



Analiza și Prognoza Seriilor de Timp

Capitolul 8: Extensiile 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 (RF – ansamblu de arbori de decizie) pentru prognoză seriilor de timp
4. Construiți rețele LSTM (Long Short-Term Memory) pentru serii temporale
5. Comparați performanța modelelor clasice vs ML (Machine Learning)
6. Alegeti metoda potrivită în funcție de context
7. Implementați aceste metode în Python



Cuprins

- Motivație
- ARFIMA: Modele cu Memorie lungă
- Random Forest pentru serii de timp
- LSTM: Deep Learning pentru serii de timp
- Comparație și Selecția modelului
- Aplicații practice
- Studiu de Caz Complet: Cursul EUR/RON
- Comparație 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**: Relații neliniare, robustețe la outlieri
- LSTM**: Tipare secvențiale complexe, dependențe pe termen lung



Când să folosim fiecare metodă?

Caracteristică	ARIMA	ARFIMA	RF	LSTM
Memorie lungă	✗	✓	✓	✓
Relații neliniare	✗	✗	✓	✓
Interpretabilitate	✓	✓	~	✗
Date puține	✓	✓	✗	✗
Variabile exogene	✓	✓	✓	✓
Incertitudine	✓	✓	~	✗

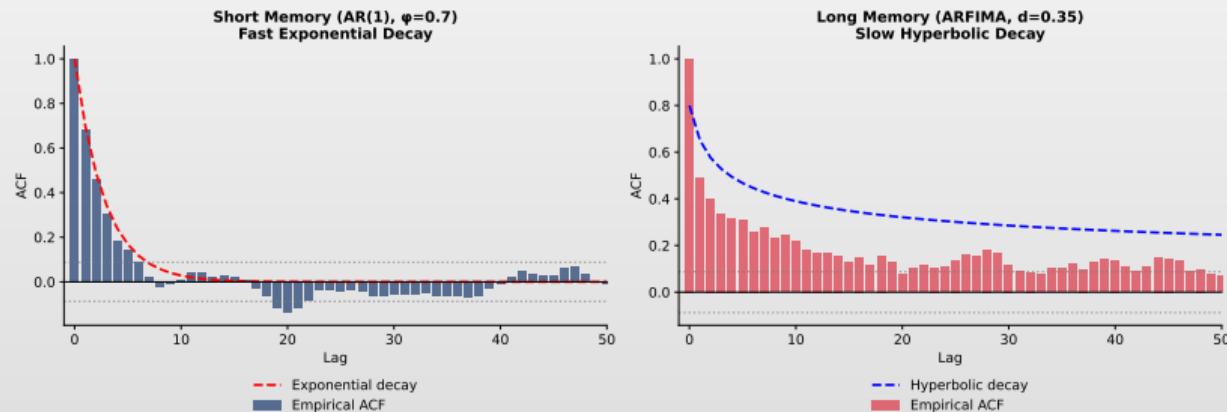
Regula de aur

- Începe **simplu** (ARIMA), apoi crește complexitatea doar dacă este justificat de date și performanță.

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

Interpretare

- Date: AR(1) simulație cu $\phi = 0.8$ și ARFIMA($0,d,0$) cu $d = 0.35$ ($n = 1000$)
- Stânga: AR(1) — autocorelații care scad exponential (memorie scurtă)
- Dreapta: ARFIMA cu $d = 0.35$ — autocorelații care scad hiperbolice (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: $\sum_{k=0}^{\infty} |\rho_k| < \infty$

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 $d > 0$)

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:** $d \in (-0.5, 0.5)$ este parametrul de diferențiere fracționară

Operatorul de diferențiere fracționară

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

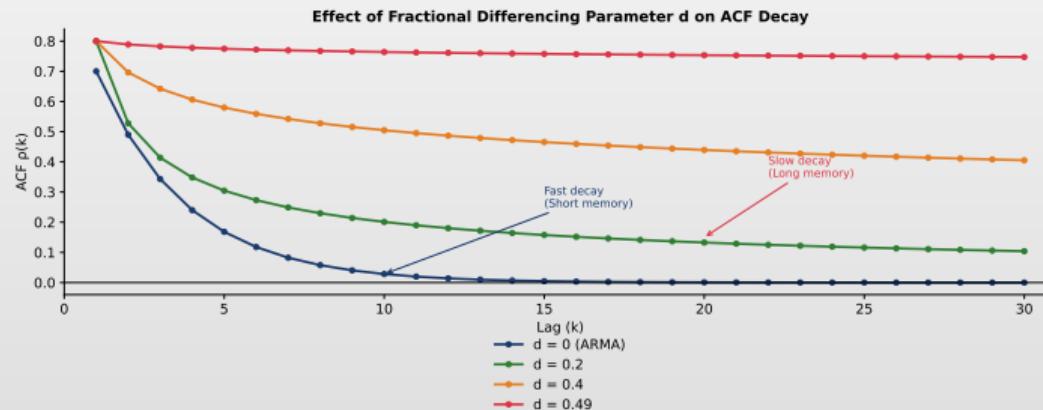
Cazuri particulare

- $d = 0$: 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($0,d,0$) simulațat 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



Q TSA_ch8_arfima_d_effect

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
$d = 1$	ACF = 1 (constant)	Random walk

Parametrul Hurst H

Relația: $d = H - 0.5$

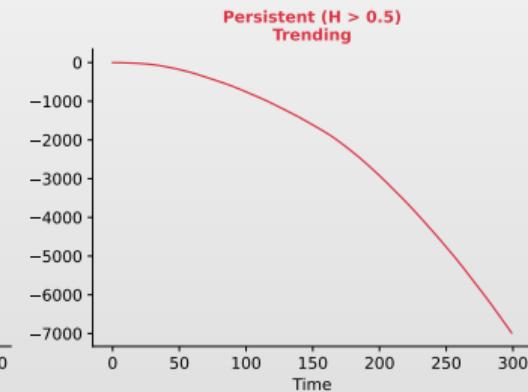
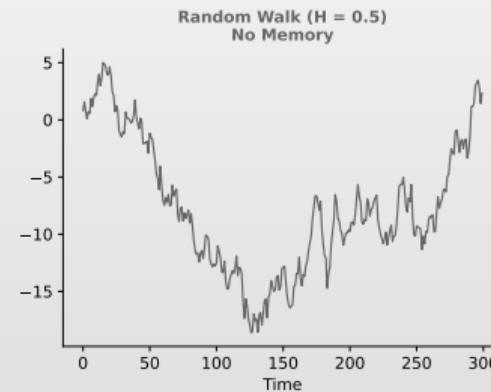
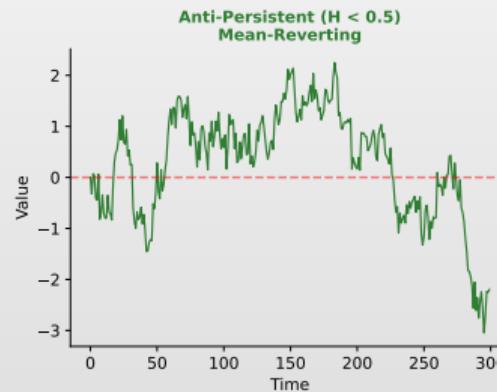
- ▶ $H = 0.5$: Mers aleator (fără memorie)
- ▶ $H > 0.5$: Persistență (urmărirea trendului)
- ▶ $H < 0.5$: Anti-persistență (revenire la medie)



Exponentul Hurst: interpretare vizuală

Interpretare

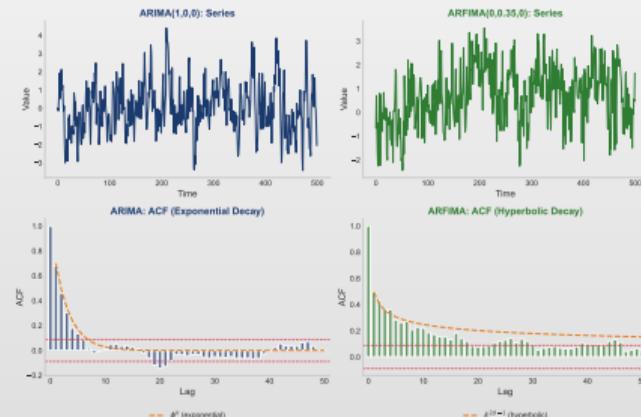
- Date:** Mișcare Browniană fracționară simulată cu $H \in \{0.3, 0.5, 0.7\}$
- $H < 0.5$: Revenire la medie $H = 0.5$: Mers aleator
- $H > 0.5$: Persistentă (urmărirea trendului)



