美股选股面板：一个个人投资决策支持系统的设计、验证与迭代

A Personal Decision-Support System for US Equity Screening: Design, Validation, and Iteration

项目报告 · Thomas · 2026 年 9 月

系统形态：Cloudflare Worker 公开网站 · 约 620 行代码 · 服务端计算，前端仅渲染

数据源：TradingView Scanner API（37 个字段），以 FactSet 作为交叉验证基准

摘要

这份报告记录我在 2026 年间独立设计并持续迭代的一套美股筛选系统。系统每次运行扫描全美市值十五亿美元以上的上市公司，按资本回报率与现金流质量对每家公司评级，再结合价格位置与券商预期把它们分进九个操作档位，最终输出一张可直接执行的分层表。

报告的重点不在系统本身，而在造它的过程里暴露出来的东西。我原先以为自己的选股判断是有逻辑的，直到试图把它写成代码，才发现大量判断依赖的是我说不清楚的直觉，而其中一部分经不起检验。整个项目真正的产出是六次被记录下来的错误：一次浮点边界导致的信号静默丢失、一次行业分类规则的双向失效、两次财务口径不一致造成的伪数据、一次两层缓存的混淆，以及一次我自己的测试方法本身带有盲区的情况。最后这一次尤其值得单独写，因为它说明检验工具和被检验对象共享同一个假设时，检验会失效。

报告按以下顺序展开：问题的界定，为什么以投入资本回报率而非市盈率或利润率作为主判据，数据源的选型与验证，系统架构，分类规则的完整定义，六次错误的记录与修正，验证方法，多周期问题的处理，我自己投资认知发生的四处转变，以及系统明确不能解决的问题。附录给出完整参数表、字段清单和分档级联的伪代码。

关于工作方式的说明

这个系统是在与一个 AI 助手持续结对的方式下建成的，大部分代码由它在我的指导下写出。研究问题由我设定，面板分哪些区、每张表显示哪些列、分档规则按什么顺序判断，这些都由我决定。所有阈值来自我自己的交易经验而不是它的建议，涉及判断的地方也由我拍板，包括哪些公司需要人工覆盖、信号区的上界该放宽到多少、四小时周期要不要单独成区。

相反方向的纠正同样发生过，并且写进了正文而不是被略去。第七节记录的 Wingstop 归因错误是助手做出的：它看到一个同比数字就为它构造了解释，我拿另一个来源的相反结论去质疑，回查财报后确认助手是错的。它还曾把我从未设过的止损位写成「你的止损」，被我当场否掉。这两次都不是花

絮。前一次直接产生了第七节关于同比基期的那条结论，后一次说明工具会把自己的推断当成关于使用者的事实陈述，而使用者未必察觉。

我把这一点放在前面说明，是因为分工本身属于方法的一部分，而不是一条脚注。第 6.6 节的结论是检验工具与被检验对象共享假设时检验会失效，这条对人与工具的配合同样成立。本报告中的每一个阈值和规则都可以在源代码里回查。

目录

一、问题的提出

1.1 起点：一个无法复现的判断过程

这个项目的起点不是一个技术想法，而是一次自我怀疑。我买卖美股已经有一段时间，也自认为形成了一套判断标准：看公司质量，看估值，看图形位置。但当朋友问我某只票该不该买、我尝试说清楚理由的时候，说出来的往往是一串互相之间没有明确权重的理由，而且换一只票、换一个时间，我给出的理由结构就变了。

这让我意识到一个问题。如果一套判断标准在不同对象上会不自觉地调整，那它更像是对已有结论的事后修饰，而不是产生结论的依据。真正的标准应该有一个性质：写下来之后，别人拿去用，应该得到和我一样的结论；如果得不到，那么分歧一定能定位到某个具体的条件上。我当时的判断没有这个性质。

把判断写成代码是检验这一点最直接的方式。代码不接受模糊。写 `ROIC >= 15` 就必须先回答 15 这个数字从哪来，写「位置合适」就必须先定义什么叫合适。凡是写不出来的，说明原本就没想清楚。

1.2 三个具体的失败

促使我动手的是三个可以复盘的判断失误，它们分别指向不同类型的问题。

第一个是 Costco。我早期用净利率筛过公司，看到 Costco 的净利率只有百分之三左右，直接把它划进了低质量那一类。后来才理解，会员制零售的商业模式就是靠极低的商品毛利换取会员费和周转速度，衡量它的正确指标是资本效率而不是利润率。用净利率去判断零售业，等于用一把为软件行业校准过的尺子去量另一个行业。这是**指标与对象不匹配**造成的错误。

第二个是诺和诺德（NVO）。它的市盈率在同类药企里明显偏低，我一度把这个当作估值优势。但如果去看券商对下一财年营收的一致预期，会发现预期营收低于当前的滚动十二个月营收，也就是说市场并不是在给它打折，而是在为增长停滞定价。低市盈率在这里是结论而非机会。这是**只看历史数据不看前瞻数据**造成的错误。

第三个是 Sterling Infrastructure（STRL）。我的技术指标在周线和日线上都没有给出任何信号，但同一个指标在四小时图上的读数很高。三个时间周期对同一只票给出三个不同的答案，而我此前默认只看其中一个。这是**观察尺度单一**造成的错误。

三个失误的共同点是，它们都不是因为我获取的信息不够，而是因为我处理信息的规则本身有问题，并且这些规则从未被写下来审视过。

1.3 项目目标的界定

基于上面的分析，我给项目设定了四个目标，并且刻意排除了一些看起来相关但实际不同的目标。

目标	具体含义
规则外显	每一个分类结论背后都有一条可以指出来的规则，没有「综合感觉」这一项
全市场覆盖	不只扫描我的自选列表，避免只在已知范围内找答案
口径可追溯	每个数字都能说清楚出处和计算方式，遇到异常能定位到具体字段
可被推翻	规则写出来之后，任何一条都可以被反例挑战，挑战成立就改

