# 目标与边界

* **目标**：用 1,000 CAD 小资金，搭建可复制、可审计、可自动执行的“日/周级量化策略 + AI 过滤”的系统；先追求**生存 + 纪律**，再追求收益。
* **风险承受**：你表示可承受 **最大回撤 50%**。本方案在配置上设定：
  + **软阈值**：-25%（触发降频/降仓/复盘）；
  + **硬阈值**：-50%（触发强制清仓 + 冷静期，人工复核后再开机）。
* **频率**：日/周级，避免高频；现金账户优先，规避 PDT 约束（实际开户以你为准）。
* **执行市场：以**美元计价的美国指数/行业 ETF\*\* 为主（SPY/QQQ/IWM 等），必要时以美债/超短债 ETF 做防守；个股仅作辅助。

# 项目目录骨架（repo 结构）

quant-ai-1k/  
├─ README.md  
├─ pyproject.toml # 依赖与打包（或 requirements.txt）  
├─ .env.example # 券商与通知密钥模板（勿上传真实密钥）  
├─ config/  
│ ├─ universe.yml # 标的池与基础权重/黑名单  
│ ├─ risk.yml # 风控硬阈值与仓位上限（含你的50%DD设定）  
│ ├─ data.yml # 数据源、频率、缓存策略  
│ ├─ model.yml # AI/ML 模型与特征配置（XGBoost/TCN等）  
│ ├─ execution.yml # 下单路由、撮合参数、时段、重试策略  
│ └─ backtest.yml # 回测窗口、滑点/佣金、walk-forward设置  
├─ data/  
│ ├─ raw/ # 原始数据（csv/parquet）  
│ ├─ processed/ # 特征工程后数据  
│ └─ cache/ # 模型/指标缓存  
├─ notebooks/ # 研究草稿（Jupyter, 可选）  
├─ src/  
│ ├─ core/  
│ │ ├─ data\_loader.py # 数据拉取&清洗（含T+1约束列）  
│ │ ├─ features.py # 技术指标、因子、目标构造  
│ │ ├─ model.py # 模型训练/推理接口（XGB/CatBoost/TCN）  
│ │ ├─ signal.py # 规则信号 + AI 过滤合成  
│ │ ├─ risk.py # 风控器：单笔/单日/回撤/冷却期/黑名单  
│ │ ├─ costs.py # 佣金/滑点/汇兑成本模型  
│ │ ├─ portfolio.py # 头寸计算与再平衡  
│ │ └─ utils.py # 公共工具（时间、日志、校验）  
│ ├─ exec/  
│ │ ├─ broker\_ibkr.py # ib\_insync 封装：下单、查询、对账  
│ │ └─ scheduler.py # 调度器（cron入口）与重试  
│ ├─ monitor/  
│ │ ├─ logger.py # 统一日志（CSV + 控制台）  
│ │ ├─ metrics.py # 绩效指标/Prometheus挂钩（可选）  
│ │ └─ alerts.py # 邮件/Telegram 通知  
│ └─ backtest/  
│ ├─ engine.py # 向量化回测（vectorbt/backtrader封装）  
│ └─ walk\_forward.py # 滚动训练与验证  
├─ scripts/  
│ ├─ fetch\_data.py # 拉数据/更新缓存  
│ ├─ train.py # 训练模型并固化（保存到 models/）  
│ ├─ backtest.py # 成本感知回测 + 报表  
│ ├─ run\_paper.py # 连接 Paper 账户跑影子单  
│ └─ run\_live.py # 实盘：信号→风控→下单→对账  
└─ models/  
 ├─ latest.pkl # 训练后模型  
 └─ versioned/ # 版本化权重

