Turn data into reliable, real-world data science systems.

Advanced Data Modeling and Forecasting with Python extends data modeling and machine learning into more complex and realistic settings. Building on foundational workflows, this book introduces advanced techniques for handling temporal data, uncovering hidden structure, addressing real-world data challenges, and deploying models into production environments.

Rather than focusing on isolated methods, it presents a unified perspective on designing, evaluating, and maintaining complete data science systems.

- Analyze and forecast time series data using statistical and machine learning approaches
- Discover structure in data through clustering and representation learning
- Handle imbalanced and noisy datasets in practical scenarios
- Combine models using advanced and hybrid modeling techniques
- Deploy models using APIs and production pipelines
- Monitor performance, detect drift, and maintain models over time
- Apply advanced modeling techniques to business and financial problems

About the Author

Dr. Shouke Wei is a researcher, scientist, and entrepreneur specializing in data analysis and modeling, wavelet-based signal processing, and AI-driven applications.

He earned his Ph.D. from Brandenburg University of Technology Cottbus–Senftenberg (Germany) and conducted postdoctoral research at Eawag (Switzerland). He has held research positions at the University of British Columbia (Canada) and served as a distinguished and adjunct professor at multiple universities in China.

His work focuses on bridging theoretical modeling with practical, real-world data science systems.

DEEPSIM PRESS

9 781067 559267

Part of the ***Practical Data Science with Python*** series

DEEPSIM PRESS

Advanced Data Modeling and Forecasting with Python

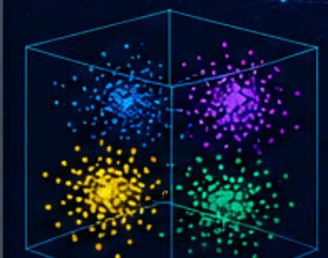
SHOUKE WEI

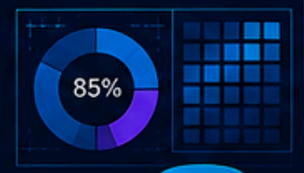
DEEPSIM PRESS

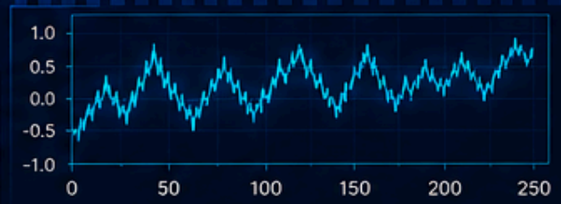
Advanced Data Modeling and Forecasting with Python

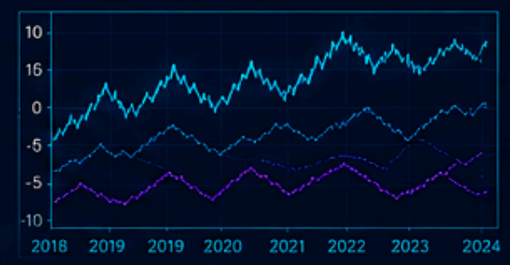
Time Series, Advanced Modeling, and Real-World Systems

A practical framework for real-world data science systems









SHOUKE WEI

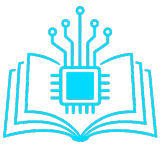
Advanced Data Modeling and Forecasting with Python

Time Series, Advanced Modeling, and Real-World Systems

Advanced Data Modeling and Forecasting with Python

Time Series, Advanced Modeling, and Real-World Systems

Shouke Wei



DEEPSIM
PRESS

Advanced Data Modeling and Forecasting with Python

First Edition, 2026

For permission requests, contact the publisher:
Email: shouke.wei@deepsim.ca

ISBN 978-1-0675592-7-4 (Hardcover)
ISBN 978-1-0675592-8-1 (Paperback)
ISBN 978-1-0675592-6-7 (eBook)
DOI 10.5281/zenodo.20043592

Published by
Deepsim Press
An independent imprint of Deepsim Intelligence Technology Inc.
Abbotsford, British Columbia, Canada
https://press.deepsim.ca

This book is intended for educational and professional readers.

For resources, updates, and companion code, visit:
https://press.deepsim.ca/advanced-model

About the Author

Shouke Wei, Ph.D., is a researcher, scientist, and entrepreneur specializing in intelligent IoT systems, robotics, big data analytics, modeling and forecasting, early-warning systems, and edge computing. With academic and industry experience across Europe, North America, and Asia, Dr. Wei is recognized for bridging advanced theory with real-world, production-ready systems.

Dr. Wei earned his Ph.D. in Environmental and Resource Management from the Department of Ecosystems and Environmental Informatics at Brandenburg University of Technology Cottbus–Senftenberg (Germany). He conducted postdoctoral research at the Swiss Federal Institute of Aquatic Science and Technology (Eawag, Switzerland) and held research positions at the University of British Columbia (Canada). He has also served as a distinguished and adjunct professor at multiple institutions in China.

Dr. Wei currently serves as CEO and Chief Scientist of Deepsim Intelligent Technology Inc. (Canada) and Shandong Deepsim Intelligent Technology Co., Ltd. (China). He is also a Postdoctoral Co-Supervisor at the Shandong Postdoctoral Innovation Practice Base and Director of Qilu Artificial Intelligence and Digital Manufacturing Innovation at Shandong Deepsim Intelligent Technology Co., Ltd., China.

Dr. Wei has led or contributed to 19 major international research projects and the development of numerous intelligent systems, including autonomous water-quality monitoring vessels, AI-based environmental early-warning platforms, medical image diagnosis systems, precision agriculture robots, and autonomous service robots.

His scholarly contributions include over 40 peer-reviewed publications, 8 books on practical wavelet transform applications, data analysis, and data modeling, more than 500 technical tutorial articles, six patents, and over 30 software copyrights. His work focuses on making advanced computational methods—particularly data analysis, modeling, and wavelet-based signal processing—accessible, practical, and impactful for researchers and practitioners worldwide.

For more information, visit: https://press.deepsim.ca/shouke/

Contents

Preface

In the previous volume, *Practical Data Modeling and Machine Learning with Python*, the focus was on building and evaluating models. We explored how to transform data into features, apply statistical and machine learning methods, and assess model performance using principled validation strategies.

Those steps form the foundation of data science. However, in practice, they are only the beginning.

Real-world modeling problems rarely conform to the assumptions of standard workflows. Data may evolve over time, exhibit hidden structure, or suffer from imbalance and noise. Models that perform well in controlled settings often degrade when exposed to dynamic environments. Deploying a model introduces additional challenges, including integration, monitoring, and continuous adaptation.

This book addresses these realities.

Purpose of This Book

The aim of this volume is to extend the modeling process beyond isolated techniques and toward **complete, real-world systems**.

Rather than focusing on individual algorithms, the book emphasizes:

- modeling under temporal dependence
- discovering structure in unlabeled data
- handling imperfect and imbalanced datasets
- combining methods into hybrid approaches
- deploying models into operational environments

- applying modeling techniques to real-world domains

In this sense, the transition from the first volume to this one is a shift in perspective:

> From building models to **designing modeling systems**.

What This Book Covers

This book is organized into eight parts, each addressing a key extension of the modeling framework.

Part I — From Models to Systems introduces the broader perspective required for advanced data science. It examines the limitations of standard modeling assumptions and outlines how modeling fits into larger, dynamic systems.

Part II — Time Series and Forecasting focuses on data with temporal structure. It covers foundational concepts, classical models such as ARIMA and SARIMA, and modern machine learning approaches to forecasting.

Part III — Unsupervised Learning and Representation explores techniques for discovering structure without labeled data, including clustering, dimensionality reduction, and representation learning methods such as autoencoders.

Part IV — Handling Real-World Data Challenges addresses practical issues that frequently arise in applied settings, with particular emphasis on imbalanced data and its impact on evaluation and model performance.

Part V — Advanced and Hybrid Modeling examines strategies for combining models and integrating statistical and machine learning approaches to achieve improved performance and flexibility.

