



中國人民大學

RENMIN UNIVERSITY OF CHINA

金融大数据分析与量化交易

基于 Xgboost 的股价预测模型及量化交易策略

Xgboost-Based Stock Price Prediction Model and
Quantitative Trading Strategy

所在学院 财政金融学院

姓 名 王晨曦

学 号 2022200147

所在专业 金融数学实验班

2024 年 12 月 11 日

基于 Xgboost 的股价预测模型及量化交易策略

摘要

量化交易，凭借其高效、可靠、及时性的特点，现如今正受到越来越多人的关注和认同。本文旨在构建一个行之有效的量化交易策略。

本文简单来说，可以分为三个部分。首先，作者基于自己在平时读到的一篇文章启发了对量化交易策略搭建的灵感，在文章的第一部分，通过随机森林的算法，利用股票的技术指标对未来股价进行预测，在论文中提出了“股票表现概率”这一概念；对于我们第二部分基于 Xgboost 的股价预测模型及量化交易策略的搭建具有重要的指导意义。

在第二部分，我们从 CSMAR 国泰安经济金融数据库下载股票数据后，构造了滞后期、窗口平均等重要特征，对未来的股票价格进行预测，并通过贝叶斯/网格优化的方式获得最优参数，取得了较好的预测效果。受启发于先前读过的论文，本文在预测结果的基础上，通过预期收益率的方式指定相关的交易信号，并进行了一些优化。在第二部分的评价及拓展部分，我们验证了 Xgboost 模型在其他股票上的可复制性，并提出了用神经网络中经常用于输出层的 Softmax 激活函数来进行各资产投资组合的配比，以此期待获得更客观的量化策略收益。同时，我们介绍了 xtquant 平台的使用，通过平台与 Python 的 API 接口，可以实现代码层面的量化交易策略的自动化交易（下单、撤单、持仓管理等）。

在文章的第三部分，我们着眼于投资组合的权重分配问题，且跳出前面专注于某国、某类（风险）资产的投资视角，从国际投资组合构建的领域，介绍国际投资背景、人为选择投资标的的理由及解释，通过线性规划的方式，对国际投资组合的构建展开了研究，并最终得到一个完整的未来投资组合选择。该部分与前两部分相比，融入了更多人为主观的判断，在策略实际采用过程中更需要人为对大形势进行判断。第三部分主要侧重于不完全相信机器预测效果，仍需人为选股和人为择时，以期达到更高的收益。

总体来说，本文介绍了基于之前的学习和探索，提出了一种新的自研的量化交易策略，该量化交易策略以 Xgboost 模型作为依托；同时介绍了如何利用 Python 求解投资组合优化的线性规划问题。

关键词：量化交易；股价预测；Xgboost；国际投资；投资组合；线性规划

Xgboost-Based Stock Price Prediction Model and Quantitative Trading Strategy

Abstract

Quantitative trading, with its efficiency, reliability and timeliness, is nowadays receiving more and more attention and recognition. This paper aims to construct a proven quantitative trading strategy.

In brief, this paper can be divided into three parts. First, the author based on his own in the usual read an article inspired to build a quantitative trading strategy, in the first part of the article, through the algorithm of random forests, the In the first part of the paper, the concept of ‘stock performance probability’ is proposed in the paper by using the technical indicators of stocks to predict the future stock price; The concept of ‘probability of stock performance’ is introduced in the paper, which is of great significance for the construction of the stock price prediction model and quantitative trading strategy based on Xgboost in the second part of the paper.

In the second part, we download the stock data from CSMAR and construct important features such as lags and window averages to predict the future stock prices. forecast the future stock prices and obtain the optimal parameters by Bayesian/Lattice Optimisation to achieve better prediction results. Inspired by the previously read papers, this paper specifies the relevant trading signals by means of expected returns based on the prediction results and performs some optimisations. In the second part of the evaluation and extension section, we validate the replicability of the Xgboost model on other stocks and propose to use the Softmax activation function, which is often used in the output layer of neural networks, for the allocation of portfolios across assets. in anticipation of more objective quantitative strategy returns. At the same time, we introduce the use of the xtquant platform, through the platform’s API in-

terface with Python, we also introduce the use of the xtquant platform, which allows us to automate the trading of quantitative trading strategies at the code level (order placing, order cancellation, position management, etc.) through the platform's API interface with Python.

In the third part of the article, we look at the issue of portfolio weighting, and go beyond the previous focus on a particular country or asset class. From the field of international portfolio construction, we introduce the background of international investment, the rationale and explanation for the artificial choice of investment targets, and through linear programming, we develop a study of international portfolio construction is investigated and finally a complete future portfolio selection is obtained. This part, compared to the previous two parts, incorporates more human subjective judgement than the first two parts, and it is more necessary to make human judgement on the future situation during the actual adoption of the strategy. The third section focuses on Instead of fully trusting the machine prediction effect, human stock selection and human timing are still needed in order to achieve higher returns.

Overall, this paper presents a new self-developed quantitative trading strategy based on previous learning and exploration. This quantitative trading strategy is based on the Xgboost model; it also describes how to solve the linear programming problem of portfolio optimisation using Python.

Keywords: Quantitative Trading; Stock Price Prediction; Xgboost; International Investing; Portfolios; Linear Programming

目 录

摘要	I
Abstract	II
1 引言	1
2 基于 Random Forest 的股票选取	3
2.1 数据获取.....	3
2.2 特征工程——技术指标	4
2.3 随机森林 Random Forest 分类模型	7
2.3.1 导入必要的库	7
2.3.2 读取数据函数	8
2.3.3 Random Forest 模型训练	9
2.3.4 模型预测	10
2.3.5 模型训练及预测 main 函数	11
3 基于 Xgboost 的股价预测模型及量化交易策略	12
3.1 Xgboost v.s. Random Forest	12
3.2 数据处理.....	12
3.2.1 数据获取	12
3.2.2 数据清洗	13
3.3 特征工程.....	14
3.4 模型训练.....	16
3.4.1 划分训练集和测试集	16
3.4.2 Xgboost 模型训练	16
3.4.3 使用 Xgboost 对预测集进行预测	21
3.5 量化交易策略搭建.....	23
3.5.1 交易信号及交易策略	23
3.5.2 量化交易策略结果展示	25
3.6 量化交易结果评价.....	30
4 利用 Xgboost 进行量化交易策略搭建的评价	34
4.1 Xgboost 策略的可复制性	34
4.2 基于现有策略的拓展与改进.....	48
4.2.1 预测模型的改进	48
4.2.2 交易策略的改进	48

4.2.3 基于 Xgboost 股价预测的投资组合策略	50
5 基于 xtquant 真实自动化交易实现	58
6 国际投资组合搭建（低频量化交易）	60
6.1 为什么要选择国际投资？	60
6.2 投资定位与经济概述	61
6.2.1 投资定位	61
6.2.2 重点国家基本面分析	62
6.2.2.1 中国	62
6.2.2.2 美国	64
6.2.2.3 保加利亚	65
6.2.2.4 澳大利亚	65
6.3 投资标的选择	66
6.3.1 国债	66
6.3.2 股票	67
6.3.3 基金	68
6.3.4 期货	68
6.4 量化投资组合模型	69
6.4.1 数据获取及清洗	70
6.4.2 各类资产优化问题——CVaR	74
6.4.3 各类别投资组合优化问题——基于 Markowitz 投资组合理论	80
6.5 国际投资组合模型搭建过程总结	84
7 结语	85
参考文献	85
附录 A CVaR 的理论基础	88
A.1 风险价值（VaR）测度的改进	88
A.2 CVaR 的定义	89
A.3 CVaR 的性质	89
A.4 CVaR 模型的理论搭建	91
附录 B 基于 Xgboost 预测模型的投资组合调仓数据	93

图目录

图 3-1 Xgboost 模型数据清洗数据切片展示 ······	14
图 3-2 Xgboost 模型预测结果 ······	21
图 3-3 Xgboost 模型预测结果对比 ······	22
图 3-4 调整后的 Xgboost 模型预测结果对比 ······	23
图 3-5 量化交易策略收益率曲线及全期持有的累积收益曲线 ······	26
图 3-6 量化交易策略买卖点示意图 ······	27
图 3-7 调整后量化交易策略收益率曲线 ······	29
图 3-8 调整后量化交易策略买卖点示意图 1 ······	29
图 3-9 调整后量化交易策略买卖点示意图 2 ······	30
图 4-1 宁德时代模型股价预测结果 ······	35
图 4-2 宁德时代策略回测结果 ······	35
图 4-3 宁德时代策略交易点 ······	36
图 4-4 云南白药模型股价预测结果 ······	39
图 4-5 云南白药策略回测结果 ······	39
图 4-6 云南白药策略交易点 ······	40
图 4-7 光启技术模型股价预测结果 ······	43
图 4-8 光启技术策略回测结果 ······	43
图 4-9 光启技术策略交易点 ······	44
图 4-10 紫天科技模型股价预测结果 ······	45
图 4-11 紫天科技策略回测结果 ······	45
图 4-12 紫天科技策略交易点 ······	46
图 4-13 京东方 A 模型股价预测结果 ······	46
图 4-14 京东方 A 策略回测结果 ······	47
图 4-15 京东方 A 策略交易点 ······	47
图 4-16 云南白药股票交易信号改进后的回测结果 ······	50
图 4-17 Softmax 激活函数图像 ······	51
图 4-18 基于 Xgboost 股价预测的投资组合策略累计收益率曲线 ······	55
图 4-19 基于 Xgboost 股价预测的投资组合策略每日收益率曲线 ······	55
图 5-1 迅投 QMT 策略交易系统 ······	58
图 6-1 2022 年 3 月至 2024 年 9 月中国 GDP 变化及各科目对 GDP 的拉动情况 ···	62
图 6-2 社会融资规模变化情况 ······	63

图 6-3 中国债券发行规模 ······	64
图 6-4 中国国债收益率与利差变动 ······	64
图 6-5 各类资产最终优化结果 ······	80

表目录

表 3-1 XGBoost 与 Random Forest 的对比	12
表 3-2 最佳参数	20
表 3-3 最佳参数	22
表 3-4 交易信号数据	31
表 3-5 交易信号数据 (续)	32
表 4-1 宁德时代最佳参数	34
表 4-2 宁德时代交易信号数据	37
表 4-3 最佳参数	38
表 4-4 云南白药交易信号数据	41
表 4-5 云南白药交易信号数据 (续)	42
表 6-1 国债资产平均收益率及标准差	72
表 6-2 股票资产平均收益率及标准差	72
表 6-3 基金资产平均收益率及标准差	73
表 6-4 期货资产平均收益率及标准差	73
表 6-5 国债资产相关系数矩阵	74
表 6-6 股票资产相关系数矩阵	74
表 6-7 基金资产相关系数矩阵	74
表 6-8 期货资产相关系数矩阵	74
表 6-9 各大类资产在最终投资组合中占比	83
表 2-1 Portfolio Returns	93

1 引言

作者在探索量化交易策略的制定的过程中，结合覃老师课堂上所讲到的内容和知识，以及在日常中与金融学课程老师交流和投递实习的过程，逐渐对量化交易有了一个初步的认识：作者认为量化交易按照交易频率来划分分为高频交易和低频交易，其中高频交易（High-Frequency Trading, HFT）和低频交易（Low-Frequency Trading, LFT）的区别主要在于交易频率、持仓时间和策略复杂度。一般来说，高频交易需要频繁的交易，其目标是捕捉市场中细微的价格变化，通过不断换手来获取超额收益，因为这就要求其模型要相对比较简单，以及运行速度要较快。而低频交易一般是通过算法来获得从理论上可能会增长的股票，之后由投资者进行逐一确定是否购买，并将持有较长日期（通过几个月到几年不等）。

但从本质上来说，量化交易其实就是通过机器学习、深度学习或其他高级算法，利用计算机的优势来进行股票投资交易的一个过程。因此，作者按自身理解，量化交易主要分为以下几个方面：

- 数据获取：获取股票、期货、外汇、期权等资产市场的行情数据，并进行数据清洗、处理等操作。
- 特征工程：对数据进行特征工程，包括技术指标、行业指标、市场风险等，以提取有用的信息。
- 建模预测：利用机器学习算法进行建模预测，包括线性回归、决策树、随机森林等各种方式。
- 交易策略：根据预测结果，制定交易策略，包括止盈止损、跟踪止损、仓位管理等。
- 实盘交易：将策略应用到实际的投资操作中，包括实时监控、风险控制、风险管理等。

本文也正是通过这样的顺序来构建一个完整的量化交易策略的：

本文首先在第一部分介绍了一篇非常有趣的学术研究^[1]，该文章探究了如何通过随机森林来进行自动化的股票选取，本文将在一部分以该论文为基础，来训练 Random Forest 模型，以此来初步验证量化策略的可行性。

在文章的第二部分，我们将利用当前金融领域和计算机领域都相对比较火爆的 Xgboost 对股票收益进行预测，并将构建出各种相对比较完整的量化交易策略。

在文章的第三部分，作者考虑到量化交易最终是为了实践，以及真的通过计算机手段来获取股票的投资策略，因此在该部分，作者介绍了一个相对比较出名的真实自动化量化交易客户端 xtquant，通过该客户端，可以获取 Python API 接口，以此来真正实施搭建的量化交易策略。

在文章的最后一部分，我们介绍了一种相对低频的量化交易手段，其主要应用到了 Markowitz 的投资组合理论^[2-3]，优先通过投资者主观进行选股，之后通过线性规划来获得最佳的投资组合。

作者希望通过本文的撰写，不只是完成覃老师《金融大数据分析与量化交易》一门课的内容，更是对这学期所学知识的一种提炼和总结，并利用课上所讲、课下所思，真正完成量化交易策略的自我搭建，做一次完整的实践课题任务。

2 基于 Random Forest 的股票选取

在该部分，我们将以 Breitung, C. 的研究^[1]为出发点，探究通过随机森林算法进行股票选取是否切实可行。值得注意的是，由于 Breitung 的研究对于构建一个完整的量化交易策略仅仅提供了观测的一个方面，同时 Xgboost 算法实则是 Random Forest 以及 Boosting Trees 的一种有机结合，因此此处对于该部分内容仅作简单介绍，本文将把重心放到利用 Xgboost 进行量化策略搭建上。

值得注意的是，在此处仅对论文中一部分观点（这部分观点恰是如何通过随即森林对投资组合进行选取的过程）进行了实现，因此将主要介绍这一部分内容。

2.1 数据获取

代码 2-1 股票数据获取

```

1 import os
2 import pandas as pd
3 from tqdm import tqdm
4 import yfinance as yf
5
6 # 注意将其替换为自己的路径path/to/folder
7 folder_path = "E:\\大三上学科\\金融大数据分析与量化交易\\作业\\hw2"
8
9 def download_stock_data(tickers, start_date, end_date):
10     for ticker in tqdm(tickers):
11         res_file = os.path.join(folder_path, "data", "security_prices", f"{ticker}.csv")
12         if not os.path.isfile(res_file):
13             data = yf.download(ticker, start=start_date, end=end_date)
14             data.to_csv(res_file)
15
16         start_date, end_date = "1980-01-01", "2023-01-01"
17         ticker_df = pd.read_csv(os.path.join(folder_path, "data", "ticker_cik.txt"), sep="\t")
18         tickers = ticker_df["ticker"].str.upper().tolist()
19         download_stock_data(tickers, start_date, end_date)

```

代码2-1利用yfinance库，下载所有指定股票的历史行情数据，并保存到指定文件夹。其中folder_path是指定文件夹的路径。再利用本文中代码进行复现时，需要将其换为自己的路径，且此处使用的是绝对路径，后面代码为方便操作，均使用相对路径。

download_stock_data 函数的输入参数为 tickers , start_date , end_date, 其遍历 txt 文件中的每个 ticker (股票代码), 使用yfinance.download() 函数下载从 start_date 到end_date的股票数据, 并保存到指定文件夹。

2.2 特征工程——技术指标

由于数据量较大, 我们以月频数据来进行计算, 因此本部分内容主要包括两部分:一部分是月收益率的计算, 一部分为月技术指标的计算。

代码 2-2 月收益率计算

```

1 import os
2 import pandas as pd
3 from tqdm import tqdm
4
5
6 def calculate_monthly_returns():
7     path = os.path.join("data", "security_prices")
8     files = os.listdir(path)
9     dfs = []
10
11     for file in tqdm(files):
12         df = pd.read_csv(os.path.join(path, file), usecols=["Date", "Adj Close"])
13         df['Date'] = pd.to_datetime(df['Date'])
14         df.set_index('Date', inplace=True)
15         monthly_returns = df.resample('M').last().pct_change().reset_index().dropna()
16         monthly_returns["investment_month"] = monthly_returns["Date"].dt.to_period('M').astype(str)
17         monthly_returns["ticker"] = file[:-4]
18         monthly_returns.rename(columns={"Adj Close": "ret"}, inplace=True)
19         monthly_returns = monthly_returns[["ticker", "investment_month", "ret"]]
20         dfs.append(monthly_returns)
21
22     pd.concat(dfs).to_csv(os.path.join("data", "monthly_returns.csv"), index=False)
23
24
25 def monthly_turnovers():
26     path = os.path.join("data", "security_prices")
27     files = os.listdir(path)
28     dfs = []
29
30     for file in tqdm(files):

```

```

31         df = pd.read_csv(os.path.join(path, file), usecols=['Date', 'Volume', 'Close'])
32         df['Date'] = pd.to_datetime(df['Date'])
33         df.set_index('Date', inplace=True)
34         monthly_returns = df.resample('M').mean().reset_index().dropna()
35         monthly_returns['investment_month'] = monthly_returns['Date'].dt.to_period('M').astype(str)
36         monthly_returns['ticker'] = file[-4]
37         monthly_returns['Turnover'] = monthly_returns['Volume'] * monthly_returns['Close']
38         monthly_returns = monthly_returns[['ticker', 'investment_month', 'Volume', 'Turnover']]
39         dfs.append(monthly_returns)
40
41     pd.concat(dfs).to_csv(os.path.join("data", "monthly_volumes.csv"), index=False)
42
43
44 calculate_monthly_returns()
45 monthly_turnovers()

```

代码 2-3 技术指标计算

```

1 import os
2 import random
3 import warnings
4 import pandas as pd
5 from tqdm import tqdm
6 import ta
7 from datetime import datetime
8 from dateutil.relativedelta import relativedelta
9
10 warnings.filterwarnings('ignore')
11 pd.set_option('display.float_format', lambda x: '%.5f' % x)
12
13 def create_indicators():
14     path = os.path.join("data", "security_prices")
15     files = os.listdir(path)
16     random.shuffle(files)
17
18     for file in tqdm(files):
19         ticker = file[-4]
20         res_file = os.path.join("data", "technical_indicators", file)

```

```
22     if not os.path.isfile(res_file):
23         try:
24             data = pd.read_csv(os.path.join(path, file), usecols=["Date", "Open",
25 "High", "Low", "Close", "Volume"])
26             df = ta.add_all_ta_features(data, "Open", "High", "Low", "Close", "
27 Volume", fillna=True)
28             df = df.drop(["Open", "High", "Low", "Volume"], axis=1)
29
30             for col in df.columns:
31                 if col not in ["Date", "Close"]:
32                     df[col + "_rel"] = df[col] / df["Close"]
33
34             df = df.drop("Close", axis=1)
35             df["ticker"] = ticker
36             df.to_csv(res_file, index=False)
37
38         except:
39             pass
40
41
42
43
44     def monthly_indicators():
45         dates = [f'{year}-{month:02d}-01' for year in range(1990, 2023) for month in
46         range(1, 13)]
47         res_file = os.path.join("data", "monthly_technical_indicators.csv")
48         files = os.listdir(os.path.join("data", "technical_indicators"))
49
50         monthly_indicators = []
51
52         for file in tqdm(files):
53             file_dfs = []
54             df = pd.read_csv(os.path.join("data", "technical_indicators", file))
55             min_date, max_date = min(df["Date"]), max(df["Date"])
56             max_date_plus_one_month = (datetime.strptime(max_date, "%Y-%m-%d") +
57             relativedelta(months=1)).strftime("%Y-%m-%d")
58
59             for date in dates:
60                 if min_date < date < max_date_plus_one_month:
61                     df_date = df[df["Date"] < date]
62
63                     if not df_date.empty:
64                         df_last_date = df_date.iloc[-1:].copy()
65                         df_last_date["investment_month"] = date[:7]
66                         file_dfs.append(df_last_date)
```

```

60
61     if file_dfs :
62         file_df = pd.concat( file_dfs )
63         cols = [ col for col in file_df .columns if col not in [ " ticker ", " Date ", " investment_month" ] ]
64         file_df = file_df [ [ " ticker ", " Date ", " investment_month" ] + cols ]
65         monthly_indicators.append( file_df )
66
67     pd.concat(monthly_indicators) .to_csv( res_file , index=False )
68
69
70     create_indicators ()
71     monthly_indicators ()

```

在代码2-3中，我们使用了 talib 库来计算技术指标，通过compute_ta_indicators()函数集中管理指标计算逻辑。之后通过monthly_indicators按月汇总所有股票的数据。

2.3 随机森林 Random Forest 分类模型

下面我们需要基于前面特征工程构建出的技术指标，构建一个随机森林模型，并以此来预测每只股票的回报表现，并生成预测结果。

2.3.1 导入必要的库

代码 2-4 随机森林模型导入必要的库

```

1 import os
2 import pickle
3 import pandas as pd
4 import numpy as np
5 from tqdm import tqdm
6 from sklearn.ensemble import RandomForestClassifier
7 from scipy.stats.mstats import winsorize

```

这里的os、pickle、pandas、numpy、tqdm、sklearn、scipy都比较常见，其实os用于文件路径的管理，pickle用于保存模型，pandas用于数据处理，numpy用于数学计算，tqdm用于进度条显示，sklearn.ensemble.RandomForestClassifier用于随机森林模型。

2.3.2 读取数据函数

代码 2-5 随机森林模型读取数据函数

```

1 def load_and_preprocess_data(train_test_boundary):
2     technical_indicators = pd.read_csv(os.path.join("data", "monthly_technical_indicators.csv"))
3
4     # Drop low liquidity stocks 去除流动性不好的股票
5     monthly_volumes = pd.read_csv(os.path.join("data", "monthly_volumes.csv"))
6     monthly_volumes = monthly_volumes[monthly_volumes["Turnover"] >= 1000000]
7     technical_indicators = pd.merge(technical_indicators, monthly_volumes[["ticker",
8         "investment_month"]], on=["ticker", "investment_month"], how="inner")
9     # 看一下还剩多少股票
10    print(len(technical_indicators["ticker"].unique()))
11
12    monthly_returns = pd.read_csv(os.path.join("data", "monthly_returns.csv"))
13
14    monthly_returns["ret"] = winsorize(monthly_returns["ret"], limits=(0, 0.01))
15    technical_indicators = pd.merge(technical_indicators, monthly_returns, how="inner", on=["ticker", "investment_month"])
16    # 计算出每月回报的中位数
17    monthly_medians = technical_indicators[["investment_month", "ret"]].groupby("investment_month").median()
18    monthly_medians.rename(columns={"ret": "median_ret"}, inplace=True)
19    # 技术指标与月度收益率进行合并
20    technical_indicators = pd.merge(technical_indicators, monthly_medians, how="inner", on="investment_month")
21
22    # 计算目标变量
23    technical_indicators["true_return_class"] = [1 if value > technical_indicators["median_ret"].iloc[i] else 0 for i, value in enumerate(technical_indicators["ret"])]
24
25    # 进一步数据清洗，去除异常值、剔除极端值
26    technical_indicators = technical_indicators.replace([np.inf, -np.inf], np.nan).dropna()
27
28    for col in tqdm(technical_indicators.columns):
29        if col not in ["Date", "investment_month", "ticker", "median_ret", "ret"]:
30            technical_indicators = technical_indicators[(technical_indicators[col] > -np.finfo(np.float32).max) & (technical_indicators[col] < np.finfo(np.float32).max)]

```

```

31
32     # 划分训练集和测试集
33     technical_indicators_train = technical_indicators [ technical_indicators [ "Date" ] <
34         train_test_boundary ].drop([ "Date", "investment_month", "ticker", "median_ret", "ret" ],
35         axis=1)
36     technical_indicators_test = technical_indicators [ technical_indicators [ "Date" ] >=
37         train_test_boundary ]
38
39     return technical_indicators_train , technical_indicators_test

```

在该部分，函数load_and_preprocess_data主要作用就是读取已经相对处理好的数据，之后我们利用列[“Turnover”]（换手率）的限制条件，去除流动性不好的股票（实现方式是通过merge函数）。之后我们对刚刚已经处理好的月收益率和技术指标两个表格DataFrame进行合并，之后计算出每月回报的中位数，并生成目标变量[“true_return_class”]列。

目标变量的生成方式如下：

$$\text{true_return_class} = \begin{cases} 1, & \text{if } \text{ret} > \text{median_ret} \\ 0, & \text{if } \text{ret} \leq \text{median_ret} \end{cases} \quad (2-1)$$

即股票的回报若大于该月的回报中位数，则标签为 1（收益较高），否则为 0（认为收益较低）。

之后，为了严谨，我们对数据进行了一些数据清洗，去除异常值、剔除极端值，并将数据分为训练集和测试集。

在剔除异常值中，我们loop遍历了每一列，通过np.finfo(np.float32).max返回float32数据类型的最大值；通过-np.finfo(np.float32).max返回float32数据类型的最小值；并通过这些边界值，筛选掉过大或者过小的数值。

这里划分训练集是通过日期train_test_boundary进行划分的，一般我们机器学习常用划分训练集的方法为sklearn.model_selection.train_test_split()，但对于股票时间序列数据，更合理的方式是如上所示通过时间进行划分。

2.3.3 Random Forest 模型训练

代码 2-6 随机森林模型训练

```

1 def train_and_save_model( technical_indicators_train ):
2     # 提取特征和标签
3     labels = list ( technical_indicators_train [ "true_return_class" ])

```

```

4     technical_indicators_train = technical_indicators_train .drop(['true_return_class',
5 ], axis=1)
6
7     # 创建随机森林对象并进行训练
8     rf = RandomForestClassifier(n_estimators=100, n_jobs=-1, random_state=1)
9     model = rf .fit ( technical_indicators_train , labels )
10    pickle.dump(model, open(os.path.join('models', 'rf.save'), 'wb'))

```

我们定义了一个名为train_and_save_model的函数,接受一个参数 technical_indicators_train , 该参数是包含技术指标和标签的数据集 DataFrame; 首先从 DataFrame 中提取标签 labels , 并删除标签列[‘true_return_class’]; 只保留技术指标数据。

之后创建RandomForestClassifier对象, 并进行训练, 并保存模型 (使用pickle包)。这里的超参数, n_estimators是树的数量, 即 100 棵树; n_jobs是并行数, 其为-1 表示使用所有可用的 CPU 核来并行训练模型; random_state是随机种子, 设置其为 1 以保证结果的可重复性。

2.3.4 模型预测

代码 2-7 随机森林模型预测

```

1 def predict_and_save_results (investment_months, model, technical_indicators_test ):
2     # 创建tqdm进度条, 遍历所有的投资月份
3     for date in tqdm(investment_months):
4         print("Predicting Date " + str(date))
5         # 筛选出当前投资月份的所有数据
6         investment_df = technical_indicators_test [ technical_indicators_test ["
7 investment_month"] == date]
8
8         # 使用模型对当前投资月份的股票进行预测
9         predicted_probas = model.predict_proba(investment_df.drop(['true_return_class
10 ', 'ret', 'ticker', 'Date', 'investment_month', 'median_ret'], axis=1))
11         # 提取出类别为1 (表示为“正”类, 即有较高回报的类别的预测概率)
12         predicted_probas_class_1 = [value[1] for value in predicted_probas]
13
13         # 保存最终的预测结果
14         result_df = investment_df[['ticker', 'Date', 'investment_month', 'ret',
15         'true_return_class']]
15         result_df ["probability"] = predicted_probas_class_1      # 添加一列“
16         probability”, 表示类别1的预测概率
16         result_df = result_df .sort_values ("probability", ascending=False)
17         # 设置预测截断点

```

```

18     cutoff_proba = 0.5
19     predicted_values = [1 if value > cutoff_proba else 0 for value in list(
20         result_df[ "probability" ])]
21     result_df[ "predicted_class" ] = predicted_values
22     result_df.to_csv(os.path.join( "results", "ML_results", "{}.csv".format(date) )
, index=False)

```

在上部分代码中，我们实现了随机森林模型的训练函数，接下来需要来看训练出的模型的预测效果，我们定义了一个名为 predict_and_save_results 的函数，用于使用训练好的模型对测试数据进行预测，并将预测结果保存下来。

在代码中，我们通过注释的形式，已经对该部分代码有较为详细的解释，因此此处不再详细进行展开。

2.3.5 模型训练及预测 main 函数

代码 2-8 随机森林模型训练及预测 main 函数

```

1 # 首先传递训练集和测试集的分界日期参数
2 train_test_boundary = "2002-12-31"
3 investment_months = [f" {year}-{month:02d}" for year in range(2003, 2021) for month in
4 range(1, 13)]
5
5 # 读取数据并预处理
6 technical_indicators_train , technical_indicators_test = load_and_preprocess_data(
7 train_test_boundary)
8 # 训练模型并保存
9 train_and_save_model(technical_indicators_train)
10
10 # 读取训练好的模型，并进行预测
11 model = pickle.load(open(os.path.join('models', 'rf.save'), 'rb'))
12 predict_and_save_results(investment_months, model, technical_indicators_test)

