

Time Series Analysis

Chapter 9: Machine learning for time series



Daniel Traian PELE

Bucharest University of Economic Studies
IDA Institute Digital Assets
Blockchain Research Center
AI4EFin Artificial Intelligence for Energy Finance
Romanian Academy, Institute for Economic Forecasting
MSCA Digital Finance

Acronyms used in this material (1/2)

- **AI**: Artificial Intelligence
- **AR**: AutoRegressive (model)
- **ARIMA**: AutoRegressive Integrated Moving Average (model)
- **AUC**: Area Under the (ROC) Curve
- **BET**: Bucharest Exchange Trading (index)
- **CPU**: Central Processing Unit
- **DHR**: Dynamic Harmonic Regression
- **DM**: Diebold–Mariano (test)
- **ENTSO-E**: European Network of Transmission System Operators for Electricity
- **ES-RNN**: Exponential Smoothing – Recurrent Neural Network (hybrid model)
- **ETS**: Error, Trend, Seasonality (exponential smoothing state space models)
- **EU**: European Union
- **GB**: Gradient Boosting (ensemble of regression trees)
- **GBRT**: Gradient Boosted Regression Trees
- **GDP**: Gross Domestic Product
- **GLM**: Generalised Linear Model
- **HAR**: Heterogeneous AutoRegressive (model)
- **HICP**: Harmonised Index of Consumer Prices
- **LSTM**: Long Short-Term Memory (network)
- **MAE**: Mean Absolute Error
- **MASE**: Mean Absolute Scaled Error
- **MDI**: Mean Decrease Impurity

Acronyms used in this material (2/2)

- **MIMO**: Multiple-Input Multiple-Output (forecasting strategy)
- **ML**: Machine Learning
- **MLP**: MultiLayer Perceptron
- **MSE**: Mean Squared Error
- **OLS**: Ordinary Least Squares
- **OLS-3**: OLS with three predictors: size, book-to-market and momentum (Gu, Kelly and Xiu, 2020)
- **OWA**: Overall Weighted Average (of the relative sMAPE and MASE, M4 competition)
- **PCR**: Principal Component Regression
- **PLS**: Partial Least Squares
- **QLIKE**: Quasi-LIKElihood (loss function)
- **RF**: Random Forest
- **RMSE**: Root Mean Squared Error
- **RMSSE**: Root Mean Squared Scaled Error
- **RNN**: Recurrent Neural Network
- **ROC**: Receiver Operating Characteristic (curve)
- **RW**: Random Walk (forecast)
- **SARIMA**: Seasonal ARIMA (model)
- **SES**: Simple Exponential Smoothing
- **SGD**: Stochastic Gradient Descent
- **US**: United States
- **WRMSSE**: Weighted Root Mean Squared Scaled Error (M5 competition)

Today's question and route

- **Question**: can random forests, gradient boosting and neural networks forecast a time series better than ARIMA, ETS and the naive forecast, and how do we check it honestly?
  - machine learning (ML): algorithms that learn a prediction rule from examples; for time series, the examples are built from the past of the series
- **Route** of the chapter
  - forecasting as supervised learning: lags, rolling windows, calendar features; recursive, direct and multi-output strategies
  - walk-forward validation and leakage; ridge and lasso; trees, random forest, gradient boosting; small neural networks and the LSTM
  - prediction intervals; local and global models; four applications and the M4, M5 competitions
- We build on Chapter 0 (accuracy measures, benchmarks) and Chapter 4 (time-series cross-validation, Diebold–Mariano)

Learning outcomes

- Turn a time series into a supervised-learning table without using future information
- Produce multi-step forecasts with the recursive, direct and multi-output strategies
- Validate a model walk-forward and recognise leakage and look-ahead bias
- Explain ridge, lasso, regression trees, random forest, gradient boosting, the MLP and the LSTM, with their main tuning parameters
- Build prediction intervals by quantile regression and conformal prediction
- Compare ML forecasts with statistical benchmarks on the same test period, with MASE and the Diebold–Mariano test

Reading and tools

- Textbook: Huang and Petukhina (2022), Ch. 10 (machine learning for time series)
  - companion, free online: Hyndman and Athanasopoulos (2021), Sec. 12.4 (neural network models); theory: Hastie, Tibshirani and Friedman (2009), Ch. 3, 9, 10, 11, 15
- Reviews: Hewamalage, Bergmeir and Bandara (2021) (recurrent networks), Lim and Zohren (2021) (deep learning), Montero-Manso and Hyndman (2021) (global models)
- Python Quantlets of this chapter: Quantlets/Ch_09
  - `scikit-learn`: `RidgeCV`, `LassoCV`, `RandomForestRegressor`, `HistGradientBoostingRegressor`, `MLPRegressor`, `TimeSeriesSplit`; a tiny LSTM in `PyTorch`
  - small models, fixed seeds, CPU only: every notebook runs in Google Colab in a few minutes
- Lecture notebook: open in Google Colab
- Video course: Applied Time Series Analysis with Python

Supervised learning: data, loss function and training

- **Data**: pairs $(\mathbf{x}_i, y_i)$, $i = 1, \ldots, n$; $\mathbf{x}_i$: the **features** (inputs), y_i: the **target**
  - goal: a function $\hat{f}$ with small expected loss $E[L(y, \hat{f}(\mathbf{x}))]$ on **new** data
  - loss: squared error $L = (y - \hat{y})^2$ (forecasts the mean), absolute error $|y - \hat{y}|$ (the median)
- **Training**: choose $\hat{f}$ in a class of functions (linear, trees, networks) by minimising the average loss on the training data
  - **hyperparameters**: settings fixed before training (tree depth, number of trees, penalty); chosen by validation
- Standard ML assumes the pairs are independent; in a time series they are ordered and dependent, so the table and the validation must respect time

From a series to a table

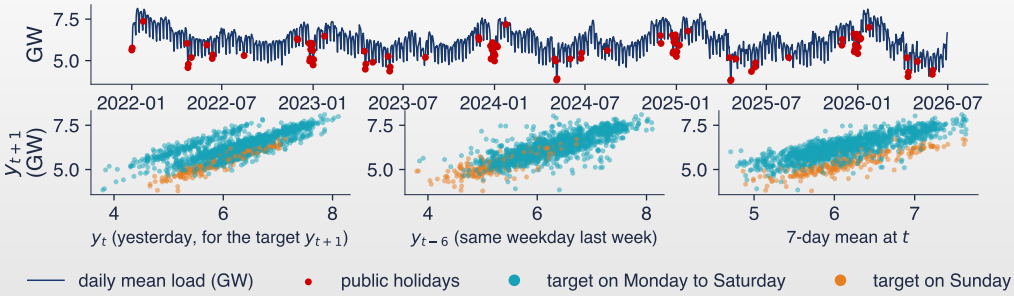
- **Lag features**: $y_t, y_{t-1}, \ldots, y_{t-p+1}$, known at the forecast origin t; target y_{t+h}
  - an AR(p) (Chapter 2) is exactly a linear regression on this table; ML replaces the linear function by a flexible one
- **Rolling-window features**: means, standard deviations, minima over the last w values
  - for example the 7-day mean $\bar{y}_t^{(7)} = \frac{1}{7}\sum_{j=0}^{6} y_{t-j}$
  - the window must end at t, the origin: a window that contains y_{t+1} leaks the target
- **Calendar features** of the target day: day of week, month, Fourier terms $\sin(2\pi k\, d/365.25)$, holidays
  - d: the day of the year; $k = 1, 2, \ldots$: the harmonic (one annual cycle for $k = 1$)
  - known in advance, so allowed (Chapter 4: Fourier terms and Orthodox Easter); exogenous variables (temperature) only if their values are known or forecast at t

Worked example: a lag table for the electricity load

Origin t	y_t	y_{t-1}	y_{t-2}	**target** y_{t+1}
Mon	5.80	4.73	5.21	6.03
Tue	6.03	5.80	4.73	6.38
Wed	6.38	6.03	5.80	6.61
Thu	6.61	6.38	6.03	6.59
Fri	6.59	6.61	6.38	5.78
Sat	5.78	6.59	6.61	5.09

- Daily mean load of Romania (GW), the last days before 30 June 2025
  - each row: what we knew at t, and the value we want for $t + 1$
- With p lags, the first $p - 1$ days have no complete row
  - a series of length n gives $n - p - h + 1$ rows for horizon h
- Weekday column: the Sunday dips are visible; a model needs the weekday of the **target** day

Electricity load: the series and three features



Daily mean load of Romania, ENTSO-E (the European Network of Transmission System Operators for Electricity), 1 January 2022–30 June 2026, 1 642 days; bottom: tomorrow's load against three features

TSA_ch9_supervised_learning

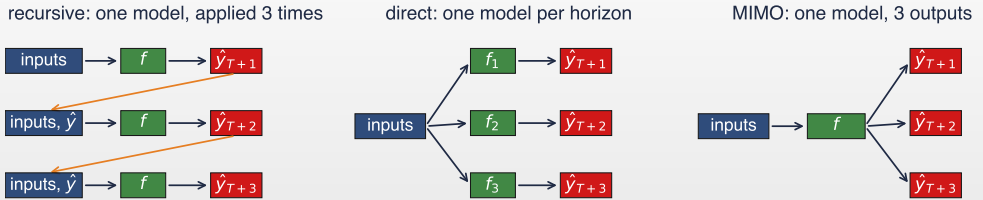
Interpreting the load features

- Mean 6.18 GW; minimum 3.79 GW on 20 April 2025 (Orthodox Easter), maximum 8.15 GW on 13 January 2022 (a winter week)
  - weekly cycle, annual cycle (winter heating, summer cooling) and deep holiday dips: the multiple seasonality of Chapter 4
- Correlation with tomorrow's load: y_t 0.77; the same weekday last week 0.82; the 7-day mean 0.74
  - with y_t alone the Sundays form a separate cloud: a single linear lag cannot represent the weekly pattern
  - the table used below has 1 614 rows and 30 features (14 lags, two means, the same weekday, calendar)

