



# Analiza și Prognoza Seriilor de Timp

## Seminar 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

## Cuprins Seminar

Test: Modele ARFIMA și Memorie Lungă

Test: Machine Learning pentru Serii de Timp

Test: Rețele LSTM

Întrebări Adevărat/Fals

Probleme Practice

Exercițiu cu asistență AI

Recapitulare

## Test 1: Exponentul Hurst

### Întrebare

O serie de timp are exponentul Hurst  $H = 0.8$ . Ce ne indică acest lucru?

- A) Seria este un mers aleator pur
- B) Seria are memorie lungă și este persistentă (trend-following)
- C) Seria este anti-persistentă (mean-reverting)
- D) Seria este staționară  $I(0)$

*Răspunsul pe slide-ul următor...*

## Test 1: Răspuns

Răspuns: B – Memorie lungă și persistență

### Interpretare Exponent Hurst:

- ☐  $H = 0.5$ : Mers aleator (fără memorie)
- ☐  $0.5 < H < 1$ : **Persistență** – tendința continuă
- ☐  $0 < H < 0.5$ : Anti-persistență – revenire la medie

Cu  $H = 0.8 > 0.5$ :

- ☐ Seria are **memorie lungă**
- ☐ Valorile mari tind să fie urmate de valori mari
- ☐ Autocorelațiile descresc lent (hiperbolic, nu exponențial)

## Test 2: Parametrul de Diferențiere Fraționară

### Întrebare

În modelul ARFIMA( $p, d, q$ ), parametrul  $d$  poate lua valori:

- A) Doar valori întregi (0, 1, 2, ...)
- B) Doar  $d = 0$  sau  $d = 1$
- C) Orice valoare reală, inclusiv fracționară
- D) Doar valori negative

*Răspunsul pe slide-ul următor...*

## Test 2: Răspuns

Răspuns: C – Orice valoare reală

Diferențiere fracționară:  $(1 - L)^d$  cu  $d \in \mathbb{R}$

Interpretare valori  $d$ :

- ☐  $d = 0$ : Seria staționară (ARMA)
- ☐  $0 < d < 0.5$ : Memorie lungă, staționară
- ☐  $d = 0.5$ : Granița staționar/nestaționară
- ☐  $0.5 < d < 1$ : Memorie lungă, nestaționară
- ☐  $d = 1$ : Diferențiere completă (ARIMA clasic)

Relația cu Hurst:  $d = H - 0.5$

## Test 3: Memoria Lungă în Serii Financiare

### Întrebare

În ce serie financiară este memoria lungă cel mai frecvent documentată?

- A) Prețurile acțiunilor
- B) Randamentele zilnice
- C) Volatilitatea (pătratul randamentelor)
- D) Volumul de tranzacționare

*Răspunsul pe slide-ul următor...*

## Test 3: Răspuns

### Răspuns: C – Volatilitatea

#### Fapte stilizate din finanțe:

- **Randamentele:** Aproximativ fără memorie ( $H \approx 0.5$ )
- **Volatilitatea:** Memorie lungă pronunțată ( $H \approx 0.7 - 0.9$ )

#### De ce?

- Volatility clustering: perioade turbulente urmate de perioade turbulente
- Persistența șocurilor în varianță
- FIGARCH: modelează explicit memoria lungă în volatilitate

Acest fapt stilizat este baza modelelor FIGARCH și HAR-RV.



## Test 4: Feature Engineering

### Întrebare

Pentru a aplica Random Forest pe serii de timp, trebuie să creăm:

- A) Variabile dummy pentru fiecare observație
- B) Caracteristici lag și statistici rolling
- C) Transformări Fourier ale seriei
- D) Numai prima diferență a seriei

*Răspunsul pe slide-ul următor...*

## Test 4: Răspuns

Răspuns: B – Caracteristici lag și statistici rolling

### Feature Engineering pentru Serii de Timp:

- ▣ **Lag features:**  $y_{t-1}, y_{t-2}, \dots, y_{t-k}$
- ▣ **Rolling statistics:**
  - ▶ Media mobilă:  $\bar{y}_{t,w}$
  - ▶ Deviația standard mobilă:  $\sigma_{t,w}$
  - ▶ Min/Max pe fereastră
- ▣ **Caracteristici calendaristice:** ziua săptămânii, luna, etc.

