OSAS:

Developing a Weekly Calendar-Spread Arbitrage Strategy on CSI 500 Stock Index Futures

Ke Deng

University of Auckland, Auckland, New Zealand

kden989@aucklanduni.ac.nz

Table of Contents

1 Business Understanding

1.1 Business Objectives Identification

The COVID-19 pandemic has had a profound impact on the global economy, and China has also been severely affected. With lockdowns and strict control measures impeding economic activities, the global economy has entered into a recession. In such a scenario, the investment market is filled with uncertainty, and many investors hold a pessimistic view of future market trends. Under the influence of such uncertainty and market pessimism, equity assets have performed poorly, leading to exacerbated losses for investors in equity investments. All these factors have prompted investors to turn to lower-risk and more stable investment strategies, such as stock index futures calendar-spread arbitrage strategy. Kong, H., and Wang, H. conducted a study on the Korean KOSP1200 index futures, focusing on the daily closing prices of the current quarter and next quarter contracts. They established a cointegration model and employed the AR(2)-GARCH(1,1) model for arbitrage purposes. The results indicated that, with appropriate arbitrage trading strategies, the in-sample data yielded an annualized return of approximately 30%. [1]

The stock index futures calendar-spread arbitrage strategy is a strategy that aims to profit from price discrepancies, and it offers relatively lower risk compared to direct investments in stocks or other equity assets. Additionally, in periods of high market volatility, the calendar-spread arbitrage strategy may present greater performance. An increasing number of institutional investors are adopting this strategy, reflecting changes in the market environment and the growing demand for risk management and improved capital efficiency. By establishing hedging positions between futures contracts with different expiration dates, the calendar-spread arbitrage strategy effectively reduces investment risk and enhances capital efficiency.

Taking the CSI 500 Stock Index Futures as an example, there are four types of contracts traded simultaneously: the contract expiring in the current month, the contract expiring in the next month, the contract expiring in the current quarter, and the contract expiring in the next quarter. This strategy has limited risk exposure, and historical data reveals significant and stable statistical characteristics in the price differentials among the four types of stock index futures contracts, suggesting the presence of a certain regularity. Consequently, during periods of market instability, calendar-spread arbitrage strategy (combinations of long and short positions) can serve as a risk-controlled cash management tool, offering returns that surpass deposit interest rates and satisfying the market demand for stable-yield assets.

Considering the current market conditions and trend of stock index futures in China, there still exists a certain profit potential in low-frequency medium to long-term calendar-spread arbitrage. A low-frequency arbitrage strategy based on predicting the future price discrepancies of calendar spreads exhibits higher tolerance for trading costs because it primarily focuses on medium to long-term price changes rather than short-term (e.g., i ntra-minute) price fluctuations. Therefore, even with higher trading costs, as long as the expected profits are significant, there remains room for profitability. Additionally, this approach provides valuable insights for roll-over strategies.

1.1.1 Business Objectives

In conclusion, the aim of constructing a weekly calendar-spread arbitrage strategy on CSI 500 Index Future is to achieve the following business objectives:

- Return increase: Implementing the calendar-spread arbitrage strategy aims to capitalize on price discrepancies among different futures contracts, thereby enhancing the overall portfolio returns, especially in a market facing a predominantly pessimistic sentiment.

- Risk management: Through establishing hedging positions in the stock index futures market, the strategy effectively manages and reduces investment risks.

- Capital management: The calendar-spread arbitrage strategy can also serve as a tool for capital management, particularly in times of market volatility. Therefore, optimizing capital utilization and enhancing its efficiency are also among the key business objectives.

1.1.2 Business Success Criteria

From a business perspective, the research will be considered successful if the following objectives are achieved:

- The annualized return of the selected strategy is no less than the median of the annual return of each CSI 500 Stock Index Futures calendar spread arbitrage strategies.
- The selected strategy 's Sharpe ratio is greater than 1.

1.2 Situation Assessment

1.2.1 Data Resources

The primary data source is from the public market, including data from data service platform providers like Wind. Wind is a leading financial data and service-providing platform, similar to Bloomberg but more focused on the Chinese financial market. Wind offers a wide range of services, including financial databases, risk management and investment management systems, financial engineering, and Internet financial information. The data provided by Wind has been widely recognized for its extensive coverage, accuracy, and consistency. In terms of timeliness, Wind provides real-time and historical data with an efficient data update mechanism. Moreover, for certain required original data in this research, such as private equity data like net asset value of hedge fund strategies, if such data is not available on the Wind platform, will be obtained from the internal network of my company.

1.2.2 Hardware Resource

Sufficient computing power is needed to run data mining and predictive models, and relevant software for data mining and analysis, such as Python and SPSS Modeler, Excel will be used.

1.2.3 Assumptions

Predicting the return of stock index futures portfolio, or in other words, predicting the price trends of stock index futures is based on two fundamental assumptions:

- The financial market is efficient, meaning that market prices reflect all available information.
- Changes in stock index futures prices can be predicted using historical data and other relevant economic indicators.
- In the trading of stock index futures, when four contracts are concurrently available, six distinct calendar spread strategies can be constructed. Here, we temporarily assume constructing long positions in forward contracts and short positions in near-term contracts to establish calendar spread strategy of long and short positions.
- Not taking into account the impact of transaction cost shocks.

1.2.4 Risks and Contingencies

It is essential to acknowledge and address potential risks and contingencies to ensure the robustness of our study. The following factors are critical to consider:

- Data acquisition risk: There is a risk that Wind may not provide all the required data or that some data may be of lower quality than expected. In such cases, we can search for the target data in other data providers, such as Ichoice (another financial data service platform in China) etc.

- Market Risk: This includes macroeconomic risks and investor structure change risks. For those risks, such as global economic downturns or inflation, new research and analysis of relevant macroeconomic conditions will be conducted, and necessary adjustments to the models will be made based on the new research findings, ensuring alignment of the investment strategy with the current financial market.

- Policy Risk: This risk stems from changes in market policies, including adjustments to monetary policy, fiscal policy, or trading policy. Any announcements from policy-making institutions will be closely monitored and studied, and necessary adjustments to the models will be made based on these new market policy factors.

- Model Risk: This refers to the risk of historical statistical characteristics not being consistent with future market asset trends. Due to the volatility of financial markets, we will regularly review the assumptions of the model to ensure assumptions are still valid, and consider using different statistical models or methods for predictions when necessary to adapt to new market conditions.

1.3 Data Mining Objectives

The concurrent existence of four distinct contracts in stock index futures forms six spread combinations. By establishing cross-period combinations through a Calendar spread strategy involving long positions in forward contracts and short positions in near-term contracts, this constitutes a predictive challenge. We aim to predict which of these combinations will yield the highest return, or at least surpass the market's median return for all six spread combinations, within the next five trading days. From a practical standpoint, we will select arbitrage combinations determined by the model for investment in the market, seeking returns with relatively objective gains and minimized drawdown.

1.3.1 Data Mining Goals

- Prediction of Strategy Return: Forecasting the cumulative return of six stock index futures cross-period combinations over the next 5 trading days. Validating the actual market returns of the model-selected combination to ensure that the selected combination, based on predicted high returns, either meets or surpasses the median level among the six combinations." Following the forecasting of returns for these six combinations, our aim is to select the combination with the highest return as our rebalancing strategy for the next five days.

- Rebalancing Day Strategy Selection: On each rebalancing day, determine the optimal arbitrage combination strategy based on the day's data using the model. Validate the model-selected strategy using out-of-sample data and monitor the returns curve to ensure that the returns meet business objectives.

- Model Algorithm Comparison: Train models using two different algorithms and compare their predictive performance. Choose two distinct algorithms such as random forest and neural networks. Train each model using the same training dataset. Evaluate each model's predictive performance on the test dataset. Compare and analyze the predictive performance of these two algorithms and select the model with the best performance for practical implementation.

1.3.2 Data Mining Success Criteria

The study's data mining success criteria involve obtaining optimized models and parameter settings that output the most accurate predictive results. At a minimum, we expect the predictive accuracy to achieve 50%, indicating that the model-selected combination has a probability of at least 50% to meet or exceed the median performance among the six combinations. Ultimately, the model will guide the implementation of weekly rebalancing to maximize returns.

1.4 Project Plan

The planning research timeline is presented in the table below.

Phase	Time	Task
Business understanding	Week1	D1-D3: Select the research topic and define the research objectives. D4-D7: Gather relevant literature and research materials to gain a deeper understanding of the project objectives and establish evaluation criteria and standards.
Data understanding	Week2	D8-D9: Collect preliminary data and assess its data quality. D9-D10: Conduct preliminary exploratory data analysis to evaluate the need for additional data or data types. D11: Gather more data based on the potential more data requirements.
Data preparation	Week2-	D12-D13: 1. Perform data cleaning, including handling missing values, outliers, and duplicates. 2. Select data subsets as samples. (Given the relatively clean and reliable data provided by the Wind database, this step should not be much complicated)
Data Transformation	Week3	D14-D16: Conduct data preprocessing, particularly for the predictors (x) used to forecast the target variable (y). Many critical indicators, such as market returns, volatility, and market style and etc, require lots of transformations from raw data. D17-D19: 1. Partition the data into training and testing sets. 2. Conduct feature engineering, including feature selection and extraction, to reduce dimensionality for the selected variables obtained during the preliminary stage.
Data-mining Method(s) Selection& Data-mining Algorithm(s) Selection	Week3-week4	D20-D21: Compare various data mining methods, analyze their strengths and weaknesses, and select the most suitable method according to the project's requirements. D21-D22: Based on the chosen data mining method, compare the performance of different algorithms and select the optimal algorithm.
Data Mining	Week4	D22-D27: Perform machine learning model training, and parameter optimization; evaluate model performance using the testing set.
Interpretation& Evaluation	Week4-week 5	D28-D32: 1. Interpret model results, analyze its strengths and weaknesses, and propose improvement strategies. 2. Based on the model evaluation results, make necessary model adjustments and optimizations.