明确排除的目标有两个。一是不做买卖信号的自动执行，系统只输出分层和理由，决定仍然由人做。二是不做收益率预测，系统不回答「这只票能涨多少」，只回答「按我的标准它属于哪一档、为什么」。这两条排除是有意的，后面第十节会说明原因。

二、方法论：为什么以投入资本回报率为主判据

2.1 市盈率衡量的不是质量

市盈率是价格除以每股盈利，分子是市场给的价，分母是会计算出来的利润。它衡量的是市场愿意为当前盈利付多少倍的价格，这是一个关于定价的量，不是关于企业本身的量。同样的市盈率可以对应完全不同的两种情况：一家高质量公司暂时被低估，或者一家公司的盈利即将下滑而市场已经知道了。仅凭市盈率无法区分这两者。

诺和诺德的例子就是后一种。所以在系统里，市盈率被降级为参考列，它出现在表格中但不参与分档。真正参与分档的是投入资本回报率、前瞻营收增速、券商评级和目标价空间。市盈率只在一个地方进入规则：超过一百倍时触发规避，那是把它当作异常值检测器而不是估值指标用。

2.2 绝对利润率门槛的跨行业失效

我最初的质量标准是四条绝对门槛：净资产收益率高于百分之十五、毛利率高于百分之四十、净利率高于百分之二十、自由现金流占营收高于百分之十。这四条来自价值投资的常见讨论，单看每一条都说得通。

问题出在它们被当作硬门槛并联使用的时候。这套门槛隐含了一个假设，即优秀企业应该在利润率上表现优秀。对软件和医药这类高毛利行业，这个假设大体成立。对零售、分销、工程承包这些行业，它系统性地给出错误答案。Costco 只是最明显的一例，同类问题还出现在 TJX、AutoZone、Lowe's 这些公司身上，它们的共同特征是靠周转速度而非单位利润赚钱。

这在方法上是一个**构念效度**问题。我想测量的构念是「企业质量」，我选择的操作化指标是利润率，但这个指标在不同行业中与目标构念的对应关系并不一致。用管理学研究的语言说，这个测量不具备跨组的测量等价性。

2.3 投入资本回报率与资金成本

投入资本回报率的定义是税后经营利润除以投入资本，衡量的是企业每投入一块钱能产生多少回报。它比利润率更接近「质量」这个构念，原因在于它同时包含了利润和效率两个维度。Costco 的净利率只有百分之三，但它的资本周转极快，所以投入资本回报率能达到百分之二十以上，这个数字正确地反映了它是一门好生意。

门槛的选取有一个可以说清楚的参照。美国大型上市公司的加权平均资本成本大体在百分之七到十这个区间，这意味着投入资本回报率低于这个水平的企业，其增长在经济意义上并不创造价值，规模越大反而消耗越多股东资本。系统把百分之八设为最低一档的下限，含义是「至少覆盖资金成本」。百分之十五和百分之二十这两档则是经验校准的结果，对应「明显优于资金成本」和「具备某种结构性优势」。

需要说明的是，后两个门槛的具体数值没有严格的理论依据，它们是在观察了一批我熟悉的公司之后定下来的。这是一个主观选择，第十节会把它列为系统的局限之一。

2.4 四条辅助门槛与双路径设计

原来的四条利润率门槛并没有被丢掉，而是改成了备用路径。最终的质量分级规则是：

等级	路径一（资本效率）	路径二（四条门槛）
A	投入资本回报率 ⩾ 20%	四条全部满足
B	投入资本回报率 ⩾ 15%	满足其中三条
C	投入资本回报率 ⩾ 8%	满足其中两条
D	以上都不满足	

两条路径是或的关系，满足任意一条即可。这样设计的考虑是：资本效率路径能正确处理低利润率的高周转生意，四条门槛路径能捕捉那些资本回报率因为会计口径原因被低估、但盈利质量确实很好的公司。两条路径互为补充，减少单一指标失效带来的误判。

四条门槛本身是：净资产收益率高于百分之十五、毛利率高于百分之四十、净利率高于百分之二十、自由现金流占营收高于百分之十。把它们从并联的硬门槛改成计数式的软门槛，是这个项目里第一处方法上的实质改动。

2.5 从历史数据到前瞻口径

诺和诺德那次失误暴露的问题是，所有基于滚动十二个月的指标都在描述过去。企业经营状况的变化会先反映在券商预期里，再反映在财报里。如果只看财报，系统对拐点的反应至少滞后一个季度。

我因此在系统里加入了两个前瞻字段：下一财年营收的一致预期和下一财年每股收益的一致预期。由它们可以算出前瞻营收增速和前瞻市盈率。前瞻营收增速被直接写进了分档规则：低于负百分之四时，无论质量等级多高，一律降入风险档并附上文字说明。

负百分之四这个阈值不是零，原因是财年口径本身有噪声。不同公司的财年起止不同，一致预期会随覆盖券商数量变化而波动，我观察到的噪声幅度大约在正负百分之二。把阈值设在负四可以避开这个噪声带，同时在负四到零这个区间仍然输出文字警示但不降档。这是一个在假阳性和假阴性之间做的权衡，倾向于宁可少杀。

三、数据工程

3.1 数据源选型

系统需要的数据有三类：基本面（资本回报率、利润率、增长、资产负债表）、价格与技术位置（均线、相对强弱、各周期高低点）、以及市场预期（券商目标价、评级、下一财年一致预期）。第三类是关键，因为它不是从公开财报能算出来的，需要机构数据商聚合。

我考察过几条路。专业终端的数据质量最好但成本不现实。免费财经网站的接口通常缺少一致预期，或者字段覆盖不完整。最后选定 TradingView 的 Scanner 接口，理由是它在一个请求里同时提供三类数据，字段覆盖足够，并且响应格式稳定。

接口的调用方式是向 `scanner.tradingview.com/america/scan` 发 POST 请求，请求体包含筛选条件、需要的字段列表、排序方式和分页范围。有一个实现细节值得记下来：把请求头的 `Content-Type` 设为 `text/plain` 而不是 `application/json`，可以让浏览器把它当作简单请求处理，绕过跨域预检。这个改动看起来微不足道，但它是整个系统能在无服务器环境里跑通的前提之一。