# 关键配置模板（节选）

## config/risk.yml

account\_ccy: USD  
max\_drawdown\_hard: -0.50  
max\_drawdown\_soft: -0.25  
position\_max\_gross: 1.00  
position\_max\_per\_asset: 0.20  
risk\_per\_trade: 0.03  
daily\_loss\_limit: -0.05  
cooldown\_days\_after\_hit: 3  
slippage\_bps: 10 # 初值，实盘校正  
commission\_per\_share: 0.005 # IBKR US 典型阶梯，min $1/订单（按实际账户调）  
stop\_mode: atr  
stop\_atr\_mult: 2.0  
takeprofit\_rr: 1.2  
```yaml  
account\_ccy: CAD  
max\_drawdown\_hard: -0.50 # 触发强平+冷却  
max\_drawdown\_soft: -0.25 # 触发降频/降仓/复盘  
position\_max\_gross: 1.00 # 净多敞口上限 100%  
position\_max\_per\_asset: 0.20 # 单标不超过 20%  
risk\_per\_trade: 0.03 # 单笔风险 3%（≈30 CAD），激进但可生存  
daily\_loss\_limit: -0.05 # 单日亏损阈值 -5% 触发冷却  
cooldown\_days\_after\_hit: 3 # 触发后冷却 3 个交易日  
slippage\_bps: 10 # 回测默认滑点 10bps，可校准  
commission\_per\_share: 0.01 # 示例，按实际券商费率调整  
stop\_mode: atr # 止损按 ATR 计算  
stop\_atr\_mult: 2.0 # 2x ATR 硬止损  
takeprofit\_rr: 1.2 # 盈亏比目标 1.2:1（可选）

## config/universe.yml（已改为 USD ETF 示例）

etfs:  
 - symbol: SPY # S&P 500  
 - symbol: QQQ # Nasdaq-100  
 - symbol: IWM # Russell 2000  
 - symbol: XLK # 科技板块  
 - symbol: XLE # 能源板块  
 - symbol: XLV # 医疗保健  
 - symbol: TLT # 20+Y 美债（防守/对冲）  
 - symbol: SHY # 1-3Y 美债/现金替代  
cash\_proxy: BIL # 超短国债ETF作现金近似（用于回测基线）  
blacklist: []  
rebalance\_freq: weekly  
base\_weights: equal  
```yaml  
etfs:  
 - symbol: XIU.TO # S&P/TSX 60 (CAD)  
 - symbol: XSP.TO # S&P 500 CAD对冲  
 - symbol: XQQ.TO # 纳指100 (CAD)  
 - symbol: XIC.TO # 全市场 (CAD)  
 - symbol: XEG.TO # 能源 (CAD)  
 - symbol: ZFV.TO # 短债/防守 (CAD)  
blacklist: []  
rebalance\_freq: weekly # 也可 monthly  
base\_weights: equal

