








Statistics of Financial Markets

Chapter 13: Machine learning



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
- **ARCH**: AutoRegressive Conditional Heteroskedasticity
- **AUC**: Area Under the (ROC) Curve
- **BET**: Bucharest Exchange Trading (index)
- **BVB**: Bursa de Valori București (the Bucharest Stock Exchange)
- **CV**: Cross-Validation
- **DM**: Diebold–Mariano (test)
- **DSR**: Deflated Sharpe Ratio
- **EODHD**: EOD Historical Data (market-data provider)
- **GARCH**: Generalised AutoRegressive Conditional Heteroskedasticity
- **GB**: Gradient Boosting
- **GJR-GARCH**: Glosten–Jagannathan–Runkle GARCH
- **HAR**: Heterogeneous AutoRegressive (model)
- **HS**: Historical Simulation
- **IBM**: International Business Machines (company)
- **MA**: Moving Average
- **ML**: Machine Learning
- **MLP**: MultiLayer Perceptron
- **MSE**: Mean Squared Error
- **NN**: Neural Network (NNk: k hidden layers)

Acronyms used in this material (2/2)

- **NN-ARCH**: Neural-Network ARCH
- **NN1**: Neural Network with 1 hidden layer
- **NN3**: Neural Network with 3 hidden layers
- **NN4**: Neural Network with 4 hidden layers
- **OLS**: Ordinary Least Squares
- **OLS-3**: OLS with 3 predictors (size, book-to-market, momentum)
- **OMV**: Österreichische Mineralölverwaltung — Austrian Mineral Oil Administration (OMV group)
- **PSR**: Probabilistic Sharpe Ratio
- **QGB**: Quantile Gradient Boosting
- **QLIKE**: Quasi-LIKElihood (loss function)
- **QR**: Quantile Regression
- **RBF**: Radial Basis Function (network)
- **RF**: Random Forest
- **RMSE**: Root Mean Squared Error
- **ROC**: Receiver Operating Characteristic (curve)
- **RV**: Realised Variance
- **SFE**: Statistics of Financial Markets, Exercises (Quantlet collection of the textbook)
- **SHAP**: SHapley Additive exPlanations
- **SSE**: Sum of Squared Errors
- **US**: United States

Today's Question and Route

- **Question**: can flexible prediction methods forecast financial risk and returns better than the models of Chapters 7–10, and how do we check it honestly?
  - **machine learning** (ML): algorithms that learn a prediction rule from data, judged by their accuracy on new data
- **Route** of the chapter
  - prediction against inference; overfitting, bias and variance; validation for time series without look-ahead bias
  - ridge and lasso; trees, random forest, gradient boosting; neural networks and the neural-network ARCH of the textbook
  - three applications: realised volatility, the sign of returns, VaR 1%; data snooping; a landmark paper

Learning Outcomes

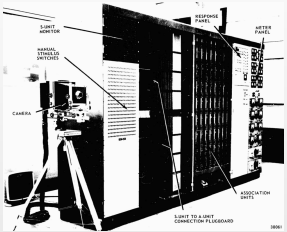
- Explain the difference between prediction and inference, and the bias–variance trade-off
- Design a walk-forward evaluation with purging, and recognise look-ahead bias and leakage
- Fit ridge, lasso, a regression tree, a random forest, gradient boosting and a small neural network
- Compare forecasts with out-of-sample R^2, QLIKE and the Diebold–Mariano test, against a proper baseline
- Forecast VaR 1% by quantile regression and correct a Sharpe ratio for the number of strategies tried

Reading and Tools

- Textbook: Franke, Härdle & Hafner (2019), Ch. 19 (neural networks)
  - statistical learning in Python: James et al. (2023); in depth: Hastie, Tibshirani & Friedman (2009)
  - machine learning in finance: Kelly & Xiu (2023); López de Prado (2018)
- Python Quantlets of this chapter: Quantlets/Ch_13
  - ported from the SFE Quantlets SFEnnarch and SFEnnjpyusd; models from scikit-learn (small, fixed seeds, a few minutes in Colab)
- Lecture notebook: open in Google Colab
- Video courses: Machine learning in Financial Risk; Random Forests

The Perceptron, 1958

- Rosenblatt (1958): a machine that learns to classify images by adjusting weights after each error
  - the **perceptron**: output 1 if $w_0 + w^\top x > 0$, else 0
  - x: the inputs (photocell signals); w: their weights, learned from examples; w_0: the threshold term; $w^\top x = \sum_j w_j x_j$
  - the Mark I Perceptron (Cornell, 1960): 400 photocells, weights set by electric motors
- Three later milestones
  - 1986: back-propagation trains networks with hidden layers Rumelhart, Hinton & Williams (1986)
  - 2001: random forests Breiman (2001b) and gradient boosting Friedman (2001)
  - 2024: Nobel Prize in Physics to Hopfield and Hinton for neural networks



The Mark I Perceptron, from its operator's manual

Figure: J.C. Hay, A.E. Murray (1960); public domain; Wikimedia Commons

Two Cultures of Statistical Modelling

- Breiman (2001a), a statistician at Berkeley, describes two cultures
  - **data modelling**: assume a stochastic model (the linear regression $y = x^\top \beta + \varepsilon$, GARCH), estimate it, test its parameters
  - **algorithmic modelling**: treat the mechanism as unknown; find a rule $\hat{f}$ that predicts y well on new data
- This course used the first culture in Chapters 2–11
  - this chapter adds the second, and asks when it pays off in finance



Evans Hall, University of California, Berkeley, home of the Department of Statistics

Photo: Gabriel Classon (2021); CC BY 2.0; Wikimedia Commons

Prediction and Inference

- **Inference**: is $\beta_j \neq 0$? how large is the effect? (standard errors, tests, Chapter 6)
  - example: does the leverage effect exist (GJR-GARCH, Chapter 9)?
- **Prediction**: how close is $\hat{y}$ to y on data the model has never seen?
  - example: next week's volatility; tomorrow's VaR 1%
  - the coefficients may be uninterpretable; what matters is the error on new data
- The two goals can conflict
  - a biased estimator (ridge, lasso) can predict better than the unbiased OLS
  - a model that predicts well does not prove a causal mechanism

Supervised Learning: Notation and Loss (1/2)

- Data (x_t, y_t), $t = 1, \ldots, n$
  - t: the index of the observation (a day, a week); n: the number of observations
  - $x_t \in \mathbb{R}^p$: the **features**, p numbers known at time t (past volatilities, past returns)
  - y_t: the **target**, the quantity we want to forecast (next week's volatility)
- Two types of problems
  - **regression**: y continuous (volatility, return)
  - **classification**: y a class (up/down, default; Chapter 12)

Supervised Learning: Notation and Loss (2/2)

- Learning = choose $\hat{f}$ in a family $\mathcal{F}$ that minimises the average **loss** $\frac{1}{n}\sum_t L(y_t, f(x_t))$
  - $f(x_t)$: the forecast of a rule f for observation t; $L(y, \hat{y})$: the cost of forecasting $\hat{y}$ when y is observed
  - $\mathcal{F}$: the candidate rules (all linear regressions, all trees of depth 3); $\hat{f}$: the rule chosen from the data
- Losses used in this course
  - squared error $(y - \hat{y})^2$ (MSE: mean squared error); QLIKE for variances (Chapter 8)
  - quantile (pinball) loss for VaR (Chapter 10); log loss for probabilities (Chapter 12)
- A **hyperparameter** is chosen before the fit
  - examples: tree depth, penalty λ, number of neurons
  - the parameters (coefficients, weights) are learned by the fit

Strengths and Limitations of Machine Learning in Finance

- **Helps**: targets with structure and enough data
  - volatility is persistent (Chapters 8–9): a lot to predict, nonlinear effects possible
  - cross-sections with thousands of stocks and many characteristics (the case study)
- **Struggles**: a low **signal-to-noise ratio**
  - daily returns are close to unpredictable (Chapter 7): a flexible model fits the noise
  - markets change (regimes, crises): the past is an imperfect training set
  - one price history: we cannot repeat the experiment, and the observations are dependent

Training Error and Test Error

- **Training error**: the average loss on the data used for the fit; **test error**: on new data from the same process
  - the training error always falls as the model becomes more flexible
- **Overfitting**: the model learns the noise of the training sample
  - symptom: a small training error and a large test error
  - a tree with one leaf per observation has zero training error and predicts badly
- **Underfitting**: the model is too rigid to capture the signal
  - example: a constant forecast of volatility in a world with volatility clustering

The Bias–Variance Decomposition (1/2)