```

3 基于 Xgboost 的股价预测模型及量化交易策略

在我们从论文^[1]学习到，股票表现概率这一关键概念后，在该部分，作者将依据自身理解，来构造完整的量化交易策略。

3.1 Xgboost v.s. Random Forest

为什么选择 Xgboost 作为量化交易中预测股价的模型呢？而不是像上一部分探索关系一样选择随机森林模型呢？下表3-1即展示了 Xgboost 与 Random Forest 的不同点。

表 3-1 XGBoost 与 Random Forest 的对比

特性	XGBoost	Random Forest
模型类型	顺序构建（Boosting）	并行构建（Bagging）
偏差与方差	更低的偏差（通过优化损失函数）	更低的方差（通过平均多个模型的结果）
特征重要性	提供详细的重要性评估（基于增益）	提供基于分裂次数和减少的误差的重要性评估
计算速度	较慢（依赖于树的顺序生成）	较快（并行训练）
正则化	内置正则化（防止过拟合）	无直接正则化，依靠随机性降低过拟合风险
缺失值处理	内置缺失值处理（分裂时自动选择最佳路径）	缺失值需提前填充
适用数据类型	对稀疏、高维特征友好	对低维、相对均衡的特征表现更好

由于 XGBoost 通过提升树（boosting）的方式构建模型，可以有效捕捉数据中的复杂非线性关系，而 Random Forest 通过简单的树平均（bagging）方式，虽然也能捕捉非线性关系，但在某些情况下可能不如 XGBoost 的表现好；同时，XGBoost 加入了正则化项，可以防止过拟合，提高模型的泛化能力，这对于金融市场噪声的管理非常有效。因此我们在整个量化交易策略搭建过程中使用了 Xgboost 作为主要模型来实现。

3.2 数据处理

3.2.1 数据获取

我们考虑国内的股票市场进行投资，在本文下文会介绍国际投资组合策略的搭建，其不仅仅购买了国外的风险资产，同时也不仅仅局限于股票这一类资产。在本部分，我们仅考虑国内 A 股市场，且不考虑标注 ST 的股票。

我们首先选取了自 2019 年 12 月 17 日至 2024 年 12 月 16 日的股票历史数据，数据来源于 CSMAR 中国经济金融研究数据库（国泰安数据库），包括字段如下表??所示。

序号	字段	数据类型	字段标题	单位
1	Stkcd	Nvarchar	证券代码	没有单位
2	Trddt	Datetime	交易日期	没有单位
3	Opnprc	decimal	日开盘价	元/股
4	Hiprc	decimal	日最高价	元/股
5	Loprc	decimal	日最低价	元/股
6	Clspkc	decimal	日收盘价	元/股
7	Dnshtrtd	decimal	日个股交易股数	股
8	Dnvaltrd	decimal	日个股交易金额	元
9	Dsmvosd	decimal	日个股流通市值	千元
10	Dsmvtll	decimal	日个股总市值	千元
11	Dretwd	decimal	考虑现金红利再投资的日个股回报率	没有单位
12	Dretnd	decimal	不考虑现金红利的日个股回报率	沛有单位
13	Adjprcwid	decimal	考虑现金红利再投资的收盘价的可比价格	元/股
14	Adjprcnd	decimal	不考虑现金红利的收盘价的可比价格	元/股
15	Cnshtrtdtl	decimal	综合日市场交易总股数	股
16	Cnvaltrdtl	decimal	综合日市场交易总金额	元
17	Cdretwdeq	decimal	考虑现金红利再投资的综合日市场回报率(等权平均法)	没有单位
18	Cdretmdeq	decimal	不考虑现金红利的综合日市场回报率(等权平均法)	没有单位
19	Cdretwdos	decimal	考虑现金红利再投资的综合日市场回报率(流通市值加权平均法)	沛有单位
20	Cdretmdos	decimal	不考虑现金红利的综合日市场回报率(流通市值加权平均法)	沛有单位
21	Cdretwdtl	decimal	考虑现金红利再投资的综合日市场回报率(总市值加权平均法)	沛有单位
22	Cdretmdtl	decimal	不考虑现金红利的综合日市场回报率(总市值加权平均法)	沛有单位
23	Cdnstckal	decimal	计算综合日市场回报率的有效公司数量	沛有单位

3.2.2 数据清洗

在 CSMAR 下载下来数据之后（注意此处没有使用 CSMAR 提供的 API），主要选取了一些有用的列进行分析，其他列全部丢失掉。

同时下载了 2019 年 12 月 17 日至 2024 年 12 月 16 日的沪深 A 股市场大盘数据，考虑到大盘数据可能会对模型预测效果有积极影响或作用，因此在自变量部分加入了大盘市场的数据。

代码 3-1 Xgboost 模型数据清洗 1

```

1 df = pd.read_csv('贵州茅台.csv', encoding='utf-8',
2                   usecols=['Trddt', 'Opnprc', 'Hiprc',
3                           'Loprc', 'Clspkc', 'Dnshtrtd',
4                           'Dnvaltrd', 'Dsmvosd', 'Dsmvtll',
5                           'Dretwd', 'Dretnd', 'Adjprcwid',
6                           'Adjprcnd'])
7

```

```

8   print(df.head(4).T)
9
10  pd_a = pd.read_csv('沪深A股.csv', encoding='utf-8')
11  print(pd_a.head(3).T)
12  a_index = pd_a['Markettype'] == 5
13  df_a = pd_a[a_index]
14  print(df_a.head(3).T)
15  df_a.drop(['Markettype'], axis=1, inplace=True)
16
17  # 将茅台的数据与A股市场的数据合并
18  df = df.merge(df_a, on='Trddt', how='left')
19  print(df.head(3).T)

```

数据切片如下图3-1所示：

	0	1	2
Trddt	2019-12-17	2019-12-18	2019-12-19
Opnprc	1149.7	1174.0	1163.0
Hiprc	1173.6	1175.2	1163.0
Loprc	1143.03	1164.0	1152.0
Clsprc	1169.98	1168.0	1157.4
Dnshtrtd	4261696	1964009	1955230
Dnvaltrd	4937041950.0	2295963803.0	2262598597.0
Dsmvosd	1469726302.04	1467239030.4	1453923333.72
Dsmvtll	1469726302.04	1467239030.4	1453923333.72
Dretwd	0.019217	-0.001692	-0.009075
Dretnd	0.019217	-0.001692	-0.009075
Adjprcwd	8759.14516	8744.321738	8664.964024
Adjprcnd	6584.320386	6573.1775	6513.523663
Cnshtrtdtl	60594887369	51467667008	45300990922
Cnvaltrdtl	591760373275.75	512729126984.539978	439067241164.590027
Cdretwdeq	0.013201	0.001334	0.006929
Cdretmdeq	0.013194	0.001332	0.006926
Cdretwdos	0.012758	-0.001217	0.001252
Cdretmdos	0.012756	-0.001224	0.001246
Cdretwdtl	0.013419	-0.001043	0.000887
Cdretmdtl	0.013411	-0.001049	0.000881

图 3-1 Xgboost 模型数据清洗数据切片展示

3.3 特征工程

由于市场中的数据，我们只能通过历史数据进行预测，因此这里构建了滞后特征 $x_{t-1}, x_{t-2}, x_{t-3}, x_{t-4}, x_{t-5}$ ，同时构建了滑动窗口特征，使用滑动窗口计算均值、最大值、最小值等统计量，如rolling_mean、rolling_max、rolling_std 等。

具体的特征工程示例代码如下面的代码3-2所示：

代码 3-2 Xgboost 模型特征工程

```

1 # 设置滞后
2 lags = [1, 2, 3, 4, 5] # 滞后1天、2天、3天、4天、5天的数据
3
4 # 看一下数据中的列
5 columns = df.columns
6 print(columns)
7 columns = columns.difference(["Trddt"]) # 去掉日期和收盘价列
8
9 # 构造滞后特征
10 for lag in lags:
11     for col in columns:
12         df[f"{col}_lag{lag}"] = df[col].shift(lag)
13
14 # 滑动窗口特征
15 window_sizes = [3, 5, 10]
16 for window in window_sizes:
17     df[f"Clsprc_roll_mean_{window}"] = df["Clsprc"].shift(1).rolling(window).mean()
18     df[f"Clsprc_roll_std_{window}"] = df["Clsprc"].shift(1).rolling(window).std()
19
20 # 删除NaN值（由于滞后和滑动窗口引入的）
21 df.dropna(inplace=True)
22
23 columns = columns.difference(["Clsprc"]) # 去掉日期和收盘价列
24
25 df.drop(columns=columns, inplace=True) # 删除原有的当下日期数据列
26
27 # 特征与目标
28 features = df.columns.difference(["Trddt", "Clsprc"])
29 target = "Clsprc"

```

需要注意的是，在特征工程的过程中，我们并没有用到较多的上课学到的技术指标，及也没有运用到TA-lib库，在本部分制造的两种类型的指标分别为：

- 滞后特征：使用 shift 函数将数据向前移动，得到滞后的数据。
- 滑动窗口特征：使用 shift 函数将数据向前移动，然后使用 rolling 函数计算统计量。

这只是本文介绍的最简单的模型，特征选取均来自作者直觉，复杂的模型会在后面再详细介绍。

3.4 模型训练

3.4.1 划分训练集和测试集

一般在机器学习或深度学习框架里,往往划分训练集和测试集采取 sklearn 库的 `train_test_split` 函数。并指定随机数种子,以保证结果的可持续性,但由于时间序列数据的特殊性,在此处划分训练集和测试界的方法依据时间序列特征进行划分(像前面对论文结果验证的过程相类似),具体代码如下:

代码 3-3 Xgboost 模型数据划分训练集和测试集

```

1 # 划分训练集和测试集
2 train_data = df[df[ "Trddt" ] <= "2023-12-31"]
3 test_data = df[df[ "Trddt" ] > "2023-12-31"]
4
5 train_X = train_data [ features ]
6 train_y = train_data [ target ]
7
8 test_X = test_data [ features ]
9 test_y = test_data [ target ]

```

其中, `features` 和 `target` 分别表示自变量和目标变量。在上文中也已介绍,已经完整定义好,因此本部分不再展开赘述。

3.4.2 Xgboost 模型训练

下面给出了简单的 Xgboost 模型训练代码,具体如下:

代码 3-4 Xgboost 模型训练示例代码 1

```

1 import xgboost as xgb
2 from sklearn . metrics import mean_squared_error, mean_absolute_error
3
4
5 # 定义XGBoost训练的参数
6 params = {
7     'objective' : 'reg:squarederror', # 回归问题, 使用均方误差作为损失函数
8     'max_depth': 7, # 树的最大深度, 控制模型的复杂度 7
9     'eta' : 0.034, # 学习率, 控制每次迭代更新的步长 0.033
10    'subsample': 0.8, # 训练时使用80%的样本, 防止过拟合 0.8
11    'colsample_bytree' : 0.8 # 每棵树训练时, 随机选择80%的特征 0.8
12 }
13
14 # 设置训练的最大迭代轮数和早停轮数
15 num_boost_round = 30000 # 最大训练轮数

```

```

16     early_stopping_rounds = 1000 # 如果在1000轮内验证集的误差没有改善，则提前停止训练
17
18     # 训练模型
19     evals_result = {} # 存储每轮训练的评估结果
20     bst = xgb.train(
21         params, # 使用上述定义的参数
22         xgb.DMatrix(X_train, label=y_train), # 训练集数据
23         num_boost_round, # 最大迭代次数
24         evals=[(xgb.DMatrix(X_test, label=y_test), 'eval')], # 验证集
25         early_stopping_rounds=early_stopping_rounds, # 设置早停机制
26         evals_result = evals_result # 存储训练过程中的评估结果
27     )
28
29     # 使用训练好的模型进行预测
30     y_pred = bst.predict(xgb.DMatrix(X_test)) # 对测试集进行预测
31
32     # 评估模型性能：计算RMSE（均方根误差）
33     rmse = mean_squared_error(y_test, y_pred, squared=False) # squared=False返回RMSE而不是MSE
34     print(f"RMSE: {rmse}") # 打印RMSE结果
35
36     # 打印训练过程中的评估结果
37     print("Training process evaluation result :")
38     print(evals_result)

```

对于 Xgboost 的相关参数，为提高模型的泛化能力，下面给出以下几个调参的方式。

(1) 贝叶斯调参

代码 3-5 Xgboost 模型贝叶斯调参示例代码

```

1 import optuna
2 import xgboost as xgb
3 from sklearn.metrics import mean_squared_error
4 from sklearn.model_selection import train_test_split
5
6 # 自定义目标函数
7 def objective(trial):
8     # XGBoost 参数空间
9     param = {
10         'objective': 'reg:squarederror',
11         'eval_metric': 'rmse',

```

```

12     'max_depth': trial . suggest_int ('max_depth', 3, 10),
13     'learning_rate' : trial . suggest_float ('learning_rate' , 0.01, 0.1, log=True),
14     'subsample': trial . suggest_float ('subsample', 0.7, 1.0),
15     'colsample_bytree': trial . suggest_float ('colsample_bytree' , 0.7, 1.0),
16     'n_estimators': trial . suggest_int ('n_estimators' , 100, 1000, step=100),
17     'gamma': trial . suggest_float ('gamma', 0, 5), # 控制是否后剪枝
18     'lambda': trial . suggest_float ('lambda' , 0, 1),
19     'alpha': trial . suggest_float ('alpha' , 0, 1),
20     'min_child_weight': trial . suggest_int ('min_child_weight' , 1, 10),
21     'scale_pos_weight': trial . suggest_float ('scale_pos_weight' , 1, 10),
22 }
23
24 # 创建 XGBoost 模型
25 model = xgb.XGBRegressor(**param)
26
27 # 训练模型
28 model. fit (X_train, y_train, eval_set=[(X_test, y_test)], verbose=False)
29
30 # 预测并评估性能
31 y_pred = model.predict (X_test)
32 rmse = mean_squared_error(y_test, y_pred, squared=False)
33
34 return rmse
35
36 # 通过 Optuna 进行贝叶斯优化
37 study = optuna.create_study (direction ='minimize')
38 study.optimize( objective , n_trials =50) # 进行 50 次试验
39
40 # 输出最佳参数
41 print(f"Best params: {study.best_params}")
42 print(f"Best RMSE: {study.best_value}")
43
44 # 使用最佳参数训练模型
45 best_params = study.best_params
46 best_model = xgb.XGBRegressor(**best_params)
47 best_model. fit (X_train, y_train)
48
49 # 评估模型性能
50 y_pred = best_model.predict (X_test)
51 rmse = mean_squared_error(y_test, y_pred, squared=False)
52 print(f"最终优化后的RMSE: {rmse}")

```

其中，`trail` 是由 Optuna 库提供的超参数优化类，用于在给定范围内选择超参数值。我们定义了参数搜索空间，`train . suggest_*`方法允许在指定搜索空间的范围，包括整数、浮点数和布尔值。

Optuna 的贝叶斯优化算法可以自动选择超参数的最佳值，并在一定次数的试验中找到最优的超参数组合；这里规定 Optuna 的优化方向是最小化目标函数（RMSE），超参数搜索的试验次数为 50。

(2) 网格搜索

代码 3-6 Xgboost 模型网格搜索示例代码

```

1  from sklearn.model_selection import GridSearchCV
2  import xgboost as xgb
3  from sklearn.metrics import mean_squared_error
4
5  # 定义参数网格
6  param_grid = {
7      'max_depth': [3, 5, 7, 10],  # 决策树的最大深度
8      'learning_rate': [0.01, 0.05, 0.1],  # 学习率
9      'subsample': [0.7, 0.8, 0.9, 1.0],  # 子采样率
10     'colsample_bytree': [0.7, 0.8, 0.9, 1.0],  # 特征采样率
11     'n_estimators': [100, 300, 500],  # 基础模型的数量
12     'gamma': [0, 1, 5],  # 控制后剪枝
13     'min_child_weight': [1, 3, 5],  # 子节点的最小权重
14 }
15
16 # 创建 XGBoost 模型
17 xgb_model = xgb.XGBRegressor(objective='reg:squarederror', eval_metric='rmse')
18
19 # 使用 GridSearchCV 进行网格搜索
20 grid_search = GridSearchCV(
21     estimator=xgb_model,
22     param_grid=param_grid,
23     scoring='neg_mean_squared_error',  # 负的均方误差 (sklearn中, 分数越大越好, 所以加负号)
24     cv=3,  # 3 折交叉验证
25     verbose=1,
26     n_jobs=-1  # 并行计算
27 )
28
29 # 训练模型, 寻找最佳参数
30 grid_search.fit(X_train, y_train)
31

```

```

32 # 输出最佳参数和对应的分数
33 print(f"Best params: {grid_search.best_params_}")
34 print(f"Best RMSE: {-grid_search.best_score_ ** 0.5}") # 将负的均方误差取平方根为
RMSE
35
36 # 使用最佳参数训练模型
37 best_params = grid_search.best_params_
38 best_model = xgb.XGBRegressor(objective='reg:squarederror', eval_metric='rmse', **
best_params)
39 best_model.fit(X_train, y_train)
40
41 # 在测试集上评估性能
42 y_pred = best_model.predict(X_test)
43 rmse = mean_squared_error(y_test, y_pred, squared=False)
44 print(f"最终优化后的RMSE: {rmse}")

```

这里不再过多介绍网格超参数优化，实际应用中二者可以结合使用，原因是

- 贝叶斯优化虽然优化速度较快，但容易找到局部极小点
- 网格优化运行速度较慢，但可以帮助贝叶斯优化寻找全局最优解

最终，本文章所选取的 Xgboost 模型的最有超参数定义为：

表 3-2 最佳参数

参数	值
max_depth	4
learning_rate	0.022760054202930168
subsample	0.7282082696017346
colsample_bytree	0.8487950420954548
n_estimators	500
gamma	1.6396083949299796
lambda	0.8595110181042634
alpha	0.41177244276583247
min_child_weight	9
scale_pos_weight	9.08393318319212

如果出现数据没有完全收敛的问题，可以考虑增加 n_trails 参数，以实现多次的训练和测试，以找到最优的超参数组合。

3.4.3 使用 Xgboost 对预测集进行预测

代码 3-7 Xgboost 模型预测示例代码

```

1 import xgboost as xgb
2 from sklearn.metrics import mean_squared_error
3
4 # 使用最优参数重新创建模型
5 best_model = xgb.XGBRegressor(
6     objective='reg:squarederror',
7     eval_metric='rmse',
8     **study.best_params # 替换为 Optuna 的最佳参数
9 )
10
11 # 用训练集训练模型
12 best_model.fit(X_train, y_train)
13
14 # 在测试集上进行预测
15 y_pred = best_model.predict(X_test)
16
17 # 评估预测结果
18 rmse = mean_squared_error(y_test, y_pred, squared=False)
19 print(f"使用最优参数的模型预测 RMSE: {rmse}")
20
21 # 输出部分预测结果对比
22 result_df = test_data.copy()
23 result_df['Predicted'] = y_pred
24 print(result_df[['Trddt', target, 'Predicted']].head())

```

此处不再过多解释代码中的内容，最终的预测 RMSE 及预测结果切片如下图3-2所示：

使用最优参数的模型预测 RMSE: 35.200310949054646			
	Trddt	Clsprc	Predicted
981	2024-01-02	1685.01	1735.763672
982	2024-01-03	1694.00	1695.333130
983	2024-01-04	1669.00	1713.091309
984	2024-01-05	1663.36	1677.991821
985	2024-01-08	1643.99	1659.710205

图 3-2 Xgboost 模型预测结果

我们把预测的结果与真实值放到一个图表中（如图3-3所示），可以看到预测的结

果与真实值基本吻合，只是预测的结果相较于真实值出现了相对明显的滞后趋势特征，这可能是导致量化策略承担较大损失的一个重要原因，应当引起我们的注意（本文同样将在后面对于策略的优化过程中关注这个问题）。

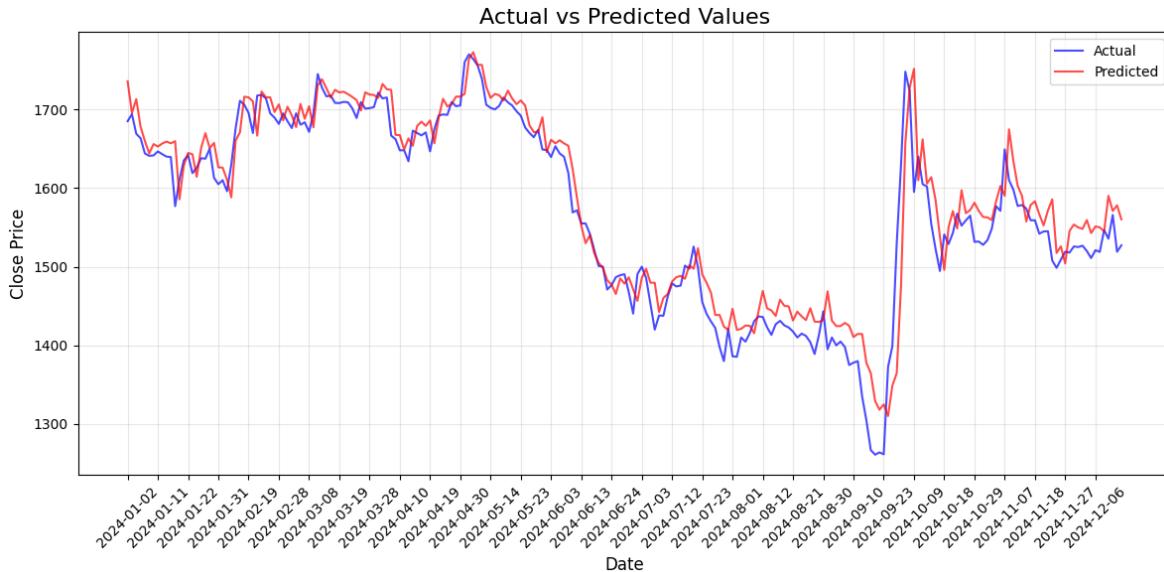


图 3-3 Xgboost 模型预测结果对比

在这里，重复运行可能导致结果不完全相同，其原因在于可能对于超参数的优化没有到达最优水平，在此处我们将 `n_trails` 参数设置为 100，以达到更好的模型性能。

最终训练出来的模型参数为：

表 3-3 最佳参数

参数	值
<code>max_depth</code>	4
<code>learning_rate</code>	0.0798196888578835
<code>subsample</code>	0.7761594790264786
<code>colsample_bytree</code>	0.8645703054188993
<code>n_estimators</code>	600
<code>gamma</code>	3.054644380206763
<code>lambda</code>	0.9319324613754306
<code>alpha</code>	0.2642698984952335
<code>min_child_weight</code>	8
<code>scale_pos_weight</code>	3.610332158210867

最终预测结果与真实值比较的图像为：

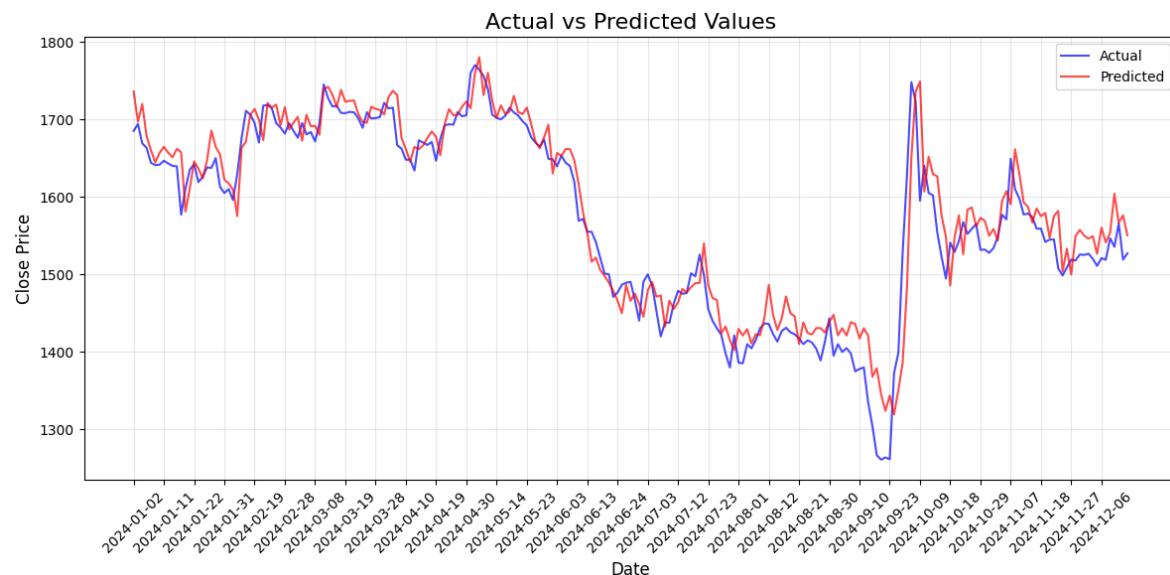


图 3-4 调整后的 Xgboost 模型预测结果对比

但是值得注意的是：模型对于股价趋势的判断完全准确，因此作为量化投资的底层模型是相对可靠的。

在处理完这些数据后，我们似乎发现模型对于股价的预测结果还可以，在后面，我们将进行交易信号的发送，以进行历史的数据回测，尝试该量化交易能给我们带来什么样的收益。

3.5 量化交易策略搭建

3.5.1 交易信号及交易策略

首先，我们先定义一个空的 DataFrame，用于储存最终的预测结果和交易信号：

代码 3-8 交 trade 信号的 DataFrame 定义

```

1 # 进行预测
2 y_pred = best_model.predict(X_test)
3
4 trade = pd.DataFrame()
5
6 trade[ "Trddt" ] = result_df[ "Trddt" ]
7 trade[ "Clsprc" ] = result_df[ "Clsprc" ]
8 trade[ 'Predicted_Clsprc' ] = y_pred
9
10 print( trade .head(20))
11
12 trade .to_csv( "trade .csv" , index=False)
13 trade = pd.read_csv('trade .csv')

```

我们产生交易信号的方式非常简单，即基于预期收盘价与前一日真实收盘价的比较：

代码 3-9 交易信号及持仓状况定义

```

1 # 初始化持仓状态列
2 trade[ 'Position' ] = 0 # 0 表示无持仓, 1 表示持仓
3
4 # 生成买入和卖出信号
5 trade[ 'Signal' ] = 0
6 for i in range(1, trade.shape[0]): # 从第二天开始, 因为第一天没有前一天的持仓状态
7     if trade[ 'Prediction_Error_Perc' ][i] > 0.01 and trade[ 'Position' ][i-1] == 0:
8         trade[ 'Signal' ][i] = 1
9         trade[ 'Position' ][i] = 1
10    elif trade[ 'Prediction_Error_Perc' ][i] < -0.003 and trade[ 'Position' ][i-1] == 1:
11        trade[ 'Signal' ][i] = -1
12        trade[ 'Position' ][i] = 0
13    else:
14        trade[ 'Position' ][i] = trade[ 'Position' ][i-1]

```

依据买入或卖出信号，我们就可以得到，在每天的交易信号中，我们应该持有或不持有股票，从而完成我们量化交易的策略构建。下面代码3-10表示了生成交易信号及持仓状况的完整代码，并计算了一直持有该股票的收益率及策略的收益率。

代码 3-10 量化交易策略完整代码

```

1 import pandas as pd
2 import numpy as np
3
4 # 将日期设置为索引
5 trade[ 'Trddt' ] = pd.to_datetime(trade[ 'Trddt' ])
6 trade.set_index('Trddt', inplace=True)
7
8 # 计算预测增长百分比
9 trade[ 'Prediction_Error_Perc' ] = (trade[ 'Clspred' ] - trade[ 'Predicted_Clspred' ]) /
10 trade[ 'Predicted_Clspred' ] * 100
11
12 # 初始化持仓状态列
13 trade[ 'Position' ] = 0 # 0 表示无持仓, 1 表示持仓
14
15 # 生成买入和卖出信号
16 trade[ 'Signal' ] = 0

```

```

16   for i in range(1, trade.shape[0]): # 从第二天开始, 因为第一天没有前一天的持仓状态
17       if trade['Growth_Perc'][i] > 0.02 and trade['Position'][i] == 0:
18           trade['Signal'][i-1] = 1
19           trade['Position'][i] = 1
20       elif trade['Growth_Perc'][i] < -0.005 and trade['Position'][i-1] == 1:
21           trade['Signal'][i-1] = -1
22           trade['Position'][i] = 0
23   else:
24       trade['Position'][i] = trade['Position'][i-1]
25
26 # 计算策略收益
27 trade['Strategy_Return'] = trade['Signal'].shift(1) * trade['Clsprc'].pct_change()
28
29 # 计算累积收益
30 trade['Cumulative_Returns'] = (1 + trade['Strategy_Return']).cumprod()
31
32 # 计算全线持有策略的累积收益
33 trade['Buy_and_Hold'] = (trade['Clsprc'] / trade['Clsprc'].iloc[0])

