

**LAB7 – Prognozowanie BTC (1D) – raport**

Data: 2026-01-11

Celem ■wiczenia by■o zbudowanie modeli ucz■cych si■ przewidywania dziennych log-zwrotów BTC na podstawie okna historycznego (window=90) oraz cech kalendarzowych (harmonicznych) i statystyk krocz■cych. Porównano dwa podej■cia: model g■sty (Dense/MLP) oraz sekwencyjny model rekurencyjny (RNN: GRU/LSTM), z dodatkowym strojeniem hiperparametrów (KerasTuner).

## **1. Dane i przygotowanie**

Wejściem jest plik CSV z dziennymi danymi BTC (kolumny: „Open time”, „Close”). Z ceny tworzone są log-ceny i log-zwroty:

`log_price[t] = ln(price[t] + ε), ε = 1e-8`  
`r[t] = log price[t] - log price[t-1]`

Uczenie odbywa się na r[t] (to jest zmienne docelowa).

## 2. Inženieria cech

Dla każdej obserwacji  $r[t]$  budowane są cechy:  **$r[t]$**  (opóźniony log-zwrot jako cecha wejściowa) **harmoniczne roczne** z fazy w roku (sin/cos dla  $k=1..harm_k$ ) **harmoniczne miesięczne** z fazy w miesiącu (sin/cos dla  $k=1..harm_k$ ) opcjonalnie: **rolling mean** i **rolling std** z ostatnich rolling\_window zwrotów Wszystkie cechy są następnie skalowane (scaler.pkl) i muszą być 1:1 identyczne w treningu i inferencji.

### 3. Modele

Porównano: **Dense (MLP)** – model działający punktowo na wektorze cech (bez jawnej pamięci sekwencji). **RNN (GRU/LSTM)** – model przyjmujący sekwencję dłuższą window=90: (1, window, d). RNN był strojony KerasTunerem (kt\_max\_trials=10, kt\_epochs=6), a potem trenowany pełniej (epochs=25).

#### 4. Wyniki ilo ciowe (test)

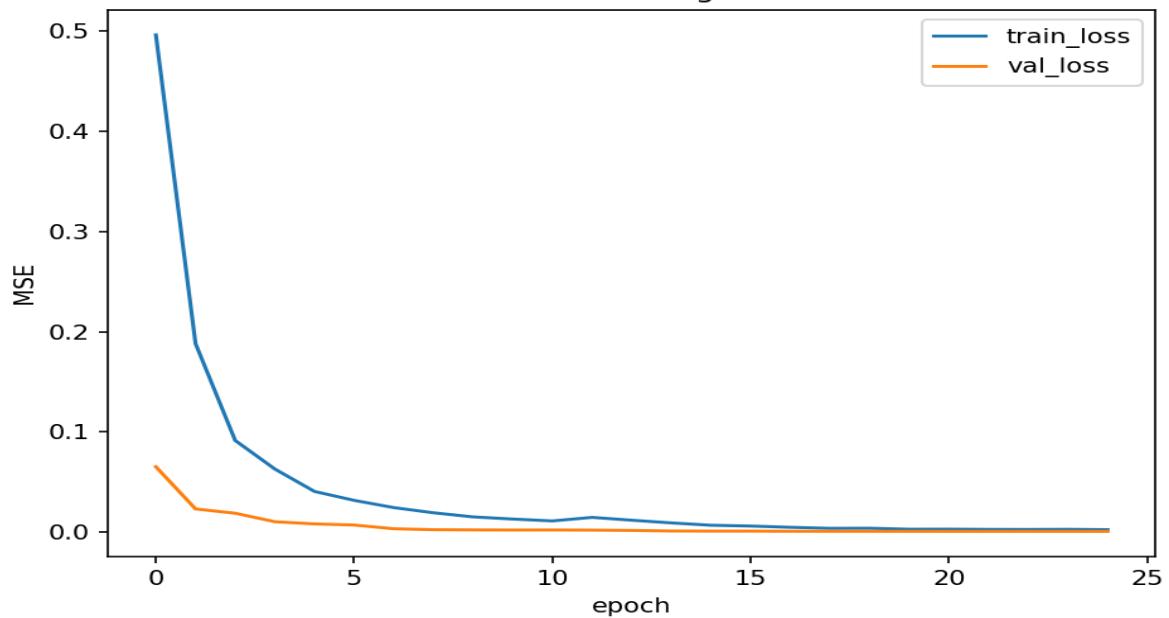
<b>Model</b>	<b>Test MSE</b>	<b>Test MAE</b>	<b>Corr(y, ■)</b>	<b>Directional acc. (sign)</b>
Dense	0.000602	0.018088	-0.035	0.489
RNN (LSTM tuned)	0.000505	0.015992	0.058	0.501

Dodatkowa obserwacja: predykcje RNN mają bardzo małe wariancje (odchylenie standardowe ≈ 1.08e-05), czyli model w praktyce zwraca prawie stałe wartości. Dla Dense odchylenie standardowe ≈ 0.0091. To wyjaśnia, dlaczego RNN może mieć niski MSE, ale na wykresie wygląda jak „linia prosta” – przewidywanie blisko średniej minimalizuje MSE, lecz nie zapie skoków zwrotów.

