



Analiza și Prognoza Seriilor de Timp

Capitolul 8: Extensii Moderne



Daniel Traian PELE

Academia de Studii Economice din București

IDA Institute Digital Assets

Blockchain Research Center

AI4EFin Artificial Intelligence for Energy Finance

Academia Română, Institutul de Prognoză Economică

MSCA Digital Finance

Obiective de învățare

La finalul acestui capitol, veți fi capabili să:

1. **Înțelegeți** conceptul de memorie lungă în seriile de timp
2. **Estimați** și interpretați modele ARFIMA (AutoRegressive Fractionally Integrated Moving Average)
3. **Aplicați** Random Forest, XGBoost și LSTM pentru prognoza seriilor de timp
4. **Analizați** arhitectura Temporal Fusion Transformer (TFT) și mecanismele sale de interpretabilitate
5. **Evaluați** modelele fundamentale (TimeGPT, Chronos) și paradigma zero-shot
6. **Comparați** ML vs. econometrie clasică în lumina competițiilor M4/M5
7. **Alegeți** metoda potrivită în funcție de context și **implementați** în Python

Cuprins

- Motivație
- ARFIMA: Modele cu Memorie lungă
- Random Forest pentru serii de timp
- LSTM: Învățare profundă (*Deep Learning*) pentru serii de timp
- Comparatie și Selecția modelului
- Aplicații practice
- Studiu de Caz Complet: Cursul EUR/RON
- Comparatie Finală: Toate Metodele
- Studiu de Caz 2: Consum Energie
- Exemple Suplimentare cu Date Reale
- Utilizare IA
- Rezumat
- Quiz
- Bibliografie

De la modele clasice la machine learning

Limitările modelelor ARIMA (AutoRegressive Integrated Moving Average)

- ▣ Presupun **memorie scurtă**: autocorelațiile scad exponențial
- ▣ Relații **liniare** între variabile
- ▣ Dificultăți cu **tipare complexe** și neliniare
- ▣ Necesită **staționaritate** (prin diferențiere)

Soluții moderne

- ▣ **ARFIMA**: Captează memoria lungă (autocorelații care scad lent)
- ▣ **Random Forest / XGBoost**: Relații neliniare, robustețe la outliers
- ▣ **LSTM / TFT**: Tipare secvențiale complexe, atenție interpretabilă
- ▣ **Foundation Models**: TimeGPT, Chronos — prognoză zero-shot

Când să folosim fiecare metodă?

Caracteristică	ARIMA	ARFIMA	RF/XGB	LSTM	TFT	Found.
Memorie lungă	×	✓	✓	✓	✓	✓
Relații neliniare	×	×	✓	✓	✓	✓
Interpretabilitate	✓	✓	~	×	✓	×
Date puține	✓	✓	×	×	×	✓
Variabile exogene	✓	✓	✓	✓	✓	~
Incertitudine	✓	✓	~	×	✓	✓

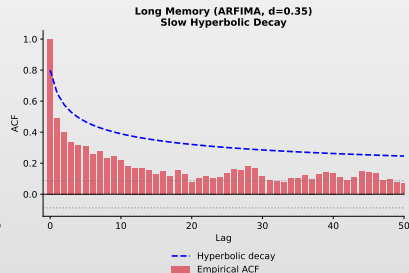
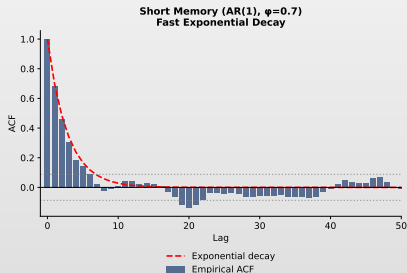
Regula de aur

- ▣ Începeți **simplic** (ARIMA), apoi creșteți complexitatea doar dacă este justificat de date și performanță.

Comparație ACF (Funcția de Autocorelație): memorie scurtă vs lungă

Interpretare

- ▣ **Date:** AR(1) simulat cu $\phi = 0.8$ și ARFIMA(0,d,0) cu $d = 0.35$ ($n = 1000$)
- ▣ **Stânga:** AR(1) — autocorelații care scad exponențial (memorie scurtă)
- ▣ **Dreapta:** ARFIMA cu $d = 0.35$ — autocorelații care scad hiperbolic (memorie lungă)



Ce este memoria lungă?

Memorie scurtă (ARMA)

- Autocorelațiile ρ_k scad **exponențial**: $|\rho_k| \leq C \cdot r^k$, $r < 1$
- Efectele șocurilor dispar **rapid**
- Sumă finită: $\sum_{k=0}^{\infty} |\rho_k| < \infty$ (mai precis, condiția este pe autocovarianțe: $\sum_{k=0}^{\infty} |\gamma(k)| < \infty$)

Memorie lungă (ARFIMA)

- Autocorelațiile scad **hiperbolic**: $\rho_k \sim C \cdot k^{2d-1}$
- Efectele șocurilor persistă **mult timp**
- Sumă infinită: $\sum_{k=0}^{\infty} |\rho_k| = \infty$ (pentru $0 < d < 0.5$, în regimul staționar)

Exemple cu memorie lungă

- Volatilitatea piețelor financiare, debite râuri, trafic rețea, inflație

Modelul ARFIMA(p,d,q)

Definiție 1 (ARFIMA)

- ▣ **Model:** Un proces $\{Y_t\}$ urmează un model **ARFIMA(p,d,q)** dacă: $\phi(L)(1-L)^d Y_t = \theta(L)\varepsilon_t$
- ▣ **Parametru:** $d \in \mathbb{R}$ este parametrul de **diferențiere fracționară** (condiția de staționaritate: $d < 0.5$)

Operatorul de diferențiere fracționară

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

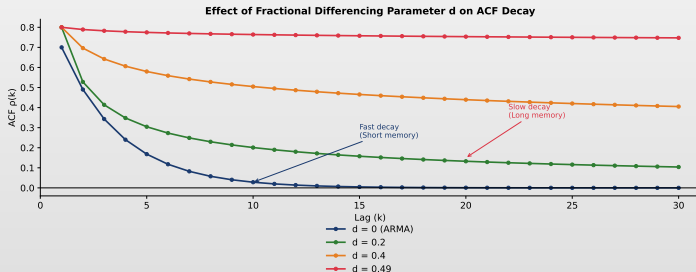
Cazuri particulare

- ▣ $d = 0$: ARMA standard (memorie scurtă)
- ▣ $0 < d < 0.5$: Memorie lungă, staționaritate
- ▣ $d = 0.5$: Limita staționarității
- ▣ $0.5 \leq d < 1$: Nestaționaritate, dar cu revenire la medie
- ▣ $d = 1$: Random walk (ARIMA standard)

Efectul parametrului d asupra ACF

Interpretare

- ▣ **Date:** ARFIMA(0, d ,0) simulat pentru $d \in \{0.1, 0.2, 0.3, 0.4\}$ ($n = 1000$)
- ▣ Cu cât d este mai mare, cu atât autocorelațiile scad mai lent
- ▣ Pentru $d \rightarrow 0.5$, autocorelațiile rămân semnificative chiar și la lag-uri foarte mari



Interpretarea parametrului d

Valoare d	Comportament ACF	Interpretare
$d = 0$	Scădere exponențială	Memorie scurtă
$0 < d < 0.5$	Scădere hiperbolică	Memorie lungă, staționară
$d = 0.5$	ACF nesumabilă	La limită
$0.5 < d < 1$	Scădere foarte lentă	Memorie lungă, nestaționară
$d = 1$	Scădere foarte lentă a ACF eşant.	Random walk

