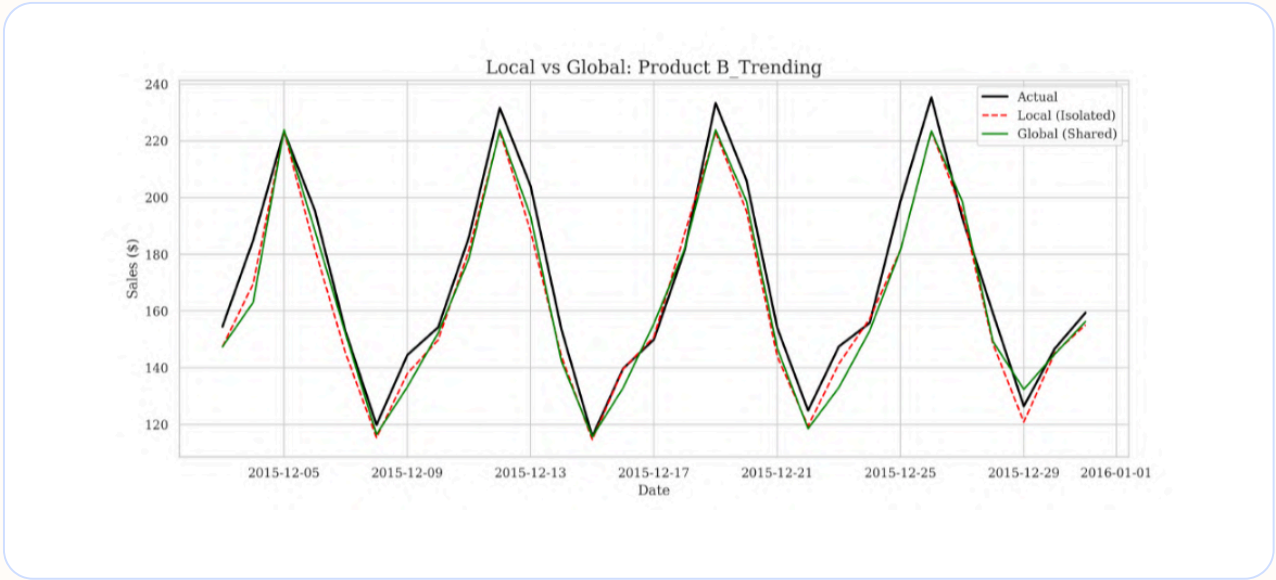


Figure 7.2: Prediction for Product B (Trending) over the last 28 days. Both models capture the weekly pattern. The visual difference is subtle, but the Local (red dashed) fits the peaks better, while the Global (green) tends to smooth. For products with unique behaviors, the local approach may be preferable.

7.5 When global, when local, when hybrid

The experiment above illustrates a general pattern. Here are practical criteria for choosing:

The **hybrid approach** deserves emphasis: train a Global model on all series, then train a lightweight local model on the residuals of series that consistently show high error. This captures universal patterns (via Global) and idiosyncratic behavior (via local correction) without

Approach	When to use	Why
Global	Many series (>50), limited data per series (<100 obs), similar patterns	Cross-learning compensates for scarce individual data
Local	Few series, abundant data per series (>500 obs), unique patterns	Enough data to learn individually; global would dilute specifics
Hybrid	Mixed portfolio: some series data-rich, others data-poor	Global as base + local residual adjustment for outlier series

Table 7.3: Decision framework for Global vs Local vs Hybrid. The key factor is the ratio of data availability to pattern complexity per series.

7.6 Evaluation beyond MAE

A single MAE number across all series can be misleading. A model with MAE = 5 might be excellent for high-volume products but terrible for low-volume ones (where 5 units could mean 100% error).

MAE by tier: Break down accuracy by product category, volume tier, or business unit. A Global model often excels on the long tail (data-scarce series) but underperforms on top sellers versus a well-tuned Local model. Reporting by tier exposes these trade-offs.

Calibration of prediction intervals: If you produce 90% prediction intervals (Chapter 6), check whether they actually contain the true value 90% of the time. Two metrics matter:

- **Coverage Rate (MCR):** Fraction of actuals that fall within the interval. Target: match the nominal level (e.g., 90%).
- **Mean Scaled Interval Width (MSIW):** How wide the intervals are. Narrow + well-calibrated is ideal. Wide intervals that achieve 99% coverage are uninformative.

A model with good MAE but poor coverage is dangerous for inventory planning: it gives precise but overconfident predictions.

7.7 Clustering series before training

When your catalog has very heterogeneous series (e.g., frozen food vs electronics vs seasonal decor), training a single Global model forces it to compromise. An intermediate approach: **cluster-wise global models**.

1. Group series by similarity: by category, by demand pattern (smooth vs intermittent), or by statistical features (mean, variance, seasonality strength).
2. Train one Global model per cluster.
3. Each model specializes in its group while still benefiting from cross-learning within the cluster.

This balances scalability (fewer models than full local) with specificity (better fit than a single global). Recent research (2025) confirms that cluster-wise approaches consistently improve over both pure global and pure local strategies on heterogeneous datasets.

7.8 Ensembling: the simplest win

One of the most consistent findings across forecasting competitions (M4, M5) is that **averaging predictions from 3–5 diverse models outperforms any individual model**. The ensemble does not need to be sophisticated: a simple arithmetic mean of a LightGBM, an XGBoost, and a CatBoost model often suffices.

Why does it work? Each model makes different errors. Averaging cancels out individual mistakes, producing a smoother and more robust forecast. Counterintuitively, optimizing the weights of the ensemble (instead of equal weights) usually makes things worse, because weight estimation introduces its own error.

7.9 Retraining in production

A common question when deploying Global models: **how often should I retrain?** The intuitive answer is "as often as possible," but recent research (2025) challenges this. Studies on large retail datasets show that models retrained monthly perform nearly identically to models retrained daily, with less than 5% accuracy difference even in the no-retraining scenario.

The practical implication: start with weekly or monthly retraining and only increase the frequency if monitoring shows meaningful degradation. This saves significant computational resources, especially at scale.

7.10 Conclusion

Global Models are the modern standard at large companies (Amazon, Uber, Meta). They enable:

- Scaling to millions of series without training millions of models.
- Improving predictions on products with little data thanks to Cross-Learning.
- Predicting new products ("Cold Start") using universal patterns.
- Reducing maintenance costs (1 model vs 10,000).

However, they are not a universal solution. For products with very particular behaviors (like Product B), a hybrid or local approach may be preferable.

Foundation Models (Chapters 8–9) take the Global idea to its extreme: a single model pre-trained on billions of time points from diverse domains. Where our Global model learned from thousands of series in one company, Foundation Models learn from millions of series across industries. The trade-off is control: you gain zero-shot capability but lose the ability to fine-tune feature engineering. Understanding when each approach wins is the subject of the next two chapters.

CHAPTER 8

Foundation Models: theory

8.1 Introduction: beyond training

Up until now, we have always followed a golden rule: "To predict a series, you must train on that series". Even in the Global Model (Chapter 7), the model needed to see the data from all products during training.

But what happens if a new product arrives today and we want to predict its sales tomorrow? We don't have time to re-train. This is the **Cold Start** problem, and it is one of the most common challenges in industry.

8.2 The cold start problem

8.2.1 When does it occur?

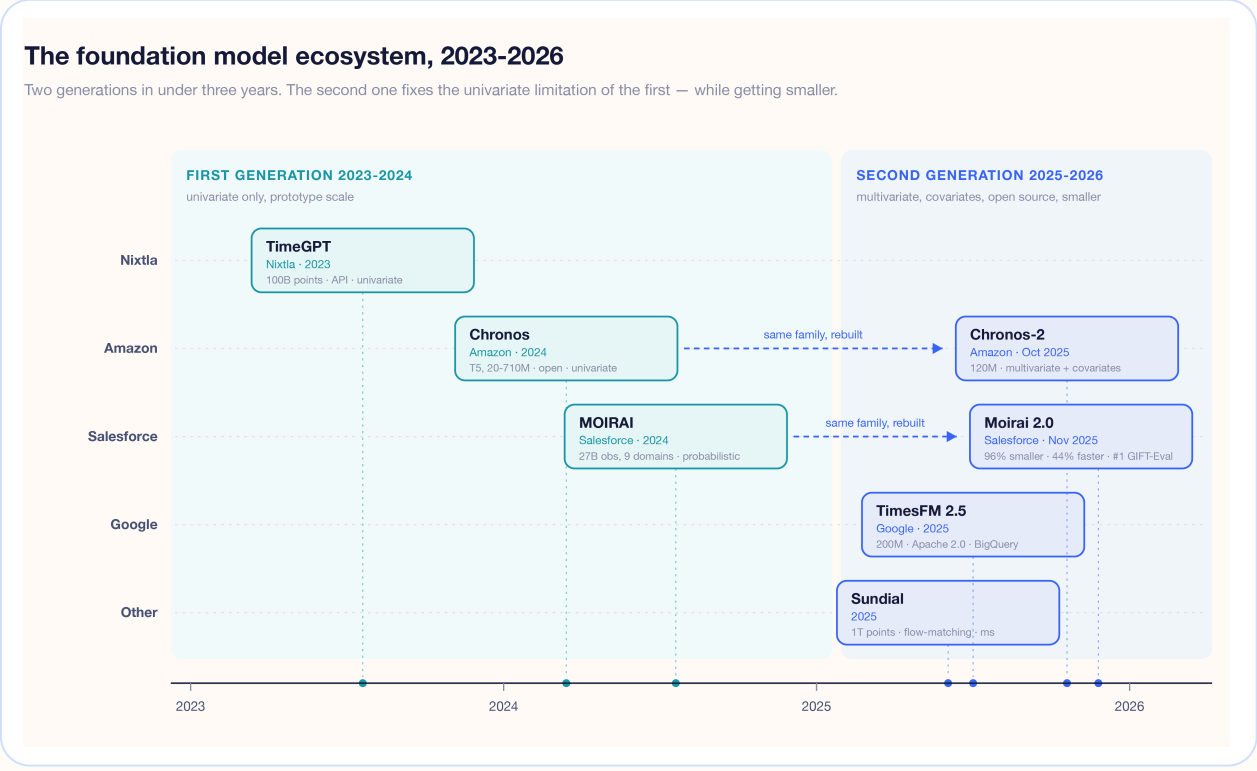
Cold Start appears in multiple real-world scenarios:

- **New product launch:** A store lists a product that has never been sold before. There is no history.
- **Geographic expansion:** A chain opens a store in a new city. Consumption patterns may be different.
- **Seasonal products:** A Christmas item that only sells for 2 months a year. When November arrives, you only have data from last year.
- **Slow-moving products:** Industrial spare parts that sell 3 times a year. There is not enough history for a statistical model.