```

3.5.2 量化交易策略结果展示

之后我们利用 Python 中常用的 matplotlib 库绘制策略收益率曲线及全期持有的累积收益曲线：

代码 3-11 量化交易策略收益率曲线及全期持有的累积收益曲线绘制

```

1 # 绘制累积收益率图表
2 plt.figure(figsize=(12, 6))
3 plt.plot(trade['Cumulative_Returns'], label='Strategy Returns')
4 plt.plot(trade['Buy_and_Hold'], label='Hold Returns')
5 plt.xlabel('Date')
6 plt.ylabel('Cumulative Returns')
7 plt.title('Cumulative Returns Comparison')
8 plt.legend()
9 plt.show()

```

绘制出的图标如下图所示：

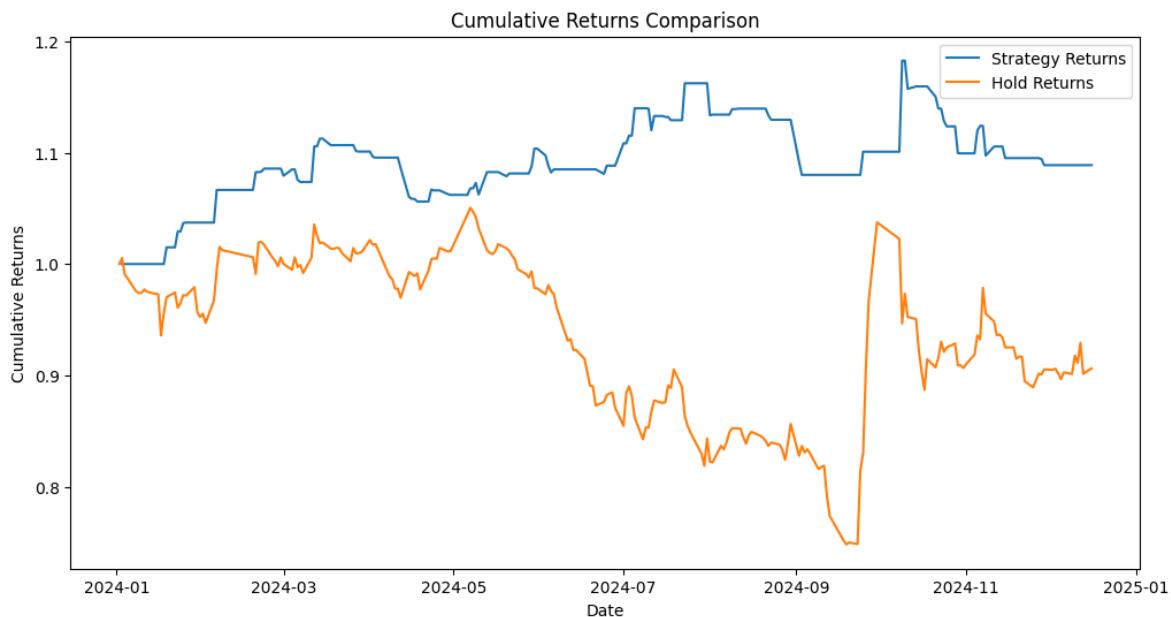


图 3-5 量化交易策略收益率曲线及全期持有的累积收益曲线

之后，我们画出了买卖点，具体代码及图片如下所示：

代码 3-12 量化交易策略买卖点绘制

```

1 # 绘制买入卖出点的价格图表
2 plt.figure(figsize=(12, 6))
3 plt.plot(trade['Clsprc'], label='Close Price', alpha=0.5)
4 buy_signals = trade.index[trade['Signal'] == 1]
5 sell_signals = trade.index[trade['Signal'] == -1]
6 plt.scatter(buy_signals, trade['Clsprc'][buy_signals], label='Buy Signal', marker='^',
7             color='g', alpha=1)
8 plt.scatter(sell_signals, trade['Clsprc'][sell_signals], label='Sell Signal', marker='v',
9             color='r', alpha=1)
10 plt.xlabel('Date')
11 plt.ylabel('Price')
12 plt.title('Buy/Sell Signals on Price Chart')
13 plt.legend()
14 plt.show()

```

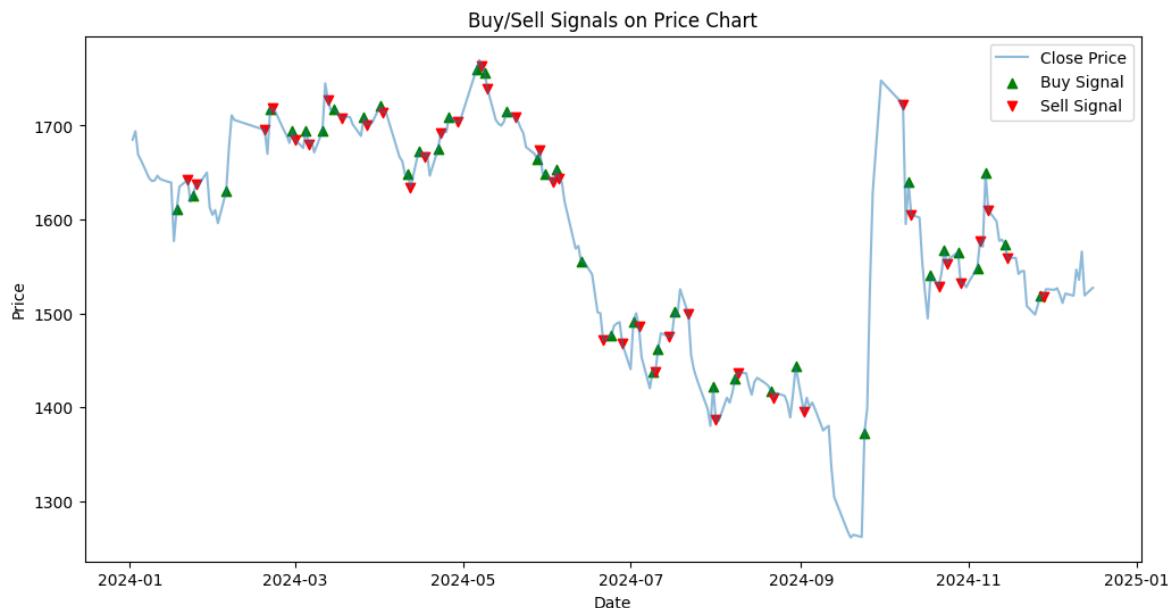


图 3-6 量化交易策略买卖点示意图

在其中，我们发现，交易策略似乎特别容易买高卖低，这主要原因是刚刚提到的对于股价预测的不准确性导致的，且预测结果往往具有一定的滞后性，因此，在实际的交易中，我们可以通过减少交易次数（通过增加交易的阈值）对这种交易风险进行回避。

具体操作完整的代码和回测示意图，如下面所示：

代码 3-13 调整后量化交易策略完整代码

```

1 import pandas as pd
2 import numpy as np
3 import matplotlib.pyplot as plt
4
5 # 计算预测增长百分比
6 trade['Growth_Perc'] = (trade['Predicted_Clsprc'] - trade['Predicted_Clsprc'].shift(1)) / trade['Predicted_Clsprc'].shift(1)
7
8 # 初始化持仓状态列
9 trade['Position'] = 0 # 0 表示无持仓, 1 表示持仓
10
11 # 生成买入和卖出信号
12 trade['Signal'] = 0
13 for i in range(1, trade.shape[0]): # 从第二天开始，因为第一天没有前一天的持仓状态
14     if trade['Growth_Perc'][i] > 0.02 and trade['Position'][i] == 0:
15         trade['Signal'][i-1] = 1
16         trade['Position'][i] = 1

```

```

17     elif trade[ 'Growth_Perc'][i] < -0.005 and trade[ 'Position'][ i-1] == 1:
18         trade[ 'Signal'][ i-1] = -1
19         trade[ 'Position'][ i] = 0
20     else:
21         trade[ 'Position'][ i] = trade[ 'Position'][ i-1]
22
23 # 计算策略收益
24 trade[ 'Strategy_Return'] = trade[ 'Signal']. shift (1) * trade[ 'Clspc']. pct_change()
25
26 # 计算累积收益
27 trade[ 'Cumulative_Returns'] = (1 + trade[ 'Strategy_Return']). cumprod()
28
29 # 计算全线持有策略的累积收益
30 trade[ 'Buy_and_Hold'] = (trade[ 'Clspc'] / trade[ 'Clspc']. iloc [0])
31
32 # 绘制累积收益率图表
33 plt . figure ( figsize =(12, 6))
34 plt . plot (trade[ 'Cumulative_Returns'], label='Strategy Returns')
35 plt . plot (trade[ 'Buy_and_Hold'], label='Hold Returns')
36 plt . xlabel ('Date')
37 plt . ylabel ('Cumulative Returns')
38 plt . title ('Cumulative Returns Comparison')
39 plt . legend()
40 plt . show()
41
42 # 绘制买入卖出点的价格图表
43 plt . figure ( figsize =(12, 6))
44 plt . plot (trade[ 'Clspc'], label='Close Price', alpha=0.5)
45 plt . plot (trade[ 'Predicted_Clspc'], label='Predicted Close Price', alpha=0.5)
46 buy_signals = trade .index[ trade[ 'Signal'] == 1]
47 sell_signals = trade .index[ trade[ 'Signal'] == -1]
48 plt . scatter (buy_signals, trade[ 'Clspc'][ buy_signals], label='Buy Signal', marker='^',
49 , color='g', alpha=1)
50 plt . scatter (sell_signals, trade[ 'Clspc'][ sell_signals], label='Sell Signal', marker='v',
51 , color='r', alpha=1)
52 plt . xlabel ('Date')
53 plt . ylabel ('Price')
54 plt . title ('Buy/Sell Signals on Price Chart')
55 plt . legend()
56 plt . show()

```

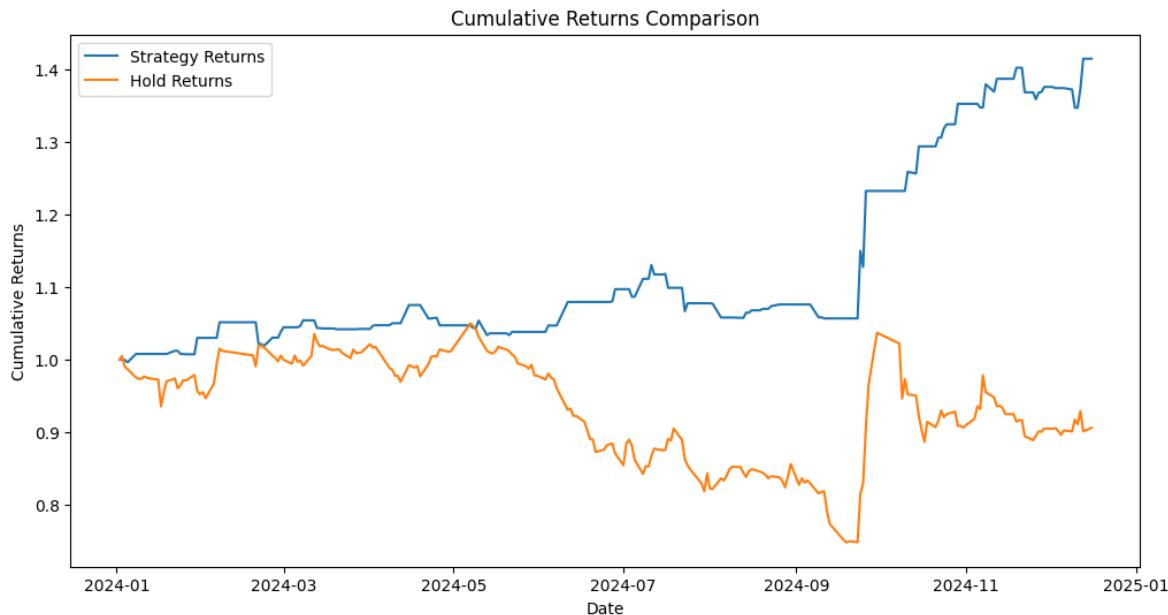


图 3-7 调整后量化交易策略收益率曲线

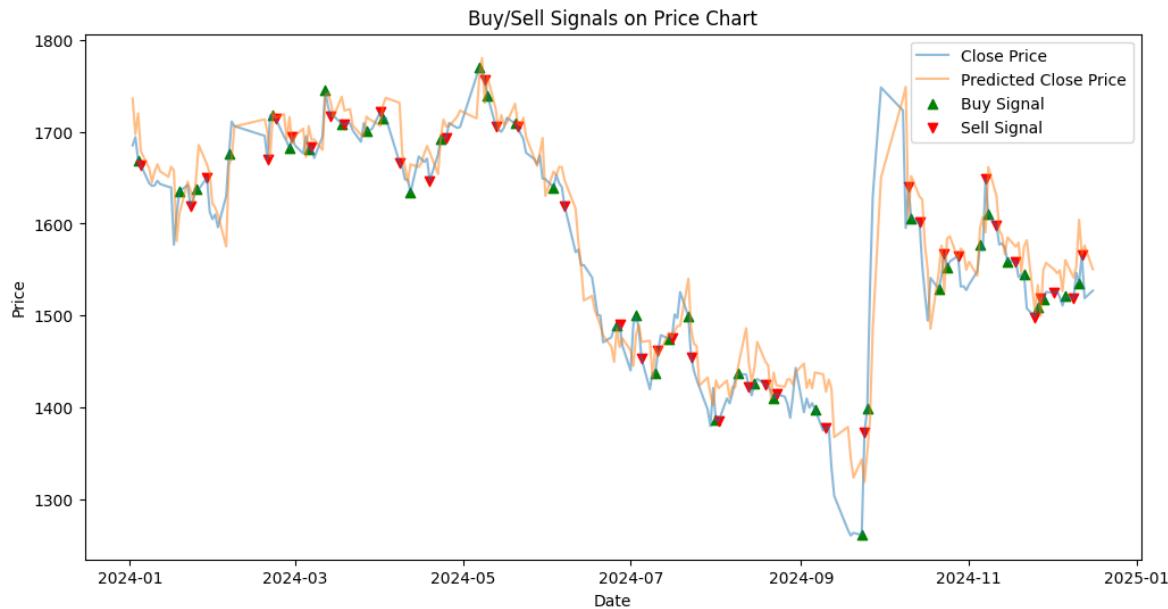


图 3-8 调整后量化交易策略买卖点示意图 1

上面图3-8中展示的是既有真实收盘价，又有预测收盘价的买卖点信息图。下面图3-9中展示的是只有真实收盘价的买卖点信息图，对于判断量化交易买卖点是否合适和有效能够给予更好的帮助。

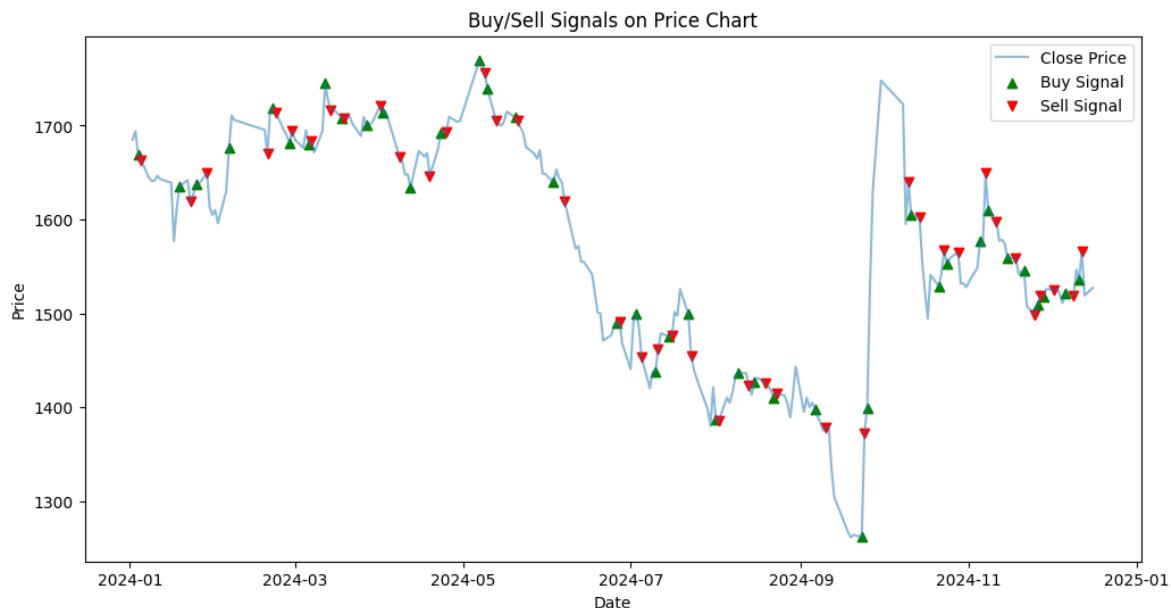


图 3-9 调整后量化交易策略买卖点示意图 2

3.6 量化交易结果评价

实现代码如代码3-14所示：

代码 3-14 量化交易结果评价

```

1 # 输出买卖点信息
2 trade_signals = pd.DataFrame()
3 trade_signals.index = trade.index
4 trade_signals = trade[trade['Signal'] != 0][['Clsprc', 'Signal']]
5 trade_signals['Action'] = trade_signals['Signal'].apply(lambda x: 'Buy' if x == 1
6 else 'Sell')
7 print("\n买卖点信息：")
8 print(trade_signals[['Action', 'Clsprc']])
9 trade_signals.to_csv('茅台交易信号.csv', index=True)
10
11 # 计算策略性能指标
12 total_return = trade['Cumulative_Returns'].iloc[-1] - 1
13 max_drawdown = ((trade['Cumulative_Returns'] / trade['Cumulative_Returns'].cummax() - 1).min())
14
15 print(f"\n策略最终收益: {total_return * 100:.2f}%")
16 print(f"策略最大回撤: {max_drawdown * 100:.2f}%")

```

最终交易信号，如下面表4-4和表3-5所示。

表 3-4 交易信号数据

Trddt	Clsprc	Signal	Action	Trddt	Clsprc	Signal	Action
2024/1/4	1669	1	Buy	2024/1/5	1663.36	-1	Sell
2024/1/19	1635	1	Buy	2024/1/23	1619	-1	Sell
2024/1/25	1638	1	Buy	2024/1/29	1650	-1	Sell
2024/2/6	1676	1	Buy	2024/2/20	1670	-1	Sell
2024/2/22	1718.38	1	Buy	2024/2/23	1714.09	-1	Sell
2024/2/28	1681.55	1	Buy	2024/2/29	1695	-1	Sell
2024/3/6	1680.55	1	Buy	2024/3/7	1683.63	-1	Sell
2024/3/12	1745	1	Buy	2024/3/14	1716.63	-1	Sell
2024/3/18	1708.26	1	Buy	2024/3/19	1708.02	-1	Sell
2024/3/27	1701	1	Buy	2024/4/1	1721.33	-1	Sell
2024/4/2	1713.99	1	Buy	2024/4/8	1666.66	-1	Sell
2024/4/12	1634.03	1	Buy	2024/4/19	1646.64	-1	Sell
2024/4/23	1692.28	1	Buy	2024/4/25	1693.04	-1	Sell
2024/5/7	1770	1	Buy	2024/5/9	1756	-1	Sell
2024/5/10	1738.98	1	Buy	2024/5/13	1706	-1	Sell
2024/5/20	1709	1	Buy	2024/5/21	1705	-1	Sell
2024/6/3	1639.39	1	Buy	2024/6/7	1619.18	-1	Sell
2024/6/26	1489.22	1	Buy	2024/6/27	1490.49	-1	Sell
2024/7/3	1500	1	Buy	2024/7/5	1453	-1	Sell
2024/7/10	1437.59	1	Buy	2024/7/11	1462.09	-1	Sell
2024/7/15	1474.9	1	Buy	2024/7/16	1476	-1	Sell
2024/7/22	1499	1	Buy	2024/7/23	1455	-1	Sell
2024/8/1	1386.16	1	Buy	2024/8/2	1385.45	-1	Sell
2024/8/9	1436.8	1	Buy	2024/8/13	1423.01	-1	Sell
2024/8/15	1426.89	1	Buy	2024/8/19	1425.44	-1	Sell
2024/8/22	1410.05	1	Buy	2024/8/23	1414.99	-1	Sell
2024/9/6	1398	1	Buy	2024/9/10	1378	-1	Sell
2024/9/23	1261.54	1	Buy	2024/9/24	1372.6	-1	Sell
2024/9/25	1399	1	Buy	2024/10/10	1640	-1	Sell
2024/10/11	1604.99	1	Buy	2024/10/14	1601.99	-1	Sell
2024/10/21	1528.8	1	Buy	2024/10/23	1567.5	-1	Sell

表 3-5 交易信号数据 (续)

Trddt	Clsprc	Signal	Action	Trddt	Clsprc	Signal	Action
2024/10/24	1552.2	1	Buy	2024/10/28	1565	-1	Sell
2024/11/5	1576.99	1	Buy	2024/11/7	1649.14	-1	Sell
2024/11/8	1609.97	1	Buy	2024/11/11	1598.01	-1	Sell
2024/11/15	1559	1	Buy	2024/11/18	1559	-1	Sell
2024/11/21	1545.13	1	Buy	2024/11/25	1498.57	-1	Sell
2024/11/26	1509	1	Buy	2024/11/27	1519.05	-1	Sell
2024/11/28	1518	1	Buy	2024/12/2	1525	-1	Sell
2024/12/6	1521.01	1	Buy	2024/12/9	1518.8	-1	Sell
2024/12/11	1535.6	1	Buy	2024/12/12	1565.8	-1	Sell

代码 3-15 量化交易结果评价体系 code

```

1 # 计算最终投资组合价值
2 final_portfolio_value = trade['Cumulative_Returns'].iloc[-1] * 100000 # 假设初始资金为 100,000
3 print("Final portfolio value: %.2f" % final_portfolio_value)
4
5 # 计算累积收益率
6 final_cumulative_return = (trade['Cumulative_Returns'].iloc[-1] - 1) * 100
7 print("Cumulative returns: %.2f %%" % final_cumulative_return)
8
9 # 计算夏普比率
10 risk_free_rate = 0.03 / 252 # 年化 3% 的无风险收益, 按每日计算
11 excess_returns = trade['Strategy_Return'] - risk_free_rate
12 sharpe_ratio = excess_returns.mean() / excess_returns.std() * np.sqrt(252) # 252 为一年交易天数
13 print("Sharpe ratio : %.2f" % sharpe_ratio)
14
15 # 计算最大回撤
16 rolling_max = trade['Cumulative_Returns'].cummax()
17 drawdown = (trade['Cumulative_Returns'] / rolling_max) - 1
18 max_drawdown = drawdown.min()
19 print("Max. drawdown: %.2f %%" % (max_drawdown * 100))
20
21 # 计算最长回撤持续时间
22 drawdown_duration = (drawdown != 0).astype(int).groupby((drawdown == 0).astype(int)).cumsum().cumsum()

```

```
23     longest_drawdown_duration = drawdown_duration.max()  
24     print("Longest drawdown duration: %d days" % longest_drawdown_duration)
```

假设我们投入资产 100000 元进行购买，仅考虑全仓和空仓的交易，最终量化交易指标：

- 策略最终投资组合价值 **Final portfolio value**: 141513.64;
- 策略最终收益 **Cumulative returns**: 41.51%;
- 策略夏普比率 **Sharpe ratio**: 2.03;
- 策略最大回撤 **Max. drawdown**: -6.51%;
- 策略最长回撤持续时间 **Longest drawdown duration**: 50 days;
- 策略交易次数 **Trades**: 40*2。

4 利用 Xgboost 进行量化交易策略搭建的评价

4.1 Xgboost 策略的可复制性

本文在上一部分中，我们选取的是贵州茅台股票作为研究对象，并基于该股票的历史数据进行了回归预测，进行了量化投资策略的回测。

下面我们将选取一些其他的股票使用本文原创的 Xgboost 量化策略进行策略的可复制性测试。

① 宁德时代

宁德时代是全球领先的动力电池制造商，在新能源电动车市场中占据重要地位。其技术优势和市场份额使其在行业中具有较强的竞争。优化后的 Xgboost 参数：

表 4-1 宁德时代最佳参数

参数	值
max_depth	3
learning_rate	0.04486725622782284
subsample	0.7228009241670307
colsample_bytree	0.9611165521785187
n_estimators	300
gamma	2.003634993236544
lambda	0.8667242682346259
alpha	0.678210913036579
min_child_weight	6
scale_pos_weight	3.73858271465928

预测效果展示：

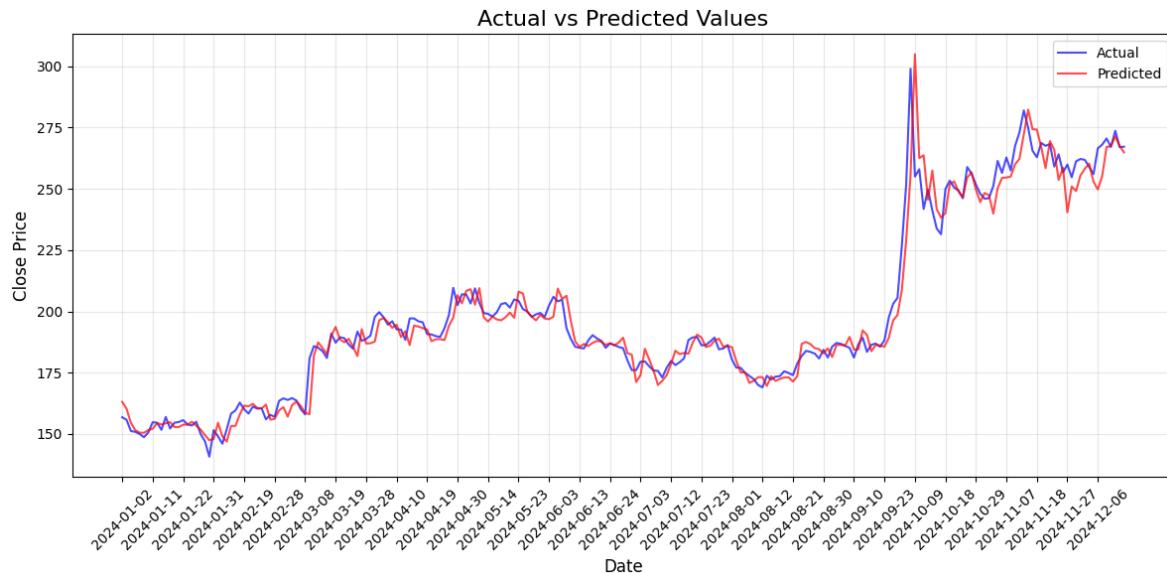


图 4-1 宁德时代模型股价预测结果

策略回测结果展示：

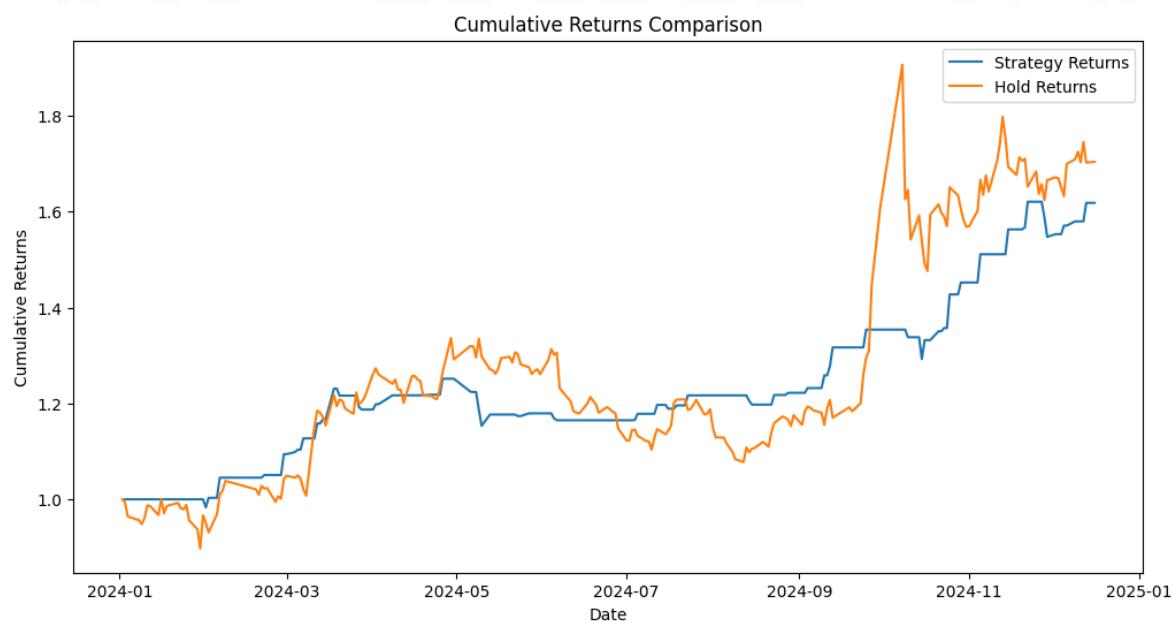


图 4-2 宁德时代策略回测结果

宁德时代股票量化交易点信息：

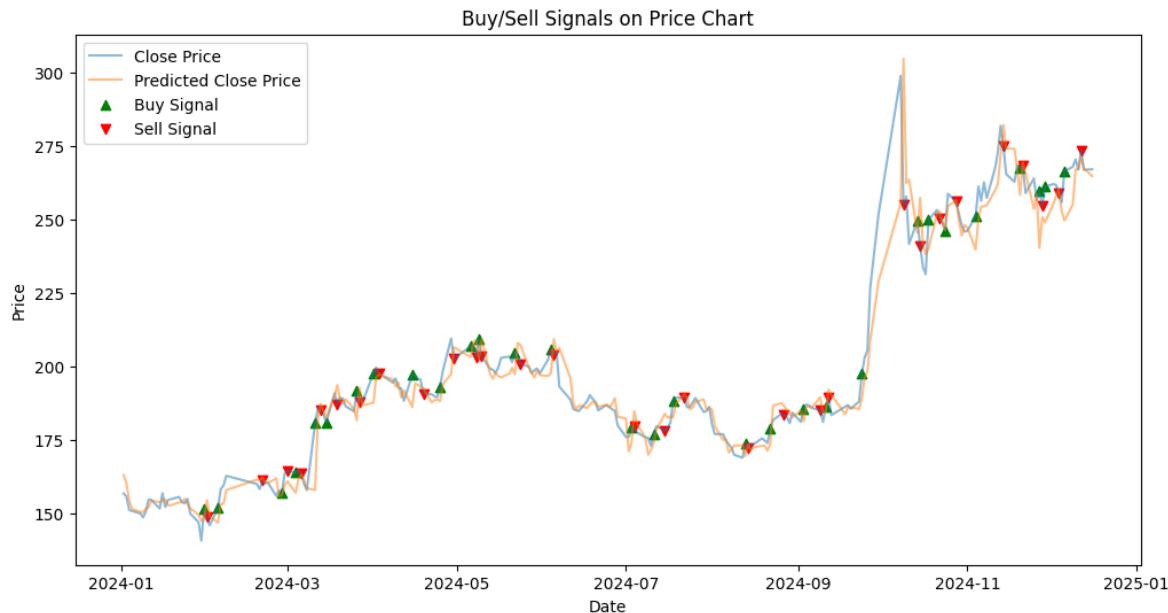


图 4-3 宁德时代策略交易点

代码 4-1 宁德时代策略回测 code

```

1 # 初始化持仓状态列
2 trade[ 'Position' ] = 0 # 0 表示无持仓, 1 表示持仓
3
4 # 生成买入和卖出信号
5 trade[ 'Signal' ] = 0
6 for i in range(1, trade.shape[0]): # 从第二天开始, 因为第一天没有前一天的持仓状态
7     if trade[ 'Growth_Perc' ][i] > 0.02 and trade[ 'Position' ][i-1] == 0:
8         trade[ 'Signal' ][i-1] = 1
9         trade[ 'Position' ][i] = 1
10    elif trade[ 'Growth_Perc' ][i] < -0.005 and trade[ 'Position' ][i-1] == 1:
11        trade[ 'Signal' ][i-1] = -1
12        trade[ 'Position' ][i] = 0
13    else:
14        trade[ 'Position' ][i] = trade[ 'Position' ][i-1]
15
16 # 计算策略收益
17 trade[ 'Strategy_Return' ] = trade[ 'Signal' ].shift(1) * trade[ 'Clsprc' ].pct_change()
18 # 计算累积收益
19 trade[ 'Cumulative_Returns' ] = (1 + trade[ 'Strategy_Return' ]).cumprod()
20 # 计算全线持有策略的累积收益
21 trade[ 'Buy_and_Hold' ] = (trade[ 'Clsprc' ] / trade[ 'Clsprc' ].iloc[0])