Parametrul Hurst H

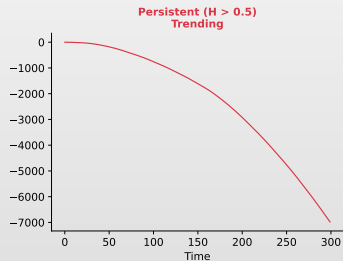
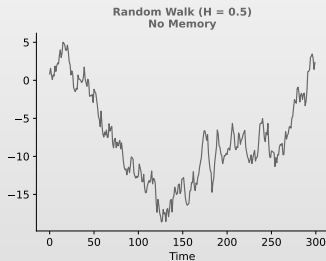
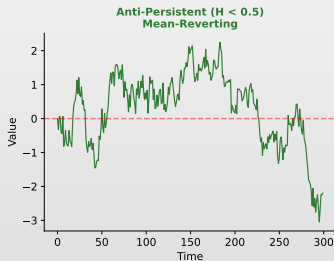
▣ **Relația:** $d = H - 0.5$

- ▶ $H = 0.5$: Fără memorie lungă (comportament ARMA standard)
- ▶ $H > 0.5$: Persistență (continuarea direcției, *trend-following*)
- ▶ $H < 0.5$: Anti-persistență (revenire la medie)

Exponentul Hurst: interpretare vizuală

Interpretare

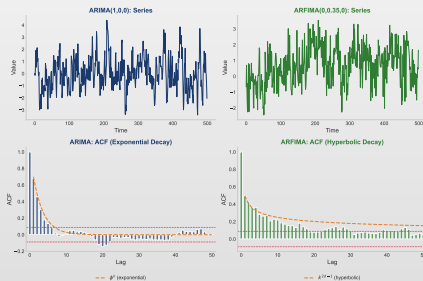
- **Date:** Mișcare Browniană fracționară simulată cu $H \in \{0.3, 0.5, 0.7\}$
- $H < 0.5$: Revenire la medie $H = 0.5$: Fără memorie lungă
- $H > 0.5$: Persistență (continuarea direcției, *trend-following*)



ARIMA vs ARFIMA: comparație simulată

Interpretare

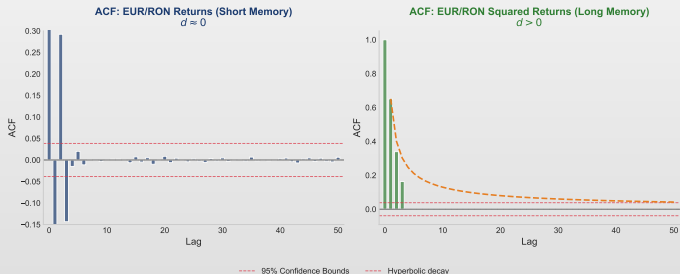
- ▣ **Date:** ARIMA(1,1,1) simulat vs ARFIMA(1, d ,1) cu $d = 0.35$
- ▣ **ARIMA** (stânga): ACF scade **exponențial** — șocurile sunt “uite” rapid
- ▣ **ARFIMA** (dreapta, $d = 0.35$): ACF scade **hiperbolic** — șocurile persistă mult timp



Exemplu date reale: analiza memoriei lungi EUR/RON

Interpretare

- ▣ **Date:** Cursul zilnic EUR/RON (Yahoo Finance, 2015–2025)
- ▣ **Randamente:** $H \approx 0.55$, $d \approx 0.05$ — aproape de limită, esențial memorie scurtă
- ▣ **Randamente pătrate:** $H \approx 0.69$, $d \approx 0.19$ — memorie lungă în volatilitate



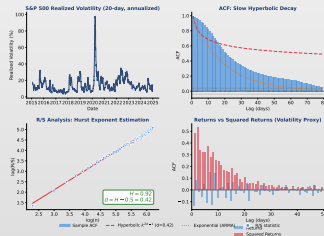
Exemplu ARFIMA: Volatilitatea realizată S&P 500

Rezultate estimare

- **Date:** Randamentele zilnice S&P 500 (Yahoo Finance, 2015–2024); Hurst: $H = 0.92$, $d = 0.42$ – memorie lungă puternică

Observație

- Volatilitatea are **memorie lungă** – folosiți ARFIMA sau FIGARCH (Fractionally Integrated GARCH).



Estimarea parametrului d

Metode de estimare

- ▣ **GPH (Geweke-Porter-Hudak)**: Regresie în domeniul frecvență
 - ▶ $\ln I(\omega_j) = c - d \cdot \ln\left(4 \sin^2 \frac{\omega_j}{2}\right) + \varepsilon_j$
- ▣ **R/S (Rescaled Range)**: Metoda lui Hurst
 - ▶ $\frac{R}{S}(n) \sim c \cdot n^H$
- ▣ **MLE (Maximum Likelihood)**: Estimare completă ARFIMA
- ▣ **Whittle**: Aproximare eficientă în domeniul frecvență

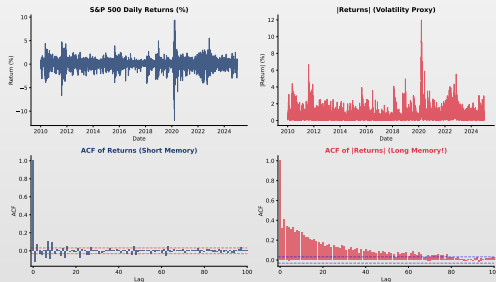
Implementare

- ▣ În Python: pachetul `arch`, sau `statsmodels.tsa.arfima.ARFIMA` (clasa dedicată pentru d fracționar)
- ▣ Se specifică `order=(p,d,q)` unde d poate fi fracționar

Exemplu real: memorie lungă în volatilitate

Notă

- Estimarea ARFIMA necesită pachete specializate
- În practică, se folosește adesea `arch` sau `fracdiff` în Python



Random Forest: concepte de bază

Ce este Random Forest?

- ▣ **Ansamblu** de arbori de decizie
- ▣ Fiecare arbore antrenat pe un **subset bootstrap** al datelor
- ▣ La fiecare nod, se selectează **aleator** un subset de variabile predictoare (*features*)
- ▣ Predicția finală = **media** predicțiilor tuturor arborilor

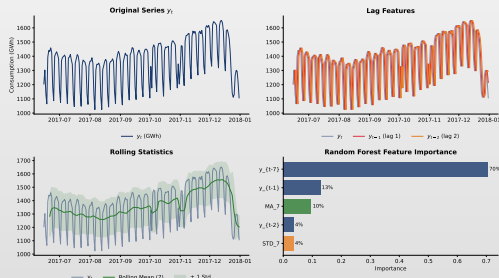
Avantaje pentru serii de timp

- ▣ Captează **relații neliniare**
- ▣ **Robust** la outliers și zgomot
- ▣ Nu necesită **staționaritate**
- ▣ Oferă **importanța variabilelor predictoare** (interpretabilitate)
- ▣ Funcționează bine cu **multe variabile**

Feature engineering: ilustrare

Interpretare

- ▣ **Date:** Consum zilnic de electricitate Germania (OPSD – Open Power System Data, 2012–2017)
- ▣ Transformăm seria temporală în variabile predictoare: lag-uri, statistici pe fereastră mobilă (rolling)
- ▣ Modelul RF învață relațiile dintre acestea și valorile viitoare



Pregătirea datelor pentru Random Forest

Feature Engineering pentru serii de timp

1. **Lag features:** $Y_{t-1}, Y_{t-2}, \dots, Y_{t-p}$
2. **Statistici pe fereastră mobilă** (*rolling statistics*): medie mobilă, deviație standard
3. **Calendar features:** ziua săptămânii, luna, sezon
4. **Trend features:** timp, trend pătratic
5. **Variabile exogene:** indicatori economici, evenimente