8.2.2 Why do traditional methods fail?

The models we have learned so far require historical data:

- **ARIMA (Ch 3):** Needs at least 2-3 complete seasonal cycles (24-36 months for monthly data) to estimate parameters.
- **XGBoost (Ch 4):** Needs enough rows to compute lags. With 7 days of data, `lag_7` does not exist.
- **Global Model (Ch 7):** Can partially help (if the new product resembles existing ones), but needs at least some data to compute features.



8.2.3 Modern strategies

1. Similarity-based transfer: Use product metadata (category, price, brand) to find "sibling" products with history and transfer their patterns. Amazon Forecast reports significant improvements in accuracy with this approach.

The industry has developed three main approaches for Cold Start:

2. Meta-Learning: Train a model that "learns to learn": given a new product with little data, the model quickly adapts its parameters. Recent research with the M5 dataset shows substantial error reductions using only 7 days of observations.

In Chapter 10 (Cold Start section) we explore strategies 1 and 2 in depth with practical examples and a comparison table. Here we focus on strategy 3: zero-shot with Foundation Models.

3. Zero-Shot Forecasting: Foundation models that predict without ever having seen the series. This is the current frontier.

8.3 Zero-shot forecasting: predicting without training

It is the ability of a model to predict a time series it has never seen before, without the need for re-training.

It is analogous to language models like GPT: you don't train GPT to write a poem about your specific company; it already knows how to write poems because it read millions of texts. You give it the "prompt" (the beginning of your series) and it completes the rest, based on universal patterns it learned during its massive pre-training.

8.3.1 Our experiment

To demonstrate this, we ran an experiment with our mini-retailer simulation:

1. We trained a Global model only with products A, B, and C.
2. We completely hid Product D (New) from it.
3. We asked it to predict Product D using only its general knowledge about "how sales work".

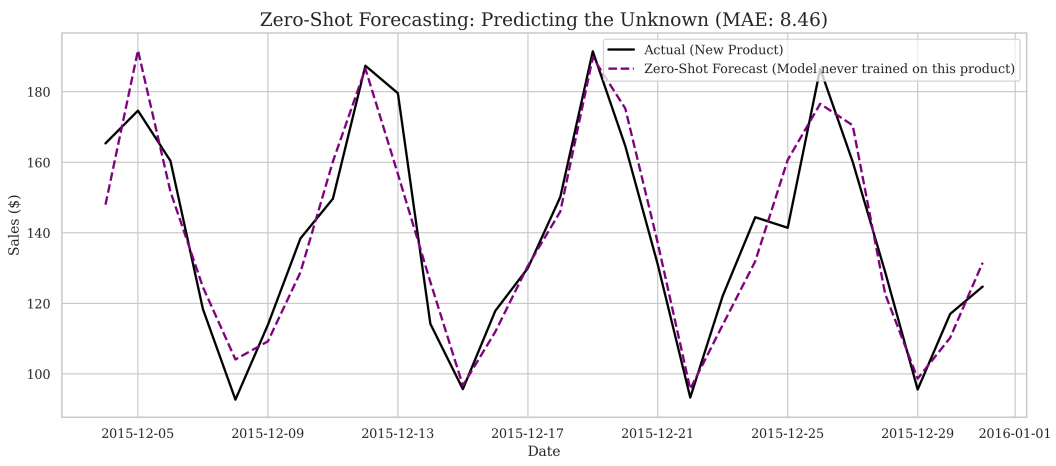


Figure 8.1: Zero-Shot Prediction. The model (purple line) reasonably predicts the New Product (black line) without ever having seen it, based on universal patterns such as weekly seasonality.

The model understood that "there is a weekly seasonality" (more sales on Saturdays) and applied it to the new product. It was not perfect, but it was a useful prediction without any prior data from the product.

8.4 Transformers for time series

Our experiment used XGBoost, but the real revolution comes from the **Transformer** architecture, originally designed for natural language processing (translation, text generation) and which has been successfully adapted to time series.

8.4.1 What is attention?

Imagine you are reading a long sentence and need to remember a word from the beginning to understand the ending. Traditional models (RNNs, LSTMs) process the sequence step by step, which causes distant information to "fade" over time (the vanishing gradient problem).

The **Attention** mechanism solves this by allowing the model to directly connect any point from the past with the present, regardless of temporal distance. The model "decides" which past points to focus on to make each prediction.

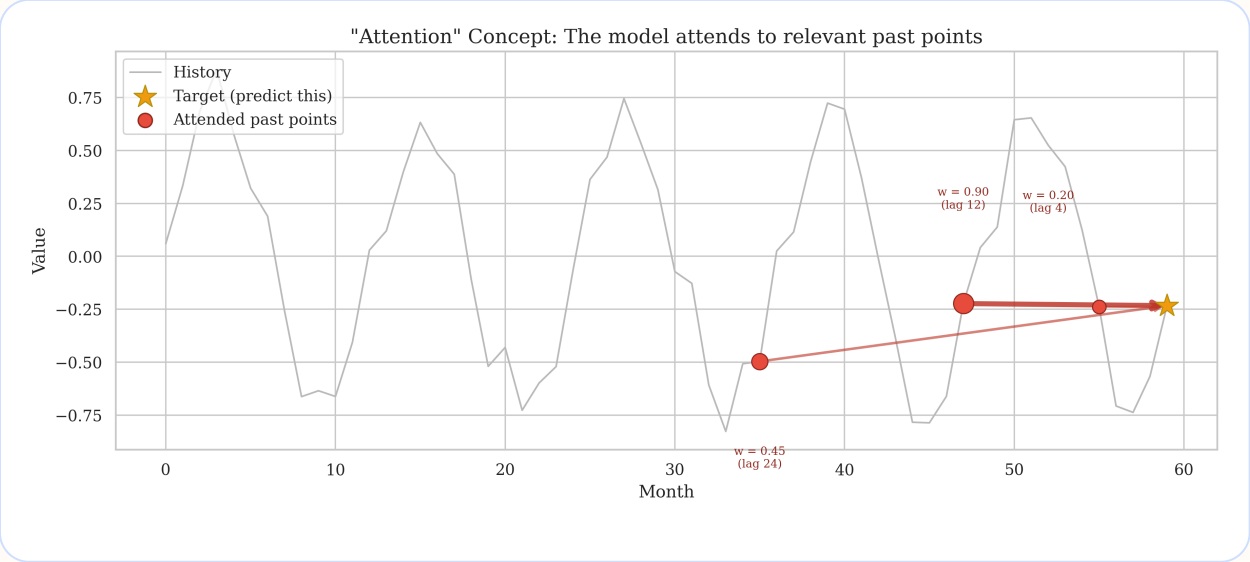


Figure 8.2: Attention Concept. To predict the target value (star), the model assigns different weights to past time steps (red dots). The strongest connection goes to the value 12 months ago — the model learns to exploit annual seasonality. Unlike RNNs, these connections are direct and parallel.

8.4.2 Key components of a transformer for time series

Transformers adapted to time series include:

1. Patching: Instead of feeding the model point by point (as an RNN would), the series is divided into "patches" or subsequences. For example, a daily series of 180 days is divided into 30 patches of 6 days each. Each patch is treated as a "token" (similar to a word in an LLM). This reduces computational complexity and enables capturing local patterns.

2. Multi-Head Attention: The model has multiple attention "heads" that learn different types of relationships simultaneously. One head may focus on weekly seasonality, another on long-term trend, and another on point events.

3. Positional Encoding: Since the Transformer processes all patches in parallel (not sequentially), it needs to know the order of each patch. A signal encoding the temporal position is added.

4. Direct vs Autoregressive Prediction: Transformers for time series typically use direct prediction: they generate all future values at once, instead of predicting one by one and feeding the previous prediction as input (which accumulates errors).

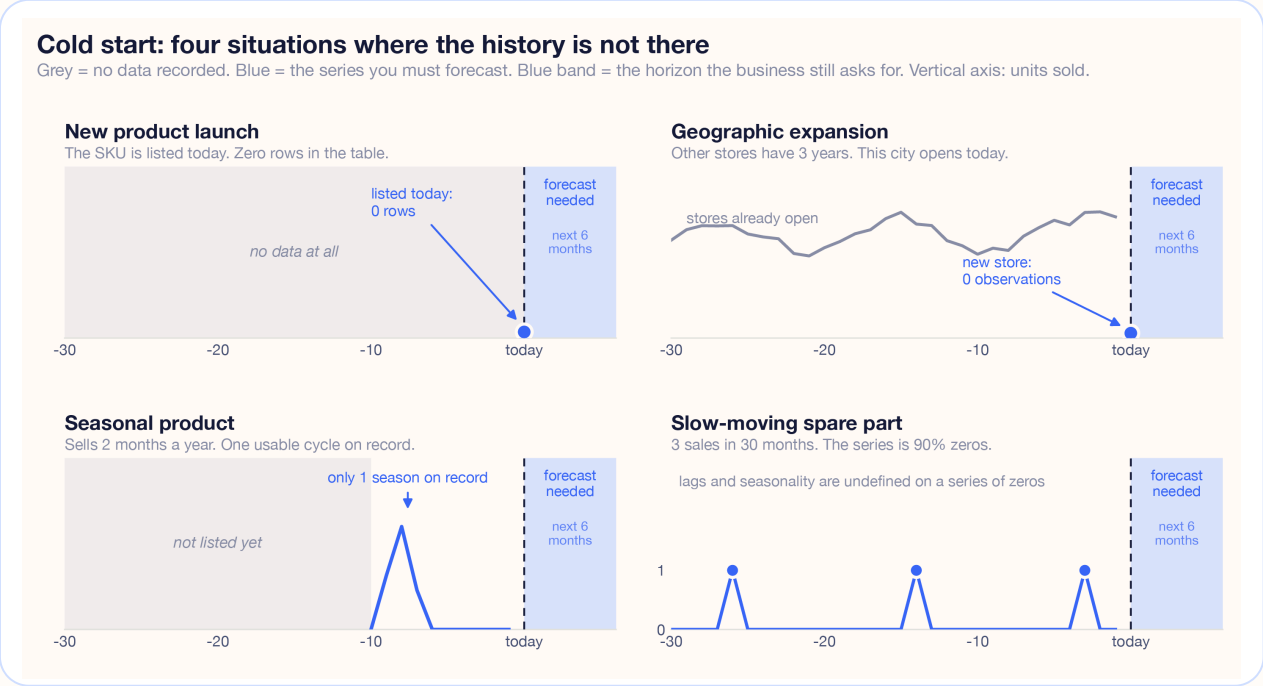
8.5 Foundation Models

A **foundation model** is a Transformer that has been pre-trained with a massive amount of data from multiple domains (finance, energy, retail, climate, traffic) and that can make predictions in any domain without additional training.