```

宁德时代交易详细信息：

表 4-2 宁德时代交易信号数据

Trddt	Clsprc	Signal	Action	Trddt	Clsprc	Signal	Action
2024/1/31	151.59	1	Buy	2024/2/1	149.02	-1	Sell
2024/2/5	151.94	1	Buy	2024/2/21	161.2	-1	Sell
2024/2/28	157.01	1	Buy	2024/3/1	164.55	-1	Sell
2024/3/4	163.9	1	Buy	2024/3/6	163.64	-1	Sell
2024/3/11	180.85	1	Buy	2024/3/13	185.18	-1	Sell
2024/3/15	181	1	Buy	2024/3/19	187.25	-1	Sell
2024/3/26	191.8	1	Buy	2024/3/27	188	-1	Sell
2024/4/1	197.84	1	Buy	2024/4/3	197.65	-1	Sell
2024/4/15	197.14	1	Buy	2024/4/19	190.8	-1	Sell
2024/4/25	193.2	1	Buy	2024/4/30	202.6	-1	Sell
2024/5/6	206.98	1	Buy	2024/5/8	203.23	-1	Sell
2024/5/9	209.4	1	Buy	2024/5/10	203.5	-1	Sell
2024/5/22	204.89	1	Buy	2024/5/24	200.98	-1	Sell
2024/6/4	205.97	1	Buy	2024/6/5	204.12	-1	Sell
2024/7/3	179.5	1	Buy	2024/7/4	179.6	-1	Sell
2024/7/11	177.01	1	Buy	2024/7/15	178.12	-1	Sell
2024/7/18	188.35	1	Buy	2024/7/22	189.52	-1	Sell
2024/8/13	173.78	1	Buy	2024/8/14	172.21	-1	Sell
2024/8/22	178.8	1	Buy	2024/8/27	183.5	-1	Sell
2024/9/3	185.65	1	Buy	2024/9/9	185.15	-1	Sell
2024/9/11	186.56	1	Buy	2024/9/12	189.35	-1	Sell
2024/9/24	197.52	1	Buy	2024/10/9	255	-1	Sell
2024/10/14	249.72	1	Buy	2024/10/15	241.22	-1	Sell
2024/10/18	249.89	1	Buy	2024/10/22	250.55	-1	Sell
2024/10/24	246.16	1	Buy	2024/10/28	256.24	-1	Sell
2024/11/4	251.2	1	Buy	2024/11/14	275	-1	Sell
2024/11/20	267.55	1	Buy	2024/11/21	268.26	-1	Sell
2024/11/27	259.9	1	Buy	2024/11/28	254.7	-1	Sell
2024/11/29	261.24	1	Buy	2024/12/4	258.97	-1	Sell
2024/12/6	266.55	1	Buy	2024/12/12	273.71	-1	Sell

宁德时代量化交易策略结果：

- 策略最终投资组合价值 **Final portfolio value**: 161830.08;
- 策略最终收益 **Cumulative returns**: 61.83%;
- 策略夏普比率 **Sharpe ratio**: 2.99;
- 策略最大回撤 **Max. drawdown**: -7.85%;
- 策略最长回撤持续时间 **Longest drawdown duration**: 90 days;
- 策略交易次数 **Trades**: 30*2。

② 云南白药 云南白药是中国知名的中药品牌，拥有悠久的历史和广泛的市场认可度。其产品在国内外市场上都有很高的知名度和美誉度。优化后的 Xgboost 参数：

表 4-3 最佳参数

参数	值
max_depth	8
learning_rate	0.08707009011414196
subsample	0.8403690699535916
colsample_bytree	0.9664828544292193
n_estimators	500
gamma	3.5831080130445323
lambda	0.3360239283529898
alpha	0.23407621576252055
min_child_weight	8
scale_pos_weight	5.212478254434845

预测效果展示：

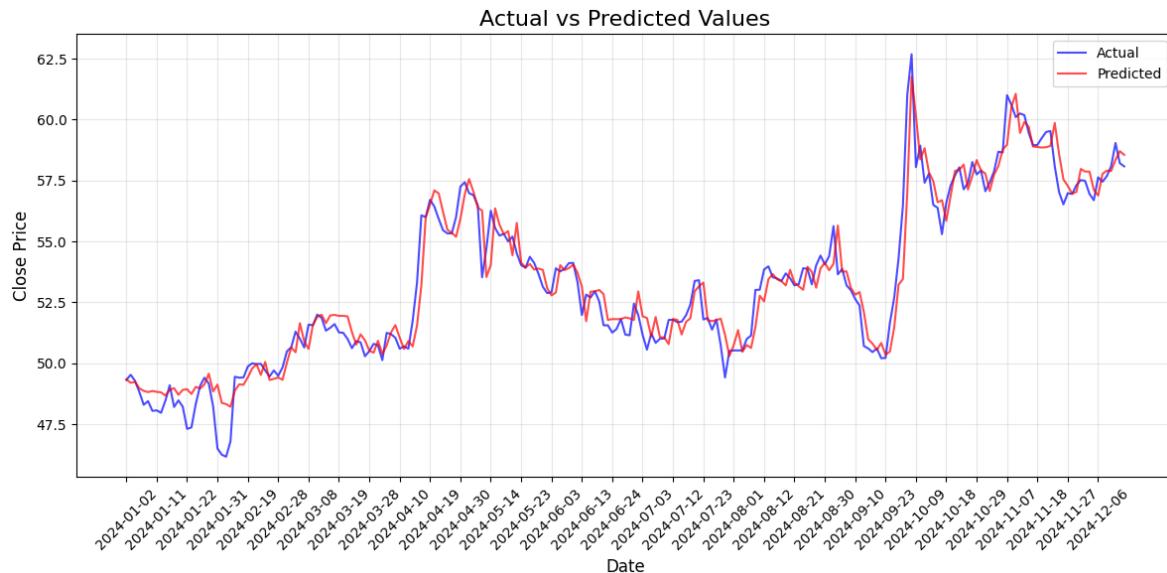


图 4-4 云南白药模型股价预测结果

策略回测结果展示：

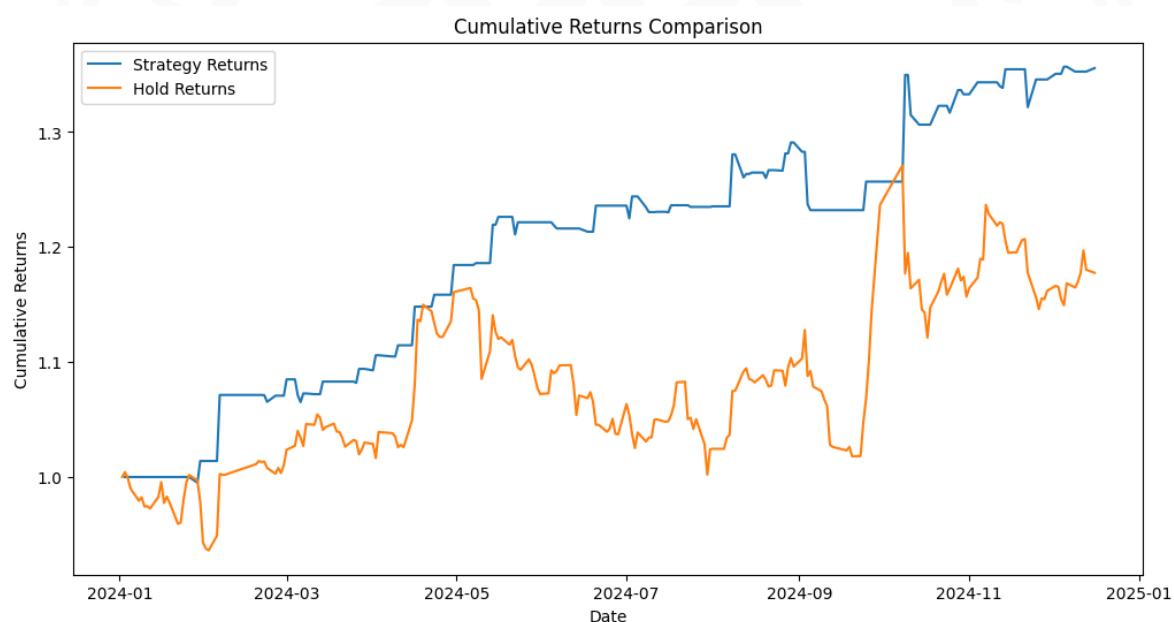


图 4-5 云南白药策略回测结果

云南白药股票量化交易点信息：

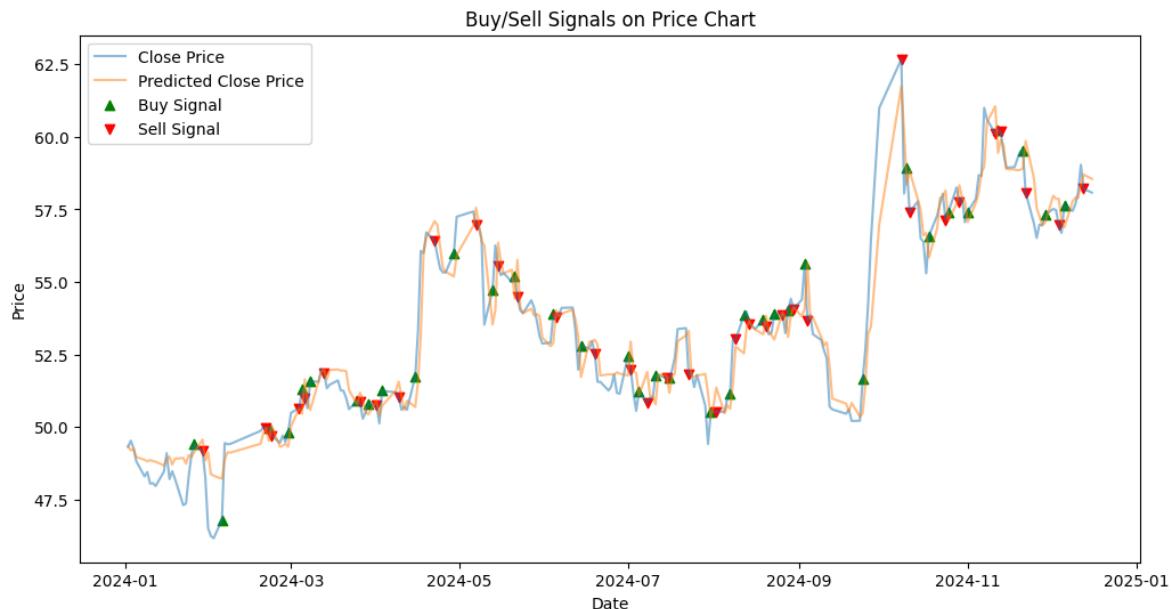


图 4-6 云南白药策略交易点

代码 4-2 云南白药策略回测 code

```

1 # 初始化持仓状态列
2 trade[ 'Position' ] = 0 # 0 表示无持仓, 1 表示持仓
3
4 # 生成买入和卖出信号
5 trade[ 'Signal' ] = 0
6 for i in range(1, trade.shape[0]): # 从第二天开始, 因为第一天没有前一天的持仓状
    态
7     if trade[ 'Growth_Perc' ][i] > 0.007 and trade[ 'Position' ][i-1] == 0:
8         trade[ 'Signal' ][i-1] = 1
9         trade[ 'Position' ][i] = 1
10    elif trade[ 'Growth_Perc' ][i] < -0.002 and trade[ 'Position' ][i-1] == 1:
11        trade[ 'Signal' ][i-1] = -1
12        trade[ 'Position' ][i] = 0
13    else:
14        trade[ 'Position' ][i] = trade[ 'Position' ][i-1]
15
16 # 计算策略收益
17 trade[ 'Strategy_Return' ] = trade[ 'Signal' ].shift(1) * trade[ 'Clspc' ].pct_change()
18 # 计算累积收益
19 trade[ 'Cumulative_Returns' ] = (1 + trade[ 'Strategy_Return' ]).cumprod()
20 # 计算全线持有策略的累积收益
21 trade[ 'Buy_and_Hold' ] = (trade[ 'Clspc' ] / trade[ 'Clspc' ].iloc[0])

```

云南白药交易点信息：

表 4-4 云南白药交易信号数据

Trddt	Clsprc	Signal	Action	Trddt	Clsprc	Signal	Action
2024/1/26	49.41	1	Buy	2024/1/29	49.17	-1	Sell
2024/2/5	46.8	1	Buy	2024/2/21	49.97	-1	Sell
2024/2/22	49.98	1	Buy	2024/2/23	49.71	-1	Sell
2024/2/29	49.83	1	Buy	2024/3/4	50.66	-1	Sell
2024/3/5	51.3	1	Buy	2024/3/6	51.01	-1	Sell
2024/3/8	51.59	1	Buy	2024/3/13	51.87	-1	Sell
2024/3/25	50.91	1	Buy	2024/3/26	50.86	-1	Sell
2024/3/29	50.8	1	Buy	2024/4/1	50.74	-1	Sell
2024/4/3	51.25	1	Buy	2024/4/9	51.05	-1	Sell
2024/4/15	51.75	1	Buy	2024/4/22	56.43	-1	Sell
2024/4/29	56	1	Buy	2024/5/7	56.98	-1	Sell
2024/5/13	54.72	1	Buy	2024/5/15	55.55	-1	Sell
2024/5/21	55.2	1	Buy	2024/5/22	54.51	-1	Sell
2024/6/4	53.9	1	Buy	2024/6/5	53.77	-1	Sell
2024/6/14	52.82	1	Buy	2024/6/19	52.54	-1	Sell
2024/7/1	52.45	1	Buy	2024/7/2	51.98	-1	Sell
2024/7/5	51.23	1	Buy	2024/7/8	50.84	-1	Sell
2024/7/11	51.78	1	Buy	2024/7/15	51.69	-1	Sell
2024/7/16	51.71	1	Buy	2024/7/23	51.8	-1	Sell
2024/7/31	50.51	1	Buy	2024/8/2	50.53	-1	Sell
2024/8/7	51.14	1	Buy	2024/8/9	53.02	-1	Sell
2024/8/12	53.85	1	Buy	2024/8/14	53.53	-1	Sell
2024/8/19	53.69	1	Buy	2024/8/20	53.49	-1	Sell
2024/8/23	53.9	1	Buy	2024/8/26	53.88	-1	Sell
2024/8/28	54.02	1	Buy	2024/8/30	54.06	-1	Sell
2024/9/3	55.63	1	Buy	2024/9/4	53.65	-1	Sell
2024/9/24	51.66	1	Buy	2024/10/8	62.68	-1	Sell
2024/10/10	58.93	1	Buy	2024/10/11	57.41	-1	Sell
2024/10/18	56.59	1	Buy	2024/10/24	57.14	-1	Sell
2024/10/25	57.4	1	Buy	2024/10/29	57.75	-1	Sell

表 4-5 云南白药交易信号数据（续）

Trddt	Clspred	Signal	Action	Trddt	Clspred	Signal	Action
2024/11/1	57.42	1	Buy	2024/11/11	60.1	-1	Sell
2024/11/12	60.25	1	Buy	2024/11/13	60.18	-1	Sell
2024/11/21	59.53	1	Buy	2024/11/22	58.08	-1	Sell
2024/11/29	57.31	1	Buy	2024/12/4	56.95	-1	Sell
2024/12/6	57.63	1	Buy	2024/12/13	58.21	-1	Sell

云南白药量化交易策略结果：

- 策略最终投资组合价值 Final portfolio value: 135565.63;
- 策略最终收益 Cumulative returns: 35.57%;
- 策略夏普比率 Sharpe ratio: 2.18;
- 策略最大回撤 Max. drawdown: -4.56%;
- 策略最长回撤持续时间 Longest drawdown duration: 24 days;
- 策略交易次数 Trades: 35*2。

③ 光启技术

光启技术在超材料和智能结构领域具有领先的技术优势。公司在超材料技术的研发和应用方面取得了显著进展，拥有大量专利和技术储备。在后面几只股票中，我们只给出交易点的图表以及最终的策略收益比较，其他的不再过多提到。

在利用 Xgboost 对光启技术进行量化交易策略搭建时，出现了相对比较大的偏差，这主要是由于光启技术在近阶段涨幅达到了之前前所未有的价格，导致超出了训练集中所有的数据，因此预测效果呈现一条线的持平。

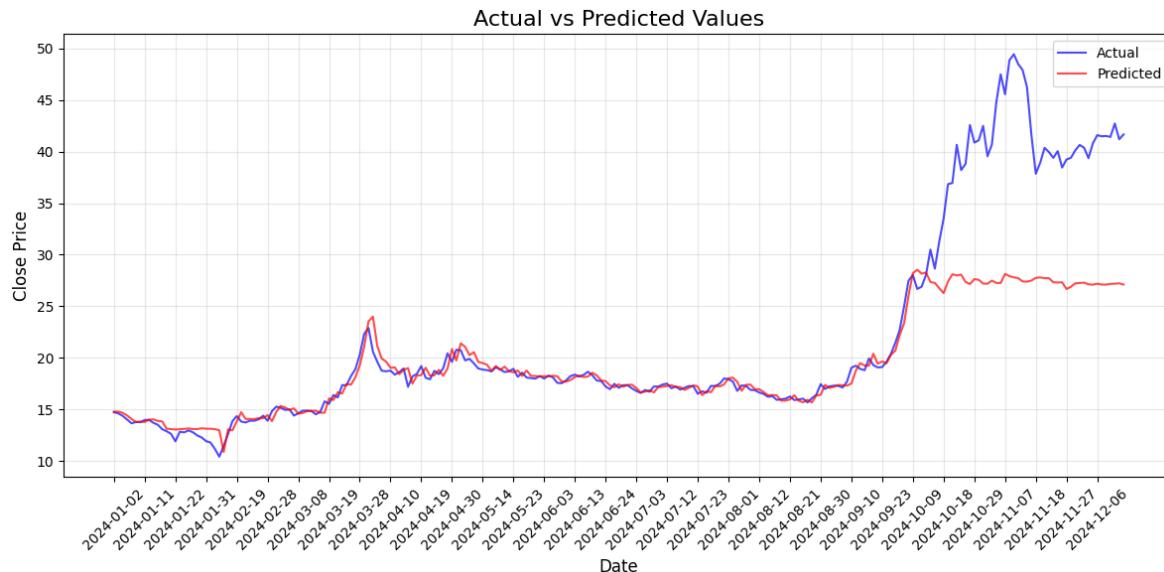


图 4-7 光启技术模型股价预测结果

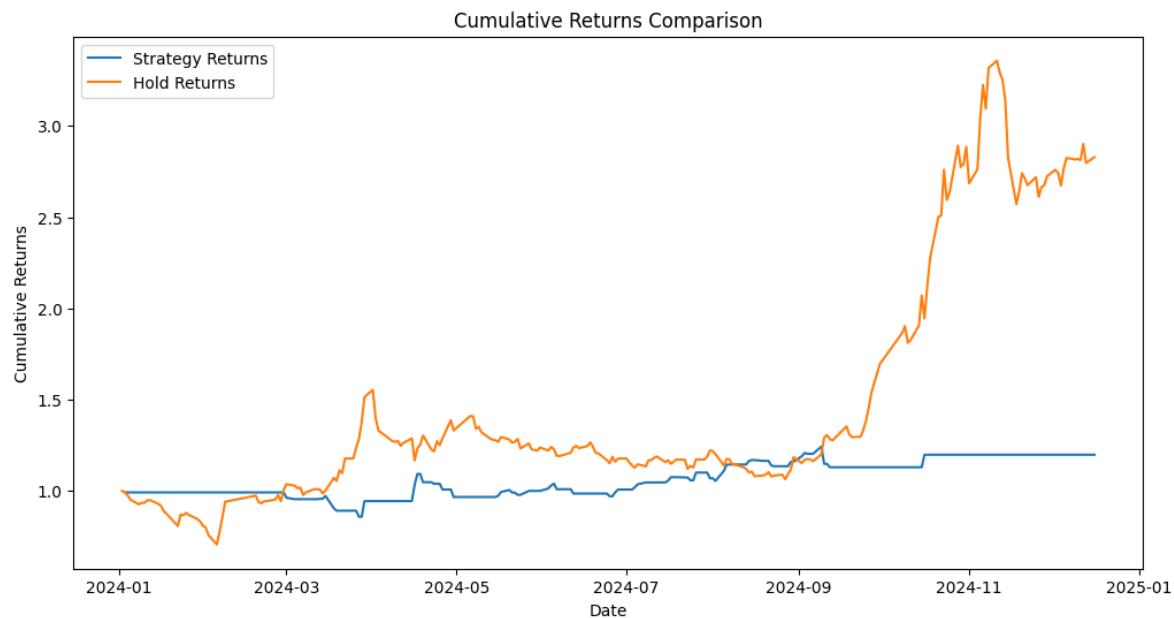


图 4-8 光启技术策略回测结果

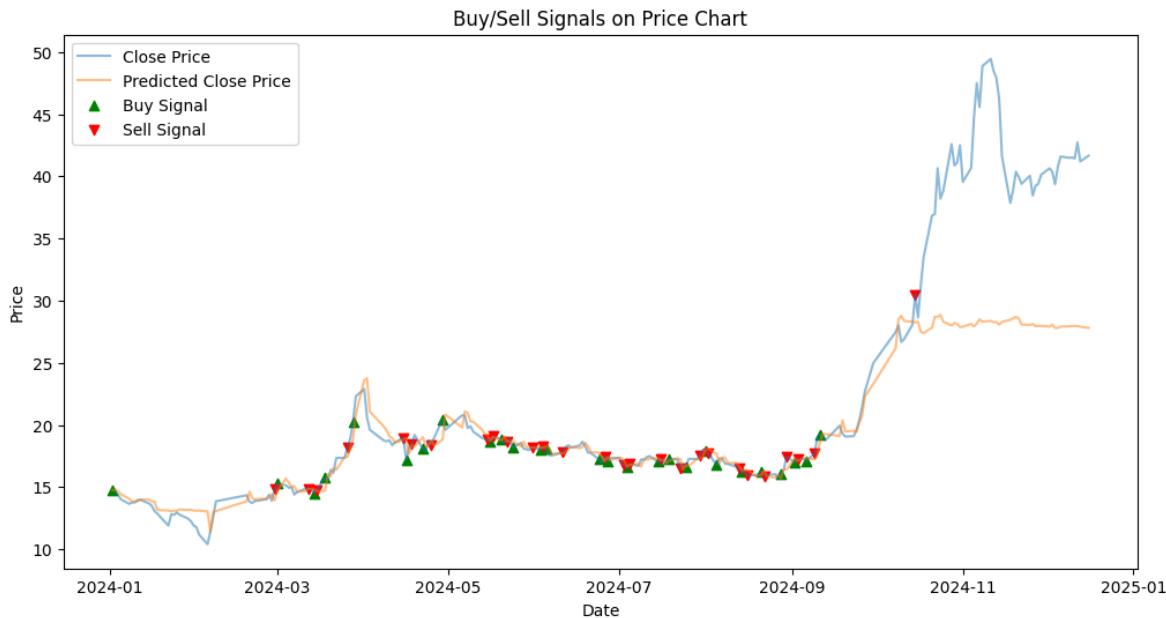


图 4-9 光启技术策略交易点

从该例子中可以看出，训练集对于最终模型的表现有着极大的影响；因此本文给出的一套可能的解决措施是：不再利用单一股票数据进行预测，选择行业的所有股票数据也许会更加合适。

光启技术量化交易策略结果：

- 策略最终投资组合价值 Final portfolio value: 119835.09;
- 策略最终收益 Cumulative returns: 19.84 %;
- 策略夏普比率 Sharpe ratio: 0.79;
- 策略最大回撤 Max. drawdown: -13.46 %;
- 策略最长回撤持续时间 Longest drawdown duration: 66 days;
- 策略交易次数 Trades: 27*2。

