好的，我先把这个脚本的总体计分方式说清楚，再逐个特征讲“想测什么”“代码里实际怎么算”“明显的问题/改进点”。

整体流程（很关键）

* 程序逐步把整盘棋的走子 push 到棋盘上；每 push 一步，就给“刚刚走子的那一方”把各项分数加到自己名下（因为 push 后轮到对手走，board.turn 变成对手，所以代码用 if board.turn != chess.WHITE: 来判断刚走的是白方）。
* 每个特征都是“逐步累加”的时间和（sum over plies），没有做归一化；因此同样风格下，长棋局会天然拿到更高总分，后续需要按步数/回合数做归一化。
* 这份代码片段里还有一些接口/颜色归属的错误（比如 calculate\_captures 不按颜色算、extract\_features 的入参和调用对不上），我在每个特征里会点出。

center\_control\_score（中心控制）  
想测什么

* 对 d4, e4, d5, e5 这些核心格的占领与控制力度（越多己方子力覆盖/控制中心越好）。

代码如何计算

* 设定中心格 D4,E4,D5,E5。
* 对每个中心格：  
  1）如果格子上“有任何棋子”（不分颜色）就 +1。  
  2）再把“己方”攻击到该格的棋子数加进去：board.attackers(chess\_color, square) 的数量。
* 每一步都加一次，形成全局累加。

问题/改进

* 把“占据中心格的任何棋子”都算分，不区分己方/对方，容易偏。应只给己方占据加分、对方占据扣分。
* 只统计己方攻击者数量，没有比较对手对中心的控制，建议改成“己方攻击者数 − 对手攻击者数”，并且占据中心格可额外奖励。
* 需除以总步数做归一化（或改成每步平均控制）。

piece\_activity\_score（子力活跃度/机动性）  
想测什么

* 己方所有子力的“控制格数量”（mobility），越能攻击/控制的格越多越活跃。

代码如何计算

* 遍历棋盘上“己方”的每一个棋子，求 len(board.attacks(square))，把这些数量累计求和。
* 每步都算一次并累加。

问题/改进

* 这是标准的“机动性”近似指标，但没有区分不同子型的价值（比如后/车控制格权重可能更大），也没有考虑被钉住导致的“名义攻击”与“可行行动”的差别。
* 建议：按子型加权；或把“合法走步数”而非“攻击格数”作为活跃度；并归一化（例如除以己方子数或总步数）。

king\_safety\_score（王安全）  
想测什么

* 王周围的防守是否充分、是否被对手威胁，兼顾兵盾结构。

代码如何计算

* 找到己方王格 king\_square。
* 己方守护王格的棋子数：len(board.attackers(chess\_color, king\_square)) 加到分里。
* 对手攻击王格的棋子数：len(board.attackers(enemy\_color, king\_square)) 从分里减去。
* 额外：遍历己方所有兵，如果该兵格被己方攻击（即被己方其他子力保护），就 +1（本意大概是“兵链/兵被保护”的加分）。

问题/改进

* “被己方攻击的兵 +1”并不直接等于“王安全”，更合理的是评估王翼/后翼的兵盾完整性（如 g/h 兵是否前进、王侧是否有空线、王附近格子是否被己方控制、对手重子是否渗透到王附近等）。
* 建议：
  + 兵盾：王侧（或后翼易位侧）的三兵是否完整、推进是否过度；
  + 王邻近圈（如以王为中心的 2 格曼哈顿距离内）己方控制格 − 对方控制格；
  + 开放/半开放线直对王的惩罚；
  + 敌后/车/马距王的“距离”与攻击线路。
* 仍需归一化。

attacking\_moves\_score（进攻/压迫性）  
想测什么

* 这一步是否制造了进攻：向对方阵地深入、对敌子施压、是否将军。

代码如何计算

* 仅看“刚刚走的那枚棋子”在新位置能攻击到的格：board.attacks(last\_move.to\_square)。
* 对每个被攻击格：
  + “纵深奖励”：如果是白方，rank≥5（对方半场）越高奖励越大（max(0, rank - 4)）；黑方用镜像逻辑。
  + 若该格有对方棋子：给一个“简化子值”分（兵=1，马/象≈3，车≈5，后≈7，王≈11），然后再减去“己方攻击该格的数量”（把它称为 defenders，但这里方向搞反了，见下）。
  + 若造成“将军”，再 +10。
* 合计作为本步的进攻分，然后累加。

