



Time Series Analysis and Forecasting

Chapter 8: Modern Extensions



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

Learning Objectives

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

1. **Understand** long memory and fractional integration
2. **Distinguish** between short and long memory processes
3. **Estimate** the fractional parameter d using GPH, Local Whittle, and MLE
4. **Apply** Random Forest for time series forecasting
5. **Build** LSTM networks for sequential data
6. **Evaluate** foundation models (Chronos, TimesFM) and the zero-shot paradigm
7. **Compare** classical vs ML vs foundation model performance
8. **Choose** the appropriate method based on data characteristics

Outline

- Motivation
- ARFIMA: Long Memory Models
- Random Forest for Time Series
- LSTM: Deep Learning for Time Series
- Foundation Models for Time Series
- Comparison and Model Selection
- Practical Applications
- Complete Case Study: EUR/RON Exchange Rate
- Final Comparison: All Methods
- Case Study 2: Energy Consumption
- Additional Examples with Real Data
- AI Use Case
- Summary
- Quiz

From Classical Models to Machine Learning

The Evolution of Time Series Methods

- ▣ **Classical ARIMA** (Box & Jenkins, 1970) — revolutionized forecasting but has limitations:
 - ▶ Assumes **short memory**: autocorrelations decay exponentially
 - ▶ **Linear** relationships only — cannot capture complex dynamics
 - ▶ Requires **stationarity** through integer differencing

Four Paradigm Shifts

- ▣ **ARFIMA** (Granger & Joyeux, 1980)
 - ▶ Fractional integration for long memory processes
- ▣ **Random Forest** (Breiman, 2001)
 - ▶ Ensemble learning for nonlinear relationships
- ▣ **LSTM** (Hochreiter & Schmidhuber, 1997)
 - ▶ Deep learning for complex sequential patterns
- ▣ **Foundation Models**: TimeGPT, Chronos — zero-shot forecasting

When to Use Each Method?

Feature	ARIMA	ARFIMA	RF	LSTM
Long memory	×	✓	~ ^a	✓
Nonlinear relationships	×	×	✓	✓
Interpretability	✓	✓	~	×
Small data	✓	✓	×	×
Exogenous variables	✓	✓	✓	✓
Uncertainty quantification	✓	✓	~	×

Principle of Parsimony (Occam's Razor)

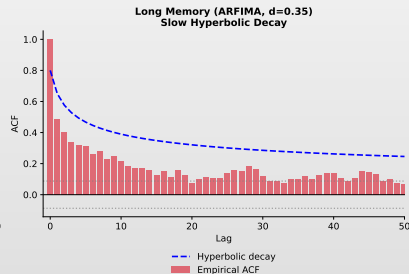
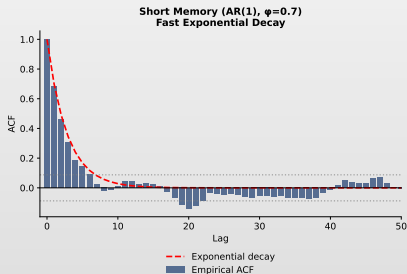
Start **simple** (ARIMA), then increase complexity only if justified by **out-of-sample** performance gains. Makridakis et al. (2018) M4 Competition: simple methods often outperform complex ML models.

^aRF captures persistence through lag features but does not explicitly model long memory structure.

ACF Comparison: Short Memory vs Long Memory

Interpretation

- **Data:** Simulated AR(1) with $\phi = 0.8$ and ARFIMA(0, d ,0) with $d = 0.35$ ($n = 1000$)
- **Left:** AR(1) — autocorrelations decay exponentially (short memory)
- **Right:** ARFIMA with $d = 0.35$ — autocorrelations decay hyperbolically (long memory)



What is Long Memory?

Short Memory (ARMA)

- Autocorrelations ρ_k decay **exponentially**: $|\rho_k| \leq C \cdot r^k$, $r < 1$
- Finite sum: $\sum_{k=0}^{\infty} |\gamma(k)| < \infty$ — shock effects disappear quickly

Long Memory (ARFIMA)

- Autocorrelations decay **hyperbolically**: $\rho_k \sim C \cdot k^{2d-1}$ as $k \rightarrow \infty$
- Infinite sum: $\sum_{k=0}^{\infty} |\rho_k| = \infty$ — shock effects persist for a long time

Examples

Financial volatility, river flows, network traffic, inflation, climate data

The ARFIMA(p,d,q) Model

Definition 1 (ARFIMA — Granger & Joyeux (1980), Hosking (1981))

A process $\{Y_t\}$ follows an **ARFIMA(p,d,q)** model if: $\phi(L)(1-L)^d Y_t = \theta(L)\varepsilon_t$ where $d \in \mathbb{R}$ is the **fractional differencing parameter** (stationarity requires $d < 0.5$).

Fractional Differencing Operator

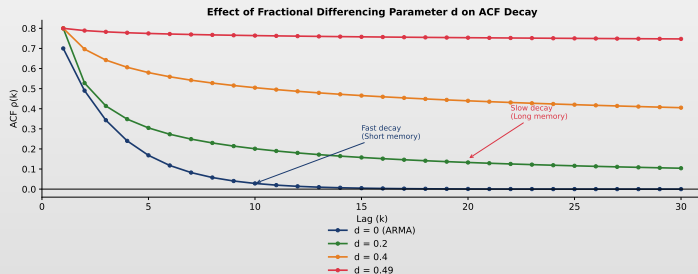
$$(1-L)^d = \sum_{k=0}^{\infty} \binom{d}{k} (-L)^k = 1 - dL - \frac{d(1-d)}{2!} L^2 - \frac{d(1-d)(2-d)}{3!} L^3 - \dots$$

- ▣ $d = 0$: Standard ARMA (short memory)
- ▣ $0 < d < 0.5$: Long memory, stationary
- ▣ $d = 0.5$: Stationarity boundary
- ▣ $0.5 \leq d < 1$: Non-stationary, mean-reverting
- ▣ $d = 1$: Random walk (standard ARIMA)

Effect of Parameter d on ACF

Interpretation

- **Data:** Simulated ARFIMA(0, d ,0) for $d \in \{0.1, 0.2, 0.3, 0.4\}$ ($n = 1000$)
- The higher d , the slower autocorrelations decay
- As $d \rightarrow 0.5$, autocorrelations remain significant even at very large lags



Interpreting the Parameter d

Value of d	ACF Behavior	Interpretation
$d = 0$	Exponential decay	Short memory
$0 < d < 0.5$	Hyperbolic decay	Long memory, stationary
$d = 0.5$	Non-summable ACF	At the boundary
$0.5 < d < 1$	Very slow decay	Long memory, non-stationary
$d = 1$	Very slow sample ACF decay	Random walk

Hurst Parameter H

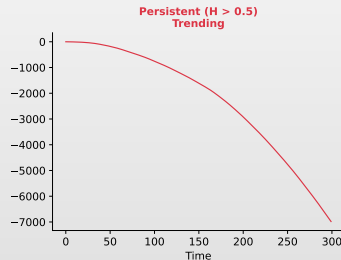
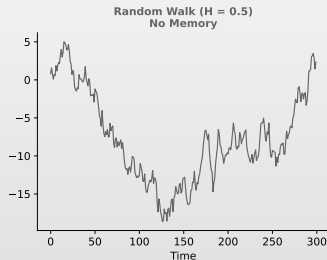
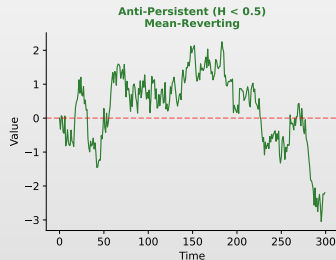
Relationship with Hurst exponent: $d = H - 0.5$