Multi-step forecasts: three strategies

- **Recursive** (iterated): one model for $h = 1$; its forecast replaces the unknown y_{T+1} and the model is applied again
  - this is how ARIMA forecasts (Chapter 3); errors accumulate when the one-step model is wrong
- **Direct**: a separate model $\hat{f}_h$ for each horizon $h = 1, \ldots, H$, trained on the target y_{t+h}
  - no error accumulation, but H models and fewer rows for long horizons; the forecasts of different h need not be coherent
- **Multi-output** (MIMO, multiple-input multiple-output): one model returns the vector $(\hat{y}_{T+1}, \ldots, \hat{y}_{T+H})$
  - natural for neural networks; review and comparison: Ben Taieb et al. (2012)

Recursive, direct and multi-output forecasts



- Blue: the inputs known at the origin T; green: the fitted models; red: the forecasts; orange: a forecast fed back as an input

TSA_ch9_supervised_learning

Worked example: recursive and direct two-step forecasts

- One-step model estimated on the table: $\hat{y}_{t+1} = 1.2 + 0.8\, y_t$; last value $y_T = 5$
  - recursive: $\hat{y}_{T+1} = 1.2 + 0.8 \cdot 5 = 5.20$; $\hat{y}_{T+2} = 1.2 + 0.8 \cdot 5.20 = 5.36$
  - implied two-step rule: $\hat{y}_{T+2} = 2.16 + 0.64\, y_T$
- Direct model estimated on the target y_{t+2}: $\hat{y}_{t+2} = 2.3 + 0.6\, y_t$; $\hat{y}_{T+2} = 5.30$
  - if the AR(1) were the true model, the direct coefficients would estimate the same 2.16 and 0.64
  - they differ: the one-step model is misspecified at two steps (for example, a weekly pattern), and the direct model adapts to the horizon
- Neither strategy wins always: try both on the validation data

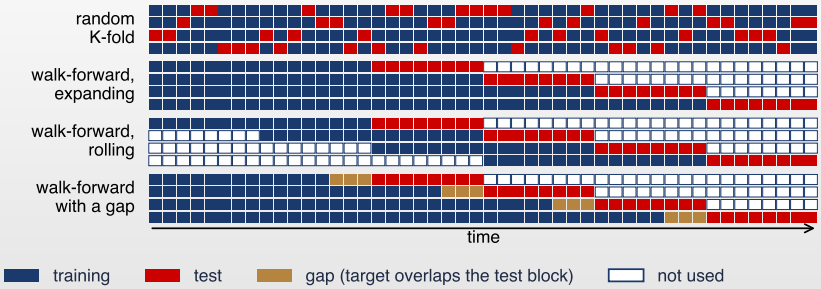
Recap: Forecasting as supervised learning

- A forecast is a prediction of y_{t+h} from features known at t: lags, rolling statistics, calendar
- An AR(p) is linear regression on lags; ML changes the function, not the table
- Multi-step: recursive (one model, iterated), direct (one model per h), MIMO (one model, H outputs)

Training, validation and test samples

- **Training** sample: estimates the parameters; **validation**: chooses the hyperparameters; **test**: measures the final accuracy, used once
  - for time series the three samples are consecutive blocks, in this order
- **Walk-forward** validation (time-series cross-validation, Chapter 4): train on data up to an origin, forecast the next block, move the origin forward
  - **expanding** window: all past data; **rolling** window: the last W observations (adapts to change, uses less data)
  - the errors of all origins are pooled into MAE, RMSE, MASE (Chapter 0) and compared with the Diebold–Mariano test
- Random K-fold cross-validation (random folds) is valid only for pure autoregressions with white-noise errors Bergmeir, Hyndman and Koo (2018)

Four validation schemes



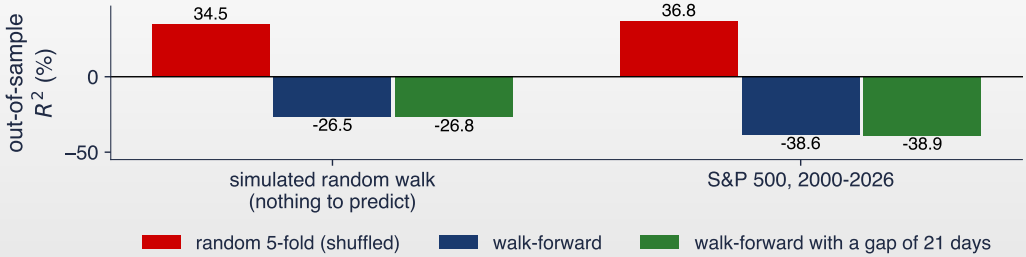
- Random K-fold mixes past and future; walk-forward keeps the order; a gap removes the training rows whose targets overlap the test block

TSA_ch9_validation

Look-ahead bias and leakage

- **Leakage**: information that would not be available at the forecast origin enters training or the features Kaufman et al. (2012)
  - result: excellent validation scores that disappear in real use; **look-ahead bias** is its name in finance
- Typical sources
  - a rolling mean or a difference that includes y_{t+1}; centred moving averages and two-sided filters (Chapter 0)
  - scaling, imputation or feature selection fitted on the whole sample before the split
  - overlapping targets (sums over the next h days) with random folds or without a gap
  - revised macro data: today's GDP vintage was not known at the origin; actual weather instead of the weather forecast
- Rule: write the feature code as a function of the data up to t only, and refit every preprocessing step inside each training window

Leakage in practice: random folds on overlapping targets



- Random forest for the sum of the next 21 daily returns; features: past sums over 5, 21, 63 days and volatilities; left: a simulated random walk; right: the S&P 500, 6 631 days

TSA_ch9_validation

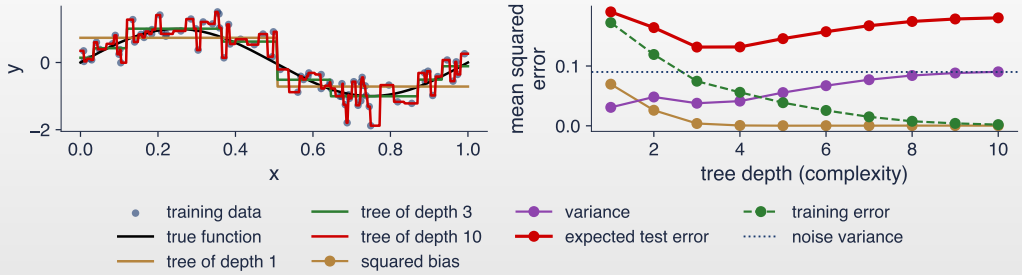
Interpreting the leakage experiment

- Random 5-fold reports $R^2 = 34.5\%$ for a random walk, where nothing is predictable, and 36.8% for the S&P 500
  - neighbouring targets share 20 of 21 returns: a test row has its near-copies in the training folds
- Walk-forward: -26.5% and -38.6%; with a 21-day gap: -26.8% and -38.9%
  - negative out-of-sample R^2: the forest is worse than the historical mean, as expected for returns (Chapter 1)
- A model is only as good as its validation; a spectacular score is a reason to look for a leak

Overfitting and the bias–variance trade-off

- Model $y = f(x) + \varepsilon$, $\mathrm{Var}(\varepsilon) = \sigma^2$; $\hat{f}$ estimated on a random training sample
  - the expected test error at a point x_0:
  - $E[(y_0 - \hat{f}(x_0))^2] = \underbrace{(E[\hat{f}(x_0)] - f(x_0))^2}_{\text{bias}^2} + \underbrace{\mathrm{Var}(\hat{f}(x_0))}_{\text{variance}} + \sigma^2$
  - f: the true function; y_0: a new observation at x_0; the expectation is over training samples and noise; σ^2: the noise that no model can remove
- **Overfitting**: a flexible model learns the noise; small training error, large test error
  - **underfitting**: a rigid model misses the signal (large bias)
- Every ML method has a complexity knob (depth, number of trees, penalty, hidden units); validation sets it

Bias and variance of regression trees



⊡ Simulation: $y = \sin(2\pi x) + \varepsilon$, $\sigma = 0.3$, 80 points, 300 training samples; trees of depth 1 to 10

TSA_ch9_validation

Interpreting the U-shaped test error

- Depth 1: bias^2 0.070, variance 0.031; depth 10: bias^2 0.000, variance 0.090
  - the expected test error is smallest at depth 3 (0.132), above the noise variance 0.09
- The training error falls to 0.002 at depth 10: it cannot choose the complexity
- Ensembles (random forest, boosting) and penalties (ridge, lasso) reduce the variance without losing much bias

Recap: Validation without leakage

- Walk-forward: train on the past, test on the next block, move on; random folds only for pure autoregressions
- Leakage: future values in features, preprocessing on the full sample, overlapping targets, revised data
- Test error = bias^2 + variance + noise; validation sets the complexity