3.2 关键验证：券商一致预期的可得性

选定数据源之后，我做的第一件事是验证前瞻字段的可信度。这是整个项目里最有价值的一次验证，因为在此之前，前瞻数据一直要靠我手动去查，这是系统最大的缺口。

验证方法是取一批我在 FactSet 上查过的公司，把 TradingView 返回的 `earnings_per_share_forecast_next_fy` 和 `revenue_forecast_next_fy` 与 FactSet 的一致预期逐一对照。以 AppLovin（APP）为例，TradingView 返回的下一财年每股收益预期是 15.957，FactSet 的一致预期是 15.95，差异在千分之一以内。其余样本的偏差也都在可接受范围。

这个结果的含义是：TradingView 的前瞻字段来自与 FactSet 相同或高度重合的券商预测池。这一验证直接把原本需要逐只手查的工作变成了全市场自动扫描，也让诺和诺德那类「低市盈率实为增长停滞」的陷阱可以被系统自动标出来。

> 这里有一点方法上的体会。数据源验证的成本远低于它避免的成本。我在验证上花了大约两小时，而它替代的是此后每次研究一只票都要重复的手动查询。更重要的是，如果不验证就直接用，系统会给出看起来精确、实际上不知道从哪来的数字，那比没有数字更危险。

3.3 扫描范围与字段清单

扫描的筛选条件有五条：交易所限于纽交所、纳斯达克、美交所；只取主上市；类型为股票；类型标签必须包含 common；市值高于十五亿美元。

第四条是后来补上的。最初没有限定 common，结果优先股被扫了进来。优先股的代码形态和普通股类似，但它的财务指标口径完全不适用，市盈率之类的字段要么为空要么是无意义的数字。这类混入很难从结果里看出来，因为它不报错，只是安静地污染统计。

市值门槛设在十五亿美元的理由是流动性。低于这个规模的公司，券商覆盖稀疏，一致预期往往只有一两家投行的数字，前瞻字段的可信度会明显下降。系统另外对流动性做了标记，十日平均成交量乘以价格低于三千万美元的会被标为流动性偏低。

请求的字段共 37 个，分成五类。分页每次取四百条，循环上限设在 2800，某一页返回不足四百条即提前结束。

类别	字段
识别与规模	name, market_cap_basic, industry, total_revenue_ttm, average_volume_10d_calc
盈利质量	return_on_invested_capital, return_on_equity, gross_margin_ttm, net_margin_ttm, free_cash_flow_margin_ttm, total_revenue_yoy_growth_ttm, price_earnings_ttm
资产负债表	cash_n_short_term_invest_fq, total_debt_fq, free_cash_flow_fq, free_cash_flow_ttm, total_current_assets_fq, total_current_liabilities_fq
价格与位置	close, SMA50, SMA200, EMA100, RSI, High.6M, price_52_week_low, price_52_week_high, Low.3M, Low.1M, Perf.1M, Perf.Y, Perf.3Y
市场预期与日程	price_target_1y, recommendation_mark, revenue_forecast_next_fy, earnings_per_share_forecast_next_fy, earnings_release_date, earnings_release_next_date

系统另外维护一份五百余只的自选清单，按行业和观察强度分组。这份清单不参与筛选，只用于在结果里标注哪些是我原本就在跟踪的，哪些是系统扫出来但我没见过的。后者用「新」标记。这个设计的目的是让系统有机会推翻我的关注范围，而不是在我已有的范围内确认我已有的想法。

3.4 数据新鲜度与两层缓存

行情数据由交易所实时或延迟十五分钟提供，取决于具体交易所。基本面指标随财报更新，季度频率。券商目标价与评级是日频。前瞻一致预期随券商修正而变，通常也是日频。

页面上会显示当前的市场状态，区分周末休市、盘前、盘中、已收盘四种情况，并明确告诉使用者当前看到的行情停在哪个时点。这一项是因为我发现自己周末看面板时会误以为数字是当天的。让系统主动说明数据的时间状态，比让使用者自己推断更可靠。

缓存分两层，这个设计后面第 6.5 节会说明它曾经出过什么问题。边缘缓存保存十五分钟，避免重复扫描全市场；发给浏览器的响应则明确标为不缓存。另外有一个强制重算的入口，但做了九十秒的限流，因为网站是公开的，不能让任何访客无限触发全市场扫描。

四、系统架构

4.1 为什么是 Cloudflare Worker

选择无服务器架构有三个考虑。第一是没有服务器要维护，我不想让这个项目变成一件需要持续运维的事。第二是部署路径极短，改完代码粘贴进控制台点部署，所有访问者立刻看到新版本。第三是它天然解决了跨域问题，因为对 TradingView 的请求是从服务端发出的。

还有一个后来才体会到的好处。无服务器环境强制我把状态降到最低，整个系统只有一个全局变量用于强制重算的限流。状态少意味着可能出错的地方少。

4.2 计算与展示的分离

所有的公式、阈值、分档规则和人工覆盖都在服务端执行。发到浏览器的是算完的结果，包括每只票的等级、档位和一段文字说明。前端代码只负责渲染、排序、筛选和折叠，不包含任何判断逻辑。

这样做主要是因为网站是公开的，我不希望参数表直接可见。但它还带来一个没预料到的好处：因为前端拿不到原始公式，我在调试时不可能靠前端临时改一下数值来看效果，只能回到服务端改完重新部署。这个摩擦反而让每一次参数调整都变成了一次有记录的改动。

4.3 列定义单一真相源

面板有四张表，字段各不相同，并且我经常需要增删列。最初表头和数据行是分别写的两段代码，改一处忘改另一处的话，表头和数据就会错位。这种错位不会报错，它只是让某一列的数字挂在了错误的标题下面，而且看起来完全正常。