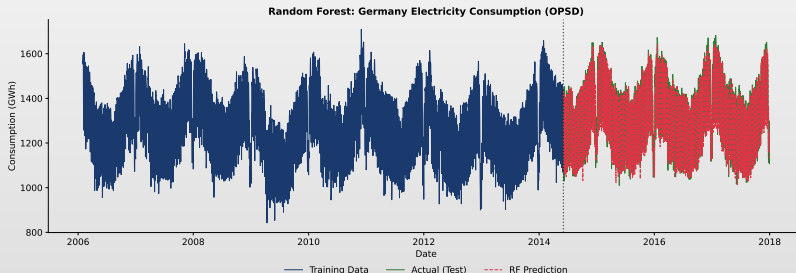
Atenție: Scurgere informațională (*data leakage*)

- ☐ Nu folosiți informații din viitor în features
- ☐ Train/test split: **temporal**, nu aleator!
- ☐ Rolling statistics: calculează doar pe date **anterioare**

Random Forest: exemplu de prognoză

Interpretare

- **Date:** Consum zilnic de electricitate Germania (OPSD, 2012–2017)
- Modelul RF antrenat pe date istorice (albastru) produce prognoze (roșu punctat)
- Prognozele urmăresc bine valorile reale din perioada de test (verde)



Importanța variabilelor predictoare și interpretare

Importanța variabilelor predictoare (*feature importance*)

- ▣ **Scăderea medie a impurității** (*Mean Decrease Impurity*, MDI): Reducerea impurității la fiecare split
- ▣ **Permutation Importance**: Cât scade performanța când feature-ul e permutat aleator

Interpretare tipică pentru serii de timp

- ▣ `lag_1` foarte important $\Rightarrow$ Autocorelație puternică
- ▣ `rolling_mean` important $\Rightarrow$ Trend local contează
- ▣ `month` important $\Rightarrow$ Sezonabilitate prezentă

Cod

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

Gradient Boosting și XGBoost pentru serii de timp

De la Random Forest la Gradient Boosting

- ▣ **Random Forest:** construiește arbori *independenți* și face media (bagging)
- ▣ **Gradient Boosting:** construiește arbori *secvențial*, fiecare corectând erorile precedentului
- ▣ **XGBoost** (Chen & Guestrin, 2016): implementare optimizată cu regularizare L_1/L_2

Avantaje pentru serii de timp

- ▣ Captează **interacțiuni neliniare** între variabile predictoare (lag-uri, calendar, exogene)
- ▣ **Importanța variabilelor:** identificarea automată a lag-urilor relevante
- ▣ **Performanță:** câștigător frecvent în competițiile M5 (Makridakis et al., 2022)
- ▣ **Limitare:** nu modelează dependența temporală nativ $\Rightarrow$ necesită ingineria variabilelor predictoare

Implementare Python

```
from xgboost import XGBRegressor — aceleași variabile predictoare ca pentru Random Forest
```

Portret de cercetător: Hochreiter & Schmidhuber



Sepp Hochreiter (*1967)

 [Wikipedia \(en\)](#)

Jürgen Schmidhuber (*1963)

 [Wikipedia \(en\)](#)

Biografie

- ▣ **Sepp Hochreiter**: informatician austriac, profesor la JKU Linz; conducător al ELLIS Unit Linz
- ▣ **Jürgen Schmidhuber**: informatician germano-elvețian, Director Științific IDSIA
- ▣ Împreună au rezolvat problema gradientului care dispare

Contribuții principale

- ▣ **LSTM** (1997) — arhitectură recurentă cu porți; rezolvă gradientul care dispare
- ▣ **Analiza gradientului** (Hochreiter, 1991) — identificarea problemei fundamentale
- ▣ **Poarta forget** (Gers et al., 2000) — extensie crucială pentru LSTM
- ▣ Fundament pentru modelarea secvențelor în NLP, voce și serii de timp

De la neuronul biologic la cel artificial

Analogia

- **Dendrite** $\Rightarrow$ **Intrări** x_i **Sinapse** $\Rightarrow$ **Ponderi** w_i
- **Soma** $\Rightarrow$ **Sumă + Activare** **Axon** $\Rightarrow$ **Ieșire** y

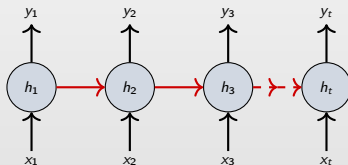


Dendrites $\rightarrow$ Inputs with weights | Soma $\rightarrow$ Weighted sum + activation | Axon $\rightarrow$ Output

Rețele neuronale recurente – RNN (Recurrent Neural Network)

Ideea de bază

- ▣ Rețele care procesează **secvențe** de date
- ▣ Au **memorie internă** (stare ascunsă, *hidden state*)
- ▣ Starea curentă depinde de input + starea anterioară

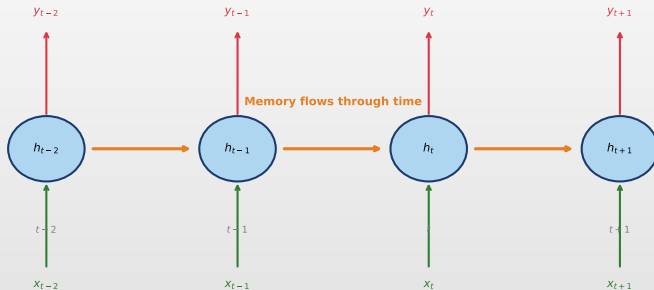


Problema gradientului care dispare (*vanishing gradient*)

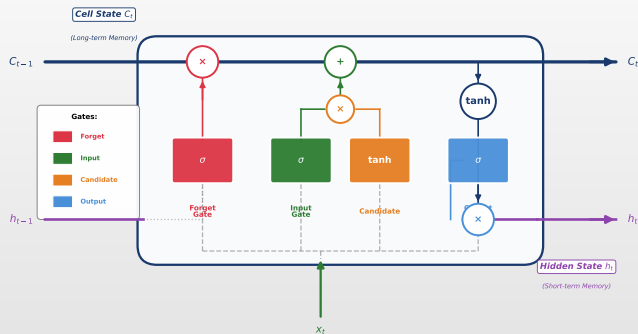
- ▣ RNN simple “uită” informația din trecut îndepărtat.

RNN desfășurată în timp

Recurrent Neural Network (Unfolded Through Time)



Celula LSTM: Diagrama detaliată



□ Poarta de uitare (*Forget Gate*)

 f_t

$$\sigma(W_f[h_{t-1}, x_t] + b_f)$$

□ Ce să uităm?

□ Poarta de intrare (*Input Gate*)

 i_t

$$\sigma(W_i[h_{t-1}, x_t] + b_i)$$

□ Ce să stocăm?

□ Poarta de ieșire (*Output Gate*)

 o_t

$$\sigma(W_o[h_{t-1}, x_t] + b_o)$$

□ Ce să transmitem?

LSTM: long short-term memory

Soluția LSTM

- Celule speciale cu 3 porți: **Forget** (f_t) – ce uităm, **Input** (i_t) – ce adăugăm, **Output** (o_t) – ce transmitem

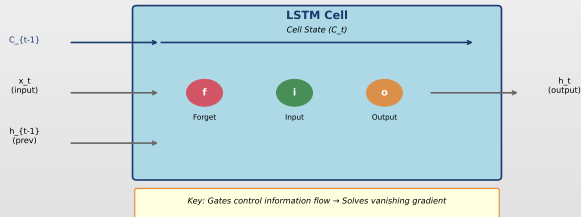
Ecuațiile LSTM

$$\begin{aligned}f_t &= \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) && \text{(Forget)} \\i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) && \text{(Input)} \\\tilde{C}_t &= \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) && \text{(Candidate)} \\C_t &= f_t \odot C_{t-1} + i_t \odot \tilde{C}_t && \text{(Cell state)} \\o_t &= \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) && \text{(Output)} \\h_t &= o_t \odot \tanh(C_t) && \text{(Hidden state)}\end{aligned}$$

Arhitectura celulei LSTM

Interpretare

- Porțile (forget, input, output) controlează ce informație este uitată, adăugată și transmisă
- **Cell state** permite gradientilor să “curgă” fără degradare



Avantajele LSTM pentru serii de timp

De ce LSTM?