8.5.1 How are they trained?

The pre-training process is analogous to that of language LLMs:

1. **Massive data collection:** Millions or billions of time series from public and private sources are gathered.
2. **Transformer training:** The model learns to predict the "next value" (or patch) given the history, just as an LLM learns to predict the "next word".
3. **Transfer:** Once trained, the model can be applied to any new series without re-training. It only needs the history as input ("prompt") and generates the prediction.



8.5.2 The Foundation Model ecosystem

The field has evolved at a remarkable pace. In less than three years (2023-2026), we have gone from the first prototypes to a mature ecosystem with multiple competitive alternatives.

First Generation (2023-2024)

TimeGPT (Nixtla, 2023): The first commercial foundation model for time series. Trained with over 100 billion data points. Accessed via API. Its strength is ease of use: a single call generates predictions with confidence intervals. Limitation: univariate only, no native covariate support.

Chronos (Amazon, 2024): An open source model that tokenizes series, converting numerical values into discrete categories. Uses the Google T5 family (20M to 710M parameters). Pre-trained with public datasets and synthetic data. Limitation: univariate only.

MOIRAI (Salesforce, 2024): Trained with the LOTSA archive: 27 billion observations from 9 domains. Works with any frequency and generates probabilistic predictions natively. Uses multiple patch sizes and mixture distributions.

Second Generation (2025-2026)

The second generation resolves several limitations of the first:

Sundial (2025): Introduces TimeFlow Loss based on flow-matching, an alternative to Chronos tokenization. Pre-trained with 1 trillion points (TimeBench). Achieves millisecond inference and generates multiple probable predictions. It represents a fundamentally different approach: instead of tokenizing the series (discretizing), it works directly with continuous values.

TimesFM 2.5 (Google, 2025): A decoder-only model with 200M parameters, pre-trained with 100 billion points. Open source (Apache 2.0) and integrated into Google BigQuery, making it easy to use in enterprise environments without the need for proprietary ML infrastructure.

Moirai 2.0 (Salesforce, Nov 2025): Architecture change from masked-encoder to decoder-only. It is 96% smaller and 44% faster than Moirai 1.0-Large, with better accuracy. It holds first place on GIFT-Eval (one of the reference benchmarks). It demonstrates that in foundation models, "less can be more".

covariates through a group attention mechanism. It generates over 300 predictions per second on a GPU. It resolves the main limitation of the first generation (univariate only).

Chronos-2 (Amazon, Oct 2025): A significant leap from the original. With 120M parameters and an encoder-only architecture, it now supports multivariate forecasting and

8.5.3 Field trends

Observing the evolution from 2023 to 2026, several clear trends emerge:

- **From univariate to universal:** Models are moving from predicting a single series to handling multiple series and covariates simultaneously.
- **Smaller and faster:** Moirai 2.0 is 96% smaller than its predecessor with better performance. Efficiency is a priority.
- **Dominant open source:** Most second-generation models are open source, unlike TimeGPT which started as a commercial service.
- **Diversity of approaches:** Tokenization (Chronos), mixture distributions (MOIRAI), flow-matching (Sundial), multi-resolution architectures. There is no single winning paradigm.

8.6 Conclusion

We now know the theory behind Foundation Models: how Transformers work, what Attention is, how they are adapted to time series with patching, and how they are pre-trained with massive data. We have also seen how the ecosystem has matured rapidly, with a second generation that resolves many limitations of the first.

In the next chapter, we will evaluate these models in practice against our baselines (SARIMA and XGBoost).

CHAPTER 9

Foundation Models in practice

9.1 Introduction

In Chapter 8, we studied the theory behind Foundation Models: Transformers, Attention, Patching, and massive pre-training. In this chapter, we will evaluate these models in practice, comparing them against the baselines we have built throughout the ebook.

We will evaluate three foundation models:

1. **TimeGPT (Nixtla):** Commercial model, API access.
2. **Chronos (Amazon):** Open source, based on tokenization.
3. **MOIRAI (Salesforce):** Open source, probabilistic predictions.

We will pit them against SARIMA (Ch 3) and XGBoost (Ch 4) on the same simulated retailer dataset, with the last 28 days reserved as test.

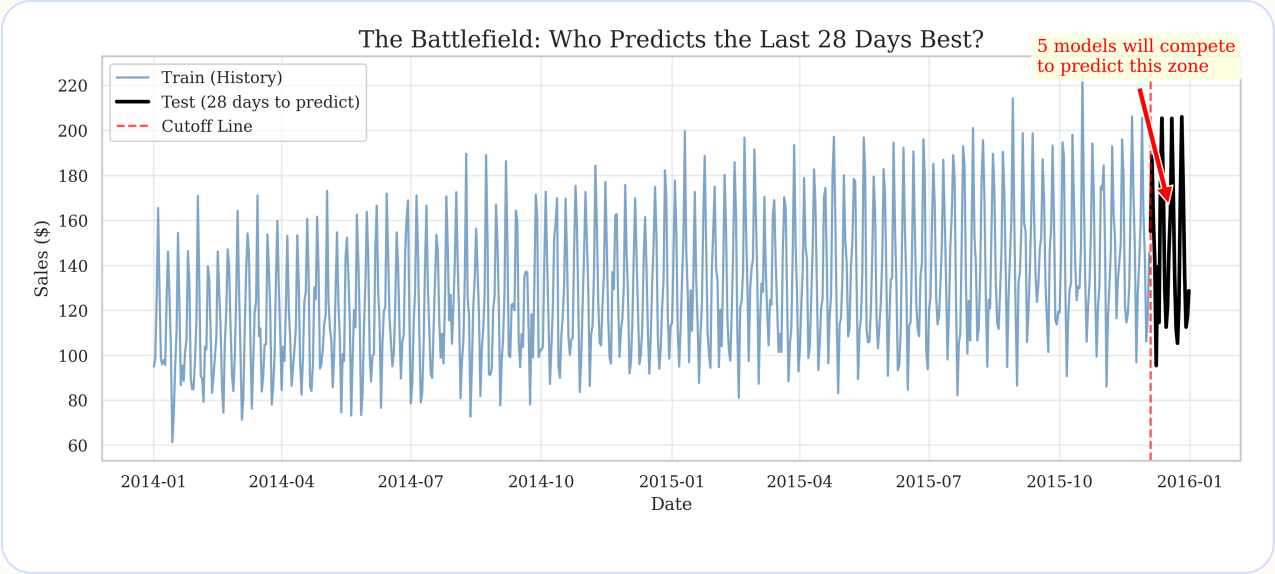


Figure 9.1: The evaluation data: simulated retailer sales. The last 28 days (black) are the test that all models must predict.

9.2 Case 1: TimeGPT (Nixtla)

9.2.1 Characteristics

TimeGPT was the first commercial foundation model for time series. It was trained with over 100 billion data points from varied domains.

Its value proposition is simplicity: it requires no feature engineering, no hyperparameter tuning, and no training. You pass the historical data and it generates the prediction with confidence intervals included.

9.2.2 Results

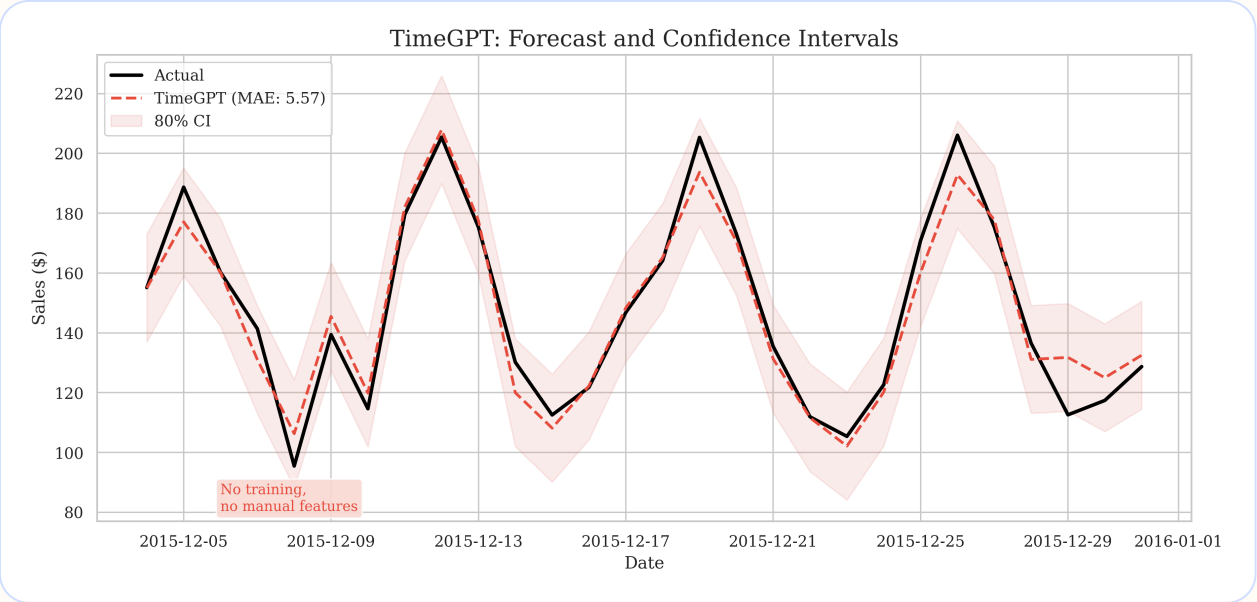


Figure 9.2: TimeGPT: prediction and 80% confidence intervals. MAE of 5.57, outperforming both baselines without any configuration needed.