- Model: $y = f(x) + \varepsilon$, $E[\varepsilon] = 0$, $\mathrm{Var}(\varepsilon) = \sigma^2$
  - f: the true, unknown relation between x and y; ε: the noise, unpredictable from x, with variance σ^2
  - $\hat{f}$: the estimate of f from a random training sample; another sample would give another $\hat{f}$
- Expected test error at a new point x_0, with $y_0 = f(x_0) + \varepsilon_0$:

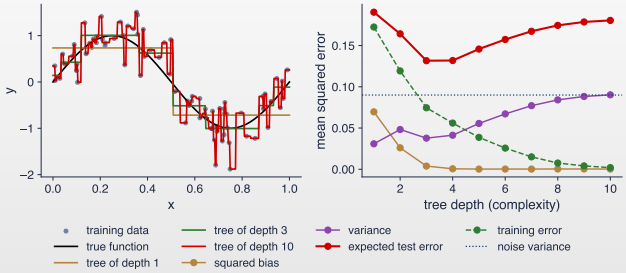
$$E\left[(y_0 - \hat{f}(x_0))^2\right] = \underbrace{\left(E[\hat{f}(x_0)] - f(x_0)\right)^2}_{\text{bias}^2} + \underbrace{\mathrm{Var}\left(\hat{f}(x_0)\right)}_{\text{variance}} + \underbrace{\sigma^2}_{\text{noise}}$$

  - $E[\cdot]$: the average over all possible training samples and over the noise ε_0
  - bias: how far the average estimate is from the truth; variance: how much $\hat{f}(x_0)$ moves from one sample to another

The Bias–Variance Decomposition (2/2)

- Derivation: write $y_0 - \hat{f} = (f - E\hat{f}) + (E\hat{f} - \hat{f}) + \varepsilon_0$, all at x_0
  - square and take the expectation: the three squares give bias^2, variance and σ^2
  - the cross terms have mean 0: $E[E\hat{f} - \hat{f}] = 0$, and ε_0 is independent of the training sample
- Reading the decomposition
  - a rigid model: large bias, small variance
  - a flexible model: small bias, large variance
  - σ^2 is the floor: no model goes below it; in daily returns σ^2 is almost everything

Bias and Variance of Regression Trees



- Simulation: $y = \sin(2\pi x) + \varepsilon$, $\sigma^2 = 0.09$, $n = 80$, 300 training samples; left: one sample with trees of depth 1, 3 and 10
- Depth 1: bias^2 0.070, variance 0.031; depth 10: bias^2 0.000, variance 0.090; the test error is smallest at depth 3 (0.132)

SFM_ch13_validation

Interpretation: the U-Shaped Test Error

- The training error falls from 0.172 to 0.002; the test error falls, then rises
  - depth 10 has almost zero training error and a test error of 0.180, worse than depth 3
- Consequences for practice
  - never select a model by its training error
  - complexity is a hyperparameter: choose it on data that were not used for the fit
  - in finance the noise is large, so the best model is usually simpler than in this simulation

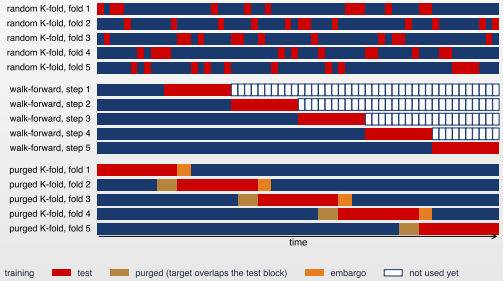
Training, Validation and Test Samples

- Three roles for the data
  - **training**: estimate the parameters of each candidate model
  - **validation**: choose the hyperparameters and the model
  - **test**: measure the error of the final choice, once
- Every look at the test sample turns it into a validation sample
  - trying ten models on the test data and reporting the best is model selection, not testing
- **K-fold cross-validation**: split the data into K folds; each fold is the validation set once, the others train the model
  - valid for i.i.d. data; for a time series only under strong conditions (no dependence left in the errors) Bergmeir, Hyndman & Koo (2018)

Look-Ahead Bias and Leakage

- **Look-ahead bias**: the forecast for day t uses information that was not available at the close of day $t-1$
  - features computed with future prices (a centred moving average, a full-sample mean)
  - scaling or standardising with statistics of the whole sample; revised data; today's index members (survivorship)
- **Leakage**: information about the test target reaches the training sample
  - overlapping targets: the 21-day return from day t and from day $t+1$ share 20 days
  - a random fold puts day $t+1$ in training and day t in test: the model has seen the answer
- Both make a model look better than it can be in real time

Three Validation Schemes



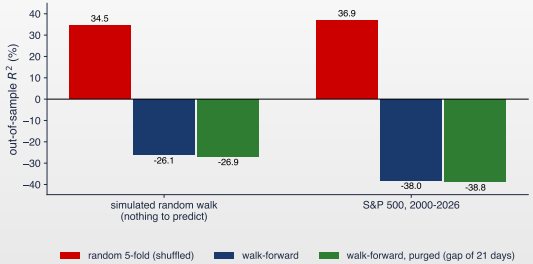
- **Walk-forward**: train on the past, test on the next block, move forward; the expanding window keeps all the past
- **Purged K-fold with embargo** López de Prado (2018): drop the training rows whose targets overlap the test block (purging) and a few rows after it (embargo)

SFM_ch13_validation

Walk-Forward Validation, Step by Step

- The design used in this chapter for the S&P 500 (data from 8 April 2008)
  - one fold per calendar year from 2013: 14 folds; the model is re-estimated each January
  - training window: all earlier days (expanding); test: every day of the year
- Purging: the target is the mean variance of the next 5 days
  - for 2013 the last 5 training rows (from 24 December 2012) are dropped: their targets reach into 2013
  - the 2013 fold trains on 1 188 days and tests on 252; the 2026 fold trains on 4 458 days
- Inside each training window, hyperparameters are tuned by a walk-forward split of the training data only
  - no embargo is needed: the test block always comes after the training block

Leakage in Practice: Random Folds on Overlapping Targets



- Random forest for the next 21-day return from past returns and volatilities; out-of-sample R^2 against the training mean
- Simulated random walk: random 5-fold 34.5%, walk-forward -26.1%; S&P 500 (6 631 days): 36.9% and -38.0%

SFM_ch13_validation

Interpretation: Leakage in the Forecast of a Random Walk

- The random walk has nothing to predict, yet random folds report $R^2 = 34.5\%$
  - the forest memorises neighbouring days, whose targets share 20 of 21 returns with the test day
- Walk-forward gives a negative R^2: the forest fits noise and loses against the training mean
  - purging (a gap of 21 days) changes little here, because in walk-forward only the boundary rows overlap
- Rule: with overlapping targets or dependent data, never shuffle the time order

OLS with Many Correlated Features

- OLS (ordinary least squares): $\hat{\beta} = \arg\min_\beta \sum_t (y_t - x_t^\top \beta)^2 = (X^\top X)^{-1} X^\top y$
  - X: the $n \times p$ matrix whose row t is $x_t^\top$; y: the vector of targets; $\arg\min_\beta$: the β that minimises the sum
  - unbiased, but its variance $\sigma^2 (X^\top X)^{-1}$ (σ^2: the error variance) explodes when features are many or strongly correlated
- Volatility features are highly correlated: today's, this week's and this month's realised variance
  - OLS may give large coefficients of opposite signs that cancel in sample and fail out of sample
- **Penalised regression**: minimise the squared error plus a penalty on the size of β
  - accept a little bias for a large reduction in variance (the trade-off of Section 2)

Ridge Regression (1/2)

- Hoerl & Kennard (1970): minimise the squared error plus a penalty on the squared coefficients

$$\hat{\beta}^{ridge} = \arg\min_{\beta} \sum_t (y_t - x_t^\top \beta)^2 + \lambda \sum_j \beta_j^2 = (X^\top X + \lambda I)^{-1} X^\top y$$

- Notation
  - β_j: the coefficient of feature j; $\sum_j \beta_j^2$: the penalty term, large when the coefficients are large
  - $\lambda \geq 0$: the strength of the penalty; $\lambda = 0$ gives OLS, $\lambda \to \infty$ gives $\hat{\beta} \to 0$
  - I: the $p \times p$ identity matrix
- Consequence
  - $X^\top X + \lambda I$ is always invertible for $\lambda > 0$: ridge works even with more features than observations

Ridge Regression (2/2)

- **One feature**, centred (mean 0): $S_{xx} = \sum_t x_t^2$, $S_{xy} = \sum_t x_t y_t$

$$\hat{\beta}^{\,ridge} = \frac{S_{xy}}{S_{xx} + \lambda} = \frac{S_{xx}}{S_{xx} + \lambda}\,\hat{\beta}^{\,OLS}, \qquad \hat{\beta}^{\,OLS} = \frac{S_{xy}}{S_{xx}}$$