Table 1: Project Timeline

2 Data Understanding

2.1 Initial Data Collection

2.1.1 Data Source

The primary data source is from the public market, including data from data service platform providers like Wind. Wind is a leading financial data and service-providing platform, similar to Bloomberg but more focused on the Chinese financial market. Moreover, for certain required original data in this research, such as private equity data like net asset value of hedge fund strategies, if such data is not available on the Wind platform, will be obtained from the internal network of my company.

2.1.2 Attribute Selection

In order to forecast the returns for stock index futures combinations with calendar spread strategy, the factors that might influence the prices of stock index futures will be analyzed.

The short-term and long-term risk-return characteristics of the China A shares market affect the basis spread of stock index futures. In the short term, when the stock market experiences high volatility, especially when unexpected events drive large increases or decreases in broad basis spreads, the speculative activity on stock index futures market tends to surge, thus rising volume-to-open interest- ratio of contracts.

In the long term, the risk-return characteristics of the China A shares market also influence Alpha returns and neutral strategy returns, thereby affecting the hedging, speculative, and arbitrage returns of stock index futures, which indirectly impacts the basis spread of stock index futures.

Open interest can reflect the trading structure of stock index futures, potentially predicting the spread of stock index futures.

Based on the aforementioned analysis of factors influencing stock index futures prices, we will be able to discern more potentially impactful data indicators from the Wind database. Initially, the following indicators have been selected as raw data, forming the foundation of the original dataset composed of 12 sub datasets.

The selected raw data comprises:

- Closing prices and turnover rates of various stock indices.
- Trading volume of the CSI 500 Index.
- Closing prices, trading volume, and open interest of CSI 500 stock index futures contracts expiring in the current month, next month, current quarter, and next quarter.
- Long and short positions of the top N brokers in CSI 500 stock index futures. (N=5,10,20)
- Net asset values of both public and private index-enhanced strategy funds based on the CSI 500.
- Net asset values of both public and private neutral strategy funds.
- Trading codes, contract expiration date, date days to maturity, and the impact points of dividend of the current month, next month, current quarter, and next quarter CSI 500 stock index futures contracts.

2.1.3 Exclusion of Irrelevant Attributes

Initially, we will manually select the relevant attributes and exclude redundant measurements or indicators that can be expressed using other variables. We will then utilize feature selections method identify variables that significantly impact each individual target variable.

2.1.4 Handling High-Dimensional Data

When it comes to selecting the modeling method, one of the primary challenges lies in dealing with a vast number of attributes or factors. The determinants influencing stock index futures prices are diverse, and our model will also involve generating numerous new factors from the raw data. This expansion could potentially lead to increased model complexity. To address this issue effectively, we will employ feature selection methods to only select the variables that exert a substantial influence on target variables. Additionally, we will leverage Principal Component Analysis (PCA) to reduce the dimensionality of the data. This approach allows us to retain vital information while mitigating computational challenges associated with an overwhelming number of attributes.

2.2 Data Description

The raw dataset comprises 79,920 records from April 16, 2015, to June 30, 2023, distributed across 1,998 rows and 46 columns. (one column is trading date, which is not include in the table below)

Table 2 provides a description of the fields, explanation, Measurement, units and numbers of the variables in the dataset:

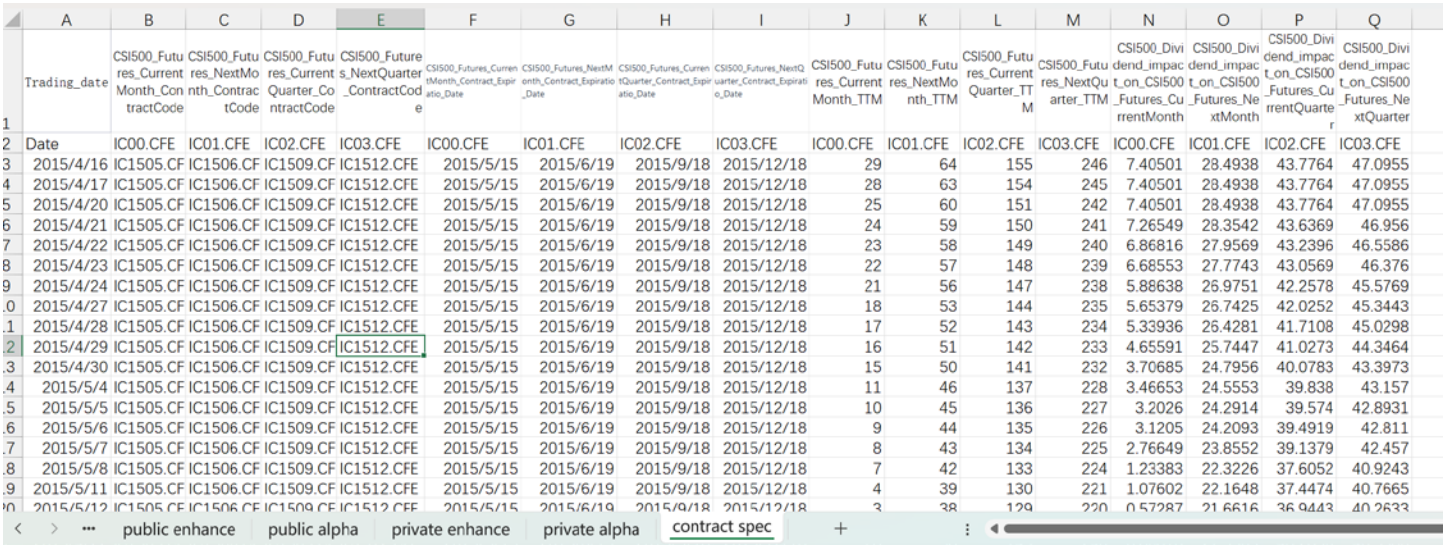
Dataset Name		Field	Field explanation	Measurement	Data unit	Numbers
index_close	1	CP_CSI 500	The closing price of the CSI 500 index.	Continuous(Numeric)	point(1point=300 CNY)	1198*5
	2	CP_CSI 300	The closing price of the CSI 300 index.	Continuous(Numeric)	point(1point=300 CNY)	
	3	CP_SSE 50	The closing price of the SSE 50 index.	Continuous(Numeric)	point(1point=300 CNY)	
	4	CP_CSI 1000	The closing price of the CSI 1000 index.	Continuous(Numeric)	point(1point=300 CNY)	
	5	CP_Wind All-A	The closing price of the Wind All-A index.	Continuous(Numeric)	point(1point=300 CNY)	
index_turnover	6	Turnover_CSI 500	Turnover rate of the CSI 500 Index	Continuous(Numeric)	percentage	1198*5
	7	Turnover_CSI 300	Turnover rate of the CSI 300 Index	Continuous(Numeric)	percentage	
	8	Turnover_SSE 50	Turnover rate of the SSE 50 index.	Continuous(Numeric)	percentage	
	9	Turnover_CSI 1000	Turnover rate of theCSI 1000 index.	Continuous(Numeric)	percentage	
	10	Turnover_Wind All-A	Turnover rate of the Wind All-A index.	Continuous(Numeric)	percentage	
index_volume	11	TradingVol_CSI 500	The trading volume of the CSI 500 index.	Continuous(Numeric)	shares	1198*1
ic_close	12	CSI500_Futures_CurrentMonth_CP_Continuous	The continuous closing price of CSI 500 futures expiring in the current month	Continuous(Numeric)	point(1point=300 CNY)	1198*4
	13	CSI500_Futures_NextMonth_CP_Continuous	The continuous closing price of CSI 500 futures expiring in the next month	Continuous(Numeric)	point(1point=300 CNY)	

	14	CSI500_Futures_CurrentQuarter_CP_Continuous	The continuous closing price of CSI 500 futures expiring in the current quarter	Continuous(Numeric)	point(1point=300 CNY)	
	15	CSI500_Futures_NextQuarter_CP_Continuous	The continuous closing price of CSI 500 futures expiring in the next quarter	Continuous(Numeric)	point(1point=300 CNY)	
ic_volume	16	CSI500_Futures_CurrentMonth_TradingVol_Continuous	The continuous trading volume of CSI 500 futures expiring in the current month	Continuous(Numeric)	contracts	1198*4
	17	CSI500_Futures_NextMonth_TradingVol_Continuous	The continuous trading volume of the CSI 500 futures expiring in the next month	Continuous(Numeric)	contracts	
	18	CSI500_Futures_CurrentQuarter_TradingVol_Continuous	The continuous trading volume of the CSI 500 futures expiring in the current quarter	Continuous(Numeric)	contracts	
	19	CSI500_Futures_NextQuarter_TradingVol_Continuous	The continuous trading volume of the CSI 500 futures expiring in the next quarter	Continuous(Numeric)	contracts	
ic_oi	20	CSI500_Futures_CurrentMonth_OI_Continuous	Continuous open interest for the CSI500 futures contracts expiring in the current month	Continuous(Numeric)	contracts	1198*4
	21	CSI500_Futures_NextMonth_OI_Continuous	Continuous open interest for the CSI500 futures contracts expiring in the next month	Continuous(Numeric)	contracts	
	22	CSI500_Futures_CurrentQuarter_Ol_Continuous	Continuous open interest for the CSI500 futures contracts expiring in the current quarter	Continuous(Numeric)	contracts	
	23	CSI500_Futures_NextQuarter_OI_Continuous	Continuous open interest for the CSI500 futures contracts expiring for the next quarter	Continuous(Numeric)	contracts	
broker_oi	24	CSI500_futures_broker_top5_long_positions	Top 5 broker long positions in CSI 500 index futures	Continuous(Numeric)	contracts	1198*6
	25	CSI500_futures_broker_top5_short_positions	Top 5 broker short positions in CSI 500 index futures	Continuous(Numeric)	contracts	
	26	CSI500_futures_broker_top10_long_positions	Top 10 broker long positions in CSI 500 index futures	Continuous(Numeric)	contracts	
	27	CSI500_futures_broker_top10_short_positions	Top 10 broker short positions in CSI 500 index futures	Continuous(Numeric)	contracts	
	28	CSI500_futures_broker_top20_long_positions	Top 20 broker long positions in CSI 502 index futures	Continuous(Numeric)	contracts	
	29	CSI500_futures_broker_top20_short_positions	Top 20 broker short positions in CSI 502 index futures	Continuous(Numeric)	contracts	
public_enhance	30	Average_NAV_Per_Unit_CSI500_Public_Index_Enhanced_Fund	Average net asset value per unit for the CSI500 public index-enhanced fund.	Continuous(Numeric)	CNY	1198*1
public_alpha	31	Average_NAV_Per_Unit_Public_Market_Neutral_Strategy_Fund	Average net asset value per unit for the public market neutral strategy fund.	Continuous(Numeric)	CNY	1198*1

