

2024 年第五届“大湾区杯”粤港澳 金融数学建模竞赛

题目 基于多元线性回归与均值方差优化的 沪深 300 个股系统性风险预测与回撤控制研究

摘 要：

基于沪深 300 个股的股市系统性风险预测的现实需求，本文建立了多元线性回归模型和均值-方差优化模型，研究股市系统性风险的成因并探讨其对权益类基金回撤控制的影响。首先，计算沪深 300 个股的平均收益率、市场流动性、市场情绪指标这三个风险计量指标；然后，采用多元线性回归方法量化沪深 300 个股的市场的核心风险指标（Alpha 系数、Beta 系数、夏普比率和波动率），建立系统性风险预测模型；其次，基于均值-方差优化方法，构建回撤控制模型，以最大回撤为约束条件优化基金抗风险能力。

针对任务一：首先根据沪深 300 个股市场数据的稳定性和公开性，设定了重点理想化假设，即假设所有市场数据（如收盘价、成交量等）能够准确代表市场的真实波动情况，忽略突发性的非系统性事件的干扰。基于这些条件，建立平均收益率、市场流动性和市场情绪的量化模型，从而准确反映市场的整体状况。首先，平均收益率模型计算每日个股收益，并基于权重合并得到沪深 300 的平均收益率。为减少极端值影响，通过温莎化处理裁剪异常值，从而提升了收益率估计的稳健性。其次，市场流动性模型通过成交量、成交额和换手率三个指标，结合 Min-Max 标准化处理，生成每日的流动性综合指标，以反映市场的活跃度和资金流通状况。最后，市场情绪模型则使用波动性指数（VIX）、资金流向指数（MFI）、风险价值（VaR）等多维指标来量化投资者情绪，捕捉市场的潜在波动风险和交易偏好。模型亮点在于，通过数据清洗和预处理的改进，这些指标共同构建了清晰的市场波动特征，揭示出沪深 300 市场的周期性变化，为后续的系统性风险预测提供了可靠的基础。

针对任务二：首先根据历史收益率数据和多维度风险因素，假设这些数据能够有效反映资产相对市场的表现。基于此条件，考虑到股票的实时变化性质，以及同时股票市场的大量交易信息产生的数据量，选择了运算速度快且便捷的线性回归模型，建立了 Alpha 系数、Beta 系数、夏普比率和波动率四个核心风险计量指标的线性回归模型，用于预测未来市场的整体风险水平。具体而言，Alpha 系数衡量超额收益，Beta 系数反映市场敏感度，夏普比率评估风险调整后收益，波动率则表示市场价格的波动性。求解过程中采用了多元线性回归方法，并利用历史数据对各指标进行拟合，以确定其权重。模型的亮点在于解释性强、计算效率高，线性回归赋予了每个风险指标合理的权重，提供了清晰的风险评估参考。主要结果显示，市场的系统性风险受多重因素共同影响，各指标在预测中起到了不同的作

用，为投资者提供了全面的风险分析工具。投资者和风险管理人员可以直观地了解不同特征的权重及其对整体风险的影响，从而更好地做出决策。

针对任务三：首先根据**市场波动性较大、投资者对回撤容忍度有限**的现实条件，设定理想化假设，即假设基金组合能够通过优化调整适应市场的上下行变化。基于这一条件，建立了**均值-方差模型**，并引入**最大回撤（MDD）**作为风险控制的主要约束条件，以在追求收益的同时有效控制下行风险。采用**优化算法**进行求解，核心思路是在保证最大回撤小于特定阈值的前提下，最大化夏普比率，以优化组合的风险调整收益。**亮点**在于回撤控制策略不仅提升了抗风险能力，还能在市场下行期间保持收益的稳定性。主要结果表明，该模型在市场剧烈波动的情况下能够有效减小权益类基金的回撤，增强了组合的稳健性，为基金风险管理提供了切实可行的量化支持。

针对任务四：首先，根据**投资者收益预期的理性需求**，假设收益预期可通过历史市场表现和无风险利率来合理估算，以帮助投资者设立实际可行的目标。在此基础上，建立了**合理收益预期模型**，以**沪深 300 指数的市场平均收益率**和**10 年期国债的无风险利率**为主要参数计算风险溢价。采用**历史数据分析法**进行求解，模型亮点在于通过**风险溢价（市场平均收益率与无风险利率的差值）**量化市场风险承受水平，从而避免因非理性预期带来的投资风险。算法思路清晰，操作简便。主要结果显示，在当前市场环境下，**沪深 300 成分股**的合理收益预期偏低，反映出其收益较低而无风险利率较高的现状。该模型为投资者提供了**理性决策依据**，有效引导了合理收益期望，有助于规避不切实际的高预期导致的投资失误。

最后，本文对建立的模型进行了相关分析，评价了模型的优缺点并给出了相应的改进方案。本文的模型在**沪深 300 股市系统性风险预测**方面有着广阔的应用前景。

关键词：多元线性回归；均值方差优化；最大回撤控制；VaR 模型；沪深 300 指数

目录

1	问题重述	5
1.1	问题背景	5
1.2	任务重述	5
2	问题分析	6
2.1	问题一的分析	6
2.2	问题二的分析	6
2.3	问题三的分析	6
2.4	问题四的分析	7
3	模型假设	7
4	符号说明	7
5	任务一模型的建立与求解	8
5.1	模型的建立	8
5.1.1	数据处理	8
5.1.2	平均收益率计算与分析	9
5.1.2.1	平均收益率经济意义	9
5.1.2.2	平均收益率算法	9
5.1.2.3	平均收益率的优缺点	10
5.1.3	市场流动性计算与分析	11
5.1.3.1	市场流动性的经济意义	11
5.1.3.2	市场流动性算法	12
5.1.3.3	市场流动性的优缺点	13
5.1.4	市场情绪指标的计算与分析	14
5.1.4.1	市场情绪指标经济意义	14
5.1.4.2	市场情绪指标的算法	14
5.1.4.3	市场情绪指标的优缺点	15
6	任务二模型的建立与求解	17
6.1	模型的建立	17
6.1.1	四个特征的模型建立	17
6.1.1.1	夏普比率	17
6.1.1.2	Beta 系数	17
6.1.1.3	Alpha 系数	18
6.1.1.4	波动率	18
6.1.2	多元线性回归模型的建立	19

6.2 模型的求解	19
7 任务三模型的建立与求解	20
7.1 模型的建立	20
8 任务四模型的建立与求解	21
8.1 模型的建立	21
8.2 模型的求解	22
9 模型的评价与推广	23
9.1 模型的评价	23
9.1.1 模型的优点	23
9.1.2 模型的缺点	24
9.2 模型的推广	24
参考文献	24
A 支撑材料列表	25
B 源程序	29
B.1 问题一平均收益率计算代码	29
B.2 问题一市场流动性代码	31
B.3 问题一市场流动性代码 2	34
B.4 问题一市场情绪指标 matlab 代码 (以 2024 年的数据为例)	36
B.5 问题一市场情绪指标可视化 Matlab 代码	37
B.6 问题二 alpha 系数的计算	39
B.7 问题二 beta 系数的计算	41
B.8 问题二夏普计算	42
B.9 问题二波动性代码	44
B.10 问题二收益率代码	45
B.11 问题二多元线性回归训练代码	46
B.12 问题三模型代码	48
B.13 问题四代码	52

1 问题重述

1.1 问题背景

在全球经济环境日益复杂的今天，系统性风险频繁爆发，对金融市场的稳定性提出了巨大挑战。金融市场不仅受到内部因素的影响，还面临来自宏观经济、地缘政治等外部环境的不确定性。在国际市场动荡的背景下，资本流动加速、波动性增加，常引发价格剧烈波动，导致投资者信心受挫。系统性风险不仅影响短期市场波动，也可能对经济基本面造成长期影响，使得风险预测和管理尤为重要。沪深 300 指数作为 A 股市场的重要基准，代表了市场整体走势，其波动不仅影响国内投资者信心，也关乎市场的安全性与韧性。在中国特色社会主义市场经济下，完善金融市场的风险防控体系，是促进市场健康发展的必然要求，也是保障国家经济安全的重要举措。构建科学的系统性风险预测模型，能在市场波动时提供早期预警，帮助权益类基金在下行中控制回撤，维护投资者利益。通过收益率、流动性和情绪等多维因素构建符合中国特色的风险管理体系，既符合国家防范金融风险的方针，也有助于提升我国资本市场的竞争力。本文将通过对沪深 300 成分股的系统性风险研究，构建一套有效的风险预测与控制模型，为权益类基金的风险管理提供量化支撑。

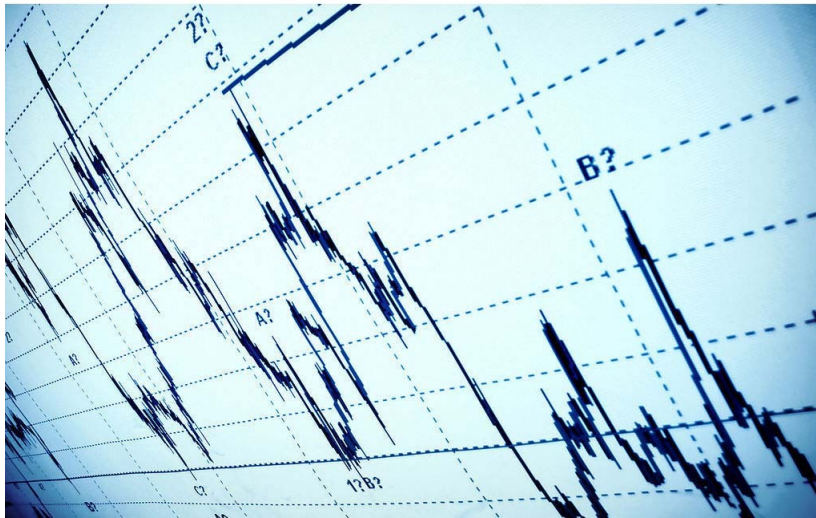


图 1: 沪深 300 指数

1.2 任务重述

本文基于当前金融市场系统性风险难以有效预测、回撤控制面临挑战的背景，围绕市场风险度量、风险预测模型构建、风控体系设计等内容，构建了相应的数学模型，旨在解决以下四个关键任务：

- (1) 针对沪深 300 成分股市场，计算并分析代表整体波动情况的核心风险指标，包括平均收益率、市场流动性和市场情绪，以全面反映市场波动的不同维度。

- (2) 通过设计至少三个核心风险计量指标，结合多种因素，构建一个系统性风险预测模型，帮助预测未来市场的整体风险水平。
- (3) 设立事前风险控制机制，在控制指标阈值的基础上优化权益类基金的回撤，增强市场下行期间基金的抗风险能力。
- (4) 设定合理的收益预期，通过量化不同市场情境下的收益表现，引导投资者树立理性的收益期望，为规避不合理的高预期带来的投资风险提供科学依据。

2 问题分析

2.1 问题一的分析

问题一要求我们对沪深 300 成分股市场的波动情况进行量化分析，通过计算平均收益率、市场流动性和市场情绪等关键指标，全面展示市场的波动特性。平均收益率用于衡量整体回报水平，帮助理解市场的盈利情况；市场流动性反映市场的活跃度和资金充足性，展示交易的便利性和价格的稳定性；市场情绪指标量化投资者的预期和行为趋势，为识别潜在风险提供参考。通过对这些指标的综合分析，我们为后续的风险预测和回撤控制奠定了数据基础，构建了市场波动的整体框架。

2.2 问题二的分析

问题二的任务是构建一个系统性风险预测模型，以量化并预测市场的下行风险。我们分析了沪深 300 市场中的系统性风险因素，重点关注收益率、流动性和情绪等变量，并使用历史数据来量化这些因素对市场风险的影响。通过建立回归模型，我们识别了影响市场整体波动的关键因素，从而有效判断市场的风险暴露程度。该模型帮助我们在市场出现下行趋势前做出合理预测，为投资者提供了规避风险的决策支持。

2.3 问题三的分析

问题三要求我们优化投资组合的最大回撤，以帮助权益类基金在市场波动中控制潜在损失。我们采用均值-方差模型，并引入最大回撤的约束条件，构建了一个在风险控制下优化收益的投资组合模型。最大回撤这一指标帮助我们衡量组合的最大损失风险，通过优化组合权重，我们平衡了风险与收益，使投资组合在市场下行中具备更强的抗风险能力。该模型为基金管理者提供了稳定的风控策略，在市场波动中提供保障。