Part VI — Deployment and Production Systems moves beyond model development to operational considerations, including model packaging, API construction, deployment pipelines, monitoring, and model maintenance.

Part VII — Business Applications demonstrates how modeling techniques are applied in practice, with examples in business decision-making, financial forecasting, and customer segmentation.

Part VIII — End-to-End Framework synthesizes the material into a unified perspective, providing a practical reference for designing, evaluating, and maintaining complete data science systems.

Approach and Philosophy

This book maintains a practical orientation, but extends it to more complex settings. The focus is not only on how to apply methods, but on how to **adapt them to realistic conditions**.

Throughout the book, emphasis is placed on:

- choosing methods appropriate to the structure of the data
- designing validation strategies that reflect real-world conditions
- understanding the limitations of models and metrics
- maintaining clarity and interpretability in increasingly complex systems

The goal is to develop judgment as much as technique.

Prerequisites

This volume assumes familiarity with the material covered in the previous books, including:

- data preparation and feature engineering
- basic statistical modeling
- core machine learning methods
- model evaluation and validation techniques

Readers should also be comfortable working with Python and common data science libraries. The focus here is not on introducing tools, but on applying them in more advanced contexts.

Final Remarks

As models become more sophisticated, the challenges shift from implementation to design, evaluation, and integration.

The central idea of this book is simple:

> *effective data science requires not only good models, but well-designed systems.*

By extending the modeling framework into more realistic and demanding settings, this book aims to provide the tools and perspective needed to move from isolated models to reliable, real-world solutions.

Shouke Wei, PhD
Deepsim Intelligent Technology Inc.
Deepsim Academy
Abbotsford, Canada
April 28, 2026

Acknowledgments

This book is the continuation of a longer journey in data modeling and machine learning—one that extends from foundational techniques to more complex and realistic applications.

I would like to acknowledge the broader scientific and engineering communities whose work has shaped the field. The ideas presented in this book build upon decades of research in statistics, machine learning, and applied data science. Many of the methods discussed here are the result of collective efforts across academia and industry.

I am especially grateful to the open-source community. The Python ecosystem—including libraries for numerical computing, machine learning, and model deployment—has made it possible to develop, test, and share practical solutions at scale. These tools not only enable experimentation but also support the transition from models to real-world systems.

This book has also been influenced by the role of teaching and ongoing practice and reflection. Working with real data, encountering limitations, and revisiting assumptions have all contributed to the perspective presented here. In particular, the emphasis on workflow, evaluation, and system design comes from recognizing that successful modeling depends as much on process as on individual techniques.

Finally, I would like to thank my family for their continued support and understanding throughout the writing process. Their patience has made it possible to complete this work.

Notation and Abbreviations

Symbol / Abbreviation	Meaning
Supervised learning fundamentals	
$\mathbf{x}$	Input feature vector for a single observation
y	Target (response) variable for a single observation
$\hat{y}$	Predicted value of the target produced by a model
$P(y \mid \mathbf{x})$	Conditional distribution of the target given features
$\mathbf{X}$	Feature matrix of shape $n \times p$ (rows: observations, columns: features)
n	Number of observations (sample size)
p	Number of input features
Distributional shift	
$P_{\text{train}}(\mathbf{x})$	Feature distribution during model training
$P_{\text{prod}}(\mathbf{x})$	Feature distribution observed in production
$P_{\text{train}}(y)$	Target distribution during model training
$P_{\text{prod}}(y)$	Target distribution observed in production
Model evaluation	
$\mathcal{L}(\cdot)$	Loss function used during training (e.g., cross-entropy, MSE)
$M^*(f)$	Optimal evaluation metric for model f: $M^*(f) = \mathbb{E}_{\mathcal{D}_{\text{test}}}[L(y,\, f(\mathbf{x}))]$; the metric should approximate the expected business loss L
$L(y, f(\mathbf{x}))$	Business loss function: the actual cost incurred when the model predicts $f(\mathbf{x})$ and the true value is y

Symbol / Abbreviation	Meaning
Acc	Classification accuracy: fraction of correctly classified observations
AUROC	Area under the receiver operating characteristic curve
F1	Harmonic mean of precision and recall
Data and indices	
t	Index over time steps
T	Total number of observations in the series
h	Forecast horizon (number of steps ahead)
H	Maximum forecast horizon
m	Seasonal period (e.g., $m = 12$ for monthly annual seasonality)
y_t	Observed value of the time series at time t
$\hat{y}_{T+h}$	Point forecast for horizon h steps ahead of time T
ε_t	White-noise innovation at time t: $\varepsilon_t \sim \mathrm{WN}(0, \sigma^2)$
σ^2	Innovation variance
Time series components	
T_t	Trend component at time t
S_t	Seasonal component at time t
R_t	Residual (irregular) component at time t
Differencing and integration	
∇	First-difference operator: $\nabla y_t = y_t - y_{t-1}$
∇_m	Seasonal difference operator: $\nabla_m y_t = y_t - y_{t-m}$
d	Order of regular differencing in ARIMA/SARIMA
D	Order of seasonal differencing in SARIMA
B	Backshift (lag) operator: $By_t = y_{t-1}$, $B^k y_t = y_{t-k}$
ARIMA model components	
p	Order of the autoregressive (AR) polynomial
q	Order of the moving average (MA) polynomial
ϕ_j	AR coefficient at lag j ($j = 1, \dots, p$)
θ_j	MA coefficient at lag j ($j = 1, \dots, q$)
$\phi(B)$	AR polynomial: $\phi(B) = 1 - \phi_1 B - \cdots - \phi_p B^p$

Symbol / Abbreviation	Meaning
$\theta(B)$	MA polynomial: $\theta(B) = 1 + \theta_1 B + \cdots + \theta_q B^q$
c	Constant / intercept term in ARIMA
ℓ	Maximised log-likelihood of a fitted model; used in AIC and BIC (note: also used as look-back window length in the LSTM context — see below)
SARIMA seasonal extensions	
P	Order of the seasonal AR polynomial
Q	Order of the seasonal MA polynomial
$\Phi_P(B^m)$	Seasonal AR polynomial: $1 - \Phi_1 B^m - \cdots - \Phi_P B^{Pm}$
$\Theta_Q(B^m)$	Seasonal MA polynomial: $1 + \Theta_1 B^m + \cdots + \Theta_Q B^{Qm}$
Φ_j	Seasonal AR coefficient at lag jm
Θ_j	Seasonal MA coefficient at lag jm
Stationarity	
μ	(Constant) mean of a stationary process
γ_k	Autocovariance at lag k: $\gamma_k = \mathrm{Cov}(y_t, y_{t-k})$
γ_0	Variance of the process: $\gamma_0 = \mathrm{Var}(y_t)$
ρ_k	Autocorrelation at lag k: $\rho_k = \gamma_k / \gamma_0$
ϕ_{kk}	Partial autocorrelation at lag k (coefficient in AR(k) regression)
Δy_t	First difference of y_t: $\Delta y_t = y_t - y_{t-1}$; dependent variable in the ADF regression
Forecast evaluation	
e_{T+h}	Forecast error at horizon h: $e_{T+h} = y_{T+h} - \hat{y}_{T+h}$
MAE	Mean absolute error: $H^{-1} \sum_{h=1}^{H} \lvert e_{T+h} \rvert$
RMSE	Root mean squared error: $\sqrt{H^{-1} \sum_{h=1}^{H} e_{T+h}^2}$
MAPE	Mean absolute percentage error: $100 H^{-1} \sum_{h=1}^{H} \lvert e_{T+h} / y_{T+h} \rvert$
Machine learning for time series	
L	Maximum lag order for lag-feature construction
w	Rolling-window size for rolling statistics