问题/改进

* 防守方数方向反了：代码用 board.attackers(chess\_color, square) 当作“防守方数量”，但这是己方攻击该格的数量，实际应看对手防守该格的数量（board.attackers(enemy\_color, square)），否则价值折扣的方向会错。
* 只看“刚走的那枚棋子”，忽略了其它子力的协同威胁（串击、双攻、对线压力）。
* “纵深奖励”公式对白方/黑方的镜像写法不对称、黑方公式可再检查。
* 建议：
  + 把“被攻击的敌子价值 − 敌方该格防守子力数量的某种折扣”作为威胁评分；
  + 将军/将杀网（mate threat）更高权重；
  + 把对王附近圈的攻击控制格也计入；
  + 可加入“捕吃/将军/威胁高价值子”的离散特征。

captures\_score（可吃子机会/实际吃子）  
想测什么

* 原意像是“战术机会/吃子积极性”。

代码如何计算（当前实现）

* 遍历当前局面的 board.legal\_moves，对每个合法走子，如果是吃子 board.is\_capture(move) 就计数 +1。
* 但是这里最大的坑：这是“轮到对手走”的局面（因为刚 push 完），所以你实际上数的是“对手现在能吃的招法数量”，却把分加在“刚走子的一方”名下，方向完全相反。

问题/改进

* 如果要衡量“玩家实际的吃子积极性”，应统计“该玩家在整盘棋里实际完成的吃子步数/比率”（看 last\_move 是否为吃子，并归属给正确颜色），或“该玩家在自己走子时的可吃子数量”（需要在该方走子时刻评估）。
* 若要衡量“给对手留下了多少战术反击机会”，那保持现在的时序是对的，但要明确这是“反击暴露度”而非“吃子积极性”。
* 也需要归一化（例如除以可行着法数，得到“吃子占比”）。

pawn\_structure\_score（兵结构稳定性）  
想测什么

* 经典弱点：成双兵（doubled）、孤兵（isolated）等越多越差。

代码如何计算

* 统计己方每条线（file）上的兵数。
* 成双兵：一条线超过 1 个兵，惩罚 -(count - 1)。
* 孤兵：如果该线两侧相邻线都没有己方兵，这条线上的兵数量作为惩罚（-count）。
* 每步累加。

问题/改进

* 忽略了“后退兵”“通过兵”“连兵/链兵”“挂兵”等更细节的结构。
* 每步都重复计罚会导致长对局惩罚被重复叠加，建议只在“局面关键节点”或“每回合一次”计入，或者最后取平均。
* 可加入“通过兵奖励”“连兵奖励”“王翼兵盾完整度”等。

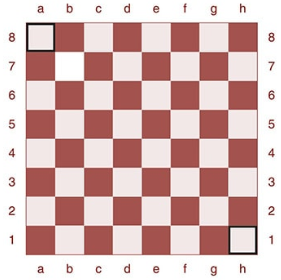
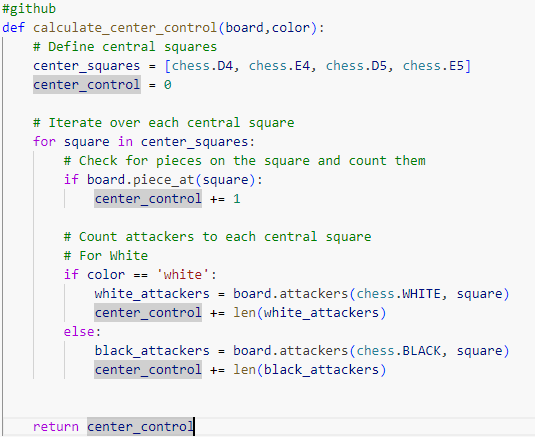
一些代码层面的明显问题（不修会影响你落地）