  - the factor $S_{xx}/(S_{xx} + \lambda)$ lies in $(0, 1]$: the share of the OLS coefficient that ridge keeps
- Example: $S_{xx} = 100$, $S_{xy} = 30$
  - OLS: $30/100 = 0.30$
  - ridge with $\lambda = 50$: $30/150 = 0.20$ (shrinkage factor 0.67)
- Ridge shrinks all coefficients towards zero, but sets none exactly to zero

The Lasso

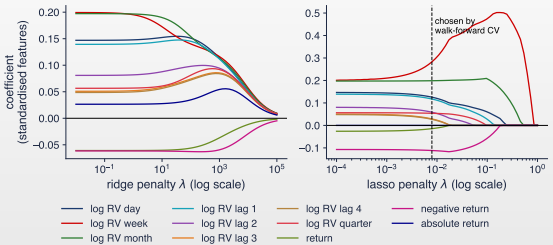
- Tibshirani (1996): $\hat{\beta}^{lasso} = \arg\min_\beta \sum_t (y_t - x_t^\top \beta)^2 + \lambda \sum_j |\beta_j|$
  - the absolute value has a kink at 0: some coefficients become **exactly zero** (variable selection)
- One feature: **soft thresholding**
  - $\hat{\beta}^{lasso} = \text{sign}(S_{xy})\ \max(|S_{xy}| - \lambda/2,\ 0)/S_{xx}$
  - $\text{sign}(S_{xy})$: $+1$ or -1; if $|S_{xy}| \leq \lambda/2$, the coefficient is exactly 0
  - same data, $\lambda = 20$: $(30 - 10)/100 = 0.20$; zero for $\lambda \geq 60$
- In practice
  - standardise the features first (the penalty depends on their units), with the training mean and standard deviation only
  - choose λ by walk-forward cross-validation; elastic net combines both penalties



Robert Tibshirani, Stanford University

Photo: R. Tibshirani (2012); CC BY-SA 3.0; Wikimedia Commons

Ridge and Lasso Paths for the Volatility Features



- Target: log of the mean variance proxy of the next 5 days, S&P 500, 2008–2012 (1 193 days); 11 standardised features (Section 7)
- Lasso with $\lambda = 0.0079$ (walk-forward CV): 10 of 11 features kept; the week coefficient rises from $+0.199$ (OLS) to $+0.283$, the absolute return drops out
- The design is exactly collinear, $|r| = r - 2\min(r, 0)$ (rank 10): the OLS coefficients of r, $\min(r, 0)$, $|r|$ are not unique (shown: the minimum-norm solution); the others are unique

SFM_ch13_shrinkage_trees

Interpretation of the Paths

- Ridge (left): all coefficients shrink smoothly and together; correlated features share the weight
  - the negative return keeps a negative sign until the end: falling markets raise future volatility (the leverage effect, Chapter 9)
- Lasso (right): the features enter one by one as λ falls
  - the weekly variance enters first: the most informative single feature
  - at the chosen λ the lasso keeps the HAR structure (day, week, month) plus the negative return, -0.112
- The lasso selects; it does not test: a zero coefficient is not a p-value

Regression Trees

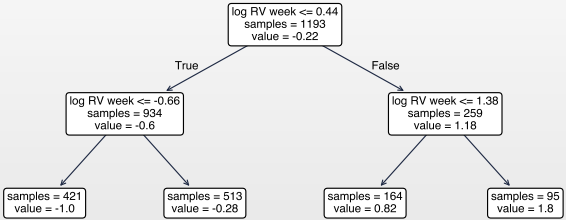
- A **regression tree** splits the feature space into rectangles and predicts the mean of y in each **leaf**
  - split rule: a feature j and a threshold s: left $x_j \leq s$, right $x_j > s$
- **Greedy fit**: at each node choose (j, s) that minimises $\mathrm{SSE} = \sum_{\text{left}}(y - \bar{y}_L)^2 + \sum_{\text{right}}(y - \bar{y}_R)^2$
  - SSE: sum of squared errors; $\bar{y}_L$, $\bar{y}_R$: the means of y in the left and right groups (the two forecasts)
  - repeat in each half until a stopping rule: maximum depth, minimum number of observations per leaf
- Strengths and weaknesses
  - nonlinear effects and interactions without specifying them; insensitive to monotone transformations of x
  - high variance: a small change in the data can change the whole tree (Section 2)

Worked Example: One Split, Step by Step

s	$\bar{y}_L$	$\bar{y}_R$	SSE
0.70	0.70	1.52	1.708
0.90	0.80	1.68	1.248
1.25	0.80	1.97	0.227
1.75	1.00	2.15	0.505
2.20	1.22	2.20	1.468

- Six weeks: x = this week's volatility (%), y = next week's
  - x: 0.6, 0.8, 1.0, 1.5, 2.0, 2.4; y: 0.7, 0.9, 0.8, 1.6, 2.1, 2.2
  - no split: $\bar{y} = 1.38$, SSE $= 2.268$
- Candidates: the midpoints between consecutive x values
  - best split $s = 1.25$: leaves 0.80 and 1.97, SSE $= 0.227$
  - one split explains $1 - 0.227/2.268 = 0.90$ of the variation

A Depth-2 Tree for Next Week's Volatility



- S&P 500, 2008–2012, target and features in log variance ($\%^2$ per day); *value*: the mean of the target in the node
- Interpretation: the first split, log RV week ≤ 0.44, is an annual volatility of about 19.8%; all splits use the weekly variance, the feature the lasso picked first

SFM_ch13_shrinkage_trees

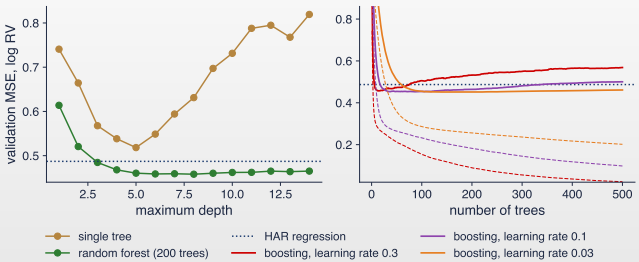
Random Forest

- Breiman (2001b): average B deep trees, each grown on a bootstrap sample (Chapter 2) of the training data (**bagging**)
  - at each split only a random subset of the features is tried: the trees become less correlated
- Why averaging works: B trees, each with variance σ^2 and pairwise correlation ρ
  - $\mathrm{Var}(\text{average}) = \rho\sigma^2 + (1-\rho)\sigma^2/B$: the part $\rho\sigma^2$, common to all trees, does not fall with B
  - with $\rho = 0.3$, $\sigma^2 = 1$: $B = 1$: 1.000; $B = 10$: 0.370; $B = 300$: 0.302
  - more trees never overfit; less correlated trees (fewer features per split) reduce the floor $\rho\sigma^2$
- Hyperparameters used here: 300 trees, at least 20 observations per leaf, half of the features tried at each split

Gradient Boosting

- Friedman (2001): build the model step by step, each new small tree fitting what the previous ones missed
  - start with $F_0 = \bar{y}$; at step m, fit a shallow tree h_m to the residuals $y_t - F_{m-1}(x_t)$
  - update $F_m = F_{m-1} + \nu\, h_m$, with the **learning rate** $0 < \nu \leq 1$
  - for squared loss the residuals are minus the gradient of the loss: hence the name
- Random forest against boosting
  - forest: deep trees in parallel, averaging reduces variance
  - boosting: shallow trees in sequence, each step reduces bias; too many steps overfit
- Any differentiable loss works: quantile loss gives quantile boosting (Section 9)

One Tree, a Forest and Boosting on Volatility Data



- S&P 500, training 2008–2018 (2 698 days), validation 2019–2022 (1 008 days); dashed right: training MSE; dotted: HAR regression (0.487)
- Single tree: best 0.518 at depth 5, 0.819 at depth 14; forest: 0.458 (depth 8) and still 0.465 at depth 14

SFM_ch13_shrinkage_trees

Interpretation: Averaging and Early Stopping

- The single tree overfits quickly; the forest is almost insensitive to depth
  - averaging removes most of the variance of deep trees; the forest beats HAR on this validation period
- Boosting: the validation error reaches its minimum, then rises while the training error keeps falling
  - learning rate 0.3: minimum 0.457 after 13 trees, 0.568 after 500
  - learning rate 0.03: minimum 0.451 after 154 trees, 0.461 after 500: smaller steps are safer
- The number of trees of boosting is a hyperparameter; the number of trees of a forest is not

The Perceptron and the Multilayer Perceptron (1/2)

