Standard Operating Procedure (SOP)
Multi-Asset ML Trading Engine

1. Purpose
This SOP describes how to set up, run, and monitor the multi-asset machine learning (ML) trading engine. It covers environment preparation, running backtests, reviewing performance metrics, and basic troubleshooting.

2. Scope
This procedure applies to research and paper-trading usage of the engine. It is not financial advice and is not intended as a guarantee of profitability in live markets.

3. Prerequisites
- Python 3.10+ installed.
- Git installed (optional but recommended).
- Basic familiarity with the terminal / PowerShell.
- Internet access for downloading historical market data via yfinance.

4. Environment Setup
4.1. Clone or copy the project
- Place the project directory in a working folder, e.g. C:\Users\<user>\Desktop\stock

4.2. Create and activate a virtual environment
- Windows (PowerShell):

```
python -m venv venv
venv\Scripts\activate
```

4.3. Install dependencies
- From the project root directory, run:

```
pip install -r requirements.txt
```

  If there is no requirements.txt, install manually:

```
pip install yfinance pandas numpy scikit-learn matplotlib xgboost optuna
```

5. Configuration
The main configuration lives in config.py via the RunConfig dataclass. It controls:
- DataConfig: symbols, period, interval.
- MLConfig: task (classification/regression), model_type, horizon, n_splits, use_optuna.
- PortfolioConfig: start_cash, max_leverage, max_position_pct, top_k_assets, fees and slippage.
- RiskConfig: Kelly scaling and drawdown/vol targeting parameters (used by simulator logic).

5.1. Set symbols and period
- Edit RunConfig in run_pipeline.main():
  - Choose a universe of symbols (e.g., large-cap tech + index).
  - Set period (e.g. "20y") and interval ("1d") to ensure enough pre-2015 data.

5.2. Choose ML settings
- task: "classification" or "regression"
- model_type: "rf", "gb", "xgb", or "ensemble"
- n_splits: number of time-series CV folds used in some training paths.
- use_optuna: set to False by default for speed.

6. Normal Operation

6.1. Run the multi-asset pipeline
- Ensure USE_MULTI_ASSET = True in run_pipeline.main().
- From the project root, run:

```
python run_pipeline.py
```

6.2. What the pipeline does
1) Downloads OHLCV data for all configured symbols (yfinance).
2) Builds technical features and multi-asset dataset.
3) Performs feature selection to drop noisy/leaky features.
4) Runs incremental walk-forward training:
   - Uses all data prior to 2015-01-01 as initial training (if available).
   - For each trading day from 2015 onward:
     * Trains an ML model on all data strictly before that day.
     * Predicts probability of up-move and/or expected return for that day.
5) Feeds the signals into the portfolio simulator:
   - Ranks assets by expected return.
   - Allocates capital across top-K assets subject to leverage and position limits.
   - Applies commissions, slippage, and spreads.
6) Prints performance metrics (Sharpe, Sortino, max drawdown, Calmar, CAGR).
7) Plots an equity curve; when live_plot is enabled, the curve updates during the backtest.

6.3. Running the single-asset demo (optional)
- Set USE_MULTI_ASSET = False in main().
- Run:

```
python run_pipeline.py
```

- The script will run the original single-asset pipeline across a list of symbols and plot their individual equity curves.

7. Monitoring and Interpreting Results

7.1. Console output
- Watch for:
  - Data download successes/failures.
  - Feature selection summary (number of features kept).
  - Walk-forward training logs (dates and window sizes, if enabled).
  - Final performance metrics printed by the simulator.

7.2. Equity curve
- Confirm that the equity curve:
  - Shows realistic drawdowns in adverse regimes (e.g., 2021–2022).
  - Exhibits recovery and new highs without being a perfectly smooth line.
- Large, fast gains with minimal drawdown may indicate remaining leakage issues.

7.3. Metrics
- Sharpe: risk-adjusted return based on total volatility.
- Sortino: risk-adjusted return based on downside volatility only.
- Max drawdown: worst peak-to-trough loss; large values (e.g., -40%) imply high risk.
- Calmar: CAGR divided by max drawdown; higher is better.
- CAGR: compounded annual growth rate over the backtest period.

8. Parameter Adjustments

8.1. Risk and allocation

- To reduce risk:
  - Lower max_leverage (e.g., from 2.0 to 1.2).
  - Lower max_position_pct (e.g., from 0.2 to 0.1).
  - Reduce top_k_assets or restrict to more liquid symbols.
- To be more aggressive:
  - Increase max_leverage cautiously.
  - Increase max_position_pct and/or top_k_assets.

8.2. Model settings

- Start with "rf" or "ensemble" for model_type.
- If training time is too slow, disable Optuna and reduce n_estimators.
- Experiment with "regression" task and ranking purely by expected return to compare against the classification-based approach.

9. Troubleshooting

9.1. yfinance download issues

- Symptom: errors indicating failures for specific tickers.
- Action:
  - Remove problematic symbols from the configuration.
  - Confirm ticker correctness (e.g., "^GSPC" for S&P 500).

9.2. Extremely good or unrealistic backtest

- Symptom: near-monotonic equity growth, very high Sharpe, minimal drawdown.
- Possible cause: data leakage (future-looking features or misaligned labels).
- Action:
  - Ensure only past data is used to train models for each test day.
  - Re-run feature selection and confirm no 'future_' columns are used as inputs.

9.3. Very slow runtime

- Walk-forward training with many symbols and long periods can be compute-heavy.
- Actions:
  - Reduce universe size (fewer symbols).
  - Shorten period (e.g., last 10y instead of 20y).
  - Switch model_type to "rf" and reduce n_estimators.
  - Temporarily disable Optuna or other heavy tuning.

10. Change Management

- Keep code changes under version control (e.g., git branches).
- When changing model types, features, or risk rules:
  - Run a fresh backtest.
  - Save and label results (config + performance) for comparison.

11. Safety and Disclaimers

- This system is for research and paper-trading.
- Historical backtests do not guarantee future performance.
- Real capital deployment should be preceded by extensive paper-trading and risk review.