

ДОДАТОК А

Код гібридної моделі Neural-SEIR

Лістинг А.1 – основний модуль гібридної моделі Neural-SEIR

```
import numpy as np
import torch
from torch import nn
from src.models.base import QUANTILE_LEVELS, Forecaster,
QuantileForecast
from src.models.baselines import normal_quantiles
from src.models.seir import _trailing, calibrate_seir

class _BetaNet(nn.Module):
    """Малий MLP: нормалізований стан (S/N, E/N, I/N) -> beta
    >= 0."""

    def __init__(self, hidden: int = 16):
        super().__init__()
        self.net = nn.Sequential(
            nn.Linear(3, hidden), nn.Tanh(),
            nn.Linear(hidden, 1), nn.Softplus(),
        )

    def forward(self, state_norm):
        return self.net(state_norm).squeeze(-1)

def _integrate_torch(state0, beta_net, *, sigma, gamma, N,
n_steps):
    """Диференційовний крок Ейлера; beta продукує мережа від
    поточного стану."""
    S, E, I, R = state0
    incidences, comps = [], []
    for _ in range(n_steps):
        state_norm = torch.stack([S, E, I]).unsqueeze(0) / N
        beta = beta_net(state_norm).squeeze(0)
        s2e = (beta * S * I / N).minimum(S) # min-кламп:
невід'ємність
        e2i = (sigma * E).minimum(E)
        i2r = (gamma * I).minimum(I)
        S = S - s2e; E = E + s2e - e2i
        I = I + e2i - i2r; R = R + i2r
        incidences.append(s2e)
        comps.append(torch.stack([S, E, I, R]))
    inc = torch.stack(incidences)
    comp = torch.stack(comps)
    return inc, comp, comp[-1]

def _profile_k(incidence, target):
    """Аналітично оптимальний масштаб k = <inc, target> /
    <inc, inc>."""
```

```

        denom = torch.dot(incidence, incidence)
        return torch.dot(incidence, target) / denom if denom > 0
    else torch.zeros(())

def _train_uode(beta_net, state0, target, *, sigma, gamma, N,
                epochs=300, lr=0.02, seed=0):
    """Навчити лише beta-мережу; траєкторний лос MSE(k*inc,
    target), Adam."""
    torch.manual_seed(seed)
    state0_t = torch.as_tensor(state0, dtype=torch.float32)
    target_t = torch.as_tensor(target, dtype=torch.float32)
    n = len(target_t)
    opt = torch.optim.Adam(beta_net.parameters(), lr=lr)
    beta_net.train()
    for _ in range(epochs):
        opt.zero_grad()
        inc, _, _ = _integrate_torch(state0_t, beta_net,
                                     sigma=sigma, gamma=gamma,
N=N, n_steps=n)
        k = _profile_k(inc, target_t)
        loss = torch.mean((k * inc - target_t) ** 2)
        loss.backward()
        opt.step()
    beta_net.eval()
    return beta_net

class NeuralSeirForecaster(Forecaster):
    """Гібрид Neural-SEIR -> квантильний прогноз (контракт
    Forecaster)."""

    name = "neural_seir"

    def __init__(self, N, *, horizon, fit_window=60,
n_members=5,
                epochs=300, lr=0.02, hidden=16, seed_base=0):
        self.N = float(N); self.horizon = horizon
        self.fit_window = fit_window; self.n_members =
n_members
        self.epochs = epochs; self.lr = lr
        self.hidden = hidden; self.seed_base = seed_base

    def fit(self, train_y):
        y = _trailing(train_y, self.fit_window)
        # warm-start механістики; sigma, gamma, стартовий стан
дали фіксовані
        (_, self.sigma_, self.gamma_), _, init_state, _, _ =
calibrate_seir(y, self.N)
        state0 = torch.as_tensor(init_state,
dtype=torch.float32)
        self.members_, self.member_k_, self.member_final_ =
[], [], []
        for m in range(self.n_members):
            #
ансамбль -> epistemic