* extract\_features 的签名和调用不一致。定义里是 (game, game\_id, player\_id\_white, player\_id\_black) 并返回“白特征、黑特征”两个 dict；但主循环里你对每个 color 调用成了 extract\_features(game, game\_id, player\_info['player\_id'], color)，而且只接收一个返回值再直接拼到 data。需要统一接口：要么一次算整盘返回白/黑两个记录；要么按步更新时就明确哪一方的记录。
* calculate\_captures 没有用到 color，而且在“对手走子时刻”被统计；如果要做“我的吃子机会”，应在“我走子时刻”统计，或在副本棋盘中翻转 board.turn 再统计。
* 多数特征是“随步叠加的总和”，请务必在输出时加归一化（如除以己方总步数、或每 10 步平均），否则会把“走得多”误当成“特征强”。
* 目前的分值是“数量直接相加”，不同特征的量纲差异很大（例如活跃度可能数百，孤兵惩罚可能个位数），需要做标准化（z-score / min-max）或先做尺度对齐。

给你一个更贴近“定义”的修正方向（简短提要）

* 中心控制：己方对中心 4 格（或扩展到 16 格）的“控制数 − 对手控制数”，占据中心格再额外奖励；输出“每步平均”。
* 活跃度：己方合法走法数或攻击格数（按子型加权），“每步平均”。
* 王安全：王邻域（1~2 格圈）的己方控制 − 对手控制；王侧兵盾完整度；面对开放线的惩罚；是否被重子对线。
* 进攻性：造成将军、制造威胁高价值子的着法、对王邻域格的控制增长、对方阵地的渗透深度（纵深/空间优势）。
* 吃子：实际吃子次数、可吃子占比（可吃子 / 合法着法），和“给对手留下的可吃子机会”（反击暴露）分开两个特征。
* 兵结构：孤/双/后退兵惩罚，通过兵/连兵奖励；按照“局后快照”或“每回合均值”输出，避免重复叠加偏置。

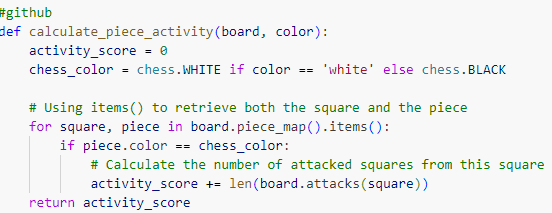
如果你要我，我可以直接把这 6 个函数改成“更合理+归一化+不串色”的版本，并给出和你现有 CSV/PGN 管道兼容的落地代码。



问题：

1. 只求了d4 e4 d5 e5 想实现“越深入敌方分数越高“的思想
2. If board.piece\_at(square)那里 只要有棋子在那里就加分，没有区分黑白手
3. 战局越长天然的这个分数就会更高，需要归一化
4. 可以攻击对方棋子和可以到达某个位置的分数应该不一样





piece\_activity\_score（子力活跃度/机动性）  
想测什么

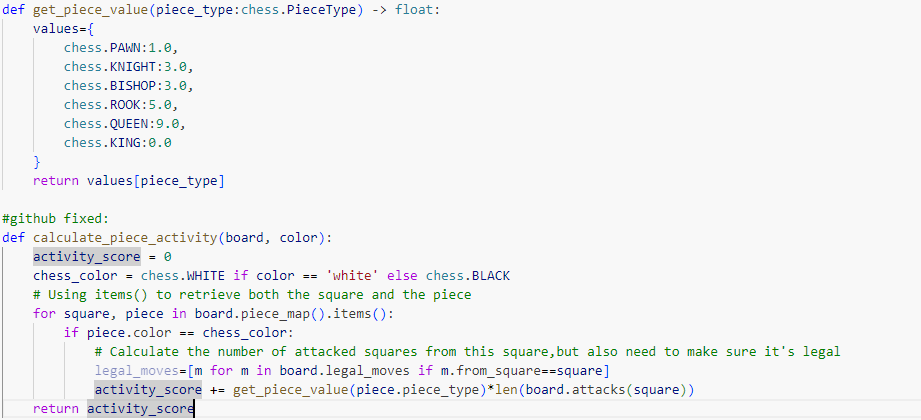
* 己方所有子力的“控制格数量”（mobility），越能攻击/控制的格越多越活跃。