- ▣ Captează **dependențe pe termen lung** (spre deosebire de RNN simplu)
- ▣ Învăță **tipare complexe** și neliniare
- ▣ Gestionează **secvențe de lungimi variabile**
- ▣ Funcționează bine cu **date multivariate**

Dezavantaje

- ▣ Necesită **multe date** pentru antrenare
- ▣ **Computațional intensiv**
- ▣ model de tip *cutie neagră* – interpretabilitate redusă
- ▣ Sensibil la **hiperparametri**
- ▣ Poate face **supraajustare** (*overfitting*) ușor

Temporal Fusion Transformer (TFT)

Q TSA_ch8_tft

Arhitectura TFT (Lim, Arık, Loeff & Pfister, 2021)

- ▣ **Variable Selection Networks:** importanța automată a variabilelor via GRN — ponderi învățate, eliminând features irelevante
- ▣ **Static covariate encoders:** variabile invariante în timp (ID magazin, categorie) condiționează procesarea temporală
- ▣ **Procesare temporală:** encoder–decoder LSTM pentru tipare locale și secvențiale
- ▣ **Atenție multi-head interpretabilă:** identifică explicit care pași temporali contribuie la prognoză
- ▣ **Regresie cuantică:** predicția simultană a cuantilelor ($\tau = 0.1, 0.5, 0.9$), intervale de incertitudine native

Inovația cheie: interpretabilitate încorporată în arhitectură

- ▣ Ponderile de atenție $\Rightarrow$ care momente din trecut contează cel mai mult
- ▣ Scoruri de selecție a variabilelor $\Rightarrow$ care features sunt relevante
- ▣ Performanță *state-of-the-art* pe benchmark-uri: electricitate, trafic, retail

TFT — Interpretabilitate și Comparație

Q TSA_ch8_tft_architecture

Mecanisme de interpretabilitate TFT

- **Scoruri de importanță:** învățate automat prin Variable Selection Networks
- **Tipare de atenție temporală:** heatmap-uri — ce pași din trecut influențează prognoza

Comparație cu alte arhitecturi deep learning

Model	Interpretabil	Covariate	Multi-orizont	Cuantile
N-BEATS (Oreshkin+, 2020)	Parțial	Nu	Da	Nu
DeepAR (Salinas+, 2020)	Nu	Da	Da	Da
TFT (Lim+, 2021)	Da	Da (3 tipuri)	Da	Da

Perspectivă critică

Modelele simple (ETS, ARIMA) rămân competitive cu deep learning pe multe tipuri de date (Zeng et al., 2023).

Ref: Lim, Arık, Loeff & Pfister (2021), *Int. J. Forecasting*, 37(4), 1748–1764.

Modele fundamentale pentru serii de timp

Paradigmă: pre-antrenare masivă, zero-shot sau fine-tuning

- ▣ Analog cu GPT/BERT: model pre-antrenat pe corpus masiv, utilizabil direct fără re-antrenare

TimeGPT (Garza & Mergenthaler-Canseco, 2024)

- ▣ Antrenat pe **100+ miliarde** puncte temporale (finanțe, energie, retail, meteo)
- ▣ Arhitectură Transformer, prognoză *zero-shot* via API (Nixtla)

Chronos (Ansari et al., 2024, Amazon)

- ▣ Tokenizarea valorilor în intervale discrete; arhitectură T5 pre-antrenată
- ▣ Dimensiuni: Mini (20M) → Large (710M parametri); prognoză *zero-shot*

TimesFM (Google, 2024)

- ▣ **200M parametri**, pre-antrenat pe 100B puncte temporale reale + sintetice
- ▣ Arhitectură decoder-only cu input/output patching; prognoză *zero-shot*

Foundation Models: Avantaje, Limite și Perspective

Avantaje

- ▣ **Fără expertiză de domeniu:** modelul a “văzut” deja mii de tipuri de serii
- ▣ **Implementare instantanee:** un singur apel API, fără pipeline de antrenare
- ▣ **Transfer learning cross-domain:** cunoștințe transferabile între domenii

Limitări

- ▣ **Cutie neagră:** mai puțin interpretabile — greu de justificat în reglementare
- ▣ **Cost computațional** al pre-antrenării (dar inferența e ieftină)
- ▣ **Sensibilitate la schimbări de distribuție:** performanța degradează pe regimuri noi

Modele emergente

- ▣ **Lag-Llama** (Rasul+, 2024), **Moirai** (Woo+, 2024), **TimesFM** (Google, 2024)
- ▣ Complementare econometriei tradiționale, nu înlocuitoare
- ▣ *Pentru arhitecturi detaliate și comparații, vezi **Capitolul 11***

Competiția M5: ML vs. Econometrie

TSA_ch8_m5_competition

Competiția M5 (Makridakis, Spiliotis & Assimakopoulos, 2022)

- ▣ **42.840 serii de timp** (date vânzări Walmart, 30.490 participanți)
- ▣ **Câștigător: LightGBM** cu inginerie intensivă a variabilelor predictoare
- ▣ ML a **dominat** primele poziții, dar ETS/ARIMA rămân **competitive per serie individuală**

Context: evoluția competițiilor M

- ▣ **M4** (2018): câștigător — hibridul **ES-RNN** (Smyl, 2020); ML *pur* a performat slab
- ▣ **M3** → **M4** → **M5**: avantajul ML crește progresiv
- ▣ Factor decisiv M5: **cross-learning** — un model pe mii de serii simultan

Lecție

Abordările hibride (statistică + ML) au depășit atât metodele clasice *pure*, cât și ML *pur*.

ML vs. Econometrie: Sinteză

TSA_ch8_timegpt

Comparație sistematică

Criteriu	Econometrie clasică	Machine Learning
Interpretabilitate	Ridică	Variabilă (scăzută pt. DL)
Date necesare	Puține (30–100 obs.)	Multe (1.000+)
Incertitudine	Intervale analitice	Bootstrap / conformal
Cauzalitate	Da (Granger, SVAR)	Nu (doar predicție)
Cross-learning	Nu	Da (avantajul din M5)
Schimbări de regim	STAR, Markov	Automat (atenție)

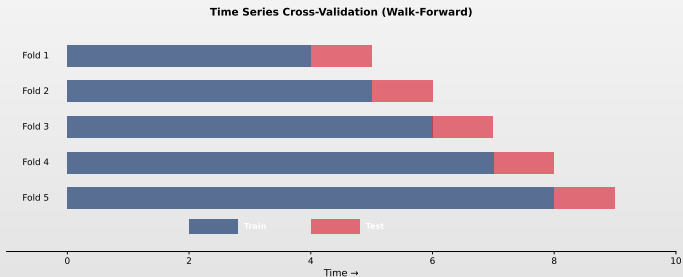
Recomandare: abordări hibride

- ▣ **Ensemble statistic + ML:** media ponderată ARIMA + LightGBM reduce varianța prognozei
- ▣ **Features statistice pentru ML:** reziduurile ARIMA, exponentul Hurst, parametrul d ca variabile predictoare suplimentare pentru gradient boosting
- ▣ **Reconciliere ierarhică:** modele statistice la nivel agregat + ML la nivel granular

Ref: Makridakis, Spiliotis & Assimakopoulos (2022), *International Journal of Forecasting*, 38(4), 1325–1346.