- A **neuron**: $h = g(b + w^\top x)$, a weighted sum of the inputs passed through an **activation function** g
  - x: the inputs; w: their weights; b: a constant (the bias term of the neuron); h: the output of the neuron
  - g: a fixed nonlinear function; without it, a network of neurons would be just a linear regression
- Activation functions
  - the perceptron uses a step function (0 or 1)
  - modern networks use smooth or piecewise linear g: logistic $1/(1 + e^{-z})$, tanh, ReLU $\max(0, z)$

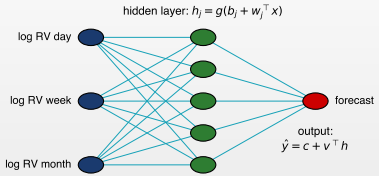
The Perceptron and the Multilayer Perceptron (2/2)

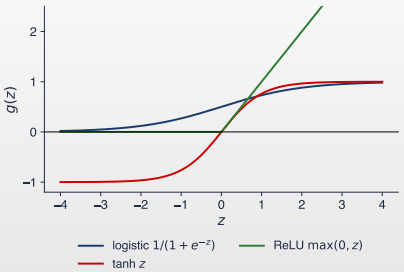
- **MLP** (multilayer perceptron) with one hidden layer of K neurons

$$\hat{y} = c + \sum_{k=1}^{K} v_k \, g(b_k + w_k^\top x)$$

  - w_k, b_k: the weights and the constant of hidden neuron k; v_k: the weight of neuron k in the output; c: the output constant
  - reading: a linear model in K nonlinear features $g(b_k + w_k^\top x)$, which are themselves learned
- Number of parameters with 11 inputs and 16 neurons
  - hidden layer: 11×16 weights $+16$ constants $= 192$; output: $16 + 1 = 17$; total 209
- **Universal approximation** Hornik, Stinchcombe & White (1989): with enough neurons, one hidden layer approximates any continuous function on a compact set

A Small Network and Its Activation Functions





- Left: three HAR inputs, five hidden neurons, one forecast; each line is a weight
- Right: logistic and tanh saturate for large $|z|$; ReLU $= \max(0, z)$ is linear for $z > 0$ and trains faster

SFM_ch13_neural_networks

Training a Network

- Minimise the loss by **gradient descent**: $\theta \leftarrow \theta - \eta \nabla_\theta L$
  - θ: all weights and constants; $\nabla_\theta L$: the gradient, the vector of partial derivatives of the loss; $\eta > 0$: the learning rate (the step size)
  - each step moves θ in the direction in which the loss falls fastest
  - **back-propagation** computes $\nabla_\theta L$ layer by layer with the chain rule Rumelhart, Hinton & Williams (1986)
  - one **epoch** = one pass over the training data; Adam is a gradient method with adaptive step sizes
- The loss is not convex: different starting weights give different networks
  - we fix the random seeds and average five networks (an **ensemble**)
- Against overfitting
  - standardise inputs and target with training statistics; an L_2 penalty on the weights (as ridge); a fixed number of epochs
  - small data, small network: one hidden layer of 16 neurons is enough here

Deep Blue and Deep Learning



IBM Deep Blue, Computer History Museum

Photo: Anton Chiang (2011); CC BY 2.0; Wikimedia Commons



Geoffrey Hinton, Nobel Prize in Physics 2024

Photo: Arthur Petron (2024); CC BY-SA 4.0; Wikimedia Commons

- 1997: Deep Blue beat the world chess champion mainly by search and hand-crafted evaluation rules, not by learning from data
- Deep learning (many layers, huge data) now dominates images and text; on financial prices the data are few and noisy, so small networks are the norm

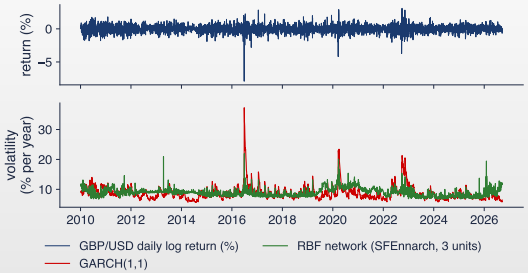
Neural-Network ARCH: the Textbook Model (1/2)

- Franke, Härdle & Hafner (2019), Ch. 19: $r_t = f(x_{t-1}) + s(x_{t-1})\,\varepsilon_t$, with f and s^2 estimated by neural networks
  - r_t: the return of day t; x_{t-1}: the inputs known at the close of day $t-1$
  - $f(x_{t-1})$: the conditional mean; $s^2(x_{t-1})$: the conditional variance; ε_t: a shock with mean 0 and variance 1
- ARCH (Engle (1982)) is the special case $s^2(x) = \omega + \sum_i \alpha_i r_{t-i}^2$
  - $\omega > 0$, $\alpha_i \geq 0$: the variance is a linear function of the past squared returns
  - the network imposes no shape on s^2

Neural-Network ARCH: the Textbook Model (2/2)

- The Quantlet SFEnnarch: GBP/USD, inputs x = the last 3 returns, the German 10-year yield, the gold return
  - step 1: a network $\hat{f}$ for the mean; residuals $e_t = r_t - \hat{f}(x_{t-1})$
  - step 2: a network for e_t^2 gives the conditional variance $\hat{s}^2(x_{t-1})$
- **RBF** (radial basis function) network with 3 units, as in the Quantlet
  - unit j: $\exp(-\|x - c_j\|^2 / s_j^2)$; c_j: its centre (found by clustering); s_j: its width; $\|x - c_j\|$: the distance from x to c_j
  - the unit is close to 1 when x is near c_j and close to 0 far from it; output: a linear combination of the 3 units
  - nothing forces $\hat{s}^2 > 0$: a variance network needs a floor, which GARCH guarantees by its constraints

Neural-Network ARCH for GBP/USD



- Daily data from 7 January 2010 (4 048 days); annualised volatility of the RBF network and of GARCH(1,1) ($\hat{\alpha} = 0.077$, $\hat{\beta} = 0.892$)
- Average volatility: network 8.9%, GARCH 8.8%; correlation of the two paths 0.45 (a small MLP: 0.43)

SFM_ch13_neural_networks

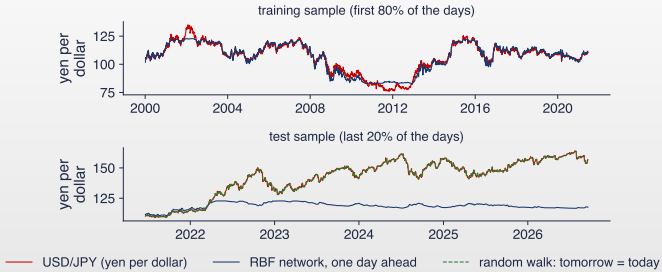
Interpretation: Neural-Network ARCH and GARCH

- After the Brexit vote, GARCH peaks at 37.3% (28 June 2016); the network only at 22.0%
  - the network sees only the last 3 returns: one large return raises its variance for 3 days, then the effect is gone
  - GARCH(1,1) (Chapter 9): $\sigma_t^2 = \omega + \alpha\, r_{t-1}^2 + \beta\, \sigma_{t-1}^2$, with σ_{t-1}^2 yesterday's conditional variance
  - the term $\beta\, \sigma_{t-1}^2$ carries the whole past: volatility clustering needs memory, not only flexibility
- The mean network explains 0.08% of the returns in sample: exchange rates are close to a random walk
  - the useful part of NN-ARCH is the variance, as for GARCH
- Lesson: good features (HAR averages, Section 7) matter more than the type of network

Forecasting an Exchange Rate Level: SFEnnjpyusd

- The Quantlet SFEnnjpyusd forecasts the yen–dollar rate from its last 3 values with an RBF network of 3 units
  - first 80% of the days for training, last 20% for testing; here USD/JPY from 2000, split on 21 June 2021
- Two traps of level forecasting
  - a level series looks predictable because $P_t \approx P_{t-1}$: the right benchmark is the random walk $\hat{P}_t = P_{t-1}$ (Chapter 7)
  - a network with bounded units cannot extrapolate: the test levels go up to 163.9, the training maximum was 134.9
- The original code rescales the test forecasts with the range of the test series, which is unknown in real time; here the training range is used

The RBF Network against the Random Walk



- RMSE (root mean squared error) in yen: training: network 2.64, random walk 0.66; test: network 27.56, random walk 0.89
- Interpretation: out of sample the network is 31 times worse than "tomorrow = today"; on returns the same network has an out-of-sample R^2 of -0.14%

SFM_ch13_neural_networks

Target and Features (1/2)