2.4 问题四的分析

问题四的任务是设定合理的收益预期，帮助投资者在不同市场情境下制定理性的回报目标。我们结合沪深 300 指数的历史收益数据和无风险利率，计算出市场的平均收益率和合理的风险溢价。通过模型量化收益预期，我们为投资者提供了一个客观的回报基准，避免因不合理的收益预期而导致的过度风险行为。合理的收益预期引导投资者在市场波动中保持理性，从而有效规避高风险带来的潜在损失。

3 模型假设

1. 假设市场在大多数时间内是有效的，市场价格能够较快地反映可用信息。
2. 假设股票收益率服从正态分布，尤其是在计算 VaR（风险价值）时。
3. 假设极端值（如极高或极低收益率）并非代表实际市场行为，允许进行温莎化处理。
4. 假设在较长的时间周期内，市场的平均收益率会趋于稳定。
5. 假设市场特征（如收益率、流动性、情绪等）与风险的变化呈线性关系。
6. 假设各股票的每日收益率数据是独立同分布的，使得各时间段内收益率特征一致。

4 符号说明

符号	说明
$P_{i,t}$	该股票在第 t 天的收盘价
N	年份数目
R_t	第 t 天的收益率
\bar{R}	平均收益率
C_{peak}	历史最高累积收益
μ	收益率均值
σ	收益率的标准差
Z_α	置信水平 α 下正态分布的分位数
R_P	投资组合的平均回报率
R_i	资产的回报率
R_m	市场组合的回报率
R_f	无风险利率
σ_p	投资组合收益标准差

* 其余符号在论文中会进行定义

5 任务一模型的建立与求解

5.1 模型的建立

5.1.1 数据处理

附件 1 中包含该赛题应当使用的沪深 300 成分股的数据。数据总体分成两种，一个是沪深 300 成分股在 xxxx 年的行情数据，主要包含开盘价、收盘价、最高价、最低价、成交量和成交额六列数据，对每一支股票在该一年内每一个工作日的行情数据都进行了给出；另一个是在 xxxx 年的沪深 300 成分股信息，包含每一支股票在该一年内的股份权重信息。部分数据见下表，以 2024 年为例（其中，详细数据见附件 1）。

表 1: 沪深 300 成分股 2024 年部分行情数据

证券代码	开盘价	最高价	最低价	收盘价	成交量	成交额
000001	8.77	8.79	8.60	8.60	115836645	1075742252
000002	10.44	10.48	10.15	10.15	81110629	830765500.1
000063	25.87	25.88	24.97	24.99	112461318	2905735778
000069	3.12	3.12	3.02	3.03	46232223	140873446.5

表 2: 沪深 300 成分股 2024 年部分成分股信息

证券代码	证券名称	权重
000001	平安银行	0.00524
000002	万科 A	0.0041
000063	中兴通讯	0.00486
000069	华侨城 A	0.00088

因此，平均收益率、市场流动性、市场情绪这三个指标都应当基于该数据集内的数据来进行计算分析，为了更好地利用数据进行接下来的计算，需要先对数据进行预处理，主要为对缺失数据的处理。

经过数据筛查发现 2014 年的沪深 300 成分股信息的所有权重均为 0，显然该数据存在缺失值。为了填补缺失值，考虑使用成交额近似估算权重，因为成交额可以一定程度上反映了市场对股票的关注度和活跃度。对于股票 i 在某个交易日的权重 ω_i ，可以表示为：

$$\omega_i = \frac{amount_i}{\sum_{j=1}^{300} amount_j} \quad (1)$$

替换后再进行指标值的计算。

5.1.2 平均收益率计算与分析

5.1.2.1 平均收益率经济意义

平均收益率是衡量市场整体盈利水平的重要指标，反映了市场的长期回报预期。较高的平均收益率通常表明市场经济状况良好，企业盈利能力较强，投资者对未来市场前景持乐观态度，因此资金流入市场增加，推动资产价格上涨。此外，平均收益率也是投资者资产配置的依据之一，用于评估不同市场或行业的投资吸引力。若市场收益率持续较低甚至为负，可能预示着经济增长放缓或存在系统性风险。投资者通过观察市场的平均收益率变动，能够及时调整其投资策略，规避潜在风险或把握市场机会。因此，平均收益率不仅是市场健康状况的直接体现，也为经济周期的研判提供了数据支持。

5.1.2.2 平均收益率算法

对于沪深 300 中的任一股票 i ，其在第 t 个交易日的收益率 $R_{i,t}$ 可表示为：

$$R_{i,t} = \frac{P_{i,t} - P_{i,t-1}}{P_{i,t-1}} \quad (2)$$

其中， $P_{i,t}$ 和 $P_{i,t-1}$ 分别为该股在第 t 天和前一交易日的收盘价。

由此得到了个股日收益率，接下来进行加权计算，引入每一支股票的相应权重，当日沪深 300 指数的平均收益率 R_t 可表示为：

$$R_t = \sum_{i=1}^{300} w_i \cdot R_{i,t} \quad (3)$$

其中权重之和： $\sum_{i=1}^{300} w_i = 1$ 。

接下来计算全年的平均收益率。不妨假设全年有 T 个交易日，则在年度内的平均收益率 \bar{R} 为：

$$\bar{R} = \frac{1}{T} \sum_{t=1}^T R_t \quad (4)$$

因此，从 2014 年到 2024 年内的沪深 300 指数多年平均收益率为：

$$\bar{R}_{2014-2024} = \frac{1}{N} \sum_{y=2014}^{2024} \bar{R}_y \quad (5)$$

正常的股票平均收益率计算如上，大体可以反映出市场盈利情况，但由题意知，平均收益率容易受极端值影响，因此考虑对平均收益率的计算进行改进。为了减少极端值的影响，选择对数据进行温莎化处理。

温莎化处理（Winsorization）是一种处理极端值的方法，将数据中过大或过小的值调整到指定的百分位范围内。对于收益率数据，我们可以选择将最低和最高的极端值分别调整到 5% 和 95% 的百分位上。对于平均收益率数据 $R = \{R_1, R_2, \dots, R_n\}$ ，首先计算收益率数据的第 p 个百分位数（如 5%）和第 $100 - p$ 个百分位数（如 95%），记为 R_p 和 R_{100-p} 。因而有

- $R_p = \text{Percentile}(R, p)$
- $R_{100-p} = \text{Percentile}(R, 100 - p)$

接着，将小于 R_p 的值替换为 R_p ，将大于 R_{100-p} 的值替换为 R_{100-p} 。处理后的收益率数据 R^* 表示为：

$$R_i^* = \begin{cases} R_p & \text{if } R_i < R_p \\ R_i & \text{if } R_p \leq R_i \leq R_{100-p} \\ R_{100-p} & \text{if } R_i > R_{100-p} \end{cases}$$

对于每一个 $R_i \in R$ ，执行以上替换即可完成温莎化处理。对温莎化处理后的数据 R^* 求均值，得到温莎化后的平均收益率 $\overline{R^*}$ ：

$$\overline{R^*} = \frac{1}{n} \sum_{i=1}^n R_i^* \quad (6)$$

在进行温莎化处理后的平均收益率可以避免极端值的影响，从而获得更准确的收益率估计。

以 2024 年为例，计算得到的沪深 300 个股的每日平均收益率可视化如图所示：

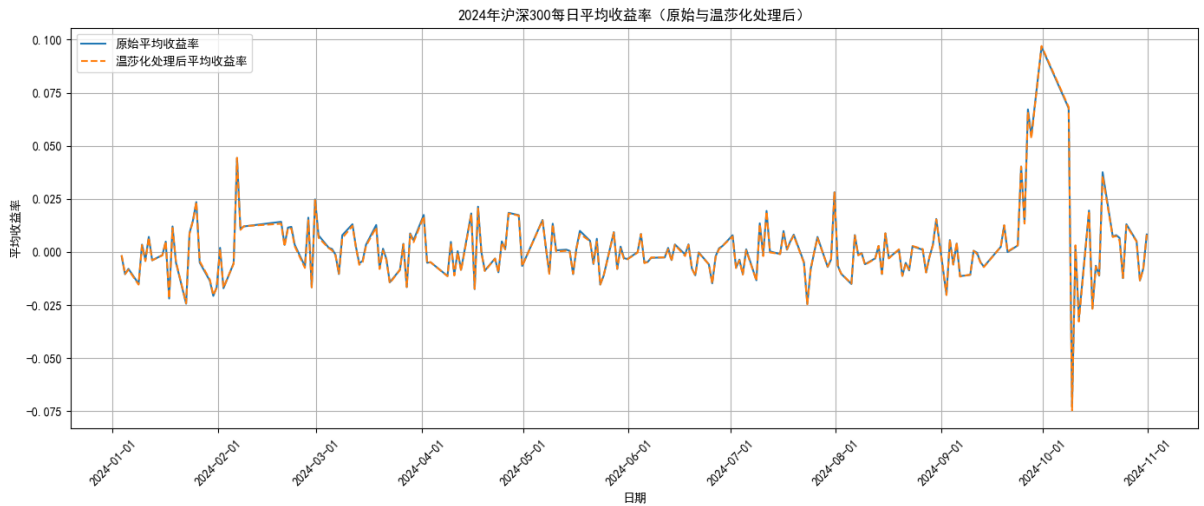


图 2: 沪深 300 指数 2024 年每日平均收益率

最终计算得到的这十年的年平均收益率可视化如图 3 所示：

5.1.2.3 平均收益率的优缺点

优点：

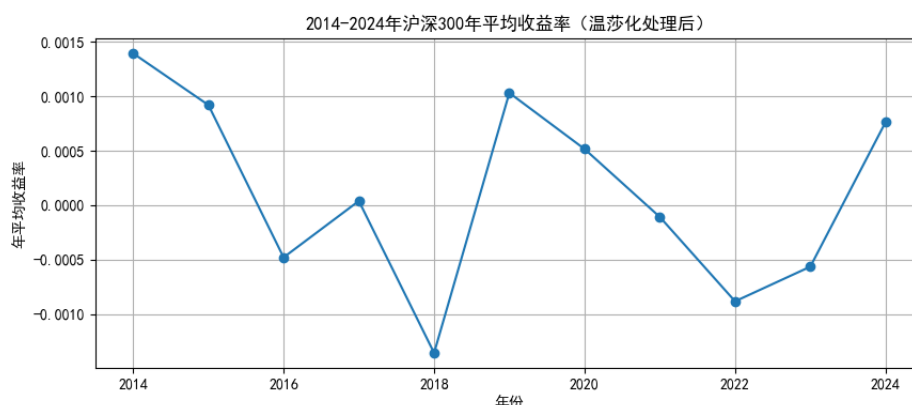


图 3: 沪深 300 指数 2014 年到 2024 年年平均收益率

(1) 经济周期指示性强：

平均收益率在不同的市场阶段具有明确的指示作用，较高的平均收益率通常反映市场的扩张，较低的收益率则可能预示经济放缓或市场衰退。

(2) 直观简洁：

平均收益率作为最基础的收益指标，其计算方法简单明了，易于理解和操作，能够直接反映市场整体的收益水平。

缺点：

(1) 忽略了风险因素：

平均收益率只考虑了回报水平，但未能反映投资风险，特别是在波动较大的市场环境下，该指标可能过于乐观或悲观，导致投资者误判市场风险。

(2) 缺乏动态性：

平均收益率为过去收益的静态反映，难以预测未来的市场变化，特别是在市场趋势反转或波动加剧的情况下，历史平均收益率的参考意义较小。因此，该指标应与其他风险衡量指标结合使用，以获得更全面的市场分析结果。

5.1.3 市场流动性计算与分析

5.1.3.1 市场流动性的经济意义

市场流动性是衡量市场健康状况和资金活跃程度的关键指标，直接影响资产的可交易性和市场的稳定性。高流动性意味着市场上有足够的买卖双方，交易可以迅速达成且成本较低，有助于大额交易的顺利进行，因此被视为市场健康和稳定的象征。相反，低流动性表明市场上缺乏足够的交易深度，可能导致交易困难和价格剧烈波动，特别是在市场出现突发事件或系统性风险时。流动性的波动对投资者情绪也有显著影响，流动性枯竭时市场风险加剧，投资者可能选择减少持仓或转向更安全的资产。因此，市场流动性不仅影响资产价格的形成，还作为市场风险的预警信号。

5.1.3.2 市场流动性算法

基于数据，市场流动性可以由成交量、成交额和换手率三个常用流动性指标来构建综合性的市场流动性指标。这些指标可以分别从市场交易量、资金流动规模以及股票流通情况等角度来衡量市场的流动性。