ARIMA vs ARFIMA: comparație simulată

Interpretare

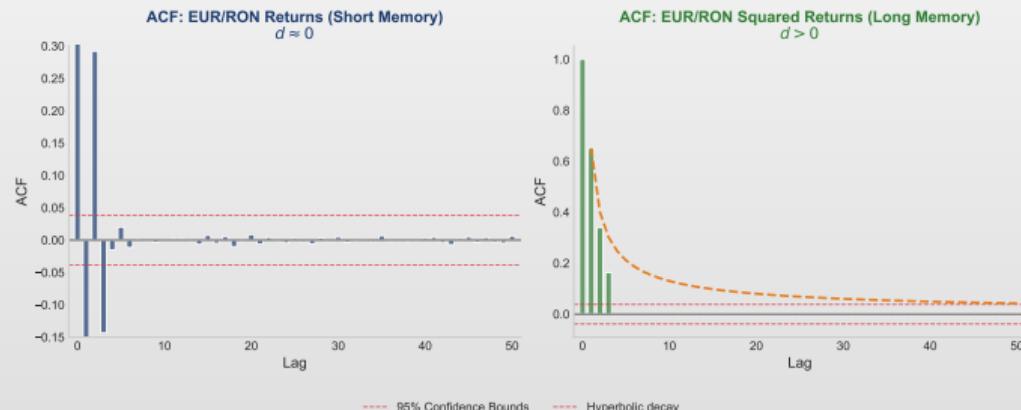
- Date: ARIMA(1,1,1) simulat vs ARFIMA(1, d ,1) cu $d = 0.35$
- ARIMA (stânga): ACF scade **exponențial** — řocurile sunt “uite” rapid
- ARFIMA (dreapta, $d = 0.35$): ACF scade **hiperbolic** — řocurile persistă mult timp



Exemplu date reale: analiza memoriei lungi EUR/RON

Interpretare

- Date:** Cursul zilnic EUR/RON (Yahoo Finance, 2015–2025)
- Randamente:** $H \approx 0.50$, $d \approx 0$ — memorie scurtă
- Randamente pătrate:** $H \approx 0.65$, $d \approx 0.15$ — memorie lungă în volatilitate



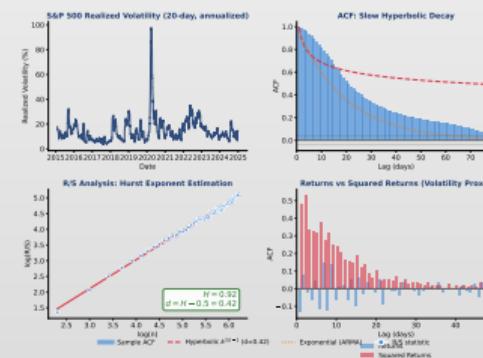
Exemplu ARFIMA: Volatilitatea realizată S&P 500

Rezultate estimare

- Date: Randamentele zilnice S&P 500 (Yahoo Finance, 2015–2024)
- Hurst: $H = 0.92$, $d = H - 0.5 = 0.42$ – memorie lungă puternică

Observație

- Volatilitatea are **memorie lungă** – șocurile persistă mai mult decât în ARMA
- 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

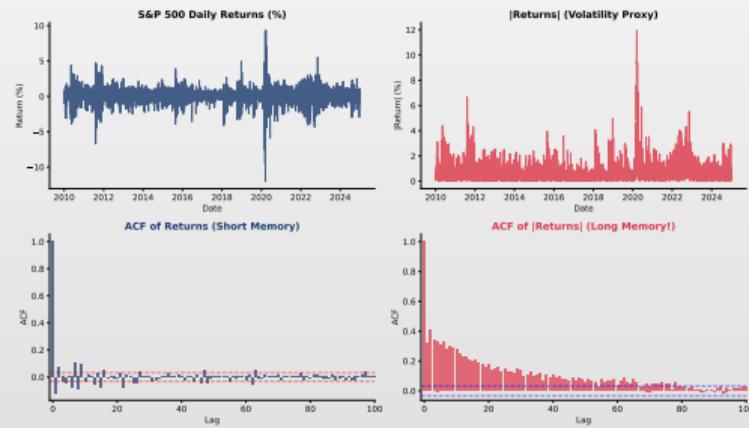
- În Python: pachetul `arch`, sau `statsmodels.tsa.arima.model.ARIMA`
- Se specifică `order=(p,d,q)` unde d poate fi fraționar



Exemplu real: memorie lungă în volatilitate

Notă

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



Q TSA_ch8_volatility_long_memory



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 features
- Predicția finală = **media** predicțiilor tuturor arborilor

Avantaje pentru serii de timp

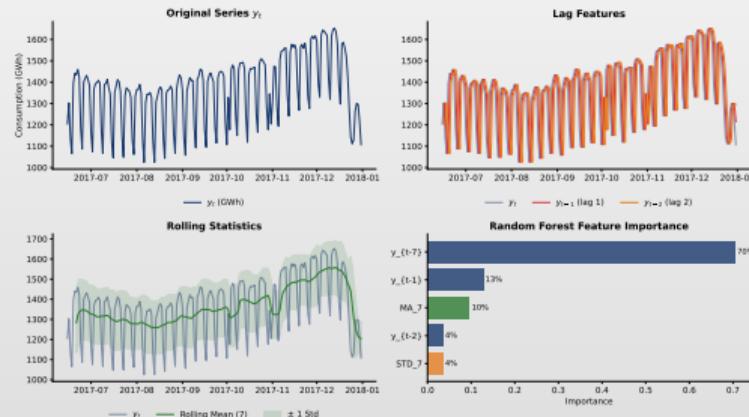
- Captează **relații neliniare**
- Robust** la outlieri și zgomot
- Nu necesită **staționaritate**
- Oferă **importanța features** (interpretabilitate)
- Funcționează bine cu **multe variabile**



Feature engineering: ilustrare

Interpretare

- Date: Consum zilnic de electricitate Germania (OPSD – Open Power System Data, 2012–2017)
- Transformăm seria temporală în features: lag-uri, statistici rolling
- Modelul RF învăță 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. **Rolling statistics:** medie mobilă, deviație standard
3. **Calendar features:** ziua săptămânii, luna, sezon
4. **Trend features:** timp, trend pătratic
5. **Variabile exogene:** indicători economici, evenimente

Atenție: Data Leakage!

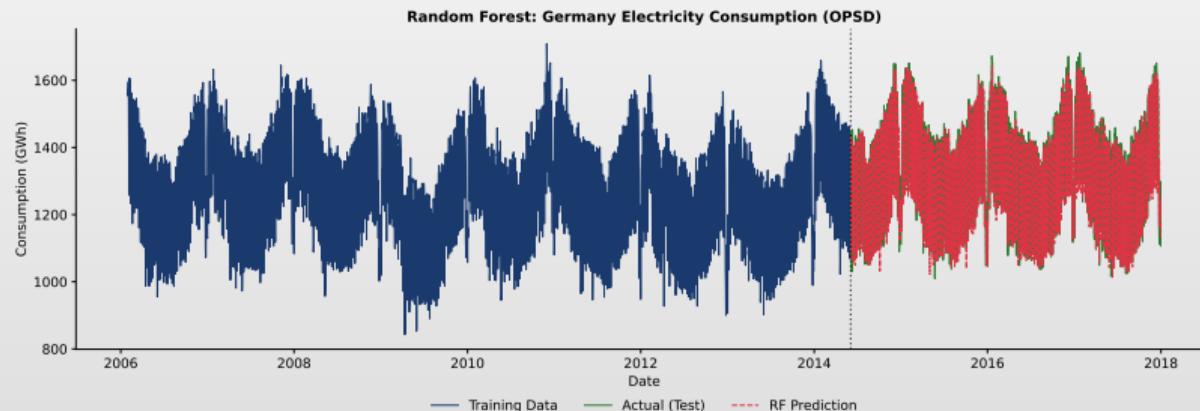
- Nu folosi informații din viitor în features
- Train/test split: **temporal**, nu aleator!
- Rolling statistics: calculează doar pe date **anterioare**