- Daily variance proxy of Chapter 8 (no intraday data): an estimate of the variance of day t from its open, high, low and close

$$v_t = o_t^2 + \tfrac{1}{2}(u_t - d_t)^2 - (2\ln 2 - 1)c_t^2$$

  - o_t: the overnight log return (from the previous close to the open)
  - u_t, d_t, c_t: log high, low and close relative to the open (the Garman–Klass term, Garman & Klass (1980))

- **Target**: $y_t = \ln\left(\frac{1}{5}\sum_{j=1}^{5} v_{t+j}\right)$, the log realised variance of the next week
  - $j = 1, \ldots, 5$: the next five trading days; the target is known only at the close of day $t + 5$
  - logs make the target closer to Normal and the forecasts positive after exp

Target and Features (2/2)

- **Features** at the close of day t
  - HAR: $\ln v_t$, ln of the mean over 5 days (week) and over 22 days (month)
  - extended (11): plus 4 lags of $\ln v_t$ ($\ln v_{t-1}, \ldots, \ln v_{t-4}$), the 66-day mean, r_t, $\min(r_t, 0)$, $|r_t|$
- Notation
  - r_t: the log return of day t; $|r_t|$: its absolute value
  - $\min(r_t, 0)$: the return if negative, 0 otherwise; it lets falls and rises act differently (the leverage effect)

The HAR Benchmark (1/2)

- **HAR** (heterogeneous autoregressive) model Corsi (2009)

$$y_t = \beta_0 + \beta_d \ln v_t + \beta_w \ln v_t^{(w)} + \beta_m \ln v_t^{(m)} + u_t$$

  - $v_t^{(w)}$, $v_t^{(m)}$: the mean of v over the last 5 days (week) and over the last 22 days (month)
  - β_0: the constant; $\beta_d, \beta_w, \beta_m$: the weights of the three horizons; u_t: the error

- Interpretation
  - three horizons of traders (daily, weekly, monthly); estimated by OLS
  - with positive weights on the three horizons, a shock to volatility fades slowly
  - a simple linear model that mimics long memory (Chapter 11) and is hard to beat

The HAR Benchmark (2/2)

- The question of this section
  - do lasso, random forest, gradient boosting or an MLP add anything to HAR out of sample?
  - a large study with intraday data finds modest gains, mainly from extra features Christensen, Siggaard & Veliyev (2023)
- Design: walk-forward of Section 3, yearly refits; S&P 500 from 2013, Bitcoin from 2018
  - variance forecast $\hat{h}_t = \exp(\hat{y}_t + \hat{s}^2/2)$, $\hat{s}^2$ the residual variance in training (the log-Normal mean)
  - $\hat{y}_t$: the forecast of the log variance; $\exp(\hat{y}_t)$ alone would underestimate the mean variance, hence the term $\hat{s}^2/2$

Evaluation (1/3): Out-of-Sample R^2

- **Out-of-sample** R^2 against a benchmark forecast $\tilde{y}_t$ Campbell & Thompson (2008)

$$R^2_{OOS} = 1 - \frac{\sum_t (y_t - \hat{y}_t)^2}{\sum_t (y_t - \tilde{y}_t)^2}$$

  - $\hat{y}_t$: the forecast of the model; sums over the test days only
  - benchmark $\tilde{y}_t$: the training mean, or HAR

- Reading
  - $R^2_{OOS} \leq 1$; $R^2_{OOS} > 0$: smaller squared error than the benchmark; $R^2_{OOS} < 0$: worse than the benchmark
  - unlike the in-sample R^2, it can be negative

- Step by step, S&P 500, MLP against HAR
  - $1 - 0.397/0.433 = 1 - 0.917$, i.e. $+8.3\%$

Evaluation (2/3): QLIKE

- **QLIKE** (Chapter 8): a loss for variance forecasts, robust to a noisy proxy Patton (2011)

$$L_t = \frac{\mathrm{RV}_t}{\hat{h}_t} + \ln \hat{h}_t$$

  - RV_t: the realised variance of the next 5 days (the target, in levels); $\hat{h}_t$: its forecast
  - for a given RV_t, the loss is smallest at $\hat{h}_t = \mathrm{RV}_t$; lower average QLIKE is better
- Asymmetry
  - for $\mathrm{RV} = 1$: $\hat{h} = 0.5$ costs 1.307, $\hat{h} = 2$ costs 1.193
  - under-prediction is punished more: for risk management, underestimating volatility is the costly error

Evaluation (3/3): the Diebold–Mariano Test

- **Diebold–Mariano** test Diebold & Mariano (1995): do two models A and B have the same expected loss?

$$d_t = L_t^A - L_t^B, \qquad \mathrm{DM} = \frac{\bar{d}}{\sqrt{\widehat{\mathrm{LRV}}/n}} \sim N(0,1) \text{ under equal accuracy}$$

  - L_t^A, L_t^B: the losses of the two models on day t; $\bar{d}$: the mean of d_t; n: the number of test days
  - $\widehat{\mathrm{LRV}}$: the Newey–West long-run variance of d_t (5 lags), because overlapping targets make d_t autocorrelated
- Reading
  - $\mathrm{DM} < 0$: A has the smaller loss; $\mathrm{DM} > 0$: B is better
  - $|\mathrm{DM}| > 1.96$: the difference is significant at 5%

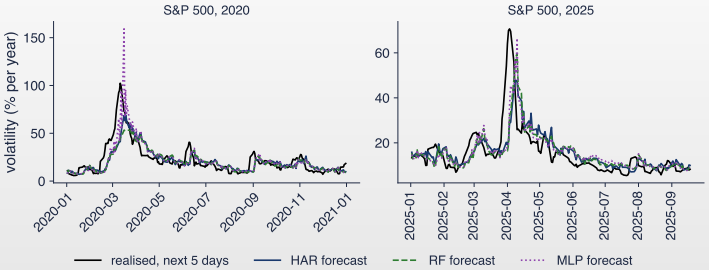
Results: S&P 500 and Bitcoin

	Model	R^2_{OOS} vs. mean	R^2_{OOS} vs. HAR	QLIKE	DM vs. HAR (p)
S&P 500	HAR	61.5	–	0.414	–
	Lasso	64.2	+7.1	0.400	−5.12 (< 0.001)
	RF	61.9	+1.0	0.416	+0.32 (0.747)
	GB	60.4	−2.7	0.421	+1.02 (0.308)
	MLP	64.7	+8.3	0.398	−3.96 (< 0.001)
Bitcoin	HAR	44.0	–	3.111	–
	Lasso	45.1	+2.0	3.108	−0.64 (0.520)
	RF	43.4	−1.2	3.119	+0.81 (0.418)
	GB	41.7	−4.2	3.148	+2.12 (0.034)
	MLP	43.8	−0.3	3.123	+1.10 (0.272)

- Forecasts for 3 444 days (S&P 500, from 2 January 2013) and 3 178 days (Bitcoin, from 1 January 2018); R^2_{OOS} in %, on log RV; GB: gradient boosting
- S&P 500: lasso and MLP beat HAR significantly (+7.1%, +8.3%); random forest and boosting do not
- Bitcoin: no model beats HAR; boosting is significantly worse (DM +2.12)

SFM_ch13_volatility_forecast

Forecasts in the Turbulent Periods of 2020 and 2025



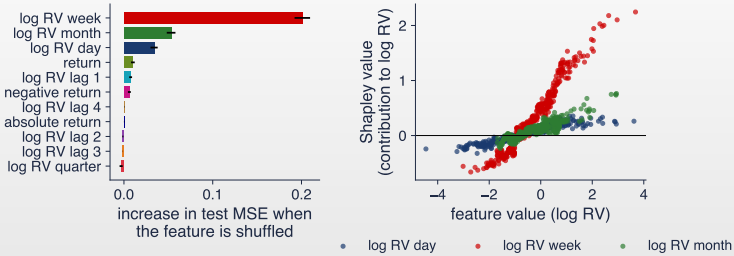
- S&P 500: annualised realised volatility of the next 5 days, $\sqrt{252\,\mathrm{RV}_t}$, and the HAR, random forest and MLP forecasts made at the close of day t
- Interpretation: all models react only after the first shock (March 2020, April 2025); the forest is capped by the largest values it saw in training, the MLP can overshoot

SFM_ch13_volatility_forecast

Feature Importance: Permutation and Shapley Values

- **Permutation importance**: shuffle one feature in the test data and measure how much the test MSE rises
  - large increase: the model relies on the feature; correlated features share (and hide) their importance
- **Shapley values** (game theory): $\hat{f}(x) = \phi_0 + \sum_j \phi_j(x)$, ϕ_j the fair share of feature j in the forecast
  - ϕ_0: the average forecast; $\phi_j(x) > 0$: feature j raises this forecast above the average, $\phi_j(x) < 0$: lowers it
  - ϕ_j averages the change of $\hat{f}$ when j joins every subset of the other features
  - SHAP (Shapley additive explanations) Lundberg & Lee (2017): fast versions for trees; for a linear model $\phi_j = \beta_j(x_j - \bar{x}_j)$
- Both describe the model, not the economy: importance is not causality