private_enhance	32	Average_NAV_Per_Unit_CSI500_Private_Index_Enhanced_Fund	Average net asset value per unit for the CSI500 private index-enhanced fund.	Continuous(Numeric)	CNY	417*1
private_alpha	33	Average_NAV_Per_Unit_Private_Market_Neutral_Strategy_Fund	Average net asset value per unit for the private market neutral strategy fund.	Continuous(Numeric)	CNY	417*1
contract_spec	34	CSI500_Futures_CurrentMonth_ContractCode	The contract code for the CSI500 futures expiring in the current month	Nominal	N/A	1198*12
	35	CSI500_Futures_NextMonth_ContractCode	The contract code for the CSI500 futures expiring in the next month	Nominal	N/A	
	36	CSI500_Futures_CurrentQuarter_ContractCode	The contract code for the CSI500 futures expiring in the current quarter	Nominal	N/A	
	37	CSI500_Futures_NextQuarter_ContractCode	The contract code for the CSI500 futures expiring in the next quarter	Nominal	N/A	
	38	CSI500_Futures_CurrentMonth_Contract_Expiratio_Date	The contract expiration date for the CSI500 futures expiring in the current month	Nominal	N/A	
	39	CSI500_Futures_NextMonth_Contract_Expiratio_Date	The contract expiration date for the CSI500 futures expiring in the next month	Nominal	N/A	
	40	CSI500_Futures_CurrentQuarter_Contract_Expiratio_Date	The contract expiration date for the CSI500 futures expiring in the current quarter	Nominal	N/A	
	41	CSI500_Futures_NextQuarter_Contract_Expiratio_Date	The contract expiration date for the CSI500 futures expiring in the next quarter	Nominal	N/A	
	42	CSI500_Futures_CurrentMonth_TTM	Time to maturity in days for the CSI500 futures contracts expiring in the current month	Continuous(Numeric)	days	
	43	CSI500_Futures_NextMonth_TTM	Time to maturity in days for the CSI500 futures contracts expiring in the next month	Continuous(Numeric)	days	
	44	CSI500_Futures_CurrentQuarter_TTM	Time to maturity in days for the CSI500 futures contracts expiring in the current quarter	Continuous(Numeric)	days	
	45	CSI500_Futures_NextQuarter_TTM	Time to maturity in days for the CSI500 futures contracts expiring in the next quarter	Continuous(Numeric)	days	
	46	CSI500_Dividend_impact_on_CSI500_Futures_CurrentMonth	The impact of dividends from all stocks of the CSI 500 Index on the price of the CSI 500 futures contracts expiring in the current month.	Continuous(Numeric)	point(1point=300 CNY)	
	47	CSI500_Dividend_impact_on_CSI500_Futures_NextMonth	The impact of dividends from all stocks of the CSI 500 Index on the price of the CSI 500 futures contracts expiring in the next month.	Continuous(Numeric)	point(1point=300 CNY)	
	48	CSI500_Dividend_impact_on_CSI500_Futures_CurrentQuarter	The impact of dividends from all stocks of the CSI 500 Index on the price of the CSI 500 futures contracts expiring in the current quarter.	Continuous(Numeric)	point(1point=300 CNY)	

	49	CSI500_Dividend_impact_on_CSI500_Futures_NextQuarter	The impact of dividends from all stocks of the CSI 500 Index on the price of the CSI 500 futures contracts expiring in the next quarter.	Continuous(Numeric)	point(1point=300 CNY)	

Table 2: Field Description



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	Trading_date	CSI500_Futures_CurrentMonth_ContractCode	CSI500_Futures_NextMonth_ContractCode	CSI500_Futures_CurrentQuarter_ContractCode	CSI500_Futures_NextQuarter_ContractCode	CSI500_Futures_CurrentMonth_Contract_Expiratio_Date	CSI500_Futures_NextMonth_Contract_Expiratio_Date	CSI500_Futures_CurrentQuarter_Contract_Expiratio_Date	CSI500_Futures_NextQuarter_Contract_Expiratio_Date	CSI500_Futures_CurrentMonth_TTM	CSI500_Futures_NextMonth_TTM	CSI500_Futures_CurrentQuarter_TTM	CSI500_Futures_NextQuarter_TTM	CSI500_Dividend_impact_on_CSI500_Futures_CurrentMonth	CSI500_Dividend_impact_on_CSI500_Futures_NextMonth	CSI500_Dividend_impact_on_CSI500_Futures_CurrentQuarter	CSI500_Dividend_impact_on_CSI500_Futures_NextQuarter
2	Date	IC00.CFE	IC01.CFE	IC02.CFE	IC03.CFE	IC00.CFE	IC01.CFE	IC02.CFE	IC03.CFE	IC00.CFE	IC01.CFE	IC02.CFE	IC03.CFE	IC00.CFE	IC01.CFE	IC02.CFE	IC03.CFE
3	2015/4/16	IC1505.CF	IC1506.CF	IC1509.CF	IC1512.CFE	2015/5/15	2015/6/19	2015/9/18	2015/12/18	29	64	155	246	7.40501	28.4938	43.7764	47.0955
4	2015/4/17	IC1505.CF	IC1506.CF	IC1509.CF	IC1512.CFE	2015/5/15	2015/6/19	2015/9/18	2015/12/18	28	63	154	245	7.40501	28.4938	43.7764	47.0955
5	2015/4/20	IC1505.CF	IC1506.CF	IC1509.CF	IC1512.CFE	2015/5/15	2015/6/19	2015/9/18	2015/12/18	25	60	151	242	7.40501	28.4938	43.7764	47.0955
6	2015/4/21	IC1505.CF	IC1506.CF	IC1509.CF	IC1512.CFE	2015/5/15	2015/6/19	2015/9/18	2015/12/18	24	59	150	241	7.26549	28.3542	43.6369	46.956
7	2015/4/22	IC1505.CF	IC1506.CF	IC1509.CF	IC1512.CFE	2015/5/15	2015/6/19	2015/9/18	2015/12/18	23	58	149	240	6.86816	27.9569	43.2396	46.5586
8	2015/4/23	IC1505.CF	IC1506.CF	IC1509.CF	IC1512.CFE	2015/5/15	2015/6/19	2015/9/18	2015/12/18	22	57	148	239	6.68553	27.7743	43.0569	46.376
9	2015/4/24	IC1505.CF	IC1506.CF	IC1509.CF	IC1512.CFE	2015/5/15	2015/6/19	2015/9/18	2015/12/18	21	56	147	238	5.88638	26.9751	42.2578	45.5769
10	2015/4/27	IC1505.CF	IC1506.CF	IC1509.CF	IC1512.CFE	2015/5/15	2015/6/19	2015/9/18	2015/12/18	18	53	144	235	5.65379	26.7425	42.0252	45.3443
11	2015/4/28	IC1505.CF	IC1506.CF	IC1509.CF	IC1512.CFE	2015/5/15	2015/6/19	2015/9/18	2015/12/18	17	52	143	234	5.33936	26.4281	41.7108	45.0298
12	2015/4/29	IC1505.CF	IC1506.CF	IC1509.CF	IC1512.CFE	2015/5/15	2015/6/19	2015/9/18	2015/12/18	16	51	142	233	4.65591	25.7447	41.0273	44.3464
13	2015/4/30	IC1505.CF	IC1506.CF	IC1509.CF	IC1512.CFE	2015/5/15	2015/6/19	2015/9/18	2015/12/18	15	50	141	232	3.70685	24.7956	40.0783	43.3973
14	2015/5/4	IC1505.CF	IC1506.CF	IC1509.CF	IC1512.CFE	2015/5/15	2015/6/19	2015/9/18	2015/12/18	11	46	137	228	3.46653	24.5553	39.838	43.157
15	2015/5/5	IC1505.CF	IC1506.CF	IC1509.CF	IC1512.CFE	2015/5/15	2015/6/19	2015/9/18	2015/12/18	10	45	136	227	3.2026	24.2914	39.574	42.8931
16	2015/5/6	IC1505.CF	IC1506.CF	IC1509.CF	IC1512.CFE	2015/5/15	2015/6/19	2015/9/18	2015/12/18	9	44	135	226	3.1205	24.2093	39.4919	42.811
17	2015/5/7	IC1505.CF	IC1506.CF	IC1509.CF	IC1512.CFE	2015/5/15	2015/6/19	2015/9/18	2015/12/18	8	43	134	225	2.76649	23.8552	39.1379	42.457
18	2015/5/8	IC1505.CF	IC1506.CF	IC1509.CF	IC1512.CFE	2015/5/15	2015/6/19	2015/9/18	2015/12/18	7	42	133	224	1.23383	22.3226	37.6052	40.9243
19	2015/5/11	IC1505.CF	IC1506.CF	IC1509.CF	IC1512.CFE	2015/5/15	2015/6/19	2015/9/18	2015/12/18	4	39	130	221	1.07602	22.1648	37.4474	40.7665
20	2015/5/12	IC1505.CF	IC1506.CF	IC1509.CF	IC1512.CFE	2015/5/15	2015/6/19	2015/9/18	2015/12/18	3	38	129	220	0.57287	21.6616	36.9443	40.2633

Figure 1: Screenshot sample of the raw dataset (Excel file)

2.3 Data Exploration

According to the practical meaning of the data, the raw data can be categorized into price, trading volume, open interest, and turnover rate. Preliminary data exploration of the raw data will be conducted to see the basic characteristics of the attribute. Python code to exploration the basic features of the raw data is shown in Figure 2.

```
# data exploration

index_close.plot()
index_close.describe()
index_close.boxplot()

ic_close.plot()
ic_close.describe()
ic_close.plot(kind='kde')

index_volume.plot()
index_volume.describe()
ic_volume.plot()
ic_volume.describe()
ic_volume.boxplot()


ic_oi.plot()
ic_oi.describe()
ic_oi.boxplot()

index_turnover.plot()
index_turnover.boxplot()

dividend.plot()
```

Figure 1: Raw data exploration in Python

2.3.1 Price

2.3.1.1 Stock index prices

Various stock indices have been chosen, including the SSE 50 Index, the CSI 300 Index, the CSI 500 Index, the CSI 1000 Index, and the WIND All-A Index. The four indices mentioned represent different styles of stocks ranging from value to growth, while the WIND All-A Index represents the performance of the entire Chinese A-share market.