对于每天的交易数据，我们首先计算市场的总成交量和总成交额。具体来说，对于每一天 t ，市场的总成交量和总成交额可以表示为：

$$V_t = \sum_{i=1}^{N_t} V_{i,t} \quad (7)$$

$$A_t = \sum_{i=1}^{N_t} A_{i,t} \quad (8)$$

其中： N_t 为当天的股票数量， $V_{i,t}$ 为第 i 只股票在当天的成交量， $A_{i,t}$ 为第 i 只股票在当天的成交额。

接着，为了反映市场整体的活跃程度，引入了换手率指标。基于已有的数据，每日的市场换手率 T_t 可近似为：

$$T_t = \frac{V_t}{\sum_j V_j} \quad (9)$$

其中 $\sum_j V_j$ 表示整个市场的成交量总和。这个公式提供了市场每日成交量相对整体成交量的比例，帮助我们评估市场的活跃度。

在得到成交量、成交额和换手率三个指标后，因为不同指标的数量级和量纲可能差异较大，因此需要对数据进行标准化处理，以便后续综合计算。在此，使用 Min-Max 标准化进行处理，将每个指标映射到 $[0,1]$ 的区间。对于每日的成交量、成交额和换手率标准化后的指标，可以表示为：

$$V'_t = \frac{V_t - \min(V)}{\max(V) - \min(V)} \quad (10)$$

$$A'_t = \frac{A_t - \min(A)}{\max(A) - \min(A)} \quad (11)$$

$$T'_t = \frac{T_t - \min(T)}{\max(T) - \min(T)} \quad (12)$$

其中：

- $\max(V)$ 和 $\min(V)$ 分别是所有天成交量中的最大值和最小值；
- $\max(A)$ 和 $\min(A)$ 分别是所有天成交额中的最大值和最小值；
- $\max(T)$ 和 $\min(T)$ 分别是所有天换手率中的最大值和最小值。

最后，把三个标准化指标求平均，得到市场流动性综合指标，因此每日的市场流动性综合指标 L_t 定义为：

$$L_t = \frac{V'_t + A'_t + T'_t}{3} \quad (13)$$

对于年度流动性指标，我们可以将每天的市场流动性指标 L_t 进行平均，得到年度流动性指标 L_y ：

$$L_y = \frac{1}{M} \sum_{t=1}^M L_t \quad (14)$$

其中 M 为年度内的总交易天数。年度流动性指标能够提供年度整体的市场流动性概况。

基于上述方法，用 python 变成计算得到了这十年市场流动性指标的可视化如图所示：

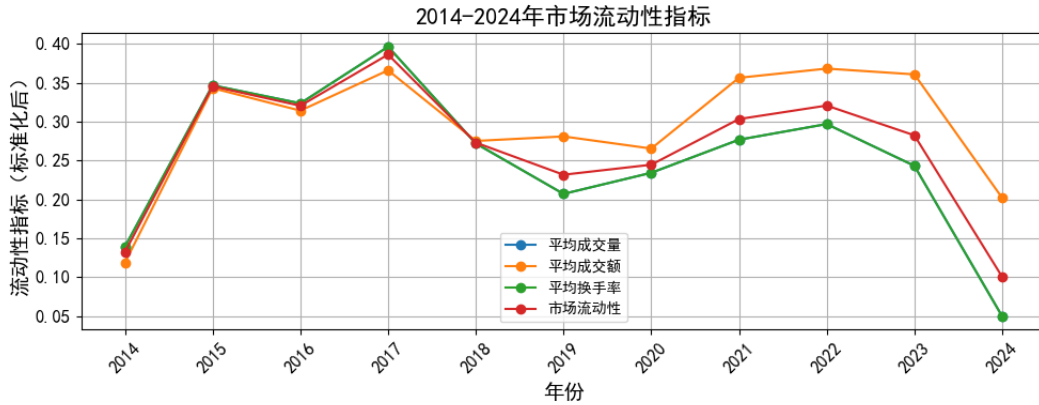


图 4: 十年市场流动性指标

5.1.3.3 市场流动性的优缺点

优点：

(1) 多维度反映流动性：

该指标结合了成交量、成交额和换手率三大因素，不仅能够捕捉市场整体交易规模，还能考虑到个体交易的活跃程度。这种多维度整合增强了流动性指标的全面性，使其更具解释力。

(2) 便于实时监控与预警：

由于流动性指标可以逐日计算，因此能够敏锐地捕捉到市场流动性的实时变化，为投资者和管理者提供及时的市场信息，以便进行动态调整和风险管理。

(3) 去除量纲影响，易于比较：通过 Min-Max 标准化，消除了不同指标间的数量级差异，使得不同时间段或不同市场间的流动性状况可以直接进行对比，便于横向和纵向分析。

缺点：

(1) 未考虑深层市场结构：

成交量、成交额和换手率虽然可以反映市场的表面流动性，但它们无法揭示深层的市场结构，如订单簿的深度、买卖双方的供需平衡情况等。因此，在市场剧烈波动时，表面流动性可能较高，但真实流动性却不足。

(2) 忽略了其他影响流动性的因素：

流动性不仅由成交量、成交额和换手率决定，还受到宏观经济环境、市场预期、政策因素等多方面影响。例如，市场情绪和政策变化可能会迅速影响流动性，但在指标中难以直接反映这些外部变量。

5.1.4 市场情绪指标的计算与分析

5.1.4.1 市场情绪指标经济意义

市场情绪指标是反映投资者对市场未来走势预期的综合性指标，量化了投资者的情绪变化和 risk 偏好。市场情绪高涨时，投资者倾向于增加投资，推动市场上行，形成“牛市”情绪；而情绪低迷时，市场上避险情绪上升，资金流向低风险资产，可能导致市场回调。情绪指标还受政策变化、经济数据等因素的影响，因此也能够反映政策效应及经济基本面的影响。例如，当政府出台刺激政策时，市场情绪往往上升，而经济数据不佳则可能引发悲观情绪。市场情绪指标为投资者提供市场波动的前瞻性信号，帮助其识别潜在的市场风险或机遇，是判断市场走势的重要辅助工具。

5.1.4.2 市场情绪指标的算法

通常情况下，市场情绪可从新闻文字等内容中运用自然语言处理技术提取得到，但基于已有的沪深 300 指数数据是难以得到的。通过阅读文献，市场情绪指标可以从波动性指数（VIX）、资金流向指数（MFI）、投资者情绪指数（ISI）以及 VaR 模型四个指标进行量化。

(1) 波动性指数（VIX）最早由芝加哥期权交易所（CBOE）在 1993 年引入，用于衡量市场的隐含波动率。VIX 被称为“恐慌指数”，因为其数值越高，表明市场对未来的不确定性越强，投资者的恐慌情绪也越浓厚。VIX 的计算基于一段时间内的收益率波动，可以近似反映市场对未来价格变化的预期。波动性指数的计算公式为：

$$VIX = \sqrt{\frac{1}{n-1} \sum_{t=1}^n (R_t - \bar{R})^2} \quad (15)$$

通过日内价格的波动情况来反映市场的紧张程度。对于沪深 300 指数，在市场动荡时，VIX 会显著上升，表明市场对未来不确定性的预期增强。

(2) 资金流向指数（MFI）是一种基于成交量和价格变化的情绪指标。与相对强弱指数（RSI）类似，MFI 通过成交量来加权价格的变化趋势，因此能够反映市场中资金的流入和流出。MFI 的取值在 0 到 100 之间，数值越高表示资金流入量较大，市场情绪偏向乐观。MFI 的计算公式如下：

$$MFI = 100 - \frac{100}{1 + \frac{\sum \text{Positive Money Flow}}{\sum \text{Negative Money Flow}}} \quad (16)$$

其中，正的资金流入量是指典型价格上升时的交易额，负的资金流出量是指典型价格下降时的交易额。在分析沪深 300 指数数据的时候，MFI 可以帮助识别市场参与者的情绪强度，判断是否出现“过度买入”或“过度卖出”的现象。例如，当 MFI 超过 80 时，可能意味着市场处于高位，存在回调风险；而 MFI 低于 20 则可能表明市场已被超卖，有反弹潜力。

- (3) 投资者情绪指数（ISI）是一种基于开盘价、收盘价和成交量的指标，用于衡量投资者在日内的情绪偏向。ISI 通过价格变化加权成交量，反映出市场中买卖双方的力量对比。ISI 的计算公式如下：

$$ISI = \frac{\sum (Close - Open) \times Volume}{Total Volume} \quad (17)$$

该指标反映了投资者对日内行情的反应。当 ISI 为正值时，表明市场总体处于买入情绪；若 ISI 为负值，则说明市场存在卖出压力。在市场剧烈波动时，ISI 可以帮助判断日内市场情绪的变化，例如在大盘高开低走时 ISI 值会显著降低，反映出投资者情绪的转向，进而可以为短线操作提供参考。

- (4) 风险价值（VaR）模型是金融风险管理的核心工具之一，用于估计在特定置信水平下的最大潜在损失。VaR 通过衡量投资组合在未来一段时间内的最大损失来反映市场风险水平，通常适用于极端市场情绪下的风险评估。假设市场价格收益率服从正态分布，VaR 的计算公式为：

$$VaR_{\alpha} = \mu + \sigma \times Z_{\alpha} \quad (18)$$

在市场极端情况下，VaR 能够提供一个“最坏情况”下的损失估计，帮助投资者衡量市场的潜在风险。对于沪深 300 指数来说，较高的 VaR 值意味着市场波动性较大，潜在损失风险增加。

四种指标由于维度和表示方法不同，不适合均一化分析处理，因而对于市场情绪指标我们使用这四个指数进行量化。通过 python 计算，导入数据后得到了从 2014 年到 2024 年每一天的市场情绪指标，可视化如图 5 所示。

5.1.4.3 市场情绪指标的优缺点

优点：

- (1) 直接反映市场恐慌情绪：

VIX 可以快速反映市场的不确定性，被广泛认为是衡量市场风险情绪的有效指标，其能够在市场出现剧烈下跌时有效地捕捉市场恐慌情绪，成为风险管理中的重要工具。

- (2) 能够反映市场的资金流动情况：

MFI 结合了成交量和价格波动，因此能够更全面地捕捉市场情绪的变化趋势。

沪深300市场情绪指标随时间的变化

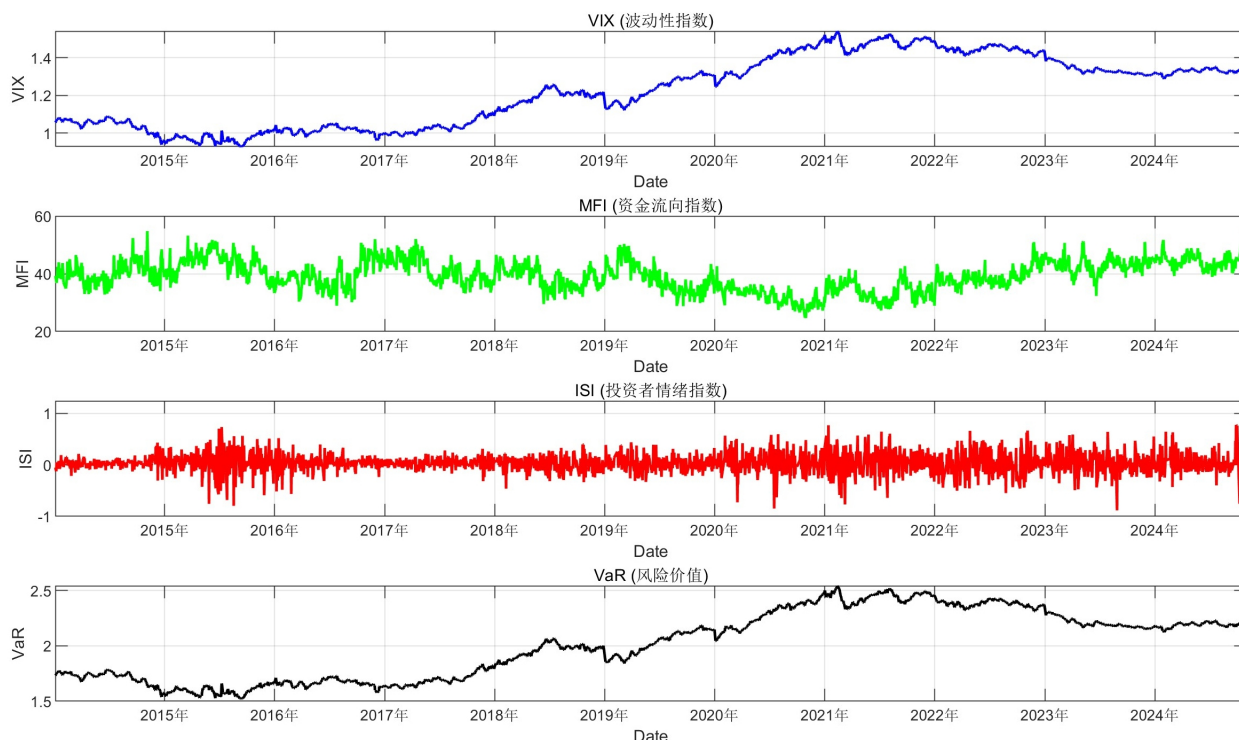


图 5: 沪深 300 指数近十年市场情绪指标

(3) 适用于日内情绪分析:

ISI 能够捕捉日内的情绪变化, 通过开盘和收盘价加权成交量来反映投资者的日内情绪偏向, 可以在分析短期市场情绪波动中具有较高的有效性。

(4) 能够量化极端情绪下的市场风险:

VaR 模型在特定置信水平下提供了极端市场风险的定量化测量, 是金融风险管理的关键指标, 帮助金融机构识别并评估尾部风险。

缺点:

(1) 对极端市场事件的敏感性不足:

在市场出现极端事件时, ISI 由于仅关注日内数据, 可能无法捕捉到事件带来的全面影响。

(2) 假设局限性导致的风险低估:

VaR 通常假设收益分布为正态分布, 但市场收益常呈现厚尾和偏态特征, 这可能导致 VaR 低估极端风险。

6 任务二模型的建立与求解

6.1 模型的建立

利用第一问得到的三个特征的数据，考虑到股票的实时变化性质，同时股票市场的大量交易信息产生的数据量，我们选择运算速度快且便捷的线性回归模型，并通过将从股票数据中提炼的 4 种指标通过线性回归的方式得到权重，由此获得 4 种特征的相关值，以准确预测风险评分。

6.1.1 四个特征的模型建立

6.1.1.1 夏普比率

夏普比率（Sharpe Ratio）是一项重要的风险调整后收益指标，用于评估投资的回报相对于其风险的情况，其计算方式如下：

$$\text{Sharpe Ratio} = \frac{R_p - R_f}{\sigma_p} \quad (19)$$

6.1.1.2 Beta 系数

Beta 系数是金融和投资中用来衡量个别资产（如股票）相对于整体市场风险敏感性的指标。它主要反映了该资产的系统性风险，也就是由于市场波动带来的不可分散风险。Beta 系数通常用于评估资产相对于市场基准的波动情况，从而帮助投资者了解该资产的风险水平。其具体计算方式如下：

$$\beta = \frac{\text{Cov}(R_i, R_m)}{\text{Var}(R_m)} \quad (20)$$

其中， $\text{Cov}(R_i, R_m)$ 表示资产回报率与市场回报率之间的协方差，表示两者的共同变动情况， $\text{Var}(R_m)$ 表示市场回报率的方差，表示市场的整体波动性。

通过比较个别资产与整体市场的回报率波动性，Beta 系数可以反映该资产对市场波动的敏感度。根据 Beta 值的大小，可以得到：

- 当 $\beta = 1$ 时，表示该资产与市场的波动方向和幅度基本一致。即当市场上涨或下跌 1% 时，资产也相应地涨或跌 1%。这是一个市场“中性”资产的表现。
- 当 $\beta > 1$ 时，该资产比市场更为敏感，属于高波动性资产。例如，Beta 为 1.5 的资产在市場上涨或下跌 1% 时，通常会上涨或下跌 1.5%。这种资产通常会带来更高的潜在回报，但伴随更高的市场风险。
- 当 $\beta < 1$ 时，该资产的波动性低于市场波动性，即它对市场波动的反应较弱。例如，Beta 为 0.7 的资产在市場上涨或下跌 1% 时，通常只会上涨或下跌 0.7%。这种资产在市場波动中表现更稳定，是低风险投资的代表。

- 当 $\beta < 0$ 时，表示该资产与市场呈反向关系，通常在市场下跌时该资产上涨，在市场上涨时下跌。虽然这种情况不多见，但某些避险资产（例如黄金）有可能表现出负 Beta 的特性。

6.1.1.3 Alpha 系数

Alpha 系数是一个投资绩效指标，用于衡量投资组合、基金或单个资产在剔除市场整体表现的影响后，所取得的超额回报。简单来说，Alpha 系数表示一个资产或投资组合相较于预期表现的“超额收益”。其计算通常基于资本资产定价模型 CAPM，计算公式如下：

$$\alpha = R_i - (R_f + \beta \times (R_m - R_f)) \quad (21)$$

其中， R_i 表示资产或投资组合的实际回报率， R_f 表示无风险利率，通常用国债等低风险资产的收益率表示， β 表示该资产或投资组合的 β 系数，反映其与市场的波动关系， α 表示 Alpha 系数，即资产的超额回报。

根据 Alpha 的值，可以得到以下结果：

- 当 $\alpha > 0$ 时，表示该资产或投资组合的表现优于市场预期，获得了超额回报。投资者通常希望获得正 Alpha，因为这表明投资获得了超过市场平均水平的回报。
- 当 $\alpha = 0$ 时，表示该资产或投资组合的表现与市场预期一致，没有产生超额收益或亏损。这意味着投资的表现与市场平均水平持平。
- 当 $\alpha < 0$ 时，表示该资产或投资组合的表现低于市场预期，未能达到市场平均回报。这表明投资决策效果不佳，未能产生应有的收益。

6.1.1.4 波动率

波动率是衡量资产价格在特定时间内变动幅度的指标，反映了资产的价格波动程度。波动率是投资风险的一个重要衡量标准，因为它表明了投资的收益或损失可能有多大。通常，波动率越高，意味着该资产的风险越大，同时也可能带来更高的潜在回报。

其基于资产的收益率来计算，在第一问我们已经计算得到了每一天的收益率以及一段时间内收益率的均值。接下来补充计算收益率的标准差，用于表示数据围绕均值的离散程度：

$$\sigma = \sqrt{\frac{1}{N-1} \sum_{t=1}^N (r_t - \mu)^2} \quad (22)$$

其中， N 是观测数据的数量， r_t 是第 t 天的收益率， μ 是收益率的均值。

进而我们可以计算年化波动率：

$$\text{Annualized Volatility} = \sigma \times \sqrt{252} \quad (23)$$

其中，252 表示一年中交易的天数（约 252 天）。对于月收益率或周收益率，可以乘以相应的因子（如 $\sqrt{12}$ 或 $\sqrt{52}$ ）。

6.1.2 多元线性回归模型的建立

基于这四个特征，建立多元线性回归模型如下，通过已有的十年数据和计算得到的指标，可以求解模型的权重。

$$y = k_{\alpha}x_{\alpha} + k_{\beta}x_{\beta} + k_{vol}x_{vol} + k_{sha}x_{sha} + b \quad (24)$$

6.2 模型的求解

通过 python 建模求解，得到的四个 k 值如下

表 3: 线性回归模型求解结果

k_{α}	k_{β}	k_{vol}	k_{sha}	b
35.6849	0.0740	-0.5346	5.1175	1

将拟合模型与特征数据结合进行可视化，得到如下四图所示：

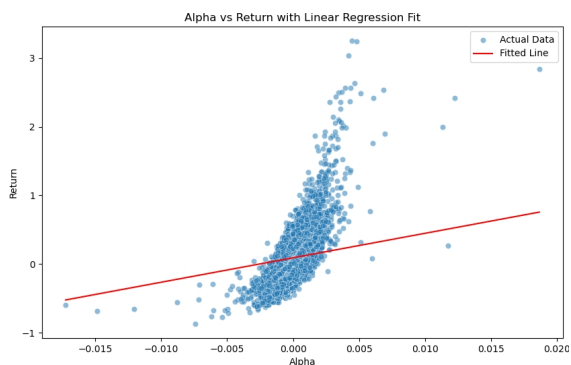


图 6: Alpha 系数特征的拟合可视化

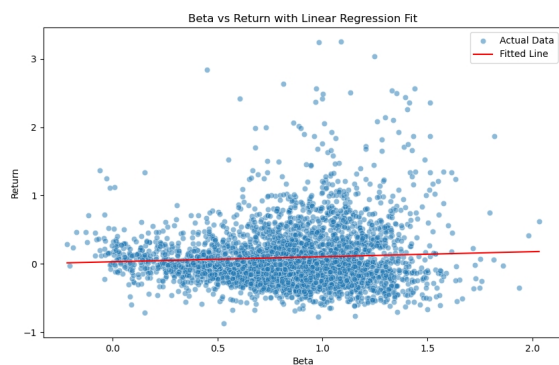


图 7: Beta 系数特征的拟合可视化

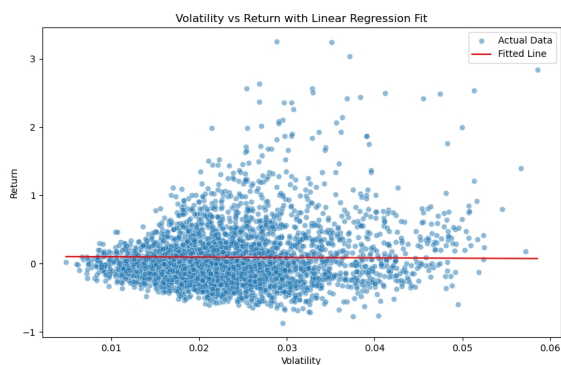


图 8: 波动率特征的拟合可视化



图 9: 夏普比率系数特征的拟合可视化

最后使用模型计算了沪深 300 指数这十年的每一天的风险评估指数，见附件。

此模型具有高度的可解释性，使得我们能够清晰地看到每个特征（如 Alpha 系数、Beta 系数、夏普比率和波动率）对最终风险评估分数的具体贡献。投资者和风险管理人員可以直观地了解不同特征的权重及其对整体风险的影响，从而更好地做出决策。

7 任务三模型的建立与求解

7.1 模型的建立

对于此问，我们考虑将最大回撤 <0.7 作为一个约束条件，使用均值方差优化模型来对收益进行优化。从 2014 年至今的沪深 300 每日收盘价数据可以计算收益率，计算如下：

$$R_t = \frac{P_t - P_{t-1}}{P_{t-1}} = \log(P_t) - \log(P_{t-1}) \quad (25)$$

接着将所有股票的日收益率汇总成一个矩阵，行代表日期，列代表股票。

最大回撤 (MDD) 是指投资组合在特定时间段内的最大资金回撤幅度。其计算过程如下：

$$\text{MDD} = \max_t \left(\frac{C_{\text{peak}} - C_t}{C_{\text{peak}}} \right) \quad (26)$$

其中， C_{peak} 为历史最高累积收益， C_t 为第 t 天的累积收益。

为了更好量化风险调整后的收益，我们引入了夏普比率，其于衡量投资回报的风险调整后收益，具体计算方式如下：

$$\text{SR} = \frac{E[R_p] - R_f}{\sigma_p} \quad (27)$$

其中， $E[R_p]$ 为投资组合预期收益， R_f 为无风险利率， σ_p 为投资组合收益的标准差。

使用均值-方差模型进行投资组合优化。目标函数为最大化夏普比率，同时引入约束条件确保最大回撤小于 0.7。收益的期望值：

$$E[R_p] = w^T \mu \quad (28)$$

投资组合的方差：

$$\sigma_p^2 = w^T \Sigma w \quad (29)$$

其中， w 为投资组合权重向量， μ 为股票收益率向量， Σ 为收益率的协方差矩阵。

最后，定义优化函数：

$$\text{Minimize} - \frac{E[R_p] - R_f}{\sqrt{w^T \Sigma w}} \quad (30)$$

为确保最大回撤不超过 0.7，定义非线性约束。通过计算投资组合的累积收益：

$$C_t = \prod_{i=1}^t (1 + R_i) - 1 \quad (31)$$

并且计算最大回撤：

$$\text{MDD} = \max_t \left(\frac{C_{\text{peak}} - C_t}{C_{\text{peak}}} \right) \leq 0.7 \quad (32)$$