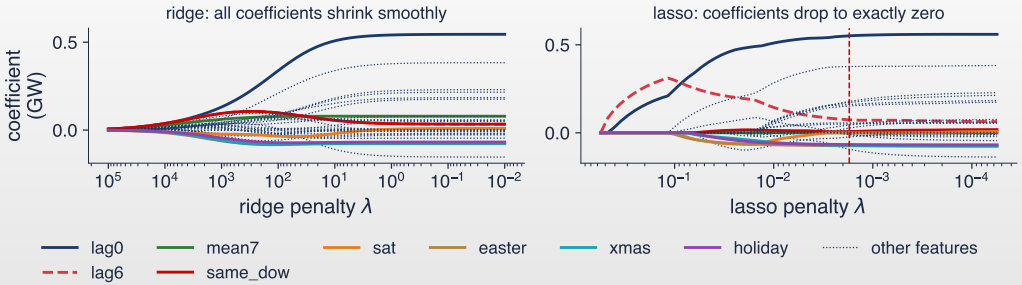
Many lags, unstable least squares

- Linear model on p standardised features: $\hat{y} = \beta_0 + \sum_j \beta_j x_j$; OLS (ordinary least squares) minimises $\sum_t (y_t - \hat{y}_t)^2$
  - neighbouring lags are highly correlated: OLS coefficients become large, of opposite signs and unstable from one window to the next
- **Standardise** first: $x_j \to (x_j - \bar{x}_j)/s_j$, with $\bar{x}_j$, s_j from the training window only
  - otherwise the penalty treats features measured in GW and in dummies unequally
- Idea of regularisation: accept a little bias in exchange for a large fall in variance

Ridge and lasso

- **Ridge** Hoerl and Kennard (1970): $\min_\beta \sum_t (y_t - \beta_\mathbf{0} - \mathbf{x}_t'\beta)^\mathbf{2} + \lambda \sum_j \beta_j^\mathbf{2}$
  - $\lambda \geq 0$: the penalty ($\lambda = 0$: OLS); $\mathbf{X}$: the matrix of standardised features; $\mathbf{y}$: the targets; $\mathbf{I}$: the identity matrix
  - closed form $\hat{\beta} = (\mathbf{X}'\mathbf{X} + \lambda\mathbf{I})^{-\mathbf{1}}\mathbf{X}'\mathbf{y}$; all coefficients shrink towards 0, none becomes exactly 0
- **Lasso** Tibshirani (1996) (least absolute shrinkage and selection operator): penalty $\lambda \sum_j |\beta_j|$
  - sets some coefficients exactly to 0: it selects lags
- **Worked example**, one standardised feature, $S_{xy} = \sum x_t y_t = 90$, $S_{xx} = \sum x_t^\mathbf{2} = 100$
  - OLS $90/100 = 0.90$; ridge, $\lambda = 50$: $90/150 = 0.60$; lasso, $\lambda = 40$: $(90 - 20)/100 = 0.70$; lasso, $\lambda \geq 180$: exactly 0
  - lasso in one dimension: $\hat{\beta} = \text{sign}(S_{xy}) \max(|S_{xy}| - \lambda/2, 0)/S_{xx}$ (soft thresholding)
- λ is chosen by walk-forward validation (`TimeSeriesSplit`), never by random folds

Ridge and lasso paths on the load features



- Target: tomorrow's load; 30 standardised features; data up to 30 June 2025 (1 249 rows); dashed: the lasso penalty chosen by walk-forward cross-validation

TSA_ch9_regularisation_trees

Interpreting the coefficient paths

- With a large penalty all coefficients are near 0; as λ falls, the first to enter are lag6 (y_{t-6}, the same weekday as the target), lag0 (y_t) and lag13; at the chosen penalty y_t dominates, with coefficient 0.55 GW per standard deviation
  - the next largest coefficients are the weekday dummies (Monday 0.38): the calendar carries most of the remaining signal
- At the chosen penalty the lasso keeps 22 of 30 features: many lags are redundant once y_t, the weekday and the holidays are in
- Ridge keeps everything but shrinks it smoothly; on the test origins of Section 6 ridge is the best ML model

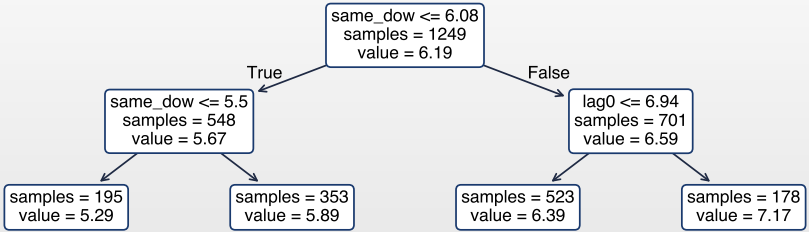
Recap: Ridge and lasso

- Penalised least squares: ridge $\lambda \sum \beta_j^2$ shrinks; lasso $\lambda \sum |\beta_j|$ shrinks and selects
- Standardise inside each training window; choose λ walk-forward
- A regularised AR with calendar dummies is a strong ML baseline

Regression trees

- A **regression tree** splits the feature space with questions "$x_j \leq c$?" and predicts, in each final region (leaf), the mean target of its training rows
  - the split is chosen greedily: the (j, c) with the smallest sum of squared errors of the two children
- Strengths: nonlinearity and interactions (weekday $\times$ season) without specifying them; no scaling needed
  - weaknesses: a single tree is unstable (high variance) and piecewise constant
- **A tree cannot extrapolate**: its forecast is always a mean of training targets
  - for trending series: model differences, growth rates or the deviation from a trend, not levels

A depth-2 tree for tomorrow's load



- Features: y_t (lag0), the same weekday last week (same_dow), weekend and holiday flags; data up to 30 June 2025; at least 20 days per leaf

TSA_ch9_regularisation_trees

Interpreting the tree

- First question: was the same weekday last week below 6.08 GW? It separates weekends and holidays from working days
  - four leaves: 5.29, 5.89, 6.39 and 7.17 GW, with 195, 353, 523 and 178 days
- Four numbers already explain $R^2 = 0.64$ of the variance in training; deeper trees add detail and variance
- The forecast for any day is one of these four means

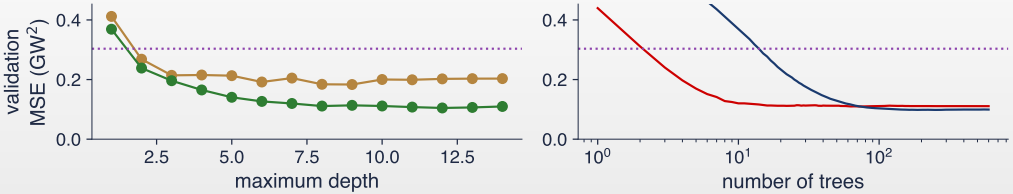
Random forest

- **Bagging** (bootstrap aggregation): grow B deep trees on bootstrap samples of the rows and average them
  - averaging B trees with correlation ρ and variance σ^2 gives variance $\rho\sigma^2 + (1-\rho)\sigma^2/B$
- **Random forest** Breiman (2001): at each split, only a random subset of features is tried (`max_features`)
  - this decorrelates the trees (ρ falls), so the average has a lower variance
- Hyperparameters: number of trees (more is never worse, only slower), `max_features`, minimum leaf size
  - the bootstrap ignores time order inside the training window; that is fine, because the validation is walk-forward

Gradient boosting

- **Boosting** Friedman (2001): add small trees one at a time, each fitted to the residuals (the negative gradient of the loss) of the current model
  - $F_m(\mathbf{x}) = F_{m-\mathbf{1}}(\mathbf{x}) + \nu\, g_m(\mathbf{x})$; F_m: the model after m trees; g_m: the m-th small tree; ν: the **learning rate** (0.01–0.3)
- **Worked example**: targets $5, 6, 8$; start with the mean $F_{\mathbf{0}} = 6.33$; residuals -1.33, -0.33, $+1.67$
  - a stump that isolates the third point predicts 1.67 there; with $\nu = 0.1$: $F_{\mathbf{1}} = 6.33 + 0.1 \cdot 1.67 = 6.500$
- Fast implementations bin the features into histograms: XGBoost Chen and Guestrin (2016), LightGBM Ke et al. (2017), `HistGradientBoostingRegressor` in scikit-learn
  - below we write GB for gradient boosting; the winners of M5 (Section 10) used LightGBM
  - hyperparameters: number of trees with **early stopping** on validation data, tree size, ν, minimum leaf size

One tree, a forest and boosting on the load



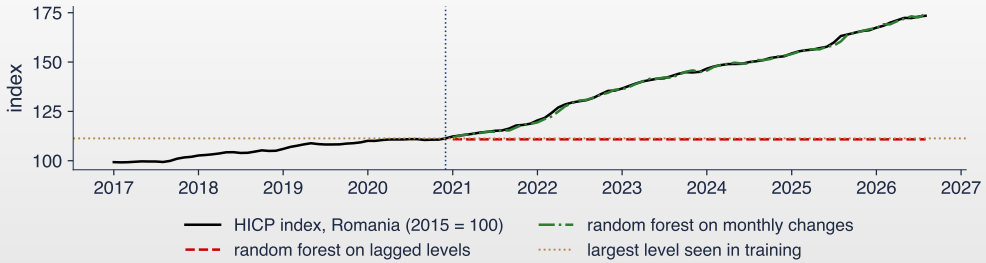
- Tomorrow's load; training up to 31 December 2024 (1069 days), validation January–June 2025 (180 days); dotted: the seasonal naive forecast