```

```

        net = _BetaNet(hidden=self.hidden)
        _train_ude(net, state0, y, sigma=self.sigma_,
gamma=self.gamma_,
                        N=self.N, epochs=self.epochs,
lr=self.lr,
                        seed=self.seed_base + m)
        with torch.no_grad():
            inc, _, final = _integrate_torch(state0, net,
sigma=self.sigma_,

gamma=self.gamma_, N=self.N, n_steps=len(y))
            k = float(_profile_k(inc, torch.as_tensor(y,
dtype=torch.float32)))
            self.members_.append(net);
self.member_k_.append(k)
            self.member_final_.append(final)
            self.sigma_ale_ = self._aleatoric_sigma(state0, y)    #
резидуальна (aleatoric)
            return self

    def _rollout(self, net, state, k):
        inc, _, _ = _integrate_torch(state, net,
sigma=self.sigma_,
                                gamma=self.gamma_,
N=self.N, n_steps=self.horizon)
        return k * inc

    def predict(self, horizon):
        with torch.no_grad():
            paths = np.vstack([self._rollout(net, final,
k).numpy()
                                for net, final, k in
zip(self.members_,
                                self.member_final_,
self.member_k_)])
            mu = paths.mean(axis=0)
            sigma = np.sqrt(paths.std(axis=0) ** 2 +
self.sigma_ale_ ** 2)
            q = normal_quantiles(mu, sigma, QUANTILE_LEVELS)
            return QuantileForecast(np.clip(q, 0.0, None))    #
таргет невід'ємний

```

ДОДАТОК Б

Повні результати порівняння моделей за горизонтами

Таблиця Б.1 – Зважена інтервальна оцінка (WIS) для COVID-19 за горизонтами

Модель	1 тиж.	2 тиж.	3 тиж.	4 тиж.
naive	5 537	9 382	13 569	18 208
seasonal_naive_m52	124 466	140 186	155 351	170 270
arima	4 548	9 318	15 946	24 497
seir	8 133	16 475	26 666	38 499
stochastic_seir	18 424	25 272	35 602	45 992
lstm	6 833	13 700	21 662	28 439
gru	6 230	11 860	18 094	23 554
transformer	4 877	7 630	12 536	17 526
neural_seir	8 238	14 955	23 248	32 149

Таблиця Б.2 – Зважена інтервальна оцінка (WIS) для грипу за горизонтами

Модель	1 тиж.	2 тиж.	3 тиж.	4 тиж.
naive	0,133	0,245	0,351	0,417
seasonal_naive_m52	0,465	0,465	0,462	0,466
sarima52	0,100	0,184	0,277	0,347
seir	0,246	0,387	0,509	0,624
stochastic_seir	1,216	1,308	1,408	1,476
lstm	0,106	0,167	0,214	0,263
gru	0,113	0,192	0,247	0,283
transformer	0,120	0,204	0,265	0,295
neural_seir	0,196	0,328	0,434	0,518

Таблиця Б.3 – Масштабована абсолютна похибка для COVID-19 за горизонтами

Модель	1 тиж.	2 тиж.	3 тиж.	4 тиж.
naive	1,60	3,26	4,85	6,46
seasonal_naive_m52	4,41	4,38	4,41	4,53
arima	1,26	2,30	3,76	5,23
seir	1,97	3,61	5,71	7,96
stochastic_seir	3,60	4,92	6,85	8,94
lstm	1,99	3,48	5,11	6,65
gru	1,83	3,14	4,62	5,93
transformer	1,65	2,38	3,69	4,84
neural_seir	1,86	3,19	4,78	6,51

Таблиця Б.4 – Масштабована абсолютна похибка (MASE) для грипу
за горизонтами

Модель	1 тиж.	2 тиж.	3 тиж.	4 тиж.
naive	0,20	0,39	0,57	0,70
seasonal_naive_m52	0,53	0,54	0,52	0,53
sarima52	0,13	0,24	0,37	0,46
seir	0,41	0,58	0,72	0,83
stochastic_seir	1,37	1,48	1,57	1,66
lstm	0,17	0,24	0,32	0,41
gru	0,17	0,29	0,37	0,41
transformer	0,18	0,31	0,39	0,43
neural_seir	0,32	0,47	0,59	0,69

ДОДАТОК В

Відомість кваліфікаційної роботи

[illegible]