代码如何计算

* 遍历棋盘上“己方”的每一个棋子，求 len(board.attacks(square))，把这些数量累计求和。
* 每步都算一次并累加。

问题/改进

* 这是标准的“机动性”近似指标，但没有区分不同子型的价值（比如后/车控制格权重可能更大），也没有考虑被钉住导致的“名义攻击”与“可行行动”的差别。
* 建议：按子型加权；或把“合法走步数”而非“攻击格数”作为活跃度；并归一化（例如除以己方子数或总步数）。





king\_safety\_score（王安全）  
想测什么

* 王周围的防守是否充分、是否被对手威胁，兼顾兵盾结构。

代码如何计算

* 找到己方王格 king\_square。
* 己方守护王格的棋子数：len(board.attackers(chess\_color, king\_square)) 加到分里。
* 对手攻击王格的棋子数：len(board.attackers(enemy\_color, king\_square)) 从分里减去。
* 额外：遍历己方所有兵，如果该兵格被己方攻击（即被己方其他子力保护），就 +1（本意大概是“兵链/兵被保护”的加分）。

问题/改进

* “被己方攻击的兵 +1”并不直接等于“王安全”，更合理的是评估王翼/后翼的兵盾完整性（如 g/h 兵是否前进、王侧是否有空线、王附近格子是否被己方控制、对手重子是否渗透到王附近等）。
* 建议：
  + 兵盾：王侧（或后翼易位侧）的三兵是否完整、推进是否过度；
  + 王邻近圈（如以王为中心的 2 格曼哈顿距离内）己方控制格 − 对方控制格；
  + 开放/半开放线直对王的惩罚；
  + 敌后/车/马距王的“距离”与攻击线路。
* 仍需归一化。