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 features și interpretare

Feature importance

- Mean Decrease Impurity (MDI)**: Reducerea impurității la fiecare split
- Permutation Importance**: Cât scade performanța când feature-ul e permuatat aleator

Interpretare tipică pentru serii de timp

- lag_1 foarte important \succ Autocorelare puternică
- rolling_mean important \succ Trend local contează
- month important \succ Sezonalitate prezentă

Cod

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



Portret de cercetător: Hochreiter & Schmidhuber



Sepp Hochreiter (*1967)

[W Wikipedia \(en\)](#)

Jürgen Schmidhuber (*1963)

[W Wikipedia \(en\)](#)

Biografie

- **Sepp Hochreiter:** informatician austriac, profesor la Johannes Kepler University Linz
- Conducător al ELLIS (European Laboratory for Learning and Intelligent Systems) Unit Linz
- **Jürgen Schmidhuber:** informatician germano-elvețian, Director Științific al IDSIA (Istituto Dalle Molle di Studi sull'Intelligenza Artificiale)
- Împreună au rezolvat problema gradientului care dispare

Contribuții principale

- **Long Short-Term Memory (LSTM, 1997)** — arhitectură recurrentă cu porți
- Rezolvă problema gradientului care dispare
- **Analiza gradientului care dispare** (Hochreiter, 1991) — identificarea problemei fundamentale
- **Poarta forget** (Gers et al., 2000) — extensie crucială pentru LSTM
- Fundament pentru modelarea modernă a secvențelor în NLP (Natural Language Processing), voce și serii de timp

De la neuronul biologic la cel artificial

Analogia

- Dendrite \succ Intrări x_i Sinapse \succ Ponderi w_i
- Soma \succ Sumă + Activare Axon \succ Ieșire y



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

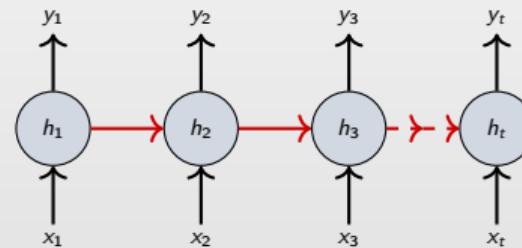
 TSA_ch8_neuron_comparison



Rețele neuronale recurente – RNN (Recurrent Neural Network)

Ideea de bază

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



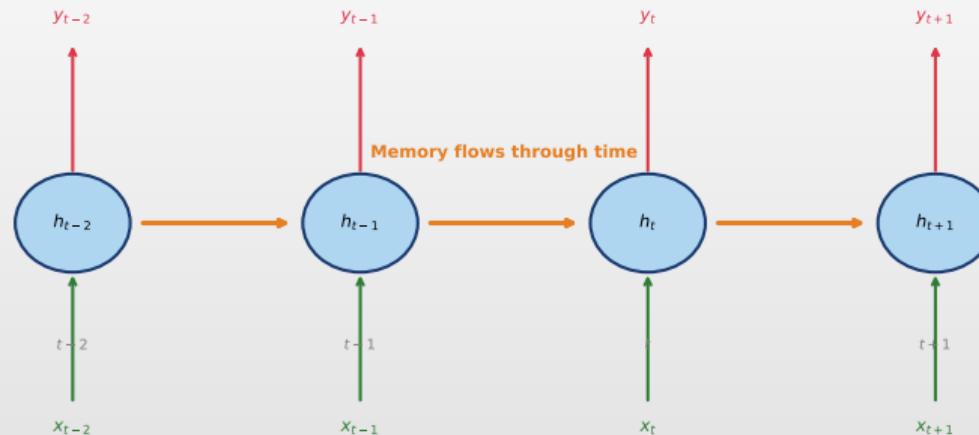
Problema: Vanishing Gradient

- RNN simple “uită” informația din trecut îndepărtat.



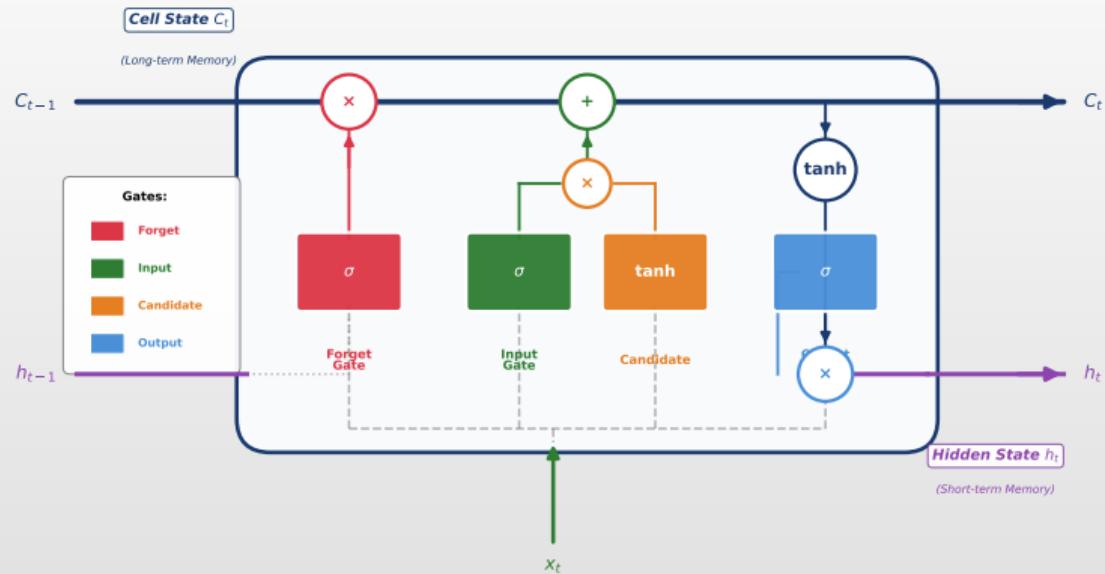
RNN desfășurată în timp

Recurrent Neural Network (Unfolded Through Time)



Q TSA_ch8_rnn_unfolded

Celula LSTM: Diagrama detaliată



- Forget Gate f_t
- $\sigma(W_f[h_{t-1}, x_t] + b_f)$
- Ce să uităm?

- Input Gate i_t
- $\sigma(W_i[h_{t-1}, x_t] + b_i)$
- Ce să stocăm?

- Output Gate o_t
- $\sigma(W_o[h_{t-1}, x_t] + b_o)$
- Ce să transmitem?

LSTM: long short-term memory

Soluția LSTM

- **Concept:** Celule speciale cu 3 porți care controlează fluxul informației
- **Forget Gate (f_t):** Ce să uităm din memoria anterioară
- **Input Gate (i_t):** Ce informație nouă să adăugăm
- **Output Gate (o_t):** Ce să trimitem la ieșire

Ecuările LSTM



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

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

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

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

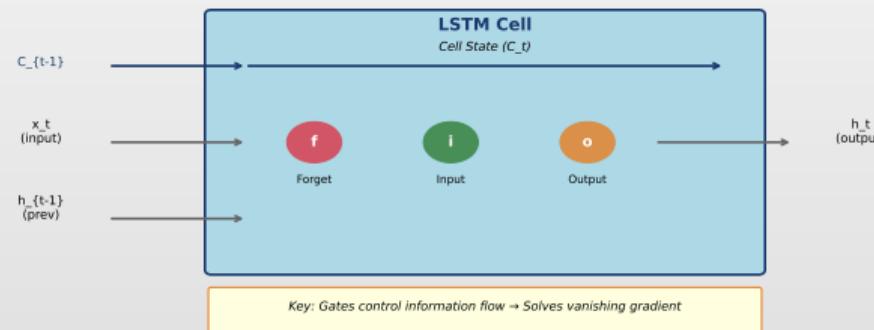
$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \quad (\text{Output})$$

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

Arhitectura celulei LSTM

Interpretare

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



TSA_ch8_lstm_architecture



Avantajele LSTM pentru serii de timp

De ce LSTM?