TSA_ch9_regularisation_trees

Interpreting the ensembles

- Single tree: best validation MSE (mean squared error) 0.183 at depth 9; random forest: 0.105 at depth 12
  - averaging removes most of the variance; the seasonal naive forecast has 0.303
- Boosting with $\nu = 0.3$: minimum 0.110 after 72 trees; with $\nu = 0.05$: 0.098 after 256 trees
  - a small learning rate needs more trees and usually ends slightly lower; early stopping picks the number of trees

Trees cannot extrapolate: the Romanian price level



- HICP (harmonised index of consumer prices) of Romania, Eurostat, 2015 = 100; one-month-ahead forecasts from random forests trained up to December 2020, on lagged levels or on monthly log changes

TSA_ch9_regularisation_trees

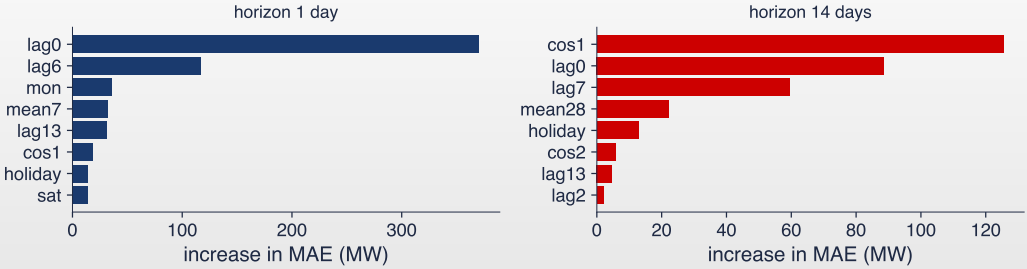
Interpreting the extrapolation failure

- The highest index in training was 111.3; by August 2026 the index reached 173.4
  - the forest on levels stays near 110.8 for five years: RMSE 36.9 index points
- The same forest on monthly changes, then cumulated, follows the index: RMSE 0.71
- Chapter 3 again: difference or transform non-stationary series before a tree sees them

The features a model uses

- **Permutation importance**: shuffle one feature in the test data and measure how much the error grows
  - model-agnostic; measured out of sample; correlated features share (and hide) their importance
- Impurity-based importance of trees (MDI, mean decrease impurity) is computed in training and favours continuous features: prefer permutation
- Importance is not causality: it describes the model, not the electricity system

Permutation importance of gradient boosting



- Direct GB models for horizons 1 and 14 days; trained up to 31 December 2024, evaluated on January–June 2025; 20 random permutations per feature

TSA_ch9_load_forecasting

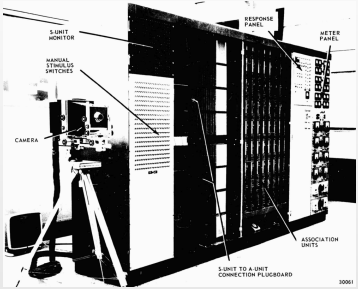
Interpreting the importances

- One day ahead: lag0 (+371 MW of MAE when shuffled) and lag6 (+117 MW) dominate
  - yesterday and the same weekday last week: the persistence and the weekly cycle
- Fourteen days ahead: cos1 (+126 MW) and lag0 (+89 MW)
  - the annual cycle (cos1, the Fourier term) gains weight as the recent values become less informative

Recap: Trees and ensembles

- A tree is a piecewise-constant function found by greedy splits; it cannot extrapolate
- Random forest averages decorrelated deep trees (less variance); boosting adds small trees on residuals (less bias)
- GB (gradient boosting, LightGBM, XGBoost, `HistGradientBoosting`): learning rate, number of trees, early stopping
- Permutation importance measures what the model uses, out of sample

The multilayer perceptron

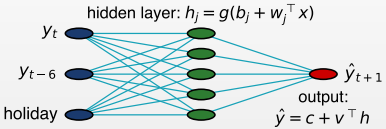


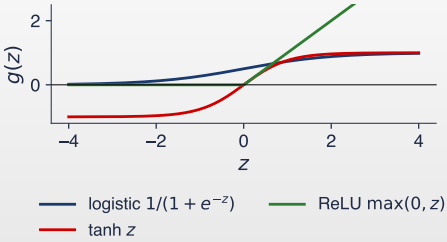
Mark I Perceptron, Cornell (1960): the first trainable neural network, built in hardware

Image: J. C. Hay, A. E. Murray (1960), Mark I Perceptron operator's manual; public domain; Wikimedia Commons

- **MLP** (multilayer perceptron), one hidden layer with q neurons:
  - $h_j = g(b_j + \mathbf{w}_j'\mathbf{x})$, $j = 1, \ldots, q$; $\quad \hat{y} = c + \sum_j v_j h_j$
  - $\mathbf{x}$: the inputs; $\mathbf{w}_j$, b_j: the weights and the bias of neuron j; h_j: its output; v_j, c: the output weights and constant
  - g: activation function (ReLU $\max(0, z)$, tanh, logistic); with $g(z) = z$ the network is linear again
- Universal approximation Hornik, Stinchcombe and White (1989): enough hidden neurons approximate any continuous function
  - parameters: 3 inputs and 5 neurons already give $3 \cdot 5 + 5 + 5 + 1 = 26$ weights

A small network and its activation functions





- Left: 3 inputs (yesterday, the same weekday last week, a holiday flag), 5 hidden neurons, one output; right: three activation functions

TSA_ch9_neural_networks

Training a network

- Minimise the training loss by **gradient descent**; the gradients come from back-propagation Rumelhart, Hinton and Williams (1986)
  - SGD (stochastic gradient descent) and Adam use small random batches; one pass through the data is an epoch
- The loss surface has many local minima: results depend on the random starting weights
  - fix the seed, and average a few networks with different seeds (an ensemble)
- Regularisation: an L2 penalty on the weights, early stopping, dropout (randomly switching off neurons)
  - inputs and targets must be standardised with training-window statistics
- Small data (a few thousand rows) favour small networks; deep networks need many series or long high-frequency data

Recurrent networks and the LSTM (1/2): the recurrent network

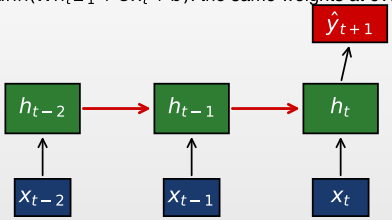
- **RNN** (recurrent neural network): a hidden state carries the past forward, with the same weights at every step
  - $h_t = \tanh(W h_{t-1} + U x_t + b)$: the new state mixes the previous state and the new input
  - x_t: the input at t; h_t: the hidden state; W, U, b: weights and bias, shared by all steps; tanh keeps each component in $(-1, 1)$
  - the forecast is read from the last state, for example $\hat{y}_{t+1} = v' h_t + c$
- The difficulty: training through many steps multiplies many derivatives
  - the gradients vanish or explode: a plain RNN forgets what happened more than a few steps back
- Honest summary Hewamalage, Bergmeir and Bandara (2021)
  - on single, short series, recurrent networks rarely beat ETS or ARIMA
  - they shine with many related series and enough data (global models, Salinas et al. (2020))

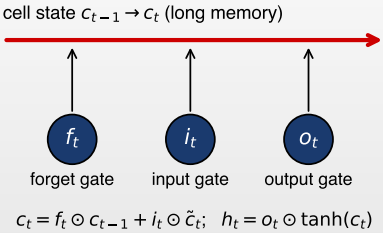
Recurrent networks and the LSTM (2/2): the LSTM cell

- **LSTM** (long short-term memory) Hochreiter and Schmidhuber (1997): a cell state c_t updated additively, controlled by three gates
  - the gates: $f_t = \sigma(W_f[h_{t-\mathbf{1}}, x_t] + b_f)$ (forget), $i_t = \sigma(W_i[h_{t-\mathbf{1}}, x_t] + b_i)$ (input), $o_t = \sigma(W_o[h_{t-\mathbf{1}}, x_t] + b_o)$ (output)
  - the candidate new content: $\tilde{c}_t = \tanh(W_c[h_{t-\mathbf{1}}, x_t] + b_c)$
  - $\sigma(z) = 1/(1 + e^{-z}) \in (0, 1)$: the logistic function; $[h_{t-\mathbf{1}}, x_t]$: the two vectors stacked; W_f, W_i, W_o, W_c and b_f, b_i, b_o, b_c: weights and biases
- The updates of the cell and of the output
  - $c_t = f_t \odot c_{t-\mathbf{1}} + i_t \odot \tilde{c}_t$: with f_t close to 1 the old memory survives many steps; i_t sets how much new content enters
  - $h_t = o_t \odot \tanh(c_t)$: the output gate decides which part of the memory is passed on; $\odot$: element-wise product
- Each gate is a small logistic layer: an LSTM with 32 units and 8 inputs has about 5 400 weights

A recurrent network and the LSTM cell

$h_t = \tanh(Wh_{t-1} + Ux_t + b)$: the same weights at every step





- Left: the RNN unrolled over three steps; right: the LSTM cell state (red) and its three gates; $\odot$: element-wise product

TSA_ch9_neural_networks

Sepp Hochreiter and the LSTM



Sepp Hochreiter (2015), Johannes Kepler University Linz

Photo: Eulenreich (2015); CC BY-SA 4.0; Wikimedia Commons

- In his 1991 diploma thesis in Munich, Hochreiter analysed why gradients vanish in recurrent networks
- With Jürgen Schmidhuber he proposed the LSTM Hochreiter and Schmidhuber (1997)
  - one of the most cited papers in neural computing
- LSTMs powered speech recognition and machine translation before transformers (Chapter 11)