- ▣ $H = 0.5$: No long memory (standard ARMA behavior)
- ▣ $H > 0.5$: Persistence (trend-following)
- ▣ $H < 0.5$: Anti-persistence (mean-reverting)

Hurst Exponent: Visual Interpretation

Interpretation

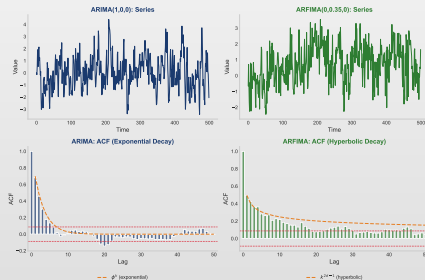
- **Data:** Simulated fractional Brownian motion with $H \in \{0.3, 0.5, 0.7\}$
- $H < 0.5$: Mean-reverting $H = 0.5$: No long memory $H > 0.5$: Persistent



ARIMA vs ARFIMA: Memory Decay Patterns

Interpretation

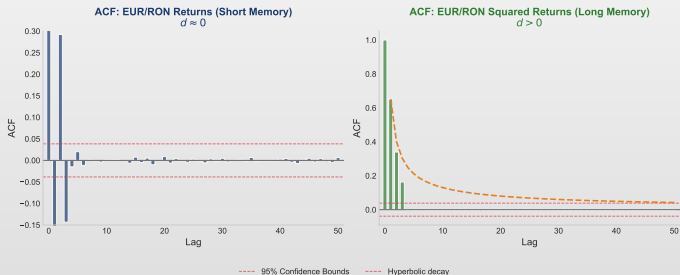
- ▣ **Data:** Simulated ARIMA(1,1,1) vs ARFIMA(1, d ,1) with $d = 0.35$
- ▣ **ARIMA** (left): ACF decays **exponentially** – shocks are quickly “forgotten”
- ▣ **ARFIMA** (right, $d = 0.35$): ACF decays **hyperbolically** – shocks persist for long periods



EUR/RON Long Memory Analysis

Interpretation

- **Data:** EUR/RON daily exchange rate (Yahoo Finance, 2019–2025)
- **Returns:** $H \approx 0.55$, $d \approx 0.05$ – near boundary, essentially short memory
- **Squared returns:** $H \approx 0.69$, $d \approx 0.19$ – long memory in volatility



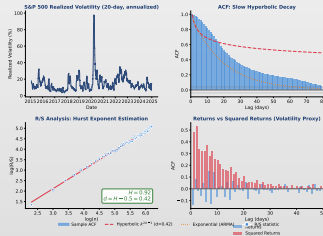
ARFIMA Example: S&P 500 Realized Volatility

Estimation Results

- ▣ **Data:** S&P 500 daily returns (Yahoo Finance, 2015–2024)
- ▣ **Hurst:** $H = 0.92$, $d = H - 0.5 = 0.42$ – strong long memory in realized volatility

Key Insight

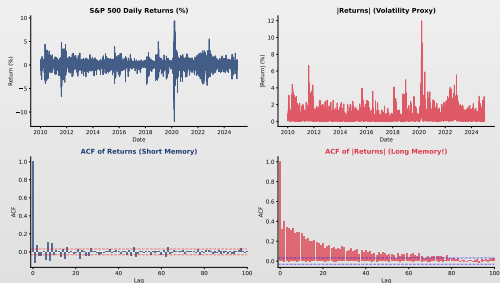
Volatility has **long memory** – shocks persist longer than ARMA; use ARFIMA or FIGARCH.



Real Example: Long Memory in Volatility

Interpretation

- **Data:** S&P 500 daily returns (Yahoo Finance, 2010–2025)
- **Stylized Fact:** Financial returns have short memory, but volatility ($|r_t|$) has long memory
- This is the basis for FIGARCH models



Estimating the Parameter d

Estimation Methods

- ▣ **GPH (Geweke-Porter-Hudak):** Frequency-domain regression
 - ▶ $\ln I(\omega_j) = c - d \cdot \ln\left(4 \sin^2 \frac{\omega_j}{2}\right) + \varepsilon_j$
- ▣ **R/S (Rescaled Range):** Hurst's method
 - ▶ $\frac{R}{S}(n) \sim c \cdot n^H$
- ▣ **MLE (Maximum Likelihood):** Full ARFIMA estimation
- ▣ **Whittle:** Efficient frequency-domain approximation

Implementation

- ▣ In Python: `arch` package, `statsmodels.tsa.arfima.ARFIMA` with `order=(p,d,q)` where d can be fractional

Random Forest: Basic Concepts

What is Random Forest? (Breiman, 2001)

- ▣ **Ensemble** of decision trees
- ▣ Each tree trained on a **bootstrap subset** of the data
- ▣ At each node, a **random** subset of features is selected
- ▣ Final prediction = **average** of all tree predictions

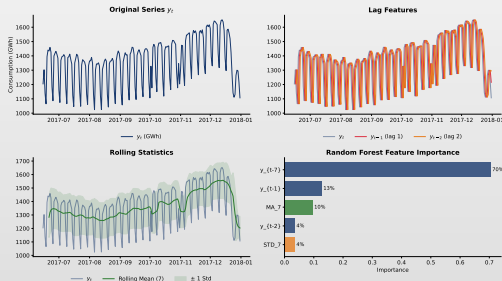
Advantages for Time Series

- ▣ Captures **nonlinear relationships**
- ▣ **Robust** to outliers and noise
- ▣ Does not require **stationarity**
- ▣ Provides **feature importance** (interpretability)
- ▣ Works well with **many variables**

Feature Engineering: Illustration

Interpretation

- ▣ **Data:** Germany daily electricity consumption (OPSD, 2012–2017)
- ▣ We transform the time series into features: lags, rolling statistics
- ▣ The RF model learns relationships between these and future values



Data Preparation for Random Forest

Feature Engineering for Time Series

1. **Lag features:** $Y_{t-1}, Y_{t-2}, \dots, Y_{t-p}$
2. **Rolling statistics:** moving average, standard deviation
3. **Calendar features:** day of week, month, season
4. **Trend features:** time, quadratic trend
5. **Exogenous variables:** economic indicators, events

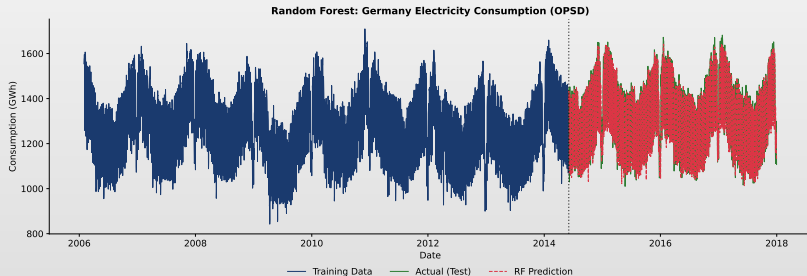
Warning: Data Leakage!