Symbol / Abbreviation	Meaning
X_t	Feature vector at time t (lags, rolling stats, calendar variables)
$f(\cdot)$	Trained regression model (e.g., LightGBM or LSTM)
K	Number of folds in walk-forward cross-validation
hour_sin_t	Cyclic sine encoding of the hour-of-day: $\sin(2\pi \cdot \text{hour}_t/24)$
hour_cos_t	Cyclic cosine encoding of the hour-of-day: $\cos(2\pi \cdot \text{hour}_t/24)$
dow_sin_t	Cyclic sine encoding of the day-of-week: $\sin(2\pi \cdot \text{dow}_t/7)$
dow_cos_t	Cyclic cosine encoding of the day-of-week: $\cos(2\pi \cdot \text{dow}_t/7)$
LSTM architecture	
ℓ	Look-back window length (sequence length fed to the LSTM)
H_{hidden}	LSTM hidden state size
$\mathbf{h}_t$	Hidden state of the LSTM at time step t
$\mathbf{c}_t$	Cell state of the LSTM at time step t
Clustering	
k	Number of clusters
C_c	Cluster c ($c = 1, \dots, k$)
$\boldsymbol{\mu}_c$	Centroid of cluster c: $\boldsymbol{\mu}_c = \frac{1}{\lvert C_c \rvert} \sum_{\mathbf{x}_i \in C_c} \mathbf{x}_i$
$\text{WCSS}(k)$	Total within-cluster sum of squares: $\sum_{c=1}^{k} \sum_{\mathbf{x}_i \in C_c} \lVert \mathbf{x}_i - \boldsymbol{\mu}_c \rVert^2$
SS_B	Between-cluster sum of squares
SS_W	Within-cluster sum of squares
$s(i)$	Silhouette score for observation i: $(b(i) - a(i))/\max\{a(i), b(i)\}$
$a(i)$	Mean intra-cluster distance for observation i
$b(i)$	Mean distance from observation i to its nearest neighbouring cluster
$\bar{s}$	Mean silhouette score over all observations
$\text{CH}(k)$	Calinski-Harabász index: $[\text{SS}_B/(k-1)]/[\text{SS}_W/(n-k)]$

Symbol / Abbreviation	Meaning
$\mathrm{DB}(k)$	Davies-Bouldin index: $\frac{1}{k}\sum_{c=1}^{k}\max_{c'\neq c}\frac{\bar{d}_c+\bar{d}_{c'}}{d(\boldsymbol{\mu}_c,\boldsymbol{\mu}_{c'})}$
$\bar{d}_c$	Mean intra-cluster distance for cluster c
Hierarchical clustering linkage	
$d(A,B)_{\text{single}}$	Single-linkage distance: $\min_{i\in A,\, j\in B}\|\mathbf{x}_i-\mathbf{x}_j\|$
$d(A,B)_{\text{complete}}$	Complete-linkage distance: $\max_{i\in A,\, j\in B}\|\mathbf{x}_i-\mathbf{x}_j\|$
$d(A,B)_{\text{average}}$	Average-linkage (UPGMA) distance: $\frac{1}{\lvert A\rvert\lvert B\rvert}\sum_{i\in A}\sum_{j\in B}\|\mathbf{x}_i-\mathbf{x}_j\|$
Consensus clustering	
M_{ij}	Co-occurrence (consensus) matrix entry: proportion of bootstrap runs in which observations i and j are assigned to the same cluster
Dimensionality reduction	
$\mathbf{S}$	Sample covariance matrix: $\mathbf{S}=\frac{1}{n-1}\mathbf{X}^\top\mathbf{X}$
$\mathbf{V}$	Matrix of principal component directions (eigenvectors of $\mathbf{S}$)
$\mathbf{V}_d$	First d eigenvectors (columns of $\mathbf{V}$)
$\mathbf{\Lambda}$	Diagonal matrix of eigenvalues: $\mathbf{\Lambda}=\mathrm{diag}(\lambda_1,\ldots,\lambda_p)$
λ_j	j-th eigenvalue of the covariance matrix ($\lambda_1\geq\lambda_2\geq\cdots\geq\lambda_p\geq 0$)
$\mathbf{Z}$	PCA score matrix (projected data): $\mathbf{Z}=\mathbf{X}\mathbf{V}_d$
$\lambda_j/\sum_{j'}\lambda_{j'}$	Proportion of Variance Explained (PVE) by the j-th principal component
p_{ij}	Symmetric joint neighbourhood probability in high-dimensional space (t-SNE): $p_{ij}=(p_{j\lvert i}+p_{i\lvert j})/(2n)$
q_{ij}	Neighbourhood probability in the low-dimensional embedding (t-SNE)
σ_i	Bandwidth of the Gaussian kernel centred on observation i (t-SNE)
$\mathcal{L}_{\text{t-SNE}}$	t-SNE objective: $\mathrm{KL}(P\|Q)=\sum_{i\neq j}p_{ij}\log(p_{ij}/q_{ij})$
w_{ij}	High-dimensional graph edge weight between observations i and j (UMAP)

Symbol / Abbreviation	Meaning
v_{ij}	Low-dimensional graph edge weight between observations i and j (UMAP)
$\mathcal{L}_{\text{UMAP}}$	UMAP cross-entropy loss: $\sum_{(i,j)\in E}\left[w_{ij}\log\frac{w_{ij}}{v_{ij}}+(1-w_{ij})\log\frac{1-w_{ij}}{1-v_{ij}}\right]$
Autoencoders and anomaly detection	
f_θ	Encoder function mapping input to latent code: $f_\theta : \mathbb{R}^p \to \mathbb{R}^d$
g_ϕ	Decoder function mapping latent code to reconstruction: $g_\phi : \mathbb{R}^d \to \mathbb{R}^p$
θ	Learnable parameters of the encoder
ϕ	Learnable parameters of the decoder
$\mathbf{z}$	Latent code (bottleneck representation): $\mathbf{z} = f_\theta(\mathbf{x})$
$\sigma(\cdot)$	Non-linear activation function; ReLU in hidden layers ($\sigma(x) = \max(0, x)$) and sigmoid in the output layer ($\sigma(x) = 1/(1+e^{-x})$)
$\mathcal{L}(\theta, \phi)$	Autoencoder reconstruction loss: $\frac{1}{n}\sum_{i=1}^{n}\|\mathbf{x}_i - g_\phi(f_\theta(\mathbf{x}_i))\|^2$
r_i	Per-observation reconstruction error: $r_i = \|\mathbf{x}_i - \hat{\mathbf{x}}_i\|^2$; used as a continuous anomaly score
$s(i, n)$	Isolation Forest anomaly score: $2^{-\mathbb{E}[h(i)]/c(n)}$
$\text{lrd}_k(i)$	Local reachability density of observation i with respect to its k nearest neighbours (LOF)
$\text{LOF}_k(i)$	Local Outlier Factor score for observation i
$N_k(i)$	Set of k nearest neighbours of observation i
Imbalanced classification	
π_1	Minority class proportion: $\pi_1 = n_{\text{minority}}/n$
Precision	Positive predictive value: $\text{TP}/(\text{TP}+\text{FP})$
Recall	Sensitivity / true positive rate: $\text{TP}/(\text{TP}+\text{FN})$
F_β	Weighted harmonic mean of precision and recall: $(1+\beta^2)\cdot\frac{\text{Precision}\cdot\text{Recall}}{\beta^2\cdot\text{Precision}+\text{Recall}}$