查询参考：

<https://www.chessprogramming.org/King_Safety>

<https://chessfox.com/how-to-evaluate-progress-in-the-king-safety-objective/>

重新改进的计分逻辑：

1. 是否完成王车易位castling

直接检测走法是否是 O-O / O-O-O。

你可以一次性把所有棋局批量扫一遍，轻松实现。

1. King tropism 敌方棋子距离king的接近程度

找国王格子，计算敌方重子（后、车、马、象）到国王的最短距离

只要会 BFS 或用 chess.square\_distance() 之类函数就行。

检查每个敌方子到己方王的距离：

* 皇后近：大惩罚
* 车近：中等惩罚
* 象马近：小惩罚

1. 防御子数量

定义国王周围一圈（或两圈）的格子，数一数己方控制这些格子的棋子

用 python-chess 的 .attackers(color, square) 就能搞定

己方在王区的子越多，+分。

1. Pawn shield 兵盾

找国王位置 → 判断是否在王翼/后翼 → 检查国王前的 2-3 格是否有己方兵。

 检查王前方的三个兵（短易位后：f,g,h 兵；长易位后：a,b,c 兵）。

 如果这些兵没动，+分；如果缺失或走远了，-分。

 如果缺兵同时旁边有开放线，-更多分。

1. Pawn storm对方兵的攻击推进

要先识别国王在哪一边（王翼/后翼），

然后判断对方兵是否在那一侧推进，且是否过了某个 rank。

需要设计启发式指标，代码逻辑会比较长。

如果对方兵靠近己方王位置两三格内，-分（因为可能开线）。

1. King zone control 王区控制

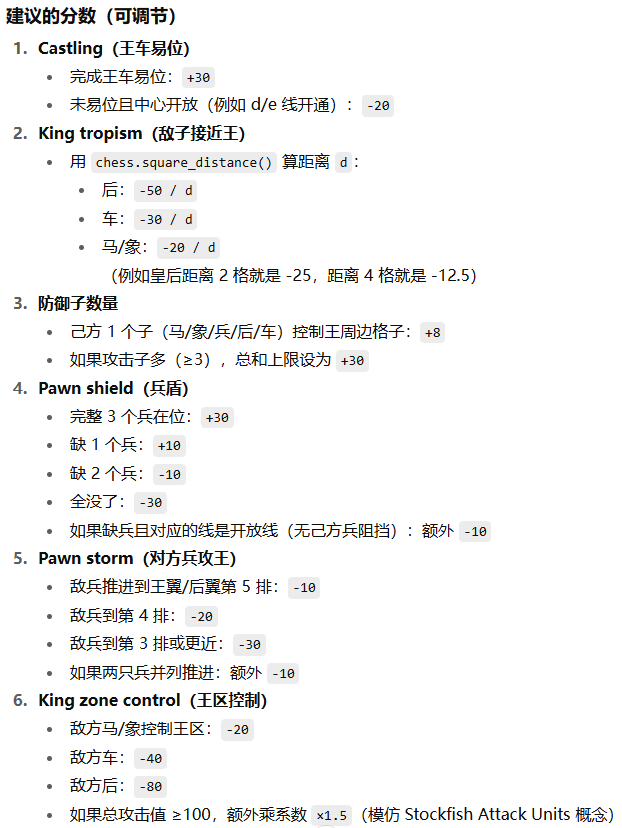
定义国王周围的 zone（通常 8 格 + 扩一圈）

用 .attacks() 算出哪些棋子控制这些格子，比较双方数量

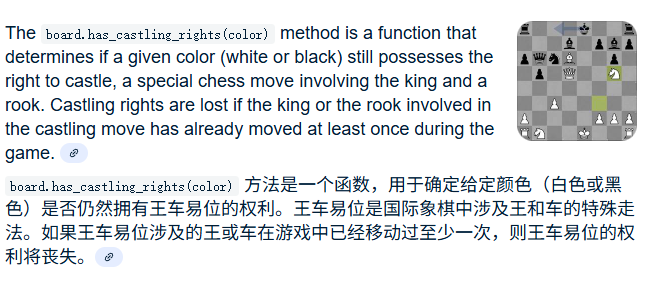
 定义王区（king zone）：王所在格子 + 周围 8 格 + 前方 2–3 格（依颜色不同）。