Time series cross-validation

□ **Important:** Setul de antrenare crește progresiv, testul este întotdeauna în viitor



Metrici de evaluare

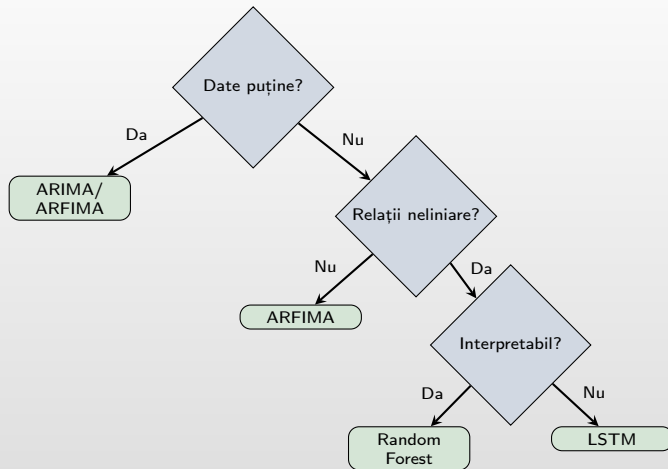
Metrici comune

- ▣ (Metricile *RMSE*, *MAE*, *MAPE* — definite în Capitolul 0)
- ▣ **MASE** (Mean Absolute Scaled Error): Comparat cu benchmark naiv — metric suplimentar pentru comparații între serii

Validare pentru serii de timp

- ▣ **Nu** folosiți cross-validation standard.
- ▣ Folosiți **Time series cross-validation** (walk-forward)
- ▣ Sau **train/validation/test** split temporal

Ghid de selecție a modelului



Studiu de caz: Bitcoin — volatilitate și memorie lungă

Acest studiu de caz este prezentat în detaliu în Capitolul 10

Analiza completă a Bitcoin (teste de staționaritate, modelare GARCH, exponent Hurst, comparație modele) este disponibilă în **Capitolul 10: Recapitulare Comprehensivă**.

Relevanța pentru Capitolul 8

- Bitcoin ilustrează **memorie lungă** în volatilitate ($H \approx 0.65\text{--}0.70$)
- ARFIMA captează persistența șocurilor mai bine decât ARMA
- Combinație optimă: ARFIMA-GARCH + features exogene via Random Forest

Studiu de caz: Consum de energie — sezonalități multiple

Acest studiu de caz este prezentat în detaliu în Capitolul 9

Analiza completă a consumului de energie (sezonalități multiple, TBATS, Prophet, comparație modele) este disponibilă în **Capitolul 9: Prophet și TBATS**.

Relevanța pentru Capitolul 8

- Sezonalitate multiplă (zilnică, săptămânală, anuală) — SARIMA nu poate modela simultan
- Prophet + regresori reduc MAPE cu ~50% față de SARIMA
- LSTM/RF: utile dacă ai >100.000 observații și tipare neliniare complexe

Formule principale – Rezumat

ARFIMA(p,d,q) & Memorie lungă

- ▣ $\phi(L)(1-L)^d Y_t = \theta(L)\varepsilon_t$, $d \in \mathbb{R}$ ($d < 0.5$ pentru staționaritate)
- ▣ ACF: $\rho_k \sim C \cdot k^{2d-1}$ Hurst: $d = H - 0.5$ $H > 0.5$: persistență

Random Forest & LSTM

- ▣ RF: $\hat{y} = \frac{1}{B} \sum_{b=1}^B T_b(x)$ B arbori, features aleatorii
- ▣ LSTM: $f_t = \sigma(W_f[h_{t-1}, x_t] + b_f)$, $C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t$ (Forget, Input, Output gates)

Metrice evaluare & Time Series CV

- ▣ RMSE, MAE, MAPE (vezi Capitolul 0 pentru definițiile complete)
- ▣ Walk-forward validation — Train $\Rightarrow$ Test (temporal split)

Studiu de caz: prognoza cursului EUR/RON

De ce EUR/RON?

- ▣ Relevanță pentru economia românească
- ▣ Potențială **memorie lungă** (persistența șocurilor)
- ▣ Tipare influențate de **factori macroeconomici**
- ▣ Date ușor accesibile (BNR – Banca Națională a României, Yahoo Finance)

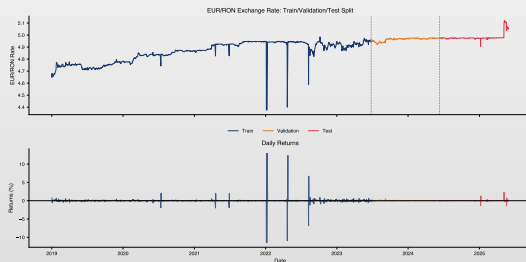
Obiectiv

- ▣ Comparăm ARIMA, ARFIMA, Random Forest și LSTM pe aceleași date
- ▣ Scopul: înțelegerea punctelor forte ale fiecărei metode

Vizualizarea seriei EUR/RON

Interpretare

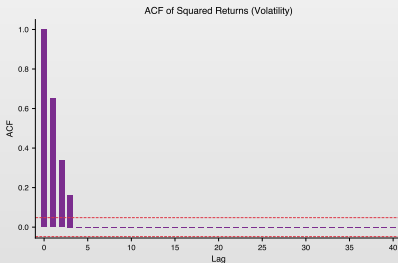
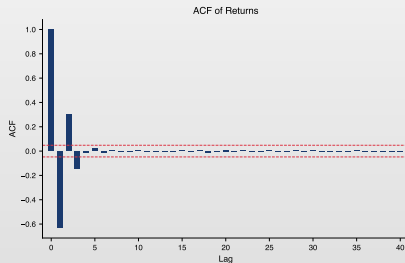
- ▣ **Date:** Cursul zilnic EUR/RON (Yahoo Finance, 2019–2025)
- ▣ **Sus:** Cursul EUR/RON — tendința de depreciere a leului și perioade de volatilitate ridicată
- ▣ **Jos:** Randamentele zilnice — volatility clustering
- ▣ Perioadele de volatilitate mare sunt urmate de alte perioade similare



Analiză ACF: randamente vs randamente pătrate

Interpretare

- ▣ **Date:** Randamentele zilnice EUR/RON (Yahoo Finance, 2019–2025)
- ▣ **Stânga:** ACF randamente — scădere rapidă; **Dreapta:** ACF randamente pătrate — scădere lentă
- ▣ Indică **volatility clustering** (efecte ARCH)



Rezultate test memorie lungă – EUR/RON

Output script (generate_ch8_eurron_foundation.py)

- ▣ Phillips-Perron p-value: 0.0000 (randamentele sunt staționare)
- ▣ Exponentul Hurst (H): 0.5476
- ▣ Parametrul d estimat: 0.0476
- ▣ Hurst randamente pătrate: $H = 0.6896$, $d = 0.1896$

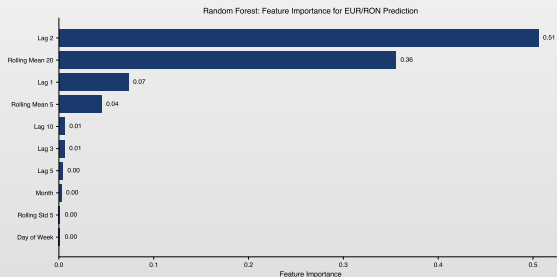
Interpretare

- ▣ Randamentele EUR/RON sunt **staționare** ($p\text{-value} < 0.05$)
- ▣ $H \approx 0.55$: aproape de limită, ușoară persistență
- ▣ $d \approx 0.05$: esențialmente **memorie scurtă** – ARMA poate fi suficient
- ▣ **Randamente pătrate**: $H \approx 0.69$, $d \approx 0.19$ – memorie lungă în volatilitate!