- Do not use future information in features
- Train/test split: **temporal**, not random!
- Rolling statistics: compute only on **past** data

Random Forest: Forecast Example

Interpretation

- ▣ **Data:** Germany daily electricity consumption (OPSD, 2012–2017)
- ▣ RF trained on historical data (blue) produces forecasts (red dashed) that closely track actual values (green)



Feature Importance and Interpretation

Feature Importance

- ▣ **Mean Decrease Impurity (MDI):** Impurity reduction at each split
- ▣ **Permutation Importance:** How much performance drops when a feature is randomly permuted

Typical Interpretation for Time Series

- ▣ `lag_1` very important $\Rightarrow$ Strong autocorrelation
- ▣ `rolling_mean` important $\Rightarrow$ Local trend matters
- ▣ `month` important $\Rightarrow$ Seasonality present

Code

- ▣ `rf.feature_importances_ or permutation_importance(rf, X_test, y_test)`

Researcher Spotlight: Hochreiter & Schmidhuber



Sepp Hochreiter (*1967)



Jürgen Schmidhuber (*1963)



Biography

- ▣ **Sepp Hochreiter**: Austrian computer scientist, Professor at Johannes Kepler University Linz and head of ELLIS Unit Linz
- ▣ **Jürgen Schmidhuber**: German-Swiss computer scientist, Scientific Director of IDSIA
- ▣ Together they solved the vanishing gradient problem

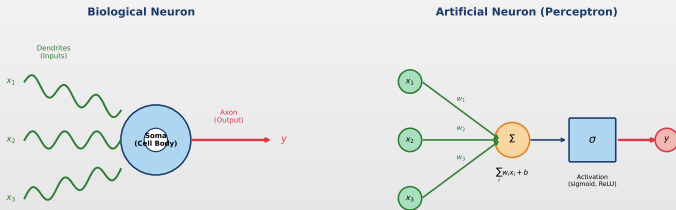
Key Contributions

- ▣ **Long Short-Term Memory (LSTM, 1997)** — gated recurrent architecture solving the vanishing gradient problem
- ▣ **Vanishing gradient analysis** (Hochreiter, 1991) — identified the fundamental training problem in deep networks
- ▣ **Forget gate** extension (Gers et al., 2000) — crucial addition enabling practical LSTM use
- ▣ Foundation for modern sequence modeling in NLP, speech, and time series

From Biological to Artificial Neurons

The Analogy

□ Dendrites $\Rightarrow$ Inputs x_i Synapses $\Rightarrow$ Weights w_i Soma $\Rightarrow$ Sum + Activation Axon $\Rightarrow$ Output y

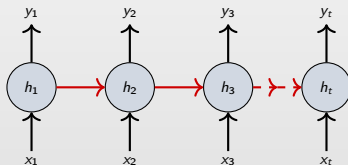


Dendrites $\rightarrow$ Inputs with weights | Soma $\rightarrow$ Weighted sum + activation | Axon $\rightarrow$ Output

Recurrent Neural Networks (RNN)

Basic Idea

- Networks that process **sequences** of data
- Have **internal memory** (hidden state)
- Current state depends on input + previous state

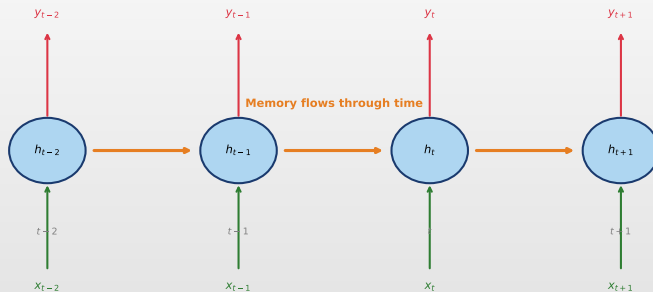


Problem: Vanishing Gradient

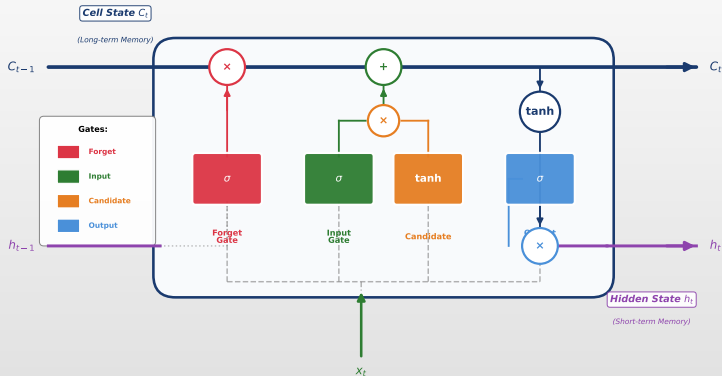
- Simple RNNs “forget” information from the distant past.

RNN Unfolded in Time

Recurrent Neural Network (Unfolded Through Time)



LSTM Cell: Detailed Diagram



Forget Gate f_t
 $\sigma(W_f[h_{t-1}, x_t] + b_f)$
What to forget?

Input Gate i_t
 $\sigma(W_i[h_{t-1}, x_t] + b_i)$
What to store?

Output Gate o_t
 $\sigma(W_o[h_{t-1}, x_t] + b_o)$
What to transmit?

LSTM: Long Short-Term Memory

The LSTM Solution

- ▣ **Concept:** Special cells with 3 gates that control information flow
- ▣ **Forget Gate** (f_t): What to forget from previous memory
- ▣ **Input Gate** (i_t): What new information to add
- ▣ **Output Gate** (o_t): What to send to output

LSTM Equations

▣

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \quad (\text{Forget})$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \quad (\text{Input})$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \quad (\text{Candidate})$$

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \quad (\text{Cell state})$$

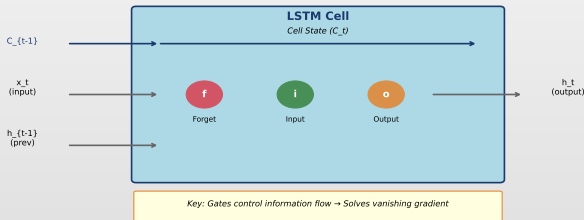
$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \quad (\text{Output})$$

$$h_t = o_t \odot \tanh(C_t) \quad (\text{Hidden state})$$

LSTM Cell Architecture

Interpretation

- The gates (forget, input, output) control what information is discarded, added, and transmitted
- **Cell state** allows gradients to “flow” without degradation



LSTM Advantages for Time Series

Why LSTM?

- ▣ Captures **long-term dependencies** (unlike simple RNN)
- ▣ Learns **complex patterns** and nonlinear relationships
- ▣ Handles **variable-length sequences**
- ▣ Works well with **multivariate data**

Disadvantages

- ▣ Requires **large datasets** for training
- ▣ **Computationally intensive**
- ▣ “**Black box**” – difficult to interpret
- ▣ Sensitive to **hyperparameters**
- ▣ Can **overfit** easily

Foundation Models for Time Series

Paradigm: massive pre-training, zero-shot or fine-tuning

- ▣ Analogous to GPT/BERT: pre-trained on massive corpus, usable directly without retraining

TimeGPT (Garza & Mergenthaler-Canseco, 2024)

- ▣ **100+ billion** time points (finance, energy, retail); Transformer; *zero-shot* via API

Chronos (Ansari et al., 2024, Amazon)

- ▣ Tokenises values into discrete bins; T5; Mini (20M) → Large (710M); *zero-shot*

TimesFM (Google, 2024)

- ▣ **200M params**, 100B real + synthetic points; decoder-only with patching; *zero-shot*