TimeGPT achieves a MAE of **5.57**, lower than SARIMA (8.08) and XGBoost (12.04). It captures weekly seasonality without us explicitly indicating it.

9.2.3 Limitations

- Requires an API key and has usage costs.
- Data must be sent to external servers (potential privacy concern).
- Being a black-box model, interpretability is nonexistent.

9.3 Case 2: Chronos (Amazon)

9.3.1 Characteristics

Chronos innovates in how it represents time series: it **tokenizes** them. It converts numerical values into discrete categories (tokens), as if the series were text. It uses the Google T5 family (20M to 710M parameters).

Chronos achieves a MAE of **6.30**. Slightly higher (worse) than TimeGPT, but with an advantage: it is completely open source and free. It does not require sending data to external servers.

Note: In October 2025, Amazon released Chronos-2, which supports multivariate forecasting and covariates (a limitation of the original), with 120M parameters and over 300 predictions per second.

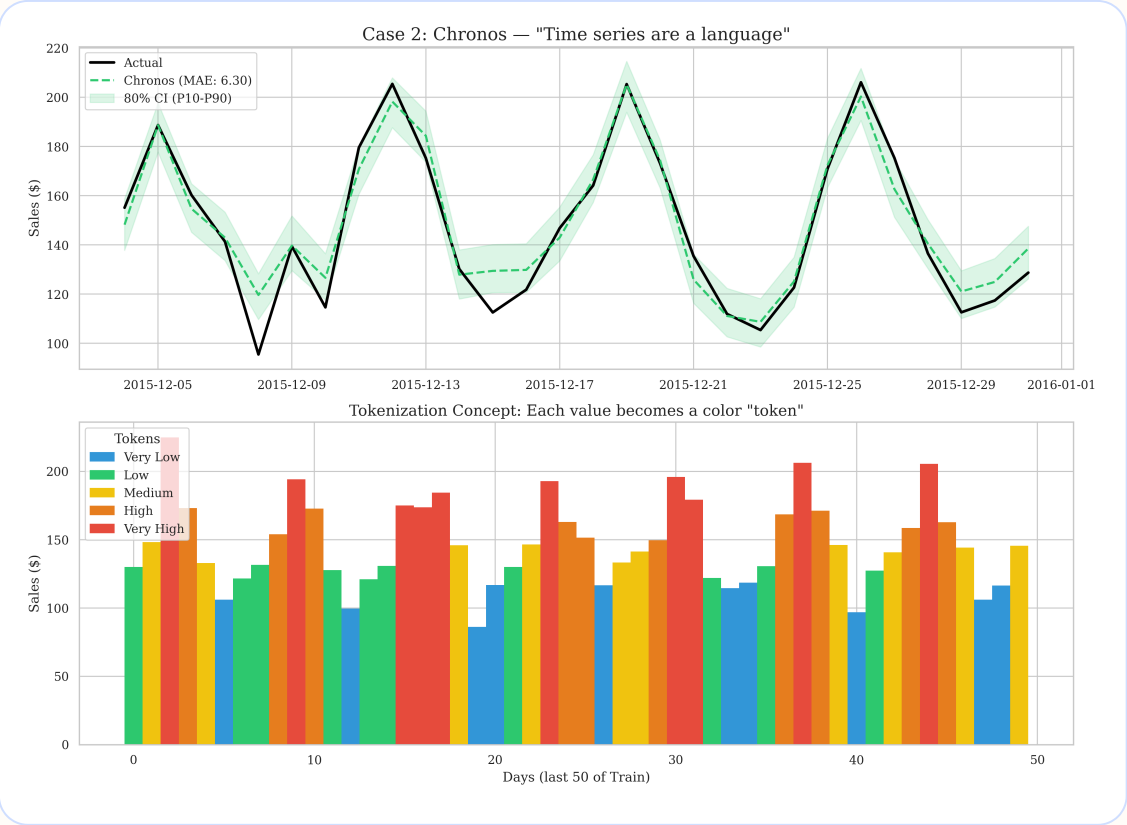


Figure 9.3: Top: Chronos prediction with probabilistic intervals (MAE: 6.30). **Bottom:** Visualization of the tokenization concept: each bar has a color according to its category (token).

9.4 Case 3: MOIRAI (Salesforce)

9.4.1 Characteristics

MOIRAI works with any frequency (hourly, daily, monthly) without changing configuration. It generates probabilistic predictions natively, producing hundreds of future scenarios.

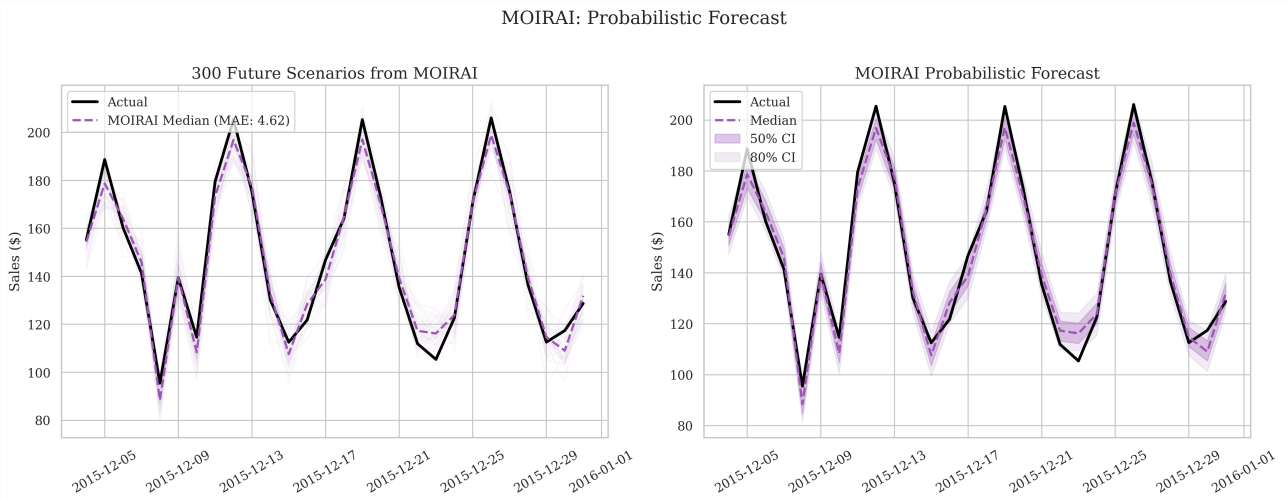


Figure 9.4: Left: 300 future scenarios generated by MOIRAI. **Right:** Intervals derived from those trajectories. Median MAE: 4.62.

MOIRAI achieves the best MAE of the benchmark: **4.62**. Its probabilistic approach allows not only predicting the expected value but also quantifying the complete uncertainty.

Note: In November 2025, Salesforce released Moirai 2.0, a model 96% smaller and 44% faster than MOIRAI 1.0, with better accuracy. It holds first place on the GIFT-Eval benchmark.

9.5 The consolidated benchmark

Model	MAE	Features	Training	Cost	Open Source	Type
SARIMA	8.08	No	Yes	Free	Yes	Classical
XGBoost	12.04	Yes (manual)	Yes	Free	Yes	ML
TimeGPT	5.57	No	No	Paid	No	Foundation
Chronos	6.30	No	No	Free	Yes	Foundation
MOIRAI	4.62	No	No	Free	Yes	Foundation

Table 9.1: Complete benchmark. The "Training" column indicates whether the model needs to be trained on the specific problem's data.[2]

9.5.1 Interpretation of results

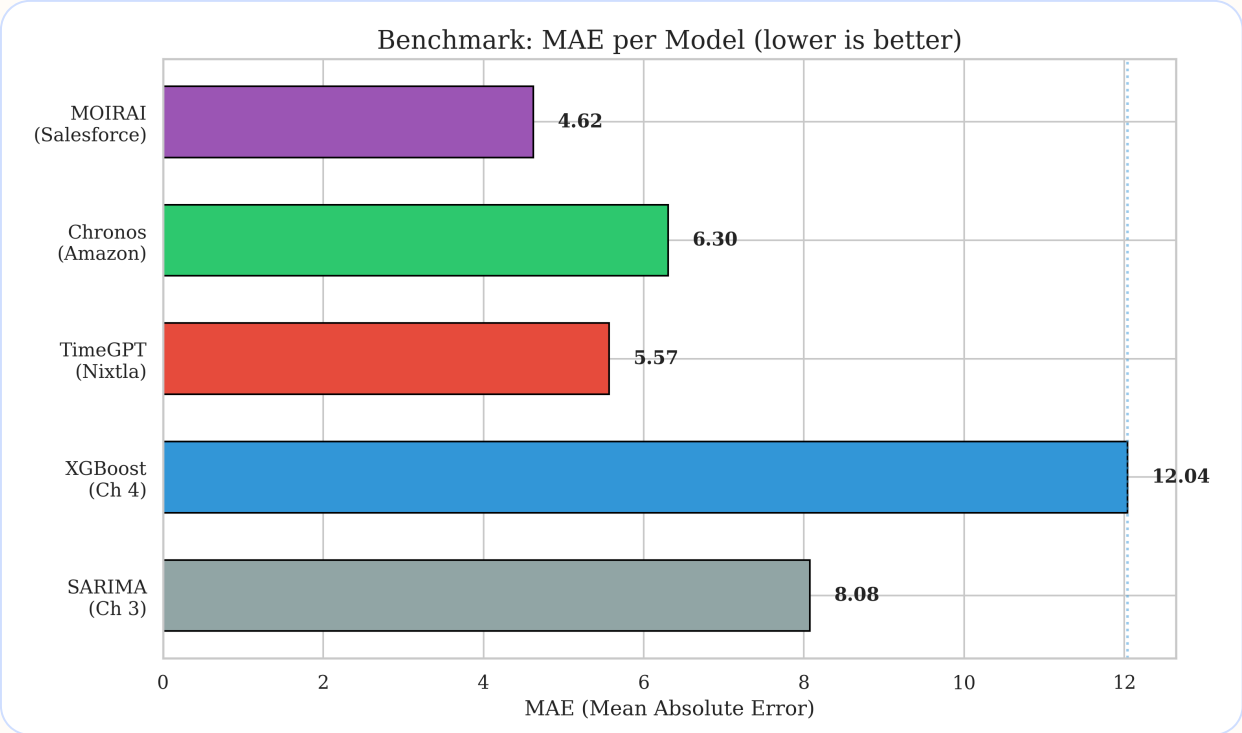


Figure 9.5: MAE for each model (lower is better).

