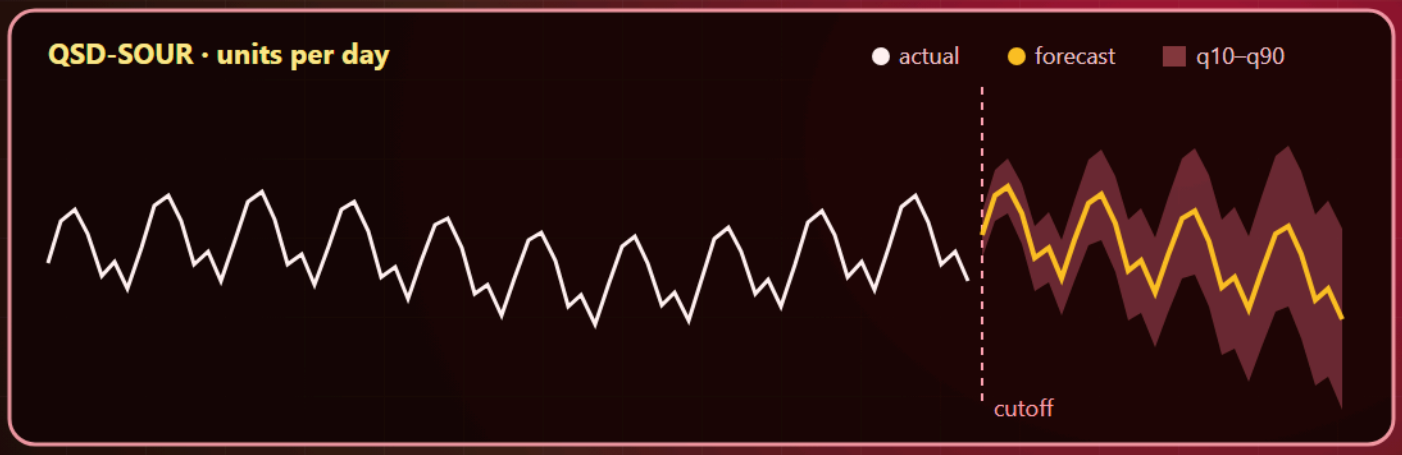
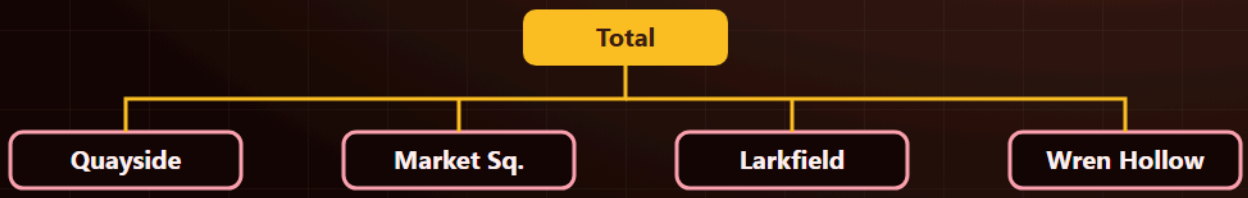



```
from bramble.gbm import gbm_forecast
feat = make_features(panel)
fc = gbm_forecast(feat,
    cutoff="2025-11-30",
    horizon=28)
# one global model, 24 series
```

BACKTEST	MASE
global boosting	0.77
ETS	0.78
seasonal naive	1.06

6 folds × 28 days



A PRACTICAL, CODE-FIRST GUIDE

Time Series Forecasting with Machine Learning

Predict demand with pandas, statsmodels, scikit-learn, and PyTorch

Baselines · ETS · SARIMAX · Backtesting · Gradient Boosting
Global Models · LSTM · Quantiles · Reconciliation · Production

STATISTICS · MACHINE LEARNING · DEEP LEARNING

THENEXBOOKS

Time Series Forecasting with Machine Learning

Predict Demand with pandas, statsmodels, scikit-learn, and PyTorch
From baselines and ARIMA to gradient boosting, LSTMs, probabilistic forecasts, and production pipelines

First Edition

THENEXBOOKS

Time Series Forecasting with Machine Learning
Predict Demand with pandas, statsmodels, scikit-learn, and PyTorch

First Edition, 2026

Copyright © 2026 thenexbooks (thenexbooks.gumroad.com). All rights reserved.

No part of this publication may be reproduced, distributed, or transmitted in any form or by any means without the prior written permission of the publisher, except for brief quotations in reviews and certain other non-commercial uses permitted by copyright law. Code examples may be used freely in your own projects.