The time series trends of the stock index prices are shown in Figure 3. It can be observed that in 2015, the various indices experienced significant fluctuations. This was mainly because of the widespread use of illegal margin trading in early 2015, which resulted in a frenzy of stock trading and a sharp increase in the indices. Subsequently, regulatory restrictions on margin trading caused a sharp decline in market sentiment, resulting in a rapid downturn in the stock indices from the peaks. As the regulatory framework for the Chinese stock market continued to improve and investor structure became more diversified, the market became more mature, leading to a decrease in index volatility and a more stable trend after 2016.

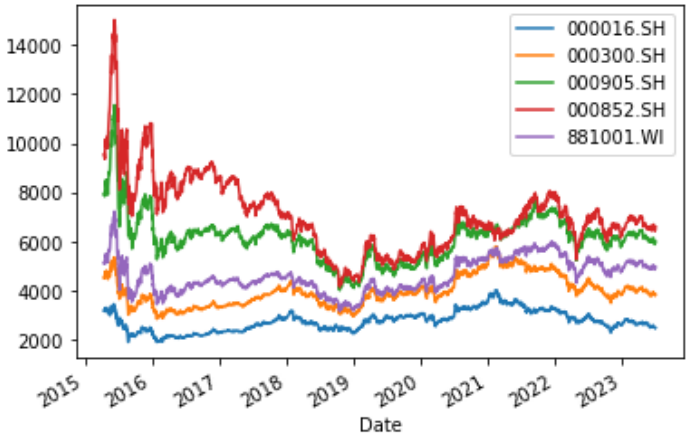


Figure 3: Trends of the price of the stock indices

The statistical data and boxplots of stock index prices are presented in Table 3 and Figure 4. Each index has a total of 1998 data points. The average price level of the indices increases with the strengthening of the growth style, accompanied by a corresponding increase in volatility. The price level and volatility of the "CSI 500" index fall between those of the "CSI 300" and "CSI 500" indices, indicating an intermediate position between the value style and growth style.

	CP_SSE 50	CP_CSI 300	CP_CSI 500	CP_CSI 1000	CP_WIND ALL-A
count	1998	1998	1998	1998	1998
mean	2770.638	3986.237	6159.951	7077.221	4635.59
std	415.4028	621.8733	962.6118	1543.416	678.2981
min	1912.721	2853.756	4018.461	4149.444	3179.535
25%	2446.437	3480.065	5685.512	6169.546	4156.73
50%	2743.014	3891.974	6223.043	6893.965	4501.631
75%	3018.022	4265.004	6511.978	7839.412	5175.707
max	4028.529	5807.719	11545.89	15006.34	7224.264

Table 3: Basic description of each stock index

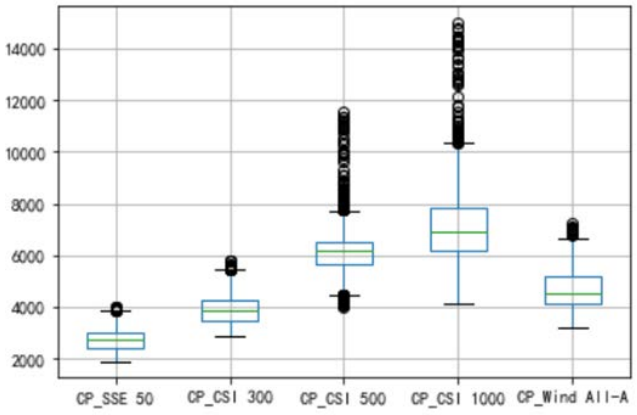


Figure 4: Boxplot of the price of each stock index

2.3.1.2 Stock Index Futures Prices

Stock index futures prices are generated through direct market trading and continuously updated during trading hours. As our focus is on the CSI 500 stock index futures, we have exclusively selected the prices of CSI 500 stock index futures contracts as the source data. The price trends of CSI 500 stock index futures are shown in Figure 5. As the price movements of stock index futures closely follow those of stock index, the prices of CSI 500 stock index futures are highly aligned with the CSI 500 index. Additionally, the price trends of contracts with different time-to-maturity (TTM) are remarkably similar.

Figure 5: Trends of the price of each CSI 500 stock index futures

The distribution indicates that most of the prices of each stock index futures are concentrated around 6000 points.

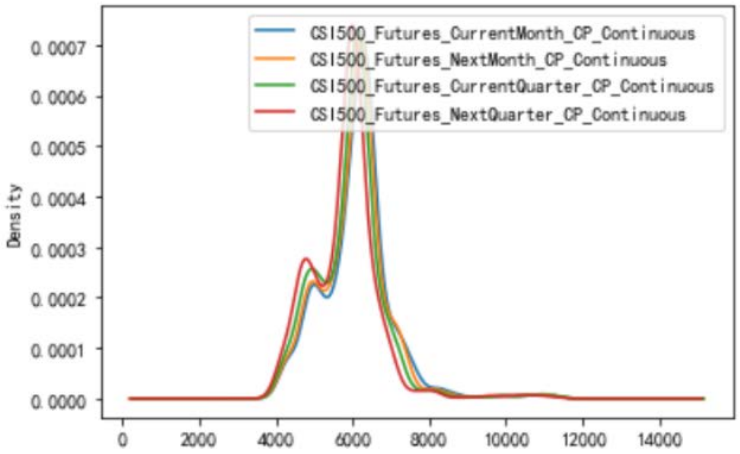


Figure 6: Distribution of the price of each CSI 500 stock index futures

2.3.2 Volume

2.3.2.1 Stock Index Volume

The trading volume of a stock index represents the total volume of all stocks included in that index.The trend of the trading volume for the CSI 500 index also has been influenced by regulatory policies, shown in Figure 7. Prior to 2018, the trading volume experienced a significant decline, but from 2019 onwards, market trading gradually recovered, leading to a partial rebound in trading volume. Generally, a substantial increase in trading volume indicates a strengthening of market investor sentiment, which can result in significant price fluctuations for both the index and futures.

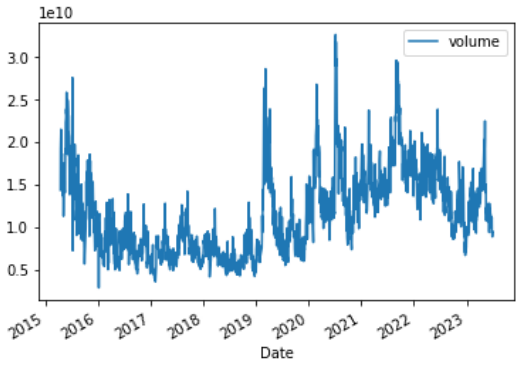


Figure 7: Trend of the volume of CSI 500 stock index

The average trading volume for the CSI 500 index is 11.5 billion shares per trading day, with the historical lowest volume recorded at 2.84 billion shares and the highest volume at 32.6 billion shares.

	TradingVol_CSI_500
count	1998
mean	1.15E+10
std	4.92E+09
min	2.84E+09
25%	7.35E+09
50%	1.08E+10
75%	1.47E+10
max	3.26E+10

Table 4: Volume of CSI 500 stock index

2.3.2.2 Stock Index Futures Volume

As mentioned above, due to regulatory policy changes in mid-2015, there was a significant decline in the trading volume. Subsequently, with the gradual relaxation of regulations and trading policies in China's derivatives market, the trading volume of stock index futures gradually increased but still remains below pre-2015 levels. This trend is shown in Figure 8.

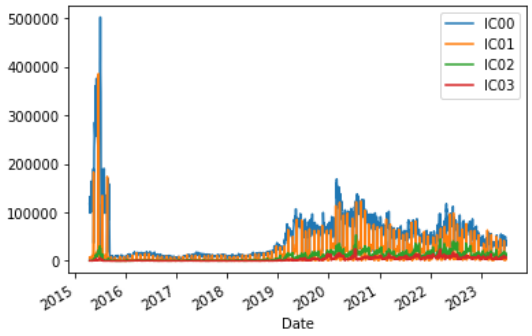


Figure 8: Trend of the volume of each CSI 500 stock index future

Based on the statistical output of the data shown in Table 5 and Figure 8, the average daily trading volume for the CSI 500 stock index futures contracts is approximately 47 thousand contracts for the current month, 12 thousand contracts for the next month, 8.58 thousand contracts for the current quarter, and 3.58 thousand contracts for the next quarter. It is evident that there are significant differences in trading activity among different contracts. Current-month contract (contract expiring in the current month) is the most active, more synchronized with the CSI 500 index volatility. The trading volume for the two following quarter contracts is relatively lower.