**Important:** Transformă problema de prognoză în problemă de regresie supervizată!

## Test 5: Validarea Încrucișată pentru Serii de Timp

### Întrebare

De ce *nu* putem folosi k-fold cross-validation standard pentru serii de timp?

- A) Este prea lent pentru serii lungi
- B) Încalcă ordinea temporală și cauzează data leakage
- C) Funcționează doar pentru clasificare
- D) Necesită prea multe date

*Răspunsul pe slide-ul următor...*

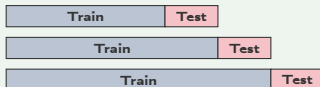
## Test 5: Răspuns

Răspuns: B – Încalcă ordinea temporală

**Problema cu k-fold standard:**

- Amestecă ordinea temporală a observațiilor
- Antrenează pe date din viitor, testează pe trecut
- **Data leakage**  $\Rightarrow$  performanță supraestimată

**Soluția: Time Series Split (Walk-Forward)**



## Test 6: Importanța Variabilelor în Random Forest

### Întrebare

Importanța variabilelor în Random Forest pentru serii de timp ne ajută să:

- A) Să eliminăm toate variabilele cu importanță mică
- B) Să identificăm care lag-uri și caracteristici sunt cele mai predictive
- C) Să determinăm cauzalitatea Granger
- D) Să calculăm intervalele de încredere

*Răspunsul pe slide-ul următor...*

## Test 6: Răspuns

Răspuns: B – Identifică caracteristicile predictive

### Utilizări ale importanței variabilelor:

- ☐ Înțelegerea structurii temporale
- ☐ Selectarea numărului optim de lag-uri
- ☐ Identificarea factorilor relevanți

### Atenție:

- ☐ Importanța *nu* implică cauzalitate
- ☐ Variabilele corelate pot împărtăși importanța
- ☐ Folosiți pentru interpretare, nu pentru inferență cauzală

## Test 7: Avantajul LSTM

### Întrebare

Care este principalul avantaj al LSTM față de RNN simple?

- A) Este mai rapidă la antrenare
- B) Rezolvă problema gradientilor care dispar/explodează
- C) Necesită mai puține date
- D) Este mai ușor de interpretat

*Răspunsul pe slide-ul următor...*

## Test 7: Răspuns

Răspuns: B – Rezolvă problema gradientilor

### Problema RNN Simple:

- ▣ Gradientii scad exponențial cu lungimea secvenței
- ▣ Nu pot învăța dependențe pe termen lung

### Soluția LSTM:

- ▣ **Cell state:** Autostradă pentru flux de informație
- ▣ **Forget gate:** Decide ce să uite
- ▣ **Input gate:** Decide ce să rețină
- ▣ **Output gate:** Decide ce să producă

Gradientii pot „curge” prin cell state fără degradare!



## Test 8: Pregătirea Datelor pentru LSTM

### Întrebare

Înainte de antrenarea LSTM, datele trebuie:

- A) Transformate logaritmice
- B) Normalizate/standardizate la intervalul  $[0,1]$  sau  $[-1,1]$
- C) Diferențiate de ordinul 2
- D) Convertite la numere întregi

*Răspunsul pe slide-ul următor...*

## Test 8: Răspuns

Răspuns: B – Normalizate/standardizate

**De ce normalizare?**

- ▣ Funcțiile de activare (sigmoid, tanh) funcționează în intervale limitate
- ▣ Convergență mai rapidă
- ▣ Stabilitate numerică

**Metode comune:**

- ▣ **Min-Max:**  $x' = \frac{x - x_{min}}{x_{max} - x_{min}} \succ [0, 1]$
- ▣ **Standard:**  $x' = \frac{x - \mu}{\sigma} \succ \text{media } 0, \text{ std } 1$

**Important:** Fit pe train, transform pe train+test!

## Test 9: Hiperparametri LSTM

### Întrebare

Care *nu* este un hiperparametru tipic pentru LSTM?

- A) Numărul de unități (neuroni) pe strat
- B) Lungimea secvenței de intrare
- C) Learning rate
- D) Parametrul de diferențiere  $d$

*Răspunsul pe slide-ul următor...*

## Test 9: Răspuns

Răspuns: D – Parametrul  $d$

$d$  este specific modelelor ARFIMA, nu LSTM!