The information in this book is provided "as is" without warranty of any kind. While every precaution has been taken in its preparation, the publisher assumes no responsibility for errors or omissions, or for damages resulting from the use of the information contained herein. Python libraries change over time; function names, defaults, and results can differ between versions and between hardware. Every metric in this book comes from the book's own synthetic data and code; your numbers on real data will differ, and a forecast is never a guarantee. Validate any model on your own data before you base business decisions on it.

Python is a trademark of the Python Software Foundation. PyTorch is a trademark of The Linux Foundation. pandas, NumPy, scikit-learn, statsmodels, Matplotlib, and other names mentioned are trademarks or project names of their respective owners. Their use in this book is for identification only and does not imply endorsement. This book is independent and is not affiliated with, sponsored by, or endorsed by any of these projects or organizations.

Bramblecrest Bakery, the four-shop bakery used as the running example in this book, is fictional, as are its shops, products, people, promotions, and all sales data. The data is synthetic and generated by a seeded script included with the book's code; any resemblance to a real business is coincidental.

Preface

Every business makes forecasts, whether it admits it or not. A bakery decides at four in the morning how many loaves to bake. A shop decides how much stock to order, a call center how many people to schedule, a warehouse how much space to keep free. When those numbers come from gut feeling or "same as last week", the result is predictable: bread in the bin on quiet days, empty shelves on busy ones, and nobody able to say whether the planning is getting better or worse.

Forecasting is the discipline of making those numbers explicit, measurable, and steadily better. Modern Python gives you an unusually strong toolkit for it: pandas for dated data, statsmodels for the classical methods that still win many contests, scikit-learn for gradient boosting on engineered features, and PyTorch for neural networks. The hard part is not calling `fit()`. It is framing the problem, avoiding the subtle ways a model can peek at the future, choosing honest metrics, comparing models fairly, expressing uncertainty, and turning a forecast into a decision someone can act on.

This book teaches all of that with one realistic project that grows from the first chapter to the last. Every model is evaluated on the same data and the same test window, so you can see exactly what each idea buys you, and where a simpler method is good enough.

Who This Book Is For

- **Python developers** who are asked to "add a forecast" to a product or a report and want to do it properly.
- **Data analysts and data scientists** who know pandas and basic machine learning but have not yet worked with time-ordered data, where ordinary cross-validation quietly gives wrong answers.
- **Engineers and planners** in retail, food service, logistics, and operations who need demand forecasts they can explain and trust.
- **Students** who want a practical, code-first path through classical statistics, machine learning, and deep learning for time series.

You should be comfortable with basic Python and have used pandas a little. We explain the statistics and the machine learning as we go; no prior forecasting experience is assumed.

What You Will Learn

- How to frame a forecasting problem: target, horizon, granularity, and the decision the forecast supports.
- How to load, clean, resample, and visualize time series with pandas, including closed days and missing data.
- How to decompose a series into trend and seasonality, read autocorrelation, and build baselines that keep you honest.

- How to measure accuracy with MAE, RMSE, MAPE, sMAPE, WAPE, and MASE, and when each one misleads.
- How to forecast with exponential smoothing, ETS, ARIMA, and SARIMAX with holidays, promotions, and Fourier terms.
- How to validate with train, validation, and test windows and rolling-origin backtests instead of random splits.
- How to engineer leak-free features and forecast with gradient boosting, including one global model for many series.
- How to build sequence-to-sequence LSTM and small Transformer forecasters in PyTorch, and when they are worth it.
- How to produce prediction intervals with quantile models and conformal prediction, and check their calibration.
- How to keep shop, product, and total forecasts consistent with hierarchical reconciliation.
- How to detect anomalies, forecast intermittent and new-product demand, turn forecasts into bake plans, and run, monitor, and retrain a forecasting pipeline in production.

How This Book Is Organized

Part I — Foundations (Chapters 1–5) introduces forecasting, time series in pandas, visualization, decomposition, baselines, and accuracy metrics. **Part II — Statistical Forecasting** (Chapters 6–10) covers exponential smoothing, ARIMA, SARIMAX with external regressors, validation and backtesting, and model diagnostics. **Part III — Machine Learning Forecasting** (Chapters 11–15) reframes forecasting as supervised learning, engineers features, and builds local and global gradient boosting models and multi-step strategies. **Part IV — Deep Learning** (Chapters 16–19) builds LSTM and Transformer forecasters in PyTorch and puts them in perspective. **Part V — Uncertainty and Structure** (Chapters 20–23) covers probabilistic forecasts, conformal prediction, hierarchical reconciliation, and anomalies. **Part VI — Forecasting in Production** (Chapters 24–28) covers intermittent and new-product demand, decisions, pipelines, deployment, monitoring, and a complete case study. The appendices provide a cheat sheet, templates and reference tables, a production checklist, and a glossary.