我把它改成了一个集中的列定义对象，每一列在其中登记显示名称、排序键、对齐方式和取值函数。四张表各自只维护一个字段名数组，表头和数据行都由同一个定义生成。

```
const CD = {
  n:    { th:'代码',     L:1, cell: r => ... },
  Qn:   { th:'质量',     sk:'Q', cell: r => ... },
  roic: { th:'ROIC',     cell: r => ... },
  ...
};
const T2 = ['n','tier','Qn','roic','rg','fg','pe','fpe', ...];   // 分层操作表
const T4 = ['n','g','tier','Q','roic','rg','fg','pe', ...];      // 全市场表

function buildHead(id, keys) { ... 由 CD 生成表头 ... }
function buildRow(r, keys)   { ... 由 CD 生成数据行 ... }
```

改完之后，增减一列变成改一个数组元素，表头和数据不可能不一致。这不是性能优化，是把一整类错误从「可能发生」变成「结构上不可能发生」。这类改动的收益不在当下，而在于以后每一次修改都少了一个出错的机会。

五、分类体系

分类分三层：质量回答公司本身好不好，位置回答现在是不是时候，信号回答有没有到某种极值。三层的结果再合成一个档位。质量按季度变化，位置和信号每天变化。

5.1 质量分级

规则见 2.4 节。有两个细节需要补充。

第一，完全没有财务数据的公司（新上市或券商未覆盖）会被判为 D 级，但系统会另外记录一个标记，把这种情况的文字说明写成「财务数据缺失无法评级，非质量差」。如果不做这个区分，新股会和真正的劣质公司混在一起，而两者需要的处理完全不同。

第二，金融行业被单独隔离。银行、保险、房地产投资信托、资产管理公司的商业模式是经营资产负债表，投入资本回报率这个指标对它们没有意义。这类公司被归入独立的一档，说明文字直接写明应该换用净息差、不良率、每股营运资金之类的体系来看。系统不假装能评价它们。

5.2 位置判定

位置用三个量定义：相对两百日均线的距离、相对强弱指标、以及最近一个月的涨跌幅。先定义两种排除状态：

- 过热：相对强弱指标不低于 66，或者月涨幅不低于百分之二十
- 过冷：相对强弱指标不高于 35，或者月跌幅不低于百分之十五

然后分三种位置：

- **买入位**：站在两百日均线上方（允许低于均线百分之二以内），相对强弱指标不低于 38，且既不过热也不过冷
- **等待位**：不满足买入位的条件，但距两百日均线不低于负百分之十五，且既不过热也不过冷
- **观察**：其余情况

把过热单独排除的理由是，即使一家公司质量很好、位置也在均线之上，如果它刚刚一个月涨了三成，此时建仓的风险收益比明显变差。这一条是为了防止系统在追高的时候给出「可分批建仓」的建议。

5.3 信号识别

信号对应我在图表上使用的一个成交量与价格极值指标。系统用三个条件把它的逻辑近似出来，基准是五十二周低点。三个信号共同的前提是上方空间足够，也就是距离六个月高点还有百分之二十五以上。

信号	条件	含义
蓝柱	距五十二周低点 ⩽ 3%	正处在极低位，含跌破的情况
近期过线	低点是近三个月内做出的，现已从低点反弹 3% 至 18%	刚从底部脱离
接近区	在低点上方 3% 至 6%，但低点是几个月前的旧低	预警，强度弱于前两者

「近期」这一条的上界原本设在百分之十二，后来放宽到十八。改动的依据是三个反例：TSLA 在 12.2、GPOR 在 13.5、LULU 在 14.2，这三只在图表上确实有信号，但被百分之十二的上界切掉了。放宽之后它们进来了，同时我复查了原本就在范围内的样本，没有出现明显的假阳性。

另外有一个新鲜度判断：五十二周低点是不是最近三个月内做出来的。这个判断最初用的是一个月，后来发现像 LULU 那样两个月前见底的会被漏掉，改成三个月。

5.4 分档级联

九个档位按严格的优先级依次判断，命中即停。顺序本身承载信息：越靠前的条件越是「一票否决」性质的。

> ① **人工覆盖 → 按标注的档位**　（数据看不见的事件，如产品线遭遇低价竞争、指引下调、诉讼悬置）
>
> ② **金融档 → tz**　（银行、保险、REIT、资管，资本回报率口径不适用）
>
> ③ **烧钱且现金告急 → 避开**　（跑道不足 6 季，或已净负债且流动比率小于 1）
>
> ④ **烧钱但现金充足 → 早期档**　（比率失真，改看现金跑道与市值减净现金）
>
> ⑤ **避开 → D 级，或资本回报率为负，或负增长且烧钱，或市盈率高于 100**
>
> ⑥ **风险 → 前瞻营收增速 ⩽ −4%，或券商评级弱于 1.9，或目标价空间小于 5%**
>
> ⑦ **一档 · 可分批 → A 或 B 级 ＋ 买入位 ＋ 空间 ⩾ 12% ＋ 评级 ⩽ 1.65**
>
> ⑧ **二档 · 等信号 → A 或 B 级 ＋（有蓝柱或近期信号，或处于等待位且空间 ⩾ 20%）**
>
> ⑨ **三档 · 观察 → 其余**

图 1　分档级联。九条规则按顺序判断，任一条命中即停止，不再往下走。

有两点需要解释。

第一，为什么风险判定（⑥）排在一档二档（⑦⑧）之前。因为前瞻营收转负这类信号的性质是否决性的，它不该被高质量等级抵消。一家资本回报率很高但明年营收预期下滑的公司，其高回报率正在成为过去式，此时用历史指标给它高评价是错的。把否决条件放在前面，等于规定了「质量不能赦免预期」。

第二，人工覆盖排在最前面，是因为它处理的恰恰是数据字段覆盖不到的信息。系统目前有十条人工覆盖记录，每一条都写明了理由，例如某家公司的核心产品遭遇低价竞争、某家公司的高净资产收益率实际是回购缩小分母造成的。这类判断无法从任何字段推出来，只能人工标注。把它做成显式的覆盖层而不是偷偷改规则，是为了让它可被审查：任何时候都能列出系统里有哪十条判断是我个人加的，以及为什么。