Foundation Models: Advantages, Limits, and Perspectives

Advantages

- ▣ **No domain expertise needed:** the model has already “seen” thousands of series types
- ▣ **Instant deployment:** a single API call, no training pipeline required
- ▣ **Cross-domain transfer learning:** knowledge transferable across domains

Limitations

- ▣ **Black box:** less interpretable — hard to justify in regulated settings
- ▣ **Computational cost** of pre-training (but inference is cheap)
- ▣ **Distribution shift sensitivity:** performance degrades on novel regimes

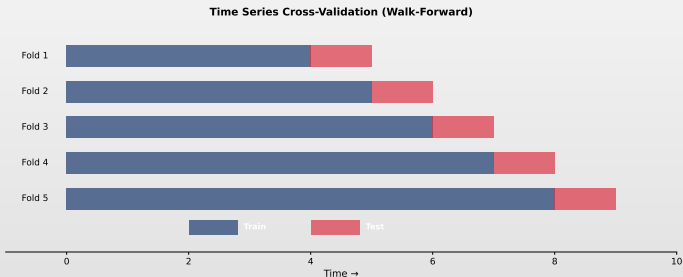
Emerging Models

- ▣ **Lag-Llama** (Rasul+, 2024), **Moirai** (Woo+, 2024), **TimesFM** (Google, 2024)
- ▣ Complementary to traditional econometrics, not a replacement
- ▣ *For detailed architectures and comparisons, see **Chapter 11***

Time Series Cross-Validation

Interpretation

- **Illustration:** Schematic of expanding-window walk-forward validation (5 folds)
- Training set grows progressively; test is always in the future $\Rightarrow$ avoids data leakage



Evaluation Metrics

Notation: y_i = actual value, $\hat{y}_i$ = predicted value, n = number of observations

Common Metrics

▣ Scale-Dependent:

- ▶ RMSE: $\sqrt{\frac{1}{n} \sum (y_i - \hat{y}_i)^2}$ — penalizes large errors
- ▶ MAE: $\frac{1}{n} \sum |y_i - \hat{y}_i|$ — robust to outliers

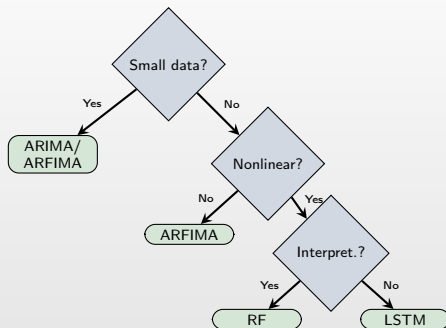
▣ Scale-Free:

- ▶ MAPE: $\frac{100}{n} \sum \left| \frac{y_i - \hat{y}_i}{y_i} \right|$ — percentage error
- ▶ MASE: $\frac{\text{MAE}}{\frac{1}{n-1} \sum_{i=2}^n |y_i - y_{i-1}|}$ — relative to naive (random walk)

Validation for Time Series

- ▣ **Critical:** Do NOT use standard k-fold cross-validation!
 - ▶ Use Time Series CV (walk-forward validation)
 - ▶ Or temporal train/validation/test split

Model Selection Guide



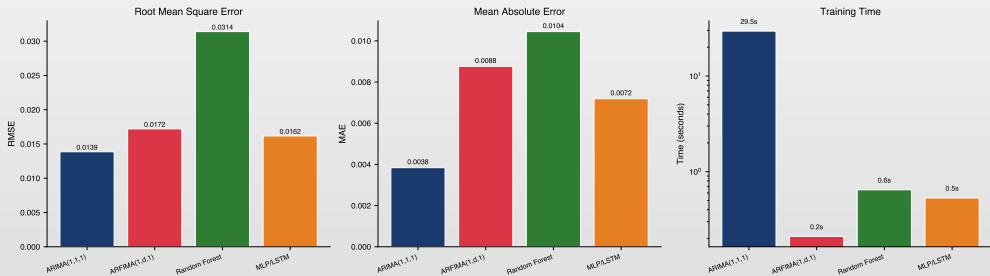
Trade-off

- ▣ ML models offer better accuracy but higher computational cost
- ▣ For small data or interpretability, ARIMA/ARFIMA remain excellent choices

Model Comparison: Accuracy vs Computational Cost

Interpretation

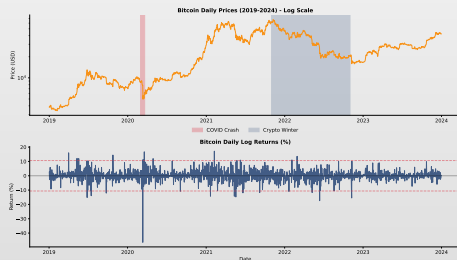
- **Data:** EUR/RON daily exchange rate (Yahoo Finance, 2019–2025)
- **Trade-off:** ML models may achieve better accuracy, but computational cost increases significantly
- For small data or interpretability, ARIMA/ARFIMA remain excellent choices



Bitcoin: Price Evolution and Returns

Key Observations

- Exponential price growth $\Rightarrow$ strongly **leptokurtic** distribution
- Daily returns: mean $\approx 0.15\%$, volatility $\approx 3.5\%$
- Volatility clustering evident $\Rightarrow$ crisis periods (2018, 2020, 2022)
- Kurtosis $\approx 10\text{--}15$ (well above the normal's 3)



Case Study: Bitcoin Price Forecasting

Why Bitcoin?

- ▣ **Extreme** volatility and complex patterns
- ▣ Potential **long memory** in volatility
- ▣ **Nonlinear** relationships with exogenous variables
- ▣ Data available at **high frequency**

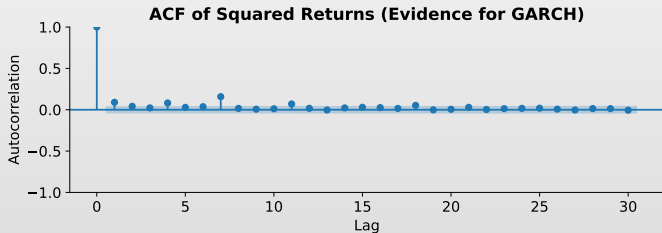
Comparative Approach

1. ARIMA on returns
2. ARFIMA for long memory
3. Random Forest with technical features
4. LSTM on price sequences

Bitcoin: ACF and Evidence for Long Memory

ACF Analysis

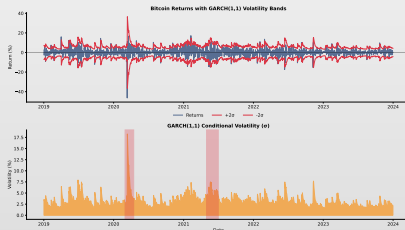
- ACF of returns: rapid decay $\Rightarrow$ short memory in the mean
- ACF of squared returns: **slow, hyperbolic** decay
 - ▶ Indicates **long memory in volatility**
 - ▶ Hurst $H \approx 0.65\text{--}0.70$ ($d \approx 0.15\text{--}0.20$)
- ARFIMA on volatility $>$ ARMA $\Rightarrow$ captures shock persistence



Bitcoin: GARCH and Risk Management

Conclusions – Bitcoin Case Study

- Differences between models are **small** for mean returns
- Major added value: **volatility modeling** (GARCH, EGARCH)
- ARFIMA captures volatility persistence (long memory)
- Random Forest: useful for **nonlinear features** (volume, sentiment)
- Optimal combination: ARFIMA-GARCH + exogenous features via RF