Each chapter ends with exercises and a **Key Takeaways** box for quick review.

The Running Example: Bramblecrest Bakery

Throughout the book we build the forecasting system for **Bramblecrest Bakery**, a fictional bakery with four shops — Quayside, Market Square, Larkfield, and the newer Wren Hollow — each selling six products: sourdough loaves, baguettes, seeded rye, butter croissants, cinnamon rolls, and blueberry muffins. That makes 24 daily demand series with weekly and yearly seasonality, holidays when the shops close or get busy, promotions, a shop with a short history, and a few data problems of the kind

real point-of-sale exports always have. Every morning the bakers need to know how much to bake for the next four weeks; every chapter brings that plan closer. The data is synthetic and generated by a seeded script, so every number in the book can be reproduced exactly on your own computer.

Versions and Conventions

The code uses **Python 3.12** with recent versions of pandas, NumPy, statsmodels, scikit-learn, Matplotlib, and PyTorch (CPU build). All examples run on an ordinary laptop CPU; each script finishes in a few minutes or less, and we limit PyTorch and scikit-learn to two threads so results are stable and the machine stays responsive. A GPU is not required. Library APIs change between versions; where a detail is version-sensitive, a Note says so. Dates are written in ISO form in code and data (`2025-12-24`) and as "December 24, 2025" in prose; numbers use US formatting such as `1,092` and `8.74`.

> **TIP**
> A recommended practice or a technique that makes your work easier.

> **NOTE**
> Background information, version differences, or a useful aside.

> **WARNING**
> A common mistake, such as a source of data leakage, that can make results look better than they are.

Code appears in `monospace`; a caption above each listing names its file, such as `bramble/metrics.py`. Long lines are wrapped to fit the page. Output blocks show the real output of the book's code on the book's data.

Contents

Preface

PART I — FOUNDATIONS

1 What Forecasting Is: Problems, Horizons, and the Bramblecrest Project

2 Time Series Data with pandas: Dates, Frequencies, and Missing Days

3 Visualizing Time Series: Line, Seasonal, and Calendar Plots

4 Decomposition and Patterns: Trend, Seasonality, STL, and Autocorrelation

5 Baselines and Forecast Accuracy: Naive Methods and Error Metrics

PART II — STATISTICAL FORECASTING

6 Exponential Smoothing: Simple, Holt, Holt-Winters, and ETS

7 Stationarity, Differencing, and ARIMA

8 SARIMAX: Seasonality, Holidays, Promotions, and Fourier Terms

9 Validation Done Right: Time-Based Splits and Rolling-Origin Backtests

10 Diagnosing Statistical Models: Residuals, Information Criteria, and Intervals

PART III — MACHINE LEARNING FORECASTING

11 Forecasting as Supervised Learning: Windows, Lags, and Leakage

12 Feature Engineering: Lags, Rolling Windows, Calendar, Holidays, and Promotions

13 Gradient Boosting Forecasts with scikit-learn

14 Global Models: One Model for Many Series

15 Multi-Step Strategies: Recursive, Direct, and Multi-Output

PART IV — DEEP LEARNING

16 Neural Networks for Sequences in PyTorch

17 LSTM Forecasting: A Sequence-to-Sequence Model with Covariates

18 Attention and Small Transformers for Time Series

19 Deep Learning in Practice: Scaling, Tuning, and When It Pays Off

PART V — UNCERTAINTY AND STRUCTURE

20 Probabilistic Forecasts: Intervals, Quantiles, and Pinball Loss

21 Conformal Prediction and Calibration

22 Hierarchical Forecasting and Reconciliation

23 Anomalies, Outliers, and Data Quality

PART VI — FORECASTING IN PRODUCTION

24 Intermittent and New-Product Demand: Croston, Cold Starts, and Short Histories

25 From Forecast to Decision: Bake Plans, Service Levels, and Business Value

26 Building a Forecasting Pipeline: Structure, Configuration, and Tests

27 Deploying and Monitoring Forecasts: Batch Runs, APIs, Drift, and Retraining

28 Case Study: The Bramblecrest Forecasting System End to End

APPENDICES

A Forecasting Cheat Sheet

B Templates and Reference

C Production Checklist

D Glossary

PART I

Foundations

What forecasting is and how to frame a forecasting problem, working with dated data in pandas, seeing patterns with charts, decomposing a series into trend and seasonality, and the baselines and accuracy metrics every later model must beat.