The Features Used by the Forest



- Left: random forest on 11 features, trained to 2019, tested 2020–2026 (1 682 days): shuffling the weekly variance raises the MSE by 0.201, the monthly by 0.053, the daily by 0.034
- Right: exact Shapley values of a forest on the 3 HAR features; mean $|\phi|$: week 0.52, month 0.13, day 0.13; the effect is nonlinear and grows with the level of volatility

SFM_ch13_volatility_forecast

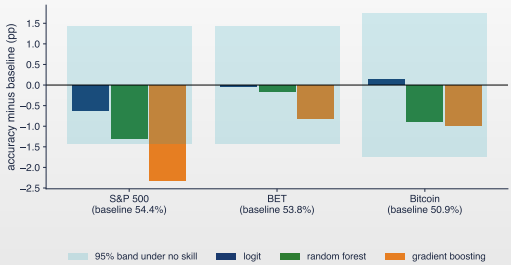
A Classification Problem

- Target: $y_t = 1\{r_{t+1} > 0\}$; features at the close of day t: the last 5 returns, sums over 5, 21 and 63 days, volatility over 21 and 63 days
  - $1\{\cdot\}$: the indicator, 1 if the condition holds and 0 otherwise; $y_t = 1$: tomorrow's return is positive
  - models: logit (Chapter 12), random forest and gradient boosting classifiers; walk-forward, yearly refits
  - S&P 500 and BET from 2008, Bitcoin from 2018
- Metrics (defined in Chapter 12)
  - accuracy: the share of days with the right sign; AUC (area under the ROC curve): 0.5 means no ranking skill
- Why expect little: weak-form efficiency (Chapter 7)
  - any reliable pattern in past returns would be traded away

The Majority-Class Baseline

- **Majority-class baseline**: always forecast the more frequent class of the training sample (here: "up")
  - S&P 500 since 2008: 54.4% of the days are up, so "always up" is right on 54.4% of the days
  - an accuracy of 53% sounds like skill; against this baseline it is a loss
- Test against the baseline: $z = (\widehat{\mathrm{acc}} - \mathrm{acc}_0)/\sqrt{\mathrm{acc}_0(1-\mathrm{acc}_0)/n}$
  - $\widehat{\mathrm{acc}}$: the accuracy of the model; acc_0: that of the baseline; n: the test days; $z \approx N(0,1)$ if the model has no extra skill
  - S&P 500, $n = 4\,705$: standard error 0.73 points; a 95% band of ± 1.4 points around the baseline
  - random forest: 53.1%, $z = -1.79$ (p 0.074); boosting: 52.1%, $z = -3.19$ (p 0.001)
- For returns, the matching baseline is the zero or historical-mean forecast Welch & Goyal (2008)

Accuracy against the Baseline



- Accuracy minus the baseline accuracy (percentage points); band: ± 1.96 standard errors around zero
- AUC of the logit and of the forest: S&P 500 0.511 and 0.507, BET 0.533 and 0.533, Bitcoin 0.515 and 0.519; logit forecasts "up" on 86% of S&P 500 days

SFM_ch13_sign_prediction

Interpretation: Limits of Sign Prediction

- No model beats "always up" significantly on any market; the flexible models do worse than the logit
  - S&P 500: logit -0.6, forest -1.3, boosting -2.3 points; Bitcoin: logit $+0.1$, forest -0.9
- The signal-to-noise ratio of daily returns is tiny
  - flexible models find patterns in the noise of the training years (Section 2); the logit is close to the baseline because it hardly moves
- Even a real edge of one point would rarely survive costs: a daily sign strategy trades very often
  - ML gains in returns come from large cross-sections at monthly horizons, not from one index tomorrow (Section 11)

Quantile Regression (1/2)

- Koenker & Bassett (1978): estimate the conditional α-quantile $q_\alpha(x) = x^\top \beta_\alpha$ by minimising the **quantile (pinball) loss**
  - $q_\alpha(x)$: the level below which y falls with probability α, given x; β_α: coefficients specific to the level α
- The loss of an error $u = y - q$:

$$\rho_\alpha(u) = u\,(\alpha - 1\{u < 0\})$$

  - below the quantile ($u < 0$) the error costs $(1-\alpha)|u|$, above it $\alpha|u|$
  - with $\alpha = 0.01$, the minimum is reached when about 1% of the observations lie below q

Quantile Regression (2/2)

- With $\alpha = 0.01$ and $q = -\mathrm{VaR}$: the average quantile loss of Chapter 10
  - $(\alpha - I_t)(r_t + \mathrm{VaR}_t)$, with $I_t = 1$ if $r_t < -\mathrm{VaR}_t$ (an exception) and 0 otherwise
- VaR 1% for day $t+1$: $\widehat{\mathrm{VaR}}_t = -\hat{q}_{0.01}(x_t)$, with x_t = daily, weekly and monthly volatility (square roots of the proxy) and r_t
  - no distribution is assumed: the data choose how volatility maps into the tail; CAViaR Engle & Manganelli (2004) is a dynamic version
- **Quantile gradient boosting**: the same loss with boosted trees (Section 5)
  - flexible, but at 1% each leaf sees few tail observations: high variance

Backtest of VaR 1%: Quantile Methods, HS and GARCH-t (1/2)

	Method	x	rate (%)	p Kupiec	p ind.	loss $\times 100$	mean VaR
S&P 500	QR	28	0.81	0.252	0.020	3.38	2.70
	QGB	42	1.22	0.213	0.015	3.51	2.46
	HS	46	1.33	0.061	<0.001	4.78	2.97
	GARCH-t	52	1.51	0.005	0.008	3.50	2.42
Bitcoin	QR	23	0.72	0.098	0.563	13.03	10.46
	QGB	47	1.48	0.012	0.235	13.19	8.86
	HS	32	1.01	0.974	0.332	13.01	9.30
	GARCH-t	41	1.29	0.117	0.558	13.15	8.52

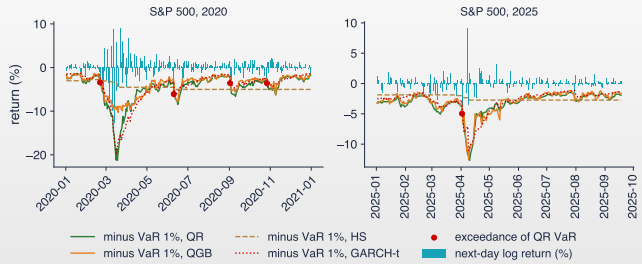
- Columns
  - x: the number of exceptions ($r_{t+\mathbf{1}} < -\widehat{\mathrm{VaR}}_t$) in n test days; rate $= x/n$, target 1%
  - p Kupiec: p-value of the Kupiec test of the rate; p ind.: p-value of the Christoffersen test of independence (Chapter 10)
  - loss $\times 100$: the mean quantile loss (smaller is better); mean VaR: the average VaR 1%, in %
- Methods
  - QR: linear quantile regression; QGB: quantile gradient boosting
  - HS: historical simulation on 500 days; GARCH-t: re-estimated each January (Chapter 10)

SFM_ch13_quantile_var

Backtest of VaR 1%: Quantile Methods, HS and GARCH-t (2/2)

- S&P 500 (3 448 days from 2 January 2013): QR has the right rate (0.81%) and the smallest loss
  - its exceptions are not fully independent (p ind. 0.020)
  - GARCH-t is exceeded too often (1.51%)
- Bitcoin (3 182 days): HS has the best rate (1.01%)
  - QGB is rejected by the Kupiec test (1.48%, p 0.012)
- Reading the p-values
  - p Kupiec below 0.05: the exception rate differs significantly from 1%
  - p ind. below 0.05: the exceptions come in clusters, so the VaR reacts too slowly

VaR 1% Forecasts in 2020 and 2025



- S&P 500: next-day returns and minus VaR 1% of the four methods; dots: exceedances of the quantile-regression VaR
- Interpretation: QR and GARCH-t move with volatility; HS reacts only slowly; in 2020 HS was exceeded 10 times, QR 4 times, and the HS exceptions cluster (p ind. < 0.001)

SFM_ch13_quantile_var

The Best of N Strategies (1/2)