Random Forest: importanța features

Interpretare

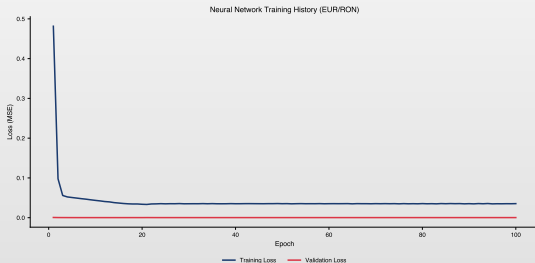
- **Date:** Cursul EUR/RON (Yahoo Finance, 2019–2025) — RF cu 10 features construite
- Lag-urile recente (lag_1, lag_2) și volatilitatea rolling sunt cele mai importante
- Features calendaristice au impact minor pentru randamente zilnice



LSTM: curba de învățare

Interpretare

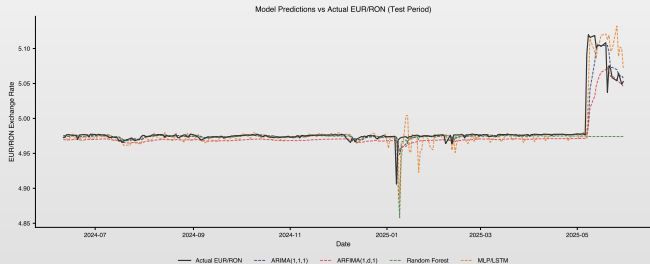
- ▣ **Date:** Cursul EUR/RON (2019–2025), 100 epoci, MSE loss
- ▣ Training loss scade rapid, apoi se stabilizează; validation loss urmărește — fără overfitting sever



Vizualizare: predicții vs valori reale

Interpretare

- Perioada de test EUR/RON — ARIMA, RF, MLP/LSTM vs valori reale
- Toate modelele captează tiparul general; niciun model nu prezice perfect vârful de volatilitate (eficiența pieței)



Comparație: Rezultate pe EUR/RON

Model	RMSE	MAE	Timp (s)	Interpretabil?
ARIMA(1,1,1)	0.0139	0.0038	24.32	Da
ARFIMA(1, d ,1)	0.0172	0.0088	0.16	Da
Random Forest	0.0314	0.0104	0.58	Parțial
MLP/LSTM	0.0162	0.0072	0.52	Nu

Proгноză rolling 1-pas pe niveluri de preț; split temporal 70/15/15; $d = 0.048$ (Hurst).

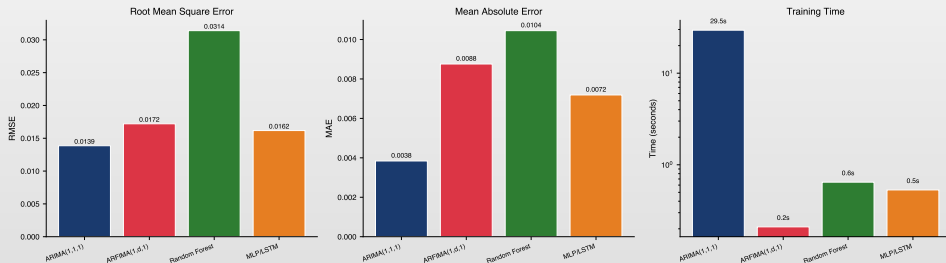
Concluzii

- ARIMA obține cel mai mic **RMSE** — greu de depășit pe piață eficientă
- ARFIMA și MLP/LSTM sunt aproape, dar **nu mai bune** decât ARIMA
- Random Forest supraaproximează ușor pe niveluri de preț
- Pentru EUR/RON, simplitatea câștigă — **principiul parsimoniei** confirmat

Comparație modele: metrici de performanță

Interpretare

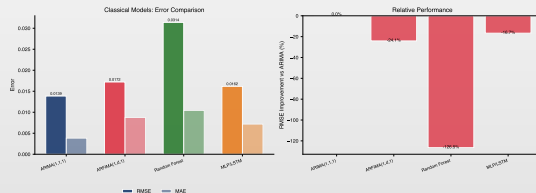
- EUR/RON (2019–2025): ARIMA vs ARFIMA vs RF vs MLP/LSTM
- Stânga:** RMSE (mai mic = mai bun) — ARIMA obține cel mai mic RMSE
- Dreapta:** Timp antrenare (scală log) — trade-off acuratețe vs. computație



Foundation Models: Previzualizare EUR/RON

Rezultate rapide: Zero-Shot pe EUR/RON

- Am aplicat **Chronos** și **TimesFM** zero-shot pe EUR/RON (2019–2025)
- Ambele modele prognozează *fără antrenare* — transfer learning pur
- Pe această piață eficientă, **ARIMA** clasic rămâne un baseline puternic



Pentru un tratament complet, vezi Capitolul 11

Capitolul 11 oferă o analiză detaliată a arhitecturilor foundation models (Chronos, TimesFM, Lag-Llama, Moirai, TimeGPT), strategii de tokenizare, zero-shot vs fine-tuning și studii de caz extinse pe multiple domenii.

Studiu de Caz 2: Consumul de Energie — Introducere

Set de date: Consumul Orar de Electricitate ENTSO-E

- ▣ **Sursă:** Platforma de Transparență ENTSO-E (date europene de electricitate)
- ▣ **Variabilă:** Consumul total orar de electricitate (MW) pentru o țară europeană
- ▣ **Perioadă:** 2015–2020, aproximativ 2,160 observații zilnice agregate
- ▣ **Frecvență:** Zilnică (agregată din date orare)

Provocarea Prognozei

- ▣ **Sezonalități multiple:** săptămânală (5 zile lucrătoare + weekend) și anuală (vară/iarnă)
- ▣ **Efecte neliniare:** temperatura, sărbătorile și activitatea economică influențează consumul
- ▣ **Schimbări structurale:** lockdown-urile COVID-19 au cauzat scăderi fără precedent în 2020
- ▣ **Obiectiv:** Compararea abordărilor clasice (SARIMA, ARFIMA) vs. ML (Random Forest, LSTM)

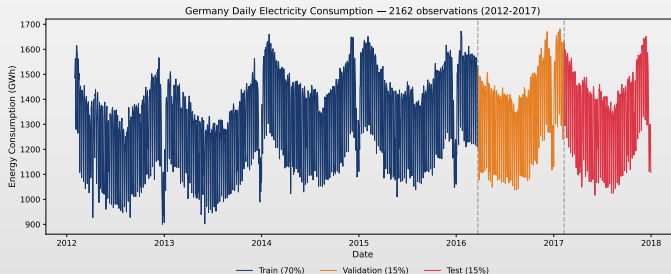
De Ce Acest Set de Date?

Prognoza energetică este o aplicație cheie din lumea reală: operatorii de rețea au nevoie de prognoze precise pe termen scurt pentru a echilibra cererea și oferta și a evita penele de curent.