Recap: Neural networks

- MLP: layers of $g(b + \mathbf{w}'\mathbf{x})$; trained by gradient descent; seed, scaling, L2 penalty, early stopping
- RNN and LSTM process sequences; the LSTM cell state avoids vanishing gradients
- Small data favour small models; neural networks need many series or long data

The forecasting exercise



750 kV line at Mărășești: the Romanian transmission grid

Photo: TrainSimFan (2022); CC BY-SA 4.0; Wikimedia Commons

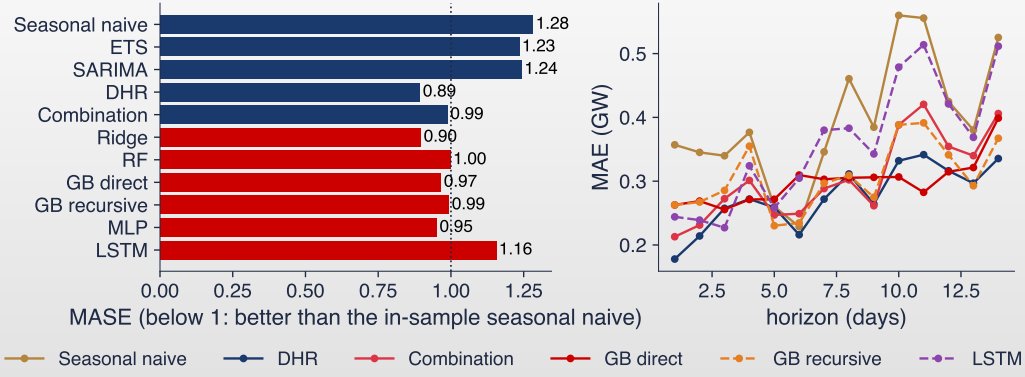
- The evaluation design of Chapter 4: 26 forecast origins every 14 days, 30 June 2025 to 15 June 2026
  - horizon 1–14 days; expanding window from 1 January 2022
  - statistical models (Chapter 4): seasonal naive, ETS, SARIMA, DHR (dynamic harmonic regression), their combination
- ML models, refitted at each origin:
  - direct: ridge, random forest (200 trees), GB (300 trees, $\nu = 0.05$), one model per horizon
  - recursive: GB on the one-step table; MIMO: an MLP (3 networks, 32 neurons) and an LSTM (56 days in, 14 out)
- MASE scale: in-sample MAE of the weekly seasonal naive method, 309 MW

Results: 26 origins, 14 horizons

Model	MAE (GW)	RMSE (GW)	MASE	DM p, against DHR
Seasonal naive	0.396	0.529	1.28	< 0.001
ETS	0.381	0.492	1.23	0.015
SARIMA	0.384	0.477	1.24	0.003
DHR	0.276	0.350	0.89	–
Combination	0.305	0.380	0.99	0.065
Ridge	0.277	0.337	0.90	0.942
RF	0.308	0.388	1.00	0.068
GB direct	0.298	0.369	0.97	0.163
GB recursive	0.307	0.388	0.99	0.113
MLP	0.294	0.362	0.95	0.546
LSTM	0.357	0.447	1.16	0.016

- DM: Diebold–Mariano test on the origin-average absolute errors (26 values), with the Harvey–Leybourne–Newbold correction Harvey, Leybourne and Newbold (1997)

MASE and accuracy by horizon



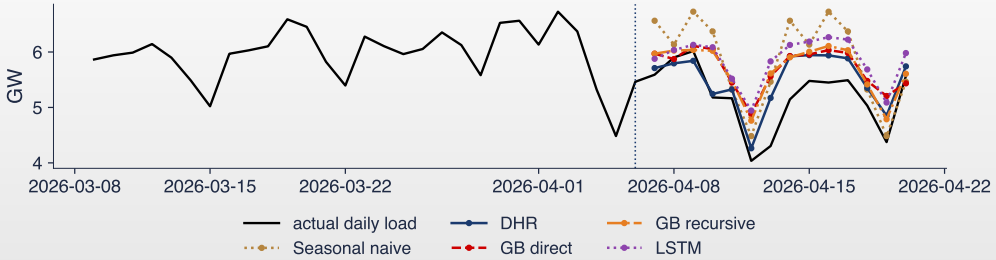
⊡ Left: MASE of the eleven models (blue: Chapter 4; red: ML); right: MAE by horizon for six models

TSA_ch9_load_forecasting

Interpreting the load results

- DHR (MASE 0.89) and ridge (0.90) are the most accurate; the difference is not significant ($p = 0.942$)
  - both are linear models with the right calendar features; the trees and networks do not beat them
- GB direct (0.97), MLP (0.95), random forest (1.00) beat the seasonal naive method clearly (DM $p = 0.0015$; 0.0074; 0.0015) and have a lower MASE than SARIMA and ETS (not tested with DM)
  - direct and recursive GB are close (0.97 and 0.99); the LSTM (1.16) is worse than every model except the naive, SARIMA and ETS
- Lesson: features matter more than the algorithm; with four years of daily data, a well-specified linear model is hard to beat

Forecasts around Orthodox Easter 2026



- Origin 6 April 2026; 14-day forecasts; MAE in this window: DHR 344 MW, GB direct 536 MW, GB recursive 503 MW, LSTM 661 MW

TSA_ch9_load_forecasting

Interpreting the Easter window

- All models except the LSTM see the Easter flag, but the trees have only four Easters in training: they predict a dip that is too shallow
  - DHR estimates one Easter coefficient from the same four episodes and pools the information more efficiently
- The LSTM sees no holiday flag: it repeats the weekly pattern and misses the holiday entirely
- Rare events need either structure (a regression coefficient) or many series that share the event (a global model)

Recap: Electricity load

- Same origins, same horizon, same errors: ML models enter the comparison of Chapter 4
- Ridge and DHR: no significant difference ($p = 0.942$); GB, random forest and the MLP have a lower MASE than SARIMA and ETS; the LSTM is the weakest ML model
- Calendar features and holidays decide the ranking

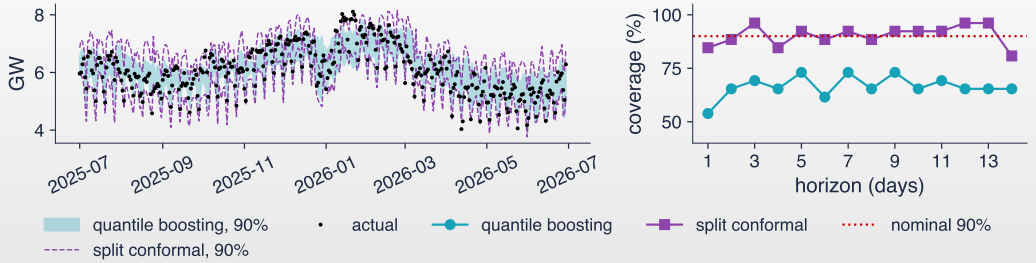
Quantile regression and the pinball loss

- A point forecast is not enough: a grid operator needs the load it will not exceed with 95% probability
  - the τ-quantile q_τ of y_{t+h}: $P(y_{t+h} \le q_\tau) = \tau$; a 90% interval: $[q_{0.05}, q_{0.95}]$
- **Pinball loss**: $L_\tau(y, q) = \tau(y - q)$ if $y \ge q$, and $(1 - \tau)(q - y)$ if $y < q$ Koenker and Bassett (1978)
  - its expected value is minimised by the true quantile; with $\tau = 0.5$ it is half the absolute error
  - example, $\tau = 0.95$, $q = 7.0$ GW: $y = 7.4$ costs $0.95 \cdot 0.4 = 0.380$; $y = 6.5$ costs $0.05 \cdot 0.5 = 0.025$
- Any learner can minimise the pinball loss: linear quantile regression, quantile GB (`loss='quantile'`), networks

Split conformal prediction

- **Idea** Lei et al. (2018): use the errors of the model on data it has not seen to set the width of the interval
  - 1. fit the model on the training data except the last n points (the calibration set)
  - 2. compute the absolute errors $|e_i|$ on the calibration set and sort them
  - 3. $\hat{q}$ = the $\lceil (n+1)(1-\alpha) \rceil$-th smallest; interval $\hat{y} \pm \hat{q}$
  - α: the target miscoverage ($\alpha = 0.1$ for a 90% interval); $\lceil \cdot \rceil$: rounding up
- Guarantee: coverage at least $1 - \alpha$ if the errors are exchangeable (their order does not matter)
  - time series are not exchangeable: the guarantee is approximate; use recent calibration data and recheck the coverage walk-forward
- Here: $n = 182$ days, $\alpha = 0.1$, $\hat{q}$ = the 165-th smallest absolute error, one calibration set per horizon and origin

90% intervals for the load



⊡ Direct GB, the 364 forecasts of Section 6 (all origins and horizons); left: intervals and outcomes; right: coverage by horizon

TSA_ch9_load_forecasting

Interpreting the intervals

- Quantile GB covers 66.5% with mean width 884 MW; split conformal covers 90.4% with width 1310 MW
  - quantile GB: 19.0% of outcomes below and 14.6% above the interval (nominal 5% and 5%)
- Quantiles estimated by trees are noisy in the tails and too narrow; conformal calibration corrects the width with real out-of-sample errors
- Always report the empirical coverage next to an interval

Recap: Prediction intervals

- Quantile forecasts minimise the pinball loss; an interval is a pair of quantiles
- Split conformal: $\hat{y} \pm$ an empirical quantile of calibration errors
- Check the coverage walk-forward