5.5 一个反直觉的结论

分类体系建好之后，出现了一个我事先没想到的结果：**A 级不等于一档**。

质量和档位是两个维度。质量回答值不值得买，档位回答现在是不是时候。一家 A 级公司如果正处在五十二周低位，它会落在二档而不是一档，因为底部区域的下跌尚未确认结束，此时建仓和在趋势恢

复后建仓是两种不同的打法。反过来，一家 A 级公司如果已经站回买入位、上方空间充足、券商评级也好，它才进一档。

这个结果之所以有意义，是因为它正好修正了我原来的一个习惯。我此前倾向于在看好一家公司的质量之后就直接买，把「好公司」和「现在该买」混为一谈。把两个维度分开之后，我才能问出一个更准确的问题：我现在是在为质量付钱，还是在为位置冒险。

六、迭代记录：六次错误及其修正

这一节是整份报告我认为最有价值的部分。六次错误分属不同类型，修正方式也不同。我按类型而不是时间顺序排列。

6.1 边界误差：浮点数造成的信号静默丢失

现象　某些我在图表上确认有蓝柱信号的股票，在系统里没有被标出来。数量不多，也没有明显规律。

原因　蓝柱的条件是距五十二周低点不超过百分之三，代码写成 `d52 <= 3`，其中 `d52 = (px / l52 - 1) * 100`。当价格恰好比低点高百分之三时，浮点运算的结果不是干净的 3，而是 3 后面跟着一串零再跟一个极小的非零位。这个值大于 3，条件判为假，信号被丢弃。

为什么难发现　它不报错，不产生异常值，只是在边界上安静地少一条记录。如果不是我恰好逐只对照过图表，这个问题可以一直存在下去。

修正　引入容差比较函数，把所有涉及阈值的比较统一替换：

```
const le = (a, b) => a <= b + 1e-9;
const gt = (a, b) => a >  b + 1e-9;
const ge = (a, b) => a >= b - 1e-9;
```

然后为每个阈值构造正好落在边界上的测试用例，共十三个，验证修正后全部通过。

这次错误说明的问题　筛选系统的失败有两种形态。一种是输出了不该输出的（假阳性），这种容易发现，因为你会看着结果觉得不对。另一种是没输出该输出的（假阴性），这种几乎不可能靠看结果发现，因为缺失的东西不会主动出现。系统设计时应该对第二种保持更高警惕，而检验它的唯一办法是拿已知答案去反查。

6.2 分类误差：金融隔离规则的双向失效

现象　一次例行检查中我发现，摩根士丹利、贝莱德、阿波罗全球管理这些公司没有被归入金融档，它们的资本回报率被当作普通指标参与了分档。同时，芝加哥商品交易所、洲际交易所、纳斯达克这三家被错误归进了金融档。

原因　最初的隔离规则是对行业标签做正则匹配。实测发现行业标签的实际取值和我以为的不一样：摩根士丹利、贝莱德、阿波罗的标签是 Investment Managers，Ares Capital 的标签是 Financial Conglomerates，Blackstone Secured Lending 的标签是 Investment Trusts，这些原正则都没覆盖。同时原正则里包含 Banks 这个词，而交易所类公司的行业描述中恰好含有这个词，于是被误扫。

修正　改成三层结构：

1. 宽正则先扫，覆盖房地产投资信托、银行、保险、资产管理、金融集团、投资信托六类标签
2. 白名单豁免十八家被正则扫中但实为收费型运营商的公司。交易所、支付网络、纯收费资管、保险经纪的商业模式是收手续费而非经营资产负债表，资本回报率对它们是适用的
3. 黑名单补五家行业标签扫不到但实为资产负债表信贷生意的公司，主要是消费信贷类

这次错误说明的问题　我犯的不是编码错误，是把自己对行业分类的想象当成了数据的实际取值。规则依赖外部数据的枚举值时，必须先去实测这些值到底有哪些，而不是凭常识推断。修正后的三层结构也不优雅，但它承认了一件事：行业标签是别人定义的，我的规则必须适配它，而不是反过来。

6.3 口径误差之一：营收基准不一致

现象　StoneCo（STNE）的前瞻营收增速显示为正百分之四百零四。

原因　前瞻增速的算法是券商预测的下一财年营收除以当前滚动十二个月营收减一。对大多数公司这没问题。但支付、信贷、保险类公司的财报营收有时按净额口径列示（扣除支付给第三方的成本），而券商预测按总额口径给出。两个口径相除，得到的数字没有任何意义。

这类问题最危险的地方在于它看起来像一个精确的数字。如果一个字段显示为空，使用者知道这里没有数据。显示正百分之四百零四，使用者会以为这是一个测量结果。

修正　加入口径异常检测，触发任一条即判定口径不一致，把前瞻增速置空并显示「口径异」：

- 净利率绝对值大于 100%（净利比营收还大，在同一口径下数学上不成立）
- 预测值与实际值的比值大于 2.5 或小于 0.4

加上这个检测后，全市场约有一百只股票被标记。我抽取八个已知案例做了验证，八个全部正确识别。

这次错误说明的问题　两个数相除之前，要先确认它们的口径一致。这在财务分析里是常识，但当计算被写进代码、批量作用在几千家公司上时，常识不会自动生效。系统必须显式地检查口径，因为它没有人的那种「这个数字不对劲」的直觉。

6.4 口径误差之二：早期烧钱的规模伪装

现象　空气化工产品（APD）被系统归入「早期烧钱型」。这是一家成立多年的大型工业气体公司。

原因　早期档的原始判定是自由现金流为负且净利率为负。APD 在某一期因为计提减值导致这两项同时为负，于是被误判。同样的问题还出现在 ECHO 身上，那家公司有一百六十亿美元净负债，完全不符合「早期」的含义。

修正　加入规模守门条件，要求营收低于八亿美元，或者净利率低于负百分之三十。前者排除成熟大公司的偶发亏损，后者保留那些规模已经不小但亏损幅度确实属于早期特征的公司。

另外还发现一个相关问题：现金跑道的计算是现金除以单季烧钱额，当某一季只是微亏时会算出四百多个季度。MicroStrategy 就出现过这种情况。四百个季度这个数字没有信息量，只会让人困惑，所以给跑道加了四十个季度的上限。