The results reveal several relevant findings:

1. In this simulated benchmark (see footnote for methodological details), all three foundation models outperform both baselines. MOIRAI (4.62), TimeGPT (5.57), and Chronos (6.30) achieve lower errors than SARIMA (8.08) and XGBoost (12.04). This happens without feature engineering, without specific training, and without hyperparameter tuning.

2. XGBoost is the worst in the benchmark. This may seem contradictory to Chapter 4, where XGBoost outperformed SARIMA. The difference is that in Ch 4 we used a complete feature set (lag_1, lag_7, lag_365, rolling_mean, rolling_std, day_of_year, is_weekend). Here we used a simplified set (without lag_365 or is_weekend). The lesson: **XGBoost without good features is worse than ARIMA**. The power of ML depends entirely on feature engineering. With the full Boosting family (LightGBM, CatBoost) and advanced features (Fourier, target encoding) as described in Chapter 4, ML results would be significantly better.

3. SARIMA surprises. With zero configuration effort (beyond choosing the parameters), it achieves a MAE of 8.08, outperforming XGBoost. Well-configured classical models remain competitive.

4. MOIRAI leads. Its probabilistic approach (multiple future scenarios) enables it to better capture uncertainty and generate more accurate predictions. Moreover, it is open source and free.

5. Important caveat: This benchmark uses a simulated dataset with clear seasonal patterns and first-generation models (2024). With the most recent versions (Chronos-2, Moirai 2.0), the results would likely be even more favorable for foundation models. On real data with noise, complex exogenous variables, or structural breaks, results could differ. That is why the following section is essential.

9.6 The other side of the coin: limitations and criticisms

The benchmark results are favorable for foundation models. However, recent scientific literature (2024-2025) presents a more nuanced picture.

9.6.1 Criticism 1: they do not always outperform simple models

In a benchmark for electricity price forecasting in European markets (2024), the statistical model MSTL (Multi-Seasonal-Trend Decomposition using Loess) — a relatively simple method that captures daily and weekly seasonalities — achieved superior and more consistent results than Chronos, TimeGPT, MOIRAI, and other foundation models.

Lesson: In domains where seasonal patterns are clear and well defined, a well-configured statistical model can match or outperform a foundation model with billions of parameters.

9.6.2 Criticism 2: the generalization problem

A 2025 study argues that foundation models for time series commit a category error: unlike language (where words have shared meaning), time series from different domains do not share semantics. A spike in financial data, a spike in an electrocardiogram, and a spike in turbine vibrations have completely different causes.

Foundation models learn generic statistical properties (seasonality, trend), but not the underlying causes. This limits their real generalization capability.

9.6.3 Criticism 3: zero-shot depends on the pre-training domain

Zero-shot performance depends heavily on whether the model was pre-trained with data similar to the target domain. If your series belongs to a domain poorly represented in the training data (e.g., specialized biomedical data), performance can be mediocre.

9.6.4 Criticism 4: covariates and exogenous variables

First-generation foundation models (TimeGPT, Chronos, MOIRAI) had difficulties incorporating exogenous variables (price, weather, promotions). In contrast, XGBoost can integrate any additional feature naturally.

This limitation is being addressed by the second generation: **Chronos-2 (Oct 2025)** already supports covariates natively through its group attention mechanism. However, covariate integration in foundation models remains an active area of research, and for problems where external variables are the dominant factor (e.g., impact of promotions on sales), classical ML models with manual features may still be preferable due to their control and explainability.

9.7 When Foundation Model, when not?

For the general model selection guide (including SARIMA, Boosting, and Global Models), see the Road Map at the beginning of the book. Here we focus on the specific question: **when does a Foundation Model add value over the alternatives?**

Situation	Foundation Model	Preferable alternative
Quick prototype, no time for feature engineering	Yes (zero-shot)	–
Cold start: new product with no history	Yes (Chronos, MOIRAI)	Similarity transfer (Ch 10)
Exogenous variables dominate (price, promos)	No	Boosting with features (Ch 4)
Domain with clear, well-defined patterns	Evaluate	Well-configured SARIMA may match (Ch 3)
50+ series, need to scale	Evaluate	Global Model + ensembling (Ch 7)
Explainability required	No	SARIMA (coeff.) or Boosting + SHAP (Ch 4)
Sensitive data, cannot leave the company	Only local open source (Chronos, MOIRAI)	Local Boosting or Global

Table 9.2: When does a Foundation Model add value? The answer depends on the combination of constraints in your problem.

The most robust recommendation: use Foundation Models as a quick baseline, then evaluate whether a model trained on your data (Boosting, Global) outperforms it. If it does not, the Foundation Model saves you all the feature engineering and maintenance work. If it does, you already have a solid benchmark. Either way, you win.

9.8 Conclusion

Foundation models represent a significant advance in forecasting accessibility: they allow obtaining reasonable predictions without modeling expertise. However, they are not a universal solution.

Current scientific evidence (2024-2026) shows that:

1. In domains with well-defined patterns, simple models can match or outperform foundation models.

2. Real generalization ("understanding" any series) is still limited, although the second generation shows significant improvements.

3. Exogenous variables, which were the exclusive territory of ML models, are beginning to be supported by the second generation (Chronos-2).

The field evolves rapidly: between 2023 and 2026, we have seen two complete generations of models. The principles of rigorous evaluation (backtesting, appropriate metrics, comparison against baselines) remain more important than the choice of model itself.

In the next chapter, we will face the real-world challenges that none of these models solve automatically: intermittent demand, cold start, and anomaly detection.

CHAPTER 10

Real-world challenges

10.1 Introduction

Throughout the previous chapters, we have worked with relatively well-behaved data: series with clear trends, regular seasonality, and no missing values. However, in industrial practice, data is rarely this clean.

In this chapter, we will address three problems that constantly appear in real forecasting projects and that require specific techniques:

1. **Intermittent Demand:** Series where 60-70% of the days have zero sales.
2. **Cold Start:** New products with no history. How to predict without data?
3. **Anomaly Detection:** Spikes, drops, and rare events that contaminate your models.

And we will close with a reflection on the most frequent mistakes in forecasting practice.

10.2 Intermittent demand: the problem of zeros

10.2.1 The real problem

Until now, our data has been "nice": AirPassengers has passengers every month, our simulated retailer has sales every day. But in the reality of retail, a large proportion of SKUs (often 60–80%) have **intermittent demand**.

We use real data from the Kaggle M5 competition. Product `FOODS_1_002` has sales on only 34% of the days. The remaining 66%, demand is zero.

10.2.2 The counterfactual: why do the models from Ch 3-4 fail?

We applied SARIMA and XGBoost to this series. The MAEs are similar (all around 0.93), but that is misleading: with such low sales (0-4 units), any model has a small MAE. The real problem is seen in the **cumulative bias**: how many extra or missing units you would order from the supplier if you used these predictions over 28 days.

What do we observe in Figure 10.2?

- **SARIMA (top panel):** Predicts a wavy line around 1 unit each day. When the actual sale is 0 (most days), the model systematically errs by predicting sales where there are none. When the sale is 8 (the spike on May 25th), SARIMA predicts 1: a huge error.
- **XGBoost (middle panel):** Similar to SARIMA: predicts low continuous values. The lags (yesterday's sales, last week's sales) are almost always 0, so XGBoost has no useful information to distinguish days with sales from days without sales.

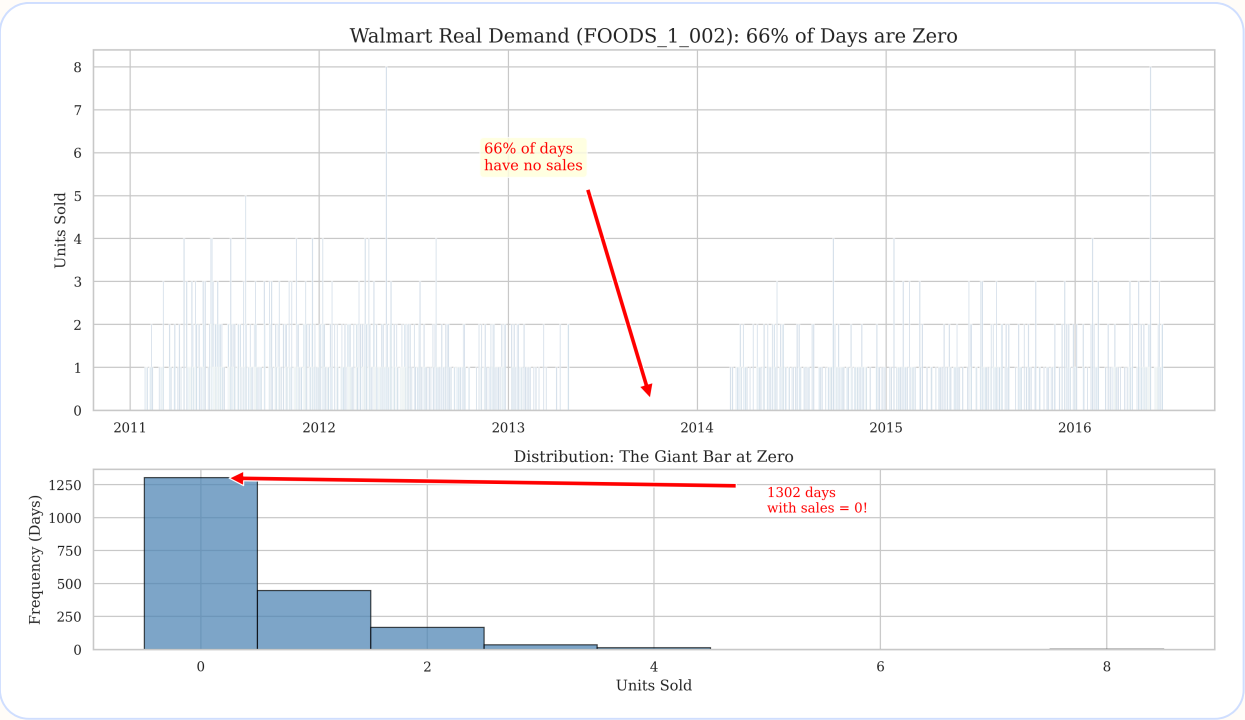


Figure 10.1: Real demand of an M5 competition product. **Top:** Sporadic sales with a majority of zeros. **Bottom:** The histogram confirms that the zero bar completely dominates.