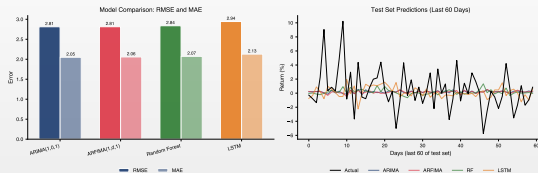


Bitcoin: ARFIMA Estimation and Model Comparison

BTC-USD Returns (2019–2024, test: 2024-02 – 2024-12)

Model	RMSE	MAE	Interpr.?
ARIMA(1,0,1)	2.81	2.05	Yes
ARFIMA(1,d,1)	2.81	2.06	Yes
Random Forest	2.84	2.07	Partial
LSTM	2.94	2.13	No

Hurst $\hat{d} = 0.13$; 70/15/15 temporal split

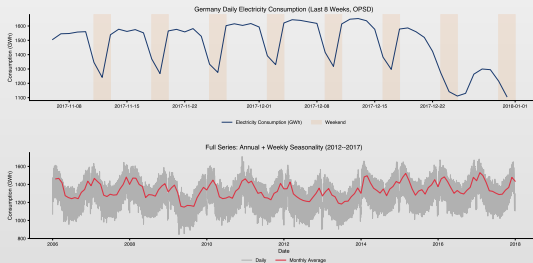


 TSA_ch8_btc_model_comparison

Energy: Demand Visualization and Multiple Seasonality

Identified Patterns

- ▣ **Daily** (24h): peak morning (8–10) and evening (18–21), minimum at night
- ▣ **Weekly** (168h): reduced consumption on weekends (~15–20% less)
- ▣ **Annual** (8766h): peak in summer (AC) and winter (heating)
- ▣ SARIMA cannot simultaneously model these 3 periods!



Case Study: Energy Consumption Forecasting

Characteristics

- ▣ **Multiple seasonality:** daily, weekly, annual
- ▣ **Long-term trend** of growth
- ▣ **Exogenous variables:** temperature, holidays, price
- ▣ **Anomalies:** special events, outages

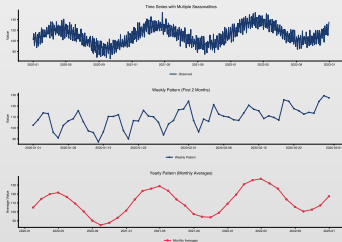
Challenges

- ▣ Patterns at different temporal scales
- ▣ Complex interactions between variables
- ▣ Need for forecasts at different horizons

Energy: Why Prophet and TBATS?

Solution: Models with Multiple Seasonality

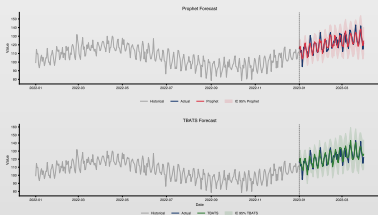
- **TBATS**: periods [24, 168, 8766] $\Rightarrow$ Fourier for each season
 - ▶ Automatic, no manual tuning, good for production
- **Prophet**: additive/multiplicative seasonality + regressors
 - ▶ Add temperature, holidays, special events
- **Classical ARIMA**: can handle only 1 season $\Rightarrow$ MAPE $\approx$ 8–10%



Energy: Prophet Decomposition and Results

Comparison Results on Energy Data (MAPE)

Model	MAPE	RMSE (MW)	95% Coverage
SARIMA (1 season)	8.5%	450	75%
TBATS	4.2%	220	82%
Prophet	4.8%	250	85%
Prophet + regressors	3.9%	200	88%



Energy: Conclusions and Practical Recommendations

Lessons Learned

- ▣ Models with **multiple seasonality** reduce MAPE by $\sim 50\%$ compared to SARIMA
- ▣ **Exogenous variables** (temperature) bring an additional 10–15% gain
- ▣ Prophet excels at **interpretability**: trend + season + holiday decomposition
- ▣ TBATS: best **out-of-the-box** $\Rightarrow$ no hyperparameter tuning needed

When to Choose Each Model?

- ▣ **Prophet**: when you have external regressors + interpretation for management
- ▣ **TBATS**: automation, production, no human intervention
- ▣ **LSTM/RF**: if you have $>100,000$ observations and complex nonlinear patterns

Full details on Prophet and TBATS $\Rightarrow$ Chapter 9

Key Formulas – Summary

ARFIMA(p,d,q)

$$\phi(L)(1-L)^d Y_t = \theta(L)\varepsilon_t \quad d \in \mathbb{R}: \text{long memory (stationary for } d < 0.5)$$

Long Memory

$$\text{ACF: } \rho_k \sim C \cdot k^{2d-1} \quad \text{Hurst: } d = H - 0.5 \quad H > 0.5: \text{persistence}$$

Random Forest

$$\hat{y} = \frac{1}{B} \sum_{b=1}^B T_b(x) \quad B \text{ trees, random features}$$

LSTM Cell

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \quad C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \quad \text{Forget, Input, Output gates}$$

Evaluation Metrics

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum (y_i - \hat{y}_i)^2} \quad \text{MAPE} = \frac{100}{n} \sum \left| \frac{y_i - \hat{y}_i}{y_i} \right|$$

Time Series CV

Walk-forward validation — Train $\Rightarrow$ Test (temporal split)

Case Study: EUR/RON Exchange Rate Forecasting

Why EUR/RON?

- ▣ Relevance for the Romanian economy
- ▣ Potential **long memory** (shock persistence)
- ▣ Patterns influenced by **macroeconomic factors**
- ▣ Easily accessible data (BNR, Yahoo Finance)

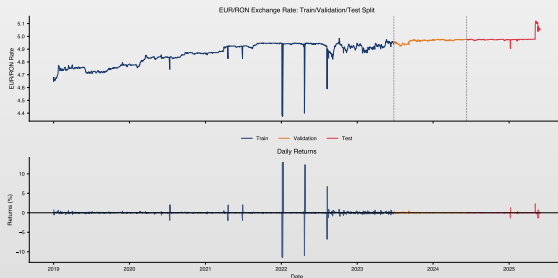
Objective

- ▣ We compare ARIMA, ARFIMA, Random Forest, and LSTM on the same data to understand the strengths of each method

EUR/RON Exchange Rate Visualization

Interpretation

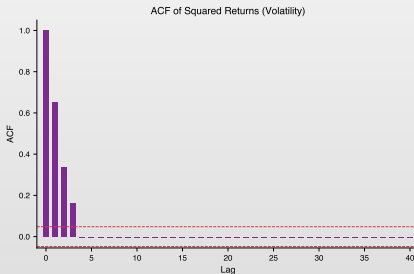
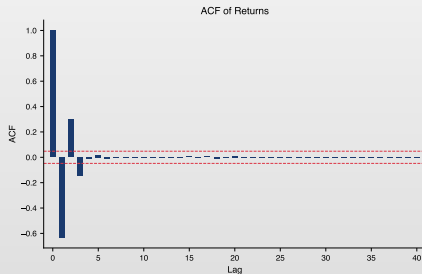
- **Data:** EUR/RON daily exchange rate (Yahoo Finance, 2019–2025)
- **Top:** EUR/RON rate — depreciation trend and periods of high volatility
- **Bottom:** Daily returns — volatility clustering (periods of high volatility are followed by similar periods)



ACF Analysis: Returns vs Squared Returns

Interpretation

- **Data:** EUR/RON daily returns and squared returns (Yahoo Finance, 2019–2025)
- **Left:** ACF of returns — rapid decay, no significant autocorrelation after lag 1
- **Right:** ACF of squared returns — slow decay indicates **volatility clustering** (ARCH effects)