Studiu de caz: Prezentarea datelor

Interpretare

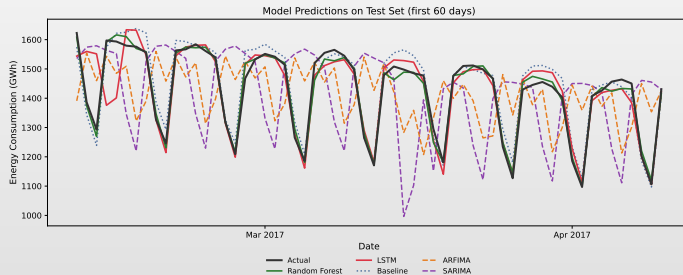
□ **Train:** 1513 obs (70%) **Validare:** 324 obs (15%) **Test:** 325 obs (15%)



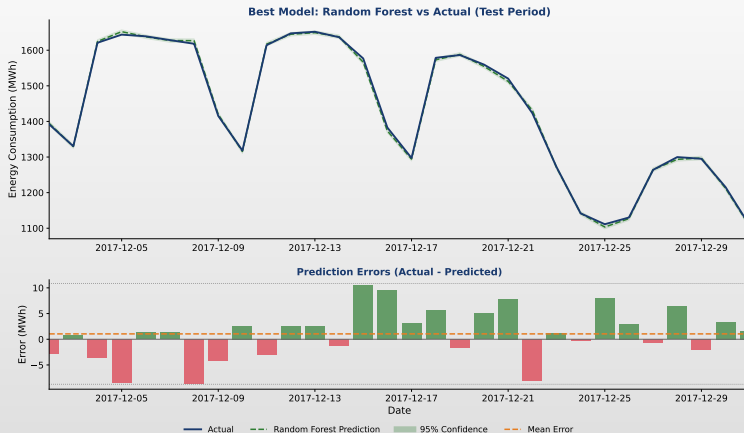
TSA_ch8_data_split

Studiu de caz: Predicții ale modelelor

Rang	Model	MAPE	Interpretare
1	Random Forest	2.2%	Cel mai bun: captează tipare neliniare
2	LSTM	3.3%	Bun, necesită mai multe date
3	Baseline	3.9%	Simplu dar competitiv
4	ARFIMA	12.3%	Memoria lungă nu e suficientă
5	SARIMA	14.6%	Dificultăți cu tiparele



Studiu de caz: Performanța celui mai bun model



Când să alegem fiecare model?

ARIMA/ARFIMA

- ▣ Date puține (< 500 obs.)
- ▣ Interpretare importantă, memorie lungă suspectată
- ▣ Baseline rapid și eficient

Random Forest

- ▣ Multe variabile exogene, relații neliniare
- ▣ Importanța features, date moderate (500–10.000 obs.)

LSTM/Deep Learning

- ▣ Date foarte mari (> 10.000 obs.), secvențe complexe
- ▣ Resurse computaționale disponibile, tipare ascunse

Exemplu 2: indicele BET (Bursa București)

Caracteristici

- ▣ **Volatility clustering** puternic
- ▣ Influențat de piețele internaționale
- ▣ Lichiditate mai redusă decât piețele dezvoltate
- ▣ Potențial pentru memorie lungă în volatilitate

Rezultate pe randamente BVB.RO (2019–2025, RMSE în %)

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

Exemplu 3: rata inflației în România

Caracteristici

- ▣ Serie **lunară** (frecvență redusă)
- ▣ **Persistență ridicată** – șocurile durează
- ▣ Influențată de politica monetară
- ▣ Potențial puternic pentru **memorie lungă**

Rezultate tipice

- ▣ ARFIMA cu $d \approx 0.35$ – captează persistența
- ▣ ARIMA subestimează persistența șocurilor
- ▣ ML nu funcționează bine (date puține, 300 obs.)

- ▣ **Lecție:** Pentru serii lunare cu puține date, modelele clasice (ARFIMA) sunt superioare.

Rezumat practic: fluxul recomandat

Fluxul recomandat de modelare

1. Începeți cu **ARIMA** ca baseline (simplu, rapid, interpretabil)
2. Testați **memorie lungă** $\Rightarrow$ ARFIMA dacă d semnificativ
3. Adăugați **features** $\Rightarrow$ Random Forest pentru relații neliniare
4. Doar cu date multe (> 10.000 obs.) și resurse $\Rightarrow$ LSTM

Lecții din studiile de caz

- ▣ **EUR/RON**: Piață eficientă $\Rightarrow$ diferențe mici între modele
- ▣ **Bitcoin**: Volatilitatea beneficiază de memorie lungă (ARFIMA)
- ▣ **Energie**: RF cel mai bun $\Rightarrow$ tipare neliniare complexe
- ▣ **Inflație**: Date puține $\Rightarrow$ ARFIMA superior ML

Exercițiu AI: Gândire critică

Prompt de testat în ChatGPT / Claude / Copilot

"Folosind yfinance, descarcă prețurile zilnice Bitcoin (BTC-USD) din 2019-01-01 până în 2024-12-31 (aprox. 2.200 observații). Calculează randamentele procentuale zilnice. Compară ARIMA, Random Forest și LSTM pentru prognoze pe 7 zile, folosind împărțirea temporală 70/15/15. Care model e cel mai bun? Vreau cod Python complet cu grafice comparative."

Exercițiu

1. Rulați prompt-ul într-un LLM (Large Language Model) la alegere și analizați critic răspunsul
2. Cum sunt construite features pentru Random Forest? Lag-uri, variabile calendar, termeni Fourier?
3. LSTM-ul este structurat corect? Forma intrărilor, scalare, split fără leakage?
4. Folosește walk-forward validation sau doar un singur split?
5. Menționează compromisurile între interpretabilitate și costul computațional?

☐ **Atenție:** Codul generat de AI poate produce cod funcțional cu aparență de corectitudine. *Asta nu înseamnă că e corect.*

Rezumat

Ce am învățat

- ▣ **ARFIMA**: Extinde ARIMA pentru memorie lungă (d fracționar)
- ▣ **Random Forest / XGBoost**: Relații neliniare, câștigătoare M5
- ▣ **LSTM / TFT**: Deep learning cu atenție interpretabilă și regresie cuantică
- ▣ **Foundation Models**: TimeGPT, Chronos — paradigma zero-shot
- ▣ **Lecția M4/M5**: abordările hibride (statistică + ML) sunt cele mai robuste

Recomandări practice

- ▣ Începeți cu modele **simple** (ARIMA) ca baseline
- ▣ Folosiți **Time Series CV** pentru evaluare corectă
- ▣ ML necesită **feature engineering** atent (lag-uri, calendar, rolling stats)
- ▣ Considerați **foundation models** pentru prototipare rapidă (zero-shot)
- ▣ Abordarea hibridă: features statistice + ML = cel mai bun compromis

Întrebarea 1

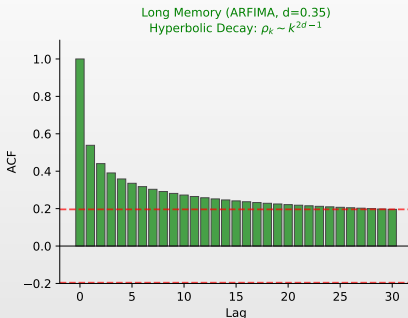
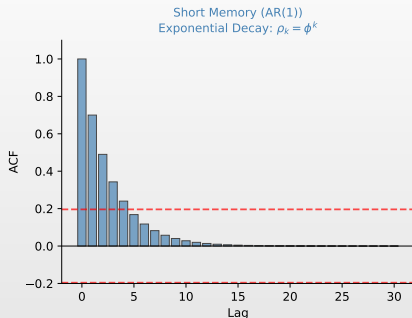
Întrebare

□ Ce semnifică $d = 0.3$ într-un model ARFIMA?