- **Croston SBA (bottom panel):** Predicts a flat constant line (~0.7 units/day). At first glance it seems "equally bad". But the difference is conceptual: Croston does not attempt to predict the exact day of the sale, but rather the average demand rate, which is what matters for inventory management.

10.2.3 The solution: Croston-SBA method

Croston's method (1972) decomposes the problem into two independent sub-problems:

1. How many days between demands? Models the intervals between non-zero demands using Simple Exponential Smoothing (SES).

2. How much demand when it occurs? Models the size of non-zero demands using another independent SES.

The final prediction combines both:

$$\hat{Y} = \frac{\hat{z}}{\hat{p}} \times \left(1 - \frac{\alpha}{2}\right)$$

where:

- $\hat{z}$ = estimated demand size when it occurs (e.g., if the last non-zero sales were 2, 3, 1, 2, $\hat{z} \approx 2$).
- $\hat{p}$ = estimated interval between non-zero demands in days (e.g., if sales occur every 3-4 days on average, $\hat{p} \approx 3.5$).
- α = smoothing parameter (typically 0.1, gives more weight to history than to the last observation).

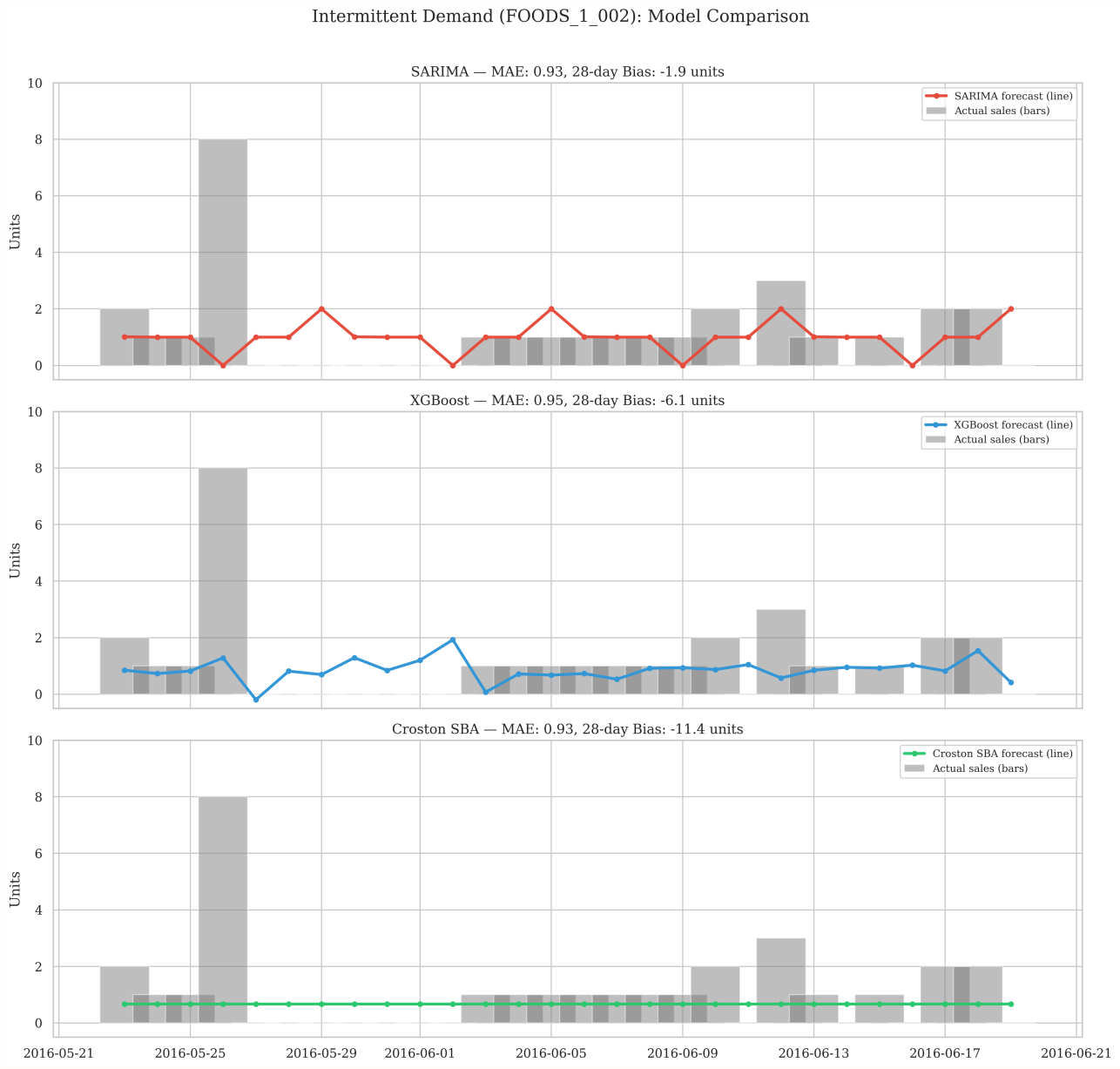


Figure 10.2: Comparison of 3 models over 28 test days. The gray bars are the actual sales (most are zero). The **colored lines with dots** are each model's predictions. Each panel shows the MAE and. cumulative bias (how many extra or missing units we would order from the supplier).

- $(1 - \alpha/2)$ = SBA (Syntetos-Boylan Approximation) correction that reduces a known positive bias of the original Croston method.

Numerical example: If we estimate that the typical demand is $\hat{z} = 2$ units and it occurs every $\hat{p} = 3$ days, with $\alpha = 0.1$:

$$\hat{Y} = \frac{2}{3} \times (1 - 0.05) = 0.667 \times 0.95 = 0.63 \text{ units/day}$$

This equals 0.63 × 7 = 4.4 units per week. An inventory manager can use this number directly: "I need to have at least 5 units in stock each week to cover the expected demand".

10.2.4 A note on metrics

In Chapter 5 we saw that MAPE is calculated as $\frac{|y-\hat{y}|}{y}$. When $y = 0$ (which in intermittent demand happens 66% of the days), the division produces $\frac{|0-\hat{y}|}{0} = \infty$. A single day with zero sales makes the average MAPE infinite, completely invalidating the metric.

Suitable alternatives for intermittent series:

- **MAE:** Always works, but does not provide percentage context.
- **RMSSE (Root Mean Squared Scaled Error):** The official metric of the Kaggle M5 competition. Scales the error relative to a naive baseline, and has no issues with zeros.
- **Fill Rate:** An inventory metric. It answers: "What percentage of actual demand could we fulfill with the stock we had?" It is the most relevant for business decisions.

10.3 Cold start: the phantom product

10.3.1 The problem

A category manager calls you: "We just listed a new product. It's been in the store for 7 days. I need a forecast for the next 4 weeks." What do you do?

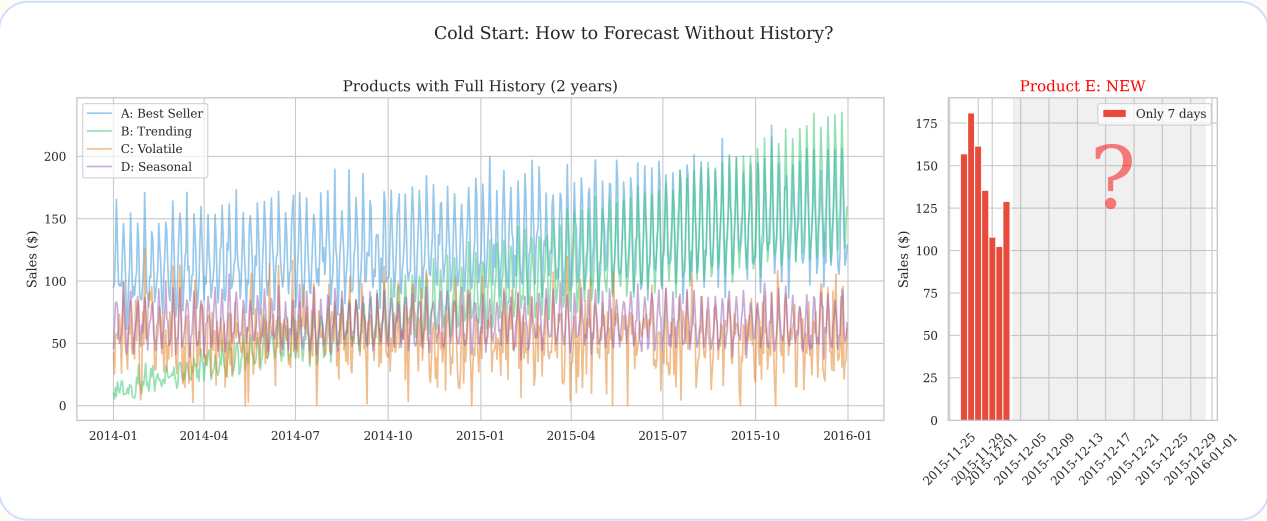


Figure 10.3: Left: 4 products with 2 years of history. **Right:** Product E has only 7 data points and a big question mark.

Attempting SARIMA with 7 data points does not work (it needs at least one complete seasonal cycle). XGBoost with 7 rows does not have enough data to compute lags. It is a dead end.

10.3.2 Strategy 1: similarity-based transfer

The idea is to find existing products with complete history that resemble the new product, and use their behavior as a proxy. There are several ways to measure that similarity:

- **By metadata:** Same category, same price range, same brand. Amazon Forecast uses this approach and reports improvements of up to 45% in accuracy by explicitly selecting similar products by attributes (brand, color, size, category).
- **By statistical correlation:** With the few available data points from the new product (e.g., 7 days), compute the correlation with the last 7 days of each existing product and choose the most similar one.
- **By visual similarity:** In fashion retail, recent research uses neural networks (ResNet-50) to compare product images and find visually similar items. If a new dress looks like another one that already has history, its demand pattern is transferred.