1 · What Forecasting Is

2 · Time Series Data with pandas

3 · Visualizing Time Series

4 · Decomposition and Patterns

5 · Baselines and Forecast Accuracy

CHAPTER 1 · PART I — FOUNDATIONS

What Forecasting Is: Problems, Horizons, and the Bramblecrest Project

Before we fit a single model, we need to know what question a forecast answers, how far ahead it must look, and what decision it feeds. This chapter frames the problem, introduces Bramblecrest Bakery and its 24 demand series, sets up your Python environment, and produces the data and the first forecast that the rest of the book improves on.

A Forecast Is a Statement About an Unknown Future Number

A forecast is a statement, made today, about a quantity we will only observe later. "Quayside will sell about 110 sourdough loaves next Saturday" is a forecast. So is "there is an 80% chance that next Saturday's sales fall between 91 and 132 loaves." Both are claims about the same unknown number; the second is simply more honest about how unsure we are.

That definition sounds obvious, but it rules out several things people often call forecasting:

- **A target is not a forecast.** "We want to sell 120 loaves on Saturday" is a goal. Mixing goals into forecasts makes it impossible to tell later whether the model or the wish was off.
- **A plan is not a forecast.** "We will bake 125 loaves" is a decision. It should *use* a forecast, but it also depends on costs, oven capacity, and how bad it is to run out. We keep the forecast and the decision separate so that each can be checked on its own terms.
- **An explanation is not a forecast.** A model that explains last year's sales very well can still forecast next month badly. In forecasting, the only fair test is how a method does on data it has not seen, and it must not have used any information that would not have been available at the time.

Throughout the book we keep coming back to that last point. A forecast is only as good as the information it could legally have used at the moment it was made. Most of the subtle bugs in forecasting code, which we will meet under the name *leakage*, come from a model quietly seeing a little bit of the future.

Point forecasts and probabilistic forecasts

A **point forecast** is a single number per future period: 110 loaves next Saturday. It hides one important fact: the real outcome will almost never be exactly 110.

A **probabilistic forecast** describes a range or a whole distribution. In practice it usually comes as a few **quantiles** ("10% chance of fewer than 81 loaves, 90% chance of fewer than 131") or as a **prediction interval** ("80% interval: 91 to 132"). Probabilistic forecasts are what a planner actually

needs, because the decision depends on how costly each kind of error is. If unsold bread is cheap and an empty shelf is expensive, the right bake quantity sits above the middle of the distribution, not at it. We build point forecasts first, because they are simpler and because every probabilistic method starts from one, and then spend Part V on intervals, quantiles, and calibration.

Horizon, granularity, frequency, and lead time

Four words describe the shape of almost any forecasting problem. It is worth pinning them down early, because changing any one of them changes which methods work.

Term	Meaning	Bramblecrest answer
Granularity	The level of detail of one forecast value: which item, which place, which time bucket	Units sold per product, per shop, per day
Horizon	How far ahead we forecast	28 days
Frequency	How often we produce a new forecast	Once a week, every Sunday night
Lead time	The gap between making the forecast and the first moment its numbers are used	A few hours: Monday's bake starts before dawn

Granularity is a trade-off. Finer granularity (per shop, per product, per day) is what operations need, but each individual series is noisier, so it is harder to forecast. Coarser granularity (total units per week) is smoother and easier, but you cannot bake from a weekly company total. Part V shows how to forecast at several levels and keep them consistent.

The **horizon** matters because errors grow the further you look ahead, and because some information simply does not exist yet for distant days. If we forecast 28 days ahead, then for the last day of the horizon the most recent sales we know are 28 days old. Every machine learning feature we build later must respect that, which is why the shortest lag in the book's gradient boosting model is exactly 28 days.

The **frequency** of the forecast run determines how quickly the system reacts to change. A weekly run means a sudden shift in demand on a Tuesday is not reflected until the following Sunday night. For a bakery, that is acceptable for planning ingredient orders and staff, and the head baker can still adjust the daily bake by judgment.

Lead time is the reason we cannot use "today's sales" to forecast today. Bramblecrest's ovens start long before the first customer walks in; anything that happens after the forecast run is invisible to the forecast.

The decision behind every forecast

The single most useful question to ask before building a forecasting system is: *what decision will this number change?* The answer tells you the granularity, the horizon, the metric, and how much accuracy is worth paying for. A forecast that changes no decision is a report, not a forecast, and it does

not deserve a weekly pipeline.