	CSI500_Futures_CurrentMonth_Trading Vol_Continuous	CSI500_Futures_NextMonth_Trading Vol_Continuous	CSI500_Futures_CurrentQuarter_Trading Vol_Continuous	CSI500_Futures_NextQuarter_Trading Vol_Continuous
count	1998	1998	1998	1998
mean	47490.54	12046.42	8580.61	3581.294
std	49646.12	20848.56	8560.716	3947.521
min	2184	68	73	13
25%	11596.5	1390.75	881	273
50%	40342.5	4067	7199	2143.5
75%	66945.5	14897.75	13879.75	6189
max	502523	385745	53353	24425

Table 5: Basic description of the volume of each CSI 500 stock index future

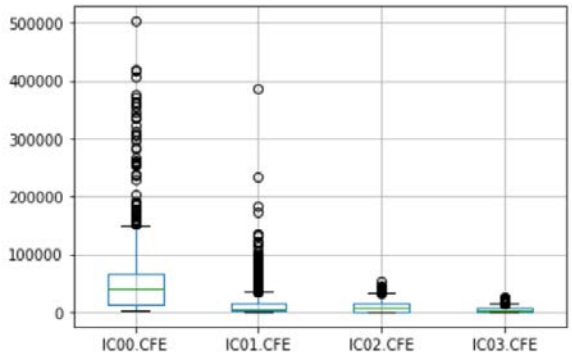


Figure 9: Boxplot of the volume of each CSI 500 stock index future

2.3.3 Open Interest

Open interest refers to the number of outstanding contracts at the end of a trading day, as disclosed by the exchange and calculated unilaterally. As stock index futures do not have the attribute of open interest, we only collect open interest data for four different contracts of CSI 500 stock index futures.

As shown in Figure 10, there was also a significant decline around mid-2015 due to regulatory policy changes regarding the trend of open interest. Subsequently, it remained at a relatively low level for an extended period until 2019 when there was a reversal in the trend. The open interest of CSI 500 stock index futures increased rapidly and has currently reached approximately three times the level of 2015.

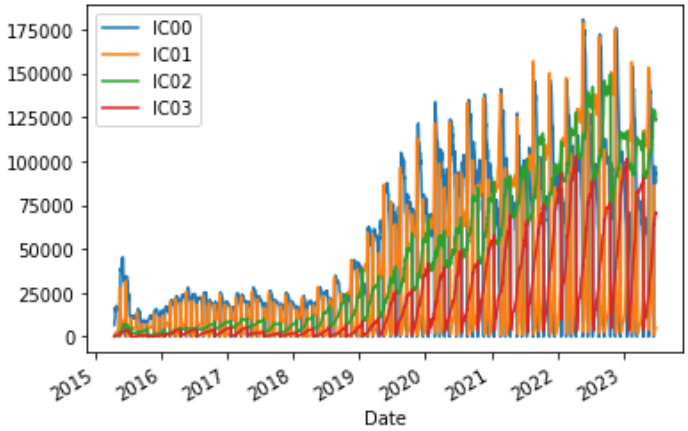


Figure 10: Trend of the OI of each CSI 500 stock index future

From Table 6, open interest among the four futures contracts shows a concentration in the IC00(current month month) and IC02 contracts (next month contract). In contrast, the other two contracts exhibit relatively lower open interest. The proportion of open interest among the four contracts can be described as approximately 4:1:4:1.

	CSI500_Futures_CurrentMonth_OI_Continuous	CSI500_Futures_NextMonth_OI_Continuous	CSI500_Futures_CurrentQuarter_OI_Continuous	CSI500_Futures_NextQuarter_OI_Continuous
count	1998	1998	1998	1998
mean	52468.83	27235.35	42457.54	20133.29
std	41569.86	38343.36	41200	25523.15
min	0	67	215	15
25%	18578.75	2804.25	5970	1878.75
50%	35235.5	9199.5	26412	5677
75%	85556.5	32460.25	78133.75	33529.5
max	180863	179287	149517	103008

Table 6: Basic description of the OI of each CSI 500 stock index future

Figure 11 further confirms the concentration of open interest in the IC00 and IC02 contracts. The underlying reason for this concentration can be attributed to the preferences of different types of traders. Short-term traders tend to favor the current-month contract due to its potential for high price volatility, aligning with their short-term trading objectives. Conversely, long-term investors show a preference for the current quarter contract as it aligns better with their long-term market expectations and investment strategies. Therefore, the next-month contract cannot fully satisfy the objectives of either short-term traders or long-term investors, leading to relatively lower interest from both groups.

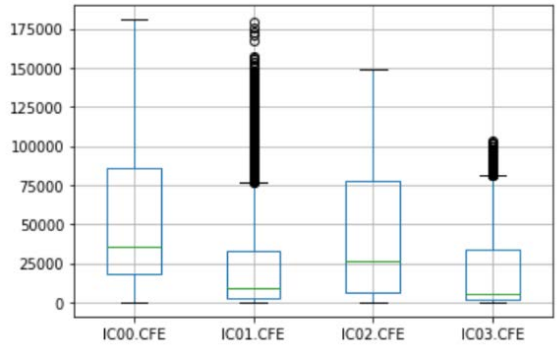


Figure 11: Boxplot of the OI of each CSI 500 stock index future

2.3.4 Turnover Rate

To gauge the market's trading activity, daily turnover rate data for various stock indices are collected. A higher turnover rate indicates more active trading.

According to the Figure 12, the turnover rates for the CSI 1000 and CSI 500 are higher compared to other indices. This can be attributed to the stock style associated with these indices. The CSI 1000 and the CSI 500 primarily consists of more mid-cap and growth stocks, which are often targeted for short-term speculation. As a result, the turnover rates for the CSI 1000 and CSI 500 are higher.

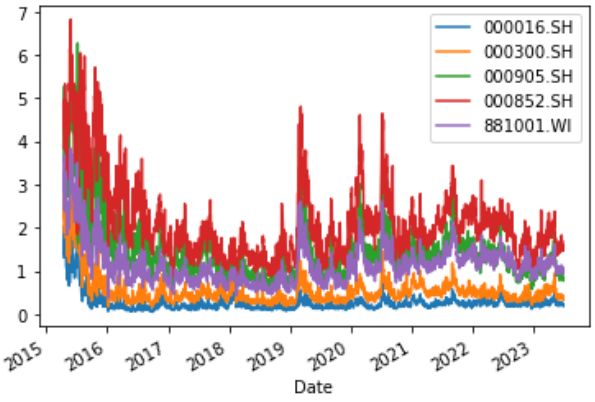


Figure 11: Trend of the turnover of stock indices

By comparing the turnover rates of different indices, it is evident that as the growth characteristics of the indices strengthen, their turnover rates also increase, which is shown in Table 7.

	Turnover_SSE 50	Turnover_CSI 300	Turnover_CSI 500	Turnover_CSI 1000	Turnover_Wind All-A
count	1998	1998	1998	1998	1998
mean	0.266885	0.539096	1.504715	2.138075	1.150774
std	0.220799	0.339784	0.75329	0.928616	0.477548
min	0.057	0.1771	0.5496	0.7041	0.3846
25%	0.1626	0.352825	1.033125	1.5238	0.8252
50%	0.2146	0.4563	1.31725	1.91325	1.0689
75%	0.2858	0.5904	1.685325	2.411175	1.31185
max	2.4073	3.0936	6.2759	6.8261	3.8362

Table 7: Basic description of the turnover of stock indices

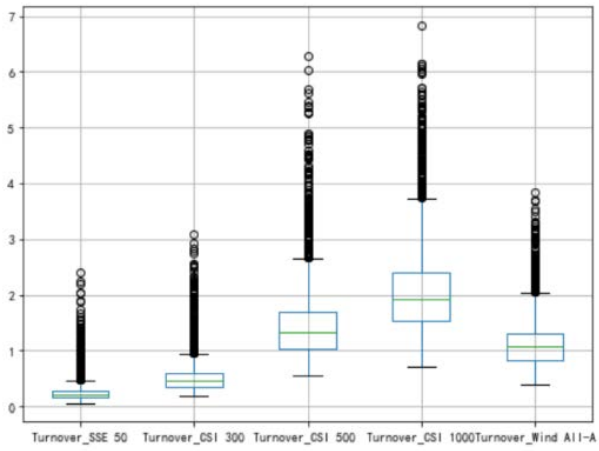


Figure 12: Boxplot of the turnover of stock indices

2.3.5 Participants in Stock Index Futures Market

2.3.5.1 Public fund

The stricter information disclosure requirements for public funds dictate that net asset values for public funds should be disclosed daily. As a result, the net asset value data for our public funds is at a daily frequency. From the net asset value trends of the two types of strategies, the index-enhanced strategy exhibits a trend similar to that of the CSI 500 Index but with higher gains. This is attributed to the index-enhanced strategy including both alpha and beta returns, mmaking its core competitive edge. The net asset value reached a peak of 1.99 in 2021. However, while achieving relative excess returns compared to the index, it was also impacted by market volatility. For instance, in mid-2015, due to regulatory policy changes as discussed earlier, there was a significant decline in the strategy's net asset value. With the gradual market recovery, the net asset value of the strategy gradually increased, followed by a decline in 2018 and further reductions due to the US-China trade war. The market's experiences with events like the Russia-Ukraine conflict in 2023 and pandemic-related lockdowns led to rapid declines in net asset values driven by pessimism. In contrast, the net asset value trend of the alpha strategy (neutral strategy) remained more stable. As it includes hedging positions in stock index futures, it effectively mitigates the impact of systematic risks, resulting in minimal drawdowns. While demonstrating limited risk, the trend remains stable in recent years. However, despite the controlled risk, the returns are relatively lower, with the net asset value reaching a maximum of 1.33.