④ 紫天科技

紫天科技在超材料和智能结构领域具有领先的技术优势。公司在超材料技术的研发和应用方面取得了显著进展，拥有大量专利和技术储备。

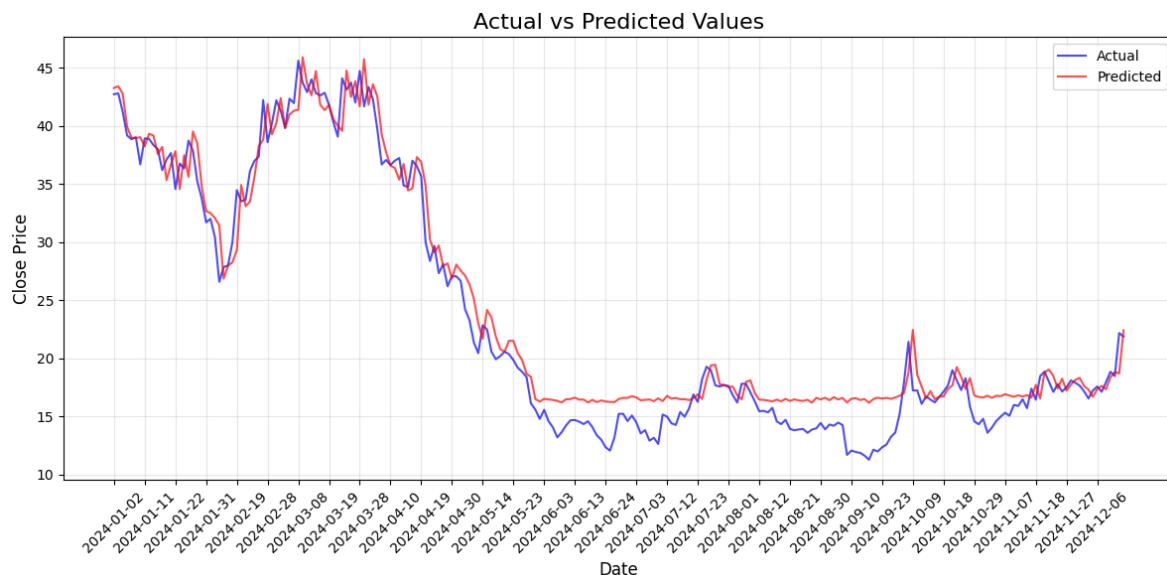


图 4-10 紫天科技模型股价预测结果

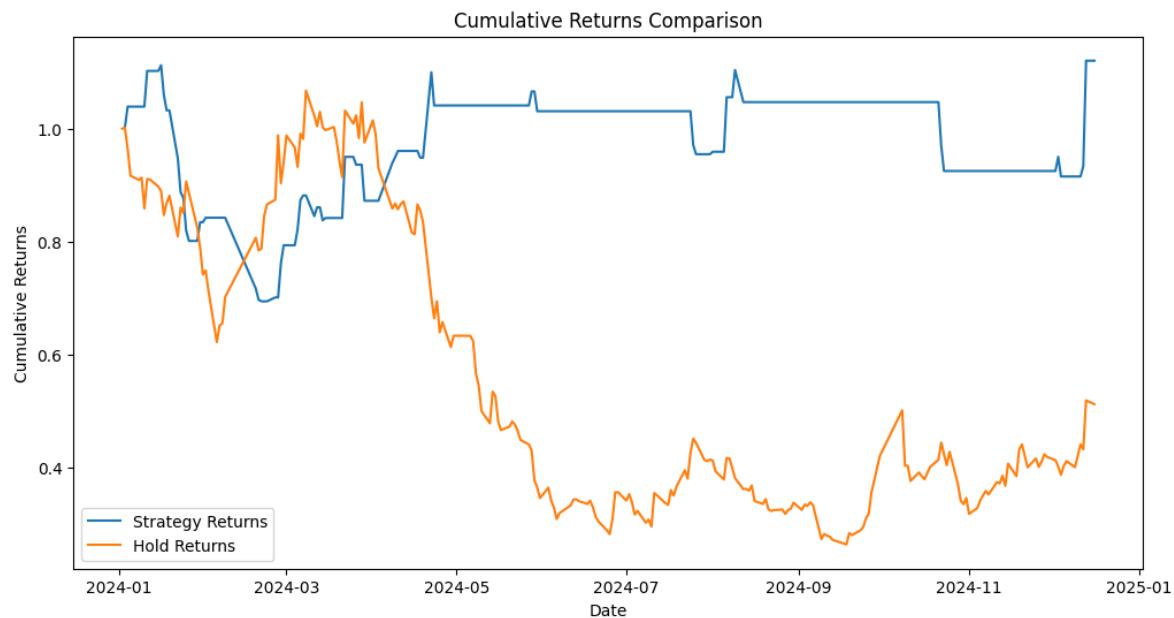


图 4-11 紫天科技策略回测结果

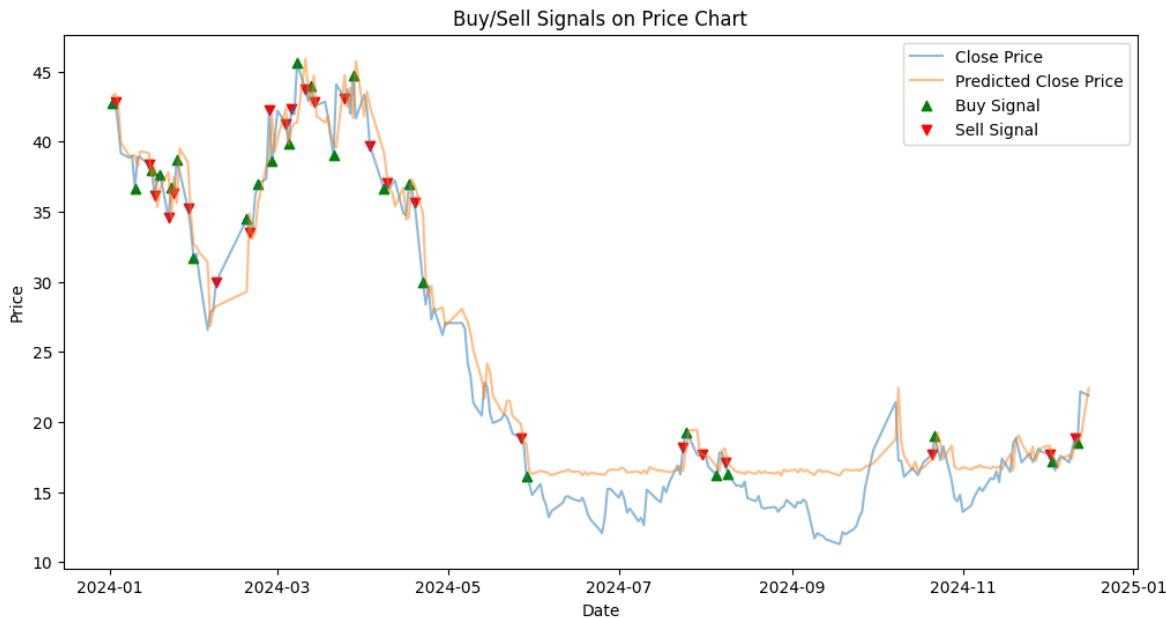


图 4-12 紫天科技策略交易点

这里同样也出现了近期跌破低价的情况，只需要提高训练集数据量即可解决这类问题。但我们的策略致使其并不影响我们对涨跌幅的判断。

紫天科技量化交易策略结果：

- 策略最终投资组合价值 Final portfolio value: 112031.46;
- 策略最终收益 Cumulative returns: 12.03 %;
- 策略夏普比率 Sharpe ratio: 0.43;
- 策略最大回撤 Max. drawdown: -37.56%;
- 策略最长回撤持续时间 Longest drawdown duration: 218 days;
- 策略交易次数 Trades: 24*2。

表现效果并不是特别的理想。

⑤ 京东方 A

京东方 A 是全球领先的显示面板制造商之一，拥有强大的市场份额和技术优势。其产品广泛应用于电视、手机、平板电脑等多个领域。

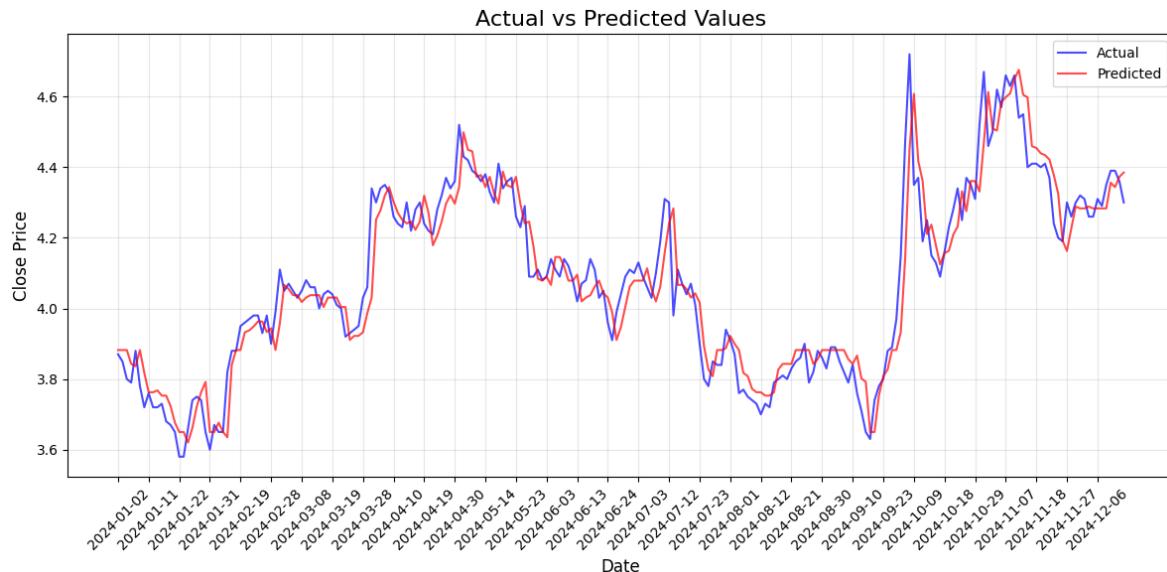


图 4-13 京东方 A 模型股价预测结果

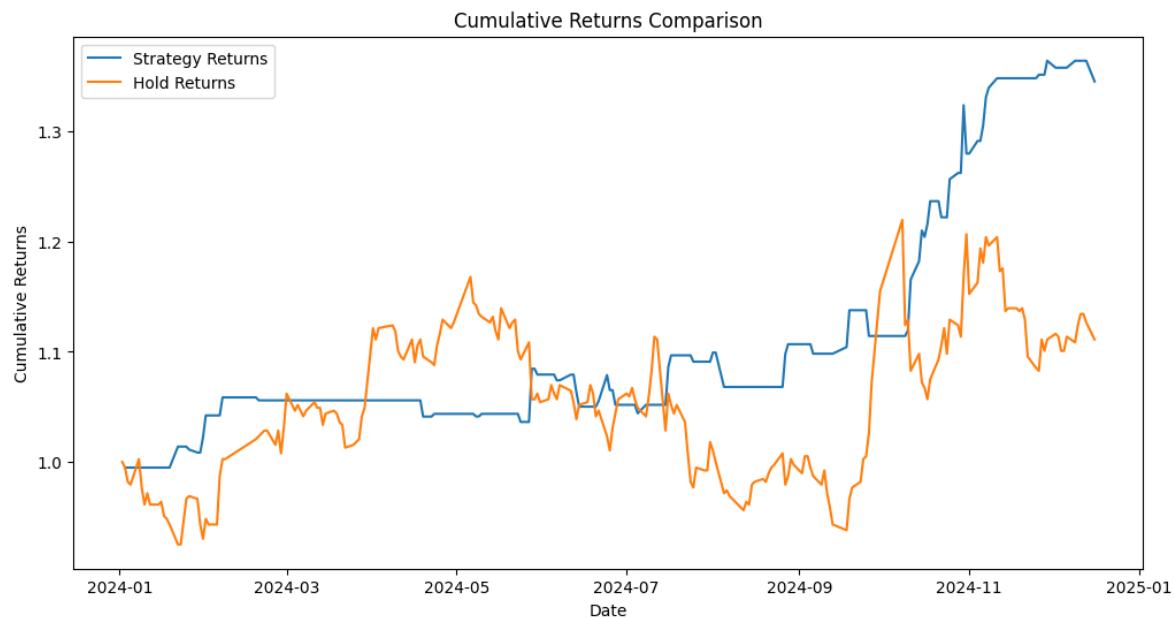


图 4-14 京东方 A 策略回测结果

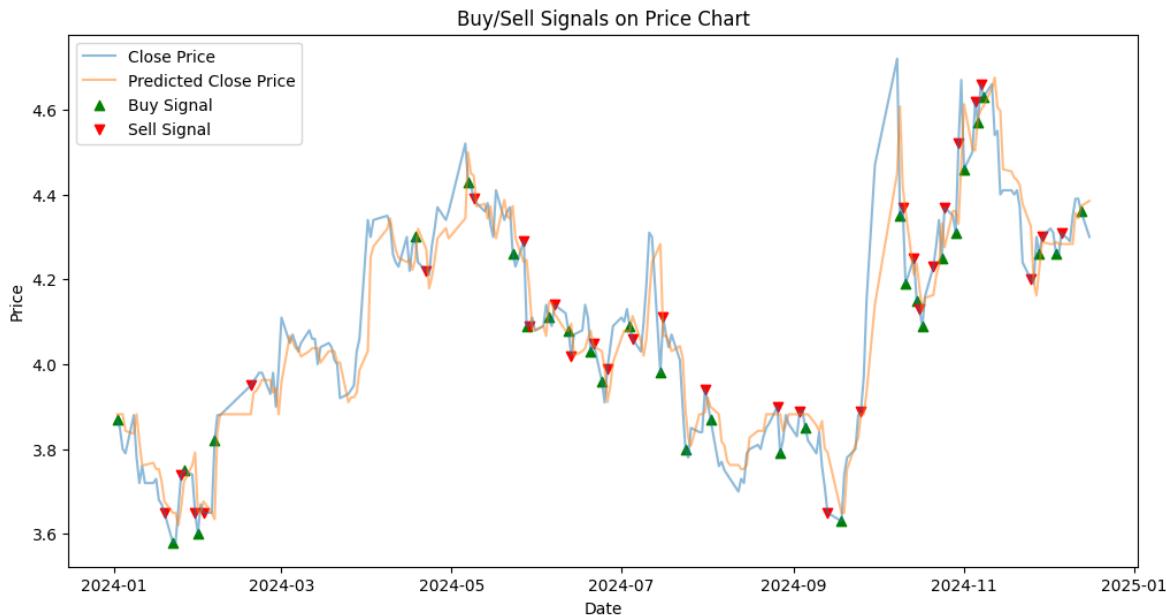


图 4-15 京东方 A 策略交易点

京东方 A 量化交易策略结果：

- 策略最终投资组合价值 Final portfolio value: 134540.95;
- 策略最终收益 Cumulative returns: 34.54 %;
- 策略夏普比率 Sharpe ratio: 2.21;
- 策略最大回撤 Max. drawdown: -3.73%;
- 策略最长回撤持续时间 Longest drawdown duration: 65 days;
- 策略交易次数 Trades: 31*2。

4.2 基于现有策略的拓展与改进

4.2.1 预测模型的改进

在本文中，我们通过 Xgboost 对股价进行了预测，其中用到了滞后期、均线等 106 个特征来进行预测，同时通过贝叶斯方法调参获得最小化的 RMSE（均方根误差）值。

在这里，我们提出几种可能比较好的特征办法，并对模型进行改进：

- 公司经营状况：公司的经营状况可以反映公司的健康状况，比如公司的营业收入、利润、现金流、资产负债率、市盈率、股息率等。上市公司按照市场要求应当定时发布季报、年报，这可能会影响最终的价格走势。
- 发放股息的频率和额度：同样的这也不是连续型的数据，但其可能会对股价有较大的影响。

- 公司重大事项变化：比如百通能源在 11 月份解禁限购股，这可能会导致股价一瞬间严重下跌，并形成新的市场支撑线和压力线。
 - 技术指标：本文不太关注于技术指标的构造，比如短期和长期移动平均线交叉时的点，对数据进行这一步处理可能能够有效地提升模型对股价的预测效果。
 - 其他：还有一些其他的市场限制，比如 $-10\%-10\%$ 的涨跌幅限制，模型无法很好的满足这一条件等。

由于本文篇幅的原因，此处不再过多的展开这部分内容。

4.2.2 交易策略的改进

对于交易策略，我们的交易信号简单的源自于价格的信息。若价格有上涨趋势，则买入股票；若价格有下跌趋势，则卖出股票。这会带来的问题有：

(1) 交易信号的不确定性：由于股票市场的不确定性，交易信号的准确性也不一定。如在本文的上面策略中就明显出现了策略买高卖低的情况，这有两方面的原因：一方面是模型预测结果的不准确性，其具有明显的滞后特征；另一方面是由于简单的去判断预测股价的上涨或下跌可能会使得交易信号实际偏移真实收盘价的上涨或下跌的方向。

对于第一点原因，比较好的方式在上文有所提到，增加更多的特征，另一种或许可行的方式是可以通过再训练模型并取平均值的方式来减少预测的误差；对于第二种原因，我们采取的方式是改变交易信号的判断方式：

代码 4-3 交易信号生成的策略改进

```
1 # 计算预测增长百分比
2 trade['Growth_Perc'] = (trade['Predicted_Clsprc'] - trade['Clsprc'].shift(1)) / trade
3 ['Clsprc'].shift(1)
4
5 # 初始化持仓状态列
6 trade['Position'] = 0 # 0 表示无持仓, 1 表示持仓
7
8 # 生成买入和卖出信号
9 trade['Signal'] = 0
10 for i in range(1, trade.shape[0]): # 从第二天开始, 因为第一天没有前一天的持仓状
11 态
12     if trade['Growth_Perc'][i] > 0.003 and trade['Position'][i-1] == 0:
13         trade['Signal'][i-1] = 1
14         trade['Position'][i] = 1
15     elif trade['Growth_Perc'][i] < 0 and trade['Position'][i-1] == 1:
```

```

14     trade[ 'Signal' ][ i-1 ] = -1
15     trade[ 'Position' ][ i ] = 0
16 else :
17     trade[ 'Position' ][ i ] = trade[ 'Position' ][ i-1 ]

```

与之前交易信号生成方式不同的是，这里我们不再使用预测的收盘价来计算涨跌幅；相反的，我们计算的是预测的未来收盘价与已知的最近的收盘价的增长（或减少）百分比。通过这种方式，结果发现，能够有效的降低交易次数，但由于模型预测能力的限制，导致预测无法与真实的收盘价非常贴合。

下面图4-16展示了调整交易信号后云南白药股票的回测结果。

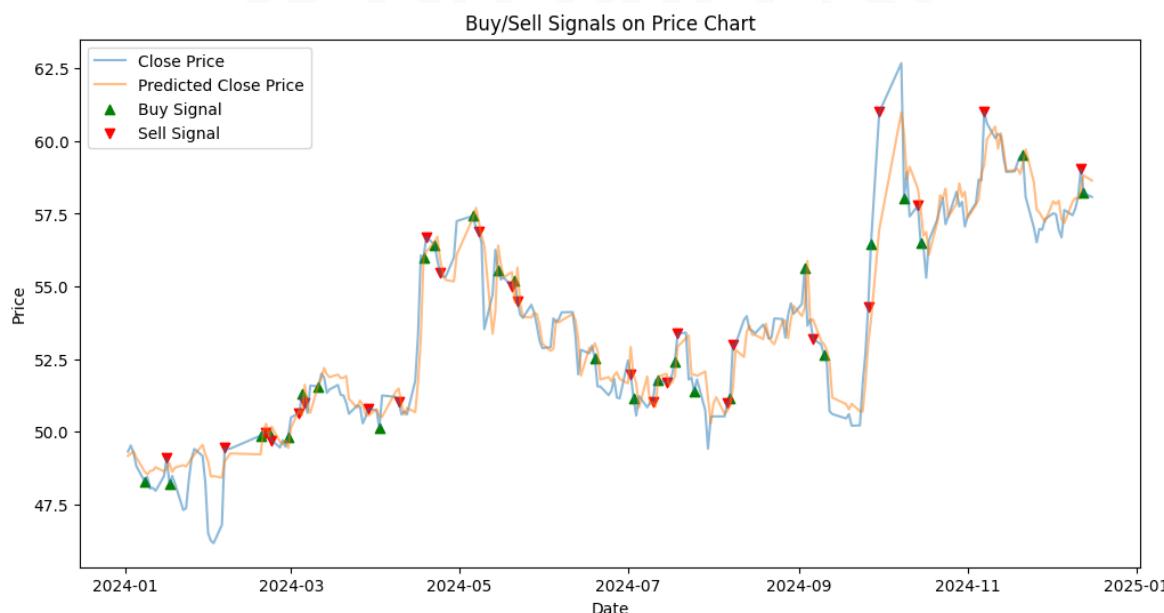


图 4-16 云南白药股票交易信号改进后的回测结果

从上图中可以看到，高买低卖的现象依然存在，并且导致最终策略的收益严重受损（主要是因为没有办法像之前一样更多的获取头寸的收益）。

4.2.3 基于 Xgboost 股价预测的投资组合策略

在之前，我们一直都局限于投资一支股票，在该部分，我们将融合投资组合的概念，通过对多个股票的预测，来构建一个投资组合策略。

由于在后半部分会详细介绍如何构建投资组合模型（见国际投资组合搭建（低频量化）部分），因此最简单合理分配资产的方式就是通过 Markowitz 投资组合理论，通过最小化风险来获得，金融领域将标准差或方差定义为资产组合的风险的测度。

另外一种方式，这里是读者个人的想法，灵感来源于神经网络中的 Softmax 激活函数。

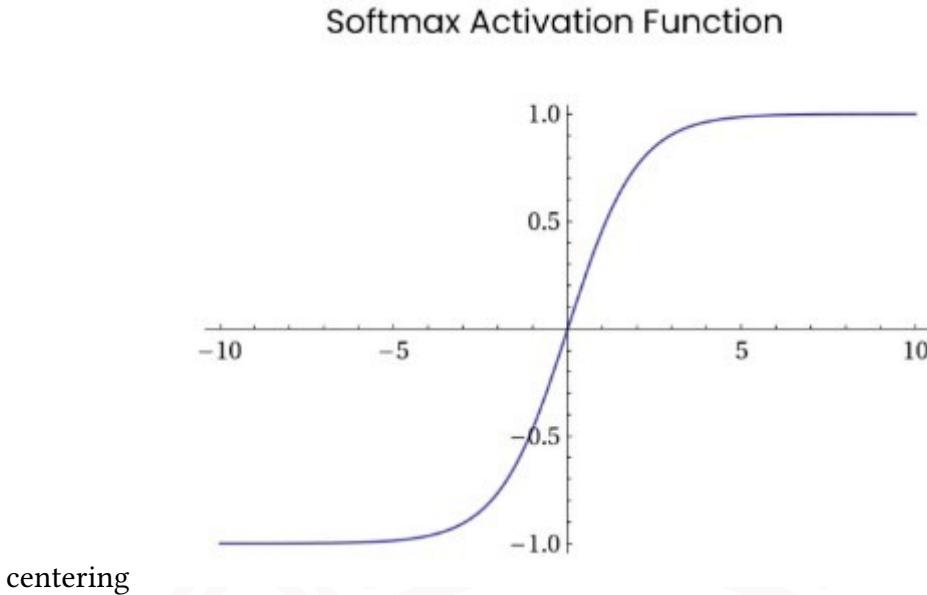


图 4-17 Softmax 激活函数图像

图4-17所示的为 Softmax 激活函数的图像，其数学表达式为：

$$\text{softmax}(x_i) = \frac{e^{x_i}}{\sum_{j=1}^K e^{x_j}} \quad (4-1)$$

Softmax 函数常在神经网络输出层充当激活函数，将输出层的值通过激活函数映射到 0-1 区间，将神经元输出构造成概率分布，用于多分类问题中，Softmax 激活函数映射值越大，则真实类别可能性越大

作者想到，其实我们利用 Xgboost 对股价进行预测，最终用作生成交易信号的依据，之前生成交易信号的代码如代码4-4所示：

代码 4-4 交易信号生成的策略改进

```

1 # 计算预测增长百分比
2 trade[ 'Growth_Perc' ] = ( trade[ 'Predicted_Clsprc' ] - trade[ 'Clsprc' ].shift(1) ) / trade
3 [ 'Clsprc' ].shift(1)
4
5 # 初始化持仓状态列
6 trade[ 'Position' ] = 0 # 0 表示无持仓，1 表示持仓
7
8 # 生成买入和卖出信号
9 trade[ 'Signal' ] = 0
10 for i in range(1, trade.shape[0]): # 从第二天开始，因为第一天没有前一天的持仓状
11 态
12     if trade[ 'Growth_Perc' ][i] > 0.003 and trade[ 'Position' ][i-1] == 0:
13         trade[ 'Signal' ][i-1] = 1
14         trade[ 'Position' ][i] = 1

```

```

13     elif trade['Growth_Perc'][i] < 0 and trade['Position'][i-1] == 1:
14         trade['Signal'][i-1] = -1
15         trade['Position'][i] = 0
16     else:
17         trade['Position'][i] = trade['Position'][i-1]

```

其中， $\text{trade}['\text{Growth_Perc}'] = (\text{trade}['\text{Predicted_Clspc}'] - \text{trade}['\text{Clspc}']).\text{shift}(1)) / \text{trade}['\text{Clspc}'].\text{shift}(1)$ ，DataFrame 列 'Growth_Perc' 表示预测的收盘价与已知的最近的收盘价的增长（或减少）百分比。其实本身就意味着我们对于未来该股票收益状况的判断，因此，我们就可以利用这个信息来构建一个投资组合策略，并通过传入 Softmax 激活函数，来计算出投资组合中各资产的配比及仓位。

下面是定义 Softmax 计算函数的代码：

代码 4-5 Softmax 计算函数

```

1 # Softmax 激活函数
2 def softmax(x):
3     e_x = np.exp(x - np.max(x)) # 防止溢出
4     return e_x / e_x.sum(axis=0)

```

通过这种方式，我们就可以确定各资产的投资权重，从而获得最终的投资组合收益。

下面我们给出实现方法，我们还是以上面提到的贵州茅台、宁德时代、云南白药三者作为投资标的来进行交易策略的回测，具体代码如下：

代码 4-6 基于 Xgboost 股价预测的投资组合策略数据清洗 code

```

1 import pandas as pd
2 import numpy as np
3 import matplotlib.pyplot as plt
4
5 df_maotai = pd.read_csv('guizhoumaotai_trade.csv')
6 df_ningde = pd.read_csv('ningdeshidai_trade.csv')
7 df_yunnan = pd.read_csv('yunnanbaiyao_trade.csv')
8
9 # 计算每个资产的增长百分比
10 df_maotai['Growth_Perc'] = (df_maotai['Predicted_Clspc'] - df_maotai['Clspc']).shift(1)) / df_maotai['Clspc'].shift(1)
11 df_ningde['Growth_Perc'] = (df_ningde['Predicted_Clspc'] - df_ningde['Clspc']).shift(1)) / df_ningde['Clspc'].shift(1)
12 df_yunnan['Growth_Perc'] = (df_yunnan['Predicted_Clspc'] - df_yunnan['Clspc']).shift(1)) / df_yunnan['Clspc'].shift(1)

```

```

13
14 # 填充 NaN 值
15 df_maotai['Growth_Perc'].fillna(0, inplace=True)
16 df_ningde['Growth_Perc'].fillna(0, inplace=True)
17 df_yunnan['Growth_Perc'].fillna(0, inplace=True)
18
19 # 合并三个资产数据
20 df_portfolio = pd.DataFrame({
21     'Trddt': df_maotai.index,
22     'Moutai_Growth': df_maotai['Growth_Perc'],
23     'Ningde_Growth': df_ningde['Growth_Perc'],
24     'Yunnan_Growth': df_yunnan['Growth_Perc'],
25     'Moutai_Return': df_maotai['Clsprc'].pct_change(),
26     'Ningde_Return': df_ningde['Clsprc'].pct_change(),
27     'Yunnan_Return': df_yunnan['Clsprc'].pct_change()
28 }).set_index('Trddt')

```

接下来，我们需要对每一行，利用 Softmax 激活函数，可以得到三只股票的配比，具体实现代码如下所示：

代码 4-7 基于 Xgboost 股价预测的投资组合策略 Softmax 计算

```

1 # Softmax 激活函数
2 def softmax(x):
3     e_x = np.exp(x - np.max(x))
4     return e_x / e_x.sum()
5
6 # 动态调仓逻辑
7 weights = []
8 portfolio_returns = []
9
10 for i, row in df_portfolio.iterrows():
11     # 获取当日资产增长百分比
12     growth = row[['Moutai_Growth', 'Ningde_Growth', 'Yunnan_Growth']].values * 100
13     # 计算权重
14     weight = softmax(growth)
15     weights.append(weight)
16
17     # 计算当日组合收益
18     asset_returns = row[['Moutai_Return', 'Ningde_Return', 'Yunnan_Return']].values
19     portfolio_return = np.dot(weight, asset_returns)
20     portfolio_returns.append(portfolio_return)
21

```

```

22 # 将权重和组合收益存入 DataFrame
23 weights = np.array(weights)
24 df_portfolio [ 'Moutai_Weight' ] = weights [ :, 0 ]
25 df_portfolio [ 'Ningde_Weight' ] = weights [ :, 1 ]
26 df_portfolio [ 'Yunnan_Weight' ] = weights [ :, 2 ]
27 df_portfolio [ ' Portfolio_Return ' ] = portfolio_returns
28
29 # 计算组合的累积收益
30 df_portfolio [ 'Cumulative_Returns' ] = (1 + df_portfolio [ ' Portfolio_Return ' ]).cumprod()
31
32 # 输出每日权重和收益
33 print( df_portfolio [ [ 'Moutai_Weight', 'Ningde_Weight', 'Yunnan_Weight', 'Portfolio_Return' ] ].head())

```

值得注意的是,上面代码growth = row[['Moutai_Growth', 'Ningde_Growth', 'Yunnan_Growth']].values * 100, 最后的 (* 100) 将对最终权重起到至关重要的作用。

接下来, 画出投资组合的累计收益率曲线, 具体实现代码如下:

代码 4-8 基于 Xgboost 股价预测的投资组合策略累计收益率曲线作图 code

```

1 # 绘制累积收益曲线
2 plt . figure ( figsize =(12, 6))
3 plt . plot ( df_portfolio [ 'Cumulative_Returns' ], label =' Portfolio Cumulative Returns ', color ='b')
4 plt . xlabel ( 'Date')
5 plt . ylabel ( 'Cumulative Returns')
6 plt . title ( 'Dynamic Portfolio with Daily Rebalancing')
7 plt . legend ()
8 plt . grid ()
9 plt . show()
10
11 # 绘制每日收益曲线
12 plt . figure ( figsize =(12, 6))
13 plt . plot ( df_portfolio [ ' Portfolio_Return ' ], label =' Portfolio Daily Returns ', color ='b')
14 plt . xlabel ( 'Date')
15 plt . ylabel ( 'Daily Returns')
16 plt . title ( 'Daily Portfolio with Daily Rebalancing')
17 plt . legend ()
18 plt . grid ()
19 plt . show()