### Hiperparametri LSTM:

- ▣ **Arhitectură:** nr. straturi, unități/strat
- ▣ **Secvență:** lungime lookback
- ▣ **Training:** learning rate, batch size, epochs
- ▣ **Regularizare:** dropout, early stopping

**Tuning:** Căutare pe grilă (*grid search*) sau optimizare bayesiană cu validare încrucișată temporală

## Întrebări Adevărat/Fals

Determinați dacă fiecare afirmație este Adevărată sau Falsă:

1. ARFIMA permite un parametru de diferențiere fracționar  $d \in (0, 0.5)$ .
2. Validarea încrucișată k-fold standard este adecvată pentru serii de timp.
3. Rețelele LSTM pot captura dependențe pe termen lung prin mecanismul de celulă.
4. Supraajustarea nu este o problemă pentru modelele de învățare automată aplicate pe serii de timp.
5. Random Forest este un model de tip bagging care agregă mai mulți arbori de decizie.
6. Temporal Fusion Transformer combină atenția cu variabile statice și cunoscute în viitor.

*Răspunsurile pe slide-ul următor...*

## Adevărat/Fals: Soluții

- 1. ARFIMA permite un parametru de diferențiere fracționar  $d \in (0, 0.5)$ .** ADEVĂRAT  
Diferențierea fracționară generalizează ARIMA:  $d \in (0, 0.5)$  înseamnă memorie lungă și staționaritate.
- 2. Validarea încrucișată k-fold standard este adecvată pentru serii de timp.** FALS  
K-fold standard amestecă ordinea temporală, cauzând data leakage. Folosiți TimeSeriesSplit.
- 3. Rețelele LSTM pot captura dependențe pe termen lung prin mecanismul de celulă.** ADEVĂRAT  
Cell state-ul LSTM permite fluxul de informație pe distanțe mari, rezolvând problema gradientilor care dispar.
- 4. Supraajustarea nu este o problemă pentru ML pe serii de timp.** FALS  
Supraajustarea este o problemă majoră, mai ales cu date limitate și multe features. Folosiți regularizare și early stopping.
- 5. Random Forest este un model de tip bagging care agregă mai mulți arbori de decizie.** ADEVĂRAT  
RF combină bootstrap aggregating (bagging) cu selecția aleatoare a features la fiecare nod.
- 6. Temporal Fusion Transformer combină atenția cu variabile statice și cunoscute în viitor.** ADEVĂRAT  
TFT folosește self-attention, variable selection networks și gating mechanisms pentru prognoze interpretabile.

## Problemă 1: Estimarea Exponentului Hurst

### Enunț

Fie seria zilnică de randamente Bitcoin. Estimați exponentul Hurst folosind metoda R/S și interpretați rezultatul.

### Pași de rezolvare:

1. Calculați media pe subintervale de diferite lungimi  $n$
2. Pentru fiecare  $n$ : calculați  $\text{Range}(R)$  și  $\text{Std}(S)$
3. Raportul  $R/S$  crește ca  $n^H$
4. Fit regresie:  $\log(R/S) = H \cdot \log(n) + c$

**Cod Python:** `nolds.hurst_rs(returns)`

 [TSA\\_ch8\\_hurst\\_interpretation](#)

## Problemă 1: Soluție și Interpretare

### Rezultat Tipic pentru Bitcoin

- Randamente:  $H \approx 0.45 - 0.55$  (aproape mers aleator)
- Volatilitate ( $|returns|$ ):  $H \approx 0.75 - 0.85$  (memorie lungă!)

### Interpretare:

- Randamentele sunt greu de prognozat (EMH aproximativ validă)
- Volatilitatea este predictibilă pe termen lung
- Implicații pentru managementul riscului și VaR

**Aplicație:** Modelele FIGARCH pot fi superioare GARCH standard

 TSA\_ch8\_eurron\_long\_memory



## Problemă 2: Random Forest pentru Prognoză

### Enunț

Construiți un model Random Forest pentru prognoza prețului Bitcoin la orizontul de 1 zi. Evaluați folosind TimeSeriesSplit.