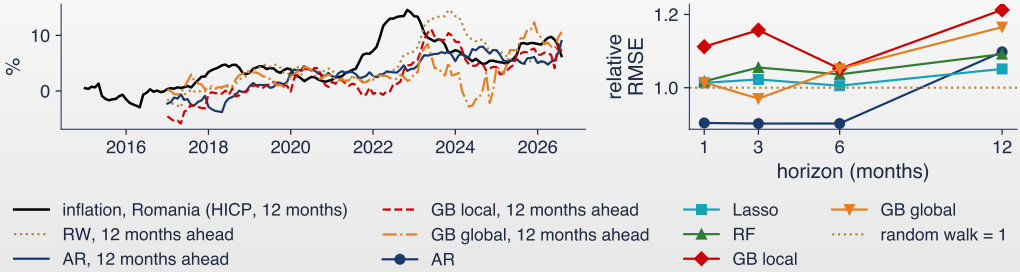
Local and global models

- **Local** model: one model per series, estimated on that series only (ARIMA, ETS, the models so far)
  - few observations per series limit the complexity a local model can afford
- **Global** model: one model estimated on the pooled rows of many related series Montero-Manso and Hyndman (2021), Januschowski et al. (2020)
  - the same features for every series (scaled lags, calendar); optionally a series identifier
  - more data per parameter: complex learners (GB, networks) become usable; the winners of M4 and M5 were global
- Risk: the series must share dynamics; a global model may average away what is special about one series

The inflation exercise

- Target: 12-month HICP inflation of Romania (Eurostat), $\pi_t = 100(P_t/P_{t-\mathbf{12}} - 1)$, P_t: the HICP index, at horizons $h = 1, 3, 6, 12$ months
  - direct strategy; the models forecast the change $\pi_{t+h} - \pi_t$, so trees never extrapolate the level
- Features at t: $\pi_t, \pi_{t-\mathbf{1}}, \pi_{t-\mathbf{2}}, \pi_{t-\mathbf{3}}, \pi_{t-\mathbf{6}}, \pi_{t-\mathbf{12}}$, the last three monthly log changes and their 3- and 6-month sums, the calendar month
  - models: random walk (RW, $\hat{\pi}_{t+h} = \pi_t$), AR (OLS on the same features), lasso, random forest, GB local (Romania only), GB global (27 EU countries pooled)
- Walk-forward: refit every January on all rows whose target month is already observed (from 2006); test origins January 2016–July 2026
- Related evidence: Medeiros et al. (2021) find that random forests beat the benchmarks for US inflation with more than 100 predictors

Romanian inflation: forecasts and relative RMSE



⊡ Left: inflation and the 12-month-ahead forecasts plotted at the target month; right: RMSE relative to the random walk

TSA_ch9_inflation_global

Inflation results

RMSE (pp), relative to RW	$h = 1$	$h = 3$	$h = 6$	$h = 12$
RW	0.63 (1.00)	1.41 (1.00)	2.25 (1.00)	3.90 (1.00)
AR	0.57 (0.90)	1.28 (0.90)	2.03 (0.90)	4.28 (1.10)
Lasso	0.64 (1.01)	1.45 (1.02)	2.26 (1.01)	4.10 (1.05)
RF	0.64 (1.02)	1.49 (1.06)	2.33 (1.04)	4.25 (1.09)
GB local	0.70 (1.11)	1.64 (1.16)	2.37 (1.05)	4.72 (1.21)
GB global	0.64 (1.01)	1.37 (0.97)	2.37 (1.05)	4.54 (1.17)

- DM p against RW, AR: 0.019 ($h = 1$), 0.186 ($h = 3$), 0.388 ($h = 6$); GB global against GB local: 0.117, 0.113, 0.999, 0.828
- Test origins: 127 ($h = 1$) to 116 ($h = 12$) months; DM with squared errors and $h - 1$ autocovariances

Interpreting the inflation results

- The linear AR is the best model up to 6 months (about 10% below the random walk); no ML model beats the random walk significantly
  - at 12 months nothing beats the random walk: the 2022 peak (14.6% in November 2022) came from energy prices, not from the past of inflation
- The global GB is better than the local one at short horizons (RMSE 1.37 against 1.64 at $h = 3$), but not significantly ($p = 0.113$)
  - pooling 27 countries gives the trees enough rows; the local GB, with 120–240 rows per refit, overfits
- About 120 monthly test errors cannot separate models whose RMSEs differ by 5%: the honest conclusion is "no evidence"

Recap: Local and global models

- A global model pools many related series and lets flexible learners use more data
- Romanian inflation: AR is best; global GB beats local GB; nothing beats the random walk at 12 months
- Small samples: report DM p-values, not only RMSE rankings

Realised volatility: HAR against ML (1/2)

- Target: the log of the mean daily variance over the next 5 days; daily variance proxy from open, high, low and close prices Garman and Klass (1980)
  - $\hat{\sigma}_t^2 = [\ln(O_t/C_{t-\mathbf{1}})]^2 + \frac{\mathbf{1}}{\mathbf{2}}[\ln(H_t/L_t)]^2 - (2\ln 2 - 1)[\ln(C_t/O_t)]^2$: the overnight return squared plus the Garman–Klass term
  - O_t, H_t, L_t, C_t: the open, high, low and close of day t
  - volatility is persistent and predictable (Chapter 5): the right ground for ML
- **HAR** (heterogeneous autoregressive) model Corsi (2009): OLS on the log variance of the last day, week and month
  - $\ln \bar{\sigma}^2_{t+\mathbf{1}:t+\mathbf{5}} = c + \beta_d \ln \hat{\sigma}_t^2 + \beta_w \ln \bar{\sigma}^2_{t-\mathbf{4}:t} + \beta_m \ln \bar{\sigma}^2_{t-\mathbf{21}:t} + u_t$; $\bar{\sigma}^2_{a:b}$: the mean of $\hat{\sigma}^2$ over days a to b; u_t: the error
  - ML models add 8 features: lags of the daily variance, the quarterly mean, the return, its negative part and its absolute value

Realised volatility: HAR against ML (2/2)

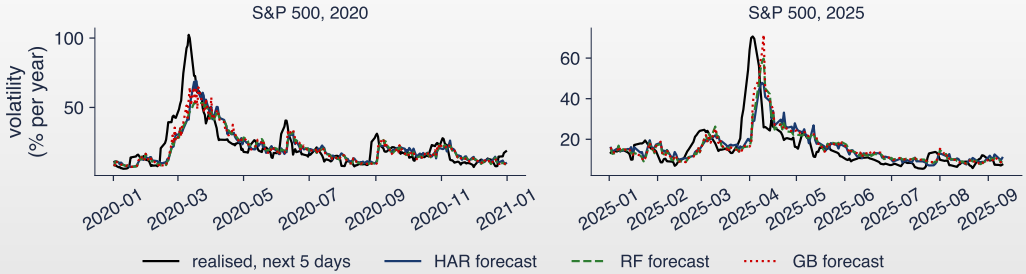
- Yearly walk-forward from 2013 with a 5-day gap between training and test
- Out-of-sample R^2 against HAR: $R^2_{\mathbf{HAR}} = 1 - \sum_t (y_t - \hat{y}_t)^2 / \sum_t (y_t - \hat{y}_t^{\mathbf{HAR}})^2$
  - y_t: the realised log variance; $\hat{y}_t$, $\hat{y}_t^{\mathbf{HAR}}$: the forecasts of the model and of HAR; positive: better than HAR
- QLIKE Patton (2011): $L(\sigma^2, h) = \sigma^2/h + \ln h$, the quasi-likelihood loss for variances (Chapter 5); lower is better
  - σ^2: the realised variance; h: the variance forecast; for a given σ^2 the loss is smallest at $h = \sigma^2$ and penalises forecasts that are too low more
  - robust to the noise of the variance proxy: it ranks forecasts as the true variance would
- Diebold–Mariano test (Chapter 4) on the daily QLIKE differences

Results: S&P 500 and DAX, 2013–2026

Model	S&P: R^2 vs HAR (%)	QLIKE	DM t	DAX: R^2 vs HAR (%)	QLIKE	DM t
HAR	+0.0	0.414	–	+0.0	1.023	–
Lasso	+7.1	0.400	−5.12	+7.9	1.015	−2.42
RF	+1.0	0.416	0.32	+3.5	1.024	0.03
GB	−0.5	0.416	0.35	+1.1	1.028	0.75
MLP	+8.3	0.398	−3.96	+8.6	1.015	−1.69

- 3 444 and 3 473 test days; DM $t < -1.96$: the model has a significantly lower QLIKE than HAR
- The lasso and the MLP gain 7–9% of R^2 over HAR and lower QLIKE significantly on the S&P 500; the random forest and GB do not beat HAR
- The gain comes from the extra features (the negative return: the leverage effect of Chapter 5), used smoothly; trees cut a smooth relation into steps

Volatility forecasts in 2020 and 2025



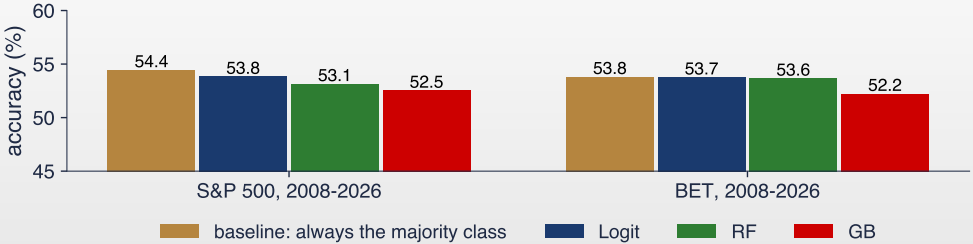
⊡ S&P 500: annualised realised volatility of the next 5 days and the HAR, random forest and GB forecasts, 2020 and 2025