```

下面图4-18和图4-19分别代表着该投资组合, 累计收益率曲线和每日收益率曲线。

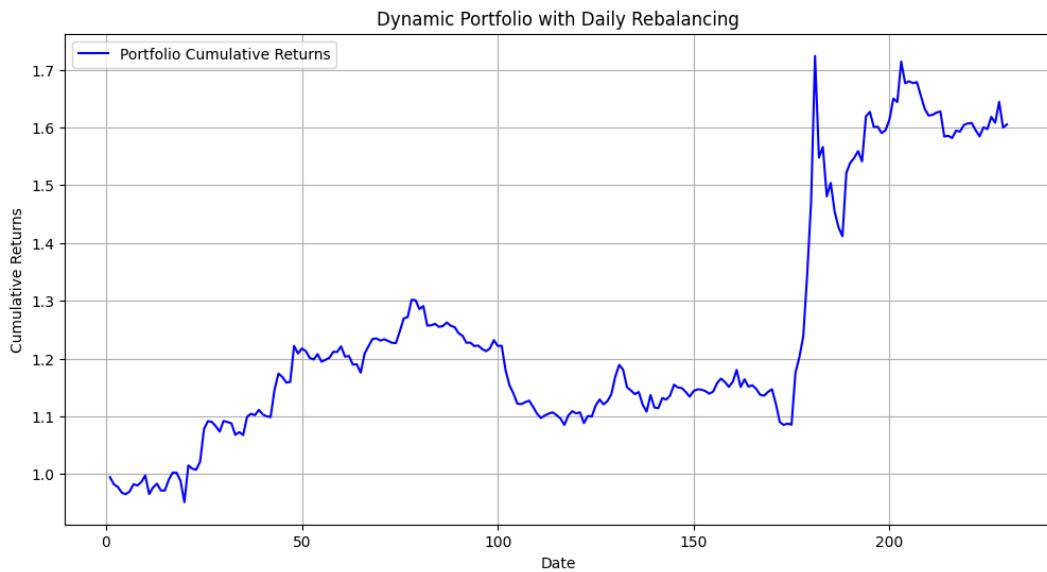


图 4-18 基于 Xgboost 股价预测的投资组合策略累计收益率曲线

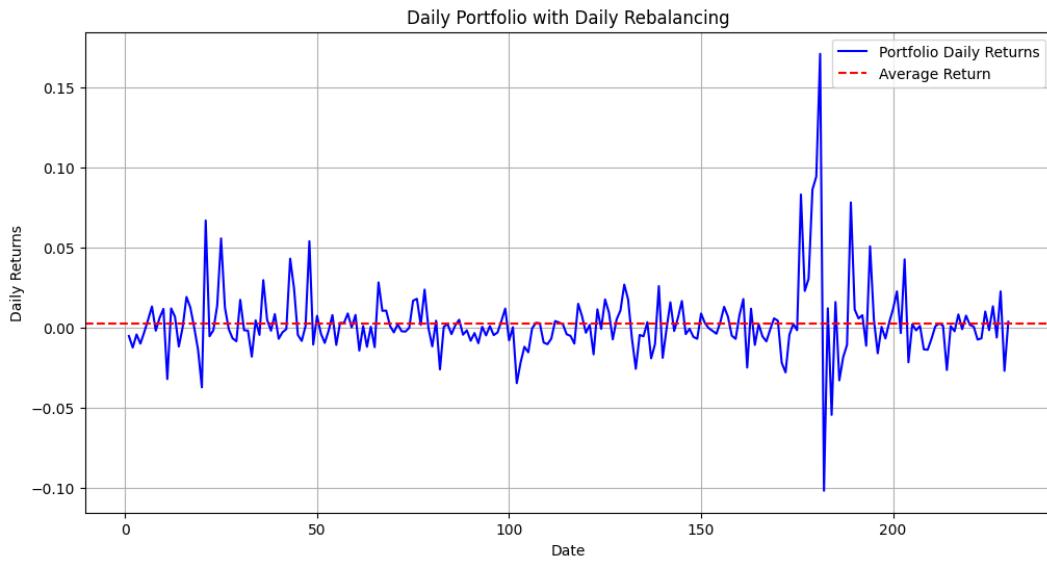


图 4-19 基于 Xgboost 股价预测的投资组合策略每日收益率曲线

最终构造的投资组合模型的收益状况为：

- 策略最终收益 **Cumulative returns**: 60.52%;
- 策略夏普比率 **Sharpe ratio**: 1.53;
- 策略最大回撤 **Max. drawdown**: -18.10%;
- 策略最长回撤持续时间 **Longest drawdown duration**: 100 days;

得出上面数据的代码如下：

代码 4-9 基于 Xgboost 股价预测的投资组合策略结果输出

```
1 # 计算收益率指标
2 def calculate_performance( df_portfolio ):
3     # 总收益率
4     total_return = df_portfolio [ 'Cumulative_Returns' ]. iloc [-1] - 1
5
6     # 年化收益率
7     trading_days = len( df_portfolio )
8     annualized_return = (1 + total_return ) ** (252 / trading_days) - 1
9
10    # 最大回撤和最长回撤天数
11    rolling_max = df_portfolio [ 'Cumulative_Returns' ]. cummax()
12    drawdown = df_portfolio[ 'Cumulative_Returns' ] / rolling_max - 1
13    max_drawdown = drawdown.min() # 最小值即最大回撤
14    drawdown_duration = (drawdown != 0). astype( int ). groupby((drawdown == 0). astype(
15        int )). cumsum(). cumsum()
16    max_drawdown_duration = drawdown_duration.max()
17
18    # 夏普比率（假设无风险收益率为年化 3%）
19    risk_free_rate = 0.03 / 252 # 无风险收益率的每日值
20    excess_daily_returns = df_portfolio [ 'Portfolio_Return' ] - risk_free_rate
21    sharpe_ratio = ( excess_daily_returns . mean() / excess_daily_returns . std () ) * np.
22    sqrt (252)
23
24    return {
25        'Total Return': total_return ,
26        'Annualized Return': annualized_return ,
27        'Max Drawdown': max_drawdown,
28        'Max Drawdown Duration': max_drawdown_duration,
29        'Sharpe Ratio' : sharpe_ratio
30    }
31
32    # 调用函数计算指标
33    performance = calculate_performance( df_portfolio )
34
35    # 打印收益率及回撤指标
36    print(f"总收益率: {performance['Total Return'] * 100:.2 f}%)")
37    print(f"年化收益率: {performance['Annualized Return'] * 100:.2 f}%)")
38    print(f"最大回撤: {performance['Max Drawdown'] * 100:.2f}%)")
39    print(f"最长回撤天数: {performance['Max Drawdown Duration']} 天")
40    print(f"夏普比率: {performance['Sharpe Ratio ']:.2 f}")
```

从最终结果中，我们可以看出，该策略最大回测较大，且投资组合的收益率波动

甚至有些超过单一购买某一资产时；造成该情况的主要原因在于利用 Softmax 激活函数获得投资组合权重的方法虽然可行（最终收益率依然可观），但并非完美。

在该部分的所有调仓数据，本文将在附录 B 中给出。

但需要注意的是，真实的量化交易可能没有办法实现这么精确的配比，其主要原因是购买股票以手（1 手 =100 股）为单位的限制；同时也存在没有办法以收盘价买入、以收盘价卖出这样的理想情况。

5 基于 xtquant 真实自动化交易实现

这里利用的是迅投公司出品的量化交易客户端软件 QMT，可以登录券商账号进行股票交易，但他暴露了基于 Python 的交易 API，可以执行程序化的交易。



图 5-1 迅投 QMT 策略交易系统

值得注意的是，一般证券账户对开通 QMT 系统有一定的基金要求，具体可以参阅文章（请点击此处）。首先需要安装 xtquant 库，下载链接可以点击此处。

代码 5-1 QMT 交易系统创建交易对象

```

1 import random
2 from xtquant.xttrader import XtQuantTrader
3
4 path = r'D:\国金证券QMT交易端\userdata_mini'
5 session_id = int(random.randint(100000, 999999))
6 xt_trader = XtQuantTrader(path, session_id)
  
```

创建 xt_trader 对象时需要输入两个参数：

- path：路径，即安装 QMT 软件的文件下的userdata_mini文件夹。
- session_id：回话 id，这里用六位随机数确定。

代码 5-2 QMT 交易系统登录

```

1 # 链接QMT客户端
2 xt_trader.start()
3 connect_result = xt_trader.connect()
4 print(connect_result)
5
  
```

```
6 if connect_result == 0:  
7     print('连接成功')  
8  
9 # 登录券商账号  
10 from xtquant.xttype import StockAccount  
11  
12 acc = StockAccount('xxxxx')  
13 subscribe_result = xt_trader.subscribe(acc)  
14 print(subscribe_result)
```

代码 5-3 QMT 交易系统下单命令

```
1 from xtquant import xtconstant  
2  
3 stock_code = '000429.SZ'  
4 order_id = xt_trader.order_stock(acc, stock_code, xtconstant.STOCK_BUY, 100,  
5 xtconstant.FIX_PRICE, 7.5)  
print(order_id)
```

其中，`xtconstant.STOCK_BUY`表单下单类型是买入，`xtconstant.FIX_PRICE`代表报价类型是限价，执行成功后，在 miniQMT 终端里，就直接可以看到委托记录，这就可以确认，我们的委托成功了。方法会返回订单变化，即下图中的订单编号。

代码 5-4 QMT 交易系统撤单命令

```
1 xt_trader.cancel_order_stock(acc, 1082130954)
```

订单编号用数字格式表示。

这里仅对实际中常用到的 PMT 交易系统的使用进行简单介绍，它可以将我们前面提到的量化交易策略应用到实际的投资操作中去。

除此之外，还有真格量化 (poboquant) 等诸多量化交易平台支持 Python API，从而可以实现真正意义上的量化交易实际应用。

6 国际投资组合搭建（低频量化交易）

依据作者个人理解，对于量化交易的全过程可以分为以下几步：

- 数据获取：获取股票、期货、外汇、期权等资产市场的行情数据，并进行数据清洗、处理等操作。
- 特征工程：对数据进行特征工程，包括技术指标、行业指标、市场风险等，以提取有用的信息。
- 建模预测：利用机器学习算法进行建模预测，包括线性回归、决策树、随机森林等各种方式。
- 交易策略：根据预测结果，制定交易策略，包括止盈止损、跟踪止损、仓位管理等。
- 实盘交易：将策略应用到实际的投资操作中，包括实时监控、风险控制、风险管理等。

在其中最为重要的主要就是特征工程的构建以及模型搭建的过程。特征的选取和效果，往往从根本上决定了模型对价格的解释能力，而模型的选择，模型的拟合程度、泛化能力也从此决定了策略的有效性和收益。

在这一部分，我们将简单介绍一下，在不适用计算机机器学习算法的基础上，如何通过构建投资组合来最小化风险，此部分依据 Markowitz 的投资组合理论^[2-3]。这里以一个国际投资组合的搭建为例子，介绍一下结合人主观看法的低频交易策略。

在该部分，与之前不同的是，之前更多的是对于股票价格的预测（或者对于股票表现概率的预测），之后判断是否应该购买股票，在该部分，我们将给出一个完整的国际投资组合，我们通过上市企业公布的年报、以及对于未来形式判断的方式，选择出合适的投资标的，之后通过线性规划（Linear Programming）的方法来确定投资组合的仓位。

6.1 为什么要选择国际投资？

首先，第一点需要考虑 A 股市场的限制，在中国，由于金融体系相对起步比较晚，金融市场发展并不是非常完善，在股市中，价格相对受到诸如国家队、投资者情绪等非理性因素的影响相对比较大，1992 年邓小平南方谈话、党的十四大确定中国的经济体制改革的目标是要建立社会主义市场经济体制。走中国特色的社会主义市场经济体制，既要抓住市场这只“无形的大手”，又要进行监管。这就会使得在中国的风险

资产市场中，市场并不一定完全有效^[4]（Fama 在 1970 年提出有效市场假说（Efficient Market Hypothesis, EMH）^[5]）。

同时，中国的股市有着严格的 T+1 交易限制，且每天有着 10% 的涨跌幅的限制，这将使得在 A 股市场中，对于价格的把控，可能不像一些资本主义制度的国家更具有市场特质，从而造成中国的量化交易策略必须具备中国的特色。

此外，在全球范围内投资可以有效分散单一市场的风险。不同国家和地区的经济周期、政治环境和市场状况各不相同，通过国际投资可以有效降低个别市场波动对整体投资组合的影响。

同时，由于当前人民币的国际地位不断提高，对于美元霸权地位产生了一定冲击，现在国际货币体系逐渐向着多元化发展^[6]，局面已经呈现出“一超一强多弱”的局面。美元居于全球第一位，占据优势；欧元是全球第二大货币，但与美元相比仍存在较大差距；人民币、英镑、日元等国际货币的总体地位依然有一些差距^[7]。通过国际投资策略的搭建，可以有效地降低本国货币贬值风险，提供对冲本国货币贬值的保护；当本国货币贬值时，持有其他国家资产可能可以相对减少损失，保持资产的实际购买力。

6.2 投资定位与经济概述

6.2.1 投资定位

与购买国内 A 股股票不同的是，在国际投资中，需要考虑国内投资者使用人民币投资外国资产的途径和形式。此处以国内投资者使用人民币投资外债（如美国国债）为例，介绍三种主要的投资机制。

(1) QDII (Qualified Domestic Institutional Investor, 合格境内机构投资者)

在人民币资本项目不可兑换、资本市场未开放条件下，在一国境内设立，经该国有关部门批准，有控制地，允许境内机构投资境外资本市场的股票、债券等有价证券投资业务的一项制度安排。QDII 机制自 2007 年实施以来，作为中国资本市场对外开放的临时措施，逐渐扩展至全球股票市场。

(2) QDLP (Qualified Domestic Limited Partnership, 合格境内有限合伙人)^[8]

以自有资金认购符合相关政策规定的试点基金的境内自然人、机构投资者或符合规定的其他投资者。2013 年，根据国家有关部署，上海率先启动 QDLP 试点，聚集了一批国际知名资产管理机构，第一批 6 家境外大型对冲基金公司共获得 3 亿美元 QDLP 额度，每家对冲基金公司分获 5000 万美元额度。自 2013 年在上海启动以来，已扩展至海南、重庆、青岛、江苏等地。政策支持下，QDLP 允许境内投资者通过特定基金管理人投资海外市场，促进了跨境资本流动和金融市场互联互通。2021 年，海南省、重庆市、广东省等地明确了合格境内有限合伙人的标准和要求，进一步规范了 QDLP 试点。2023 年，中国人民银行等部门继续推进 QDLP 试点，拓宽了境外投资渠道，为投

资者提供了更广阔的投资机会。

(3) MRF (Mutual Recognition of Funds, 互认基金)

在两个或多个国家之间相互认可并允许销售的基金产品。通过互认基金，投资者可以购买到其他国家的基金产品，而无需直接投资于那些国家的资本市场。

QDII 出台十余年以来，创造了丰富的外汇需求，一定程度上为我国减少了贸易顺差和资本项目盈余，考虑到其产品具有专业性强、投资积极主动的特点，投资比例理论上可达 100%，且门槛低至 1000 元人民币，适合广泛投资者，为内地投资者提供了全球资产配置的机会且有助于分散风险，我们在本文中希望申请成为 QDII 进行境外投资。

6.2.2 重点国家基本面分析

我们在此仅作简单介绍，且观点仅为作者个人看法，仅供读者参考。

6.2.2.1 中国

(1) 外需支撑作用明显增强，但难以弥补内需不足的拖累

2024 年，中国出口保持较快增长，是经济增长的重要支撑。但投资、消费等内需不足问题持续显现，供强需弱矛盾进一步凸显。前三季度，资本形成总额、最终消费、货物和服务净出口对经济增长的贡献率分别为 26.3%、49.9% 和 23.8%，分别较上年同期下降 5.1、33.6 个百分点，提高 38.7 个百分点^[9]（如下图6-1所示）。

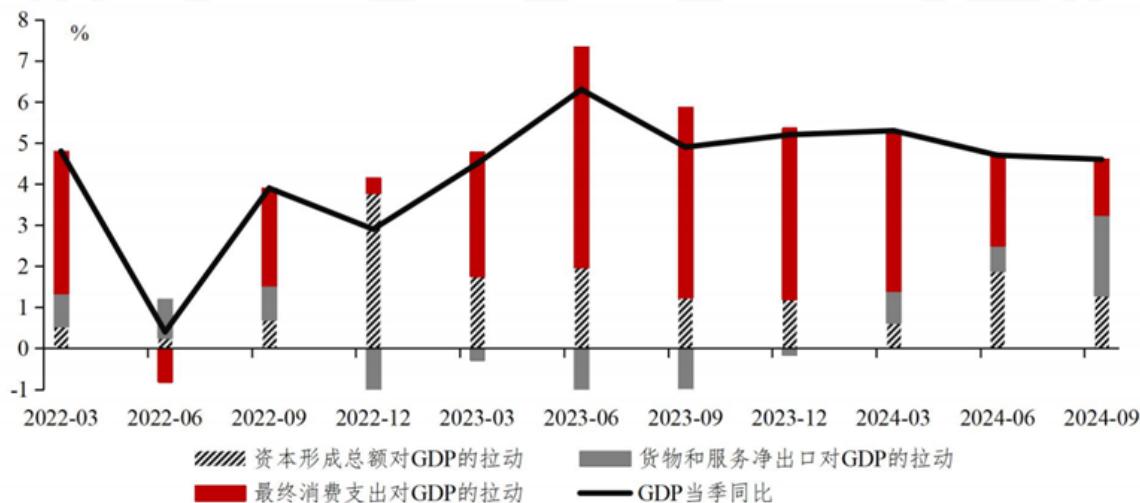


图 6-1 2022 年 3 月至 2024 年 9 月中国 GDP 变化及各科目对 GDP 的拉动情况

资料来源：Wind，中国银行研究院

(2) 新兴产业持续培育，但难以弥补房地产对经济增长的拖累

2024 年，工业对稳定经济增长发挥了重要支撑作用。1-10 月，规模以上工业增加值和高技术产业分别同比增长 5.8%、9.1%，较上年同期分别提高 1.7 和 7.2 个百分点；

前三季度，工业对经济增长的贡献率为 37.5%，较上年同期大幅提高 12.7 个百分点，高技术产业投资对全部固定资产投资增长的贡献率达到 27.1%。但与此同时，工业回升面临较多掣肘，稳增长压力持续抬升。一、二、三季度，工业增加值同比分别增长 6%、5.9% 和 5.1%，呈现逐季下行走势^[10]。

(3) 政策发力叠加融资需求回暖，货币社融增速有望稳步回升

2024 年 1-10 月社融累计新增 27.06 万亿元，同比少增 4.13 万亿元，出现明显下降，原因包括：一是经济主体融资意愿偏弱，二是监管部门调控思路发生转变。

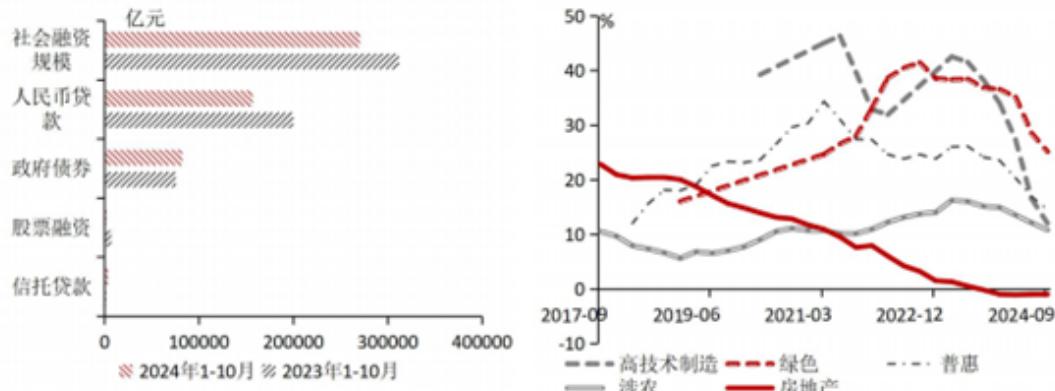


图 6-2 社会融资规模变化情况

资料来源：Wind，中国银行研究院

(4) 货币政策价格型调控机制进一步健全，整体利率水平继续下降

2024 年，面对实体经济曲折恢复、社会预期总体偏弱的情况，央行更加注重采取支持性政策取向。一方面，更加注重发挥基准利率价格型调控工具作用，淡化中期借贷便利利率的政策利率色彩，明确公开市场 7 天期回购操作利率是主要政策利率。当前 DR007 和 R007 之间的波动区间已明显收窄。从结果来看，整体利率水平继续下降。2024 年 7 月，7 天逆回购利率由 1.8% 降至 1.7%，1 年期 MLF 利率由 2.5% 降至 2.3%，9 月政策利率又大幅降息 0.2 个百分点，带动银行间同业拆借、LPR 等市场利率持续下行。另一方面，通过进一步降准等方式，向金融市场注入更大规模流动性，增强资金供给能力，引导市场利率下行。

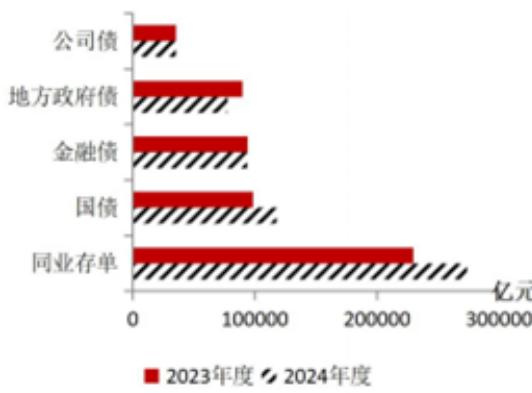


图 6-3 中国债券发行规模



图 6-4 中国国债收益率与利差变动

资料来源：Wind, 中国银行研究院

6.2.2.2 美国

由于篇幅限制问题，接下来几个国家仅作简单介绍，不再详细展开，读者可依据个人兴趣查阅相关资料。

(1) 美国制造业 PMI 继续萎缩，非制造业 PMI 继续扩张

2024 年 9 月美国供应管理协会编撰的 PMI 分别为 47.2，在今年 3 月短暂扩张后再次进入萎缩空间。非制造业为 54.9，连续三个月扩张，且 9 月扩张速度创下 2023 年 2 月以来最快，但波动比较剧烈。9 月，美国非制造业 PMI 与制造业 PMI 之间的差值升至 7.7 个点，创下 2019 年底以来的最大差距，凸显美国经济的严重分化。

(2) 美国个人收入与支出趋缓

(3) 房地产市场在下探过程中寻找支撑，企业启动补库存周期

疫情以来，低利率、财政纾困政策对于居民收入的补贴、以及疫情引发的居家办公需求是推动美国房地产投资和开工持续增长的主要原因。伴随着利率回升和房价上涨，2022 年美国房地产市场遭遇向下调整，但在 2023 年有所企稳。2024 年以来，美国房地产市场再次向下调整。

(4) 失业率小幅下降，劳动力市场仍具韧性

2024 年 9 月美国失业率为 4.1%，失业率小幅下降；新增非农就业为 25.4 万人；劳动参与率连续 3 个月持平；平均时薪同比增速为 4.0%，环比增速分别为 0.4%。此外，离职率、职位空缺率都有所下降。综合来看，美国劳动力市场紧张程度得到继续缓解。

(5) 金融市场情绪乐观

美债收益率曲线结束倒挂^①。2024 年 8 月，美国 2 年期和 10 年期国债收益率曲

^① 作者备注：收益率曲线倒挂是指短期债券（如 2 年期国债）的收益率高于长期债券（如 10 年期国债）的收益率。正常情况下，长期债券的收益率应高于短期债券，因为投资者要求较高的回报以补偿时间和风险。但是在某些情况下，这种关系会逆转，即收益率曲线出现倒挂，收益率曲线倒挂通常被视为经济衰退的预警信号。

线结束倒挂。至此，2022 年 7 月开始的美债历史上最长记录的倒挂阶段性结束。美元指数在出现较大幅度贬值后阶段性企稳。在美联储降息预期的推动下，3 季度美元指数明显下行。10 月以来在美国就业数据超预期、中东地缘政治动荡加剧、欧洲经济疲弱，美元指数明显反弹。美股续创历史新高，但震荡加剧。

6.2.2.3 保加利亚

(1) 宏观经济政策较为稳健

加利亚在中东欧地区以保加利亚为多党议会的政治体制和市场经济体制，是世贸组织和欧盟成员。近年来，保稳健的宏观经济政策而闻名。2023 年，保加利亚政局基本稳定，经济出现小幅增长，当年 GDP 为 1881 亿列弗，人均 GDP 为 18098 列弗。2024 年三季度，保加利亚名义 GDP 初值为 537 亿列弗，同比增长 10.5%，增速较上季度下跌 0.4 个百分点，比上年同期加快 4.0 个百分点。

(2) 财政系统运行良好、财政赤字率和债务水平较低，未出现经济崩盘风险

2023 年，保加利亚政府财政收入 646 亿列弗，约占 GDP 的 34.3%；支出 833 亿列弗，约占 GDP 的 44.3%；政府债务为 414 亿列弗，约占 GDP 的 22.0%。

(3) 经济属外向型，经济规模小，对外资依赖度高

截至 2023 年 12 月，保加利亚外汇储备为 820 亿列弗，年度 FDI 流量为 72 亿列弗。

(4) 失业率小幅增长

2023 年，保加利亚失业率为 4.34%，增长 0.17 个百分点。

6.2.2.4 澳大利亚

(1) 宏观经济数据表现温和

经济增长放缓：2024 年第三季度，澳大利亚 GDP 同比增长仅 0.8%，为自 1990 年代经济衰退以来的最低水平（除疫情影响外）。连续七季 GDP 增速下滑反映出经济结构性疲软。高通胀和利率压力导致家庭消费支出疲软，零售额增长大幅低于预期。

(2) 核心通胀压力持续

核心通胀高于目标水平，保持在 4% 左右，尽管略低 2023 年峰值，但仍远高于澳大利亚央行的 2-3% 通胀目标区间。高通胀对居民消费能力形成压制

(3) 财政政策维持宽松

2024 年澳大利亚政府在基建和绿色转型领域加大投资，以推动经济复苏。尽管预算赤字有所收窄，但政府债务占 GDP 比例仍在上升，达到历史高点（约 50%）。

(4) 货币政策稳健但面临挑战

货币政策保持高位：2024 年澳大利亚央行维持现金利率在 4.35%，为全球主要发达国家中较高水平。尽管核心通胀未显著下降，但市场预期可能于 2025 年初开始降

息。高利率环境下，企业融资成本增加，房贷违约率有所上升。

(5) 劳动力市场保持韧性

2024 年 9 月失业率为 3.8%，维持在相对低位。高利率导致企业招聘意愿下降，但就业市场仍较为稳健。

6.3 投资标的选择

对投资标的选择，我们的主要依据为上市公司年报、季报披露出的财务数据，以及社会舆论信息，融合作者对各个上市企业（或行业、或板块、或股票指数）的理解和判断得出。

为了实现各资产类别的风险对冲，我们不再局限于股票作为唯一的风资产种类，在该部分投资标的的选择中，我们选取国债、股票、基金和期货四大类资产作为投资目标。

这样做的目的也非常简单，主要如下所示：

- **多元化风险：**这四类资产在市场周期、经济状况及利率变化等因素下的表现各异，具有较低的相关性，可以有效分散风险，降低单一市场波动对整体投资组合的影响。
- **不同的风险回报特性：**每种资产类别的风险和回报特点不同。（国债通常低风险、低回报，适合稳定收入；股票风险较高，但回报潜力大；基金根据具体类型（股票型、债券型等）灵活配置，能够优化风险收益比；期货波动性较大，具有高风险，高回报的特点。）
- **宏观经济适应性：**股票、基金、国债和期货对不同的宏观经济环境有不同的敏感度，组合这几类资产，投资组合能够适应各种经济周期变化。

6.3.1 国债

(1) 美国国债

美国一年期国债收益率为较高水平的 4.28%，它被认为是全球最安全的投资工具之一，拥有较高的主权信用评级，且美元作为储备货币具有很强吸引力；美国国债市场是全球最大的债券市场，流动性极高，其收益稳定性和可预测性为短期投资者提供了确定性，是规避风险、获得稳定回报的理想选择。

(2) 澳大利亚国债

澳大利亚政府在国债市场上信息高度透明，买家可以轻松获取相关市场信息和政府部门的财政状况报告；且澳大利亚政府有充足的税收和收入来源，能够偿还到期债务，并能将利率控制在可持续水平的能力，有助于保持国债的价值和稳定性。

(3) 中国国债

中国政府支持发展债券市场，提供了较好的投资环境。且近年来，中国逐步开放债券市场，国际投资者可通过“债券通”等机制轻松投资中国国债；同时对于以人民币为主要货币的投资来说，投资中国国债无需承担汇率波动的风险。

(4) 保加利亚国债

保加利亚政府为外国投资者提供一系列保障和保护措施，包括税收豁免或减免、补助和津贴等激励措施，具有无可比拟的税收优势；同时保加利亚投资环境良好，配套服务市场潜力大，经济活力逐步增强，GDP 增势明显，具有庞大的发展潜力，有潜在投资红利。