Once the "sibling" product is identified, its history is scaled by the sales level of the new product. If the sibling sells an average of 100 units/day and the new one has been selling 70 units/day for 7 days, the sibling's entire history is multiplied by 70/100 = 0.7.

Advantage: Works with any subsequent model (you can use the transferred history as input for SARIMA, XGBoost, or any other). **Limitation:** If there are no truly similar products in the catalog, the transfer can introduce biases.

10.3.3 Strategy 2: global model with cross-learning

The Global Model from Chapter 7, by training on all existing products, learns universal patterns ("Saturdays sell more", "December has a peak"). When a new product arrives, the model applies these universal patterns by combining them with the few available features (day of the week, month) and the scarce lags that can be computed.

Advantage: Does not need to explicitly identify a "sibling" product; cross-learning is automatic. **Limitation:** If the new product has a behavior radically different from everything seen (e.g., a product from an entirely new domain), the model will apply patterns that do not correspond.

10.3.4 Strategy 3: zero-shot with Foundation Models

The foundation models (Chronos, MOIRAI) from Chapters 8-9 were pre-trained with billions of series from multiple domains. They can receive the 7 data points from the new product as "context" and generate predictions based on universal patterns learned during pre-training.

Advantage: Does not require proprietary infrastructure or data from other products. **Limitation:** Performance depends on whether the domain of the new product was represented in the pre-training data (see Chapter 9, Criticism 3).

10.3.5 Which one to choose?

The choice depends on concrete factors of the context:

In practice, many companies combine strategies: they use similarity-based transfer as a quick first estimate and then refine with a global or foundation model when more data becomes available.

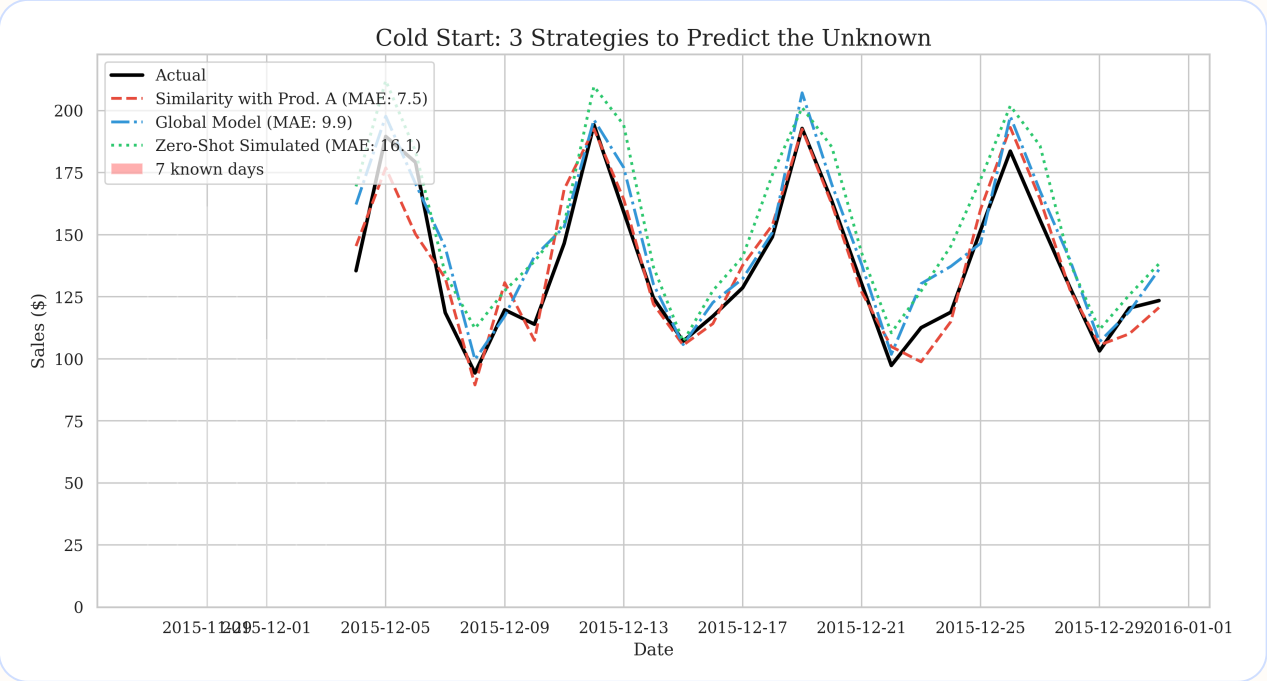


Figure 10.4: Comparison of the 3 strategies on Product E (28-day prediction with only 7 days of history). Each strategy addresses the problem from a different angle.

Factor	Similarity	Global Model	Zero-Shot
Catalog with similar products	Ideal	Works well	Not necessary
Product from a new domain	Not viable	Risk of bias	Best option
Existing ML infrastructure	Not required	Requires trained model	Requires API or GPU
Implementation speed	Fast (heuristic)	Medium (re-train)	Fast (API/download)
Control and explainability	High (you know which sibling it chose)	Medium	Low (black box)

Table 10.1: Selection guide for Cold Start. There is no universally better strategy; the choice depends on available infrastructure, catalog composition, and explainability requirements.

10.4 Anomaly detection: finding the unusual

10.4.1 Why does it matter?

We do not always want to predict the future. Sometimes we want to detect the *abnormal*: bank fraud, sensor failures, unexplained traffic spikes.

We use real data from the Numenta Anomaly Benchmark: taxi trips in NYC.

10.4.2 The counterfactual: the cost of ignoring anomalies

We injected 5 artificial anomalies (×1.6 spikes and their adjacent days ×1.3) into the training data. Then we trained two SARIMA models: one with the contaminated data and another with clean data.



Figure 10.5: Top: Data with injected anomalies (red dots). **Bottom:** The contaminated model's predictions are distorted compared to the clean model.

The contaminated model "learned" the false spikes as if they were real patterns, distorting its future predictions. **Lesson: always review and clean your data before training.**

10.4.3 Three detection methods

Method 1: Z-Score on Residuals. Train a model, compute residuals ($y - \hat{y}$), and flag as an anomaly if $|\text{residual}| > 3\sigma$. It is the most conservative (only detects very obvious anomalies).

Method 2: Confidence Intervals (Ch 6). Use the model's 99% intervals. If a point falls outside, it is an anomaly. The advantage is that the interval adapts to the series level.

Method 3: Isolation Forest. An ML algorithm that "isolates" rare points. It does not need a temporal model and can detect multivariate anomalies. It is the most sensitive.

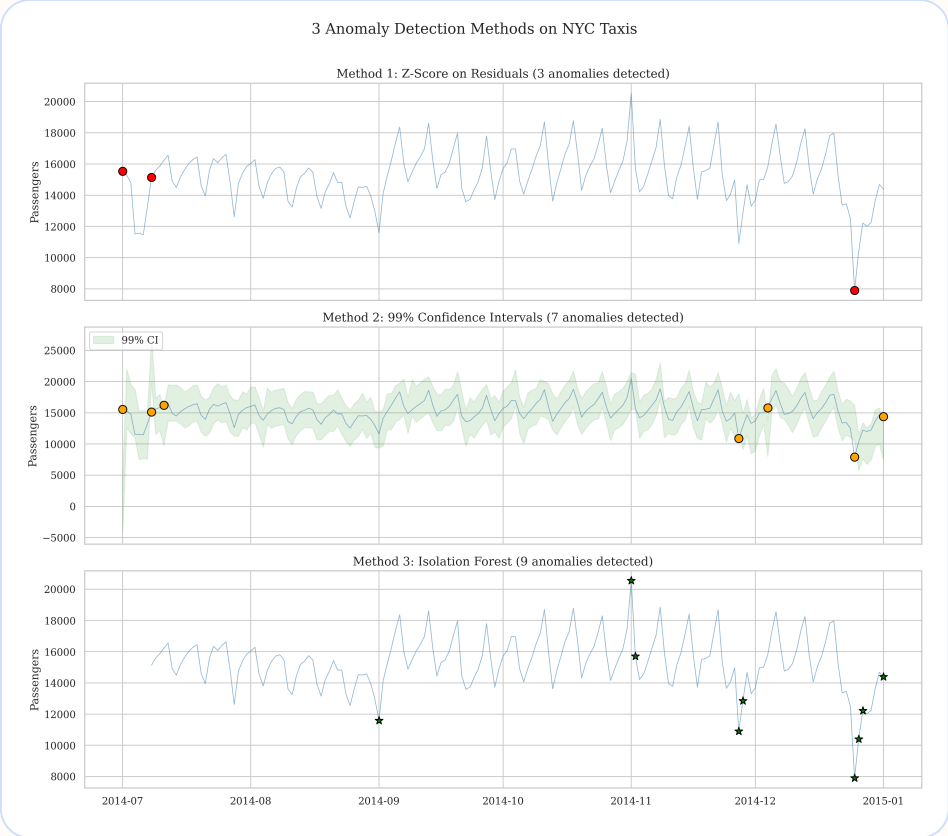


Figure 10.6: Each panel shows a detection method. Z-Score (3 anomalies) is conservative. 99% Intervals (7) is intermediate. Isolation Forest (9) is the most sensitive.

10.5 Common mistakes in forecasting practice

Throughout this ebook, we have presented techniques and models. But in professional practice, the most costly errors are usually not technical but methodological. Recent literature and industry experience identify recurring patterns worth documenting.

10.5.1 Data handling errors

Using random splits instead of temporal ones. This is probably the most serious and most frequent error among beginners. Randomly mixing training and test data (as done in classification) violates the temporal structure: the model "sees the future" during training. Results look excellent in validation but fail in production. The solution is to always use temporal splits: train on the past and evaluate on the future, as we saw in Chapter 5.

Not visually inspecting data before modeling. Summary statistics (mean, variance) can hide patterns that a simple plot reveals instantly: level shifts, missing data, outliers, frequency changes. Before any model, you should always plot the data.