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

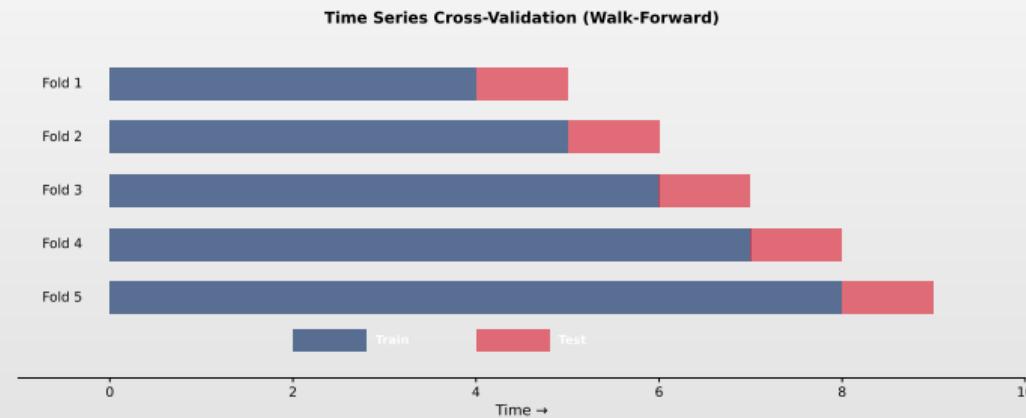
Dezavantaje

- Necesită **multe date** pentru antrenare
- Computațional intensiv**
- “**Black box**” - greu de interpretat
- Sensibil la **hiperparametri**
- Poate face **overfitting** ușor



Time series cross-validation

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



Q TSA_ch8_timeseries_cv

Metrici de evaluare

Metrici comune

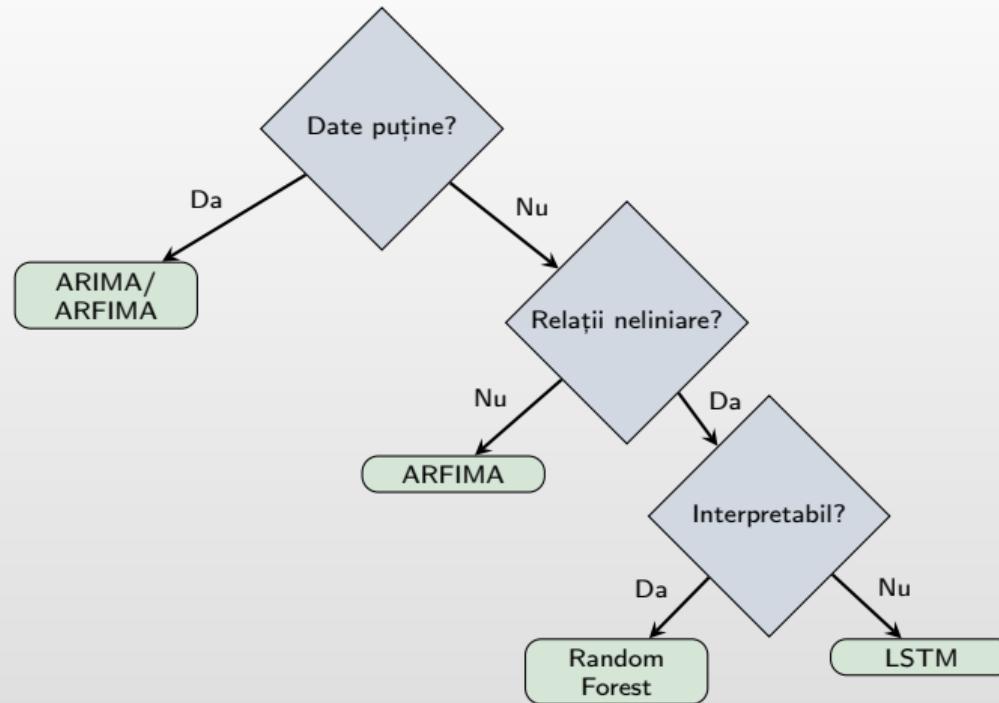
- RMSE** (Root Mean Squared Error): $\sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2}$ \succ Eroare în unități originale
- MAE** (Mean Absolute Error): $\frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i|$ \succ Robust la outlieri
- MAPE** (Mean Absolute Percentage Error): $\frac{100}{n} \sum_{i=1}^n \left| \frac{y_i - \hat{y}_i}{y_i} \right|$ \succ Eroare procentuală
- MASE** (Mean Absolute Scaled Error): Comparat cu benchmark naiv

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



Bitcoin: evoluția prețului și randamentele

Observații

- Creștere exponențială a prețului \succ distribuție puternic **leptokurtotică**
- Randamentele zilnice: medie $\approx 0.15\%$, volatilitate $\approx 3.5\%$
- Volatility clustering evident \succ perioadele de criză (2018, 2020, 2022)
- Kurtosis $\approx 10\text{--}15$ (mult peste 3 al normalei)



Studiu de caz: prognoza prețului Bitcoin

De ce Bitcoin?

- Volatilitate **extremă** și tipare complexe
- Potențială **memorie lungă** în volatilitate
- Relații **neliniare** cu variabile exogene
- Date disponibile la **frecvență înaltă**

Abordare comparativă

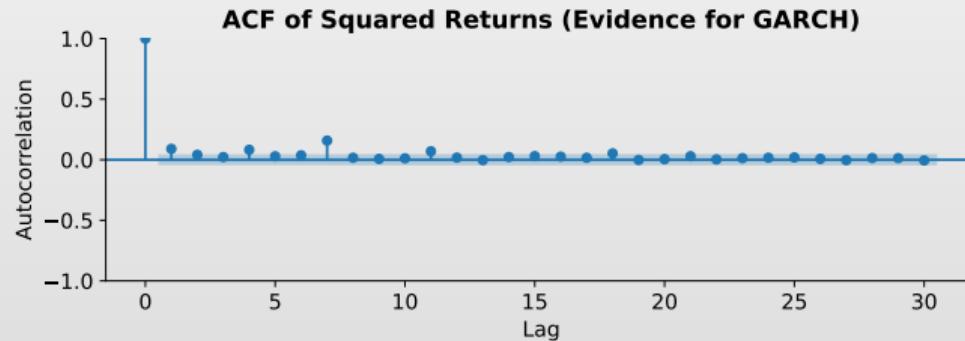
1. ARIMA pe randamente
2. ARFIMA pentru memorie lungă
3. Random Forest cu features tehnice
4. LSTM pe secvențe de prețuri



Bitcoin: ACF și evidență pentru memorie lungă

Analiză ACF

- ◻ ACF randamente: scădere rapidă \succ memorie scurtă în medie
- ◻ ACF randamente pătrate: scădere lentă, hiperbolică
 - ▶ Indică **memorie lungă în volatilitate**
 - ▶ Hurst $H \approx 0.65\text{--}0.70$ ($d \approx 0.15\text{--}0.20$)
- ◻ ARFIMA pe volatilitate $>$ ARMA \succ captează persistența șocurilor



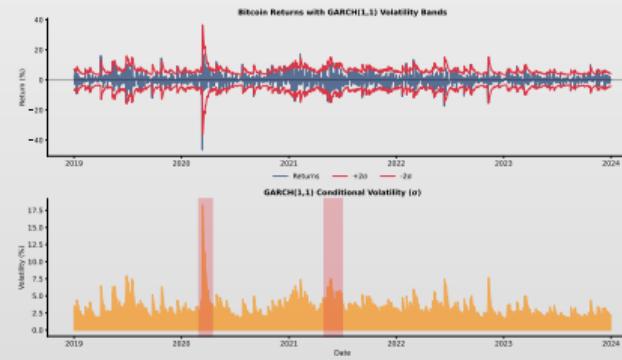
 TSA_ch10_btc_acf_squared



Bitcoin: GARCH și managementul riscului

Concluzii – Studiu Bitcoin

- Diferențele între modele sunt **mici** pentru media randamentelor
- Valoarea adăugată majoră: **modelarea volatilității cu GARCH** (Generalized Autoregressive Conditional Heteroskedasticity) și EGARCH (Exponential GARCH)
- ARFIMA captează persistența în volatilitate (memorie lungă)
- Random Forest: util pentru **features neliniare** (volum, sentiment)
- Combinăție optimă: ARFIMA-GARCH + features exogene via RF