Symbol / Abbreviation	Meaning
β	Relative weight of recall over precision in F_β; $\beta = 1$ gives F_1, $\beta = 2$ weights recall twice as heavily
MCC	Matthews Correlation Coefficient: $\frac{\text{TP}\cdot\text{TN}-\text{FP}\cdot\text{FN}}{\sqrt{(\text{TP}+\text{FP})(\text{TP}+\text{FN})(\text{TN}+\text{FP})(\text{TN}+\text{FN})}}$
AP	Average Precision (area under the precision-recall curve); equals π_1 for the no-skill baseline
τ	Decision threshold applied to predicted probabilities: $\hat{y} = \mathbb{1}[\hat{p} \geq \tau]$
τ^*	Optimal decision threshold maximising a target metric (e.g., F_1 or F_β)
$\hat{p}$	Predicted probability of the positive class: $\hat{p} = P(y = 1 \mid \mathbf{x})$
w_c	Class weight for class c; under `balanced` weighting: $w_c = n/(n_c \cdot C)$
C	Number of classes
SMOTE and resampling	
$\mathbf{x}_{\text{syn}}$	Synthetic minority observation generated by SMOTE: $\mathbf{x}_{\text{syn}} = \mathbf{x}_i + \lambda(\mathbf{x}_{i'} - \mathbf{x}_i)$
λ	Interpolation coefficient in SMOTE: $\lambda \sim \text{Uniform}(0, 1)$
Hybrid forecasting	
$\hat{T}_{T+h}$	STL trend forecast extrapolated h steps ahead
$\hat{S}_{T+h}$	STL seasonal forecast extrapolated h steps ahead
$\hat{R}^{\text{GB}}_{T+h}$	Gradient boosting prediction of the STL residual component h steps ahead
Stacking ensembles	
M	Number of base learners in a stacking ensemble
f_m	Base model m in the stacking ensemble $(m = 1, \ldots, M)$
$\tilde{y}_i^{(m)}$	Out-of-fold prediction for observation i from base model m
$\mathcal{D}$	Full training dataset

Symbol / Abbreviation	Meaning
$\mathcal{D}_{\text{test}}$	Held-out test dataset used for a single, unrepeatable final evaluation of out-of-sample performance
Wavelet transform	
$\psi(t)$	Mother wavelet function: a short oscillatory function with zero mean
cA_j	Approximation coefficients at level j: output of the low-pass filter capturing coarse-scale behaviour
cD_j	Detail coefficients at level j: output of the high-pass filter capturing fine-scale fluctuations
Drift detection	
PSI	Population Stability Index: $\sum_{b=1}^{B}(p_b^{\text{prod}} - p_b^{\text{ref}})\ln(p_b^{\text{prod}}/p_b^{\text{ref}})$; values < 0.1 negligible, ≥ 0.2 significant
D_n	Kolmogorov-Smirnov statistic: $\sup_x \lvert F^{\text{ref}}(x) - F^{\text{prod}}(x)\rvert$, where F denotes an empirical CDF
W	Rolling window size (number of requests) used to compute rolling accuracy, F1, and ECE
ECE	Expected Calibration Error: $\sum_{b=1}^{B} \frac{\lvert B_b\rvert}{n} \lvert \overline{p}_b - \overline{y}_b\rvert$
CLV prediction and business regression	
ϕ_{ij}	SHAP value of feature j for observation i: the average marginal contribution of feature j to the prediction, averaged over all feature orderings
$\mathbb{E}[f(\mathbf{x})]$	Global mean prediction of the model across the training set; baseline (intercept) in the SHAP decomposition
R^2	Coefficient of determination: proportion of target variance explained by the model; $R^2 = 1 - \text{SS}_{\text{res}}/\text{SS}_{\text{tot}}$
sMAPE	Symmetric MAPE: $\frac{200}{n}\sum_{i=1}^{n} \frac{\lvert y_i - \hat{y}_i\rvert}{\lvert y_i\rvert + \lvert \hat{y}_i\rvert}$
Financial time series and GARCH	

Symbol / Abbreviation	Meaning
r_t	Daily log-return at time t: $r_t = \log(P_t/P_{t-1})$, where P_t is the closing price
σ_t^2	Conditional variance (GARCH) at time t: the model's estimate of return variance given information up to $t-1$
σ_t	Conditional volatility (standard deviation) at time t; annualised by multiplying by $\sqrt{252}$
ω	Long-run variance intercept in GARCH: $\omega > 0$; determines the unconditional variance floor
α	ARCH coefficient in GARCH(1,1): weight on the lagged squared innovation ε_{t-1}^2
β	GARCH coefficient in GARCH(1,1): weight on the lagged conditional variance σ_{t-1}^2
S	Sharpe ratio of a trading strategy: $S = \bar{r}_{\text{strategy}}/\sigma_{\text{strategy}}$; annualised by multiplying by $\sqrt{252}$ for daily data
$\text{CV}_{\text{WF}}(f)$	Walk-forward cross-validation loss: $\frac{1}{T-t_0}\sum_{t=t_0}^{T-1} L(y_{t+1},\, f_t(\mathbf{x}_{t+1}))$
Gap statistic (cluster count selection)	
W_k	Total within-cluster dispersion for a k-cluster solution: $W_k = \sum_{c=1}^{k} \frac{1}{2\lvert C_c\rvert}\sum_{i,i' \in C_c} \lVert \mathbf{x}_i - \mathbf{x}_{i'} \rVert^2$
$\text{Gap}(k)$	Gap statistic: $\mathbb{E}^*[\log W_k] - \log W_k$; optimal k is the smallest satisfying $\text{Gap}(k) \geq \text{Gap}(k+1) - s_{k+1}$
s_{k+1}	Standard error of $\mathbb{E}^*[\log W_{k+1}]$ across Monte Carlo replicates
Segmentation validation	
$H(C_c)$	Shannon entropy of cluster c with respect to an external categorical variable: $H(C_c) = -\sum_v p_{cv} \log p_{cv}$
NMI	Normalised Mutual Information: $2 \cdot I(\text{clusters}; \text{labels})/[H(\text{clusters}) + H(\text{labels})]$; ranges from 0 to 1

Symbol / Abbreviation	Meaning
Abbreviations	
ACF	Autocorrelation function
ADF	Augmented Dickey-Fuller (unit-root test)
ADASYN	Adaptive Synthetic Sampling (density-weighted minority oversampling)
AE	Autoencoder
AIC	Akaike Information Criterion: $-2\ell + 2k$
AR	Autoregressive (model)
ARI	Adjusted Rand Index (measure of agreement between two clusterings, corrected for chance)
ARIMA	Autoregressive Integrated Moving Average
API	Application Programming Interface
AUROC	Area Under the Receiver Operating Characteristic curve
BIC	Bayesian Information Criterion: $-2\ell + k \log T$
CI / CD	Continuous Integration / Continuous Deployment
CLV	Customer Lifetime Value: total revenue attributed to a customer over a defined future window
CV	Cross-Validation
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
DWT	Discrete Wavelet Transform
ECE	Expected Calibration Error
ENN	Edited Nearest Neighbours (majority-class boundary cleaning)
GARCH	Generalised Autoregressive Conditional Heteroscedasticity: $\sigma_t^2 = \omega + \alpha\varepsilon_{t-1}^2 + \beta\sigma_{t-1}^2$
GAN	Generative Adversarial Network
GJR-GARCH	Glosten-Jagannathan-Runkle GARCH: extends GARCH with an asymmetric leverage term
GMM	Gaussian Mixture Model
i.i.d.	Independent and identically distributed
KNN	K-Nearest Neighbours
KS	Kolmogorov-Smirnov (two-sample distributional test)

Symbol / Abbreviation	Meaning
LGBM	LightGBM (gradient-boosted decision trees)
LOF	Local Outlier Factor (neighbourhood-based anomaly detection)
LR	Logistic Regression
LSTM	Long Short-Term Memory (recurrent neural network)
MA	Moving Average (model)
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
MASE	Mean Absolute Scaled Error: MAE of the model divided by MAE of the naïve seasonal benchmark
MCC	Matthews Correlation Coefficient
MLP	Multilayer Perceptron
MLOps	Machine Learning Operations
MSE	Mean Squared Error
NMI	Normalised Mutual Information
OLS	Ordinary Least Squares
OOF	Out-of-Fold (predictions generated on held-out folds to train a meta-learner)
PACF	Partial autocorrelation function
PCA	Principal Component Analysis
PR-AUC	Area under the Precision-Recall curve (synonym: Average Precision)
PSI	Population Stability Index
PVE	Proportion of Variance Explained (by a principal component)
RF	Random Forest
RFM	Recency, Frequency, Monetary: a compact customer behaviour representation derived from transaction history
RMSE	Root Mean Squared Error
ROC	Receiver Operating Characteristic curve
SARIMA	Seasonal Autoregressive Integrated Moving Average
SHAP	SHapley Additive exPlanations

Symbol / Abbreviation	Meaning
sMAPE	Symmetric Mean Absolute Percentage Error
SMOTE	Synthetic Minority Oversampling Technique
SMOTEENN	SMOTE oversampling followed by Edited Nearest Neighbours cleaning
STL	Seasonal and Trend decomposition using Loess
t-SNE	t-distributed Stochastic Neighbour Embedding
UMAP	Uniform Manifold Approximation and Projection
VAE	Variational Autoencoder
VaR	Value at Risk: the loss threshold not exceeded with probability $1 - \alpha$ over a given horizon

Part I

Introduction and Perspective

1 From Models to Real-World Systems

In many practical settings, building a model is not the most difficult step. Fitting a logistic regression, training a gradient-boosted tree, or fine-tuning a neural network has become a routine activity, supported by mature libraries and abundant tutorials. The real challenge lies elsewhere: ensuring that the model remains reliable when applied to data that is evolving, incomplete, or structurally different from the data used during development.