## config/model.yml（升级为“榨干 4090”的深度时序配置；Mac 先用 CPU/MPS 预跑）

model: tcn\_torch # 可选: tcn\_torch | tft\_torch | transformer\_small  
features:  
 - returns\_5d  
 - returns\_20d  
 - rsi\_14  
 - macd  
 - vol\_20d  
 - atr\_14  
 - trend\_100\_over\_200  
label: next\_5d\_excess\_up  
threshold: 0.60 # 用于推理时的开仓阈值（分类输出）  
train:  
 framework: pytorch\_lightning  
 precision: 16-mixed # AMP 混合精度（4090上启用）  
 accelerator: auto # Mac 上为 cpu/mps；4090 机为 gpu  
 devices: 1  
 max\_epochs: 100  
 early\_stopping\_patience: 10  
 batch\_size: 512  
 lr: 1e-3  
 optimizer: adamw  
 scheduler: cosine  
 num\_workers: 4  
hpo: # 可选：用 optuna 进行超参搜索  
 enable: true  
 n\_trials: 30  
 params:  
 lr: [1e-4, 1e-3]  
 batch\_size: [256, 512, 1024]  
 dropout: [0.0, 0.1, 0.2]  
```yaml  
model: xgboost\_classifier  
features:  
 - returns\_5d  
 - returns\_20d  
 - rsi\_14  
 - macd  
 - vol\_20d  
 - atr\_14  
 - trend\_100\_over\_200 # 规则过滤相关特征  
label: next\_5d\_excess\_up # 下期5日是否跑赢现金/基准  
threshold: 0.60 # 胜率≥60% 才放行  
train:  
 test\_size: 0.2  
 cv: time\_series\_split  
 class\_weight: balanced

## config/backtest.yml

start: 2016-01-01  
end: 2025-09-30  
cash: 1000  
include\_costs: true  
apply\_t\_plus\_one: true  
walk\_forward:  
 window\_train\_days: 730  
 window\_test\_days: 90  
 step\_days: 30

# 规则信号 + AI 过滤（逻辑示意）

1. **规则层**（必须全部满足才进入候选）：
   * MA100 > MA200（长期趋势向上）；
   * 波动过滤：近期年化波动 ≤ 目标阈值（或用ATR/价格通道）；
   * 成本门槛：预估优势 < 2 × (佣金+滑点) → 放弃。
2. **AI 层**（二选一或并用）：
   * 胜率分类器 P(up\_next\_5d) ≥ 0.60 → 放行；
   * 尺度模型 size ∈ [0,1]，最终头寸 = min(size, per-asset cap)。
3. **风控层**：
   * 单笔风险 = position\_value × stop\_distance ≤ risk\_per\_trade × equity；
   * 触发 **日损/软DD/硬DD** → 降频/清仓/冷却；
   * 任何异常（下单失败、数据缺失）→ **停机 + 通知**。

# 关键脚本骨架（节选伪代码）

## src/core/signal.py

from .features import build\_features  
from .model import load\_model, infer\_proba  
from .costs import expected\_edge  
  
def generate\_signals(df, cfg\_model, cfg\_risk):  
 feats = build\_features(df)  
 # 规则过滤  
 long\_trend = feats["ma100"] > feats["ma200"]  
 candidates = df[long\_trend].copy()  
 # 预估优势需覆盖成本  
 candidates = candidates[expected\_edge(candidates) > 2\*candidates["est\_cost"]]  
 # AI 过滤  
 model = load\_model(cfg\_model)  
 proba = infer\_proba(model, feats.loc[candidates.index])  
 pass\_mask = proba >= cfg\_model["threshold"]  
 signals = candidates[pass\_mask]  
 return signals.assign(prob=proba[pass\_mask])

## src/core/risk.py

def position\_size(equity, atr, price, cfg\_risk):  
 stop\_dist = cfg\_risk["stop\_atr\_mult"] \* atr  
 risk\_cap = cfg\_risk["risk\_per\_trade"] \* equity  
 shares = max(0, int(risk\_cap / stop\_dist))  
 return shares  
  
class RiskManager:  
 def \_\_init\_\_(self, state, cfg):  
 self.state = state # 追踪权益、当日盈亏、回撤、冷却期等  
 self.cfg = cfg  
 def check(self, proposed\_orders):  
 # 检查日损、软/硬DD、单标上限、净多上限  
 # 若触发则下调规模或拒单  
 return filtered\_orders

## scripts/run\_live.py（主流程）

# 伪代码：信号→风控→下单→对账  
load\_env()  
cfg = load\_all\_configs()  
broker = IBKRClient(cfg["execution"]) # 基于 ib\_insync  