这次错误说明的问题　当一个分类的名称（早期）和它的判定条件（两项为负）之间存在语义缺口时，缺口迟早会被某个边缘案例撑开。修正的方式不是调整阈值，而是补上那个被省略的语义条件，也就是「规模还小」这一层含义。

6.5 基础设施误差：两层缓存的混淆

现象　新增了强势股回调池这个功能之后，页面显示筛出零只。我确认过规则本身是对的。

诊断过程　这次花了三轮。第一轮判断是边缘缓存里存着旧版本的数据，于是加了代码指纹作为缓存键的一部分，任何一次部署改动都会让旧缓存自动失效。部署后仍然是零。第二轮我直接请求接口，确认返回的数据里回调池标记是正常的，有四十五只和三十只。数据是对的，页面是错的。第三轮才找到真正原因。

原因　接口返回的响应头带着为边缘缓存设置的十五分钟有效期。这个响应头同时也被浏览器读到了，于是浏览器自己把这份 JSON 存了十五分钟。部署新版本后，页面代码是新的，数据却是浏览器缓存里的旧版本，旧版本没有回调池字段，所以筛出零只。

修正　把两层缓存的响应头分开。存进边缘缓存的那份带有效期，发给浏览器的那份明确标为不存储。前端请求时另外加时间戳参数并显式声明不使用缓存。

```
// 存边缘缓存：带 max-age
ctx.waitUntil(cache.put(key, new Response(body, {
  headers: { "Cache-Control": "public, max-age=" + P.cacheMinutes * 60 }
})));

// 发浏览器：必须 no-store
return new Response(body, {
  headers: { "Cache-Control": "no-store, must-revalidate" }
});
```

6.6 观察者误差：测试方法自身的盲区

上一节那个问题我诊断了三轮才找到，原因值得单独写一节。

我在验证接口时，每次请求都习惯性地带上时间戳参数，用来确保拿到的是新数据。这个习惯本身是对的，但它导致我的每一次测试请求都绕开了浏览器缓存。也就是说，**我用来检验系统的方法，恰好**

免疫于系统当时正在犯的那个错误。所以我每次测试都看到正确结果，而用户（包括我自己用正常方式打开页面时）看到的是错的。

这是六个错误里我认为最有价值的一个，因为它不是关于金融或代码的，而是关于验证本身。检验工具如果和被检验对象共享某个假设，那么在这个假设出问题时，检验就失效了。这个现象在管理研究里有对应的概念，比如用同一份问卷同时测量自变量和因变量会产生共同方法偏差。形式不同，结构是一样的：测量手段和被测对象不独立。

我从这次得到的具体做法是，验证时至少要用一条和日常操作路径完全相同的通道，而不是全部用「更干净」的测试通道。测试环境比生产环境干净，本身就是一种偏差。

七、验证方法

系统的验证分四种，各自能查出的问题类型不同。

方法	做法	能查出什么
边界测试	为每个阈值构造正好落在边界上的输入，共十三个用例	浮点误差、比较符号写反、区间开闭错误
反例校准	收集我在图表上确认有信号但系统没标出的案例，逐个追查	假阴性，也就是漏掉的信号
交叉验证	把系统的数值与 FactSet 上的对应数值逐一对照	数据源口径问题、字段含义误解
结构自检	检查表格字段数组里的每个键是否都在列定义中存在，检查各渲染函数在数据未加载时是否有守卫	重构引入的不一致、空指针

反例校准是效果最好的一种。它的做法是：我先在图表上独立判断一批股票有没有信号，再拿系统的输出去对照，只关注两者不一致的部分。信号区上界从百分之十二放宽到十八，就是这样得出的。最近一次做了十四个样本的对照，全部一致。

另外有一次纠正值得记录，方向是我纠正助手（见前面「关于工作方式的说明」）。分析 Wingstop（WING）时，助手看到一个每股收益同比负百分之二十九点六的数字，据此判断盈利在下滑，并且推测了几个可能的原因。我拿另一个来源给出的相反结论去质疑它，回查实际财报后确认助手是错的：当期净利润和每股收益都在增长，负百分之二十九点六来自上一年同期包含一次性收益造成的基数扭曲。

这次的教训不只是「要核对数据」。更具体地说，错误在于看到一个同比数字就直接解释它，而没有先问这个数字的基期里有没有异常项。同比是两个时点相减，异常可能在任何一端。看到反常的同比

数字时，正确的第一步是拆开看两端，而不是给它找解释。找解释这个动作本身会让人过早地相信那个数字是真的。

八、多周期问题

STRL　那个例子是这样的：同一个技术指标，在周线上的读数是　0.88，日线是　5.43，四小时是 29.03。三个数相差一个数量级以上。

指标本身没有出错。它的窗口长度是固定根数，所以在不同周期上覆盖的实际时间跨度完全不同。周线一百五十根约等于三年，日线约等于七个月，四小时约等于三到四个月。同一只股票在三年尺度上远离低点、在三个月尺度上正处于低点，这在逻辑上完全可以同时成立，它描述的是不同时间范围内的相对位置。

问题在于系统只使用了五十二周口径，所以四小时级别的机会看不见。STRL　距离五十二周低点还有百分之六十九，日线和周线的信号永远不会亮。

直接的解决办法是把距三个月低点不超过百分之三作为新条件，但这样筛出来一百只，里面既有牛股的正常回撤，也有持续阴跌的股票。两者在「距三月低点很近」这一点上没有区别，但操作含义完全相反。

我加了两个条件把它们分开：年内涨幅不低于百分之二十，以及距离五十二周高点在负百分之二十五以内。前者要求它今年确实是领涨的，后者要求这次回调确实有深度。加上之后从一百只收窄到四十五只，逐个看过，都是领涨股的深度回调，包括 STRL 本身。

这个回调池在系统里是独立的一区，不和底部信号混在一起。原因是两者的操作逻辑不同：底部信号是在赌趋势反转，回调池是在跟随既有趋势，前者的失效点是跌破五十二周低点，后者的失效点是跌破三个月低点。把逻辑不同的东西放进同一张表，使用者会不自觉地用同一套方式对待它们。