## 5. Wykresy

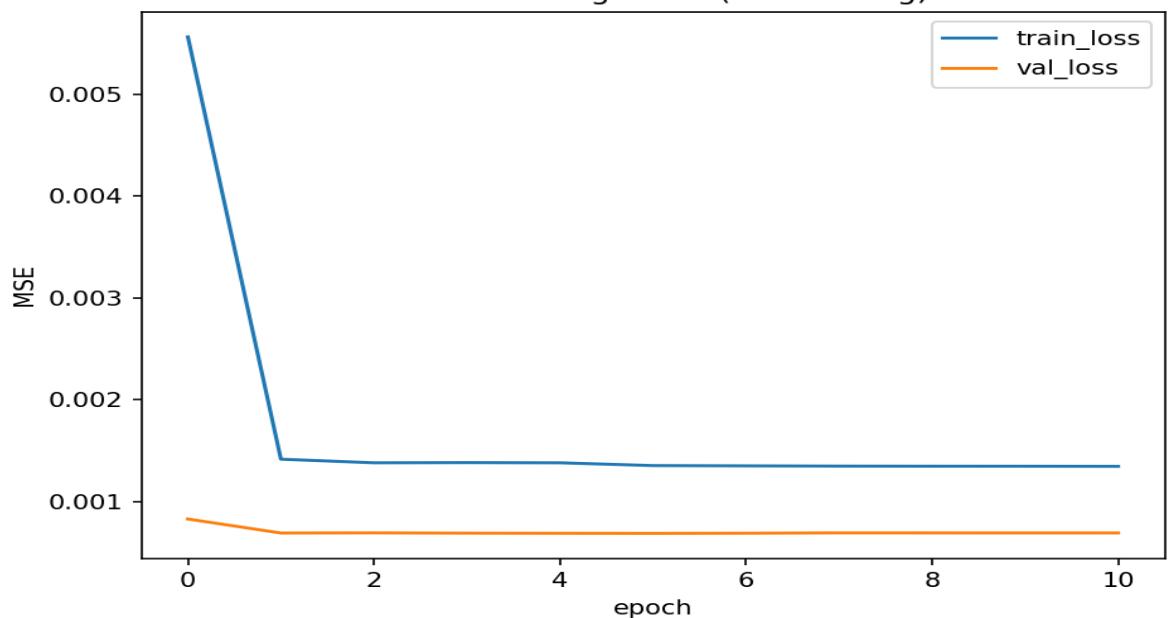
### *Krzywa uczenia: Dense*

Dense – learning curve

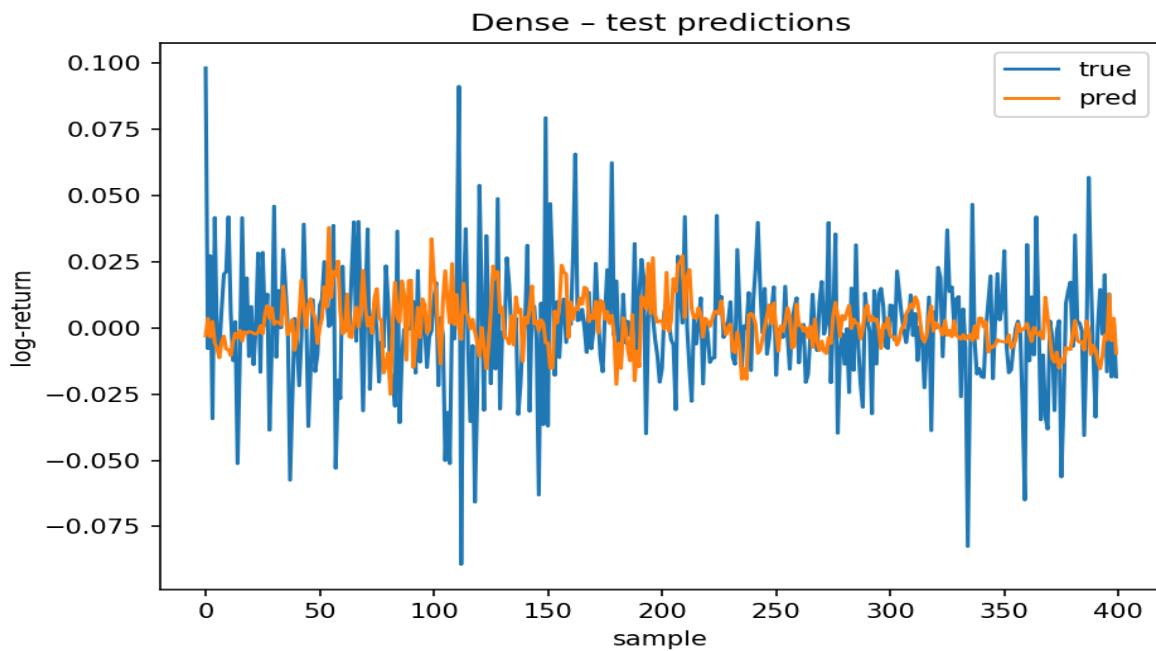


Krzywa uczenia: RNN (po tuningu)