Ignoring anomalies in training data. As we demonstrated in this chapter, training with contaminated data makes the model "learn" false patterns. Data cleaning is not optional; it is the first step.

10.5.2 Model selection errors

Using the most complex model available by default. In the Kaggle M5 competition, relatively simple statistical methods ranked among the top 50. A well-configured SARIMA can outperform a poorly trained neural network. Complexity without justification is not sophistication; it is noise.

Not establishing a naive baseline. If your sophisticated model does not beat the rule "tomorrow will be the same as today" (naive forecast), there is a fundamental problem. Every model should be compared against a simple baseline before declaring that it "works".

Using MAPE as a metric on series with zeros. As we saw in the intermittent demand section, a single zero in the actual data makes MAPE explode to infinity, invalidating the entire evaluation.

10.5.3 Errors in communicating results

Reporting point predictions without uncertainty. As we discussed in Chapter 6, a prediction without a confidence interval is incomplete. The decision maker needs to know not only "how much", but "with what level of certainty".

Evaluating with a single test period. A model may get lucky in a calm month and fail in a volatile one. Backtesting with multiple folds (Chapter 5) is the only honest way to evaluate expected performance.

10.6 Final reflections

We have come a long way: from decomposing a series into its basic components (Chapter 1) to facing the problems that no model solves automatically. This is a good moment to look back and consolidate what we have learned.

Time series forecasting combines statistical rigor with engineering pragmatism. There is no universal solution: each problem has its particularities (data frequency, presence of zeros, availability of external variables, number of series) that determine which tool is most appropriate.

Classical models like ARIMA remain relevant due to their interpretability and efficiency in well-defined univariate problems. Machine Learning models (XGBoost, LightGBM, CatBoost) dominate when there are multiple sources of information, exogenous variables, and non-linear patterns. Global Models scale to thousands of series by sharing knowledge across them. And Foundation Models open the door to predictions without specific training, albeit with limitations that are still being researched.

What remains constant, regardless of the tool chosen, are the fundamental principles: understand the data before modeling it, validate rigorously to avoid self-deception, quantify uncertainty to make better decisions, and maintain a critical attitude toward any result that seems too good to be true.

The field evolves at an unprecedented pace: between 2023 and 2026, we have seen two complete generations of foundation models, with drastic improvements in efficiency (Moirai 2.0 is 96% smaller than its predecessor) and capabilities (Chronos-2 now supports covariates). It is likely that by the time this text is read, new models will exist that surpass those described here. But the fundamentals — stationarity, temporal validation, feature engineering, uncertainty quantification — will remain the foundation upon which any future advance is built.

Next steps

This guide covers the fundamentals and the most widely used tools, but the field has layers of complexity beyond what we addressed here. Four natural directions to keep growing:

- **Hierarchical forecasting and reconciliation:** In practice, you rarely forecast a single series in isolation. A company needs coherent forecasts at the SKU, category, store, and region levels, with individual forecasts summing to the total. Reconciliation techniques (bottom-up, top-down, optimal reconciliation) solve this problem and are essential in any forecasting system at scale.
- **Causal inference in time series:** Forecasting answers "what will happen?", but many business decisions require answering "what would happen if I change X?". Intervention analysis and counterfactual reasoning allow estimating the impact of a promotion, a price change, or an external event, separating it from the organic behavior of the series.
- **Adaptive models and online learning:** The models we presented are retrained periodically on batches of data. But in high-frequency systems (dynamic pricing, real-time fraud detection), models need to update incrementally with each new observation without retraining from scratch.
- **Multi-objective optimization:** In production, minimizing MAE is not the only goal. A real forecasting system balances accuracy, inventory cost, customer service level, and computational cost. This intersection of forecasting and operations research is where models become business decisions.

Bibliography

General references

Hyndman, R. J., & Athanasopoulos, G. (2021). *Forecasting: Principles and Practice* (3rd ed.). OTexts. `https://otexts.com/fpp3/`

Box, G. E. P., Jenkins, G. M., Reinsel, G. C., & Ljung, G. M. (2015). *Time Series Analysis: Forecasting and Control* (5th ed.). Wiley.

Chapter 1, the anatomy of a time series

Cleveland, R. B., Cleveland, W. S., McRae, J. E., & Terpenning, I. (1990). STL: A Seasonal-Trend Decomposition Procedure Based on Loess. *Journal of Official Statistics*, 6(1), 3–73.

Box & Jenkins (2015), Series G, source of the `AirPassengers` dataset (monthly international airline passengers, 1949–1960).

Chapter 2, stationarity and exploratory analysis

Dickey, D. A., & Fuller, W. A. (1979). Distribution of the Estimators for Autoregressive Time Series with a Unit Root. *Journal of the American Statistical Association*, 74(366), 427–431.

Said, S. E., & Dickey, D. A. (1984). Testing for unit roots in autoregressive-moving average models of unknown order. *Biometrika*, 71(3), 599–607. (Augmented Dickey-Fuller.)

Chapter 3, classical tools: ARIMA

Ljung, G. M., & Box, G. E. P. (1978). On a measure of lack of fit in time series models. *Biometrika*, 65(2), 297–303.

Hyndman, R. J., & Khandakar, Y. (2008). Automatic Time Series Forecasting: The `forecast` Package for R. *Journal of Statistical Software*, 27(3), 1–22. (Basis of `auto_arima`.)

Chapter 4, time series as supervised regression

Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. *Proceedings of KDD '16*, 785–794.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T.-Y. (2017). LightGBM: A Highly Efficient Gradient Boosting Decision Tree. *Advances in Neural Information Processing Systems*, 30.

Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. V., & Gulin, A. (2018). CatBoost: unbiased boosting with categorical features. *Advances in Neural Information Processing Systems*, 31.

Lundberg, S. M., & Lee, S.-I. (2017). A Unified Approach to Interpreting Model Predictions. *Advances in Neural Information Processing Systems*, 30. (SHAP.)

Ben Taieb, S., Bontempi, G., Atiya, A. F., & Sorjamaa, A. (2012). A review and comparison of strategies for multi-step ahead time series forecasting based on the NN5 forecasting competition. *Expert Systems with Applications*, 39(8), 7067–7083. (Recursive, direct and hybrid strategies.)

Chapter 5, validation and error metrics

Tashman, L. J. (2000). Out-of-sample tests of forecasting accuracy: an analysis and review. *International Journal of Forecasting*, 16(4), 437–450.

Bergmeir, C., & Benítez, J. M. (2012). On the use of cross-validation for time series predictor evaluation. *Information Sciences*, 191, 192–213.

Hyndman, R. J., & Koehler, A. B. (2006). Another look at measures of forecast accuracy. *International Journal of Forecasting*, 22(4), 679–688. (MASE.)

Chapter 6, navigating uncertainty

Vovk, V., Gammerman, A., & Shafer, G. (2005). *Algorithmic Learning in a Random World*. Springer.

Angelopoulos, A. N., & Bates, S. (2023). Conformal Prediction: A Gentle Introduction. *Foundations and Trends in Machine Learning*, 16(4), 494–591.

Stankevičiūtė, K., Alaa, A. M., & van der Schaar, M. (2021). Conformal Time-Series Forecasting. *Advances in Neural Information Processing Systems*, 34.

Chapter 7, global models

Montero-Manso, P., & Hyndman, R. J. (2021). Principles and algorithms for forecasting groups of time series: Locality and globality. *International Journal of Forecasting*, 37(4), 1632–1653.

Salinas, D., Flunkert, V., Gasthaus, J., & Januschowski, T. (2020). DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 36(3), 1181–1191.

Bandara, K., Bergmeir, C., & Smyl, S. (2020). Forecasting across time series databases using recurrent neural networks on groups of similar series: A clustering approach. *Expert Systems with Applications*, 140, 112896.

Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2022). M5 accuracy competition: Results, findings and conclusions. *International Journal of Forecasting*, 38(4), 1346–1364.

Chapters 8 and 9, transformers and foundation models

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention Is All You Need. *Advances in Neural Information Processing Systems*, 30.

Nie, Y., Nguyen, N. H., Sinthong, P., & Kalagnanam, J. (2023). A Time Series is Worth 64 Words: Long-term Forecasting with Transformers. *ICLR 2023*. (Patching.)

Garza, A., & Mergenthaler-Canseco, M. (2023). TimeGPT-1. arXiv:2310.03589.

Ansari, A. F., Stella, L., Turkmen, C., Zhang, X., Mercado, P., Shen, H., et al. (2024). Chronos: Learning the Language of Time Series. arXiv:2403.07815.

Woo, G., Liu, C., Kumar, A., Xiong, C., Savarese, S., & Sahoo, D. (2024). Unified Training of Universal Time Series Forecasting Transformers. *ICML 2024*. (MOIRAI and the LOTSA archive.)

Das, A., Kong, W., Sen, R., & Zhou, Y. (2024). A decoder-only foundation model for time-series forecasting. *ICML 2024*. (TimesFM.)

Liu, Y., Qin, G., Shi, X., Hu, Z., Wang, J., & Long, M. (2025). Sundial: A Family of Highly Capable Time Series Foundation Models. arXiv:2502.00816.

Aksu, T., Woo, G., Liu, J., Liu, X., Liu, C., Savarese, S., Xiong, C., & Sahoo, D. (2024). GIFT-Eval: A Benchmark for General Time Series Forecasting Model Evaluation. arXiv:2410.10393.

Amazon Web Services (2025). *Chronos-2* model card and technical report. Hugging Face: `amazon/chronos-2`.

Salesforce AI Research (2025). *Moirai 2.0* model card. Hugging Face: `Salesforce/moirai-2.0-R-small`.

Chapter 10, real-world challenges

Croston, J. D. (1972). Forecasting and Stock Control for Intermittent Demands. *Operational Research Quarterly*, 23(3), 289–303.

Syntetos, A. A., & Boylan, J. E. (2005). The accuracy of intermittent demand estimates. *International Journal of Forecasting*, 21(2), 303–314. (Croston-SBA.)

Kourentzes, N. (2014). On intermittent demand model optimisation and selection. *International Journal of Production Economics*, 156, 180–190.

Liu, F. T., Ting, K. M., & Zhou, Z.-H. (2008). Isolation Forest. *Proceedings of ICDM 2008*, 413–422.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. *CVPR 2016*, 770–778. (ResNet-50, used for visual similarity transfer.)