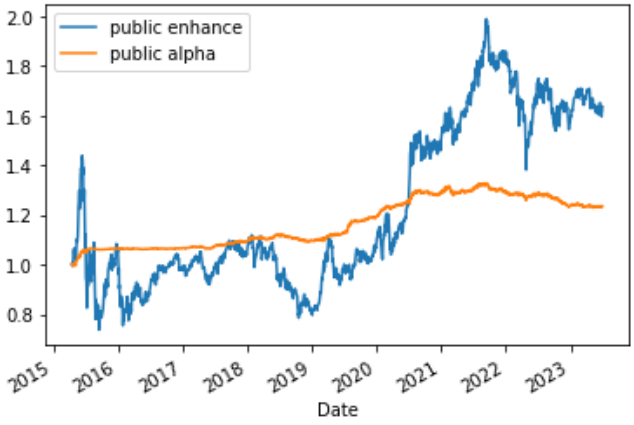


Figure 13: NAV for public fund in index-enhanced strategy vs. neutral strategy

	Average_NAV_Per_Unit_CSI500_ Public_Index_Enhanced_Fund	Average_NAV_Per_Unit_Public_ Market_Neutral_Strategy_Fund
count	1998	1998
mean	1.23	1.17
std	0.33	0.10
min	0.74	1.00
25%	0.98	1.07
50%	1.07	1.12
75%	1.59	1.28
max	1.99	1.33

Table 8: Basic description of NAV for public fund in different strategies

2.3.5.2 Private fund

Regulatory requirements for private funds are relatively more tolerant compared to public funds. Policies do not mandate the daily disclosure of net asset values for private funds, resulting in our ability to collect only weekly net asset values for private fund strategies. Analyzing private fund data reveals that private funds exhibit higher returns compared to public funds with similar strategies. The index-enhanced strategy for private funds reached a peak net asset value of 3.7, while the private alpha strategy reached a peak net asset value of 2.32. This is attributed to the generally smaller scale of private funds, allowing for more flexible operations. Additionally, the more relaxed regulatory constraints enable greater flexibility in market participation for private funds.

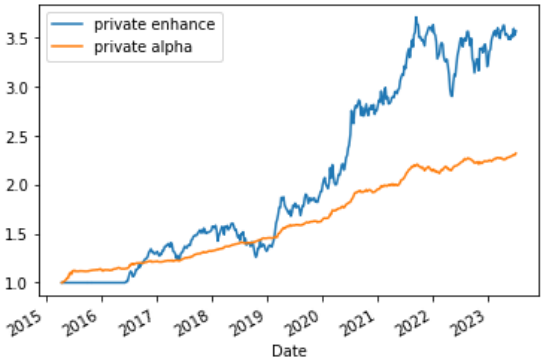


Figure 14: NAV for priavte fund in index-enhanced strategy vs. neutral strategy

	Average_NAV_Per_Unit_CSI500_Private_Index_Enhanced_Fund	Average_NAV_Per_Unit_Private_Market_Neutral_Strategy_Fund
count	417	417
mean	2.097431	1.637473
std	0.925672	0.414951
min	1	1
25%	1.33	1.2272
50%	1.7679	1.5832
75%	2.9887	2.0512
max	3.7081	2.3187

Table 9: Basic description of NAV for private fund in different strategies

2.3.6 Dividend Distribution

Most A-share listed companies distribute dividends after their annual reports, with a few distributing dividends after their semi-annual reports. Since the deadline for annual report disclosure is at the end of April, dividend distributions are primarily concentrated in the months of May to July each year. Conversely, there is minimal dividend distribution between December and January. Additionally, the dividend amounts from these companies typically increase year by year as time progresses. The time series graph depicting the impact of dividends on stock index futures expiring in the current month aligns with the common patterns of dividend distribution in A-share markets.

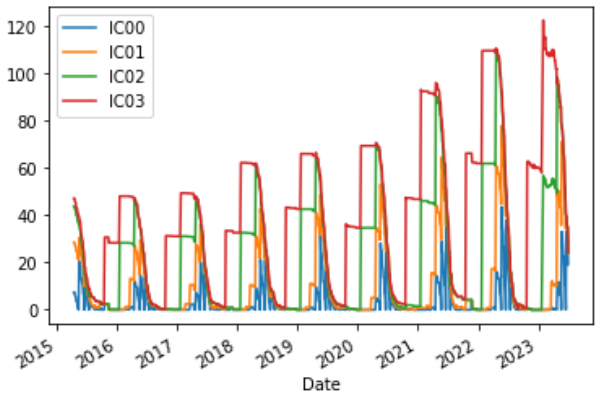


Figure 15: Dividend impact on stock index futures expiring in the current month

2.3.7 Data Quality Verification

2.3.7.1 Evaluation of Data Adequacy:

CSI 500 stock index futures have been listed and traded since April 16, 2015, and the Wind database contains complete data from this date onwards. We will select a subset of data covering several years in future chapter, excluding the influence of extreme market conditions (e.g., the 2015 Chinese stock market crash) to ensure the dataset covers a sufficiently long time period, including both bull markets and bear markets.

2.3.7.2 Data Frequency Mismatch and Missing Value:

During the process of data merging, an issue with mismatched data frequencies is encountered due to the difference in recording frequency between the private fund indicators and the rest of the data, shown in Figure16. Specifically, the private fund indicators were recorded at a weekly frequency, while the other data were recorded at a trading daily frequency. To address this problem, we aim to unify all the data to a daily frequency.

```
   ...: fund_value.isnull().sum().sort_values(ascending = False)
Out[2]:
private enhance    1581
private alpha      1578
public enhance        0
public alpha          0
dtype: int64
```

Figure 16: Identification of missing value for two variables

To achieve this, we need to process the weekly frequency data of the private fund indicators. One viable approach is to forward fill and back fill , which involves filling missing values with the most recent available observation. This method is commonly used in handling time series financial data. We will process missing value in Section 4.1.3.

2.3.7.3 Outlier or Extreme Value:

By observing the boxplot of the dataset and the outcome of data audit, the presence of outliers or extreme values has been identified (e.g., the 2015 Chinese stock market crash). However, based on a deeper understanding of the financial market and experience over time, these extreme values are not indicative of data errors, but rather extreme market conditions occurrence, which are driven by market unexpected events, macroeconomic factors, or other non-conventional situations. These extreme market conditions can lead to significant fluctuations in the prices of financial assets such as stocks and futures, thereby resulting in the appearance of extreme values in the data.

3 Data Preparation

3.1 Data Selection for Raw Dataset

Regarding the fields within the original dataset, we can safely exclude the contract codes for the four contracts and the expiration date. Contract codes are nominal and serve the purpose of acquiring specific contract-related information. Contract expiration date would also be redundant since we already have another attribute that denotes time to maturity (TTM). Consequently, these variables will not be considered in our later stages of model selection.

Other variables, in terms of economic significance, hold certain theoretical relevance to cross-period combination returns and will also be utilized in subsequent derivations of new variables. Thus, no removal of these variables will be carried out at this stage.

3.2 Data Cleaning for Raw Dataset

3.2.1 Missing Data for Raw dataset

As described in Section 2.3.7, we have identified two crucial variables within the private fund data: "Average_NAV_Per_Unit_CSI500_Private_Index_Enhanced_Fund”"Average_NAV_Per_Unit_Private_Market_Neutral_Strategy_Fund." These variables contain 1581 and 1576 missing values, respectively. This data gap primarily stems from

the disclosure regulations of private funds, which are relatively more lenient than public fund. We have access to weekly frequency data rather than daily frequency data for private fund.

To ensure data continuity and address these missing values, we imput them using nearest historical data. Code is shown in Figure 19. Following this approach, the data has been transformed to daily frequency, ensuring data uniformity and continuity.

```
# private fund-related data are on weekly basis, we have to make it consistent with all other daily frequecy data

private_enhance.index = pd.to_datetime(private_enhance.index)
private_alpha.index = pd.to_datetime(private_alpha.index)

df = [public_enhance.reset_index(),public_alpha.reset_index()]
public_fund_value = reduce(lambda left,right: pd.merge(left,right, on='Date', how='outer'),df).set_index('Date')
public_fund_value.plot()
public_fund_value.describe()

df = [private_enhance.reset_index(),private_alpha.reset_index()]
private_fund_value = reduce(lambda left,right: pd.merge(left,right, on='Date', how='outer'),df).set_index('Date')
private_fund_value.plot()
private_fund_value.describe()


df = [public_enhance.reset_index(),public_alpha.reset_index(),private_enhance.reset_index(),private_alpha.reset_index(
fund_value = reduce(lambda left,right: pd.merge(left,right, on='Date', how='outer'),df).set_index('Date')

fund_value.isnull().sum().sort_values(ascending = False)

fund_value = fund_value.fillna(method = 'ffill')
fund_value = fund_value.fillna(method = 'bfill')

fund_value.isnull().sum().sort_values(ascending = False)
```

Figure 19: Python code to fill the null using nearest historical data

After filling in the missing values, we rechecked the dataset for any remaining blanks. As depicted in Figure 20, there are now no missing values present in the dataset.

```
   ...: fund_value = fund_value.fillna(method = 'ffill')
   ...: fund_value = fund_value.fillna(method = 'bfill')
   ...:
   ...: fund_value.isnull().sum().sort_values(ascending = False)
Out[3]:
public enhance     0
public alpha       0
private enhance    0
private alpha      0
dtype: int64
```

Figure 20: Checking for missing value after filling the blank

3.3 Data Construction for Derived Dataset

3.3.1 Derived target variables

The concurrent existence of four stock future contracts results in six spread combinations. To ensure consistency between the prediction objective and business goal, we have chosen the annualized returns over the next 5 trading days from the long-short cross-period combinations of two contracts as our target variable for prediction. The formula for computing the return of the cross-period combination is as follows:

$$Y_t(C_n, C_f, k) = \frac{(P_{f,t+k} - P_{n,t+k}) - (P_{f,t} - P_{n,t})}{P_{f,t} + P_{n,t}} \times \frac{250}{k}$$

Among these, $Y_t(C_n, C_f, k)$ represents the annualized return over the next k trading days for a cross-period combination that involves taking a long position in the far-month contract C_f and a short position in the near-month contract $C_n$on day t. Because our business goal is to develop weekly rebalance strategy, here we define k=5. The values $P_{f,t}$ and $P_{n,t}$ represents the closing prices of the far-month contract and the near-month contract, respectively, on day t.