A model that achieves 96% accuracy on a held-out test set, but was validated with random splitting on time-ordered data, may perform far worse in production. A classifier trained on a balanced benchmark dataset may fail catastrophically when the true positive rate in deployment is one in a thousand. A pipeline that runs perfectly in a Jupyter notebook may produce silent errors when serialised, reloaded, and served six months later by a different team.

These failures are not caused by choosing the wrong algorithm. They arise from a mismatch between the **assumptions embedded in the modeling workflow** and the **conditions that hold in practice**. Recognising and resolving that mismatch is what distinguishes applied data science from academic benchmarking.

This chapter introduces the perspective required to move from individual models to **complete modeling systems** — the conceptual foundation on which every chapter in this book builds.

1.1 Chapter Overview

This chapter covers the following topics:

- When Standard Assumptions Break Down
- Modeling as a System
- Extending the Workflow
- Scope of This Book
- Companion Resources and Project Configuration

By the end of this chapter, you will be able to:

- Identify the three core assumptions of classical supervised learning and articulate the conditions under which each is violated in practice
- Describe the five-component modeling system — data pipelines, training, validation, evaluation, and deployment — and explain how failures in one component propagate to others
- Map the chapter topics of this book to the specific workflow extensions they address
- Set up the shared project configuration and reproduce the environment used throughout all examples

1.2 When Standard Assumptions Break Down

Much of classical supervised modeling rests on three simplifying assumptions that are rarely stated explicitly but underpin nearly every standard result:

1. **Independence and identical distribution (i.i.d.)**: observations are drawn independently from the same fixed distribution. This assumption justifies random train-test splitting, cross-validation, and the use of standard error estimates.
2. **Distributional stationarity**: the distribution that generates training data is the same distribution that generates test and production data. This assumption justifies evaluating a model on a held-out set and trusting that performance will generalise.
3. **Metric alignment**: the evaluation metric used during model selection captures the quantity that actually matters to the business or application. This assumption justifies using aggregate metrics like accuracy or AUROC as proxies for real-world value.

In controlled settings — academic benchmarks, well-defined competitions, stable experimental data — these assumptions hold well enough to be useful. In real-world applications, each of them is routinely violated.

1.2.1 Temporal Dependence and the i.i.d. Assumption

When data is collected over time, consecutive observations are rarely independent. Stock prices exhibit autocorrelation. Patient outcomes are correlated within hospitals and within time periods. Sensor readings from industrial equipment follow seasonal and cyclical patterns. In each case, randomly shuffling the data before splitting into train and test sets produces an evaluation that is **optimistically biased**: the model has effectively seen future information during training, making its test performance unrepresentative of its performance on truly unseen data.

The correct approach — time-ordered splitting, or walk-forward validation — is conceptually straightforward but has practical consequences throughout the pipeline: feature engineering must respect the temporal boundary, preprocessing transformers must be fitted only on training data, and performance metrics must be computed on data that was genuinely held out. Chapters 2–4 develop these ideas in depth for time series data.

1.2.2 Distributional Shift

The assumption that training and production data follow the same distribution is violated in three distinct ways, each requiring a different response:

- **Covariate shift**: the distribution of input features $P(\mathbf{x})$ changes, while the conditional $P(y \mid \mathbf{x})$ remains stable. A fraud detection model trained on 2022 transactions may encounter systematically different feature distributions in 2024 as payment behaviours evolve.
- **Label shift**: the marginal distribution of the target $P(y)$ changes without a corresponding change in features. A clinical model may encounter a higher proportion of severe cases post-pandemic than it saw during training.

- **Concept drift**: the relationship $P(y \mid \mathbf{x})$ itself changes. The features that predicted default risk in a low-interest-rate environment may no longer be predictive when rates rise sharply.

Each type of shift degrades model performance in a different way and requires a different monitoring and retraining strategy. Chapter 14 develops the statistical tools — the Population Stability Index, the Kolmogorov-Smirnov test, and rolling performance tracking — for detecting and responding to each type.

1.2.3 Metric Misalignment and Class Imbalance

Accuracy is an appropriate metric when classes are balanced and all errors have equal cost. In many real applications, neither condition holds. A model that predicts "no fraud" on every transaction will achieve 99.8% accuracy on a dataset where fraud represents 0.2% of cases, while being completely useless for its intended purpose. The correct metrics — precision-recall curves, F1 score at a given threshold, cost-sensitive evaluation — depend on the application and must be chosen before model selection, not after.

Class imbalance compounds this problem by distorting the learning process itself: models trained on imbalanced data tend to be biased toward the majority class, not because they are poorly specified but because the loss function implicitly weights majority-class errors more heavily. Chapters 8 and 9 address oversampling, undersampling, cost-sensitive learning, and threshold calibration for imbalanced settings.

1.3 Modeling as a System

A model should not be viewed in isolation. It is a component of a broader system, and its behaviour depends on the behaviour of every other component. The five elements of a complete modeling system are illustrated in Figure 1.1.

```
# ── Visualise the modeling system ──────────────────────────────────────
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
from matplotlib.patches import FancyArrowPatch
import warnings
```

```
warnings.filterwarnings("ignore")

fig, ax = plt.subplots(figsize=(13, 5))
ax.set_xlim(0, 13)
ax.set_ylim(0, 5)
ax.axis("off")

components = [
    (1.2,  2.5, "Data\nPipelines",    "#DBEAFE", "#1E40AF"),
    (3.7,  2.5, "Training &\nValidation", "#D1FAE5", "#065F46"),
    (6.2,  2.5, "Evaluation &\nDiagnostics", "#FEF3C7", "#92400E"),
    (8.7,  2.5, "Deployment\n& Serving",  "#FCE7F3", "#9D174D"),
    (11.2, 2.5, "Monitoring\n& Maintenance", "#F3E8FF", "#5B21B6"),
]

box_w, box_h = 2.0, 1.6
for x, y, label, fc, ec in components:
    rect = mpatches.FancyBboxPatch(
        (x - box_w / 2, y - box_h / 2), box_w, box_h,
        boxstyle="round,pad=0.1", facecolor=fc, edgecolor=ec, linewidth=2
    )
    ax.add_patch(rect)
    ax.text(x, y, label, ha="center", va="center", fontsize=9.5,
            fontweight="bold", color=ec, multialignment="center")

for i in range(len(components) - 1):
    x0 = components[i][0]   + box_w / 2
    x1 = components[i+1][0] - box_w / 2
    y  = 2.5
    ax.annotate("", xy=(x1, y), xytext=(x0, y),
                arrowprops=dict(arrowstyle="->", lw=1.8, color="#374151"))

# Feedback arrow: Monitoring → Training
ax.annotate(
    "", xy=(3.7, 2.5 - box_h / 2 - 0.05),
    xytext=(11.2, 2.5 - box_h / 2 - 0.05),
    arrowprops=dict(arrowstyle="->", lw=1.5, color="#6B7280",
                    connectionstyle="arc3,rad=-0.3")
)
ax.text(7.45, 0.5, "Feedback: drift detection triggers retraining",
        ha="center", va="center", fontsize=8, color="#6B7280", style="italic")
```

```
ax.set_title("The Five-Component Modeling System", fontsize=12,
             fontweight="bold", pad=10)
plt.tight_layout()

from pathlib import Path
FIGURES = Path("images")
FIGURES.mkdir(exist_ok=True)
plt.savefig(FIGURES / "ch01_modeling_system.png", dpi=150,
bbox_inches="tight")
plt.show()
```