TSA_ch9_finance

Forecasting the sign of the next-day return: choice of the baseline

- Classification: target $1\{r_{t+1} > 0\}$ (1 if tomorrow's return is positive, 0 otherwise); features: the last five returns, sums over 5, 21, 63 days, volatility over 21 and 63 days
  - models: logistic regression (logit), random forest, GB; yearly walk-forward from 2008
- **The right baseline** is not 50%: markets rise on more days than they fall
  - "always up" (the majority class of the training data) is right on 54.4% of S&P 500 days and 53.8% of BET days
- Accuracy, S&P 500: logit 53.8%, random forest 53.1%, GB 52.5%; BET: 53.7%, 53.6%, 52.2%
- No model beats the baseline; AUC (area under the ROC curve) 0.505–0.533, close to 0.5: no ranking skill either
  - AUC: the probability that a randomly chosen up day gets a higher model score than a randomly chosen down day; 0.5: no skill, 1: perfect ranking

Accuracy against the baseline



- Out-of-sample accuracy 2008–2026: 4 705 S&P 500 days and 4 686 BET days

TSA_ch9_finance

Interpreting the sign experiment

- A model with "53% accuracy" sounds skilful; against the 54.4% of "always up" it is worse than doing nothing
  - GB on the S&P 500 is significantly worse than the baseline ($p = 0.008$): flexible models fit noise
- Weak-form efficiency (Chapter 1): past returns contain almost no information about the sign of the next one
- Volatility is predictable, the direction is not: ML confirms what the statistical models of Chapters 1 and 5 showed

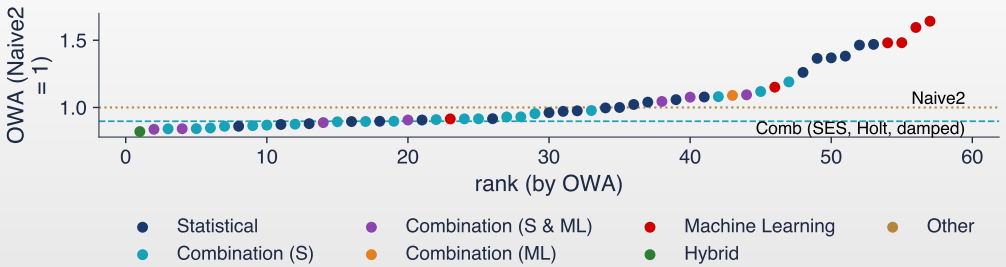
Recap: Volatility and the sign of returns

- Realised volatility: lasso and MLP with extra features beat HAR; trees do not
- Sign of returns: compare with the majority class, not with 50%; no model beats it

The M competitions

- Since 1982 Spyros Makridakis has organised open forecasting competitions: all methods forecast the same series, evaluated by the organisers on hidden data
  - the honest version of the walk-forward test: no participant can tune on the test data
- Makridakis, Spiliotis and Assimakopoulos (2018a): on 1 045 monthly series of M3, popular ML methods were less accurate than eight statistical methods at every horizon, and much slower
- M4 (2018), Makridakis, Spiliotis and Assimakopoulos (2020): 100 000 series of six frequencies, 61 methods, ranked by OWA (overall weighted average)
  - OWA $= \frac{1}{2}(\text{sMAPE}/\text{sMAPE}_{N2} + \text{MASE}/\text{MASE}_{N2})$; $N2$: Naive2, the naive forecast of the seasonally adjusted series; below 1: better than Naive2
  - sMAPE (symmetric mean absolute percentage error) $= \frac{200}{H}\sum_{h=1}^{H} |y_{T+h} - \hat{y}_{T+h}|/(|y_{T+h}| + |\hat{y}_{T+h}|)$, in %
- M5 (2020), Makridakis, Spiliotis and Assimakopoulos (2022): 42 840 hierarchical daily series of Walmart unit sales, 28-day horizon

M4: the accuracy of all methods



- Official evaluation file of M4 (github.com/Mcompetitions/M4-methods): OWA of the 59 ranked methods, submissions and benchmarks, coloured by type; 2 methods with OWA above 1.7 are off the chart

TSA_ch9_case_studies

Interpreting M4

- Winner: the hybrid ES-RNN of Smyl (2020), exponential smoothing inside an LSTM trained on all series (a global model): OWA 0.821, sMAPE 11.37%
  - Comb, the average of SES, Holt and damped trend: OWA 0.898, sMAPE 12.55%; Naive2: sMAPE 13.56%
- The 6 pure ML methods: the best ranked 23 (OWA 0.915); none beat Comb, only one beat Naive2 Makridakis, Spiliotis and Assimakopoulos (2018b)
  - 12 of the 17 best methods were combinations
- Lesson: on short, heterogeneous series, combine statistical models, or let ML learn across series; pure local ML loses

M5: accuracy of gradient boosting on retail data



A Walmart store: the M5 data are the daily unit sales of 3 049 products in 10 stores

Photo: Walmart Corporate; CC BY 2.0; Wikimedia Commons

- 5 507 teams from 101 countries; error measure WRMSSE (weighted root mean squared scaled error)
  - RMSSE: the RMSE of the forecasts over the in-sample RMSE of the one-step naive forecast
  - WRMSSE: the average RMSSE, weighted by sales value
  - the winner (WRMSSE 0.520) was 22.4% more accurate than the best benchmark, bottom-up exponential smoothing (0.671)
- Most top methods used LightGBM Ke et al. (2017)
  - the winner averaged LightGBM models pooled by store, category and department, recursive and direct
  - global models, rich calendar and price features, many related series: the conditions where ML wins

Gu, Kelly and Xiu (2020): machine learning for stock returns

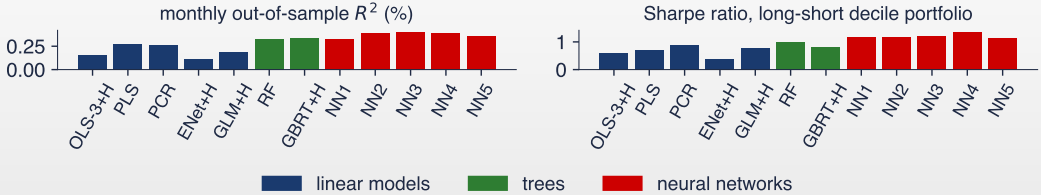


Charles M. Harper Center, University of Chicago Booth School of Business

Photo: Michael Barera (2013); CC BY-SA 4.0; Wikimedia Commons

- Gu, Kelly and Xiu (2020): monthly returns of about 30 000 US stocks, 1957–2016, 920 predictors
  - 12 methods, from OLS to neural networks with five layers
  - walk-forward: 18 years of training, 12 of validation, 30 years of test, refitted yearly
- Out-of-sample monthly R^2: at most 0.40% (three-layer network); OLS with all predictors: -3.46%
- Small R^2, large value: a long-short portfolio sorted on the forecasts reaches an annual Sharpe ratio of 1.35 (four layers), against 0.61 for OLS-3
  - long-short: buy the decile with the highest forecasts, sell the decile with the lowest
  - Sharpe ratio: the mean annual return divided by its annual standard deviation

Gu, Kelly and Xiu (2020): out-of-sample R^2 and Sharpe ratios



- Published numbers: Table 1 (monthly out-of-sample R^2, all stocks) and Table 7 (annualised Sharpe ratio of the value-weighted long-short decile portfolio); +H: Huber loss
- OLS-3: OLS on size, book-to-market and momentum; PLS, PCR: partial least squares, principal component regression; ENet: elastic net
- GLM: generalised linear model with splines; RF: random forest; GBRT: gradient-boosted regression trees; NN1–NN5: networks with 1–5 hidden layers

TSA_ch9_case_studies

Recap: Case studies

- M4: combinations and a hybrid global model win; pure local ML loses to simple benchmarks
- M5: global LightGBM models with calendar and price features win clearly
- Gu, Kelly and Xiu: tiny but real predictability in the cross-section of stocks; nonlinear models add value

Foundation models: a pointer to Chapter 11

- A **foundation model** for time series is a large network (a transformer) pre-trained on millions of series and used without retraining (zero-shot)
  - the extreme global model: Chronos, TimesFM, Moirai, Lag-Llama (self-study, Chapter 11)
- The same rules apply: walk-forward on data the model has not seen in pre-training, the same benchmarks, MASE and DM

Possible contribution of AI

- **Code**: a first draft of the feature table, the walk-forward loop and the evaluation table
- **Explanation**: a second explanation of boosting, of the LSTM gates or of conformal prediction
- **Exploration**: many feature sets and hyperparameters at once (on validation data only)
- Example prompt
  - `Write Python code that builds lag, rolling-mean and calendar features for daily electricity load, trains HistGradientBoostingRegressor with the direct strategy for horizons 1-14, evaluates it walk-forward against the seasonal naive forecast with MASE and the Diebold-Mariano test, and refits every preprocessing step inside each training window.`

Checks you must run

- ⊡ Every feature uses data up to the origin only: check the shifts of lags and rolling windows on a printed table
- ⊡ No `KFold(shuffle=True)`, no scaler or feature selection fitted on the full sample
- ⊡ The benchmarks are there: naive or seasonal naive, ETS or ARIMA, on the same origins
- ⊡ The comparison has a DM test; the classification has the majority-class baseline
- ⊡ Seeds are fixed and the result survives another seed
- ⊡ Every cited reference exists: check the DOI