We calculate the returns for each timestamp over the next five days for the six cross-period arbitrage combinations. The code to compute returns is provided below in Figure 21.

```
futures = 'IC'
k = 5
Y = pd.DataFrame()
for near,far in [['00','01'],['00','02'],['00','03'],['01','02'],['01','03'],['02','03']]:
    spread = ic_close[futures+far] - ic_close[futures+near]   # ic_close['IC01'] - ic_close['IC00']
    Y['_'.join((near,far))] = (spread.diff(k).shift(-k) / (ic_close[futures+far] + ic_close[futures+near])) / k * 250

Y
```

Figure 21: Python code to computing returns for 6 combinations over next 5 days

The calculated outcomes are as shown in Figure 22. Notice that the last five rows are filled with NA. This is because there are no future stock index futures contract prices for the last five days, these values remain unavailable. These null values will be addressed in section 3.5.

```
...: Y
Out[4]:
              00_01     00_02     00_03     01_02     01_03     02_03
Date
2015-04-16 -0.020182 -0.322455 -0.456532 -0.303355 -0.437896 -0.134264
2015-04-17 -0.012311 -0.188065 -0.300257 -0.176660 -0.289431 -0.112717
2015-04-20  0.107753  0.141607 -0.149896  0.033502 -0.260522 -0.295794
2015-04-21 -0.085104 -0.115068 -0.168509 -0.029816 -0.083038 -0.053239
2015-04-22 -0.091932 -0.072139 -0.047156  0.020426  0.046096  0.025696
...              ...       ...       ...       ...       ...       ...
2023-06-26       NaN       NaN       NaN       NaN       NaN       NaN
2023-06-27       NaN       NaN       NaN       NaN       NaN       NaN
2023-06-28       NaN       NaN       NaN       NaN       NaN       NaN
2023-06-29       NaN       NaN       NaN       NaN       NaN       NaN
2023-06-30       NaN       NaN       NaN       NaN       NaN       NaN

[1998 rows x 6 columns]
```

Figure 22: Returns for 6 combinations over next 5 day

We examined some fundamental characteristics of the returns for the six combinations through box plots and density plots, as illustrated in Figures 23 and 24, respectively. Upon analyzing these figures, we notice that the return of "00-01" combination exhibits a more peak at the center, while "00-03" return displays a broader and flatter distribution with fatter tails. Additionally, all six combination returns have a mean value approximately centered around zero.

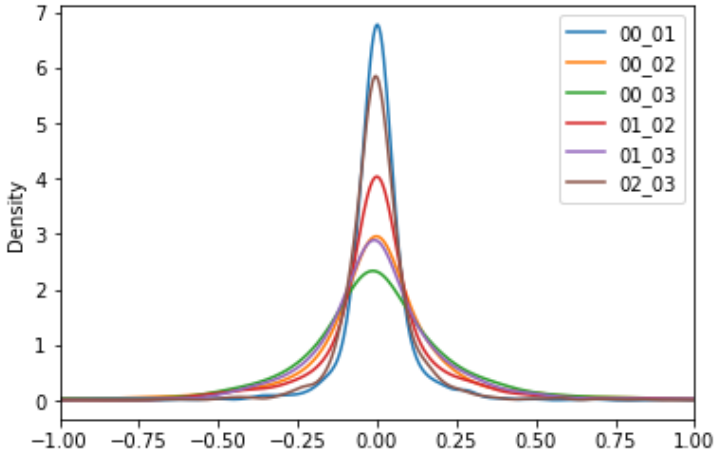


Figure 23: Density plot of the return of the six combinations

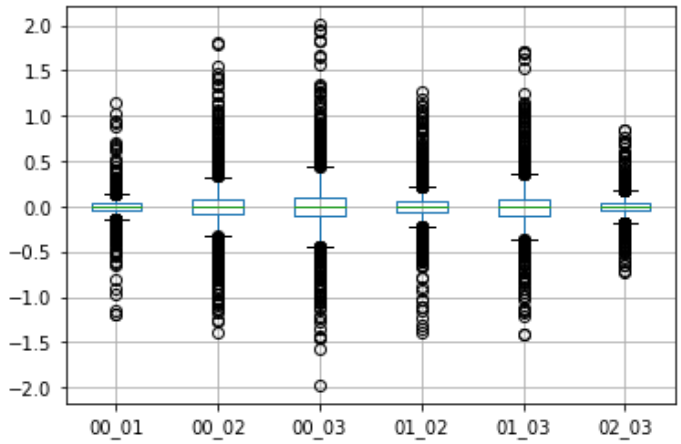


Figure 24: Box plot of the return of the six combinations

3.3.2 Derived Factors Related to Risk-Return Characteristic

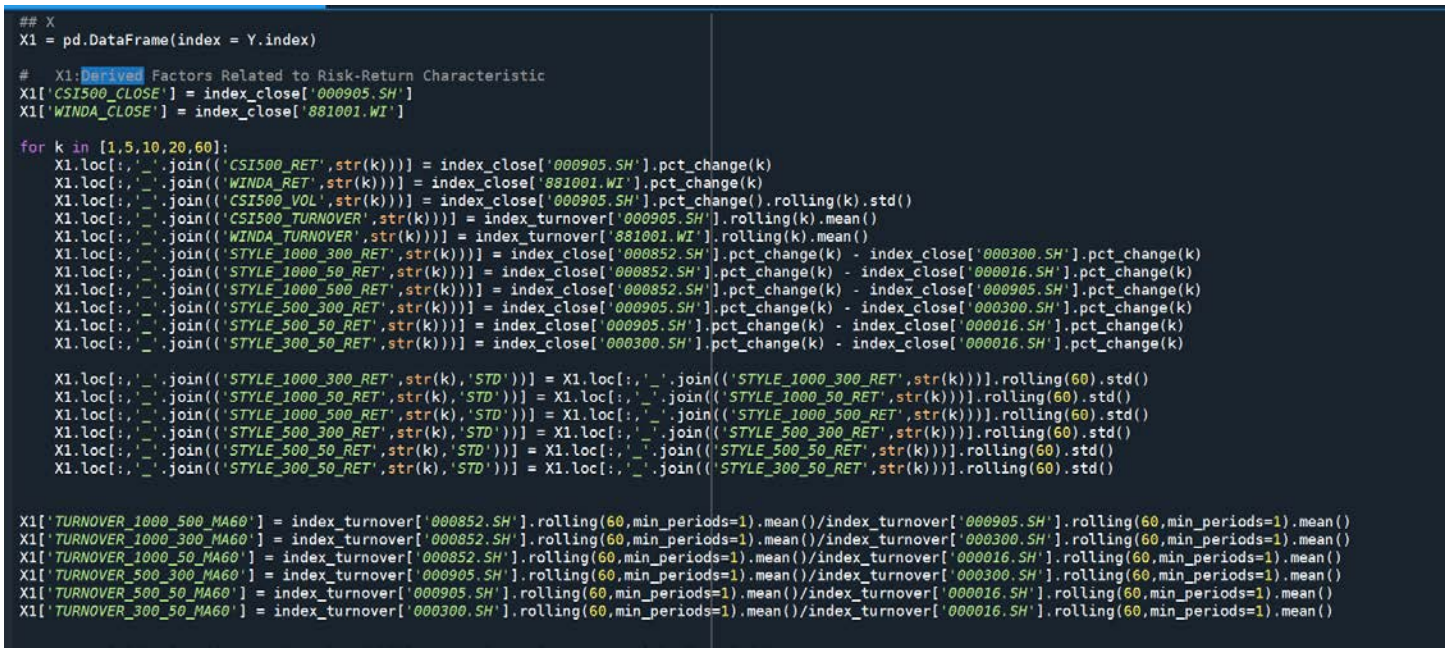
Returns and volatility measure the changes in prices of stock index futures and are important indicator for investors. By analyzing historical returns and the volatility in returns, investors can identify trends, cycles, and abnormal fluctuations, thereby providing clues for future price movements. Turnover rate reflects the level of trading activity of assets in the market. Changes in turnover rate can impact price trends, hence considering turnover rate in forecasting can capture market sentiment and investor behavior. Market style involves the relative performance of different asset categories, such as the contrast between value stocks (SSE50) and growth stocks (CSI 1000). In stock index futures forecasting, incorporating market style helps to understand the overall market positioning and investor preferences, thus enabling a more accurate prediction of price trends.

Factors related to risk-return characteristic are derived from the raw dataset, and the new derived factors are listed in the Table 10.