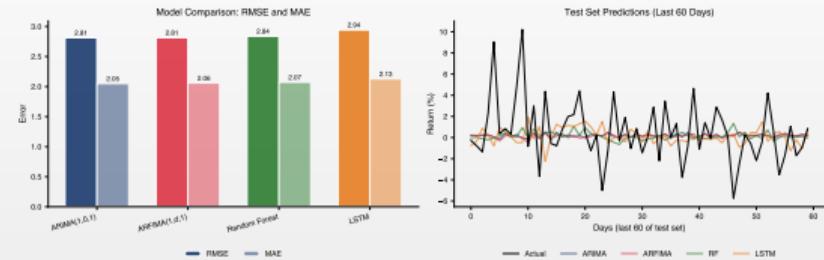


Bitcoin: estimare ARFIMA și comparație modele

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

Model	RMSE	MAE	Interpr.?
ARIMA(1,0,1)	2.81	2.05	Da
ARFIMA(1,d,1)	2.81	2.06	Da
Random Forest	2.84	2.07	Partial
LSTM	2.94	2.13	Nu

- Hurst $\hat{d} = 0.13$; împărțire temporală 70/15/15



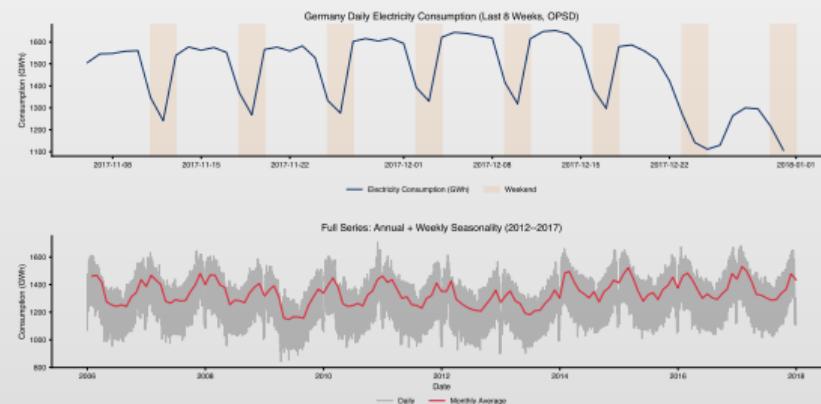
TSA_ch8_btc_model_comparison



Energie: vizualizarea cererii și sezonialitatea multiplă

Tipare identificate

- Zilnic** (24h): vârf dimineată (8–10) și seara (18–21), minim noaptea
- Săptămânal** (168h): consum redus în weekend (~15–20% mai puțin)
- Anual** (8766h): vârf vara (aer condiționat) și iarna (încălzire)
- SARIMA (Seasonal ARIMA) nu poate modela simultan aceste 3 perioade!



 TSA_ch9_electricity_demand



Studiu de caz: prognoza consumului de energie

Caracteristici

- Sezonalitate multiplă:** zilnică, săptămânală, anuală
- Tendință** de creștere pe termen lung
- Variabile exogene:** temperatură, zi liberă, preț
- Anomalii:** evenimente speciale, defecțiuni

Provocări

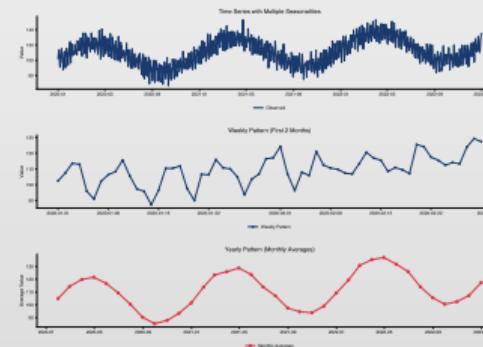
- Tipare la scale temporale diferite
- Interacțiuni complexe între variabile
- Necesitatea prognozelor pe orizonturi diferite



Energie: de ce Prophet și TBATS?

Soluția: modele cu sezonalitate multiplă

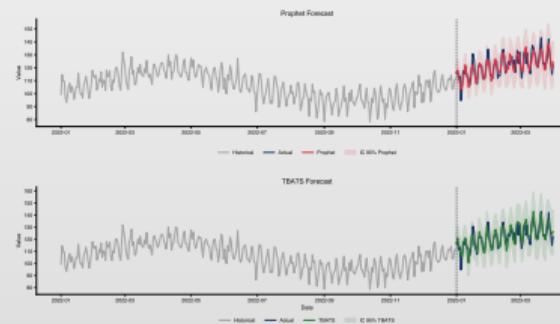
- **TBATS** (Trigonometric, Box-Cox, ARMA, Trend, Seasonal): perioade [24, 168, 8766] \curvearrowright Fourier pentru fiecare sezon
 - ▶ Automat, fără reglaj manual, bun pentru producție
- **Prophet**: sezonalitate aditivă/multiplicativă + regresori
 - ▶ Adaugă temperatură, zile libere, evenimente speciale
- **ARIMA clasic**: poate doar 1 sezon \curvearrowright MAPE \approx 8–10%



Energie: descompunere Prophet și rezultate

Rezultate comparație pe date energie (MAPE)

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



Energie: concluzii și recomandări practice

Lecții învățate

- Modelele cu **sezonalitate multiplă** reduc MAPE cu ~50% față de SARIMA
- Variabilele exogene** (temperatură) aduc câștig suplimentar de 10–15%
- Prophet exceleză la **interpretabilitate**: descompunere trend + sezon + holiday
- TBATS: cel mai bun **out-of-the-box** > fără reglaj de hiperparametri

Ce model alegem?

- Prophet**: când ai regresori externi + interpretare pentru management
- TBATS**: automatizare, producție, fără intervenție umană
- LSTM/RF**: dacă ai >100.000 observații și tipare neliniare complexe

-
- Detalii complete despre Prophet și TBATS > Capitolul 9*



Formule principale – Rezumat

ARFIMA(p,d,q)

- $\phi(L)(1 - L)^d Y_t = \theta(L)\varepsilon_t \quad d \in (-0.5, 0.5)$: memorie lungă

Memorie lungă

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

Random Forest

- $\hat{y} = \frac{1}{B} \sum_{b=1}^B T_b(x) \quad B$ arbori, features aleatorii

LSTM Cell

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

Metrici evaluare

Analiza și Prognoza Serială (y_t vs. \hat{y}_{t+1})

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



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 aceleasi date
- Scopul: înțelegerea punctelor forte ale fiecărei metode



Vizualizarea seriei EUR/RON

Interpretare

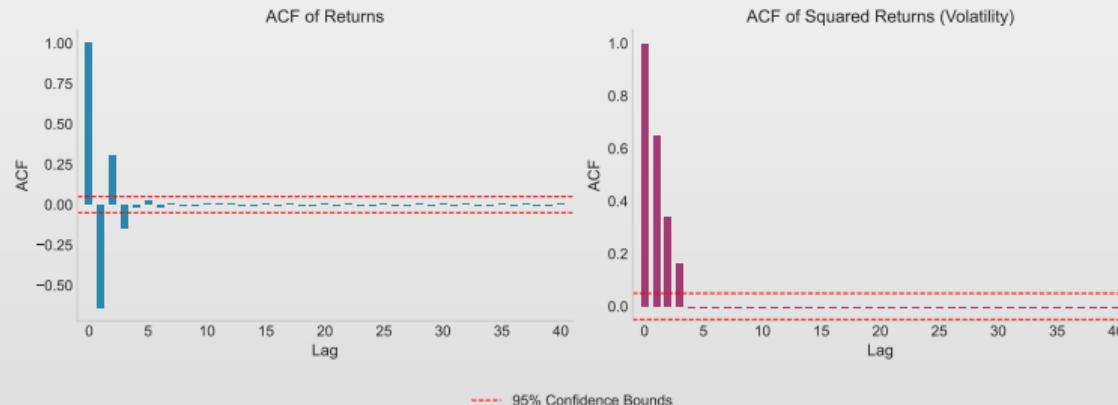
- Date: Cursul zilnic EUR/RON (Yahoo Finance, 2019–2025)
- Sus: Cursul EUR/RON — tendință de depreciere a leului și perioade de volatilitate ridicată
- Jos: Randamentele zilnice — volatility clustering
- Perioadele de volatilitate mare sunt urmate de alte perioade similare