对数据优化后输出权重、预期收益和波动率，利用优化后的投资组合进行历史数据回测，观察实际的最大回撤是否符合要求。

8 任务四模型的建立与求解

8.1 模型的建立

散户在投资时对收益预期的非理性往往源于没有依据的预期。通过历史数据分析，可以设定更为合理的收益预期，公式如下：

$$R_m = \left(\prod_{i=1}^N (1 + R_i) \right)^{\frac{1}{N}} - 1 \quad (33)$$

其中， R_i 是第 i 年的收益率， N 是投资的年数。

无风险收益率定义如下：

$$R_f = \text{Average}(R_f) \quad (34)$$

其中， R_f 是 10 年期国债的历史收益率的平均值。

十年国债通过网上查阅资料，我们得到了数据，可视化如下：

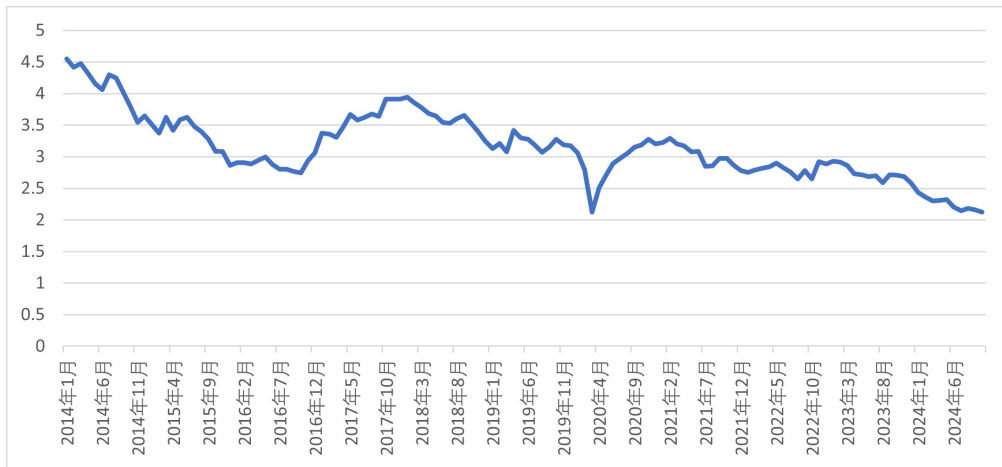


图 10: 10 年国债数据

风险溢价：

$$RP = R_m - R_f \tag{35}$$

最终定义合理收益预期：

$$E(R) = R_f + RP \tag{36}$$

8.2 模型的求解

通过建模计算得到结果如下：

表 4: 第四问指标求解结果

市场平均收益率	无风险收益率	风险溢价	合理收益预期
0.30%	3.14%	-2.84%	0.30%

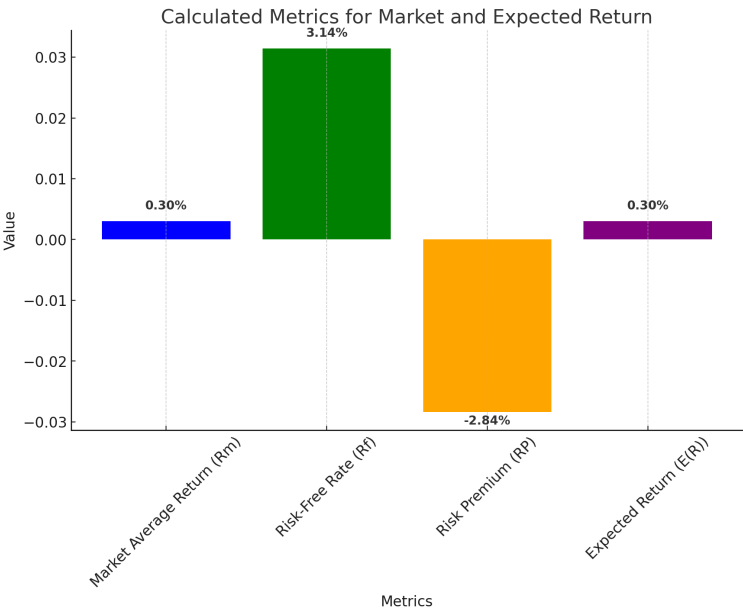


图 11: 各指标结果对比

结果显示，合理的收益预期较低，这是因为市场平均收益率明显低于无风险收益率，导致负的风险溢价。这可能反映了沪深 300 成分股在这个时间段的收益相对较低，而国债收益率较高。

把 2014 年至 2024 年间沪深 300 指数的年化收益率与 10 年期国债收益率（作为无风险收益率）进行对比，得到如下可视化图：

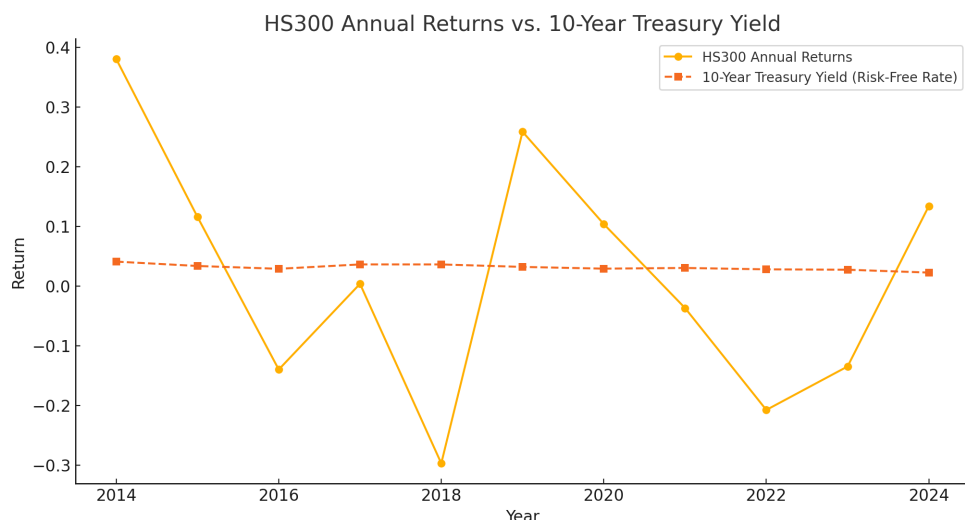


图 12: 10 年国债数据与沪深 300 指数年化收益率对比

其中，黄线（HS300 Annual Returns）显示了沪深 300 指数的年化收益率，可以看到该收益率在不同年份之间波动较大，有些年份出现了明显的负收益。橙色虚线（10-Year Treasury Yield）则代表了 10 年期国债的年平均收益率，整体波动较小，较为平稳，保持在正收益区域。

9 模型的评价与推广

9.1 模型的评价

9.1.1 模型的优点

1. 模型设计上不仅适用于沪深 300 指数的分析框架，还具备跨市场、跨资产类别的适应性。通过适当调整模型的参数和风险控制机制，该模型能够适用于其他市场以及不同的资产类别，为不同市场的投资决策提供支持。
2. 模型采用温莎化等数据处理技术，对极端值进行合理的控制和修正。这使得模型在面对异常数据或极端市场波动时表现更稳定，有效减少了异常波动带来的误导，为投资者提供了更可靠的市场风险评估。
3. 模型基于均值-方差框架，在收益优化的同时保持了风险控制，达到了风险和收益的平衡。通过最大化夏普比率这一目标，模型实现了风险调整后的最佳回报，为投资者提供了有效的风险管理工具，确保其收益在可接受的风险范围内最大化。
4. 风险预测使用多元线性回归模型，模型结构简洁且结果透明，能够清晰展示每个特征对风险评估分数的具体影响。相比于复杂的非线性模型，线性回归的可解释性更强，更便于投资者理解和分析。

9.1.2 模型的缺点

1. 模型假设股票收益率服从正态分布，但金融市场中的收益率分布往往呈现“肥尾”特征，即极端值出现的概率较高。这种偏离正态分布的特性可能导致模型低估极端风险的发生概率，使得模型的风险预测在极端情况下不够可靠。
2. 模型未充分考虑宏观经济变量（如利率、通胀、失业率等）对市场波动的影响，而这些因素对系统性风险的影响往往显著。忽视宏观经济因素的变化可能导致模型的预测结果不够全面，尤其在经济周期转换时存在较大偏差。

9.2 模型的推广

在实际金融市场中，系统性风险和投资回报并非局限于特定市场或指数。我们的模型可以通过以下几种方式进行推广：

(1) 跨市场和资产类别的适用性：

本模型可推广至其他股票市场和资产类别，通过调整参数适应不同的市场环境，如标普500指数或债券、外汇市场。不同市场中的风险特征和流动性指标的调整，使模型在多样化的投资环境下具备较强的适应性。

(2) 宏观经济变量的引入：

引入通胀、利率等宏观经济变量能增强模型的解释力，帮助模型在不同的经济周期中更好地反映市场风险。这一调整使模型具备更广泛的适应性和预测能力，适用于多种经济情境。

(3) 动态风险控制机制：

将固定的最大回撤阈值调整为动态设置，能根据市场波动灵活调整风险控制强度。该机制在市场波动剧烈时提升保护性，在平稳期放宽限制，平衡收益和风险，实现更智能的风控策略。

参考文献









- [1] 孙胜达. 投资者情绪、股票市场流动性及波动性的时变关联性研究 [D]. 沈阳: 沈阳工业大学, 2022.
- [2] 张浩博. 我国股票市场流动性与流动性风险溢价研究 [D]. 长春: 吉林大学, 2016.
- [3] 吴述金, 毕俊娜, 张中兴, 等. 金融建模 [M]. 王铮, 主编. 北京: 科学出版社, 2022.
- [4] Trading Economics. 中国政府债券收益率 [EB/OL].













<https://zh.tradingeconomics.com/china/government-bond-yield>, 访问日期: 2024年11月7日.

附录












A 支撑材料列表












i. 任务一代码与结果:












-  市场情绪指标代码
-  average_profit
-  data_processing.ipynb
-  market_fludity_2
-  market_fluidity
-  平均每日收益率_2014_2024
-  市场流动性每日指标
-  市场情绪指标












-  T1Sentiment_plot
-  T1Sentiment2014
-  T1Sentiment2015
-  T1Sentiment2016
-  T1Sentiment2017
-  T1Sentiment2018
-  T1Sentiment2019
-  T1Sentiment2020
-  T1Sentiment2021
-  T1Sentiment2022
-  T1Sentiment2023
-  T1Sentiment2024












ii. 任务二代码与结果:

 alpha_coefficients_2014
 alpha_coefficients_2015
 alpha_coefficients_2016
 alpha_coefficients_2017
 alpha_coefficients_2018
 alpha_coefficients_2019
 alpha_coefficients_2020
 alpha_coefficients_2021
 alpha_coefficients_2022
 alpha_coefficients_2023
 alpha_coefficients_2024

 beta_coefficients_2014
 beta_coefficients_2015
 beta_coefficients_2016
 beta_coefficients_2017
 beta_coefficients_2018
 beta_coefficients_2019
 beta_coefficients_2020
 beta_coefficients_2021
 beta_coefficients_2022
 beta_coefficients_2023
 beta_coefficients_2024

 total_return_results_2014
 total_return_results_2015
 total_return_results_2016
 total_return_results_2017
 total_return_results_2018
 total_return_results_2019
 total_return_results_2020
 total_return_results_2021
 total_return_results_2022
 total_return_results_2023
 total_return_results_2024

 sharpe_ratio_results_2014
 sharpe_ratio_results_2015
 sharpe_ratio_results_2016
 sharpe_ratio_results_2017
 sharpe_ratio_results_2018
 sharpe_ratio_results_2019
 sharpe_ratio_results_2020
 sharpe_ratio_results_2021
 sharpe_ratio_results_2022
 sharpe_ratio_results_2023
 sharpe_ratio_results_2024

-  volatility_results_2014
-  volatility_results_2015
-  volatility_results_2016
-  volatility_results_2017
-  volatility_results_2018
-  volatility_results_2019
-  volatility_results_2020
-  volatility_results_2021
-  volatility_results_2022
-  volatility_results_2023
-  volatility_results_2024

iii. 任务三代码与结果:

 T3

iv. 任务四代码与结果:

 T4

B 源程序

B.1 问题一平均收益率计算代码

```
1 import numpy as np
2 import pandas as pd
3 import os
4
5 data_folder = "./沪深300成分股的数据/"
6
7
8 # 温莎化处理函数
9 def winsorize(series, lower_percentile=5, upper_percentile=95):
10     lower_limit = np.percentile(series, lower_percentile)
11     upper_limit = np.percentile(series, upper_percentile)
12     return np.clip(series, lower_limit, upper_limit)
13
14
15 # 加载数据的函数，包括成分股信息和行情数据的合并
16 def load_data(year):
17     stock_info_path = os.path.join(data_folder, f'hs300stocks_{year}.csv')
18     kdata_path = os.path.join(data_folder, f'hs300stocks_kdata_{year}.csv')
19
20     # 读取CSV文件
21     stock_info = pd.read_csv(stock_info_path)
22     kdata = pd.read_csv(kdata_path)
23
24     # 合并成分股信息和每日行情数据
25     data = pd.merge(kdata, stock_info[['code', 'weight']], on='code')
26     return data
27
28
29 # 计算每日收益率的函数
30 def calculate_daily_returns(data):
31     # 计算前一日收盘价
32     data['prev_close'] = data.groupby('code')['close'].shift(1)
33     # 计算每日收益率
34     data['daily_return'] = (data['close'] - data['prev_close']) / data['prev_close']
35     # 去掉因移动计算而产生的NaN值
36     data = data.dropna(subset=['daily_return'])
```

```

37 return data
38
39
40 # 计算每日温莎化处理后的平均收益率
41 def calculate_daily_average_return(data):
42     # 温莎化处理每日收益率，消除极端值的影响
43     data['daily_return_winsorized'] = data.groupby('time')['daily_return'].transform(
44         lambda x: winsorize(x, lower_percentile=5, upper_percentile=95)
45     )
46     # 按日期计算温莎化后的每日平均收益率
47     daily_index_returns =
48         data.groupby('time')['daily_return_winsorized'].mean().reset_index()
49     daily_index_returns.columns = ['Date', 'AverageDailyReturn'] # 重命名列
50     return daily_index_returns
51
52 # 主函数，遍历所有年份并计算每日平均收益率
53 def main():
54     all_daily_returns = pd.DataFrame() #
55         创建一个空的数据框以存储所有年份的每日平均收益率
56
57     # 遍历2014至2024年
58     for year in range(2014, 2025):
59         print(f"正在处理{year}年的数据...")
60         # 加载数据
61         data = load_data(year)
62         # 计算每日收益率
63         data = calculate_daily_returns(data)
64         # 计算每日平均收益率
65         daily_returns_df = calculate_daily_average_return(data)
66         # 将每年的每日收益率数据合并
67         all_daily_returns = pd.concat([all_daily_returns, daily_returns_df],
68             ignore_index=True)
69
70     # 将结果保存到Excel文件
71     output_file = '平均每日收益率_2014_2024.xlsx'
72     all_daily_returns.to_excel(output_file, index=False)
73
74     print(f"每日平均收益率数据已保存至 {output_file}")

```

```

74
75 # 执行主函数
76 if __name__ == "__main__":
77     main()

```

B.2 问题一市场流动性代码

```

1  import pandas as pd
2  import numpy as np
3  import matplotlib.pyplot as plt
4  import os
5
6  data_folder = './沪深300成分股的数据/'
7
8
9  def load_data(year):
10     stock_info_path = os.path.join(data_folder, f'hs300stocks_{year}.csv')
11     kdata_path = os.path.join(data_folder, f'hs300stocks_kdata_{year}.csv')
12
13     # 读取CSV文件
14     stock_info = pd.read_csv(stock_info_path)
15     kdata = pd.read_csv(kdata_path)
16
17     # 合并成分股信息和每日行情数据
18     data = pd.merge(kdata, stock_info[['code', 'weight']], on='code', how='left')
19     return data
20
21
22 def min_max_scaling(series):
23     if series.max() == series.min():
24         return pd.Series([0] * len(series)) # 如果所有值相同, 返回全0的系列
25     return (series - series.min()) / (series.max() - series.min())
26
27
28 def calculate_liquidity_index(stocks_data):
29     # 确保时间列存在并转换为日期格式
30     if 'time' not in stocks_data.columns:
31         raise KeyError("数据中缺少 'time' 列, 请检查数据格式。")
32

```

```

33 stocks_data['time'] = pd.to_datetime(stocks_data['time'], errors='coerce')
34
35 # 检查日期转换是否成功
36 if stocks_data['time'].isnull().any():
37     raise ValueError("日期格式不正确，无法转换为日期。请检查数据中的 'time' 列。")
38
39 # 计算流动性指标
40 grouped_data = stocks_data.groupby('time').agg({
41     'volume': 'sum',
42     'amount': 'sum'
43 }).reset_index()
44
45 # 换手率的近似计算
46 total_volume = grouped_data['volume'].sum()
47 grouped_data['TurnoverRate'] = grouped_data['volume'] / total_volume if
    total_volume > 0 else 0
48
49 # 标准化数据 (Min-Max)
50 grouped_data[['volume', 'amount', 'TurnoverRate']] = grouped_data[['volume',
    'amount', 'TurnoverRate']].apply(
51     min_max_scaling)
52
53 # 计算每年的流动性指标
54 yearly_liquidity = grouped_data.groupby(grouped_data['time'].dt.year).agg({
55     'volume': 'mean',
56     'amount': 'mean',
57     'TurnoverRate': 'mean'
58 }).reset_index()
59
60 # 计算市场流动性 (三个指标的平均值)
61 yearly_liquidity['MarketLiquidity'] = yearly_liquidity[['volume', 'amount',
    'TurnoverRate']].mean(axis=1)
62
63 # 返回流动性指标
64 liquidity_df = pd.DataFrame({
65     'Year': yearly_liquidity['time'].astype(int), # 将年份设置为整数格式
66     'AverageVolume': yearly_liquidity['volume'],
67     'AverageAmount': yearly_liquidity['amount'],
68     'AverageTurnoverRate': yearly_liquidity['TurnoverRate'],
69     'MarketLiquidity': yearly_liquidity['MarketLiquidity']

```



```

70 })
71
72 return liquidity_df
73
74
75 def visualize_liquidity_index(all_liquidity_df):
76     plt.rcParams['font.sans-serif'] = ['SimHei']
77     plt.rcParams['axes.unicode_minus'] = False
78     # 确保年份按顺序排序
79     all_liquidity_df = all_liquidity_df.sort_values(by='Year')
80
81     # 绘制折线图
82     plt.figure(figsize=(6, 10))
83     plt.plot(all_liquidity_df['Year'], all_liquidity_df['AverageVolume'],
84              marker='o', label='平均成交量')
85     plt.plot(all_liquidity_df['Year'], all_liquidity_df['AverageAmount'],
86              marker='o', label='平均成交额')
87     plt.plot(all_liquidity_df['Year'], all_liquidity_df['AverageTurnoverRate'],
88              marker='o', label='平均换手率')
89     plt.plot(all_liquidity_df['Year'], all_liquidity_df['MarketLiquidity'],
90              marker='o', label='市场流动性')
91
92     plt.title('2014-2024年市场流动性指标', fontsize=16, fontweight='bold')
93     plt.xlabel('年份', fontsize=14, fontweight='bold')
94     plt.ylabel('流动性指标（标准化后）', fontsize=14, fontweight='bold')
95     plt.xticks(all_liquidity_df['Year'], rotation=45) # 显示年份并旋转标签
96     plt.xticks(fontsize=12, fontweight='bold')
97     plt.yticks(fontsize=12, fontweight='bold')
98     plt.legend()
99     plt.grid()
100    plt.tight_layout()
101    plt.show()
102
103    # 主程序
104    if __name__ == "__main__":
105        all_liquidity_df = pd.DataFrame() # 创建一个空的数据框以存储所有年份的流动性指标
106        for year in range(2014, 2025):
107            stocks_data = load_data(year)
108            liquidity_df = calculate_liquidity_index(stocks_data)

```

```

106 all_liquidity_df = pd.concat([all_liquidity_df, liquidity_df], ignore_index=True)
107
108 # 打印每年的流动性指标
109 print(f"{year} 年的平均成交量: {liquidity_df['AverageVolume'].mean():.2f}, "
110       f"平均成交额: {liquidity_df['AverageAmount'].mean():.2f}, "
111       f"平均换手率: {liquidity_df['AverageTurnoverRate'].mean():.4f}, "
112       f"市场流动性: {liquidity_df['MarketLiquidity'].mean():.4f}")
113
114 # # 检查最终合并的数据框是否为空
115 # if not all_liquidity_df.empty:
116 #     visualize_liquidity_index(all_liquidity_df)
117 # else:
118 #     print("没有有效的流动性指标数据可供绘图。")

```

B.3 问题一市场流动性代码 2

```

1 import pandas as pd
2 import numpy as np
3 import os
4
5 data_folder = './沪深300成分股的数据/'
6
7
8 def load_data(year):
9     stock_info_path = os.path.join(data_folder, f'hs300stocks_{year}.csv')
10    kdata_path = os.path.join(data_folder, f'hs300stocks_kdata_{year}.csv')
11
12    # 读取CSV文件
13    stock_info = pd.read_csv(stock_info_path)
14    kdata = pd.read_csv(kdata_path)
15
16    # 合并成分股信息和每日行情数据
17    data = pd.merge(kdata, stock_info[['code', 'weight']], on='code', how='left')
18    return data
19
20
21 def min_max_scaling(series):
22     if series.max() == series.min():
23         return pd.Series([0] * len(series)) # 如果所有值相同, 返回全0的系列

```

```

24 return (series - series.min()) / (series.max() - series.min())
25
26
27 def calculate_daily_liquidity_index(stocks_data):
28     # 确保时间列存在并转换为日期格式
29     if 'time' not in stocks_data.columns:
30         raise KeyError("数据中缺少 'time' 列，请检查数据格式。")
31
32     stocks_data['time'] = pd.to_datetime(stocks_data['time'], errors='coerce')
33
34     # 检查日期转换是否成功
35     if stocks_data['time'].isnull().any():
36         raise ValueError("日期格式不正确，无法转换为日期。请检查数据中的 'time' 列。")
37
38     # 计算流动性指标
39     grouped_data = stocks_data.groupby('time').agg({
40         'volume': 'sum',
41         'amount': 'sum'
42     }).reset_index()
43
44     # 换手率的近似计算
45     total_volume = grouped_data['volume'].sum()
46     grouped_data['TurnoverRate'] = grouped_data['volume'] / total_volume if
        total_volume > 0 else 0
47
48     # 标准化数据 (Min-Max)
49     grouped_data[['volume', 'amount', 'TurnoverRate']] = grouped_data[['volume',
        'amount', 'TurnoverRate']].apply(
50 min_max_scaling)
51
52     # 计算每日市场流动性 (三个指标的平均值)
53     grouped_data['MarketLiquidity'] = grouped_data[['volume', 'amount',
        'TurnoverRate']].mean(axis=1)
54
55     # 返回每日的流动性指标
56     liquidity_df = pd.DataFrame({
57         'Date': grouped_data['time'],
58         'MarketLiquidity': grouped_data['MarketLiquidity']
59     })
60

```

```

61 return liquidity_df
62
63
64 # 主程序
65 if __name__ == "__main__":
66     all_liquidity_df = pd.DataFrame() #
        创建一个空的数据框以存储所有年份的每日流动性指标
67     for year in range(2014, 2025):
68         stocks_data = load_data(year)
69         daily_liquidity_df = calculate_daily_liquidity_index(stocks_data)
70         all_liquidity_df = pd.concat([all_liquidity_df, daily_liquidity_df],
            ignore_index=True)
71
72 # 将结果保存到Excel文件
73 output_file = '市场流动性每日指标.xlsx'
74 all_liquidity_df.to_excel(output_file, index=False)
75
76 print(f"所有年份的每日市场流动性指标已保存到 {output_file}。")

```

B.4 问题一市场情绪指标 matlab 代码 (以 2024 年的数据为例)

```

1 % 读取数据
2 data = readtable('hs300stocks_kdata_2024.csv');
3 dates = unique(data.time); % 使用实际的日期列名
4 numDates = length(dates);
5 alpha = 0.95; % VaR和CVaR置信水平
6
7 % 初始化各个情绪指标
8 VIX = zeros(numDates, 1);
9 MFI = zeros(numDates, 1);
10 ISI = zeros(numDates, 1);
11 VaR_values = zeros(numDates, 1);
12
13 % 循环日期计算每日的情绪指标
14 for i = 2:numDates
15     dailyData = data(data.time == dates(i), :); % 使用实际的日期列名
16     returns = diff(log(dailyData.close)); % 使用实际的收盘价列名计算对数收益率
17
18 % 1. 波动性指数 (VIX)

```

```

19 VIX(i) = sqrt(var(returns));
20
21 % 2. 资金流向指数 (MFI)
22 typicalPrice = (dailyData.high + dailyData.low + dailyData.close) / 3; %
    使用实际列名
23 moneyFlow = typicalPrice .* dailyData.volume;
24 posFlow = sum(moneyFlow(returns > 0));
25 negFlow = sum(moneyFlow(returns < 0));
26 MFI(i) = 100 - (100 / (1 + posFlow / negFlow));
27
28 % 3. 投资者情绪指数 (ISI)
29 ISI(i) = sum((dailyData.close - dailyData.open) .* dailyData.volume) /
    sum(dailyData.volume);
30
31 % 4. VaR
32 mu = mean(returns);
33 sigma = std(returns);
34 VaR_values(i) = mu + sigma * norminv(alpha);
35 end
36
37 % 创建表格
38 T = table(dates(2:end), VIX(2:end), MFI(2:end), ISI(2:end), VaR_values(2:end),
    ...
39 'VariableNames', {'Date', 'VIX', 'MFI', 'ISI', 'VaR'});
40
41 % 输出结果到Excel文件
42 outputFilename = 'Market_Sentiment_Indicators_2024.xlsx';
43 writetable(T, outputFilename);
44
45 disp(['市场情绪指标已导出到文件: ', 'Market_Sentiment_Indicators_2024.xlsx']);