(5) 组合优势

这四个国家涵盖了**发达经济体、新兴市场和小型经济体**，投资组合的多元化有助于分散风险，且某一国家或地区的经济衰退不太可能同时影响到其他国家或地区，从而降低系统性风险。

6.3.2 股票

(1) 标准普尔 500 指数

标普 500 指数涵盖美国 500 家大型企业，涉及多个行业，如金融、消费品、能源、科技等，提供多元化投资，降低了单一行业波动风险。且作为全球最受欢迎的股票指数之一，其具有极高的市场流动性。2024 年截至 12 月 3 日，标普 500 指数上涨 26.84%，高盛和摩根士丹利预测，2025 年底标普 500 指数可能触及 6500 点，美银预测 2025 年底标普 500 指数有望达到 6666 点，瑞银和德银更是给出了 7000 点的目标点位。

(2) MSCI 新兴市场指数

MSCI 新兴市场指数涵盖了全球 24 个新兴市场国家的股市，随着中国经济复苏，特别是消费升级和科技创新政策的推动，印度的快速增长以及全球需求复苏和大宗商品价格的稳定，新兴市场的能源和资源出口国（如巴西和南非）获增长动力，驱动 MSCI 新兴市场指数增长。

(3) 欧洲斯托克 50 指数

随着欧洲央行的货币政策趋于稳定和欧洲各国经济复苏，欧洲股市，特别是斯托克 50 指数中的成分股，能够从全球需求回升中受益，尤其是消费品、金融和工业领域。欧洲斯托克 50 指数包括了许多全球领先的大型企业，如德国的西门子、法国的 L'Oréal 等，这些公司在全球化竞争中具有强大的盈利能力和市场地位。

(5) 组合优势

- 涵盖不同地区和市场的优质股票，提供了强大的**地域多元化优势**，减少单一市场波动带来的风险，并提高组合的稳定性。

- 这些指数的成分股包含了全球领先的企业，有助于实现中长期的资本增值。

6.3.3 基金

(1) 中欧医疗健康混合 A

混合型基金，股票持仓集中，专注于投资医疗健康行业等与人们生命健康和生活质量相关的行业，在老龄化社会、全球健康问题的背景下，医疗健康行业的需求日益增长，基金在行业发展趋势上具有前瞻性。

(2) 线上零售 ETF (Amplify Online Retail ETF)

IBUY 专注于在线零售行业，投资于全球在电子商务和移动商务领域具有显著收入的公司。随着全球消费者购物习惯的数字化转型，在线零售市场持续增长，投资者可得到其中潜在的收益机会；且 IBUY 作为被动管理的基金，管理费用相对较低，有助于提高净回报。

(3) 富达云计算 ETF (SKYY)

SKYY 专注于云计算领域，投资于提供云计算服务的公司，包括基础设施、平台和软件即服务（SaaS）等多个领域。随着全球数字化转型的加速，云计算行业呈现出强劲的增长趋势；同时 SKYY 涵盖大型科技企业和新兴公司，可降低单一投资的风险；另外作为 ETF，SKYY 具有高流动性。

(4) 组合优势

- 覆盖了医疗健康、在线零售、云计算三个行业，具备较好的行业多样性，可以在不同的市场环境中分散风险。（例如，医疗健康行业通常对经济周期波动的敏感度较低，而云计算和在线零售则可能在经济增长期表现强劲。）
- 选择的行业都是当前和未来科技发展中不可忽视的重要领域，具有长期增长潜力，能够参与到技术创新和消费趋势的主流变化中。
- 将主动管理型的中欧医疗健康混合 A 与两个被动管理的 ETF 基金组合，结合了低费用的 ETF 基金和中等费用的混合型基金，有助于降低整体投资成本，同时实现分散投资和灵活配置，在保证风险分散的同时提高整体组合的潜在回报。

6.3.4 期货

(1) WTI 原油期货 (CLF5)

原油价格与全球经济复苏、能源需求、以及地缘政治紧张局势密切相关。在全球经济逐步复苏的背景下，能源需求特别是来自中国和新兴市场的需求增加，可能会推动油价上涨。同时，中东地区、俄乌冲突等地缘政治因素对油价有重要影响。2024 年，

美国和中国等主要经济体的复苏有望支撑能源需求，原油期货价格的波动将带来短期投资机会。

(2) 黄金期货 (GCG5)

黄金作为避险资产，通常在全球经济不确定性加剧时，表现出强劲的上涨势头。在 2023 年和 2024 年，由于通胀压力和货币政策的不确定性，黄金作为对冲通胀和地缘政治风险的工具，可能会吸引大量投资者。随着全球央行（如美联储和欧洲央行）的利率政策调整和经济的不确定性，黄金价格可能会迎来上行的机会。如果全球经济增长不确定性加大，或是美国经济面临更多压力，黄金期货将在避险需求中获得支撑，短期内可能出现较大的价格波动。

(3) 小麦期货 (ZWH5)

小麦期货受气候因素、全球供需关系以及国际贸易政策的影响较大。2023 年俄罗斯与乌克兰冲突对全球小麦供应产生了较大影响，而 2024 年全球农业形势可能继续受到气候、政治等因素的影响。特别是在全球粮食安全问题成为重点关注议题时，小麦价格可能波动剧烈。随着天气异常、全球需求波动、以及主要出口国的供应情况变化，小麦期货在短期内可能带来较高的投资回报。

(4) 玉米期货 (ZCH5)

玉米不仅作为粮食作物，还是生物燃料（如乙醇）的主要原料。随着能源转型的推进和全球农业需求的增加，玉米的需求仍然强劲。美国等主要玉米生产国的农业支持政策可能对价格产生影响。如果政策出台以增加供应或稳定市场，价格波动可能会受到抑制。

(5) 组合优势

- 每种期货合约的价格受到不同的市场和外部因素影响，可以对冲来自不同经济周期和地缘政治背景的风险。（例如，原油价格与全球经济和能源需求密切相关，而黄金则是避险资产，通常在全球经济不确定性加剧时表现较好。当全球经济不确定时，黄金可能上涨，而原油可能因为需求减少而下跌。小麦和玉米期货则受气候变化和农业政策等因素影响。）
- 期货资产通常与股票和债券的表现不完全相关，帮助降低投资组合整体波动性。
- 玉米和小麦期货可以较好的平衡季节性的风险波动，由于二者的收获时期不同，因此价格变化可能呈现出负相关的特性。

6.4 量化投资组合模型

在我们选择完国际市场中的投资标的之后，接下来我们将利用 Python 和线性规划（Linear Programming, LP）的方法，对最优化问题进行求解。

在该部分中，我们将采取两种方式来获得最终的投资组合（Investment Portfolio），分别在每大类资产中优化 CVaR，即条件风险价值（Conditional Value at Risk, CVaR）；之后在四大类资产中，利用 Markowitz 投资组合理论^[2]进行最小化风险。

6.4.1 数据获取及清洗

在国债、股票、基金、期货四类资产中，我们选取 2023 年 11 月 9 日至 2024 年 12 月 6 日，共 223 共同交易日的数据集进行操作；对于股票、基金、期货三类资产，我们依据 223 天的交易日的收盘价算出日收益率，共 222 天有效数据，并算出其平均收益率、标准差来度量各类资产的预期未来收益率及风险；对于国债资产，我们选择了对应的 222 天的平均年收益率作为各国债资产的预期未来收益率，并用年收益率波动的标准差来衡量国债风险。

相关的数据会附在报告中。

代码 6-1 数据清洗代码

```
1 import pandas as pd
2 import numpy as np
3 import cvxpy as cp
4 import os
5
6 types = ['股票', '基金', '期货', '国债']
7 os_path = "path/to/portfolio"
8
9 for i_ori in types:
10     def convert_dates( date_series ):
11         def convert_single_date( date_str ):
12             try:
13                 # 尝试转换中文日期格式
14                 return pd.to_datetime( date_str , format='%Y年%m月%d日')
15             except ValueError:
16                 # 如果转换失败，假设日期已经是标准格式
17                 return pd.to_datetime( date_str )
18
19             # 对日期序列中的每个日期进行转换
20             return date_series .apply( convert_single_date )
21
22     data = pd.DataFrame()
23
24     i = os.path.join( os_path, i_ori )
25
26     kinds = os. listdir ( i )
```

```
27
28     print(kinds)
29
30     for j in range(len(kinds)):
31         entire_path = os.path.join(i, kinds[j])
32         if j == 0:
33             data = pd.read_csv(entire_path, usecols=['日期', '收盘'])
34             data = data.rename(columns={'日期': 'date', '收盘': kinds[j].split('.csv')[0]})[0])
35             data['date'] = convert_dates(data['date'])
36         else:
37             temp_df = pd.read_csv(entire_path, usecols=['日期', '收盘'])
38             temp_df = temp_df.rename(columns={'日期': 'date', '收盘': kinds[j].split('.csv')[0]})[0])
39             temp_df['date'] = convert_dates(temp_df['date'])
40             data = pd.merge(data, temp_df, on='date')
41
42     print(data.head())
43
44     data.set_index('date', inplace=True)
45     # 将日期从倒序转为正序
46     data = data.sort_index(ascending=True)
47
48     data = data[data.index.isin(date_df)]
49
50     if i_ori != '国债':
51         data = data.replace(',', '', regex=True).apply(pd.to_numeric)
52         returns = data.pct_change().dropna()
53
54         # 对 returns 中的每个值进行 (returns.shape[0]) 次方运算
55         # n = returns.shape[0]
56         # returns = (1+returns) ** n - 1
57
58
59         print(returns.head(-5))
60     else:
61         returns = data.copy()
62         returns = returns.iloc[1:]
63
64         returns.to_csv(os.path.join(os_path, i_ori + '_returns.csv'))
65         print(f'{i_ori} 收益率summary:')
```

```

67     mean_returns = returns .mean()
68     print("平均收益率:")
69     print(mean_returns)
70
71     std_devs = returns .std()
72     print("标准差:\n")
73     print(std_devs)
74
75     cov_matrix = returns .cov()
76     print("协方差矩阵:")
77     print(cov_matrix)
78
79     corr_matrix = returns .corr()
80     print("相关系数矩阵:")
81     print(corr_matrix)

```

在该部分，我们特别处理了国债，因为国债本身即为收益率（不存在收盘价，只有收盘时的年利率）。其中，函数convert_dates尝试将日期字符串从中文格式转换为标准日期格式。如果转换失败，则假设日期已是标准格式。之后对各类资产进行遍历，利用pandas库读取数据，并进行空值等清洗处理。最终输出平均收益率、方差-协方差矩阵、相关系数矩阵等重要指标。

下面展示的是最终输出结果：

表 6-1 国债资产平均收益率及标准差

国家	平均收益率 (%)	标准差
中国	1.75	0.3362
保加利亚	3.28	0.1362
澳大利亚	4.17	0.1655
美国	4.83	0.3631

表6-1展示了国债资产的平均收益率及标准差。

表 6-2 股票资产平均收益率及标准差

指数	平均收益率 (%)	标准差
MSCI 新兴市场指数	12.81	0.0097
标准普尔 500 指数	35.92	0.0081
欧洲斯托克 50 指数	18.01	0.0088

表6-2展示了股票资产的平均收益率及标准差。

表 6-3 基金资产平均收益率及标准差

基金	平均收益率 (%)	标准差
中欧医疗健康混合 A	-16.79	0.0215
Global X 物联网 ETF	51.11	0.0152
富达云计算 ETF	65.96	0.0147

表6-3展示了基金资产的平均收益率及标准差。

表 6-4 期货资产平均收益率及标准差

期货	平均收益率 (%)	标准差
WTI 原油期货	-15.42	0.0201
玉米期货	-7.22	0.0161
小麦期货	-3.15	0.0192
黄金期货	29.39	0.0102

表6-4展示了期货资产的平均收益率及标准差。

需要注意的是，在上面标注出的平均收益率实际为 222 天收益率的平均值，年化平均收益率实则通过下面公式6-1得出

$$\text{年化平均收益率} = (\text{日化平均收益率} + 1)^{222} - 1 \quad (6-1)$$

标准差为 222 天日收益率的平均值，通过下面式6-2得出

$$\sigma = \sqrt{E(r_i - Er)^2} = \sqrt{\frac{1}{221} \sum_{i=1}^{222} (r_i - \bar{r})^2} \quad (6-2)$$

其中 r_i 表示每日的收益率， \bar{r} 表示 222 天收益率的平均值；这里分母是 221 的原因是，这是对方差的无偏估计（unbiased estimate）^②。

同时，我们看到代码6-1中包含了县官系数矩阵的运算，下面将相关系数矩阵展示出来：

^② 作者备注：伍德里奇《计量经济学》^[11]

表 6-5 国债资产相关系数矩阵

	中国	保加利亚	澳大利亚	美国
中国	1	0.220278	0.132751	0.537193
保加利亚	0.220278	1	-0.045990	-0.086859
澳大利亚	0.132751	-0.045990	1	0.478224
美国	0.537193	-0.086859	0.478224	1

表 6-6 股票资产相关系数矩阵

	MSCI 新兴市场指数	标准普尔 500 指数	欧洲斯托克 50 指数
MSCI 新兴市场指数	1	0.378431	0.462832
标准普尔 500 指数	0.378431	1	0.434583
欧洲斯托克 50 指数	0.462832	0.434583	1

表 6-7 基金资产相关系数矩阵

	中欧医疗健康混合 A	Global X 物联网 ETF	富达云计算 ETF
中欧医疗健康混合 A	1	0.378431	0.462832
全球 X 物联网 ETF	0.378431	1	0.434583
富达云计算 ETF	0.462832	0.434583	1

表 6-8 期货资产相关系数矩阵

	WTI 原油期货	玉米期货	小麦期货	黄金期货
WTI 原油期货	1	0.038618	0.010580	0.073209
玉米期货	0.038618	1	0.469429	0.003056
小麦期货	0.010580	0.469429	1	0.067717
黄金期货	0.073209	0.003056	0.067717	1

上面表6-5、6-6、6-7、6-8展示了各类资产的相关系数矩阵。

6.4.2 各类资产优化问题——CVaR

关于 CVaR 的知识背景介绍，我们将放到附录中进行，此处仅做简单的公式介绍，供读者了解具体优化逻辑；对 CVaR 的具体定义、推导，感兴趣的读者，可以到附录中进行详细的了解。

我们对各类资产使用 CVaR 作为风险损失测度，并通过最小化 CVaR 进行优化：

$$\min(CVaR) \quad (6-3)$$

$$s.t. \quad \sum_{i=1}^n w_i = 1 \text{ 且 } w_i \geq 0, \forall i \quad (6-4)$$

其中， n 为每类资产的个数， w_i 表示第 i 个资产的权重， $CVaR$ 表示投资组合的 CVaR。

相信读者读到此处已经理解了线性规划的对象，但为了使得读者更深刻理解公式中各符号的具体含义，我们给出如下概念解释：

投资组合收益

$$\text{portfolio return} = R \cdot w = \sum_{i=1}^n w_i \cdot r_i \quad (6-5)$$

其中， R 为收益率矩阵， w 为权重向量。

CVaR (条件风险价值)

$$CVaR = VaR + \frac{\text{mean}(\{\text{portfolio return} - VaR\}^+)}{1 - \alpha} \quad (6-6)$$

其中， $\{\cdot\}^+$ 表示的是函数 $\max\{0, \cdot\}$ ，即若里面的数 > 0 ，则取里面的数，若里面的数 ≤ 0 ，则取 0，其实际意义表示 excess loss，是指投资组合 portfolio return 中的超过 VaR 的损失部分；mean 表示求平均值；VaR 是一个特定置信水平 α 下的损失临界值； α 为置信水平，这里定义 $\alpha = 0.99$ 。

下面我们给出实现该过程的代码：

代码 6-2 各类别资产 CVaR 优化过程代码

```

1 alpha = 0.99
2 n = returns .shape[1]
3 w = cp. Variable (n)
4 a = cp. Variable ()
5 exceedance = cp. Variable ( returns .shape [0], nonneg=True)
6 portfolio _return = returns .values @ w
7 constraints = [cp.sum(w) == 1, w >= 0]
8 constraints += [cp.sum(exceedance) <= (1 - alpha) * len(returns)]
9 constraints += [exceedance >= portfolio _return - a]
10 positive_diff = cp.maximum(-portfolio _return + a, 0)
11 expected_positive_diff = cp.mean(positive_diff )
12 objective = cp.Minimize(a + (1 / (1 - alpha)) * expected_positive_diff )

```

```

13     problem = cp.Problem(objective , constraints )
14     problem.solve ()
15
16     expected_portfolio_return  = mean_returns.values @ w.value
17     expected_portfolio_risk   = np.sqrt(w.value.T @ cov_matrix.values @ w.value)
18
19     print(f"\{ i_ori \} Optimal Weights:", w.value)
20     portfolio_return_value  = returns .values @ w.value
21     portfolio_returns_dict [ i_ori ] = portfolio_return_value

```

上面的代码6-2也是在for `i_ori` in `types`:循环 loop 下的，其中`types = ['股票', '基金', '期货', '国债']`。

最终，要保存最终结果，这里方式是通过`with open(os.path.join(os_path, i_ori) + '_final_results .txt', 'w') as f:`输出为 txt 文件，完整代码如下代码6-3所示。

代码 6-3 各类别资产优化完整代码

```

1 import pandas as pd
2 import numpy as np
3 import cvxpy as cp
4 import os
5
6 # 初始变量
7 result_files_pre  = []
8 result_files  = []
9 portfolio_returns_dict  = {} # 存储每个 i_ori 的 portfolio_return_value
10
11 types = ['股票', '基金', '期货', '国债']
12 os_path = "path/to/portfolio"
13
14 for i_ori in types:
15     def convert_dates( date_series ):
16         def convert_single_date ( date_str ):
17             try:
18                 # 尝试转换中文日期格式
19                 return pd.to_datetime( date_str , format='%Y年%m月%d日')
20             except ValueError:
21                 # 如果转换失败, 假设日期已经是标准格式
22                 return pd.to_datetime( date_str )
23
24         # 对日期序列中的每个日期进行转换
25         return date_series .apply(convert_single_date )
26

```

```
27     data = pd.DataFrame()
28
29     i = os.path.join(os_path, i_ori)
30
31     kinds = os.listdir(i)
32
33     print(kinds)
34
35     for j in range(len(kinds)):
36         entire_path = os.path.join(i, kinds[j])
37         if j == 0:
38             data = pd.read_csv(entire_path, usecols=['日期', '收盘'])
39             data = data.rename(columns={'日期': 'date', '收盘': kinds[j].split('.csv')[0]})[0]
40             data['date'] = convert_dates(data['date'])
41         else:
42             temp_df = pd.read_csv(entire_path, usecols=['日期', '收盘'])
43             temp_df = temp_df.rename(columns={'日期': 'date', '收盘': kinds[j].split('.csv')[0]})[0]
44             temp_df['date'] = convert_dates(temp_df['date'])
45             data = pd.merge(data, temp_df, on='date')
46
47     print(data.head())
48
49     data.set_index('date', inplace=True)
50     # 将日期从倒序转为正序
51     data = data.sort_index(ascending=True)
52
53     data = data[data.index.isin(date_df)]
54
55     if i_ori != '国债':
56         data = data.replace(',', '', regex=True).apply(pd.to_numeric)
57         returns = data.pct_change().dropna()
58
59         # 对 returns 中的每个值进行 (returns.shape[0]) 次方运算
60         # n = returns.shape[0]
61         # returns = (1+returns) ** n - 1
62
63
64     print(returns.head(-5))
65 else:
66     returns = data.copy()
```

```
67     returns = returns.iloc [1:]  
68  
69     returns.to_csv(os.path.join(os_path, i_ori + '_returns.csv'))  
70     print(f'{i_ori} 收益率summary:')  
71  
72     mean_returns = returns.mean()  
73     print("平均收益率:")  
74     print(mean_returns)  
75  
76     std_devs = returns.std()  
77     print("标准差:\n")  
78     print(std_devs)  
79  
80     cov_matrix = returns.cov()  
81     print("协方差矩阵:")  
82     print(cov_matrix)  
83  
84     corr_matrix = returns.corr()  
85     print("相关系数矩阵:")  
86     print(corr_matrix)  
87  
88     alpha = 0.99  
89     n = returns.shape[1]  
90     w = cp.Variable(n)  
91     a = cp.Variable()  
92     exceedance = cp.Variable(returns.shape[0], nonneg=True)  
93     portfolio_return = returns.values @ w  
94     constraints = [cp.sum(w) == 1, w >= 0]  
95     constraints += [cp.sum(exceedance) <= (1 - alpha) * len(returns)]  
96     constraints += [exceedance >= portfolio_return - a]  
97     positive_diff = cp.maximum(-portfolio_return + a, 0)  
98     expected_positive_diff = cp.mean(positive_diff)  
99     objective = cp.Minimize(a + (1 / (1 - alpha)) * expected_positive_diff)  
100    problem = cp.Problem(objective, constraints)  
101    problem.solve()  
102  
103    expected_portfolio_return = mean_returns.values @ w.value  
104    expected_portfolio_risk = np.sqrt(w.value.T @ cov_matrix.values @ w.value)  
105  
106    print(f'{i_ori} Optimal Weights:', w.value)  
107    portfolio_return_value = returns.values @ w.value  
108    portfolio_returns_dict [i_ori] = portfolio_return_value
```

```

109
110     with open(os.path.join(os_path, i_ori) + '_final_results .txt', 'w') as f:
111         f.write("平均收益率:\n")
112         f.write(mean_returns.to_string())
113         f.write("\n\n标准差:\n")
114         f.write(std_devs.to_string())
115         f.write("\n\n协方差矩阵:\n")
116         f.write(cov_matrix.to_string())
117         f.write("\n\n相关系数矩阵:\n")
118         f.write(corr_matrix.to_string())
119         f.write("\n\n-----\n")
120         f.write("\nOptimal Weights:\n")
121         f.write(str(w.value))
122         f.write("\n\nExpected Portfolio Return:\n")
123         f.write(str(expected_portfolio_return))
124         f.write("\n\nExpected Portfolio Risk (Standard Deviation):\n")
125         f.write(str(expected_portfolio_risk))

126
127     print(f"{i_ori} 结果已写入 results.txt 文件")
128
129     # 创建新的DataFrame用于存储每个i_ori的portfolio_return_value并将其保存到CSV文件
130     dfdd = pd.DataFrame({i_ori: portfolio_returns_dict [i_ori] })
131     if i_ori == '国债':
132         dfdd['date'] = data.index[1:]
133     else:
134         dfdd['date'] = data.index[1:]
135     dfdd.set_index('date', inplace=True)
136     dfdd.to_csv(os.path.join(os_path, i_ori + '_portfolio_return.csv'))
137
138     # 将所有 portfolio_return_value 汇总到一个新的DataFrame中
139     portfolio_returns_df = pd.DataFrame( portfolio_returns_dict )
140     portfolio_returns_df .index = date_df [1:]
141     # 保存到CSV文件中
142     portfolio_returns_df .to_csv(os.path.join(os_path, 'portfolio_returns_summary.csv'),
143     encoding='utf_8_sig')
143     print("所有组合收益率已汇总并保存到 portfolio_returns_summary.csv 文件中")

```

最终各大类资产的最有资产组合权重配比如下面图表6-5所示。

国债	配比(%)	股票	配比(%)	基金	配比(%)	期货	配比(%)
中国	3. 7395	MSCI新兴市场指数	7. 3101	中欧医疗健康混合A	41. 8209	WTI原油期货	15. 0910
保加利亚	50. 2621	SPY	47. 1601	IBUY	31. 4424	玉米期货	30. 2858
澳大利亚	38. 3578	Eurostoxx50指数	45. 5298	SKYY	26. 7368	美国小麦	10. 4891
美国	7. 6407					黄金期货	44. 3414
年化收益率	3. 683%		28. 938%		25. 984%		11. 382%
资产标准差	4. 67E-06		0. 006980		0. 012660		0. 008437

图 6-5 各类资产最终优化结果

6.4.3 各类别投资组合优化问题——基于 Markowitz 投资组合理论

在该部分，读取的数据是上一部分运行的结果，这里的优化问题如下面公式所示（同样只做简单的介绍）：

$$\min \omega^T \sum \omega \quad (6-7)$$

$$s.t. \quad 1^T \omega = 1, \quad \omega \geq 0 \quad (6-8)$$

$$\omega^T R \geq r_0 \quad (6-9)$$

上面 ω 是 4×1 的四维向量，代表四类资产的投资权重， Σ 是四类资产的协方差矩阵（非相关系数矩阵），限制条件中，第一个限制条件要求四类资产的投资权重相加和为 1，并且每个权重都非负，即不允许做空交易，同时需要保证投资组合的收益大于等于 r_0 ，其中 r_0 是人为规定的想要达到的预期收益率。

具体实现的代码如代码6-4所示：

代码 6-4 Markowitz 投资组合优化代码

```
1 import pandas as pd
2 import numpy as np
```

```
3 from scipy.optimize import minimize
4
5 # 读取CSV文件
6 file_path = 'path/to/your/data/portfolio_returns_summary.csv' # 替换为实际文件路径
7 data = pd.read_csv(file_path)
8
9 # columns_name = 日期, 股票, 基金, 期货, 国债
10 columns = ["date", "股票", "基金", "期货", "国债"]
11 data = data[columns]
12
13 # 日期格式转换并排序
14 data["date"] = pd.to_datetime(data["date"])
15 data = data.sort_values("date")
16
17 # 计算股票、基金、期货的累计收益率和年化收益率
18 for asset in ["股票", "基金", "期货"]:
19     data[f"{asset}_累计收益率"] = 1*((1 + data[asset]).cumprod() - 1)
20     data[f"{asset}_年化收益率"] = 1**(((1 + data[f"{asset}_累计收益率"])**(222 / (data.index + 1))) - 1)
21
22 # 提取股票、基金、期货和国债的年化收益率
23 annualized_returns = 100 * data[["股票_年化收益率", "基金_年化收益率", "期货_年化收益率"]]
24 annualized_returns["国债_年化收益率"] = data["国债"] # 国债年化收益率直接使用
25 print(annualized_returns)
26
27 # 计算年化协方差矩阵
28 annualized_cov_matrix = annualized_returns.cov()
29
30 # 提取最后一行的年化收益率作为目标收益
31 expected_returns = annualized_returns.iloc[-1].values
32
33 # 定义马克维茨模型优化
34
35 # 目标函数：最小化组合的风险（方差）
36 def portfolio_variance(weights, cov_matrix):
37     return weights.T @ cov_matrix @ weights
38
39 # 收益约束
40 def portfolio_return_constraint(weights, expected_returns, target_return):
41     return weights.T @ expected_returns - target_return
42
```

```
43 # 权重总和约束
44 def weights_sum_constraint(weights):
45     return np.sum(weights) - 1
46
47 # 初始权重
48 num_assets = len(expected_returns)
49 initial_weights = np.ones(num_assets) / num_assets
50
51 # 目标期望收益
52 target_return = 13.7 # 使用平均年化收益率作为目标
53
54 # 约束和边界
55 constraints = [
56     {'type': 'eq', 'fun': portfolio_return_constraint, 'args': (expected_returns,
57     target_return)},
58     {'type': 'eq', 'fun': weights_sum_constraint},
59 ]
60 bounds = [(0, 1) for _ in range(num_assets)]
61
62 # 优化
63 result = minimize(
64     portfolio_variance,
65     initial_weights,
66     args=(annualized_cov_matrix,),
67     method='SLSQP',
68     bounds=bounds,
69     constraints=constraints,
70 )
71
72 # 输出最优配置权重
73 if result.success:
74     optimal_weights = result.x
75     assets = annualized_returns.columns
76     optimal_allocation = {assets[i]: optimal_weights[i] for i in range(num_assets)}
77
78 # 计算组合预期年化收益率和年化风险（标准差）
79 portfolio_return = np.dot(optimal_weights, expected_returns)
80 portfolio_variance = portfolio_variance(optimal_weights, annualized_cov_matrix)
81 portfolio_std_dev = np.sqrt(portfolio_variance)
82
83 # 夏普比率计算（使用国债的平均年化收益率作为无风险收益率）
84 risk_free_rate = data["国债"].mean() # 近似使用国债平均收益率
```

```

84     sharpe_ratio = ( portfolio_return - risk_free_rate ) / portfolio_std_dev
85
86     # 输出结果
87     print("最优配置比例: ", optimal_allocation)
88     print("标准协方差矩阵: \n", pd.DataFrame(annualized_cov_matrix, index=assets,
89                                         columns=assets))
90     print(f"组合预期年化收益率: {portfolio_return :.4 f}")
91     print(f"组合年化方差: { portfolio_variance :.6 f}")
92     print(f"组合年化标准差: {portfolio_std_dev :.4 f}")
93     print(f"组合夏普比率: {sharpe_ratio :.4 f}")
94     # 导出结果到 CSV 文件
95     output_file = "path/to/your/output/ optimal_portfolio_results .csv"
96     results = {
97         "资产类别": [ "组合预期年化收益率", "组合年化方差", "组合年化标准差", "组合
98             夏普比率"],
99         "权重或指标值": [ portfolio_return , portfolio_variance , portfolio_std_dev ,
100                         sharpe_ratio ],
101     }
102     results_df = pd.DataFrame(results)
103     results_df .to_csv( output_file , index=False)
104     print(f"结果已导出至: {output_file }")
105 else :
106     print("优化失败: ", result .message)