For Bramblecrest the decision is concrete: how many loaves, croissants, rolls, and muffins each shop bakes each morning, and how much flour, butter, and blueberries to order ahead of that. Baking too much means unsold units are donated at the end of the day, which costs the ingredients and labor. Baking too little means customers leave without the product they came for, which costs the profit on each missed sale and, over time, some goodwill. Chapter 25 turns this trade-off into a formula and shows that the best bake quantity is a particular *quantile* of the forecast distribution, not the average.

> **NOTE**
>
> Forecasting vocabulary varies between fields. Statisticians say *prediction interval*, machine learning practitioners often say *uncertainty estimate*, and supply chain teams talk about *safety stock*. Appendix D, the Glossary, lists the terms we use, with common synonyms where they exist.

Meet Bramblecrest Bakery

Our running example is **Bramblecrest Bakery**, a fictional bakery with four shops. The owner, **Elsie Carrow**, has noticed two problems that pull in opposite directions: on quiet weekdays a lot of bread is donated at closing time, and on busy Saturdays the croissants are gone by mid-morning. She asked **Nadia Okonjo**, the bakery's data analyst, to build a forecasting system. The person who will use the numbers every week is **Mireille Dunford**, the head baker and production planner, who turns the forecast into a bake plan.

You will work alongside Nadia throughout the book. Every chapter adds a piece of her system, and every model is tested on the same data and the same weeks, so you can see exactly what each idea buys.

The shops

shop_id	Name	Opened	Character
QSD	Quayside	2023-01-02	Waterfront, busiest shop, strong summer peak, busier weekends, growing
MKT	Market Square	2023-01-02	Town center, mild winter peak
LRK	Larkfield	2023-01-02	Suburban, smallest of the original three, flat
WRN	Wren Hollow	2024-03-04	Newest shop, ramp-up after opening, short history

The first three shops have been trading since the start of the data, January 2, 2023. Wren Hollow opened on March 4, 2024, so it has only 665 days of history by the end of the data. That short history is a gift for teaching: it is exactly the situation where some methods struggle and others shine, and

Chapter 14 and Chapter 24 come back to it.

The products

product_id	Name	Category	Price	Cost
SOUR	Sourdough Loaf	bread	$7.50	$2.10
BAGU	Baguette	bread	$3.80	$0.95
RYE	Seeded Rye	bread	$6.90	$2.00
CROI	Butter Croissant	pastry	$3.60	$1.10
CINN	Cinnamon Roll	pastry	$4.20	$1.30
MUFF	Blueberry Muffin	pastry	$3.40	$0.90

Unsold units are donated at the end of the day, so they have no salvage value. The gap between price and cost is large for every product, which already hints at the answer of Chapter 25: running out is more expensive than baking one too many, so the bake plan should lean slightly high.

Twenty-four series and one flagship

Four shops times six products gives **24 daily series**, the "bottom level" of Bramblecrest's data. Each one has an identifier that joins the shop and the product with a hyphen, such as `QSD-SOUR` for Quayside sourdough or `WRN-CINN` for Wren Hollow cinnamon rolls. When we explain a method on a single series, we always use the flagship series **QSD-SOUR**, Quayside's sourdough loaf. It is busy, strongly weekly, has a summer peak and an upward trend, reacts to holidays and promotions, and contains enough noise to keep every model humble.

Above the 24 bottom series sit natural totals: each shop, each product, and the whole bakery. Elsie looks at the bakery total, the shop managers look at their shop, and Mireille needs the bottom level. Chapter 22 makes sure those numbers add up.

The business question

Nadia's brief from Elsie fits in one sentence, and we will quote it often:

"How many of each product should each shop bake every day for the next four weeks?"

From that sentence, Nadia derives the design decisions that every chapter follows:

- The **target** is units sold per series per day.
- The **horizon** is **28 days**, starting the Monday after the run.
- The forecast runs every **Sunday night** after the shops close, so the last known day is always a Sunday.
- Days on which the bakery is **closed** (New Year's Day, Thanksgiving, Christmas Day) are forecast as zero by rule, not by a model.

- The calendar of holidays and promotions is known in advance and may be used for the future days. Sales on future days may not.

> **WARNING**
>
> "Units sold" is not quite the same as "demand". When a shop sells out at 11 a.m., the sales record says 120, but more customers would have bought if there had been more bread. Real bakeries face this *censoring* problem, and it biases forecasts downward over time if the bake plan is always tight. Bramblecrest's synthetic data records demand directly, which keeps the book's examples clean; with real data, record stock-outs (time of sell-out, or units baked) so you can detect censored days later.

What Makes Demand Forecastable?

Some things can be forecast well, and some cannot be forecast at all. Bakery demand sits comfortably in the first group, and it helps to say why, because the same reasoning tells you what to expect from any new forecasting problem.