```

B.5 问题一市场情绪指标可视化 Matlab 代码

```

1 % 获取当前工作目录
2 folderPath = pwd;
3
4 % 设置输出文件名
5 outputFilename = 'Combined_Market_Sentiment_Indicators.xlsx';
6

```

```

7 % 创建文件名模式以匹配所有文件
8 fileList = dir(fullfile(folderPath, 'Market_Sentiment_Indicators_*.xlsx'));
9
10 % 初始化最终结果表格
11 allData = [];
12
13 % 遍历每个Excel文件并合并数据
14 for k = 1:length(fileList)
15 % 读取当前Excel文件的数据
16 filePath = fullfile(folderPath, fileList(k).name);
17 data = readtable(filePath);
18
19 % 将当前文件的数据追加到allData中
20 allData = [allData; data];
21 end
22
23 % 输出合并结果到新的Excel文件
24 writetable(allData, outputFilename);
25
26 % 正确的文件输出信息
27 disp(['所有文件的数据已合并到文件: ', outputFilename]);
28
29 % 读取合并后的数据
30 allData = readtable(outputFilename);
31
32 % 将日期列转换为MATLAB的日期格式
33 allData.Date = datetime(allData.Date, 'Format', 'yyyy-MM-dd');
34
35 % 绘制折线图
36 figure;
37
38 % 绘制VIX
39 subplot(4, 1, 1);
40 plot(allData.Date, allData.VIX, 'b-', 'LineWidth', 1.5);
41 title('VIX (波动性指数)');
42 xlabel('Date');
43 ylabel('VIX');
44 grid on;
45
46 % 绘制MFI

```

```

47 subplot(4, 1, 2);
48 plot(allData.Date, allData.MFI, 'g-', 'LineWidth', 1.5);
49 title('MFI (资金流向指数)');
50 xlabel('Date');
51 ylabel('MFI');
52 grid on;
53
54 % 绘制ISI
55 subplot(4, 1, 3);
56 plot(allData.Date, allData.ISI, 'r-', 'LineWidth', 1.5);
57 title('ISI (投资者情绪指数)');
58 xlabel('Date');
59 ylabel('ISI');
60 grid on;
61
62 % 绘制VaR
63 subplot(4, 1, 4);
64 plot(allData.Date, allData.VaR, 'k-', 'LineWidth', 1.5);
65 title('VaR (风险价值)');
66 xlabel('Date');
67 ylabel('VaR');
68 grid on;
69
70 % 调整图表布局
71 sgtitle('沪深300市场情绪指标随时间的变化');

```

B.6 问题二 alpha 系数的计算

```

1 import pandas as pd
2 import numpy as np
3 from sklearn.linear_model import LinearRegression
4
5 # 读取数据
6 df = pd.read_csv('./data/hs300stocks_kdata_2024.csv')
7
8 # 转换日期为标准日期格式
9 df['time'] = pd.to_datetime(df['time'])
10
11 # 按日期分组，计算所有证券的平均收盘价，作为市场指数的近似

```

```

12 market_index = df.groupby('time')['close'].mean().reset_index()
13 market_index.columns = ['time', 'market_close']
14
15 # 计算市场的每日收益率
16 market_index = market_index.sort_values('time')
17 market_index['market_return'] = market_index['market_close'].pct_change()
18
19 # 将市场收益率合并到主数据框中
20 df = df.merge(market_index[['time', 'market_return']], on='time', how='left')
21
22 # 计算每个证券的每日收益率
23 df['stock_return'] = df.groupby('code')['close'].pct_change()
24
25 # 定义一个函数来计算每个证券的 Alpha 系数
26 def calculate_alpha(stock_code, df):
27     # 筛选出对应证券的数据
28     stock_data = df[df['code'] == stock_code].dropna(subset=['stock_return',
29         'market_return'])
29
30     # 如果数据量不足，跳过该证券
31     if len(stock_data) < 2:
32         return np.nan
33
34     # 回归模型
35     X = stock_data['market_return'].values.reshape(-1, 1) # 市场收益率
36     y = stock_data['stock_return'].values # 证券收益率
37
38     # 创建并训练模型
39     model = LinearRegression()
40     model.fit(X, y)
41
42     # Alpha 系数即为回归模型的截距项
43     alpha = model.intercept_
44     return alpha
45
46 # 计算所有证券的 Alpha 系数
47 alpha_results = {}
48 for stock_code in df['code'].unique():
49     alpha = calculate_alpha(stock_code, df)
50     alpha_results[stock_code] = alpha

```



```

51
52 # 转换为DataFrame并输出结果
53 alpha_df = pd.DataFrame(list(alpha_results.items()), columns=['StockCode',
54                               'Alpha'])
55
56 # 将 Alpha 系数结果保存到 CSV 文件
57 alpha_df.to_csv('alpha_coefficients_2024.csv', index=False)
58 print("Alpha coefficients have been saved to 'alpha_coefficients.csv'")

```

B.7 问题二 beta 系数的计算

```

1  import pandas as pd
2  import numpy as np
3  from sklearn.linear_model import LinearRegression
4
5  # 读取数据
6  df = pd.read_csv('./data/hs300stocks_kdata_2024.csv')
7
8  # 转换日期为标准日期格式
9  df['time'] = pd.to_datetime(df['time'])
10
11 # 按日期分组，计算所有证券的平均收盘价，作为市场指数的近似
12 market_index = df.groupby('time')['close'].mean().reset_index()
13 market_index.columns = ['time', 'market_close']
14
15 # 计算市场的每日收益率
16 market_index = market_index.sort_values('time')
17 market_index['market_return'] = market_index['market_close'].pct_change()
18
19 # 将市场收益率合并到主数据框中
20 df = df.merge(market_index[['time', 'market_return']], on='time', how='left')
21
22 # 计算每个证券的每日收益率
23 df['stock_return'] = df.groupby('code')['close'].pct_change()
24
25 # 定义一个函数来计算每个证券的 Beta 系数
26 def calculate_beta(stock_code, df):
27     # 筛选出对应证券的数据
28     stock_data = df[df['code'] == stock_code].dropna(subset=['stock_return',

```

```

        'market_return'])
29
30 # 如果数据量不足, 跳过该证券
31 if len(stock_data) < 2:
32     return np.nan
33
34 # 回归模型
35 X = stock_data['market_return'].values.reshape(-1, 1) # 市场收益率
36 y = stock_data['stock_return'].values # 证券收益率
37
38 # 创建并训练模型
39 model = LinearRegression()
40 model.fit(X, y)
41
42 # Beta系数即为回归模型的系数
43 beta = model.coef_[0]
44 return beta
45
46 # 计算所有证券的 Beta 系数
47 beta_results = {}
48 for stock_code in df['code'].unique():
49     beta = calculate_beta(stock_code, df)
50     beta_results[stock_code] = beta
51
52 # 转换为DataFrame并输出结果
53 beta_df = pd.DataFrame(list(beta_results.items()), columns=['StockCode', 'Beta'])
54 print(beta_df)
55 beta_df.to_csv('beta_coefficients_2024.csv', index=False)
56 print("Beta coefficients have been saved to 'beta_coefficients.csv'")

```

B.8 问题二夏普计算

```

1 import pandas as pd
2 import numpy as np
3
4 # 读取数据
5 df = pd.read_csv('./data/hs300stocks_kdata_2024.csv')
6
7 # 转换日期为标准日期格式

```

```

8 df['time'] = pd.to_datetime(df['time'])
9
10 # 计算每个证券的每日收益率
11 df = df.sort_values(['code', 'time'])
12 df['stock_return'] = df.groupby('code')['close'].pct_change()
13
14 # 假设年化无风险收益率为2%，转化为每日收益率
15 risk_free_rate_daily = (1 + 0.02) ** (1 / 252) - 1 # 252个交易日
16
17 # 定义一个函数来计算每个证券的夏普比率
18 def calculate_sharpe_ratio(stock_code, df):
19     # 筛选出对应证券的数据
20     stock_data = df[df['code'] == stock_code].dropna(subset=['stock_return'])
21
22     # 如果数据量不足，跳过该证券
23     if len(stock_data) < 2:
24         return np.nan
25
26     # 计算平均每日收益率
27     avg_return = stock_data['stock_return'].mean()
28
29     # 计算收益率的标准差（波动率）
30     volatility = stock_data['stock_return'].std()
31
32     # 计算夏普比率
33     sharpe_ratio = (avg_return - risk_free_rate_daily) / volatility
34     return sharpe_ratio
35
36 # 计算所有证券的夏普比率
37 sharpe_ratio_results = {}
38 for stock_code in df['code'].unique():
39     sharpe_ratio = calculate_sharpe_ratio(stock_code, df)
40     sharpe_ratio_results[stock_code] = sharpe_ratio
41
42 # 转换为DataFrame并输出结果
43 sharpe_ratio_df = pd.DataFrame(list(sharpe_ratio_results.items()),
44                                columns=['StockCode', 'SharpeRatio'])
45
46 # 将夏普比率结果保存到 CSV 文件
47 sharpe_ratio_df.to_csv('sharpe_ratio_results_2024.csv', index=False)

```

```
47 print("Sharpe Ratio results have been saved to 'sharpe_ratio_results.csv'")
```

B.9 问题二波动性代码

```
1 import pandas as pd
2 import numpy as np
3
4 # 读取数据
5 df = pd.read_csv('./data/hs300stocks_kdata_2024.csv')
6
7 # 转换日期为标准日期格式
8 df['time'] = pd.to_datetime(df['time'])
9
10 # 计算每个证券的每日收益率
11 df = df.sort_values(['code', 'time'])
12 df['stock_return'] = df.groupby('code')['close'].pct_change()
13
14 # 定义一个函数来计算每个证券的波动率
15 def calculate_volatility(stock_code, df):
16     # 筛选出对应证券的数据
17     stock_data = df[df['code'] == stock_code].dropna(subset=['stock_return'])
18
19     # 如果数据量不足，跳过该证券
20     if len(stock_data) < 2:
21         return np.nan
22
23     # 计算每日收益率的标准差，即为波动率
24     volatility = stock_data['stock_return'].std()
25     return volatility
26
27 # 计算所有证券的波动率
28 volatility_results = {}
29 for stock_code in df['code'].unique():
30     volatility = calculate_volatility(stock_code, df)
31     volatility_results[stock_code] = volatility
32
33 # 转换为DataFrame并输出结果
34 volatility_df = pd.DataFrame(list(volatility_results.items()),
35                               columns=['StockCode', 'Volatility'])
```

```

35
36 # 将波动率结果保存到 CSV 文件
37 volatility_df.to_csv('volatility_results_2024.csv', index=False)
38 print("Volatility results have been saved to 'volatility_results.csv'")

```

B.10 问题二收益率代码

```

1 import pandas as pd
2
3 # 读取数据
4 df = pd.read_csv('./data/hs300stocks_kdata_2024.csv')
5
6 # 转换日期为标准日期格式
7 df['time'] = pd.to_datetime(df['time'])
8
9 # 按证券代码和日期排序
10 df = df.sort_values(['code', 'time'])
11
12 # 定义一个函数来计算每只证券的累计收益率
13 def calculate_total_return(stock_code, df):
14     # 筛选出对应证券的数据
15     stock_data = df[df['code'] == stock_code]
16
17     # 如果数据量不足，跳过该证券
18     if len(stock_data) < 2:
19         return None
20
21     # 计算累计收益率：(最后一个收盘价 - 第一个收盘价) / 第一个收盘价
22     start_price = stock_data['close'].iloc[0]
23     end_price = stock_data['close'].iloc[-1]
24     total_return = (end_price - start_price) / start_price
25     return total_return
26
27 # 计算所有证券的累计收益率
28 return_results = {}
29 for stock_code in df['code'].unique():
30     total_return = calculate_total_return(stock_code, df)
31     return_results[stock_code] = total_return
32

```

```

33 # 转换为DataFrame并输出结果
34 return_df = pd.DataFrame(list(return_results.items()), columns=['StockCode',
    'TotalReturn'])
35
36 # 将累计收益率结果保存到 CSV 文件
37 return_df.to_csv('total_return_results_2024.csv', index=False)
38 print("Total return results have been saved to 'total_return_results.csv'")

