1. Data Preprocessing (Train_Labeling.py)
  | - Reads OHLCV data
  | - Computes **Trend Label** (Bullish = 1, Bearish = 0)
  | - Computes **Volatility Label** (Stable = 0, Medium = 1, High = 2)
  | - Saves labeled dataset
  ▼
2. Trading Environment (TradingEnv)
  | - Loads labeled data
  | - Cleans & normalizes OHLCV + technical indicators
  | - Provides market state (window of past n steps)
  | - Executes buy/sell/hold actions
  | - Returns rewards & next state
  ▼
3. Macro-Agent (Strategy Selector)
  | - Inputs: Market state (processed OHLCV + technical indicators)
  | - Deep Neural Network (128 → 128 → 3)
  | - Outputs: Selects Macro-Strategy (Uptrend, Downtrend, Sideways)
  |
  ▼
4.Micro-Agents (Execution)
  ├─ **Uptrend Micro-Agent**
  | - Takes state as input
  | - Predicts Buy / Sell / Hold
  | - Uses a deep neural network (64 → 3)
  |
  ├─ **Downtrend Micro-Agent**
  | - Takes state as input
  | - Predicts Buy / Sell / Hold
  | - Uses a deep neural network (64 → 3)
  |
  ├─ **Sideways Micro-Agent**
  | - Takes state as input
  | - Predicts Buy / Sell / Hold
  | - Uses a deep neural network (64 → 3)
  |
  ▼
5.Replay Buffer (Experience Storage)
  | - Stores (state, macro_action, micro_action, reward, next_state, done)
  | - Used for experience replay during training
  |
  ▼
6. Training Process (Reinforcement Learning)
  | - **Macro-Agent Training**:
  | - Uses Deep Q-Learning (DQN)
  | - Loss: Mean Squared Error (MSE)
  | - Optimizer: Adam
  | - Updates target network every `N` steps
  |
  | - **Micro-Agent Training:**
  | - Trained separately for each market condition
  | - Uses Deep Q-Networks (DQN)
  | - Loss: MSE
  | - Optimizer: Adam
  |
  ▼

7.Evaluation & Performance Metrics during training
  | - Sharpe Ratio
  | - Sortino Ratio
  | - Max Drawdown
  | - Profit Factor
  | - Win Rate