- **We understand the drivers.** People buy more bread on Saturdays, before holidays, and when it is on promotion. Factors we understand can usually be measured and fed to a model.
- **There is plenty of history.** Three years of daily data covers every weekday about 156 times and every holiday three times. That is enough to estimate weekly patterns very well and yearly patterns roughly.
- **The future resembles the past.** A bakery's customers do not change their habits overnight. When they do, because a new shop opens or a road is closed for works, our models will be wrong for a while, and we need monitoring to notice.

Some parts of Bramblecrest's demand remain genuinely unpredictable. A sudden rainy Saturday, a festival on the quay that nobody told the bakery about, or a broken oven will never appear in a model built from sales history and a calendar. We cannot remove that noise. We can only measure how large it is and plan around it, which is exactly what probabilistic forecasts do.

The most useful habit for judging forecastability is to compare every model against a simple **baseline**. If a sophisticated model cannot beat "same as last week", either the series is mostly noise or the model is doing something wrong. Both are worth knowing.

The Model Ladder

The book climbs a ladder of methods. Each rung is more flexible than the one below, and each must justify its extra complexity with better accuracy on data it has not seen. Sometimes it does, sometimes it does not, and the book shows both outcomes honestly.

1. **Baselines** (Chapter 5): naive, seasonal naive, moving and seasonal averages, drift. Cheap, transparent, and surprisingly hard to beat.

2. **Statistical models** (Chapters 6 to 10): exponential smoothing and ETS, ARIMA, and SARIMAX with holidays, promotions, and yearly Fourier terms. One model per series, with a long track record and built-in intervals.
3. **Machine learning** (Chapters 11 to 15): forecasting reframed as supervised learning with lag, rolling, calendar, and promotion features, solved with gradient boosting. One *global* model can learn from all 24 series at once.
4. **Deep learning** (Chapters 16 to 19): a sequence-to-sequence LSTM and a small Transformer in PyTorch, both using known future covariates.
5. **Uncertainty and structure** (Chapters 20 to 23): quantiles, conformal intervals, hierarchical reconciliation, and anomaly detection.
6. **Production** (Chapters 24 to 28): intermittent demand, decisions and bake plans, pipelines, deployment, monitoring, and an end-to-end case study.

Every model is evaluated with the same splits: a **validation window** (November 3 to 30, 2025) for choosing between options, a **test window** (December 1 to 28, 2025) used once at the end, and a six-fold rolling-origin backtest that tells us how stable the ranking is. Chapter 9 explains why we need all three.

A preview of where we end up

Here is the leaderboard on the December 2025 test window, with all 24 series and all 28 days. You do not need to understand the metrics yet; Chapter 5 defines them. MAE is the average absolute error in units per series per day, WAPE is the total absolute error as a percentage of total sales, and a MASE below 1 means "better than the seasonal-naive benchmark measured on the training history". Lower is better in every column.

Model	MAE	WAPE (%)	MASE
Global gradient boosting (GBM)	**8.706**	**18.134**	**0.742**
ETS (exponential smoothing)	9.174	19.110	0.780
Local GBM (one per series)	9.203	19.170	0.794
Seasonal average (4 weeks)	9.392	19.563	0.811
LSTM	9.935	20.696	0.836
Small Transformer	10.972	22.856	0.937
Seasonal naive	12.170	25.351	1.033

Three lessons are already visible. First, a well-built global gradient boosting model wins on this data. Second, the margin is modest: a careful statistical model and even a four-week seasonal average are not far behind. Third, the neural networks, at the tiny scale of 24 series and about three years of

history, do not beat boosting. Each number in that table is earned in a later chapter, with code you can run, and Chapter 9 checks the ranking on six more windows so we do not crown a winner on one lucky month.

Accuracy is not the final goal, though. In Chapter 25 we convert forecasts into bake plans and count the money: baking from "last week's sales" earns $91,074.70 over the four test weeks, baking from the global GBM point forecast earns $95,637.05, and baking at the 0.72 quantile of a quantile model earns $96,237.20, even though it throws away more bread. The right forecast depends on what the errors cost.

> **NOTE**
>
> Every number in this book describes Bramblecrest's synthetic data and nothing else. It does not tell you that gradient boosting beats LSTMs by some fixed percentage in general, or that ETS is always close behind. On your data the ranking may differ, which is exactly why the book teaches you to run the comparison yourself.

Setting Up Your Environment

All code in this book runs on an ordinary laptop CPU with Python 3.12. No GPU is needed, and nothing is downloaded from the internet: the data is generated on your machine by a seeded script.