Analiză ACF: randamente vs randamente pătrate

Interpretare

- Date: Randamentele zilnice și randamentele pătrate EUR/RON (Yahoo Finance, 2019–2025)
- Stânga: ACF al randamentelor — scădere rapidă, fără autocorelație semnificativă după lag 1
- Dreapta: ACF al randamentelor pătrate — scădere lentă
- Indică **volatility clustering** (efecte ARCH – AutoRegressive Conditional Heteroskedasticity)



Rezultate test memorie lungă – EUR/RON

Output tipic

- Phillips-Perron p-value: 0.0001 (randamentele sunt staționare)
- Exponentul Hurst (H): 0.47
- Parametrul d estimat: -0.03
- Serie ușor ANTI-PERSISTENȚĂ (revenire la medie)

Interpretare

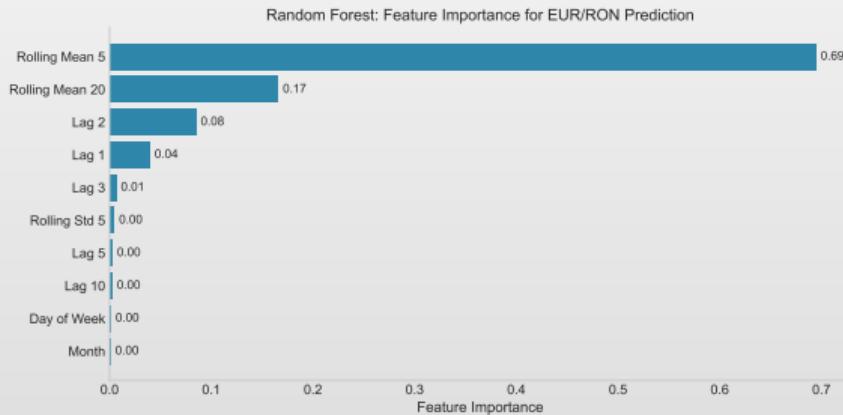
- Randamentele EUR/RON sunt **staționare** ($p\text{-value} < 0.05$)
- $H \approx 0.47 < 0.5$: ușoară tendință de revenire la medie
- $d \approx 0$: **memorie scurtă** – ARMA poate fi suficient
- Totuși, **volatilitatea** poate avea memorie lungă!



Random Forest: importanță features

Interpretare

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



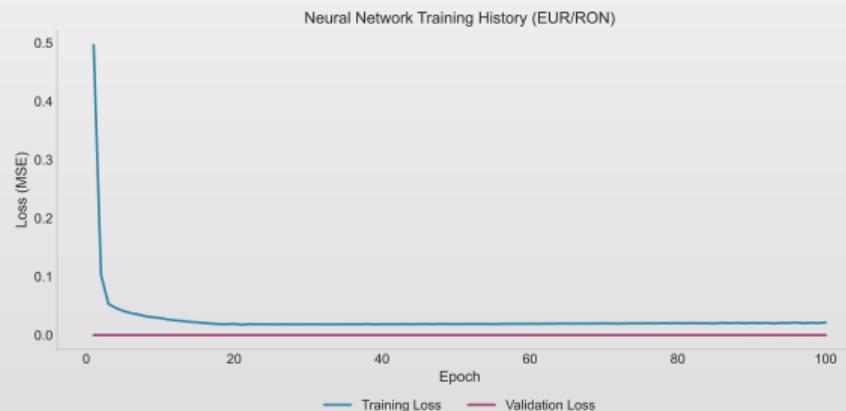
 TSA_ch8_case_feature_importance



LSTM: curba de învățare

Interpretare

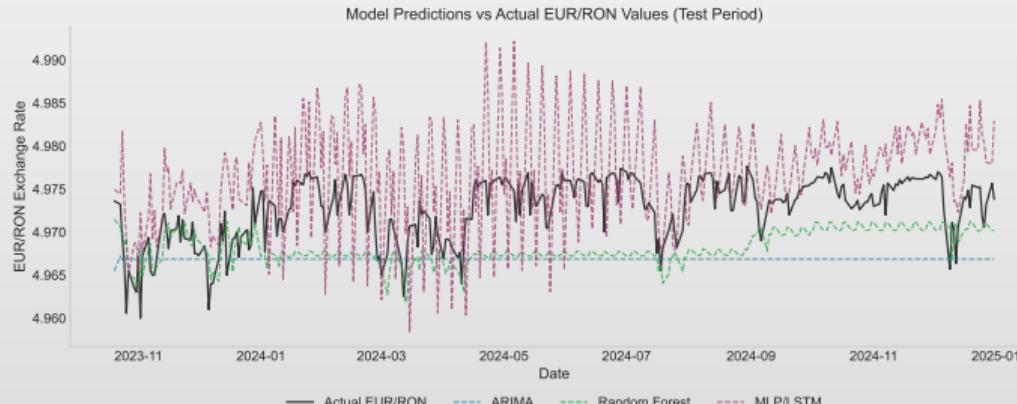
- **Date:** Cursul EUR/RON (Yahoo Finance, 2019–2025) — Rețea Neuronală (100 epoci, MSE – Mean Squared Error – loss)
- **Loss Training:** Scade rapid în primele epoci, apoi se stabilizează
- **Loss Validation:** Urmărește training loss — nu avem overfitting sever



Vizualizare: predicții vs valori reale

Interpretare

- Date: Perioada de test EUR/RON — predicțiile ARIMA, Random Forest, MLP/LSTM vs valori reale
- Toate modelele captează tiparul general
- Niciun model nu prezice perfect vârfurile de volatilitate
- Aceasta reflectă **eficiența pieței și limitele predicției** pentru serii financiare



Comparație: Rezultate pe EUR/RON

Model	RMSE	MAE	Timp (s)	Interpretabil?
ARIMA(1,1,1)	0.0069	0.0062	0.08	Da
Random Forest	0.0057	0.0050	0.51	Da (features)
MLP (Multi-Layer Perceptron)/LSTM	0.0071	0.0059	0.47	Nu

Concluzii

- Pentru EUR/RON, diferențele sunt **mici** – piața este eficientă
- Random Forest oferă cel mai bun compromis **acuratețe/interpretabilitate**
- LSTM are cost computațional mare pentru câștig marginal
- ARIMA rămâne o alegere solidă pentru **baseline**



Comparație modele: metrii de performanță

Interpretare

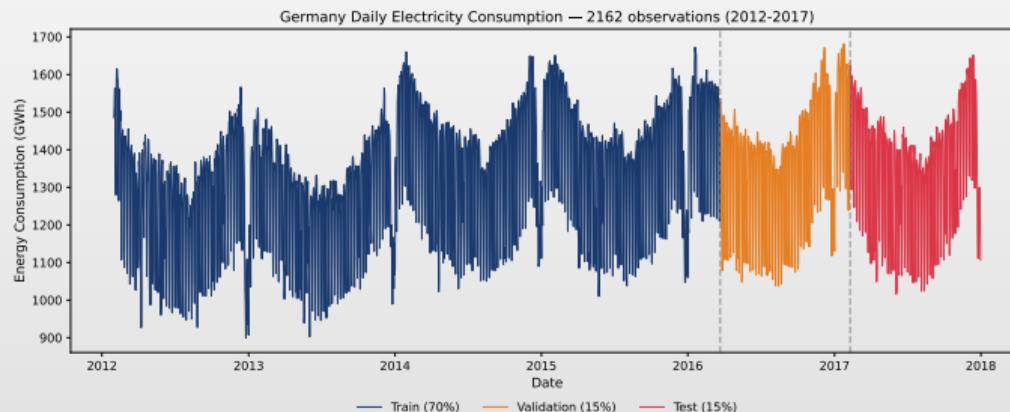
- **Date:** Cursul EUR/RON (Yahoo Finance, 2019–2025) — ARIMA vs RF vs MLP/LSTM
- **Stânga:** Metrici de eroare (mai mic = mai bine) — RF obține cel mai mic RMSE și MAE
- **Dreapta:** Timp de antrenare (scală log) — modelele ML necesită mai multe resurse
- **Trade-off:** Acuratețe ușor mai bună, dar cost computațional semnificativ mai mare