- **Data snooping**: trying many models or rules on the same data and reporting the best White (2000)
  - with enough trials, a rule without skill looks excellent in sample
  - hundreds of published return predictors would not survive a multiple-testing correction Harvey, Liu & Zhu (2016)
- The annual Sharpe ratio of a strategy without skill over Y years is about $N(0, 1/Y)$
  - Sharpe ratio: mean excess return divided by its standard deviation (Chapter 1); Y: the number of years of data
  - mean 0: no skill; variance $1/Y$: the estimation error, which falls with the length of the sample

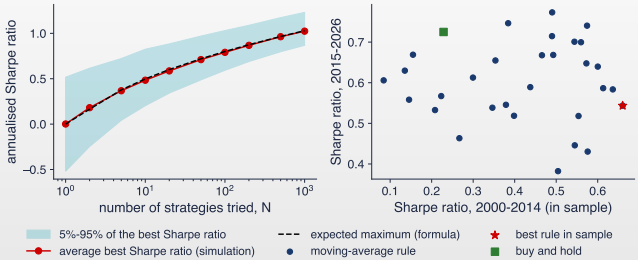
The Best of N Strategies (2/2)

- Expected maximum of N independent zero-skill trials Bailey & López de Prado (2014)

$$\mathrm{E}[\max] \approx \sigma\left[(1-\gamma)\,\Phi^{-1}\left(1-\tfrac{1}{N}\right)+\gamma\,\Phi^{-1}\left(1-\tfrac{1}{Ne}\right)\right]$$

  - σ: the standard deviation of the Sharpe ratios across trials ($1/\sqrt{Y}$ above); N: the number of trials
  - Φ^{-1}: the quantile function of the standard Normal distribution; $e \approx 2.718$; $\gamma = 0.5772$: the Euler–Mascheroni constant
  - $\Phi^{-1}(1-1/N)$ grows roughly like $\sqrt{2\ln N}$: the luck premium rises slowly, but without limit
- Example: 10 years, $\sigma = 1/\sqrt{10} = 0.316$, $N = 100$
  - $0.316\,[(1-\gamma)\,2.326+\gamma\,2.680] = 0.80$: a Sharpe ratio this large is expected from pure luck
  - simulation: the best of 10, 100 and 1000 zero-skill strategies has an average Sharpe ratio of 0.48, 0.79, 1.02

Selection Bias in Practice



- Left: the best annual Sharpe ratio among N strategies without skill (10 years); right: 30 moving-average rules on the S&P 500 (long or cash), 2000–2014 against 2015–2026
- Interpretation: the best rule in sample (MA 50/200, Sharpe 0.66) earns 0.54 afterwards, rank 22 of 30, below buy and hold (0.72); in and out of sample Sharpe ratios correlate 0.04

SFM_ch13_data_snooping

The Deflated Sharpe Ratio (1/2)

- Bailey & López de Prado (2014): the probability that the true Sharpe ratio exceeds what the best of N trials would show by luck

$$\mathrm{DSR} = \Phi\Big(\frac{(\widehat{SR} - SR_0)\sqrt{T-1}}{\sqrt{1 - \hat{\gamma}_3 \widehat{SR} + \frac{\hat{\gamma}_4 - 1}{4}\widehat{SR}^2}}\Big)$$

- Notation
  - $\widehat{SR}$: the per-day Sharpe ratio of the chosen strategy; T: the number of days
  - SR_0: the expected maximum of N zero-skill trials (above), with σ = the standard deviation of the N Sharpe ratios tried
  - $\hat{\gamma}_3$: the skewness, $\hat{\gamma}_4$: the kurtosis of the daily returns (3 for the Normal distribution); Φ: the standard Normal distribution function
- Reading
  - the DSR is a probability in $(0, 1)$; above 0.95: the Sharpe ratio is significant after the correction for N trials
  - negative skewness and fat tails enlarge the denominator and lower the DSR

The Deflated Sharpe Ratio (2/2)

- **Step by step**: the best MA rule, $N = 30$, $T = 3\,769$ days
  - $\widehat{SR} = 0.0416$ per day ($\times\sqrt{252} = 0.66$); $\hat{\gamma}_3 = -0.44$, $\hat{\gamma}_4 = 12.0$
  - $SR_0 = 0.0105\,[(1-\gamma)\,1.834 + \gamma\,2.249] = 0.0219$ (annual 0.35)
  - z, the argument of Φ in the DSR formula: $z = 1.20$, so DSR $= \Phi(z) = 0.88$
  - without the correction for N (PSR, probabilistic Sharpe ratio, $SR_0 = 0$): 0.994
- Interpretation: alone the rule looks significant; after 30 trials it is not (DSR below 0.95), as its later performance confirms

Empirical Asset Pricing via Machine Learning

- Gu, Kelly & Xiu (2020): which methods best predict the monthly return of individual US stocks?
  - nearly 30,000 stocks, 1957–2016; 94 firm characteristics, interacted with 8 macroeconomic variables, plus 74 industry dummies: 920 features
- Design: 18 years of training (1957–1974), 12 of validation (1975–1986), 30 of test (1987–2016)
  - refit once a year; the training window grows, the validation window rolls forward: a walk-forward design
  - models: OLS, penalised and dimension-reduction regressions, random forest, boosted trees
  - networks NN1–NN5: NNk has k hidden layers



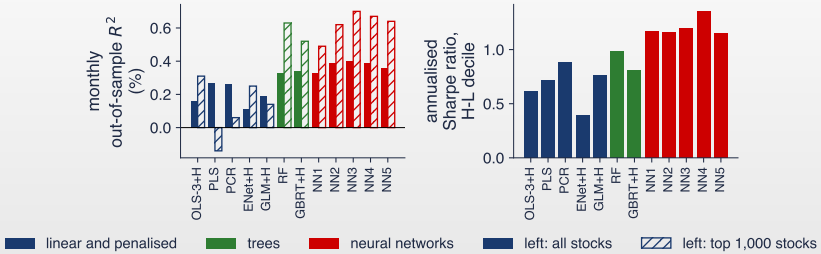
Charles M. Harper Center, Chicago Booth, where Gu and Xiu worked

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

Evaluation in the Paper

- $R^2_{OOS} = 1 - \sum_{(i,t)}(r_{i,t+1} - \hat{r}_{i,t+1})^2 / \sum_{(i,t)} r^2_{i,t+1}$: the benchmark is a forecast of **zero**, not the historical mean
  - $r_{i,t+1}$: the return of stock i in month $t+1$; $\hat{r}_{i,t+1}$: its forecast; the sums run over all stock-months of the test period
  - the authors note that the historical mean of a stock is so noisy that it is easy to beat: a zero benchmark is harder
- Diebold–Mariano tests between all pairs of models, with a Bonferroni correction for the 12 comparisons
  - the same tool as Section 7, applied to a panel of stocks
- Economic value: each month, sort stocks into deciles by forecast; buy the top decile, sell the bottom one
  - value-weighted portfolios; Sharpe ratio, drawdown and turnover (the share of the portfolio traded each month)

Results: Predictive R^2 and Economic Value



- Left: monthly R^2_{OOS} (Table 1): OLS with all 920 features -3.46%; OLS-3 (size, value, momentum) 0.16%; random forest 0.33%; NN3 0.40% (0.70% for the 1,000 largest stocks)
- Right: annual Sharpe ratio of the value-weighted decile spread (Table 7): OLS-3 0.61, random forest 0.98, NN4 1.35

SFM_ch13_case_gkx

Interpretation and Limits

- What the paper shows
  - nonlinear models (trees, networks) beat linear ones because they capture interactions between characteristics
  - all methods agree on the main signals: price trends (momentum, short-term reversal), liquidity, volatility
  - shallow networks are enough: 3–4 layers beat 5; "deep" learning does not help with monthly returns
- What to keep in mind
  - an R^2_{OOS} below 1% per month is a large result here: compare with the sign forecasts of Section 8
  - neural portfolios turn over more than 100% of their value each month: trading costs reduce the gains
  - the evidence is a cross-section of thousands of stocks; it says little about timing one index

An Open Research Question

- **Do machine-learning volatility forecasts help on the Bucharest Stock Exchange, where liquidity is low and intraday data are scarce?**
  - today: on the S&P 500 the lasso gains +7.1% over HAR; on Bitcoin no significant gain (+2.0%, p 0.520)
  - the BET has no high and low prices in our data, so the proxy must come from squared returns, which are noisier
- Why it is open: thin trading, a different calendar from New York, few crises in the sample
- Related work, as further reading: Pele et al. (2023) (statistical classification of crypto assets); Pele & Mazurencu-Marinescu-Pele (2026) (how precise forecast comparisons can be)
- AI tools can speed up such a study; they do not replace checking it Wang et al. (2023)