Python and packages

Install Python 3.12 from python.org (or your operating system's package manager). On Windows, the `py` launcher lets you pick the version explicitly, which is how we write commands in this book; on macOS and Linux, use `python3.12` instead of `py -3.12`. Create a project folder, copy the book's sample code into it, and create a virtual environment so the book's packages do not interfere with anything else on your machine:

Terminal (Windows)

```
cd bramblecrest
py -3.12 -m venv .venv
.venv\Scripts\activate
python -m pip install -r requirements.txt
```

Terminal (macOS / Linux)

```
cd bramblecrest
python3.12 -m venv .venv
source .venv/bin/activate
python -m pip install -r requirements.txt
```

The `requirements.txt` file pins the versions the book was verified with. Recent versions of pandas, NumPy, statsmodels, scikit-learn, Matplotlib, and PyTorch are needed; much older ones will fail in places, because pandas in particular has renamed several frequency aliases and changed some defaults in recent releases.

requirements.txt

```
# Versions the book's code was verified with (Python 3.12, Windows, CPU only).
# Newer versions usually work; exact numbers in the book may differ slightly
# with other versions.
numpy==2.5.3
pandas==3.0.6
scipy==1.18.1
statsmodels==0.15.0
scikit-learn==1.9.1
matplotlib==3.11.2
python-dateutil==2.9.0.post0
pytest==9.1.1
# PyTorch CPU build: install from https://pytorch.org for your platform
torch==2.14.1
```

PyTorch is only needed from Chapter 16 onward. Install the CPU build for your platform following the instructions on pytorch.org; the GPU build works too, but nothing in this book needs it.

> **TIP**
>
> Check what you actually have installed with `python -m pip show pandas statsmodels scikit-learn`. If a result in your terminal differs from the book in the last decimal place, a different library version is the most likely reason. If it differs a lot, re-run the data generator first: most surprises come from stale data files.

The project folder

The book's code is a small Python package called `bramble` plus one numbered script per topic. You always run scripts from the project folder, so that `import bramble` works without installing anything.

```
bramblecrest/
  _common.py              thread caps, UTF-8 output, save(name, obj) -> results/
  bramble/                the package we build, chapter by chapter
    config.py             seed, dates, splits, shops, products, series ids
    calendar_events.py    holiday calendar and promotion schedule
    generate.py           the seeded synthetic data generator
    data.py               loaders: CSV files -> a complete daily panel
    metrics.py            MAE, RMSE, MAPE, sMAPE, WAPE, MASE, pinball loss
    baselines.py          naive, seasonal naive, averages, drift
    stats_models.py       STL, ETS, SARIMAX helpers
    features.py, gbm.py   machine learning features and gradient boosting
    lstm.py, transformer.py
    probabilistic.py, backtest.py, hierarchy.py, anomalies.py, intermittent.py
  01_generate_data.py ... 14_bake_plan.py
```

```
  data/                   CSV files written by 01_generate_data.py
  results/                JSON results written by the numbered scripts
  tests/test_core.py      fast unit tests
```

The file `bramble/config.py` is the single source of truth for the project's constants. It holds the random seed (2025), the date range, the 28-day horizon, the weekly season length of 7, the validation and test windows, and the list of 24 series identifiers. When a later chapter says "the test window", it means `TEST_START` to `TEST_END` in this file.

bramble/config.py (excerpt)

```
SEED = 2025
START = pd.Timestamp("2023-01-02")          # a Monday
END = pd.Timestamp("2025-12-28")            # a Sunday -> 156 full weeks, 1,092 days
HORIZON = 28                                # we always forecast 28 days ahead
SEASON = 7                                  # weekly season length for daily data

TRAIN_END = pd.Timestamp("2025-11-02")      # fit here for the validation window
VALID_START = pd.Timestamp("2025-11-03")
VALID_END = pd.Timestamp("2025-11-30")
TEST_START = pd.Timestamp("2025-12-01")     # refit through VALID_END for the test window
TEST_END = pd.Timestamp("2025-12-28")

SHOP_IDS = list(SHOPS)
PRODUCT_IDS = list(PRODUCTS)
SERIES_IDS = [f"{s}-{p}" for s in SHOP_IDS for p in PRODUCT_IDS]  # 24 bottom series
FLAGSHIP = "QSD-SOUR"                   # single-series examples
```

The helper `_common.py` is imported first by every numbered script. It caps the number of threads that NumPy and scikit-learn use, switches standard output to UTF-8 so printing works the same on every operating system, and provides a `save()` function that writes results to `results/` as JSON.

