Guideline: Learning a Feature-Driven SDE for Stock Price Prediction (with Partial-Information Backtesting)

Prepared for: Project — Investing Automation

1. Executive Summary

This report describes a practical methodology to learn a feature-conditioned stochastic differential equation (SDE) with jumps for daily stock returns, a compact baseline neural network that maps an as-of information set to SDE parameters, and a partial-information backtesting protocol. The model outputs calibrated daily densities and multi-day price ranges via Monte-Carlo rollout, which can be consumed by a transaction-cost-aware rebalancer.

2. Modeling Overview

We model log-price $X_t=\log S_t$ with a state-dependent jump–diffusion:

$$
dX_t = \mu_\theta(\phi_t)\,dt + \sigma_\theta(\phi_t)\,dW_t + \sum_{k=1}^{N_t} Y_{t,k}, \quad N_t \sim \text{Poisson}\!\big(\lambda_\theta(\phi_t)\,dt\big), \; Y_{t,k}\sim F_\theta(\cdot \mid \phi_t).
$$

Here, ϕ_t collects features available as-of time t; μ,σ,λ,F are neural functions of ϕ_t.

Simulation occurs in log space; $S_t = e^{X_t}$ ensures positivity. This Itô–Lévy setup is valid under standard Lipschitz/linear-growth and jump integrability conditions and captures volatility clustering (σ), heavy tails/gaps (jumps), and regime dependence via features.

3. Information Set ϕ_t (Leakage-Safe)

All inputs must be timestamped and lagged as-of the decision cutoff (EOD or pre-open).

- Price/OHLCV: open/close returns, gaps, intraday range (Parkinson/Garman–Klass/RS), ATR(14), realized vol(20/60), volume z-scores.
- Technicals: momentum (5/20/60/120d), RSI(2/14), % above/below MA(50/200), cross flags.
- Events & Sentiment: earnings calendar (days to/since ER), EPS surprise, guidance tone, FinBERT news polarity, analyst revision momentum.
- Regime: FGI level & change, VIX level & change, credit/term-spread proxies.
- Valuation & Quality (lagged): PER, Fwd PER, PEG, P/B, EV/EBITDA, PS (sector-neutral ranks), ROE/ROIC, FCF margin, sales/EBIT growth, accruals.
- Microstructure & Constraints: size, ADV/float, short interest %, borrow cost proxy, sector/country one-hots.

4. Baseline Neural Network

Backbone: compact residual MLP suited for tabular finance with explicit missingness handling.

- Inputs: standardized continuous features (cross-sectional robust z-scores, winsorized), categorical embeddings (sector/country), missingness mask m concatenated to inputs.
- Stem: LayerNorm → Dense(256) → GELU → Dropout(0.1).
- Two residual blocks: Dense(256) → GELU → Dropout(0.1) → Dense(256); gated by $\sigma(W_g[\phi;m]+b_g)$; Add & Norm.

- Heads (domain-constrained outputs):
  - Drift $\mu(\phi) = \tanh(\text{linear})\cdot\mu_{max}$ (caps small daily drift).
  - Diffusion $\sigma^2(\phi)$ = softplus(linear)+ε.
  - Jump intensity $\lambda(\phi)$ = softplus(linear).
  - Jump size Y ~ Skew-t with parameters: location ξ (linear), scale ω = softplus+ε, dof ν = 2+softplus, skew α (linear).
- Optional regime gates: multiply μ by $g_\mu(\text{FGI},\text{VIX})\in(0,1]$, inflate σ,λ by $g_\sigma, g_\lambda \geq 1$.
- Regularization: L2 on μ-head, dropout 0.1, gradient-clip 1.0–2.0, early-stopping on validation NLL and coverage error.

5. Training Objective (Daily Conditional Likelihood)

Target: daily log-return $\Delta X_{i,t} = \log(S_{t+1}/S_t)$. Use a 0–1 jump mixture likelihood:

- No-jump: $\mathcal N(\mu\,\Delta t,\, \sigma^2 \Delta t)$.
- One-jump: convolution of Normal with skew-t; Poisson weights $w_0=e^{-\lambda \Delta t}$, $w_1=(\lambda \Delta t)e^{-\lambda \Delta t}$.

Loss: negative log-likelihood plus regularizers (drift shrinkage, smoothness/Jacobian penalty).
Optionally add calibration penalty for coverage on model-implied 5/95% quantiles.

6. Inference & Monte-Carlo Rollout

Single-day: evaluate mixture density and extract Q05/Q50/Q95 numerically.
Multi-day H: generate scenarios for future drivers (frozen, AR(1), bootstrap, or stress), then roll out Monte-Carlo paths in log space (Euler–Maruyama + Poisson jumps). Aggregate to price ranges and tail probabilities for portfolio use.