Output:

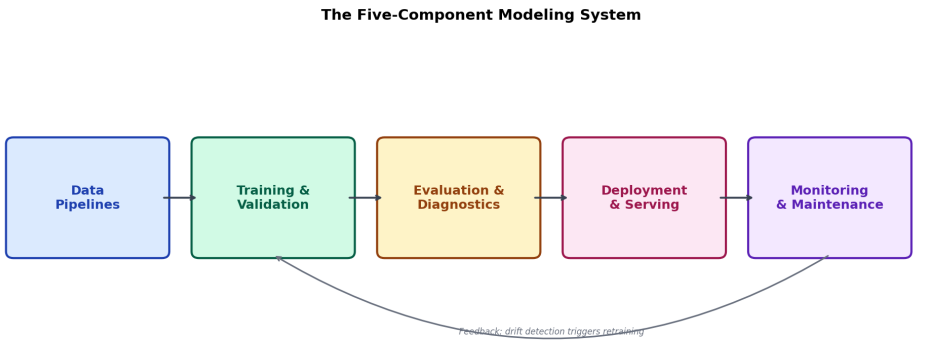


Figure 1.1: The five-component modeling system. Data pipelines transform raw inputs into model-ready features. Training and validation procedures fit and evaluate model candidates. Evaluation and diagnostics confirm that the chosen model meets performance requirements. Deployment and serving expose the model to real-world inputs via an API or batch job. Monitoring and maintenance track model behaviour over time and close the loop by triggering retraining when drift or degradation is detected.

These components interact with each other in ways that are not always obvious:

- A change in **data preprocessing** — rescaling a feature differently, imputing missing values by a new strategy — alters the feature space that the model was trained on and can degrade performance without any change to the model itself.
- A flawed **validation strategy** — random splitting of time-ordered data, target leakage through a preprocessing step applied before the split —

can produce evaluation metrics that overstate real-world performance by a wide margin.
- An unmonitored **deployment** accumulates silent prediction errors as the input distribution drifts away from the training distribution, with no alert triggered until a downstream business metric deteriorates.

Effective modeling requires managing this system as a whole. A change to any one component must be evaluated for its effect on every other.

1.4 Extending the Workflow

The standard supervised learning workflow — collect data, split, fit, evaluate — remains the foundation. But it must be extended to handle the conditions that arise in practice. This book focuses on five key extensions:

Temporal structure introduces ordering and time dependence that violates i.i.d. assumptions. Features must be engineered from lagged values and rolling statistics without lookahead. Train-test splits must respect temporal ordering. Forecasting models must quantify uncertainty over future horizons. Chapters 2–4 develop the full time series toolkit: foundations and decomposition (Chapter 2), classical ARIMA and SARIMA models (Chapter 3), and machine learning approaches to forecasting including LSTM networks and walk-forward validation (Chapter 4).

Unsupervised learning addresses settings where no labeled outcomes are available. Patterns — customer segments, anomalous observations, compact feature representations — must be discovered from the structure of the data alone. Because there is no ground truth, evaluation requires different tools: silhouette scores, elbow analysis, reconstruction error, and qualitative inspection. Chapters 5–7 cover clustering methods and evaluation (Chapter 5), dimensionality reduction and feature learning with PCA (Chapter 6), and autoencoders for representation learning and anomaly detection (Chapter 7).

Data imbalance and noise distort both the learning process and standard evaluation metrics. Oversampling with SMOTE, undersampling, cost-sensitive loss functions, and calibrated threshold selection are not special-case corrections — they are standard components of production pipelines for fraud detection, medical diagnosis, predictive maintenance, and any application where the

positive class is rare and costly to miss. Chapter 8 develops the diagnostic framework for understanding and evaluating imbalanced data; Chapter 9 covers resampling and algorithmic solutions.

Hybrid modeling approaches combine the interpretability and uncertainty quantification of statistical models with the flexibility and scalability of machine learning. Stacking, blending, and residual correction pipelines can outperform either approach alone while remaining auditable. Chapter 10 covers hybrid modeling strategies and Chapter 11 covers advanced techniques for model combination, including stacking architectures and ensemble methods.

Deployment and monitoring close the loop between model development and real-world use. A model that is never deployed provides no value; a model that is deployed but never monitored will silently degrade. Chapters 12–14 develop the complete operational stack: serialisation and packaging (Chapter 12), FastAPI serving and deployment pipelines (Chapter 13), and drift detection with the Population Stability Index and rolling performance tracking (Chapter 14).

1.5 Scope of This Book

The chapters that follow develop these ideas in a structured way:

- time series analysis and forecasting
- clustering and representation learning
- techniques for handling imbalanced data
- advanced and hybrid modeling strategies
- deployment and production systems
- real-world applications and case studies

The final chapter integrates all components into a unified framework for end-to-end data science, revisiting a single real-world problem from raw data collection through deployment and monitoring.

1.5.1 Perspective

As modeling problems become more complex, the emphasis shifts from applying individual methods to **making informed decisions** about how and when to

use them.

This includes:

- selecting methods that match the structure of the data
- designing validation strategies that reflect real-world conditions
- interpreting results with an awareness of limitations

In this context, advanced modeling is not defined by complexity alone, but by clarity, discipline, and the ability to adapt methods to practical constraints.

1.5.2 Prerequisite Knowledge

This book assumes familiarity with Python and the core supervised learning workflow covered in the companion volume. Readers should be comfortable with pandas and scikit-learn at an intermediate level — loading and cleaning tabular data, fitting and evaluating classifiers and regressors, and interpreting confusion matrices and ROC curves. All code examples can be run independently using the resources provided in Section 1.6.

1.6 Companion Resources and Project Configuration

1.6.1 Official Resources

All datasets, notebooks, scripts, and environment configuration files used in this book are available online.

Official resource hub: https://press.deepsim.ca/advanced-model

GitHub repository (code and reproducible workflows): https://github.com/shoukewei/advanced-model

The repository contains:

- all datasets used throughout the chapters
- complete, runnable notebooks and scripts with expected outputs
- environment configuration files (`requirements.txt`, `pyproject.toml`, `uv.lock`)

- a shared configuration module (`config.py`) that manages file paths and global settings across all chapters

1.6.2 Project Structure and Configuration

Every chapter in this book imports a shared `config.py` module that defines the directory structure, random seed, and figure settings used consistently across all examples:

```
# Standard imports used at the top of every chapter ──────────────────────
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from pathlib import Path
import sys, warnings
warnings.filterwarnings("ignore")

sys.path.insert(0, str(Path().resolve().parent))
from config import DATA_RAW, DATA_PROCESSED, FIGURES, RANDOM_SEED, MODELS

# Consistent visual style
sns.set_theme(style="whitegrid", palette="muted")
rng = np.random.default_rng(RANDOM_SEED)

print(f"DATA_RAW       : {DATA_RAW}")
print(f"DATA_PROCESSED : {DATA_PROCESSED}")
print(f"FIGURES        : {FIGURES}")
print(f"MODELS         : {MODELS}")
print(f"RANDOM_SEED    : {RANDOM_SEED}")
```

Output:

```
DATA_RAW       : /path/to/project/data/raw
DATA_PROCESSED : /path/to/project/data/processed
FIGURES        : /path/to/project/images
MODELS         : /path/to/project/models
RANDOM_SEED    : 42
```

The `config.py` file ensures that every example produces the same results regardless of the reader's local directory structure. Readers who prefer to run examples without the repository can substitute absolute paths directly, or set `DATA_RAW = Path(".")` and place data files in the working directory.