```

B.11 问题二多元线性回归训练代码

```

1 import pandas as pd
2 import glob
3 from sklearn.linear_model import LinearRegression
4
5 # 用于存储年度合并数据和训练数据
6 all_data = []
7 training_data = []
8
9 # 遍历2014-2024年
10 for year in range(2014, 2025):
11     # 定义文件路径
12     alpha_file = f"alpha_coefficients/alpha_coefficients_{year}.csv"
13     beta_file = f"beta_coefficients/beta_coefficients_{year}.csv"
14     sharpe_file = f"sharpe_ratio/sharpe_ratio_results_{year}.csv"
15     volatility_file = f"volatility/volatility_results_{year}.csv"
16     return_file = f"return/total_return_results_{year}.csv"
17
18 # 读取每年的特征文件和收益率文件
19 alpha_df = pd.read_csv(alpha_file)
20 beta_df = pd.read_csv(beta_file)
21 sharpe_df = pd.read_csv(sharpe_file)
22 volatility_df = pd.read_csv(volatility_file)
23 return_df = pd.read_csv(return_file)
24
25 # 合并特征数据
26 merged_df = alpha_df.merge(beta_df, on="StockCode", suffixes=('_alpha', '_beta'))
27 merged_df = merged_df.merge(sharpe_df, on="StockCode")
28 merged_df = merged_df.merge(volatility_df, on="StockCode", suffixes=('_sharpe',
    '_volatility'))

```

```

29 merged_df = merged_df.merge(return_df, on="StockCode")
30
31 # 重命名列，确保统一的列名
32 merged_df.columns = ["StockCode", "Alpha", "Beta", "SharpeRatio", "Volatility",
33                       "Return"]
34
35 # 添加年份列
36 merged_df["Year"] = year
37
38 # 将数据添加到训练集列表
39 training_data.append(merged_df[["Alpha", "Beta", "SharpeRatio", "Volatility",
40                                 "Return"]])
41
42 # 将合并的数据存储
43 all_data.append(merged_df)
44
45 # 合并所有年份的训练数据
46 training_df = pd.concat(training_data, ignore_index=True)
47
48 # 分割特征和目标值
49 X = training_df[["Alpha", "Beta", "SharpeRatio", "Volatility"]]
50 y = training_df["Return"]
51
52 # 使用线性回归训练权重
53 model = LinearRegression()
54 model.fit(X, y)
55
56 # 获取训练得到的权重
57 weights = {
58     "Alpha": model.coef_[0],
59     "Beta": model.coef_[1],
60     "SharpeRatio": model.coef_[2],
61     "Volatility": model.coef_[3]
62 }
63
64 print("训练得到的权重:", weights)
65
66 # 用训练好的权重计算每只股票的风险评分
67 final_results = []

```

```

67 for year_data in all_data:
68     # 计算风险评分
69     year_data["RiskScore"] = (
70     year_data["Alpha"] * weights["Alpha"] +
71     year_data["Beta"] * weights["Beta"] +
72     year_data["SharpeRatio"] * weights["SharpeRatio"] +
73     year_data["Volatility"] * weights["Volatility"]
74     )
75     final_results.append(year_data)
76
77     # 合并所有年份的风险评分数据
78     final_data = pd.concat(final_results, ignore_index=True)
79
80     # 保存最终结果到新文件
81     final_data.to_csv("trained_risk_assessment_results_2014_2024.csv", index=False)
82     print("风险评估结果已保存到 'trained_risk_assessment_results_2014_2024.csv'")

```

B.12 问题三模型代码

```

1  import pandas as pd
2  import numpy as np
3  from scipy.optimize import minimize
4  import matplotlib.pyplot as plt
5  import xlswriter
6
7  # 读取股票价格数据
8  try:
9      price_data = pd.read_csv(
10         r'E:\大学\数学建模\2024粤港澳大湾区数学建模\沪深300成分股的数据
11         \沪深300成分股的数据\hs300stocks_kdata_2018.csv')
12     except FileNotFoundError:
13         print("文件 hs300stocks_kdata_2018.csv 未找到，请检查路径是否正确。")
14         exit()
15
16     # 将日期列 "time" 转换为日期格式并保留日期部分
17     price_data['time'] = pd.to_datetime(price_data['time']).dt.date
18     price_data = price_data.set_index(['time', 'code'])
19
20     # 按股票代码计算每日收盘价的收益率矩阵

```



```

21 returns =
    price_data['close'].unstack(level=1).pct_change(fill_method=None).dropna()
22
23 # 读取权重数据
24 try:
25     weights_data = pd.read_csv(
26         r'E:\大学\数学建模\2024粤港澳大湾区数学建模\沪深300成分股的数据
27         \沪深300成分股的数据\hs300stocks_2018.csv')
28 except FileNotFoundError:
29     print("文件 hs300stocks_2018.csv 未找到，请检查路径是否正确。")
30     exit()
31
32 # 确保权重数据的列名为 'code' 和 'weight'
33 if 'code' not in weights_data.columns or 'weight' not in weights_data.columns:
34     print("请检查权重数据的列名，确保其分别为 'code' 和 'weight'")
35     exit()
36
37 # 设置股票代码为索引，并按 returns 中的股票代码对齐权重数据
38 weights_data = weights_data.set_index('code')
39 weights_data = weights_data.loc[returns.columns] # 按 returns 中的股票代码对齐
40 stock_weights = weights_data['weight']
41 stock_weights /= stock_weights.sum() # 归一化权重
42
43 # 确保对齐后的 returns 和 stock_weights 长度一致
44 if len(stock_weights) != returns.shape[1]:
45     print("权重数据的长度与收益率数据的列数不匹配，请检查数据的一致性。")
46     exit()
47
48 # 定义无风险利率
49 risk_free_rate = 0.03
50
51 # 定义夏普比率的负值（添加最大回撤惩罚项）
52 def sharpe_ratio_with_drawdown_penalty(weights, returns, risk_free_rate,
    max_drawdown=0.7, penalty_factor=100):
53     mean_returns = returns.mean()
54     cov_matrix = returns.cov()
55     portfolio_return = np.dot(weights, mean_returns)
56     portfolio_volatility = np.sqrt(np.dot(weights.T, np.dot(cov_matrix, weights)))
57     sharpe_ratio = (portfolio_return - risk_free_rate) / portfolio_volatility
58

```

```

59 # 计算组合的最大回撤
60 portfolio_returns = returns.dot(weights)
61 cumulative_returns = (portfolio_returns + 1).cumprod() - 1
62 peak = cumulative_returns.cummax()
63 drawdown = (peak - cumulative_returns) / peak
64 max_drawdown_observed = drawdown.max()
65
66 # 若最大回撤超出0.7，增加惩罚项
67 penalty = penalty_factor * max(0, max_drawdown_observed - max_drawdown)
68 return -sharpe_ratio + penalty
69
70 # 约束条件：权重之和为1
71 constraints = [{'type': 'eq', 'fun': lambda weights: np.sum(weights) - 1}]
72
73 # 将单只股票的权重上限降到5%以增加分散化
74 weight_limit = 0.05
75 bounds = tuple((0, weight_limit) for _ in range(len(stock_weights)))
76
77 # 初始权重
78 initial_weights = stock_weights.values
79
80 # 优化
81 optimized = minimize(sharpe_ratio_with_drawdown_penalty, initial_weights,
82                      args=(returns, risk_free_rate),
83                      method='SLSQP', bounds=bounds, constraints=constraints)
84
85 # 输出优化结果
86 if optimized.success:
87     optimized_weights = optimized.x
88     expected_return = np.dot(optimized_weights, returns.mean())
89     volatility = np.sqrt(np.dot(optimized_weights.T, np.dot(returns.cov(),
90                                                             optimized_weights)))
91
92 # 计算最终组合的最大回撤
93 portfolio_returns = returns.dot(optimized_weights)
94 cumulative_returns = (portfolio_returns + 1).cumprod() - 1
95 peak = cumulative_returns.cummax()
96 drawdown = (peak - cumulative_returns) / peak
97 max_drawdown_observed = drawdown.max()

```

```

97 print("Optimized Portfolio Weights:", optimized_weights)
98 print("Expected Portfolio Return:", expected_return)
99 print("Portfolio Volatility:", volatility)
100 print("Max Drawdown:", max_drawdown_observed)
101
102 # 可视化累计收益和回撤
103 plt.figure(figsize=(14, 7))
104
105 # 绘制累计收益曲线
106 plt.subplot(2, 1, 1)
107 cumulative_returns.plot()
108 plt.title("Cumulative Returns of Optimized Portfolio")
109 plt.xlabel("Date")
110 plt.ylabel("Cumulative Return")
111
112 # 绘制最大回撤曲线
113 plt.subplot(2, 1, 2)
114 drawdown.plot()
115 plt.title("Drawdown of Optimized Portfolio")
116 plt.xlabel("Date")
117 plt.ylabel("Drawdown")
118
119 plt.tight_layout()
120 plt.show()
121
122 # 导出结果到Excel文件
123 with pd.ExcelWriter('optimized_portfolio_results.xlsx', engine='xlsxwriter') as
    writer:
124 # 优化权重输出到Excel
125 optimized_weights_df = pd.DataFrame({'Stock': returns.columns, 'Weight':
    optimized_weights})
126 optimized_weights_df.to_excel(writer, sheet_name='Optimized Weights',
    index=False)
127
128 # 输出其他指标
129 summary_df = pd.DataFrame({
130 'Expected Return': [expected_return],
131 'Volatility': [volatility],
132 'Max Drawdown': [max_drawdown_observed]
133 })

```

```

134 summary_df.to_excel(writer, sheet_name='Summary', index=False)
135
136 # 输出累计收益和回撤数据
137 cumulative_returns.to_frame(name='Cumulative Returns').to_excel(writer,
    sheet_name='Cumulative Returns')
138 drawdown.to_frame(name='Drawdown').to_excel(writer, sheet_name='Drawdown')
139 else:
140 print("Optimization failed:", optimized.message)

```

B.13 问题四代码

```

1 import pandas as pd
2 import numpy as np
3 import zipfile
4 import os
5
6 # 步骤 1: 解压并加载沪深300成分股数据
7 hs300_zip_path =
    r"E:\大学\数学建模\2024粤港澳大湾区数学建模\沪深300成分股的数据.zip"
8
9 # 解压缩文件内容到指定目录
10 with zipfile.ZipFile(hs300_zip_path, 'r') as zip_ref:
11 zip_ref.extractall("E:/大学/数学建模/2024粤港澳大湾区数学建模
12 /沪深300成分股的数据/extracted_hs300_data")
13
14 # 加载每年的沪深300数据, 并使用“weight”列的均值作为年收益的代理值
15 hs300_data_dir = 'E:/大学/数学建模/2024粤港澳大湾区数学建模/沪深300成分股的数据/'
16 hs300_annual_files = [f for f in os.listdir(hs300_data_dir) if
    f.startswith('hs300stocks_') and f.endswith('.csv')]
17
18 # 存储年收益率
19 annual_returns = {}
20 for file_name in hs300_annual_files:
21 year = int(file_name.split('_')[-1].split('.')[0])
22 file_path = os.path.join(hs300_data_dir, file_name)
23 hs300_data = pd.read_csv(file_path)
24 if 'weight' in hs300_data.columns:
25 annual_returns[year] = hs300_data['weight'].mean()
26

```

```

27 # 步骤 2: 使用几何平均公式计算市场平均收益率 R_m
28 N = len(annual_returns)
29 R_m = (np.prod([1 + r for r in annual_returns.values()]) ** (1 / N)) - 1
30
31 # 步骤 3: 加载并处理10年期国债数据
32 treasury_data_path = '/mnt/data/十年国债每月数据.xlsx'
33 treasury_data = pd.read_excel(treasury_data_path, header=None)
34 treasury_data.columns = ['Date', 'Yield']
35 treasury_data['Date'] = pd.to_datetime('1899-12-30') +
    pd.to_timedelta(treasury_data['Date'], 'D')
36 R_f = treasury_data['Yield'].mean() / 100 # 转换百分比为小数
37
38 # 步骤 4: 计算风险溢价 RP
39 RP = R_m - R_f
40
41 # 步骤 5: 计算合理收益预期 E(R)
42 E_R = R_f + RP
43
44 # 输出结果
45 print("市场平均收益率 (R_m):", R_m)
46 print("无风险收益率 (R_f):", R_f)
47 print("风险溢价 (RP):", RP)
48 print("合理收益预期 (E(R)):", E_R)

```