Studiu de caz: Prezentarea datelor

Interpretare

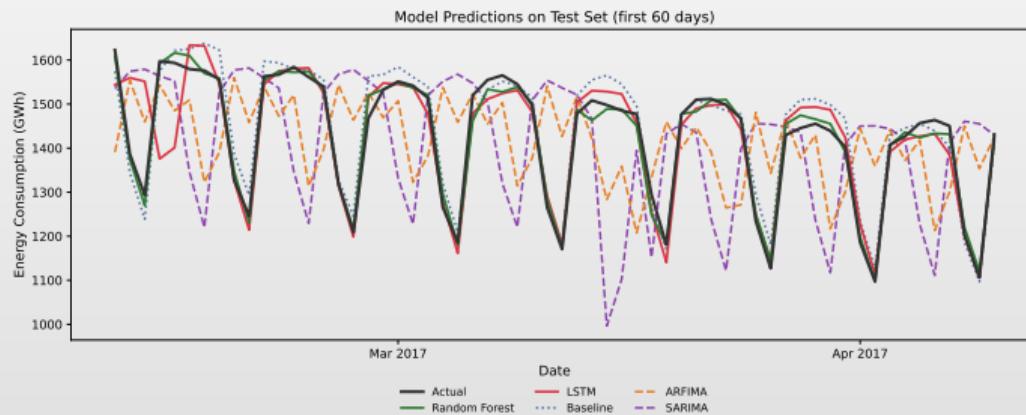
Train: 1513 obs (70%) **Validare:** 324 obs (15%) **Test:** 325 obs (15%)



 **TSA_ch8_data_split**

Studiu de caz: Predicții ale modelelor

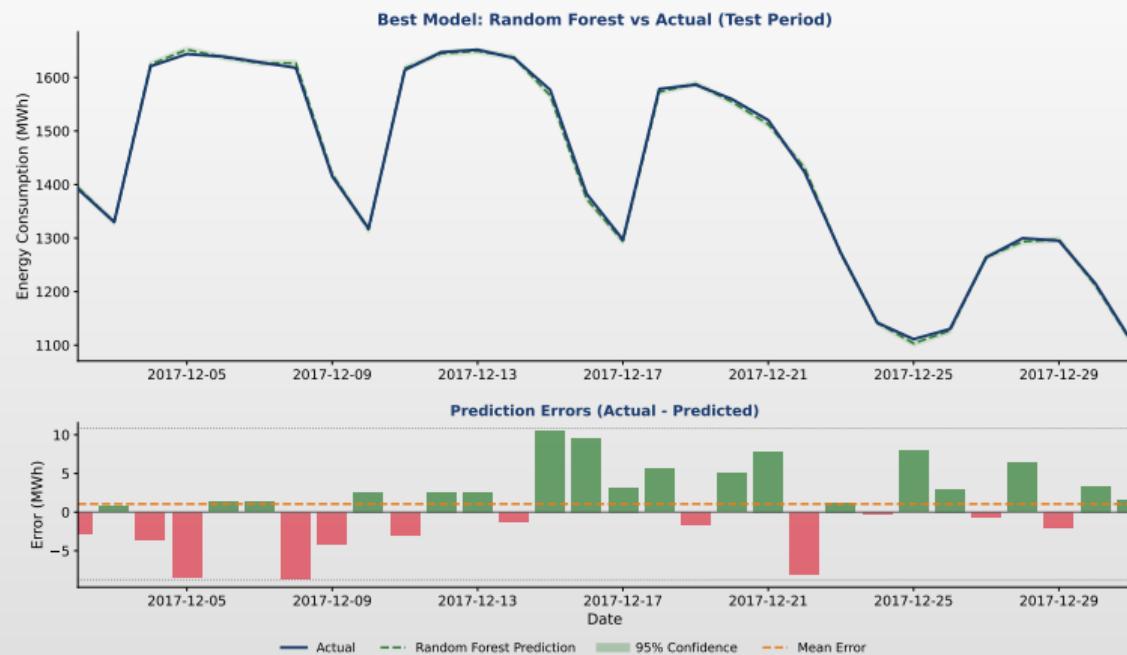
Rang	Model	MAPE	Interpretare
1	Random Forest	2.2%	Cel mai bun: captează tipare neliniare
2	LSTM	3.3%	Bun, necesită mai multe date
3	Baseline	3.9%	Simplu dar competitiv
4	ARFIMA	12.3%	Memoria lungă nu e suficientă
5	SARIMA	14.6%	Dificultăți cu tiparele



Q TSA_ch8_model_predictions



Studiu de caz: Performanța celui mai bun model



Q TSA_ch8_best_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 tipice (RMSE pe randamente)

- GARCH(1,1): 1.45 – cel mai bun pentru volatilitate
- ARFIMA pentru volatilitate: 1.52
- Random Forest: 1.48
- LSTM: 1.51



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. Începe cu ARIMA ca baseline (simplu, rapid, interpretabil)
2. Testează **memorie lungă** \succ ARFIMA dacă d semnificativ
3. Adaugă **features** \succ Random Forest pentru relații neliniare
4. Doar cu date multe (> 10.000 obs.) și resurse \succ LSTM

Lecții din studiile de caz

- EUR/RON:** Piață eficientă \succ diferențe mici între modele
- Bitcoin:** Volatilitatea beneficiază de memorie lungă (ARFIMA)
- Energie:** RF cel mai bun \succ tipare neliniare complexe
- Inflație:** Date puține \succ 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 rula fără erori și arăta profesional. *Asta nu înseamnă că e corect.*



Rezumat

Ce am învățat

- **ARFIMA:** Extinde ARIMA pentru memorie lungă (d fracționar)
- **Random Forest:** Ansamblu de arbori, relații neliniare, interpretabil
- **LSTM:** Deep learning pentru secvențe, dependențe complexe
- **Trade-offs:** Complexitate vs interpretabilitate vs date necesare

Recomandări practice

- Începe cu modele **simple** (ARIMA) ca baseline
- Folosește **Time Series CV** pentru evaluare corectă
- ML necesită **feature engineering** atent
- LSTM: doar cu **multe date și resurse computaționale**



Întrebarea 1

Întrebare

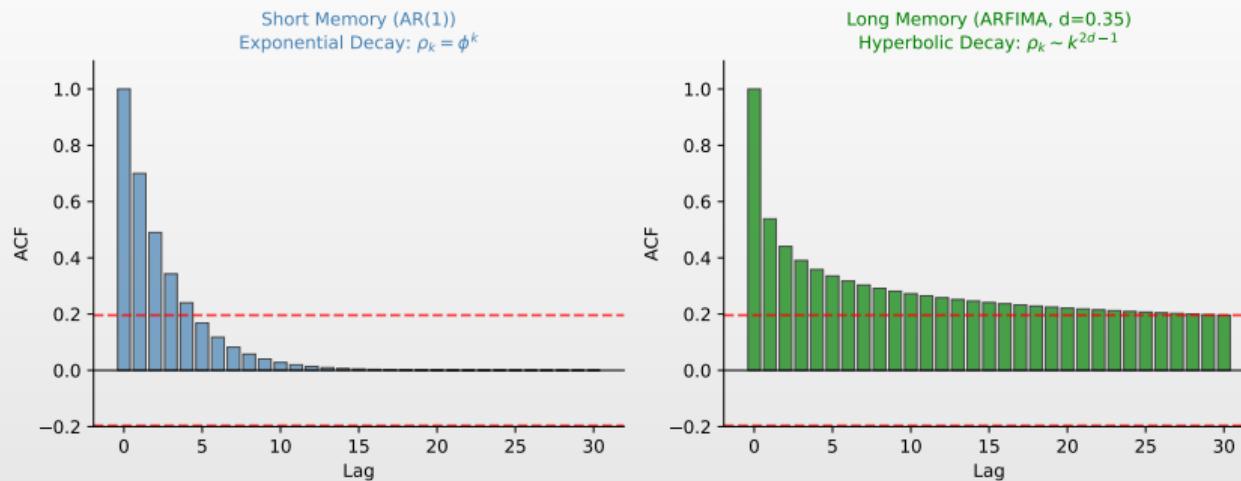
- Ce semnifică $d = 0.3$ într-un model ARFIMA?