Long Memory Test Results – EUR/RON

Script Output (generate_ch8_eurron_foundation.py)

- ▣ Phillips-Perron p-value: 0.0000 (returns are stationary)
- ▣ Hurst Exponent (H): 0.5476
- ▣ Estimated parameter d: 0.0476
- ▣ Hurst squared returns: $H = 0.6896$, $d = 0.1896$

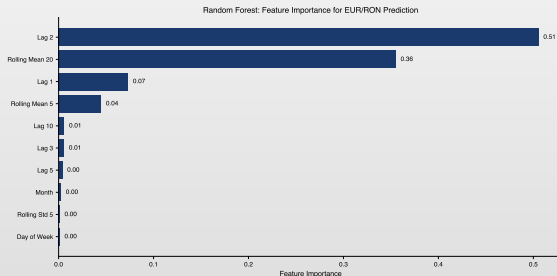
Interpretation

- ▣ EUR/RON returns are **stationary** ($p\text{-value} < 0.05$)
- ▣ $H \approx 0.55$: near the boundary, slight persistence
- ▣ $d \approx 0.05$: essentially **short memory** – ARMA may be sufficient
- ▣ **Squared returns**: $H \approx 0.69$, $d \approx 0.19$ – long memory in volatility!

Random Forest: Feature Importance

Interpretation

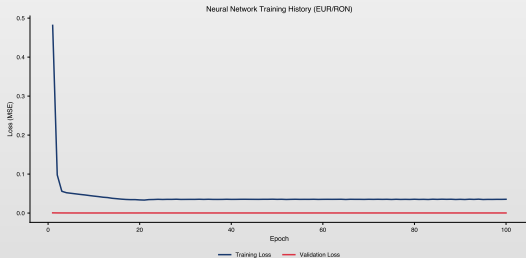
- **Data:** EUR/RON exchange rate (Yahoo Finance, 2019–2025) — RF with 10 engineered features
- Recent lags (lag_1, lag_2) and rolling volatility are the most important features
- Calendar features have minor impact for daily return prediction



LSTM: Learning Curve

Interpretation

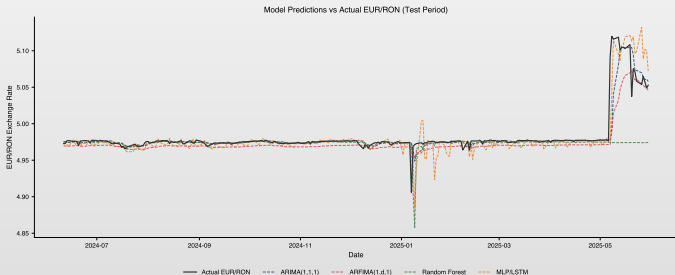
- **Data:** EUR/RON exchange rate (Yahoo Finance, 2019–2025) — Neural Network (100 epochs, MSE loss)
- **Training Loss:** Decreases rapidly in early epochs, then stabilizes
- **Validation Loss:** Tracks training loss — no severe overfitting



Visualization: Predictions vs Actual Values

Interpretation

- **Data:** EUR/RON test period — ARIMA, Random Forest, MLP/LSTM predictions vs actual values
- All models capture the general pattern, but none perfectly predicts volatility spikes
- This reflects **market efficiency** and **prediction limits** for financial series



Comparison: Results on EUR/RON

Model	RMSE	MAE	Time (s)	Interpretable?
ARIMA(1,1,1)	0.0139	0.0038	24.32	Yes
ARFIMA(1, d ,1)	0.0172	0.0088	0.16	Yes
Random Forest	0.0314	0.0104	0.58	Partial
MLP/LSTM	0.0162	0.0072	0.52	No

Rolling 1-step-ahead forecast on price levels; 70/15/15 temporal split; $d = 0.048$ (Hurst).

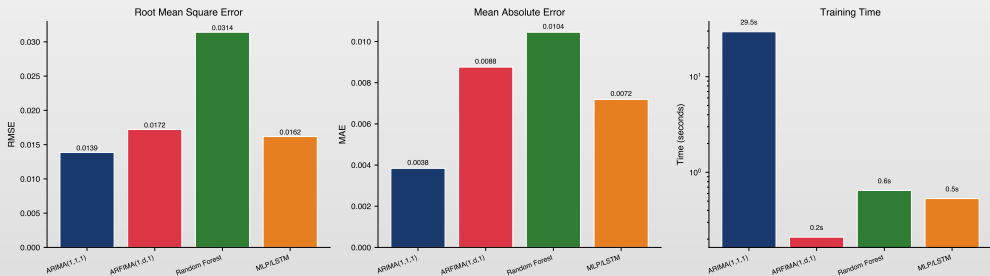
Conclusions

- ▣ ARIMA achieves the **lowest RMSE** — hard to beat on an efficient market
- ▣ ARFIMA and MLP/LSTM are close, but **not better** than ARIMA
- ▣ Random Forest overfits slightly on price levels
- ▣ For EUR/RON, simplicity wins — **parsimony principle** confirmed

Model Comparison: Performance Metrics

Interpretation

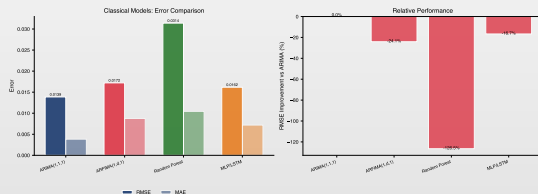
- ▣ **Data:** EUR/RON exchange rate (Yahoo Finance, 2019–2025) — ARIMA vs ARFIMA vs RF vs MLP/LSTM
- ▣ **Left:** RMSE (lower = better) — ARIMA achieves the lowest RMSE
- ▣ **Right:** Training time (log scale) — trade-off between accuracy and computation



Foundation Models: EUR/RON Preview

Quick Results: Zero-Shot on EUR/RON

- ▣ We applied **Chronos** and **TimesFM** zero-shot to EUR/RON (2019–2025)
- ▣ Both models forecast *without any training* — pure transfer learning
- ▣ On this efficient market, classical **ARIMA** remains a strong baseline



For a comprehensive treatment, see Chapter 11

Chapter 11 provides a deep dive into foundation model architectures (Chronos, TimesFM, Lag-Llama, Moirai, TimeGPT), tokenization strategies, zero-shot vs fine-tuning, and extended use cases across multiple domains.

Case Study 2: Energy Consumption — Introduction

Dataset: ENTSO-E Hourly Electricity Consumption

- ▣ **Source:** ENTSO-E Transparency Platform (European electricity data)
- ▣ **Variable:** Total hourly electricity consumption (MW) for a European country
- ▣ **Period:** 2015–2020, approximately 2,160 daily aggregated observations
- ▣ **Frequency:** Daily (aggregated from hourly data)

Forecasting Challenge

- ▣ **Multiple seasonalities:** weekly (5 working days + weekend) and annual (summer/winter)
- ▣ **Nonlinear effects:** temperature, holidays, and economic activity drive consumption
- ▣ **Structural changes:** COVID-19 lockdowns caused unprecedented demand drops in 2020
- ▣ **Goal:** Compare classical (SARIMA, ARFIMA) vs. ML (Random Forest, LSTM) approaches

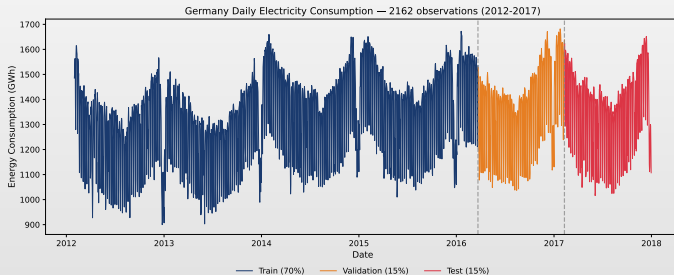
Why This Dataset?