这里有一点方法上的收获。我最初以为这是一个「要不要增加四小时周期」的技术问题，答案是要或不要。实际上它是一个转译问题：四小时周期真正提供的信息是「更短窗口内的相对位置」，而这个信息可以用日线数据里的三个月高低点来表达，不需要真的去取四小时数据。想清楚指标在测量什么，比想清楚它用什么周期更重要。

九、投资认知的四处转变

9.1 从找便宜到分层

我原来的思路是在市场里找被低估的东西，判断标准是估值指标偏低。做完这个系统之后，我的思路变成先确定一家公司属于哪一档，再决定这一档该怎么处理。

区别在于，前者假设存在一个客观的「合理价值」，我的工作是发现价格与它的偏离。后者不做这个假设，它只是承认不同状态的公司需要不同的应对：一档可以分批建仓，二档要等信号，三档只观察，风险档需要单独判断，早期档看的是现金跑道而不是任何比率。

这个转变实际上降低了系统的野心。它不再试图回答「这只票值多少钱」，只回答「按我的标准它现在处于什么状态」。后一个问题我有能力回答，前一个我没有。把问题缩小到自己能回答的范围，是这个项目给我的一个比较重要的调整。

9.2 从绝对指标到资本效率

Costco 那次误判改变的不只是一个指标的选择。它让我意识到，任何指标都隐含着一个关于「好生意长什么样」的假设，而这个假设是有适用范围的。利润率隐含的假设是好生意靠单位利润赚钱，这个假设对软件成立，对零售不成立。

换成资本回报率并不是找到了一个没有假设的指标，它同样有假设，比如它对重资产和轻资产企业的处理就不完全对称，对商誉占比高的公司也会失真。区别只在于它的假设范围更宽，覆盖的行业更多。所以系统保留了四条利润率门槛作为第二条路径，就是承认没有单一指标能覆盖所有情况。

9.3 从判断到规则

这是四处转变里最根本的一处。把判断写成规则，失去的是灵活性，得到的是两样东西。

一是可追溯。系统说某只票是二档，我可以立刻指出是哪一条规则让它落在二档的。如果我不同意这个结论，分歧就被定位到了那一条规则上，而不是停留在「我感觉不太对」。

二是可积累。不成文的判断没法积累，因为它每次都重新形成一遍，上一次的教训不会自动带到下一次。写成规则之后，每一次修正都留在系统里，下一次不会再犯同一个错。六次错误的记录本身就是这个积累的载体。

代价是真实的。规则会在边缘案例上给出机械的答案，人工覆盖那十条就是规则处理不了的部分。但把这些例外做成一个显式的、可以被列举和审查的覆盖层，比让它们隐藏在「综合判断」里要好。至少我知道自己一共加了多少私货，以及每一条的理由是什么。

9.4 从确信到可证伪

项目开始时我想做的是一个能给出正确答案的系统。现在我认为它的价值不在给出答案，而在于它给出的每个答案都能被具体地反驳。

差别是这样的：如果系统说某只票是一档，而我认为不是，那么我们的分歧必然落在质量等级、位置判定、空间和评级这几项的某一项上。我可以指出是哪一项，并且论证那一项的规则该怎么改。改完之后，全市场几千只股票的分类会一起变，我可以马上看到这个改动是否引入了新的问题。

这个性质让系统变成了一个可以持续改进的东西，而不是一个用来确认既有想法的工具。第 3.3 节里给非自选股票加「新」标记，也是出于同样的考虑：如果系统只在我已经关注的范围内工作，它永远不可能告诉我我漏掉了什么。

十、局限

这一节列出系统确实不能做到的事。我认为这部分和前面的内容同样重要，因为一个不说明边界的工具会被用在它不适用的地方。

无法进行真正的回测

数据源提供的是当前值而非历史时点值。也就是说我无法知道某只股票在去年三月那一天的资本回报率数据在当时是什么样的，我只能拿到今天的数字。没有时点数据就无法做严格的回测，因为任何用今天数据去评价过去决策的做法都会引入前视偏差。因此系统目前所有的验证都是关于规则一致性的，不是关于收益率的。系统没有证明它能赚钱，也不声称能。

自选清单带有样本选择偏差

那五百余只的清单来自我此前的关注范围，它本身就是一次筛选的结果，而且是我用尚未成型的标准做的筛选。扫描全市场部分缓解了这个问题，但清单仍然影响我看结果时的注意力分配。「新」标记是一个补救，效果有限。

阈值带有主观性

百分之十五和百分之二十这两个质量门槛、空间百分之十二、评级 1.65、月涨幅百分之二十这些数字，是经验校准的结果，不是优化出来的。我没有做过参数敏感性分析，不知道把一档的空间门槛从十二改成十五会让结果变化多少。这是一个已知的待办。

一致预期本身有系统性偏差

券商预测在方向上存在众所周知的乐观倾向，尤其是对小市值和覆盖数少的公司。系统把前瞻增速当作输入，就继承了这个偏差。目前的处理只是把市值门槛设在十五亿美元以过滤覆盖最稀疏的部分，这是缓解不是解决。

系统不解决的问题

它不判断商业模式的可持续性，不评估管理层，不理解竞争格局的变化，不处理监管和政策。人工覆盖那十条恰好都属于这一类，这说明这类信息的重要程度不低，只是系统碰不到。把系统的输出当作完整判断是错误的用法，它只是把可量化的那部分先处理好，让人把注意力放在剩下的部分。

十一、对管理与组织决策的一点延伸

这个项目的对象是投资，但过程里遇到的几个问题在管理研究和组织实践中有对应。我把它们列在这里，不是要做类比论证，只是记录我自己感受到的连接点。

隐性知识的显性化及其代价　把我的选股判断写成规则的过程，本质上是把个人的隐性知识转成组织可用的显性知识。这个过程有明确收益，包括可传递、可审查、可积累。但也有明确损失，主要是边缘情况的处理能力下降，人工覆盖那十条就是补偿这个损失的机制。这与知识管理文献里关于隐性知识编码化的讨论是同一个议题，我的体会是编码化的收益和损失都比预想的更具体。