1.6.3 Reproducing the Environment

All examples in this book were developed and tested with the package versions specified in `pyproject.toml`. The recommended setup uses `uv` for fast, reproducible environment creation (Wei 2026b, 2026a). To reproduce the environment and run the notebooks, follow these steps:

Download companion resources from the official resource hub (Section 1.6), or clone the GitHub repository, then run in the terminal:

```
# Clone the repository
git clone https://github.com/shoukewei/advanced-model
cd advanced-model

# Create and activate the environment
uv sync

# Launch Jupyter
uv run jupyter lab
```

Alternatively, using pip:

```
pip install -r requirements.txt
```

The key dependencies are scikit-learn, lightgbm, torch, fastapi, pydantic, statsmodels, prophet, and scipy. Specific version pins are provided in the lock file.

1.7 Summary

The transition from individual models to complete modeling systems requires a broader perspective — one that considers not only how models are built, but how they are evaluated, deployed, and maintained over time.

This chapter established the conceptual foundation for the topics that follow:

- **Standard assumptions break down** in real applications through temporal dependence (violating i.i.d.), distributional shift (violating stationarity), and metric misalignment (violating the assumption that accuracy reflects value). Recognising which assumption is violated in a given setting is the prerequisite for choosing the right corrective technique.
- **Modeling is a system** (Section 1.3): data pipelines, training and validation, evaluation and diagnostics, deployment and serving, and monitoring and maintenance interact with each other. A failure in any component can propagate to degrade the others in ways that are not immediately visible.
- **Five workflow extensions** (Section 1.4) address the most common departures from idealised conditions: temporal structure (Chapters 2–4), unsupervised learning (Chapters 5–7), class imbalance and noise (Chapters 8–9), hybrid modeling (Chapters 10–11), and deployment with monitoring (Chapters 12–14). Each extension is developed in depth across the subsequent parts.
- **The companion repository** (Section 1.6) provides all datasets, code, and environment configuration needed to reproduce every example. The shared `config.py` module ensures consistent file paths and random seeds across all chapters.

The chapter topics that follow develop each of these extensions systematically, building toward a unified framework for end-to-end applied data science.

1.8 Exercises

Exercise 1: Consider a model trained to predict whether a loan applicant will default within 12 months, using historical application data from 2019–2021. Identify which of the three core assumptions (i.i.d., distributional stationarity, metric alignment) is most likely violated for each of the following deployment scenarios: (a) the model is deployed in 2022 during a period of rising interest rates; (b) the bank's portfolio has shifted toward small-business loans, which were rare in the training data; (c) the model is evaluated using accuracy on a dataset where 98% of applicants did not default. For each scenario, suggest the part of this book (Part II–VI) that addresses the violation.

Exercise 2: Clone the companion repository and run the environment setup commands. Confirm that the `config.py` paths resolve correctly on your machine by running the configuration import block from Section 1.6 and verifying that the output directories exist. Then navigate to the Chapter 2 notebook and run the first code cell to confirm that the dataset loads without error.

Exercise 3: For each of the five workflow extensions described in Section 1.4 — temporal structure (Part II), unsupervised learning (Part III), imbalance and noise (Part IV), hybrid modeling (Part V), and deployment (Part VI) — identify one real-world domain where that extension is the primary challenge (not one of the examples already given in the chapter). For each domain, describe the specific mechanism by which the standard assumption breaks down, and explain why ignoring it would lead to a misleading model evaluation.

Exercise 4: Draw the five-component modeling system diagram (Figure 1.1) for a specific application of your choice — for example, a churn prediction model for a subscription service, a predictive maintenance system for wind turbines, or a medical image classifier. For each component, identify the concrete implementation (e.g., "Data Pipelines: hourly sensor readings from 500 turbines, preprocessed with a rolling 24-hour window") and one specific failure mode that could arise at that component without propagating an obvious error.

1.9 Quiz

1. The **i.i.d. assumption** in supervised learning is violated most directly by: A. A dataset with a 99:1 class imbalance between negative and positive examples B. A time series dataset where consecutive observations are autocorrelated and random train-test splitting leaks future information into training C. A dataset where the evaluation metric (accuracy) does not reflect the cost of false negatives D. A regression problem where the target variable has a skewed distribution that is not corrected by a log transform
2. **Covariate shift** refers specifically to: A. A change in the conditional $P(y \mid \mathbf{x})$ caused by a new relationship between features and the target B. A change in the marginal distribution of the target $P(y)$ without a corresponding change in features C. A change in the distribution of input

features $P(\mathbf{x})$ while the conditional $P(y \mid \mathbf{x})$ remains stable D. A change in both $P(\mathbf{x})$ and $P(y \mid \mathbf{x})$ simultaneously, requiring full model retraining

3. A model that predicts "no fraud" on every transaction achieves 99.8% accuracy on a dataset where fraud occurs in 0.2% of cases. This failure illustrates which breakdown of standard assumptions? A. Temporal dependence — the fraud patterns have evolved since the training data was collected B. Metric misalignment — accuracy is an inappropriate metric when classes are imbalanced and error costs are asymmetric C. Distributional shift — the test set was drawn from a different population than the training set D. Feature leakage — a preprocessing step has introduced information about the target into the feature set
4. In the five-component modeling system, which component is responsible for detecting that a model's predictions are degrading due to distributional shift — and triggering the retraining feedback loop? A. Data Pipelines B. Training and Validation C. Evaluation and Diagnostics D. Monitoring and Maintenance
5. Which of the following best explains why a **flawed validation strategy** is more dangerous than a poorly chosen model? A. A flawed validation strategy requires more compute to fix, whereas a poorly chosen model can be swapped without reprocessing the data B. A flawed validation strategy produces misleading performance estimates that make an unreliable model appear trustworthy, whereas a poorly chosen model will typically show clearly lower performance in a correctly implemented evaluation C. A flawed validation strategy affects only the training set, whereas a poorly chosen model affects both training and test performance D. A flawed validation strategy is specific to tree-based models and does not affect linear models or neural networks

Answers: 1-B, 2-C, 3-B, 4-D, 5-B

References

Alpaydin, Ethem, and Cenk Kaynak. 1998. "Cascading Classifiers." *Kybernetika* 34 (4): 369–74.

Bailey, David H., Jonathan Borwein, Marcos Lopez de Prado, and Qiji Jim Zhu. 2016. *The Probability of Backtest Overfitting*. In *Journal of Computational Finance*, vol. 20. https://doi.org/10.21314/JCF.2016.322.

Bellman, Richard E. 1957. *Dynamic Programming*. Princeton University Press.

Bollerslev, Tim. 1986. "Generalized Autoregressive Conditional Heteroskedasticity." *Journal of Econometrics* 31 (3): 307–27. https://doi.org/10.1016/0304-4076(86)90063-1.

Box, George E. P., and Gwilym M. Jenkins. 1976. *Time Series Analysis: Forecasting and Control*. Revised. Holden-Day.

Breunig, Markus M., Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. 2000. "LOF: Identifying Density-Based Local Outliers." *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, 93–104. https://doi.org/10.1145/342009.335388.

Candanedo, Luis M., Véronique Feldheim, and Dominique Deramaix. 2017. "Data Driven Prediction Models of Energy Use of Appliances in a Low-Energy House." *Energy and Buildings* 140: 81–97. https://doi.org/10.1016/j.enbuild.2017.01.083.

Chawla, Nitesh V., Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. 2002. "SMOTE: Synthetic Minority over-Sampling Technique." *Journal of Artificial Intelligence Research* 16: 321–57. https://doi.org/10.1

613/jair.953.

Chen, Chao, Andy Liaw, and Leo Breiman. 2004. "Using Random Forest to Learn Imbalanced Data." Technical Report.

Chen, Daqing. 2015. "Online Retail II Data Set." *UCI Machine Learning Repository*.