Energy forecasting is a key real-world application: grid operators need accurate short-term forecasts to balance supply and demand and avoid blackouts.

Case Study: Data Overview

Interpretation

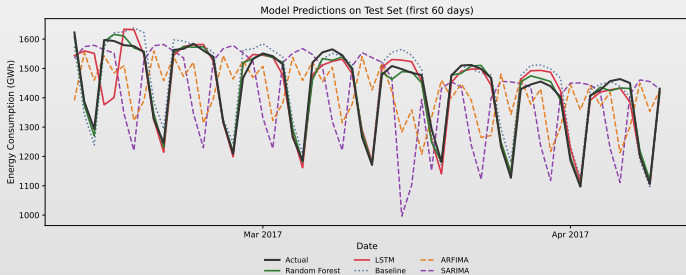
■ **Train:** 1513 obs (70%) **Validation:** 324 obs (15%) **Test:** 325 obs (15%)



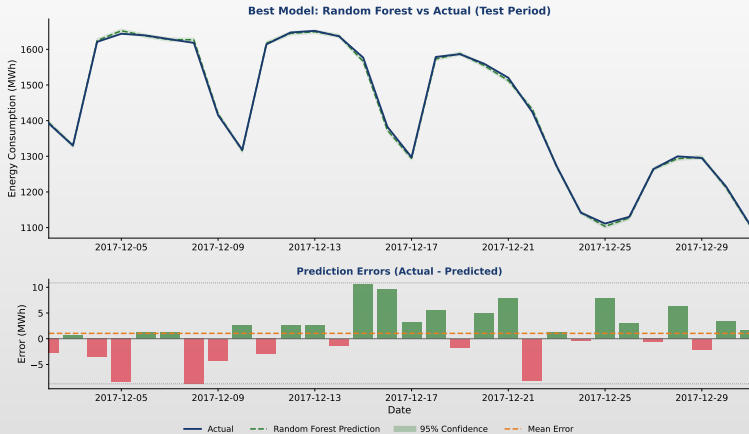
TSA_ch8_data_split

Case Study: Model Predictions

Rank	Model	MAPE	Interpretation
1	Random Forest	2.2%	Best: captures nonlinear patterns
2	LSTM	3.3%	Good, needs more data
3	Baseline	3.9%	Simple but competitive
4	ARFIMA	12.3%	Long memory not sufficient
5	SARIMA	14.6%	Struggles with patterns



Case Study: Best Model Performance



When to Choose Each Model?

ARIMA/ARFIMA

- ▣ Few data (< 500 obs.), interpretation important, suspected long memory, quick baseline

Random Forest

- ▣ Many exogenous variables, nonlinear relationships, feature importance, moderate data

LSTM/Deep Learning

- ▣ Very large data ($> 10,000$), complex sequences, computational resources, hidden patterns

Golden Rule

- ▣ Start simple (ARIMA), add complexity only if performance increases significantly!

Example 2: BET Index (Bucharest Stock Exchange)

Characteristics

- ▣ Strong **volatility clustering**
- ▣ Influenced by international markets
- ▣ Lower liquidity than developed markets
- ▣ Potential for long memory in volatility

Results on BVB.RO Returns (2019–2025, RMSE in %)

- ▣ ARFIMA(1, d ,1): 1.35 – captures long memory ($H = 0.67$, $d = 0.17$)
- ▣ ARIMA(1,0,1): 1.38
- ▣ MLP/LSTM: 1.40
- ▣ Random Forest: 1.44

Example 3: Romania Inflation Rate

Characteristics

- ▣ **Monthly** series (low frequency)
- ▣ **High persistence** – shocks last long
- ▣ Influenced by monetary policy
- ▣ Strong potential for **long memory**

Typical Results

- ▣ ARFIMA with $d \approx 0.35$ – captures persistence
- ▣ ARIMA underestimates shock persistence
- ▣ ML does not work well (few data, 300 obs.)

- ▣ **Lesson:** For monthly series with few data, classical models (ARFIMA) are superior!

Practical Summary: Model Selection

Criterion	ARIMA	ARFIMA	RF	LSTM
Data needed	Few	Few	Medium	Many
Long memory	No	Yes	Partial	Partial
Nonlinearity	No	No	Yes	Yes
Interpretability	Yes	Yes	Partial	No
Computation time	Fast	Fast	Medium	Slow
Exog. variables	Limited	Limited	Yes	Yes

Recommended Workflow

1. Start with **ARIMA** as baseline
2. Test for **long memory** $\Rightarrow$ ARFIMA if d is significant
3. Add **features** $\Rightarrow$ Random Forest
4. Only with lots of data and resources $\Rightarrow$ LSTM

AI Exercise: Critical Thinking

Prompt to test in ChatGPT / Claude / Copilot

"Using yfinance, download daily Bitcoin prices (BTC-USD) from 2019-01-01 to 2024-12-31 (approx. 2,200 observations). Compute daily percentage returns. Compare ARIMA, Random Forest, and LSTM for 7-day-ahead forecasting using a 70/15/15 temporal split. Which model is best? Give me complete Python code with comparison charts."

Exercise:

1. Run the prompt in an LLM of your choice and critically analyze the response.
2. How are features engineered for Random Forest? Lags, calendar variables, Fourier terms?
3. Is the LSTM properly structured? Input shape, scaling, train/test split without leakage?
4. Does it use walk-forward validation or just a single train/test split?
5. Does it mention interpretability and computational cost trade-offs?

Warning: AI-generated code may run without errors and look professional. *That does not mean it is correct.*

Summary

What We Learned

- ▣ **ARFIMA**: Extends ARIMA for long memory (fractional d)
- ▣ **Random Forest**: Ensemble of trees, nonlinear relationships, interpretable
- ▣ **LSTM**: Deep learning for sequences, complex dependencies
- ▣ **Trade-offs**: Complexity vs interpretability vs data requirements

Practical Recommendations

- ▣ Start with **simple** models (ARIMA) as baseline
- ▣ Use **Time Series CV** for proper evaluation
- ▣ ML requires careful **feature engineering**
- ▣ LSTM: only with **lots of data** and computational resources

Question 1

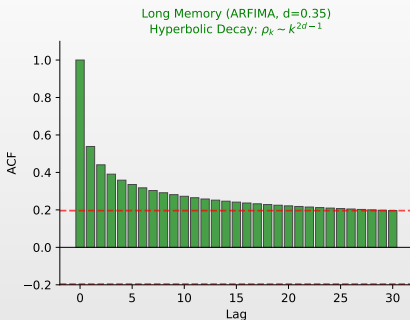
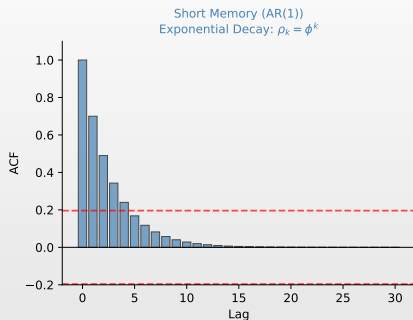
Question

□ What does $d = 0.3$ mean in an ARFIMA model?