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 îndepărтate au încă influență semnificativă



Întrebarea 2

Întrebare

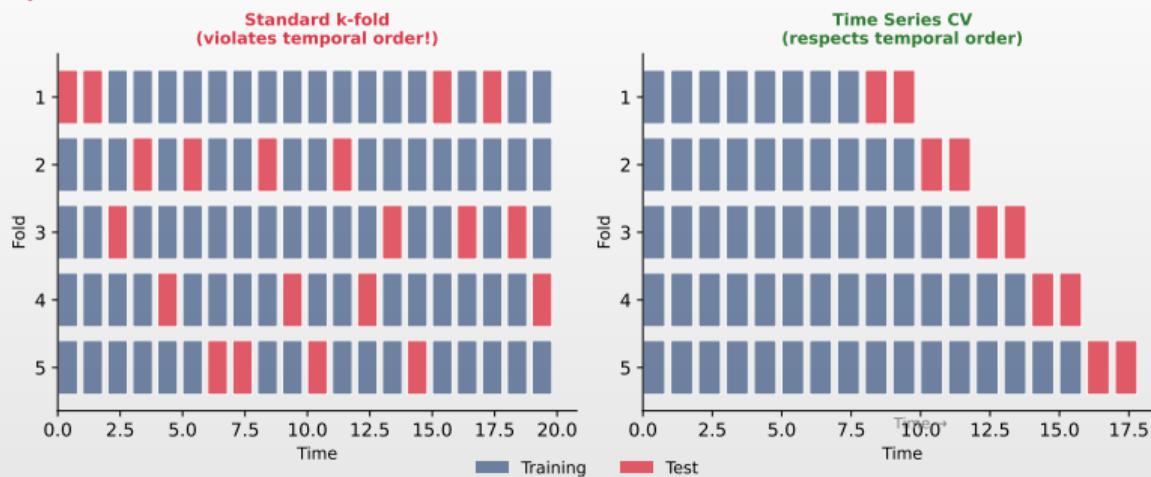
De ce trebuie să folosim Time Series CV în loc de k-fold standard?

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?

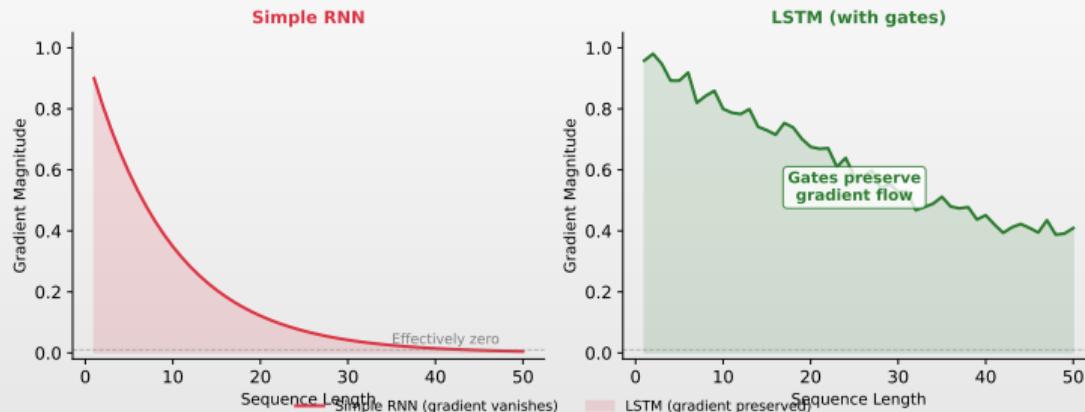
Variante de răspuns

- (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)

- Poartile forget, input și output ale LSTM controlează fluxul de informație, preservând gradientul pe secvențe lungi
- RNN simplu pierde semnalul gradientului după ~10–20 pași



Întrebarea 4

Întrebare

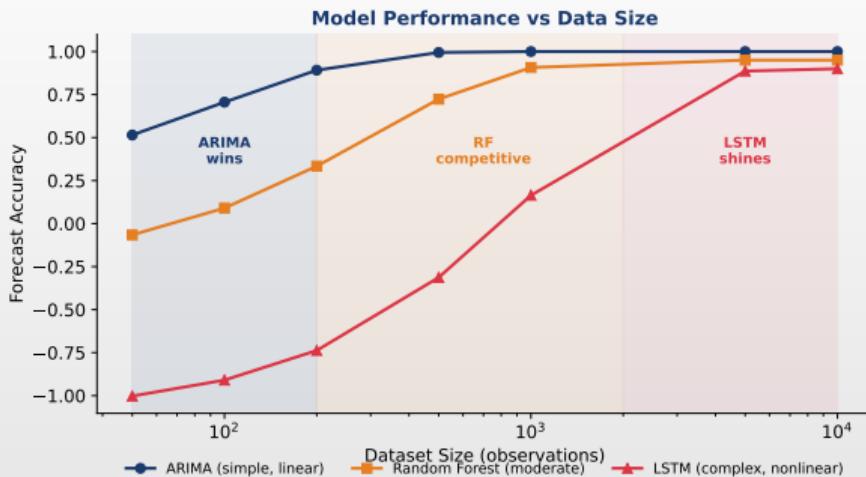
Aveți un set de date mic (100 observații) cu relații liniare. Ce model este cel mai potrivit?

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?

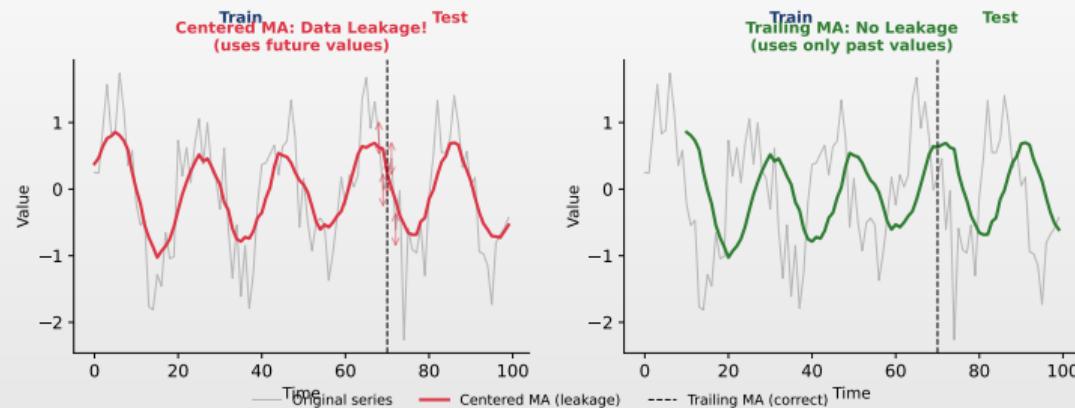
Variante de răspuns

- (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 (folosesc valori viitoare), k-fold standard (amestecă ordinea temporală)
- Calculul statisticilor pe tot setul de date înainte de split



Ce urmează?

Extensii și subiecte avansate

- Transformer** pentru serii de timp (Temporal Fusion Transformer)
- Prophet** (Facebook/Meta) pentru sezonalitate
- Neural Prophet:** Prophet + rețele neuronale
- Ensemble methods:** Combinarea mai multor modele
- 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

Modele cu prag și regim-switching

- 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 and the Business Cycle, *Econometrica*, 57(2), 357–384.
- Petropoulos, F., et al. (2022). Forecasting: Theory and Practice, *International Journal of Forecasting*, 38(3), 845–1054.

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/>