### Pipeline:

1. **Feature engineering:**
  - ▶ Lag-uri:  $y_{t-1}, y_{t-2}, \dots, y_{t-7}$
  - ▶ Rolling mean/std: 7, 14, 30 zile
2. **Train/Test split:** TimeSeriesSplit(n\_splits=5)
3. **Model:** RandomForestRegressor(n\_estimators=100)
4. **Evaluate:** RMSE, MAE, Direction Accuracy

## Problemă 2: Cod și Rezultate

### Cod Python

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import TimeSeriesSplit

tscv = TimeSeriesSplit(n_splits=5)
rf = RandomForestRegressor(n_estimators=100)

for train_idx, test_idx in tscv.split(X):
    rf.fit(X[train_idx], y[train_idx])
    pred = rf.predict(X[test_idx])
```

### Rezultate tipice:

- ▣ Direction accuracy: 52-55% (puțin peste random)
- ▣ Feature importance: lag-1 și rolling\_std domină

## Problemă 3: LSTM pentru Serii de Timp

### Enunț

Implementați un model LSTM simplu pentru prognoza Bitcoin. Comparați cu Random Forest.

#### Arhitectură LSTM simplă:

1. Input: secvențe de 30 de zile
2. LSTM layer: 50 unități
3. Dense output: 1 neuron (prognoza)
4. Loss: MSE, Optimizer: Adam

#### Pași importanți:

- ▣ Normalizare MinMaxScaler
- ▣ Reshape la [samples, timesteps, features]
- ▣ Early stopping pentru a evita supraajustare

## Problemă 3: Cod LSTM

### Cod Keras/TensorFlow

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense

model = Sequential([
    LSTM(50, input_shape=(30, 1)),
    Dense(1)
])

model.compile(optimizer='adam', loss='mse')
model.fit(X_train, y_train, epochs=50,
        validation_split=0.1, verbose=0)
```

### Comparație tipică RF vs LSTM:

- RMSE similar (LSTM ușor mai bun pe date netede)
- RF: mai rapid, mai interpretabil
- LSTM: captează mai bine pattern-uri complexe

## Exercițiu AI: Gândire critică

### Prompt de testat în ChatGPT / Claude / Copilot

"Folosind yfinance, descarcă prețurile zilnice Ethereum (ETH-USD) din 2020-01-01 până în 2024-12-31 (aprox. 1.800 observații). Calculează randamentele logaritmice. Compară trei modele: ARIMA, XGBoost (cu lag-uri și variabile calendar ca features) și un LSTM simplu (2 straturi, lookback 30 zile). Folosește walk-forward validation cu fereastră de antrenare de 365 zile și orizont de 14 zile. Evaluează cu RMSE și direcțional accuracy. Vreau cod Python complet cu grafice comparative."

### Exercițiu:

1. Rulați prompt-ul într-un LLM la alegere și analizați critic răspunsul.
2. Cum construiește features pentru XGBoost? Lag-uri, variabile calendar, termeni Fourier?
3. LSTM-ul e structurat corect? (scalare MinMax, forma intrărilor 3D, fără data leakage)
4. Folosește walk-forward validation sau doar un singur split train/test?
5. Discută compromisul între interpretabilitate, complexitate computațională și acuratețe?

**Atenție:** Codul generat de AI poate rula fără erori și arăta profesional. *Asta nu înseamnă că e corect.*

## Rezumat: Când să Folosim Fiecare Metodă

### ARFIMA

- ▣ Serii cu memorie lungă (volatilitate, hidrologie)
- ▣ Când  $0 < d < 0.5$  este teoretic justificat
- ▣ Interpretabilitate statistică importantă

### Random Forest

- ▣ Relații neliniare între caracteristici
- ▣ Feature importance pentru înțelegere
- ▣ Date structurate, nu foarte lungi

### LSTM

- ▣ Secvențe lungi cu dependențe complexe
- ▣ Date suficiente pentru deep learning
- ▣ Pattern-uri dificil de capturat cu metode clasice

## Formule Cheie

### ARFIMA și Memorie Lungă

- Diferențiere fracționară:  $(1 - L)^d y_t = \varepsilon_t$
- Exponent Hurst:  $d = H - 0.5$
- ACF pentru memorie lungă:  $\rho(k) \sim k^{2d-1}$  (descreștere lentă)

### Machine Learning

- Feature lag:  $X_t = [y_{t-1}, y_{t-2}, \dots, y_{t-k}]$
- RMSE:  $\sqrt{\frac{1}{n} \sum (y_i - \hat{y}_i)^2}$
- Direction Accuracy:  $\frac{1}{n} \sum 1[\text{sign}(\Delta y) = \text{sign}(\Delta \hat{y})]$

### LSTM

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

- Cell update:  $C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$

# Vă mulțumesc!

Întrebări?

`danpele@ase.ro`