Answer Choices

- (A) The series needs 0.3 differences to become stationary
- (B) Long memory: stationary but ACF decays hyperbolically (slowly)
- (C) The series is non-stationary with a unit root
- (D) Short memory: ACF decays exponentially (fast)

Question 1: Answer



Answer: (B)

- For $0 < d < 0.5$: stationary but $ACF \sim k^{2d-1}$ decays much slower than exponential. This “long memory” means distant observations still matter.

Question 2

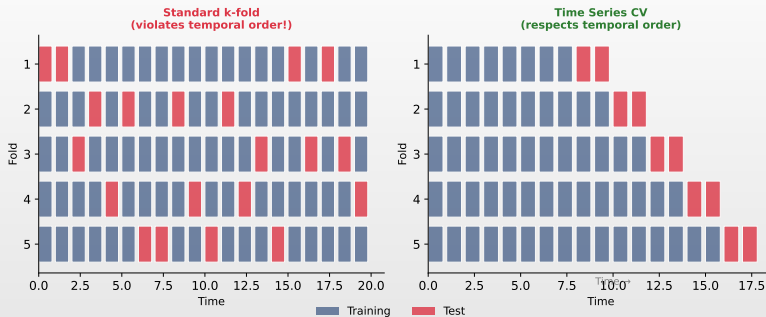
Question

□ Why should you use Time Series Cross-Validation instead of standard k-fold?

Answer Choices

- (A) k-fold is computationally more expensive
- (B) Time Series CV uses more training data
- (C) k-fold violates temporal order, causing data leakage
- (D) There is no difference; both methods are equivalent

Question 2: Answer



Answer: (C)

- Standard k-fold randomly shuffles data, using future observations to predict past ones. Time Series CV always trains on past and tests on future, respecting causality.

Question 3

Question

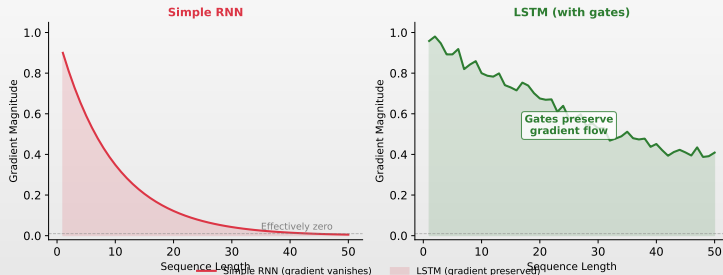
□ What is the main advantage of LSTM over simple RNNs?

Answer Choices

- (A) LSTM uses fewer parameters
- (B) LSTM solves the vanishing gradient problem via gating mechanisms
- (C) LSTM is faster to train
- (D) LSTM does not require sequential data

Question 3: Answer

Vanishing Gradient Problem: RNN vs LSTM



Answer: (B)

- LSTM's forget, input, and output gates control information flow, preserving gradients across long sequences. Simple RNNs lose gradient signal after ~ 10 – 20 steps.

Question 4

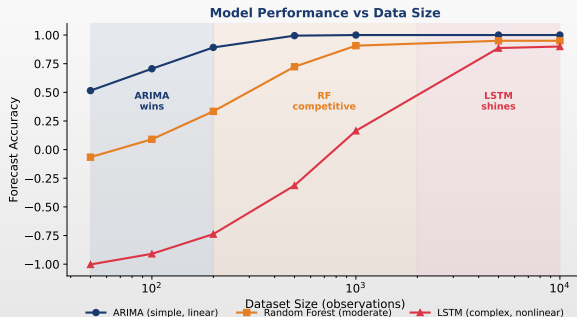
Question

- ☐ You have a small dataset (100 observations) with linear relationships. Which model is most appropriate?

Answer Choices

- (A) LSTM — deep learning captures all patterns
- (B) Random Forest — handles any relationship
- (C) ARIMA/ARFIMA — parsimonious and effective with small data
- (D) Ensemble of all models for best accuracy

Question 4: Answer



Answer: (C)

- ML models (RF, LSTM) need large datasets to generalize. With 100 observations and linear dynamics, ARIMA's few parameters avoid overfitting and often outperform complex models.

Question 5

Question

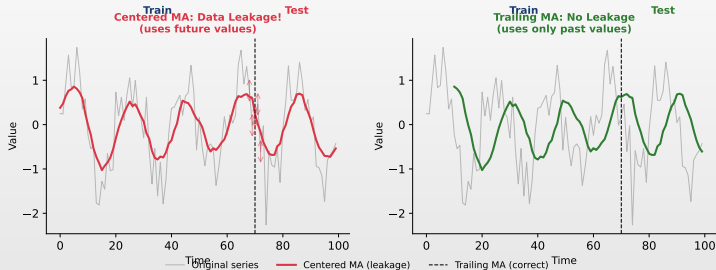
▣ What is “data leakage” in the context of ML for time series?

Answer Choices

- (A) Missing values in the dataset
- (B) Using future information in features or during training
- (C) Having too many features relative to observations
- (D) The model memorizing the training data

Question 5: Answer

Data Leakage in Time Series Feature Engineering



Answer: (B)

- Examples: centered moving averages (use future values), standard k-fold (mixes temporal order), computing statistics over the full dataset before splitting.

What Comes Next?

Extensions and Advanced Topics

- ▣ **Transformer** for time series (Temporal Fusion Transformer)
- ▣ **Prophet** (Meta) for seasonality
- ▣ **Neural Prophet**: Prophet + neural networks
- ▣ **Ensemble methods**: Combining multiple models
- ▣ **Anomaly detection** with ML

Questions?

Bibliography I

Long Memory and ARFIMA

- Granger, C.W.J., & Joyeux, R. (1980). An Introduction to Long-Memory Time Series Models and Fractional Differencing, *Journal of Time Series Analysis*, 1(1), 15–29.
- Baillie, R.T. (1996). Long Memory Processes and Fractional Integration in Econometrics, *Journal of Econometrics*, 73(1), 5–59.
- Beran, J. (1994). *Statistics for Long-Memory Processes*, Chapman & Hall.

Neural Networks and Deep Learning for Time Series

- Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory, *Neural Computation*, 9(8), 1735–1780.

Bibliography II

Threshold, Regime-Switching, and Structural Change Models

- Bai, J., & Perron, P. (2003). Computation and Analysis of Multiple Structural Change Models, *J. Applied Econometrics*, 18(1), 1–22.
- Hansen, B.E. (2011). Threshold Autoregression in Economics, *Statistics and Its Interface*, 4(2), 123–127.
- Hamilton, J.D. (1989). A New Approach to the Economic Analysis of Nonstationary Time Series, *Econometrica*, 57(2), 357–384.
- Petropoulos, F., et al. (2022). Forecasting: Theory and Practice, *Int. J. Forecasting*, 38(3), 845–1054.

Online Resources and Code

- **Quantlet**: <https://quantlet.com> – Code platform for quantitative methods
- **Quantinar**: <https://quantinar.com> – Learning platform for quantitative methods
- **GitHub TSA**: https://github.com/QuantLet/TSA/tree/main/TSA_ch8 – Python code for this chapter

Thank You!

Questions?

Course materials available at: <https://danpele.github.io/Time-Series-Analysis/>



Quantlet



Quantinar