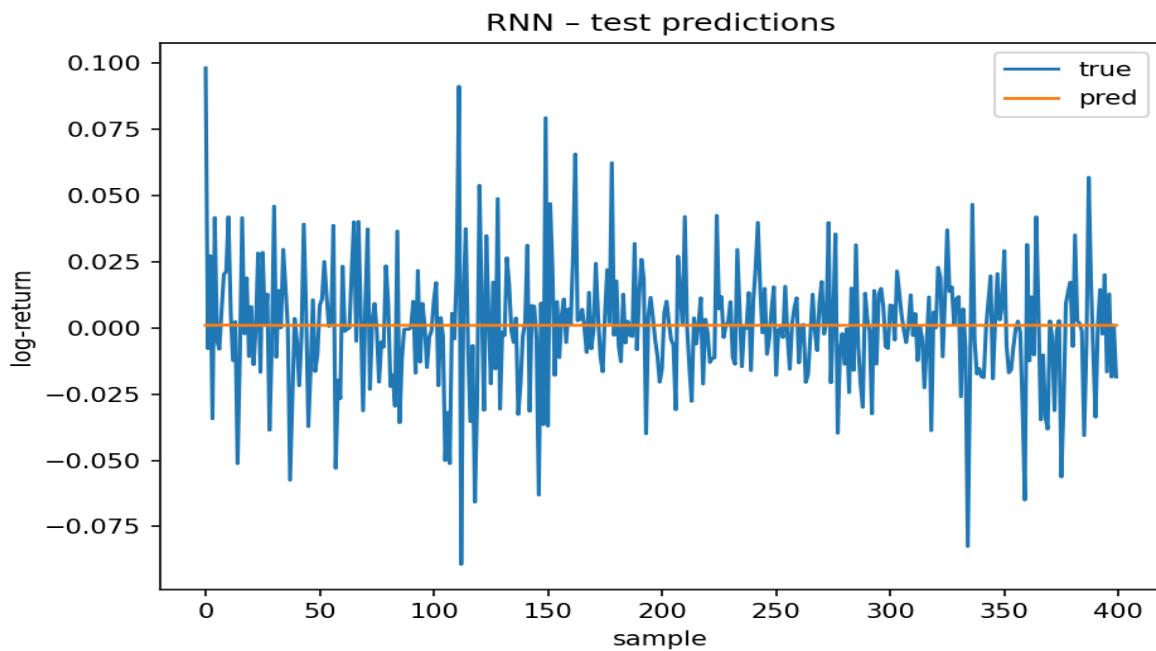
RNN – learning curve (after tuning)



Predykcje na teście: Dense (true vs pred)



*Predykcje na testcie: RNN (true vs pred)*



## 6. Interpretacja i wnioski

**Dlaczego Dense i RNN dają inne prognozy?** To inne klasy modeli (inna funkcja, inna inductive bias). RNN może „zadowolić się” przewidywaniem stalej wartości blisko średniej, co obniża MSE, ale daje mało użyteczne predykcje. **Dlaczego wyniki uruchomienia różniczą się między sobą?** Uczenie sieci jest stochastyczne (losowa inicializacja wag, losowe batchowanie), a na CPU dodatkowo cztery operacje może być niedeterministyczna (oneDNN / różna kolejność sumowania FP). To normalne. **Czy dodanie harmonicznych miesięcznych pomaga?** Nie da się uczciwie stwierdzić bez ablacji i (A/B): trening „z” i „bez” miesięcznych na tym samym podziale danych, tych samych seedach i kilku powtórzeniach. W samych logach widać, że metryki testowe potrafią się wahać między uruchomieniami – to może przykrywać mały efekt cech. **Co jest realnie najlepszym kryterium?** Dla finansów sama MSE na log-zwrocie bywa mylna. Warto rapportować też korelację, trafność kierunku oraz test strategii (np. prosta strategia long/short z kosztami) – dopiero wtedy widać, czy model ma „signal”.

## 7. Reprodukowalność (opcjonalnie)

Aby zmniejszyć rozjazdy między uruchomieniami: ustaw seedy: **PYTHONHASHSEED**, **np.random.seed**, **tf.random.set\_seed** w TensorFlow: **tf.config.experimental.enable\_op\_determinism()** (jeśli dostępne) dla CPU: można wybrać czy oneDNN: **TF\_ENABLE\_ONEDNN\_OPTS=0** (kosztem wydajności)