Possible Uses of AI Tools

- **Literature**: a list of studies on HAR and machine learning for emerging and frontier markets
- **Code**: a first draft of the walk-forward comparison of HAR, lasso and random forest for the BET and three BVB stocks
- **Robustness**: other proxies (absolute returns, two-day returns), other horizons (1, 5, 22 days), S&P 500 features of the previous close
- Example prompt
  - `Write Python code that compares HAR, lasso and a random forest for the weekly log variance of the BET index (proxy: squared daily returns), with yearly walk-forward refits from 2012, purging the last 5 training rows, and reports the out-of-sample R2, QLIKE and Diebold-Mariano tests.`

What to Check

- Time order: no shuffled folds; features use only data up to the close of day t; the last training targets do not reach into the test period
- Scaling and tuning inside the training window only (standardisation, λ, the number of trees)
- Calendars: the BVB closes before New York; S&P 500 features must come from the previous New York close
- A benchmark in every table (HAR, the majority class, historical simulation) and a test of the difference
- The number of models tried, reported; references: every cited paper must exist; check the DOI

Project Idea

- **Question**: do machine-learning models forecast the volatility of BVB stocks better than HAR?
  - data: BET, Banca Transilvania, OMV Petrom, BRD (EODHD); S&P 500 and DAX as external features
- Steps
  - a variance proxy for each stock (range where available, squared returns otherwise), weekly target
  - HAR, lasso, random forest and a small MLP, walk-forward from 2014, yearly refits with purging
  - out-of-sample R^2, QLIKE and Diebold–Mariano; Shapley values for the external features
- Deliverable: one table, one chart, and a paragraph on what the data can and cannot show
- Declare any AI use, and list the errors of the AI that you corrected

Key Takeaways

- Machine learning is judged by its error on new data; the bias–variance trade-off sets how flexible a model should be
- Time series need walk-forward validation with purging; random folds on overlapping targets fake predictability
- Ridge and lasso shrink; forests average trees; boosting adds trees; networks learn nonlinear features
- Volatility: HAR is hard to beat, the gains are a few per cent; the sign of returns: no gain over "always up"
- VaR 1%: linear quantile regression works; flexible quantile models lack tail data
- Report the number of trials and deflate the Sharpe ratio; compare every model with a simple benchmark

Key Formulas

Quantity	Formula
Bias–variance	$E[(y_0 - \hat{f})^2] = (E\hat{f} - f)^2 + \mathrm{Var}(\hat{f}) + \sigma^2$
Ridge	$(X^\top X + \lambda I)^{-1} X^\top y$; 1D: $S_{xy}/(S_{xx} + \lambda)$
Lasso (1D)	$\mathrm{sign}(S_{xy})\max(\lvert S_{xy}\rvert - \lambda/2, 0)/S_{xx}$
Random forest	$\mathrm{Var} = \rho\sigma^2 + (1-\rho)\sigma^2/B$
Boosting	$F_m = F_{m-1} + \nu h_m$
MLP	$\hat{y} = c + \sum_k v_k g(b_k + w_k^\top x)$
HAR	$y_t = \beta_0 + \beta_d \ln v_t + \beta_w \ln v_t^{(w)} + \beta_m \ln v_t^{(m)} + u_t$
R^2_{OOS}	$1 - \sum(y-\hat{y})^2 / \sum(y-\tilde{y})^2$
Quantile loss	$\rho_\alpha(u) = u(\alpha - 1\{u<0\})$
DSR	$\Phi((\widehat{SR} - SR_0)\sqrt{T-1}/\sqrt{1 - \hat{\gamma}_3\widehat{SR} + (\hat{\gamma}_4 - 1)\widehat{SR}^2/4})$

Self-Assessment

- **Question**: a model has a training R^2 of 80% and an out-of-sample R^2 of -5%. What happened?
  - **Answer**: overfitting: the model learned the noise of the training sample and loses against the benchmark on new data
- **Question**: why is random K-fold wrong for a 21-day return target?
  - **Answer**: neighbouring targets overlap, so the test answers leak into training; use walk-forward with purging
- **Question**: a classifier is right on 53% of the days and the market rose on 54% of them. Is it useful?
  - **Answer**: no: "always up" is right on 54% of the days
- Next: Chapter 14, crypto assets

References (1/3)

- Bailey, D.H., López de Prado, M. (2014). The deflated Sharpe ratio: correcting for selection bias, backtest overfitting, and non-normality. *Journal of Portfolio Management*, 40(5), 94–107.
- 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.
- Breiman, L. (2001a). Statistical modeling: the two cultures. *Statistical Science*, 16(3), 199–231.
- Breiman, L. (2001b). Random forests. *Machine Learning*, 45(1), 5–32.
- Campbell, J.Y., Thompson, S.B. (2008). Predicting excess stock returns out of sample: can anything beat the historical average? *Review of Financial Studies*, 21(4), 1509–1531.
- Christensen, K., Siggaard, M., Veliyev, B. (2023). A machine learning approach to volatility forecasting. *Journal of Financial Econometrics*, 21(5), 1680–1727.
- Christoffersen, P.F. (1998). Evaluating interval forecasts. *International Economic Review*, 39(4), 841–862.
- Corsi, F. (2009). A simple approximate long-memory model of realized volatility. *Journal of Financial Econometrics*, 7(2), 174–196.
- Diebold, F.X., Mariano, R.S. (1995). Comparing predictive accuracy. *Journal of Business & Economic Statistics*, 13(3), 253–263.
- Engle, R.F. (1982). Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation. *Econometrica*, 50(4), 987–1007.
- Engle, R.F., Manganelli, S. (2004). CAViaR: conditional autoregressive value at risk by regression quantiles. *Journal of Business & Economic Statistics*, 22(4), 367–381.
- Franke, J., Härdle, W.K., Hafner, C.M. (2019). *Statistics of Financial Markets: An Introduction*, 5th ed. Springer.

References (2/3)

- Friedman, J.H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, 29(5), 1189–1232.
- Garman, M.B., Klass, M.J. (1980). On the estimation of security price volatilities from historical data. *Journal of Business*, 53(1), 67–78.
- Gu, S., Kelly, B., Xiu, D. (2020). Empirical asset pricing via machine learning. *Review of Financial Studies*, 33(5), 2223–2273.
- Harvey, C.R., Liu, Y., Zhu, H. (2016). . . . and the cross-section of expected returns. *Review of Financial Studies*, 29(1), 5–68.
- Hastie, T., Tibshirani, R., Friedman, J. (2009). *The Elements of Statistical Learning*, 2nd ed. Springer.
- Hoerl, A.E., Kennard, R.W. (1970). Ridge regression: biased estimation for nonorthogonal problems. *Technometrics*, 12(1), 55–67.
- Hornik, K., Stinchcombe, M., White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5), 359–366.
- James, G., Witten, D., Hastie, T., Tibshirani, R., Taylor, J. (2023). *An Introduction to Statistical Learning with Applications in Python*. Springer.
- Kelly, B., Xiu, D. (2023). Financial machine learning. *Foundations and Trends in Finance*, 13(3–4), 205–363.
- Koenker, R., Bassett, G. (1978). Regression quantiles. *Econometrica*, 46(1), 33–50.
- Kupiec, P.H. (1995). Techniques for verifying the accuracy of risk measurement models. *Journal of Derivatives*, 3(2), 73–84.

References (3/3)

- Lundberg, S.M., Lee, S.-I. (2017). A unified approach to interpreting model predictions. *Advances in Neural Information Processing Systems*, 30.
- López de Prado, M. (2018). *Advances in Financial Machine Learning*. Wiley.
- Patton, A.J. (2011). Volatility forecast comparison using imperfect volatility proxies. *Journal of Econometrics*, 160(1), 246–256.
- Pele, D.T., Mazurencu-Marinescu-Pele, M. (2026). Finite-sample precision limits for expected shortfall forecast comparisons. *Mathematics*, 14(13), 2316.
- Pele, D.T., Wesselhöfft, N., Härdle, W.K., Kolossiatis, M., Yatracos, Y.G. (2023). Are cryptos becoming alternative assets? *European Journal of Finance*, 29(10), 1064–1105.
- Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6), 386–408.
- Rumelhart, D.E., Hinton, G.E., Williams, R.J. (1986). Learning representations by back-propagating errors. *Nature*, 323, 533–536.
- Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, 58(1), 267–288.
- Wang, H., Fu, T., Du, Y., Gao, W., et al. (2023). Scientific discovery in the age of artificial intelligence. *Nature*, 620, 47–60.
- Welch, I., Goyal, A. (2008). A comprehensive look at the empirical performance of equity premium prediction. *Review of Financial Studies*, 21(4), 1455–1508.
- White, H. (2000). A reality check for data snooping. *Econometrica*, 68(5), 1097–1126.