7. Algorithm A — Monte-Carlo SDE Rollout (daily step)

```
function SIMULATE_PATHS(S0, H, M, model, feature_builder, feature_forward):
    X0 = log(S0)
    paths = zeros(M, H+1); paths[:,0] = S0
    state = init_state(history_up_to_t)
    for m in 1..M:
        Xt = X0
        for h in 1..H:
            ϕ_t = feature_builder(state)                 # as-of features
            ϕ_t = feature_forward(ϕ_t, state, step=h)    # scenario for unknown drivers
            μ, σ², λ, θ_J = model(concat(ϕ_t, mask))
            ε ~ Normal(0,1);  N ~ Poisson(λ)
            J = sum_{k=1..N} SkewT(θ_J)
            ΔX = μ + sqrt(σ²) * ε + J
            Xt = Xt + ΔX
            S_t = exp(Xt)
            update_state(state, S_t, ϕ_t)
            paths[m,h] = S_t
    return paths
```

8. Algorithm B — Training (Purged & Embargoed Walk-Forward)

```
for fold in rolling_time_folds(T, purge=H, embargo=H):
    (train_idx, val_idx) = fold
    for epoch in 1..E:
        for batch in loader(train_idx):
            ϕ, ΔX, mask = batch.as_of_inputs()
            μ, σ², λ, θ_J = model(concat(ϕ, mask))
            nll = NLL(ΔX | μ, σ², λ, θ_J)
            reg = drift_shrink(μ_head) + smooth_penalty(model)
            loss = nll + reg
            backprop(loss); clip_grad_norm(); step()
        # Early stop on val NLL + coverage error
```

9. Algorithm C — Deployment-Style Forecast

```
function FORECAST(panel_t):
    for ticker in panel_t:
        ϕ_t, mask = build_as_of_features(ticker, cutoff="EOD")
        μ, σ², λ, θ_J = model(concat(ϕ_t, mask))
        density = mixture_density(μ, σ², λ, θ_J)
        (Q05, Q50, Q95) = numeric_quantiles(density)
        store_forecast(ticker, μ, σ², λ, Q05, Q50, Q95)
```

10. Partial-Information Backtesting

Define an information policy □ (EOD, pre-open, or sparse). Enforce as-of joins: at time τ, only data with timestamp $\le \tau$ are visible. Fundamentals propagate from posting time (not period end). Walk-forward CV uses purge (last H days removed before validation) and embargo (hold-out gap after validation) to prevent leakage from overlapping returns.

Backtest loop:
1) Build as-of features per ticker under policy □.
2) Forecast densities/quantiles; optionally simulate paths for scenarios.
3) Translate to portfolio decisions (e.g., mean-CVaR with turnover and sector caps).
4) Execute at next open/VWAP with costs; record P&L and risk.

Evaluate forecast coverage, interval width efficiency, PIT uniformity, and trading KPIs (net return, Sharpe/Sortino, CVaR, turnover, drawdown).

11. Repository Skeleton

```
sde-forecast/
  data/                # as-of loaders, policy □ enforcement
  features/            # OHLCV/technicals/events/sentiment/valuation builders
  model/               # nets.py (backbone + heads), losses.py (mixture NLL)
  simulate/            # rollout.py (Algorithm A), quantiles.py
  backtest/            # policy.py, runner.py (loop), costs.py, metrics.py
  config/              # base.yaml (hyperparams, horizons, policy, universe)
  ui/                  # dashboard.py (Gradio panel for ranges & diagnostics)
```

12. Hyperparameters (Starting Points)

- Optimizer: AdamW, lr 1e-3, weight decay 1e-4; batch size 8k–64k (panel).
- Epochs: 20–60 with early stopping (patience 5).
- Regularization: dropout 0.1; μ-head L2 1e-3; gradient clip 1.0.
- Targets: daily log-returns; optional winsorize targets at 4–6σ (stability).
- Quantile calibration: rolling residual quantiles to shift model-implied quantiles.

13. Pitfalls & Tips

- Drift identifiability is weak at daily freq → cap $|\mu|$ and regularize; let σ and jumps explain most variability.

- Around earnings/events → larger λ and fatter jump tails; consider sub-daily steps.
- Cross-name covariance for portfolio risk can be overlaid with a factor model at allocation time (the SDE is per-name).
- Time zones and posting lags are critical for honest backtests; define a crisp as-of cut.
- Train with random feature masking to immunize against feed outages or partial information.