universe = load\_universe()  
prices = fetch\_realtime\_or\_close(universe)  
features = build\_features(prices)  
signals = generate\_signals(prices, cfg["model"], cfg["risk"])  
orders = risk\_manager.propose(signals, equity=broker.account\_equity())  
placed = broker.place\_orders(orders)  
reconcile\_and\_log(placed)  
alert\_if\_needed()

# 环境与依赖（先在 Mac 预研，再迁移到 4090 训练机）

## macOS（Apple Silicon 或 Intel）

* 包管理：安装 **Homebrew** 与 **miniforge**（conda）。
* Python 环境：conda create -n quant python=3.11 → conda activate quant。
* 科学栈：pip install pandas polars numpy scikit-learn xgboost catboost pyyaml loguru。
* 时序/回测：pip install vectorbtpro（或开源版 vectorbt）/backtrader（二选一）。
* PyTorch：安装 **MPS** 版本（pip install torch torchvision torchaudio，参考官方指引）。
* 其他：brew install ta-lib 后 pip install TA-Lib（如需），pip install ib-insync optuna pytorch-lightning。

## 4090 训练机（Ubuntu 22.04）

* 安装 **NVIDIA 驱动 + CUDA/cuDNN**，匹配版本的 **PyTorch (CUDA)**。
* 追加：pip install flash-attn（可选，视CUDA/SM兼容而定）、pip install optuna 用于 HPO。
* XGBoost GPU：pip install xgboost（官方轮子已支持 CUDA）。

实盘执行主要在 CPU 上；4090 主要用于训练/调参与较大 batch 的推理。

# USD 与外汇/资金流（IBKR 实操要点）

* **资金与币种**：账户可持多币种。若初始为 CAD，需在 IBKR 进行 **CAD→USD** 的外汇交易（CAD.USD 货币对，IDEALPRO）或使用自动货币转换；在成本模型中记录**汇兑点差与佣金**。
* **结算与PDT**：美股**T+1**结算；若使用**现金账户**可规避 PDT 标记但需等待资金结算再复用；若用**保证金账户**需严格控制日内交易次数以免触发 **PDT<25kUSD** 限制。
* **回测处理**：回测以 **USD** 为记账货币；若有跨币种资产，需以每日 **USDCAD** 汇率进行权益折算与成本叠加。

# 数据与成本建模

* **数据**：日频为主；保存为 Parquet，按日期/标的分区。
* **成本**：回测内置佣金 + 滑点（起步 10–20bps）；上线后用实盘成交对滑点回填校准。
* **T+1 约束**：现金账户回测必须启用资金占用/释放时序，避免“资金重复使用”的假象。

# 监控与审计

* **日志**：每笔订单/成交/PNL/触发的风控写入 CSV/SQLite。
* **可视化**：日终输出权益曲线、回撤、胜率、换手、费用占比。
* **报警**：日损、下单失败、数据断档、风控触发→邮件/Telegram。

# 推进计划（T+14 天上线，USD 版）

**D0**：在 **Mac** 完成环境搭建与仓库初始化，生成示例数据与报表模板。 **D1–D2**：拉取 USD ETF 日频数据（SPY/QQQ/IWM/XLK/XLE/XLV/TLT/SHY + 基准 BIL）；实现 features.py 与向量化回测雏形。 **D3–D4**：规则层（趋势/波动/成本门槛）跑通；做含成本的基础回测，输出权益/回撤/换手/费用占比。 **D5–D6**：在 Mac 上训练 **TCN/TFT** 小规模模型，接入 signal.py 胜率过滤；完成 **walk-forward**。 **D7**：在 IBKR 进行 **CAD→USD** 转换流程演练（Paper），验证记账与成本记录；完善 costs.py 的 FX 项。 **D8–D9**：迁移到 **4090**，开启 **AMP + HPO（optuna 30 trials）**，保存最优权重与推理管线。 **D10**：接入 broker\_ibkr.py（Paper）跑 **信号→风控→下单→对账** 全闭环。 **D11–D12**：影子周（Paper 全自动），修复异常、校准滑点/佣金/汇兑成本。 **D13–D14**：Go/No-Go 评审；若 Go，则以 **100–200 USD/笔** 极小仓位实盘。

# Go-Live 审核清单（通过才允许上实盘）

* 回测/影子周包含成本、T+1、失败重试、冷却期逻辑；
* 单笔风险 ≤ 3%；单标 ≤ 20%；总敞口 ≤ 100%；
* 日损/软DD/硬DD 能被自动识别并触发；
* 下单失败/拒单/超时均会 **停机 + 报警**；
* 配置/模型均有版本号，能回滚；
* 演练过“断电/断线/券商维护”的人工切换流程。

# 日常SOP（实盘）

1. **开盘前**：数据校验→生成当日候选→检查风控状态（是否处于冷却期）。
2. **开盘后**：在设定时段运行 run\_live.py；异常立停并报警。
3. **收盘后**：对账→落库→输出日报（权益、回撤、费用、触发次数）。
4. **每周**：参数/模型轻回顾（不频繁改动）；
5. **每月**：策略复盘与风险审计（是否接近软/硬阈值）。

# 注意事项与红线

* **遵守硬阈值**：达到 -50% **必须**强制停机清仓，至少 10 个交易日冷静期；
* **不要** 无上限加仓/马丁；
* **不要** 把 LLM 直接接入生产改阈值；
* **先 Paper 后实盘**，任何变更先影子运行一周；
* **备份**：每日配置/模型/日志快照。

# 下一步

* 我可以基于此骨架，按你选定的 ETF 池与券商，生成**可运行的最小代码模板**（含示例数据与一键回测脚本）。
* 告诉我你想用的券商（IBKR/其他）、标的池（若有偏好）、以及是否用 XGBoost 还是 TCN/小型TFT，我就把 features.py / model.py / backtest.py / run\_live.py 的**初版代码**填上。