Variante de răspuns

- (A) Seria necesită 0.3 diferențieri pentru a deveni staționară
- (B) Memorie lungă: staționară dar ACF scade hiperbolic (lent)
- (C) Seria este nestaționară cu rădăcină unitară
- (D) Memorie scurtă: ACF scade exponențial (rapid)

Întrebarea 1: Răspuns



Răspuns: (B)

- Pentru $0 < d < 0.5$: staționară dar $ACF \sim k^{2d-1}$ scade mult mai lent decât exponențial
- Observațiile îndepărtate au încă influență semnificativă

Întrebarea 2

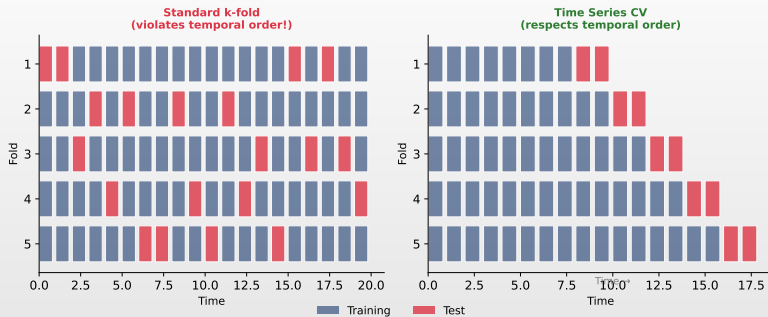
Întrebare

☐ De ce trebuie să folosim Time Series CV în loc de k-fold standard?

Variante de răspuns

- (A) k-fold este mai costisitor computațional
- (B) Time Series CV folosește mai multe date de antrenament
- (C) k-fold violează ordinea temporală, cauzând data leakage
- (D) Nu există diferență; ambele metode sunt echivalente

Întrebarea 2: Răspuns



Răspuns: (C)

- ☐ K-fold standard amestecă datele aleator, folosind observații viitoare pentru a prezice trecutul
- ☐ Time Series CV antrenează pe trecut și testează pe viitor, respectând cauzalitatea

Întrebarea 3

Întrebare

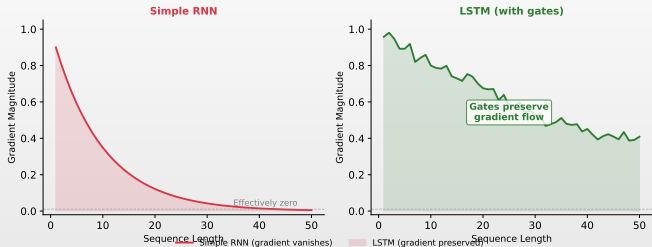
☐ Care este avantajul principal al LSTM față de RNN simplu?

Variante de răspuns

- (A) LSTM folosește mai puțini parametri
- (B) LSTM rezolvă problema vanishing gradient prin mecanismul de porți
- (C) LSTM se antrenează mai rapid
- (D) LSTM nu necesită date secvențiale

Întrebarea 3: Răspuns

Vanishing Gradient Problem: RNN vs LSTM



Răspuns: (B)

- Porțile LSTM controlează fluxul de informație, preservând gradientul pe secvențe lungi; RNN simplu pierde semnalul după ~10–20 pași

Întrebarea 4

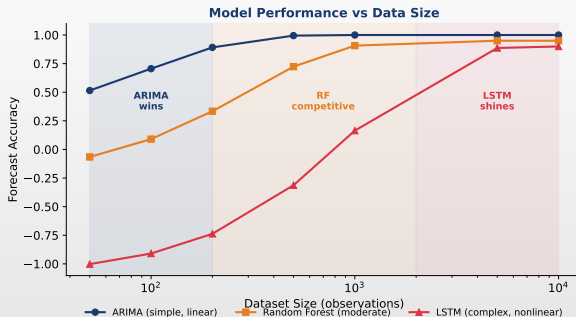
Întrebare

▣ Aveți un set de date mic (100 observații) cu relații liniare. Ce model este cel mai potrivit?

Variante de răspuns

- (A) LSTM — deep learning captează toate tiparele
- (B) Random Forest — gestionează orice relație
- (C) ARIMA/ARFIMA — parcimonios și eficient cu date puține
- (D) Ansamblu de toate modelele pentru acuratețe maximă

Întrebarea 4: Răspuns



Răspuns: (C)

- Modelele ML (RF, LSTM) necesită seturi mari de date pentru a generaliza
- Cu 100 observații și dinamică liniară, parametrii puțini ai ARIMA evită overfitting-ul

Întrebarea 5

Întrebare

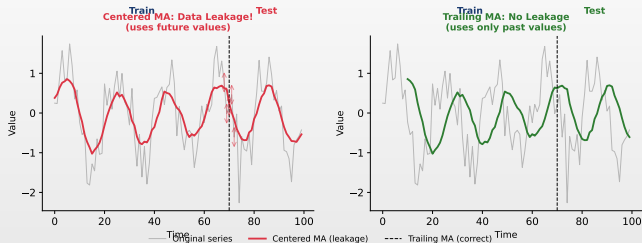
☐ Ce este “data leakage” în contextul ML pentru serii de timp?

Variante de răspuns

- (A) Valori lipsă în setul de date
- (B) Folosirea informației din viitor în features sau în antrenare
- (C) Prea multe features relativ la observații
- (D) Modelul memorează datele de antrenament

Întrebarea 5: Răspuns

Data Leakage in Time Series Feature Engineering



Răspuns: (B)

- Exemple: medii mobile centrate (valori viitoare), k-fold standard (amestecă ordinea temporală), statistici pe tot setul înainte de split

Ce urmează?

Extensii și subiecte avansate

- ▣ **Prophet** (Facebook/Meta) pentru sezonalitate multiplă
- ▣ **Neural Prophet**: Prophet + rețele neuronale
- ▣ **Conformal prediction**: intervale de incertitudine model-agnostice
- ▣ **Reconciliere ierarhică**: prognoze coerente la multiple niveluri de agregare
- ▣ **Anomaly detection** cu ML

Întrebări?

Bibliografie I

Memorie lungă și 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.

Rețele neuronale și deep learning pentru serii de timp

- Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory, *Neural Computation*, 9(8), 1735–1780.
- Bai, J., & Perron, P. (2003). Computation and Analysis of Multiple Structural Change Models, *Journal of Applied Econometrics*, 18(1), 1–22.

Bibliografie II

Transformers, Foundation Models și Competiții

- ▣ Lim, B., Arık, S.Ö., Loeff, N., & Pfister, T. (2021). Temporal Fusion Transformers for Interpretable Multi-horizon Time Series Forecasting, *International Journal of Forecasting*, 37(4), 1748–1764.
- ▣ Garza, A., & Mergenthaler-Canseco, M. (2024). TimeGPT-1, arXiv:2310.03589.
- ▣ Ansari, A.F., et al. (2024). Chronos: Learning the Language of Time Series, arXiv:2403.07815.
- ▣ Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2022). M5 Accuracy Competition: Results, Findings, and Conclusions, *International Journal of Forecasting*, 38(4), 1325–1346.

Bibliografie III

Modele cu prag, regim-switching și referințe suplimentare

- ▣ Hansen, B.E. (2011). Threshold Autoregression in Economics, *Statistics and Its Interface*, 4(2).
- ▣ Hamilton, J.D. (1989). A New Approach to Nonstationary Time Series, *Econometrica*, 57(2).
- ▣ Petropoulos, F., et al. (2022). Forecasting: Theory and Practice, *Int. J. Forecasting*, 38(3).
- ▣ Smyl, S. (2020). Hybrid Exponential Smoothing and RNN, *Int. J. Forecasting*, 36(1).

Resurse online și cod

- ▣ **Quantlet**: <https://quantlet.com> – Platformă de cod pentru metode cantitative
- ▣ **Quantinar**: <https://quantinar.com> – Platformă de învățare pentru metode cantitative
- ▣ **GitHub TSA**: https://github.com/QuantLet/TSA/tree/main/TSA_ch8 – Cod Python pentru acest capitol

Vă Mulțumim!

Întrebări?

Materialele cursului sunt disponibile la: <https://danpele.github.io/Time-Series-Analysis/>

 Quantlet

 Quantinar