_common.py

```
import json
import os
import sys

# Shared PC: cap BLAS/OpenMP threads (scikit-learn, numpy) BEFORE they are imported.
for var in ("OMP_NUM_THREADS", "OPENBLAS_NUM_THREADS", "MKL_NUM_THREADS"):
    os.environ.setdefault(var, "2")

from bramble.config import RESULTS_DIR  # noqa: E402

sys.stdout.reconfigure(encoding="utf-8")


def save(name: str, obj) -> None:
    RESULTS_DIR.mkdir(exist_ok=True)
    with open(RESULTS_DIR / f"{name}.json", "w", encoding="utf-8") as f:
        json.dump(obj, f, indent=2, default=str)
```

Why cap threads? Two reasons. Results from some numerical libraries can differ in the last digits depending on how work is split across threads, so a fixed thread count makes runs more repeatable. And a forecasting job usually shares a machine with other work; there is no reason for it to grab every core.

Generating the Data

Bramblecrest does not exist, so its sales history comes from a generator: a Python function that simulates daily demand with a fixed random seed. Chapter 2 prints the core of the generator and explains each ingredient. For now, run it:

01_generate_data.py

```
"""Step 1: generate the Bramblecrest Bakery CSV files (seed 2025)."""
import _common  # noqa: F401  (first: thread caps, UTF-8 stdout)
from bramble.config import DATA_DIR
from bramble.generate import generate_data

sales = generate_data()
print(f"Wrote {len(sales):,} sales rows to {DATA_DIR}")
print(sales.head(6).to_string(index=False))
```

Terminal

```
py -3.12 01_generate_data.py
```

Output

```
Wrote 23,465 sales rows to ...\bramblecrest\data
      date shop_id product_id  units_sold
2023-01-02     LRK       BAGU          20
2023-01-02     LRK       CINN          19
2023-01-02     LRK       CROI          53
2023-01-02     LRK       MUFF          13
2023-01-02     LRK        RYE           8
2023-01-02     LRK       SOUR          36
```

The script writes five CSV files into `data/`: `sales.csv` (the point-of-sale export, one row per shop, product, and trading day), `shops.csv`, `products.csv`, `holidays.csv`, and `promotions.csv`. Because the seed is fixed, the files are byte-identical every time you run the script, on any machine with compatible library versions. If you ever doubt that your data matches the book, delete the `data/` folder and run the script again.

Notice that the rows are sorted by date, then by shop and product in alphabetical order, which is why Larkfield (LRK) appears first. Notice too what is *not* there: there are no rows for days when the bakery was closed, and there are a few rows missing because, as in real life, the point-of-sale export occasionally failed. Chapter 2 separates those two situations carefully, because confusing them is one of the most common mistakes in demand data.

The data covers January 2, 2023 (a Monday) through December 28, 2025 (a Sunday): exactly 156 weeks, or **1,092 days**. Across all shops, products, and days, Bramblecrest sold 1,116,951 units: 312,662 in 2023, 385,972 in 2024, and 418,317 in 2025 up to December 28. Part of the growth comes from Wren Hollow opening in 2024, and part from Quayside's steady growth.

Finally, check that the unit tests pass. They take a few seconds and confirm that your environment computes the metrics, baselines, calendars, and reconciliation the same way the book does:

Terminal

```
py -3.12 -m pytest -q tests
```

Output

```
.....                                                                    [100%]
5 passed in ...s
```

> **NOTE**
>
> The data is synthetic. That is a deliberate choice, not a shortcut. Synthetic data lets everyone reproduce every number exactly, avoids any privacy or licensing question, and, most usefully, lets us know the truth. We know, for example, that the generator makes Christmas Eve bread sales 70% higher than a normal day of the same kind, so in Chapter 8 we can check whether a model actually learned that. The price is realism: real sales data is messier, with changing product ranges, price changes, weather, and competitors. The methods transfer; the exact numbers do not.

Your First Forecast

Let us make a forecast before we know any theory. The simplest sensible rule for a bakery is "next week will look like last week": Monday's forecast is last Monday's sales, Tuesday's is last Tuesday's, and so on. Repeated four times, that gives a 28-day forecast. Forecasters call it the **seasonal naive** method, with a season of 7 days.

We forecast Quayside sourdough for the test window, December 1 to 28, 2025, using only data up to Sunday, November 30. We use a helper, `clean_history`, that Chapter 2 explains properly: it returns the series up to a cutoff date with closed days filled in from the same weekday in the two previous weeks.

ch01_first_forecast.py

```
"""Chapter 1: a first forecast for Quayside sourdough - "same as last week"."""
import _common  # noqa: F401  (first: thread caps, UTF-8 stdout)
import numpy as np
import pandas as pd

from bramble.backtest import clean_history
```