Chen, Tianqi, and Carlos Guestrin. 2016. *XGBoost: A Scalable Tree Boosting System*. 785–94. https://doi.org/10.1145/2939672.2939785.

Chicco, Davide, and Giuseppe Jurman. 2020. "The Advantages of the Matthews Correlation Coefficient (MCC) over F1 Score and Accuracy in Binary Classification Evaluation." *BMC Genomics* 21 (6). https://doi.org/10.1186/s12864-019-6413-7.

Cook, R. Dennis, and Sanford Weisberg. 1982. *Residuals and Influence in Regression*. Chapman; Hall.

Cortez, Paulo, Antonio Cerdeira, Fernando Almeida, Telmo Matos, and Jose Reis. 2012. "A Proactive Intelligent Decision Support System for Predicting the Popularity of Online News." *UCI Machine Learning Repository*.

Dal Pozzolo, Andrea, Olivier Caelen, Reid A. Johnson, and Gianluca Bontempi. 2015. "Calibrating Probability with Undersampling for Unbalanced Classification." *2015 IEEE Symposium Series on Computational Intelligence*, 159–66. https://doi.org/10.1109/SSCI.2015.33.

Engle, Robert F. 1982. "Autoregressive Conditional Heteroscedasticity with Estimates of the Variance of United Kingdom Inflation." *Econometrica* 50 (4): 987–1007. https://doi.org/10.2307/1912773.

Fader, Peter S., Bruce G. S. Hardie, and Ka Lok Lee. 2005. "'Counting Your Customers' the Easy Way: An Alternative to the Pareto/NBD Model." *Marketing Science* 24 (2): 275–84. https://doi.org/10.1287/mksc.1040.0098.

Fama, Eugene F. 1970. "Efficient Capital Markets: A Review of Theory and Empirical Work." *The Journal of Finance* 25 (2): 383–417. https://doi.org/10.2307/2325486.

Gupta, Sunil, Dominique Hanssens, Bruce Hardie, et al. 2006. "Modeling Customer Lifetime Value." *Journal of Service Research* 9 (2): 139–55. https://doi.org/10.1177/1094670506293810.

Han, Hui, Wen-Yuan Wang, and Bing-Huan Mao. 2005. "Borderline-SMOTE: A New over-Sampling Method in Imbalanced Data Sets Learning." *Advances in Intelligent Computing — ICIC 2005*, 878–87. https://doi.org/10.1007/11538059_91.

Hastie, Trevor, Robert Tibshirani, and Jerome Friedman. 2009. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. 2nd ed. Springer. https://hastie.su.domains/ElemStatLearn/.

He, Haibo, Yang Bai, Edwardo A. Garcia, and Sheng Li. 2008. "ADASYN: Adaptive Synthetic Sampling Approach for Imbalanced Learning." *2008 IEEE International Joint Conference on Neural Networks (IJCNN)*, 1322–28. https://doi.org/10.1109/IJCNN.2008.4633969.

Hennig, Christian. 2015. "What Are the True Clusters?" *Pattern Recognition Letters* 64: 53–62. https://doi.org/10.1016/j.patrec.2015.04.009.

Hochreiter, Sepp, and Jürgen Schmidhuber. 1997. "Long Short-Term Memory." *Neural Computation* 9 (8): 1735–80. https://doi.org/10.1162/neco.1997.9.8.1735.

Jain, Anil K. 2010. *Data Clustering: 50 Years Beyond K-Means*. In *Pattern Recognition Letters*, vol. 31. https://doi.org/10.1016/j.patrec.2009.09.011.

Ke, Guolin, Qi Meng, Thomas Finley, et al. 2017. "LightGBM: A Highly Efficient Gradient Boosting Decision Tree." *Advances in Neural Information Processing Systems* 30. https://proceedings.neurips.cc/paper/2017/hash/6449f44a102fde848669bdd9eb6b76fa-Abstract.html.

Kotler, Philip, and Kevin Lane Keller. 2016. *Marketing Management*. 15th ed. Pearson.

Liu, Fei Tony, Kai Ming Ting, and Zhi-Hua Zhou. 2008. "Isolation Forest." *2008 Eighth IEEE International Conference on Data Mining (ICDM)*, 413–22.

Liu, Xu-Ying, Jianxin Wu, and Zhi-Hua Zhou. 2009. "Exploratory Undersampling for Class-Imbalance Learning." *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 39: 539–50. https://doi.org/10.1109/TSMCB.2008.2007853.

Lo, Andrew W., and A. Craig MacKinlay. 1988. "Stock Market Prices Do Not Follow Random Walks: Evidence from a Simple Specification Test." *The Review of Financial Studies* 1 (1): 41–66. https://doi.org/10.1093/rfs/1.1.41.

Lopez de Prado, Marcos. 2018. *Advances in Financial Machine Learning*. Wiley.

Lundberg, Scott M., and Su-In Lee. 2017. "A Unified Approach to Interpreting Model Predictions." *Advances in Neural Information Processing Systems* 30. https://proceedings.neurips.cc/paper/2017/hash/8a20a8621978632d76c43dfd28b67767-Abstract.html.

Maaten, Laurens van der, and Geoffrey Hinton. 2008. "Visualizing Data Using t-SNE." *Journal of Machine Learning Research* 9 (Nov): 2579–605.

Mallat, Stéphane G. 1989. "A Theory for Multiresolution Signal Decomposition: The Wavelet Representation." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 11 (7): 674–93. https://doi.org/10.1109/34.192463.

McInnes, Leland, John Healy, and James Melville. 2018. "UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction." *arXiv Preprint arXiv:1802.03426*. https://arxiv.org/abs/1802.03426.

New York City Taxi and Limousine Commission. 2023. *TLC Trip Record*

Data. Https://www.nyc.gov/site/tlc/about/tlc-trip-record-data.page.

Oreshkin, Boris N., Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. 2020. "N-BEATS: Neural Basis Expansion Analysis for Interpretable Time Series Forecasting." *International Conference on Learning Representations*. https://openreview.net/forum?id=r1ecqn4YwB.

Salinas, David, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. 2020. "DeepAR: Probabilistic Forecasting with Autoregressive Recurrent Networks." *International Journal of Forecasting* 36 (3): 1181–91. https://doi.org/10.1016/j.ijforecast.2019.07.001.

Tibshirani, Robert, Guenther Walther, and Trevor Hastie. 2001. "Estimating the Number of Clusters in a Data Set via the Gap Statistic." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 63 (2): 411–23. https://doi.org/10.1111/1467-9868.00293.

Tomek, Ivan. 1976. "Two Modifications of CNN." *IEEE Transactions on Systems, Man, and Cybernetics* 6: 769–72. https://doi.org/10.1109/TSMC.1976.4309452.

Wedel, Michel, and Wagner A. Kamakura. 2000. *Market Segmentation: Conceptual and Methodological Foundations*. 2nd ed. Kluwer Academic Publishers.

Wei, Shouke. 2026a. *Practical Data Modeling and Machine Learning with Python: From Data Preparation to Model Evaluation and Optimization*. 1st ed. Deepsim Press. https://doi.org/10.5281/zenodo.19753396.

Wei, Shouke. 2026b. *UV Project Setup Guide*. Https://press.deepsim.ca/uv-project-setup-guide/.

Wilson, Dennis L. 1972. "Asymptotic Properties of Nearest Neighbor Rules Using Edited Data." *IEEE Transactions on Systems, Man, and Cybernetics* 2 (3): 408–21. https://doi.org/10.1109/TSMC.1972.4309137.

Wolpert, David H. 1992. "Stacked Generalization." *Neural Networks* 5 (2):

241–59. https://doi.org/10.1016/S0893-6080(05)80023-1.

Woods, Kevin S., Christopher C. Doss, Kevin W. Bowyer, and Jeffrey L. Solka. 1993. "Comparative Evaluation of Pattern Recognition Techniques for Detection of Microcalcifications in Mammography." *Proceedings of SPIE Medical Imaging* 1905: 123–32.

Index