Key takeaways

- ML forecasting = supervised learning on a table of lags, rolling statistics and calendar features known at the origin
- Walk-forward validation only; leakage produces spectacular and false scores
- Ridge and lasso, random forest, GB, MLP, LSTM: each has a complexity knob set by validation
- Trees cannot extrapolate: model changes, not levels
- Our data: linear models with good features match or beat ML on load and inflation; ML helps for volatility; nothing predicts the sign of returns
- ML wins with many related series (global models, M4 hybrid, M5 LightGBM); always compare with benchmarks, MASE and DM

Key formulas

Quantity	Formula
Direct forecast	$\hat{y}_{T+h} = \hat{f}_h(y_T, \ldots, y_{T-p+\mathbf{1}}, \mathbf{z}_{T+h})$, $\mathbf{z}_{T+h}$: calendar features
Ridge, lasso	$\min \sum_t (y_t - \mathbf{x}_t'\beta)^{\mathbf{2}} + \lambda \sum_j \beta_j^{\mathbf{2}}$; $\quad +\lambda \sum_j \lvert\beta_j\rvert$
Bias–variance	$E[(y_{\mathbf{0}} - \hat{f}(x_{\mathbf{0}}))^{\mathbf{2}}] = \text{bias}^{\mathbf{2}} + \text{Var}(\hat{f}(x_{\mathbf{0}})) + \sigma^{\mathbf{2}}$
Boosting	$F_m(\mathbf{x}) = F_{m-\mathbf{1}}(\mathbf{x}) + \nu\, g_m(\mathbf{x})$
MLP	$\hat{y} = c + \sum_j v_j\, g(b_j + \mathbf{w}_j'\mathbf{x})$
LSTM	$c_t = f_t \odot c_{t-\mathbf{1}} + i_t \odot \tilde{c}_t, \quad h_t = o_t \odot \tanh(c_t)$
Pinball loss	$L_\tau(y, q) = \max(\tau(y-q), (\tau-1)(y-q))$
Split conformal	$\hat{y} \pm \hat{q}, \quad \hat{q} = \lvert e\rvert_{(\lceil (n+\mathbf{1})(\mathbf{1}-\alpha)\rceil)}$
MASE	$\text{MAE}/\frac{\mathbf{1}}{n-m}\sum_{t>m} \lvert y_t - y_{t-m}\rvert$

Self-assessment

- **Question**: a 7-day moving average centred on t is used as a feature for y_{t+1}. Is that allowed?
  - **Answer**: no: it contains $y_{t+1}, y_{t+2}, y_{t+3}$; use the trailing mean of $y_{t-6}, \dots, y_t$
- **Question**: a random forest trained on GDP levels up to 2019 forecasts 2024. What goes wrong?
  - **Answer**: its forecast cannot exceed the largest level seen in training; model growth rates
- **Question**: a classifier predicts the sign of daily returns with 53% accuracy. Is it useful?
  - **Answer**: only if it beats the share of up days in the test period (about 54% for the S&P 500), significantly
- Next: Chapter 10, state space models, the Kalman filter and Markov switching

References (1/4)

- Ben Taieb, S., Bontempi, G., Atiya, A. F., & Sorjamaa, A. (2012). A review and comparison of strategies for multi-step ahead time series forecasting based on the NN5 forecasting competition. *Expert Systems with Applications*, 39(8), 7067–7083. doi:10.1016/j.eswa.2012.01.039
- Bergmeir, C., Hyndman, R. J., & Koo, B. (2018). A note on the validity of cross-validation for evaluating autoregressive time series prediction. *Computational Statistics & Data Analysis*, 120, 70–83. doi:10.1016/j.csda.2017.11.003
- Breiman, L. (2001). Random forests. *Machine Learning*, 45(1), 5–32. doi:10.1023/A:1010933404324
- Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785–794. doi:10.1145/2939672.2939785
- Corsi, F. (2009). A simple approximate long-memory model of realized volatility. *Journal of Financial Econometrics*, 7(2), 174–196. doi:10.1093/jjfinec/nbp001
- Diebold, F. X., & Mariano, R. S. (1995). Comparing predictive accuracy. *Journal of Business & Economic Statistics*, 13(3), 253–263. doi:10.1080/07350015.1995.10524599
- Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5), 1189–1232. doi:10.1214/aos/1013203451
- Garman, M. B., & Klass, M. J. (1980). On the estimation of security price volatilities from historical data. *The Journal of Business*, 53(1), 67–78. doi:10.1086/296072
- Gu, S., Kelly, B., & Xiu, D. (2020). Empirical asset pricing via machine learning. *The Review of Financial Studies*, 33(5), 2223–2273. doi:10.1093/rfs/hhaa009

References (2/4)

- Harvey, D., Leybourne, S., & Newbold, P. (1997). Testing the equality of prediction mean squared errors. *International Journal of Forecasting*, 13(2), 281–291. doi:10.1016/S0169-2070(96)00719-4
- Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning* (2nd ed.). Springer. doi:10.1007/978-0-387-84858-7
- Hewamalage, H., Bergmeir, C., & Bandara, K. (2021). Recurrent neural networks for time series forecasting: Current status and future directions. *International Journal of Forecasting*, 37(1), 388–427. doi:10.1016/j.ijforecast.2020.06.008
- Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8), 1735–1780. doi:10.1162/neco.1997.9.8.1735
- Hoerl, A. E., & Kennard, R. W. (1970). Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1), 55–67. doi:10.1080/00401706.1970.10488634
- Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5), 359–366. doi:10.1016/0893-6080(89)90020-8
- Huang, C., & Petukhina, A. (2022). *Applied Time Series Analysis and Forecasting with Python*. Springer. doi:10.1007/978-3-031-13584-2
- Hyndman, R. J., & Athanasopoulos, G. (2021). *Forecasting: Principles and Practice* (3rd ed.), Section 12.4, Neural network models. OTexts. otexts.com/fpp3/nnetar.html
- Hyndman, R. J., & Koehler, A. B. (2006). Another look at measures of forecast accuracy. *International Journal of Forecasting*, 22(4), 679–688. doi:10.1016/j.ijforecast.2006.03.001

References (3/4)

- Januschowski, T., Gasthaus, J., Wang, Y., Salinas, D., Flunkert, V., Bohlke-Schneider, M., & Callot, L. (2020). Criteria for classifying forecasting methods. *International Journal of Forecasting*, 36(1), 167–177. doi:10.1016/j.ijforecast.2019.05.008
- Kaufman, S., Rosset, S., Perlich, C., & Stitelman, O. (2012). Leakage in data mining: Formulation, detection, and avoidance. *ACM Transactions on Knowledge Discovery from Data*, 6(4), 1–21. doi:10.1145/2382577.2382579
- Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T.-Y. (2017). LightGBM: A highly efficient gradient boosting decision tree. *Advances in Neural Information Processing Systems*, 30. proceedings.neurips.cc
- Koenker, R., & Bassett, G. (1978). Regression quantiles. *Econometrica*, 46(1), 33–50. doi:10.2307/1913643
- Lei, J., G'Sell, M., Rinaldo, A., Tibshirani, R. J., & Wasserman, L. (2018). Distribution-free predictive inference for regression. *Journal of the American Statistical Association*, 113(523), 1094–1111. doi:10.1080/01621459.2017.1307116
- Lim, B., & Zohren, S. (2021). Time-series forecasting with deep learning: A survey. *Philosophical Transactions of the Royal Society A*, 379(2194), 20200209. doi:10.1098/rsta.2020.0209
- Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2018a). Statistical and machine learning forecasting methods: Concerns and ways forward. *PLOS ONE*, 13(3), e0194889. doi:10.1371/journal.pone.0194889
- Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2018b). The M4 Competition: Results, findings, conclusion and way forward. *International Journal of Forecasting*, 34(4), 802–808. doi:10.1016/j.ijforecast.2018.06.001

References (4/4)

- Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2020). The M4 Competition: 100,000 time series and 61 forecasting methods. *International Journal of Forecasting*, 36(1), 54–74. doi:10.1016/j.ijforecast.2019.04.014
- Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2022). M5 accuracy competition: Results, findings, and conclusions. *International Journal of Forecasting*, 38(4), 1346–1364. doi:10.1016/j.ijforecast.2021.11.013
- Medeiros, M. C., Vasconcelos, G. F. R., Veiga, Á., & Zilberman, E. (2021). Forecasting inflation in a data-rich environment: The benefits of machine learning methods. *Journal of Business & Economic Statistics*, 39(1), 98–119. doi:10.1080/07350015.2019.1637745
- Montero-Manso, P., & Hyndman, R. J. (2021). Principles and algorithms for forecasting groups of time series: Locality and globality. *International Journal of Forecasting*, 37(4), 1632–1653. doi:10.1016/j.ijforecast.2021.03.004
- Patton, A. J. (2011). Volatility forecast comparison using imperfect volatility proxies. *Journal of Econometrics*, 160(1), 246–256. doi:10.1016/j.jeconom.2010.03.034
- Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323(6088), 533–536. doi:10.1038/323533a0
- Salinas, D., Flunkert, V., Gasthaus, J., & Januschowski, T. (2020). DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 36(3), 1181–1191. doi:10.1016/j.ijforecast.2019.07.001
- Smyl, S. (2020). A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting. *International Journal of Forecasting*, 36(1), 75–85. doi:10.1016/j.ijforecast.2019.03.017
- Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B*, 58(1), 267–288. doi:10.1111/j.2517-6161.1996.tb02080.x