构念效度与测量等价性　用利润率测量企业质量在不同行业中失效，这在方法论上是测量不具备跨组等价性。我遇到的三次口径问题（利润率跨行业、营收净额与总额、早期定义的语义缺口）都属于同一类：操作化的指标与想测量的构念之间的对应关系在某些子群中断裂了。做跨组比较的时候，先检查测量是否等价，这个习惯是这个项目让我建立起来的。

共同方法偏差的一般形式　第 6.6 节那个测试盲区，结构上等同于共同方法偏差：测量手段与被测对象共享了某个来源，导致偏差不可见。我现在的做法是在验证时刻意保留一条与真实使用完全一致的通道，哪怕它更脏。

分歧的可定位性　这是我认为对组织决策最有用的一点。规则化的真正价值不是保证结论正确，而是让分歧变成可定位的。两个人对同一只股票有不同看法时，如果双方都在用同一套显式规则，分歧必然落在某一条具体规则上，讨论就有了焦点。如果双方都在用综合判断，讨论只能停留在立场交换。这个性质在任何需要多人达成决策的场景里都成立，而且和结论是否正确无关。

附录 A　系统参数表

参数	取值	含义与依据
mcapMin	15 亿美元	扫描下限。低于此规模券商覆盖稀疏，一致预期可信度下降
blueLowPct	3%	蓝柱：距五十二周低点的上限
blueRoomPct	25%	三个信号的共同前提：距六个月高点的空间下限
rExitLo / rExitHi	3% / 18%	近期过线的反弹区间。上界原为 12%，因 TSLA、GPOR、LULU 三个反例放宽
nearLo / nearHi	3% / 6%	接近区，仅作预警
fgRisk	−4%	前瞻营收增速降档阈值。不设为 0 是为避开财年口径约 ±2% 的噪声
cacheMinutes	15	边缘缓存有效期
强制重算限流	90 秒	网站公开，防止访客无限触发全市场扫描
质量 A / B / C 门槛	20 / 15 / 8%	投入资本回报率。8%　的依据是美国大型公司加权平均资本成本区间的下沿
一档：目标价空间 / 评级	≥12% / ≤1.65	空间指现价到券商一年目标价的距离。评级为数值制，数字越小越偏买入。两项均为经验校准，未做敏感性分析
二档：等待位目标价空间	≥20%	等待位要求更大空间以补偿位置劣势
过热 / 过冷	RSI 66 / 35	辅以月涨幅 ≥20% 或月跌幅 ≥15%
买入位	SMA200 −2%	允许略低于均线，RSI 需 ≥38
现金告急	跑道 <6 季	或已净负债且流动比率 <1
跑道上限	40 季	微亏时会算出四百余季，无信息量，故封顶
早期规模守门	营收 <8 亿	或净利率 <−30%。防止成熟公司偶发亏损被误判
口径异常检测	比值 >2.5 或 <0.4	或净利率绝对值 >100%
流动性标记	3000 万美元	十日平均成交量乘以价格低于此值，标为流动性偏低
浮点容差	1e−9	所有阈值比较统一使用

附录 B　数据字段清单（37 项）

见正文 3.3 节的分类表。全部字段在一次请求中取回，分页每次四百条，循环上限 2800。

附录 C　分档级联伪代码

```
// 质量分级
q4 = 0
if (ROE > 15)          q4++
if (毛利率 > 40)        q4++
if (净利率 > 20)        q4++
if (FCF占营收 > 10)     q4++

if      (ROIC >= 20 || q4 == 4)  Q = "A"
else if (ROIC >= 15 || q4 == 3)  Q = "B"
else if (ROIC >=  8 || q4 == 2)  Q = "C"
else                             Q = "D"

// 位置判定
hot  = RSI >= 66 || 月涨幅 >= 20
cold = RSI <= 35 || 月跌幅 >= 15
if      (距SMA200 >= -2  && !hot && !cold && RSI >= 38)  pos = "买入"
else if (距SMA200 >= -15 && !hot && !cold)               pos = "等待"
else                                                       pos = "观察"

// 分档级联（命中即停）
if      (人工覆盖存在)                                  t = 覆盖指定档位
else if (金融)                                           t = "金融"
else if (烧钱 && 现金告急)                               t = "避开"
else if (烧钱)                                           t = "早期"
else if (Q == "D" || ROIC < 0
         || (增长 < 0 && FCF < 0) || PE > 100)     t = "避开"
else if (评级 > 1.9 || 空间 < 5 || 前瞻增速 <= -4) t = "风险"
else if ((Q == "A" || Q == "B") && pos == "买入"
         && 空间 >= 12 && 评级 <= 1.65)             t = "一档"
else if ((Q == "A" || Q == "B")
         && (有蓝柱 || 有近期信号
             || (pos == "等待" && 空间 >= 20)))      t = "二档"
else                                                     t = "三档"
```

附录 D　错误与修正汇总

类型	现象	原因	修正
边界误差	部分蓝柱信号未被标出	浮点运算使恰好等于阈值的值略大于阈值	引入 1e−9 容差比较函数，构造十三个边界用例验证
分类误差	资管公司未隔离，交易所被误隔离	行业标签的实际取值与假设不符，正则双向失效	改为正则加白名单（18 家）加黑名单（5 家）三层
口径误差	前瞻增速显示 +404%	营收按净额列示，预测按总额给出	加入比值与净利率的异常检测，置空并标注
口径误差	成熟工业公司被判为早期	判定条件缺少「规模还小」这层语义	加规模守门；跑道封顶四十季
基础设施	新功能显示零只	边缘缓存的有效期被浏览器一同采纳	两层缓存响应头分离，浏览器侧标为不存储
观察者误差	上一项诊断耗时三轮	测试请求带时间戳，恰好绕开了出问题的那层缓存	验证时保留一条与真实使用完全一致的通道

Thomas · 2026 年 9 月