```

最终的优化结果如下表6-9所示:

表 6-9 各大类资产在最终投资组合中占比

资产类别	权重 (%)	年化收益率 (%)	资产标准差
国债	34.1077	3.6825	4.67×10^{-6}
股票	17.7498	28.9377	0.006980
基金	19.4307	25.9842	0.012660
期货	28.7118	11.3823	0.008437

最终构架出的国际投资组合模型中，组合预期年化收益率达到了 13.70%，组合年化标准差为 0.1132。

注：在此处没有给出最大回撤，原因是对于价格的预期始终采取的都是简单的以过去的平均值作为标准。

6.5 国际投资组合模型搭建过程总结

在该部分，我们构建了国际投资组合模型，与之前的量化交易策略和课上所学知识略有不同的是，在此处，作者不再把重心放在对股票价格进行预测上，而是把主要着力点放在了对资产组合中个资产的配比的优化中。值得注意的是，本文的该部分和上一部分，可以认为是是从两个角度对量化投策略进行合理的搭建。

但是，如果简单的应用过去一年平均收益率作为未来预期收益率，应用过去一年的标准差度量未来预期的风险，的确在对未来的预测效果上或许有较大程度的偏差，但是在本部分中，应用到作者本学期课程《最优化方法》中的线性规划模型，同时应用到作者本学期课程《投资学》中的投资组合模型构建方法，同时在考虑汇率风险时，主要用到《国际金融》课程的相关知识，同时在进行 CVaR 优化中，应用到了《泛函分析》中测度、度量的相关概念。

在上面利用 xgboost 进行股票择时的模型中，则主要灵感来源于作者本学期课程《人工智能与 Python 程序设计》的相关知识及房产定价的实践项目；以及课程《金融时间序列分析》中对于时间序列数据的处理方法。

因此，总体来说，本篇报告融合了作者在课程学习中的各种灵感，对于真正的量化投资策略的搭建，可以考虑融合股票择时与投资组合选择，以此来最大化投资的收益、最小化投资的风险。

7 结语

本文是《金融大数据分析与量化交易》最终期末量化交易策略搭建报告，其中内容灵感主要来自于：

1. 覃老师在课上讲到的一些算法和量化交易策略。
2. 平时在公众号、网站等领域看到的一些关于量化投资的介绍文章，以及本课程期中大作业对于量化行业和量化公司的调研。
3. 《投资学》课程中学习到的 Markowitz 投资组合理论，以及其他的风险投资基本概念。
4. 《金融时间序列分析》中一些对时间序列数据的了解和常见的处理方式。
5. 《人工智能与 Python 程序设计》课程中学习到的机器学习方法，以及房产定价的实战项目中特征工程、模型训练等的实践。
6. 《最优化方法》课程对于线性规划（Linear Programming, LP）问题的学习。
7. 《泛函分析》、《国际金融》课程中的知识，使我对于一些概念的理解更加深刻。

本文有很多附件，在“随机森林”文件夹中，包含了所有的文档（从数据获取、到数据清洗、到特征工程、到模型训练、到模型评价）；在“Xgboost”文件夹中，包含了从 CSMAR 下载下来的三只股票的数据，以及 notebook 文件，其是完整的代码，且其中的“portfolio_returns.csv”即为本文附件 B 没有完全表示的调仓结果；在“international”文件夹中，是本文关于国际投资策略搭建的数据和代码。

参考文献

- [1] BREITUNG C. Automated stock picking using random forests[J/OL]. Journal of Empirical Finance, 2023. DOI: <https://doi.org/10.1016/j.jempfin.2023.05.001>.
- [2] MARKOWITZ H M. Portfolio selection[J]. Journal of Finance, 1952, 7(1): 77-91.
- [3] MARKOWITZ H M. Portfolio selection: Efficient diversification of investments[M]. New York: John Wiley & Sons, 1959.
- [4] FAMA E F. Market efficiency, long-term returns, and behavioral finance[J]. Journal of Financial Economics, 1998, 49(3): 283-306.
- [5] FAMA E F. Efficient capital markets: A review of theory and empirical work[J]. The Journal of Finance, 1970, 25(2): 383-417.
- [6] 余永定. 最后的屏障：资本项目自由化和人民币国际化之辩[M]. 中国社会科学出版社, 2014.
- [7] 熊爱宗杨嘉豪. 国际货币体系向多元化演进[J]. 世界知识, 2023(11).
- [8] MATEIS C. Extending disjunctive logic programming by t-norms[C/OL]//Logic Programming and Nonmonotonic Reasoning. Springer, 1999: 219-229. https://doi.org/10.1007/3-540-46767-X_21.
- [9] 中国经济观察 - KPMG[EB/OL]. 毕马威会计师事务所, 2024. <https://assets.kpmg.com/content/dam/kpmg/cn/pdf/zh/2024/08/china-economic-monitor-q3-2024.pdf>.
- [10] 毕马威会计师事务所. 中国经济观察: 2024 年四季度[EB/OL]. 毕马威会计师事务所, 2024. <https://assets.kpmg.com/content/dam/kpmg/cn/pdf/zh/2024/11/china-economic-monitor-q4-2024.pdf>.
- [11] WOOLDRIDGE J M. Introductory econometrics: A modern approach[M]. 7th ed. Cengage Learning, 2019.
- [12] JORION P. Value at risk: The new benchmark for managing financial risk[M/OL]. 3rd ed. McGraw-Hill, 2006. <https://www.mcgraw-hill.com/>.
- [13] 刘晓星. 基于 CVaR 的投资组合优化模型研究[J]. 商业研究, 2006, 346(14): 15-18.
- [14] Basel Committee on Banking Supervision (BCBS) - Overview[EB/OL]. [2023-11-15]. <https://www.bis.org/bcbs/index.htm>.
- [15] LIU H, GAO Y. Study on volatility of stock market: Empirical analysis based on arma-tgarch-m model[J/OL]. International Business School, Shaanxi Normal University, 2015. <http://html.rhhz.net/BJHKHTDXXBSKB/20170409.htm>.
- [16] HUANG W, ZHENG Z. 基于高阶矩的金融资产定价和配置[J/OL]. Journal of Fuzhou University (Philosophy and Social Sciences), 2010, 1(95): 23-29. <https://core.ac.uk/download/pdf/41378321.pdf>.
- [17] ARTZNER P, DELBAEN F, EBER J M, et al. Conditional risk measures[J/OL]. ETH Z, 1997. <https://people.math.ethz.ch/~delbaen/ftp/preprints/CoherentMF.pdf>.
- [18] ARTZNER P, DELBAEN F, EBER J, et al. Coherent measures of risk[J]. Mathematical Finance, 1999, 9(3): 203-228.
- [19] ROCKAFELLAR R T, URYASEV S. Optimization of conditional value-at-risk[J]. Journal of Risk, 2000, 2(3): 21-41.

- [20] URYASEV S. Conditional value-at-risk: Optimization algorithms and applications[J]. Financial Engineering News, 2000, 14(February): 1-5.
- [21] PLUNG G C. Some remarks on the value-at-risk and conditional value-at-risk[J]. Journal of Risk, 1999.
- [22] URYASEV S. Constrained optimization: Methodology and applications[M]. Kluwer Academic Publishers, 2000: 272-281.
- [23] 徐永春高岳林. 金融风险管理的 CVaR 方法及实证分析[J]. 金融研究, 2009, 2(11): 11-12.
- [24] MAUSER H, ROSEN D. Applying scenario optimization to portfolio credit risk[J]. Journal of Risk Finance, 1999, 2(2): 36-48.
- [25] ANDERSSON F, MAUSSER H, ROSEN D, et al. Credit risk optimization with conditional value-at-risk[J/OL]. Mathematical Programming, Series B, 2001, 89: 273-291. <https://link.springer.com/article/10.1007/PL00011399>. DOI: 10.1007/s101070000201.

附录 A CVaR 的理论基础

A.1 风险价值 (VaR) 测度的改进

依据 Jorion 给出的定义, VaR (Value at Risk) 是指在一定的概率 (置信) 水平下, 某一金融资产或投资组合在未来特定一段时间内操守的最大可能损失^[12], 可以将其表示为

$$\text{Prob}(\Delta p > VaR) = 1 - \beta$$

其中, Δp 表示投资组合在持有期 Δt 内的损失, β 表示置信水平, VaR 为置信水平 β 下处于风险中的价值, 即在某个确定的概率水平下, 损失不会超过 VaR 。这一定义以最简单的形式将不同市场风险因子集成为一个数, 充分考虑了金融资产对某种风险来源 (如利率、汇率、股票价格等基础性金融变量) 的一个最优解敞口和市场逆向变化的可能性, 较为准确的测量了由不同风险来源及相互作用而产生的潜在损失^[13]。

由于 VaR 为不同金融工具构成的复杂的投资组合提供了统一的综合性的测量框架, 因此迅速发展成为测量和控制金融风险管理的国际主流技术, 被巴塞尔委员会等国际金融机构却认为金融风险测度的国际标准^[14]。但从数学层面来看, VaR 本质上只是某一个置信水平下的分位点, 并通过这个置信水平得到的分位点来描述后面整个尾部分布, 因此无法考察超过分位点的下方风险信息, 导致 VaR 尾部损失测量的非充分性^[13]。由于 VaR 尾部其实是小概率事件, 即 VaR 低估了小概率发生的巨额损失情形 (比如股市崩盘、金融危机等)。

另外, 大量的实证研究表明, 金融资产回报的波动与正态分布相差甚远, 普遍存在厚尾性和非对称性^[15-16]。而当 VaR 在损益分布为非正态分布时, 其不是一致性的风险度量, 不满足次可加性, 因而不满足凸性, 进而不满足存在唯一的最优组合决策 (局部最优不一定为整体最优)^[13]。

为了弥补 VaR 的不足, 1997 年, Artzner 提出了 CVaR (Conditional VaR 或 Expected shortfall) 度量, CVaR 是指投资组合的损失大于某个给定 VaR 值得条件下, 该投资组合的损失平均值^[17]; 1999 年, Artzner 又通过不同的方式证明了 CVaR 满足次可加性和凸性, 符合一致性的风险度量^[18]; 2000 年, Rockafeller 等人证明了基于 CVaR 的投资组合优化必定存在最小风险的解^[19]; CVaR 被认为是比 VaR 更好的一致性的风险度量^[20]。

A.2 CVaR 的定义

条件风险值 (CVaR) 度量的是超过 VaR 的尾部损失的平均值，代表了超额损失的平均水平，其定义以 VaR 为基础^[21]。设 Y 表示一种资产或资产组合的损益，其是一个随机变量，分布函数为 F_Y ， $F_Y(u) = P\{Y \leq u\}$ ，记 $F_Y^{-1}(v)$ 为它左连续的逆运算，则 $F_Y^{-1}(v) = \min\{u : F_Y \geq v\}$ ，其概率密度为 $f(y)$ 。而在给定的时间间隔 Δt 内和置信水平 α 下，VaR 值即为相应的 α 分位数。

$$VaR_\alpha = F_Y^{-1}(\alpha) \quad (\text{A-1})$$

其中，

$$VaR_\alpha(Y) = -VaR_{1-\alpha}(-Y) = E[Y | Y \leq VaR_\alpha(Y)]$$

依据 CVaR 的定义，我们可以得到

$$CVaR_\alpha(Y) = E[Y | Y \geq VaR_\alpha(Y)] \quad (\text{A-2})$$

$$= \frac{1}{1-\alpha} \int_{VaR}^{+\infty} y \cdot f(y) dy \quad (\text{A-3})$$

$$= \frac{1}{1-\alpha} \int_\alpha^1 F^{-1}(v) dv \quad (\text{A-4})$$

Plung 给出了如下等价形式^[21]

$$CVaR_\alpha(Y) = \inf\left\{a + \frac{1}{1-\alpha} \cdot E[(Y - a)^+] : a \in \mathbb{R}\right\} \quad (\text{A-5})$$

其中， $(y - a)^+ = \max(y - a, 0)$ 。

A.3 CVaR 的性质

(1) 次可加性 (sub-additive)

$$CVaR_\alpha(Y_1 + Y_2) \leq CVaR_\alpha(Y_1) + CVaR_\alpha(Y_2) \quad (\text{A-6})$$

次可加性是资产组合决策优化的一个基本前提条件，意味着投资组合的风险值不超过其各个组成部分的风险值之和，因此为资产组合分散风险提供了理论支撑。

(2) 正齐次性 (positive homogeneous)

$$CVaR_\alpha(h \cdot Y) = h \cdot CVaR_\alpha(Y) \quad (A-7)$$

此条件反映了没有分散风险的效应，实则为次可加性的特例。

(3) 单调性 (monotonous)

$$CVaR_\alpha(Y_1) \leq CVaR_\alpha(Y_2), \quad \text{if } Y_1 \succeq_{D(2)} Y_2 \quad (A-8)$$

该条件说明与二阶随机占优是一致的；此处假设投资者风险厌恶。若一个投资组合占优于另一个投资组合，即前者的随机回报的各分量大于或等于后者随机回报所对应的分量，则前者的风险至少不大于后者。

(4) 传递不变性 (translation invariant)

$$CVaR_\alpha(Y + c) = CVaR_\alpha(Y) + c, \quad c \geq 0 \quad (A-9)$$

其中 c 表示无风险资产的收益。

该条件表明，若增加无风险的头寸到资产组合中，组合风险将随无风险头寸的增加而减少。

(5) 凸性 (convexity)

$CVaR$ 是凸性的，对于 $\forall Y_1, Y_2, 0 \leq \lambda \leq 1$ ，有

$$CVaR_\alpha[\lambda Y_1 + (1 - \lambda)Y_2] \leq \lambda CVaR_\alpha(Y_1) + (1 - \lambda)CVaR_\alpha(Y_2) \quad (A-10)$$

正是由于 $CVaR$ 具有凸性的性质，使得其可以通过线性规划 (LPT) 实现最小化。式 (A-5) 加上约束条件，既可以变换为关于资产组合风险的最小化问题^[22]。

整体上说， $CVaR$ 和 VaR 测度了损失分布的不同特性。 VaR 的计算揭示了损失超过它的可能性，但是没有计算出具体超过的损失程度；而 $CVaR$ 的风险测度是大于 VaR 的尾部损失的平均值，因此需要使用到超过 VaR 的尾部损失的所有值和发生概率，其对尾部的损失测度更加充分^[13,23]。如果相同资产的分布函数在 VaR 阈值处没有出现跳跃时，置信水平越接近 1，两者对损失衡量越趋于一致^[19]；当不考虑尾部因素时，二者的测度值相等。

A.4 CVaR 模型的理论搭建

请注意，该部分内容是理论上的模型搭建，在实际应用中，本文国际投资组合模型的搭建过程即为实际应用方法，此外还有模拟法等。

设 $z = f(x, y) : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}$ ，表示一个投资组合的损失函数；决策变量 $x \in X \subset \mathbb{R}^n$ 表示一个投资组合的权重向量， X 表示投资组合的可行集；随机变量 $y \in Y \subset \mathbb{R}^m$ 表示许多影响收益率的观测变量的未来值（如利率、物价等），当 y 是一个已知概率分布的随机变量时， $P(y) : \mathbb{R}^m \rightarrow \mathbb{R}$ ； z 为一个随机变量，且其分布依赖于 x 的选择。用 $\psi(x, \alpha)$ 表示概率函数：

$$\psi(x, \alpha) = \int_{f(x, y) \leq \alpha} p(y) dy \quad (A-11)$$

$\psi(x, \alpha)$ 为损失函数 $f(x, y)$ 没有超过临界值 α 的概率。这里 VaR 函数 $\alpha(x, \beta)$ 定义为

$$\alpha(x, \beta) = \min\{\alpha \in \mathbb{R} : \psi(x, \alpha) \geq \beta\} \quad (A-12)$$

其中， $\beta, \alpha \in (0, 1)$ 为置信水平。依据前面对 CVaR 的定义，设 CVaR 的函数为 $\Psi(x)$ ^[24]

$$\Psi(x) = \frac{1}{1 - \beta} \int_{f(x, y) \geq \alpha(x, \beta)} f(x, y) \cdot p(y) dy \quad (A-13)$$

$\Psi(x)$ 表示损失函数 $f(x, y)$ 在超过分位数 $\alpha(x, \beta)$ 条件下的损失期望值。其中，决策向量 x 属于可行集 $X \subset \mathbb{R}^n$ ， $p(y)$ 表示观测变量 y 的概率分布。于是，求解损失函数 $\Psi(x)$ 的最小值可以转化为求以下函数的最小值^[25]

$$\min_{\alpha \in \mathbb{R}} F_\beta(x, \alpha) = \Psi(x) \quad (A-14)$$

其中， $F_\beta(x, \alpha) = \alpha + (1 - \beta)^{-1} \cdot \int_{y \in \mathbb{R}^m} [f(x, y) - \alpha]^+ p(y) dy$

式 (A-14) 关于 α 的最优解 α^* 可以由式 (A-14) 对 α 求导数取零值得到，即

$$\begin{aligned} 1 + (1 - \beta)^{-1} \cdot (\psi(x, \alpha) - 1) &= 0 \\ \Rightarrow \psi(x, \alpha) &= \beta \end{aligned}$$

于是，我们就有 $\min_{x \in X, \alpha \in \mathbb{R}} F_\beta(x, \alpha) = \min_{x \in X} \min_{\alpha \in \mathbb{R}} F_\beta(x, \alpha) = \min_{x \in X} F_\beta(x, \alpha(x, \beta)) = \min_{x \in X} \Psi(x)$ 。由此我们发现，通过对函数 $F_\beta(x, \alpha)$ 进行最小化，可以同时求出 VaR 和优化 CVaR，VaR 为极小值点，CVaR 为极小值。

在一般情况下， $F_\beta(x, \alpha)$ 是光滑的，即密度函数 $p(y)$ 和损失函数 $f(x, y)$ 是光滑的且函数 $f(x, y)$ 基于 y 斜率不为 0^[20]。同时，由于函数 $F_\beta(x, \alpha)$ 都是凸的^[20]，所以最小化 CVaR 就可以转化为解决一个光滑凸的优化问题。

如果式 (A-14) 的积分能够用逼近分析来计算，那么就可以用非线性规划来优化函数 $F_\beta(x, \alpha)$ 。当密度函数 $p(y)$ 的解析表达式无法求得时，可以应用情景分析法^①来模拟。例如，根据投资组合资产价格的历史数据或者用蒙特卡洛模拟来为证券定价，由此得到 J 个数据，分别为 y_1, y_2, \dots, y_J 。这里相当于用密度函数 $p(y)$ 的样本 y_j 取近似计算函数 $F_\beta(x, \alpha)$ ，此时有

$$\int_{y \in \mathbb{R}^m} (f(x, y) - \alpha)^+ p(y) dy \approx \frac{1}{J} \sum_{j=1}^J (f(x, y_j) - \alpha)^+$$

由于损失函数 $f(x, y_j)$ 和可行集 X 时凸的，所以有

$$\begin{aligned} \widetilde{F}_\beta(x, \alpha) &\stackrel{\text{def}}{=} \alpha + ((1 - \beta)J)^{-1} \sum_{j=1}^J (f(x, y_j) - \alpha)^+ \\ &\min_{x \in X, \alpha \in \mathbb{R}} \widetilde{F}_\beta(x, \alpha) \end{aligned} \tag{A-15}$$

求解式 (A-15) 就可以求出向量 x^* ，VaR 为 α^* ，最优的 CVaR 等于 $\widetilde{F}_\beta(x^*, \alpha^*)$ 。

相关实证研究表明，一般情况下，损失函数 $f(x, y)$ 是关于 x 的线性函数^[19]，即

$$f(x, y) = x_1 f_1(y) + x_2 f_2(y) + \dots + x_n f_n(y) \tag{A-16}$$

此时函数 $F_\beta(x^*, \alpha^*)$ 是关于 (x, α) 的凸的分段线性函数，引入虚拟变量 z_j ， $z_j = (f(x, y_j) - \alpha)^+$ ，则 $z_j \geq f(x, y_j) - \alpha$ 。由此最初的 CVaR 函数就可以转化为线性函数和线性约束，于是，上述的优化问题就可以转化为如下的线性规划 LP 问题

$$\min_{x \in \mathbb{R}^n, z \in \mathbb{R}^J, \alpha \in \mathbb{R}} \{\alpha + ((1 - \beta)J)^{-1} \sum_{j=1}^J z_j\} \tag{A-17}$$

$$s.t. x \in X \tag{A-18}$$

$$z_j \geq f(x, y_j) - \alpha, \quad z_j \geq 0, \quad j = 1, 2, \dots, J \tag{A-19}$$

^① 作者备注：情景分析法又称脚本法或者前景描述法，是假定某种现象或某种趋势将持续到未来的前提下，对预测对象可能出现的情况或引起的后果作出预测的方法。通常用来对预测对象的未来发展作出种种设想或预计，是一种直观的定性预测方法。

附录 B 基于 Xgboost 预测模型的投资组合调仓数据

表 2-1 Portfolio Returns

Trddt	Moutai_Weight	Ningde_Weight	Yunnan_Weight	Portfolio_Return
2024-01-02	0.3333	0.3333	0.3333	
2024-01-03	0.0494	0.8962	0.0544	-0.0054
2024-01-04	0.7457	0.0122	0.2421	-0.0126
2024-01-05	0.5113	0.2532	0.2355	-0.0044
2024-01-08	0.3500	0.1828	0.4672	-0.0101
2024-01-09	0.1706	0.4440	0.3854	-0.0030
2024-01-10	0.2643	0.5056	0.2301	0.0048
2024-01-11	0.0893	0.4561	0.4546	0.0131
2024-01-12	0.2519	0.0403	0.7078	-0.0021
2024-01-15	0.1784	0.0762	0.7454	0.0060
2024-01-16	0.3553	0.1526	0.4920	0.0116
2024-01-17	0.6381	0.1193	0.2426	-0.0323
2024-01-18	0.2112	0.2887	0.5000	0.0118
2024-01-19	0.5859	0.0125	0.4016	0.0066
2024-01-22	0.2527	0.0446	0.7027	-0.0120
2024-01-23	0.0395	0.0489	0.9116	0.0000
2024-01-24	0.0303	0.0333	0.9364	0.0190
2024-01-25	0.3717	0.0092	0.6191	0.0124
2024-01-26	0.8686	0.0131	0.1183	0.0000
2024-01-29	0.1550	0.7305	0.1146	-0.0139
2024-01-30	0.1025	0.7667	0.1308	-0.0375
2024-01-31	0.0136	0.9054	0.0810	0.0668
2024-02-01	0.0337	0.0221	0.9442	-0.0055
2024-02-02	0.0107	0.0003	0.9889	-0.0018
2024-02-05	0.0086	0.0002	0.9912	0.0137
2024-02-06	0.0343	0.0055	0.9601	0.0556
2024-02-07	0.5976	0.0357	0.3668	0.0125
2024-02-08	0.6437	0.0349	0.3213	-0.0011

表 2-1 – Continued from previous page

Trddt	Moutai_Weight	Ningde_Weight	Yunnan_Weight	Portfolio_Return
2024-02-19	0.3943	0.3809	0.2247	-0.0068
2024-02-20	0.4661	0.2273	0.3065	-0.0087
2024-02-21	0.0594	0.8583	0.0823	0.0172
2024-02-22	0.3027	0.3740	0.3233	-0.0018
2024-02-23	0.1973	0.4451	0.3576	-0.0020
2024-02-26	0.2978	0.4974	0.2047	-0.0183
2024-02-27	0.3418	0.3276	0.3306	0.0044
2024-02-28	0.6216	0.0947	0.2837	-0.0047
2024-02-29	0.2383	0.6472	0.1145	0.0295
2024-03-01	0.4170	0.0770	0.5061	0.0047
2024-03-04	0.6039	0.0380	0.3581	-0.0021
2024-03-05	0.2773	0.4886	0.2341	0.0083
2024-03-06	0.4591	0.2504	0.2905	-0.0071
2024-03-07	0.6506	0.1239	0.2254	-0.0030
2024-03-08	0.6372	0.0999	0.2629	-0.0011
2024-03-11	0.4683	0.2532	0.2784	0.0429
2024-03-12	0.6851	0.0683	0.2466	0.0244
2024-03-13	0.2462	0.3227	0.4311	-0.0050
2024-03-14	0.4281	0.0235	0.5483	-0.0085
2024-03-15	0.1495	0.0695	0.7810	0.0008
2024-03-18	0.0076	0.9783	0.0141	0.0538
2024-03-19	0.2867	0.4648	0.2485	-0.0107
2024-03-20	0.1223	0.6034	0.2743	0.0072
2024-03-21	0.2139	0.0216	0.7644	-0.0040
2024-03-22	0.2515	0.4899	0.2586	-0.0100
2024-03-25	0.4447	0.1392	0.4160	-0.0021
2024-03-26	0.4861	0.0615	0.4524	0.0077
2024-03-27	0.4126	0.2712	0.3162	-0.0109
2024-03-28	0.3460	0.3145	0.3395	0.0029
2024-03-29	0.6382	0.0315	0.3303	0.0027
2024-04-01	0.6389	0.0530	0.3082	0.0087

表 2-1 – Continued from previous page

Trddt	Moutai_Weight	Ningde_Weight	Yunnan_Weight	Portfolio_Return
2024-04-02	0.4834	0.3847	0.1318	-0.0003
2024-04-03	0.3702	0.2000	0.4298	0.0078
2024-04-08	0.3375	0.2970	0.3656	-0.0145
2024-04-09	0.2835	0.3650	0.3515	0.0008
2024-04-10	0.3601	0.4375	0.2023	-0.0121
2024-04-11	0.4890	0.2229	0.2881	0.0004
2024-04-12	0.3613	0.4028	0.2359	-0.0123
2024-04-15	0.5469	0.1996	0.2535	0.0281
2024-04-16	0.5440	0.0816	0.3744	0.0103
2024-04-17	0.6235	0.1364	0.2401	0.0105
2024-04-18	0.6281	0.0482	0.3238	0.0009
2024-04-19	0.5580	0.0186	0.4234	-0.0031
2024-04-22	0.2100	0.5738	0.2163	0.0020
2024-04-23	0.3120	0.1143	0.5736	-0.0024
2024-04-24	0.3905	0.3168	0.2927	-0.0027
2024-04-25	0.6321	0.0371	0.3308	-0.0003
2024-04-26	0.3411	0.4889	0.1703	0.0167
2024-04-29	0.4485	0.2858	0.2658	0.0178
2024-04-30	0.4588	0.1920	0.3493	0.0016
2024-05-06	0.4974	0.3190	0.1836	0.0236
2024-05-07	0.3176	0.4244	0.2580	-0.0005
2024-05-08	0.1786	0.6238	0.1976	-0.0119
2024-05-09	0.3295	0.2789	0.3916	0.0042
2024-05-10	0.4197	0.3395	0.2408	-0.0263
2024-05-13	0.4267	0.1007	0.4726	0.0004
2024-05-14	0.7235	0.1343	0.1422	0.0021
2024-05-15	0.6588	0.1185	0.2227	-0.0042
2024-05-16	0.6438	0.1114	0.2448	0.0013
2024-05-17	0.5760	0.0443	0.3797	0.0048
2024-05-20	0.4406	0.0123	0.5471	-0.0046
2024-05-21	0.6906	0.1038	0.2056	-0.0018

表 2-1 – *Continued from previous page*

Trddt	Moutai_Weight	Ningde_Weight	Yunnan_Weight	Portfolio_Return
2024-05-22	0.2950	0.0599	0.6451	-0.0083
2024-05-23	0.3089	0.5845	0.1066	-0.0035
2024-05-24	0.4850	0.3169	0.1981	-0.0099
2024-05-27	0.4591	0.1696	0.3713	0.0004
2024-05-28	0.6199	0.1916	0.1885	-0.0050
2024-05-29	0.4720	0.1928	0.3352	0.0008
2024-05-30	0.3081	0.5209	0.1710	-0.0048
2024-05-31	0.4652	0.1343	0.4006	-0.0032
2024-06-03	0.5198	0.2800	0.2002	0.0040
2024-06-04	0.6737	0.0220	0.3043	0.0117
2024-06-05	0.0958	0.8244	0.0798	-0.0081
2024-06-06	0.4810	0.2463	0.2727	0.0001
2024-06-07	0.2849	0.5635	0.1516	-0.0348
2024-06-11	0.1134	0.8056	0.0810	-0.0220
2024-06-12	0.2437	0.6677	0.0886	-0.0121
2024-06-13	0.4414	0.1382	0.4205	-0.0156
2024-06-14	0.0233	0.9258	0.0509	-0.0006
2024-06-17	0.2532	0.3341	0.4127	0.0028
2024-06-18	0.1678	0.0996	0.7326	0.0026
2024-06-19	0.2881	0.0834	0.6285	-0.0094
2024-06-20	0.2127	0.3248	0.4625	-0.0105
2024-06-21	0.1815	0.2333	0.5851	-0.0070
2024-06-24	0.2125	0.4950	0.2925	0.0040
2024-06-25	0.1687	0.0866	0.7447	0.0030

后面的部分请详见附件 csv 文件 “portfolio_returns.csv”。