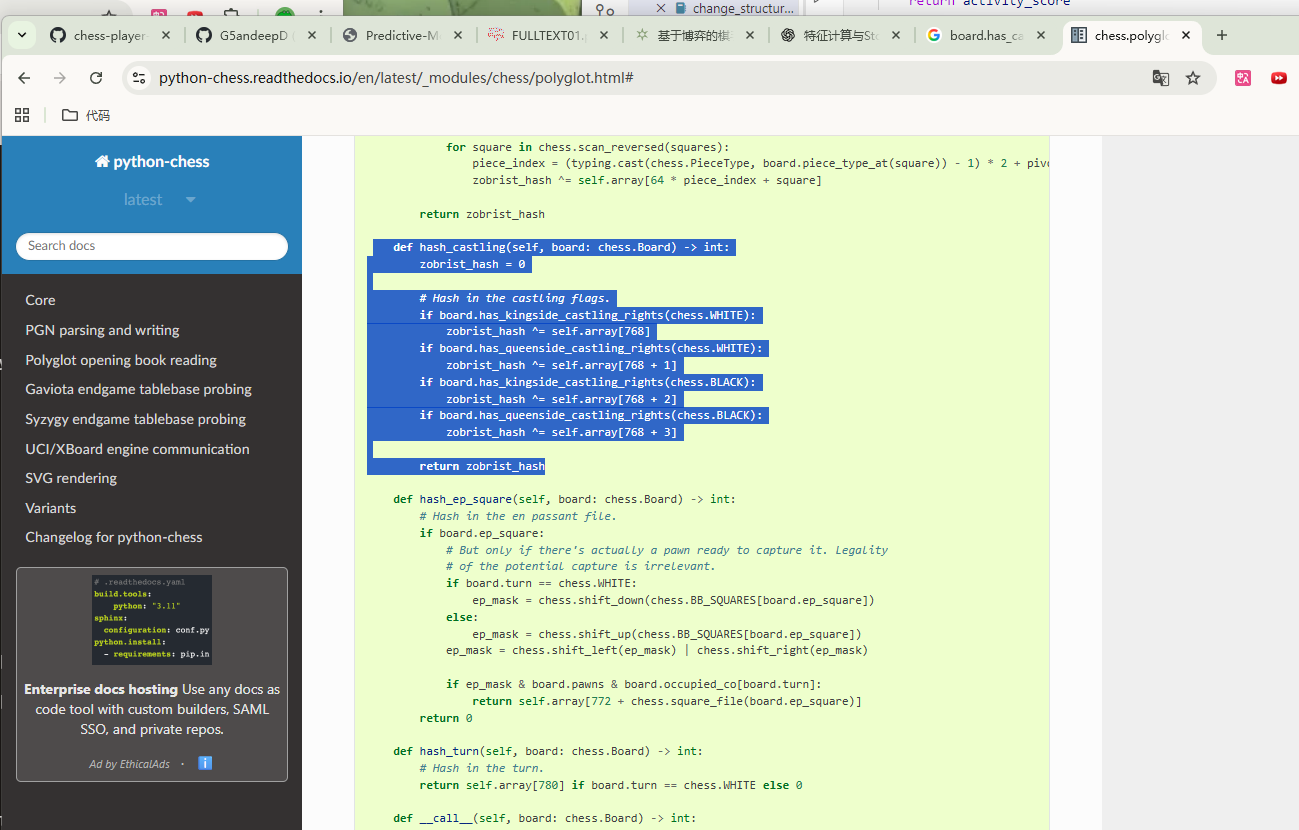
 如果敌方子攻击这个区域，加攻击分（根据子力：马/象≈20，车≈40，后≈80）。

 如果攻击子数达到阈值，可以放大惩罚（像 Stockfish 的 Attack Units 机制）



Castling:





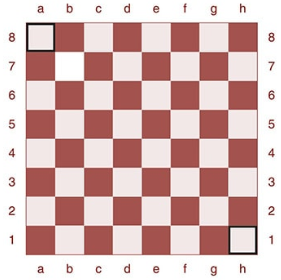
Pawn shield

Rules：

**触发条件**

* 只有当国王还在自己底线的前两排时（白方 rank 0–1，对应第 1–2 排；黑方 rank 6–7，对应第 7–8 排），才计算兵盾。
* 如果国王已经跑到中局或残局更靠前的位置，就返回 0 分，不算兵盾。

**兵盾定义（目标格子）**

* 取国王当前 file（纵列），以及左右各一列，共 3 个 file。
* 在这 3 个 file 上，检查 **国王正前方一排** 的格子：
  + 白方：rank + 1
  + 黑方：rank - 1
* 例如：
  + 白王在 e1 → 检查 d2, e2, f2
  + 白王在 g2 → 检查 f3, g3, h3
  + 黑王在 c8 → 检查 b7, c7, d7
  + 黑王在 f7 → 检查 e6, f6, g6

**开放线定义**

* 一个纵列（file）上如果没有己方兵，则称为开放线。
* 判断方法：沿着 file 全部 8 个格子看一遍，如果有己方兵就不是开放线；否则就是开放线。

**加分 / 扣分规则**

* 每个兵盾格子如果有己方兵 → +5
* 每个兵盾格子如果没有己方兵 → -5
* 如果某个兵盾格子缺兵，且对应 file 是开放线 → 额外 -5
* 最低 -30，最高 +15

| **情况** | **分数** |
| --- | --- |
| 3 个兵都在 | +15 |
| 2 个兵在，1 个缺 | +5 |
| 1 个兵在，2 个缺 | -5 |
| 3 个都缺 | -15 |
| 3 个都缺，且这 3 个 file 都是开放线 | -30 |