Categories	Variable	Variable Explanation	Parameter k Values
Returns	CSI500_CLOSE	CSI500 Closing Price	
	WINDA_CLOSE	Wind All A Closing Price	
	CSI500_RET_K	CSI500 k-day Cumulative Returns	k=1,5,10,20,60
	WINDA_RET_K	Wind All A k-day Cumulative Returns	k=1,5,10,20,60
Volatility	CSI500_VOL_K	CSI500 k-day Rolling Standard Deviation	k=1,5,10,20,60
Turnover Rate	CSI500_TURNOVER_k	CSI500 k-day Rolling average Turnover Rate	k=1,5,10,20,60
	WINDA_TURNOVER_k	Wind All A k-day Rolling average Turnover Rate	k=1,5,10,20,60
Market Style	STYLE_1000_300_RET_k	Difference in k-day Returns between CSI1000 and CSI300	k=1,5,10,20,60
	STYLE_1000_50_RET_k	Difference in k-day Returns between CSI1000 and SSE50	k=1,5,10,20,60
	STYLE_1000_500_RET_k	Difference in k-day Returns between CSI1000 and CSI500	k=1,5,10,20,60
	STYLE_500_300_RET_k	Difference in k-day Returns between CSI500 and CSI300	k=1,5,10,20,60
	STYLE_500_50_RET_k	Difference in k-day Returns between CSI500 and SSE50	k=1,5,10,20,60
	STYLE_300_50_RET_k	Difference in k-day Returns between CSI300 and SSE50	k=1,5,10,20,60
	STYLE_1000_300_RET_k_STD	60-day Rolling Standard Deviation of k-days difference in returns between CSI1000 and CSI300	k=1,5,10,20,60
	STYLE_1000_50_RET_k_STD	60-day Rolling Standard Deviation of k-days difference in returns between CSI1000 and CSI50	k=1,5,10,20,60
	STYLE_1000_500_RET_k_STD	60-day Rolling Standard Deviation of k-days difference in returns between CSI1000 and CSI500	k=1,5,10,20,60
	STYLE_500_300_RET_k_STD	60-day Rolling Standard Deviation of k-days difference in returns between CSI500 and CSI300	k=1,5,10,20,60
	STYLE_500_50_RET_k_STD	60-day Rolling Standard Deviation of k-days difference in returns between CSI500 andSSE50	k=1,5,10,20,60
	STYLE_300_50_RET_k_STD	60-day Rolling Standard Deviation of k-days difference in returns between CSI300 andSSE50	k=1,5,10,20,60
	TURNOVER_1000_500_MA60	The ratio of 60-day rolling turnover rates between the CSI 1000 and the CSI 500	
	TURNOVER_1000_300_MA60	The ratio of 60-day rolling turnover rates between the CSI 1000 and the CSI 300	
	TURNOVER_1000_50_MA60	The ratio of 60-day rolling turnover rates between the CSI 1000 and the SSE 50	
	TURNOVER_500_300_MA60	The ratio of 60-day rolling turnover rates between the CSI 500 and the CSI 300	
	TURNOVER_500_50_MA60	The ratio of 60-day rolling turnover rates between the CSI 500 and the SSE50	
	TURNOVER_300_50_MA60	The ratio of 60-day rolling turnover rates between the CSI 300 and the SSE 50	

Table 10 : Factors related to risk-return characteristic

We utilized Python to generate all these new variables, code provided in Figure 23.



```
## X
X1 = pd.DataFrame(index = Y.index)

#   X1:Derived Factors Related to Risk-Return Characteristic
X1['CSI500_CLOSE'] = index_close['000905.SH']
X1['WINDA_CLOSE'] = index_close['881001.WI']

for k in [1,5,10,20,60]:
    X1.loc[:,'_'.join(('CSI500_RET',str(k)))] = index_close['000905.SH'].pct_change(k)
    X1.loc[:,'_'.join(('WINDA_RET',str(k)))] = index_close['881001.WI'].pct_change(k)
    X1.loc[:,'_'.join(('CSI500_VOL',str(k)))] = index_close['000905.SH'].pct_change().rolling(k).std()
    X1.loc[:,'_'.join(('CSI500_TURNOVER',str(k)))] = index_turnover['000905.SH'].rolling(k).mean()
    X1.loc[:,'_'.join(('WINDA_TURNOVER',str(k)))] = index_turnover['881001.WI'].rolling(k).mean()
    X1.loc[:,'_'.join(('STYLE_1000_300_RET',str(k)))] = index_close['000852.SH'].pct_change(k) - index_close['000300.SH'].pct_change(k)
    X1.loc[:,'_'.join(('STYLE_1000_50_RET',str(k)))] = index_close['000852.SH'].pct_change(k) - index_close['000016.SH'].pct_change(k)
    X1.loc[:,'_'.join(('STYLE_1000_500_RET',str(k)))] = index_close['000852.SH'].pct_change(k) - index_close['000905.SH'].pct_change(k)
    X1.loc[:,'_'.join(('STYLE_500_300_RET',str(k)))] = index_close['000905.SH'].pct_change(k) - index_close['000300.SH'].pct_change(k)
    X1.loc[:,'_'.join(('STYLE_500_50_RET',str(k)))] = index_close['000905.SH'].pct_change(k) - index_close['000016.SH'].pct_change(k)
    X1.loc[:,'_'.join(('STYLE_300_50_RET',str(k)))] = index_close['000300.SH'].pct_change(k) - index_close['000016.SH'].pct_change(k)

    X1.loc[:,'_'.join(('STYLE_1000_300_RET',str(k),'STD'))] = X1.loc[:,'_'.join(('STYLE_1000_300_RET',str(k)))].rolling(60).std()
    X1.loc[:,'_'.join(('STYLE_1000_50_RET',str(k),'STD'))] = X1.loc[:,'_'.join(('STYLE_1000_50_RET',str(k)))].rolling(60).std()
    X1.loc[:,'_'.join(('STYLE_1000_500_RET',str(k),'STD'))] = X1.loc[:,'_'.join(('STYLE_1000_500_RET',str(k)))].rolling(60).std()
    X1.loc[:,'_'.join(('STYLE_500_300_RET',str(k),'STD'))] = X1.loc[:,'_'.join(('STYLE_500_300_RET',str(k)))].rolling(60).std()
    X1.loc[:,'_'.join(('STYLE_500_50_RET',str(k),'STD'))] = X1.loc[:,'_'.join(('STYLE_500_50_RET',str(k)))].rolling(60).std()
    X1.loc[:,'_'.join(('STYLE_300_50_RET',str(k),'STD'))] = X1.loc[:,'_'.join(('STYLE_300_50_RET',str(k)))].rolling(60).std()


X1['TURNOVER_1000_500_MA60'] = index_turnover['000852.SH'].rolling(60,min_periods=1).mean()/index_turnover['000905.SH'].rolling(60,min_periods=1).mean()
X1['TURNOVER_1000_300_MA60'] = index_turnover['000852.SH'].rolling(60,min_periods=1).mean()/index_turnover['000300.SH'].rolling(60,min_periods=1).mean()
X1['TURNOVER_1000_50_MA60'] = index_turnover['000852.SH'].rolling(60,min_periods=1).mean()/index_turnover['000016.SH'].rolling(60,min_periods=1).mean()
X1['TURNOVER_500_300_MA60'] = index_turnover['000905.SH'].rolling(60,min_periods=1).mean()/index_turnover['000300.SH'].rolling(60,min_periods=1).mean()
X1['TURNOVER_500_50_MA60'] = index_turnover['000905.SH'].rolling(60,min_periods=1).mean()/index_turnover['000016.SH'].rolling(60,min_periods=1).mean()
X1['TURNOVER_300_50_MA60'] = index_turnover['000300.SH'].rolling(60,min_periods=1).mean()/index_turnover['000016.SH'].rolling(60,min_periods=1).mean()
```

Figure 23: Python code to derive factors related to risk-return characteristic

3.3.3 Derived Factors Related to Hedging Demand

Alpha returns can impact the performance of a neutral strategy, thereby affecting the scale of the neutral strategy, and ultimately influencing the short position scale of neutral strategy products in stock index futures. Different from intuition, during continuous market declines, the hedging of short positions in stock index futures may not increase, and sometimes, the basis may even exhibit a converging trend. This is due to the fact that the neutral strategy contributes significantly to the primary short positions in stock index futures. When the market falls, Alpha strategies often underperform, resulting in decreasing need of neutral strategies. And this reduction could lead to a decrease in the demand for short position hedging in stock index futures. Hence, selecting indicators related to Alpha returns, neutral strategy return as predictive variables is essential.

Factors related to Alpha returns and neutral strategy are derived from the raw dataset, and the new derived factors are listed in the Table 11.

Categories	Variable	Variable Explanation	Parameter k Values
Alpha Return	PUBLIC_ENHANCE_EXRET_k	Cumulative ALPHA returns of CSI 500 Public Index Enhanced Funds for k days	k=1,5,10,20,60
	PUBLIC_ENHANCE_EXRET_MEAN_k	Mean (60-day moving average) of Cumulative ALPHA returns of CSI 500 Public Index Enhanced Funds for k days	k=1,5,10,20,60
	PUBLIC_ENHANCE_EXRET_MEDIAN_k	Median (60-day moving average) of Cumulative ALPHA returns of CSI 500 Public Index Enhanced Funds for k days	k=1,5,10,20,60
	PUBLIC_ENHANCE_EXRET_STD_k	Standard deviation (60-day moving average) of Cumulative ALPHA returns of CSI 500 Public Index Enhanced Funds for k days	k=1,5,10,20,60
	PRIVATE_ENHANCE_EXRET_k	Cumulative ALPHA returns of CSI 500 Private Index Enhanced Funds for k days	k=1,5,10,20,60
	PRIVATE_ENHANCE_EXRET_MEAN_k	Mean (60-day moving average) of Cumulative ALPHA returns of CSI 500 Private Index Enhanced Funds for k days	k=1,5,10,20,60
	PRIVATE_ENHANCE_EXRET_MEDIAN_k	Median (60-day moving average) of Cumulative ALPHA returns of CSI 500 Private Index Enhanced Funds for k days	k=1,5,10,20,60
	PRIVATE_ENHANCE_EXRET_STD_k	Standard deviation (60-day moving average) of Cumulative ALPHA returns of CSI 500 Private Index Enhanced Funds for k days	k=1,5,10,20,60
Neutral Strategy Return	PUBLIC_ALPHA_RET_k	Past K-day returns of Public Neutral Strategies on CSI 500	k=1,5,10,20,60
	PRIVATE_ALPHA_RET_k	Past K-day returns of private Neutral Strategies on CSI 500	k=1,5,10,20,60

Table 11: Derived Factors Related to Alpha strategy and neutral strategy

We utilized Python to generate all these new variables related to Alpha strategy and neutral strategy, code provided in Figure 24.

```
# X2:Derived Factors Related to Hedging Demand
X2 = pd.DataFrame(index = Y.index)
for k in [1,5,10,20,60]:
    X2.loc[:,'_'.join(('PUBLIC_ENHANCE_EXRET',str(k)))] = fund_value['public enhance'].pct_change(k) - index_close['000905.SH'].pct_change(k)
    X2.loc[:,'_'.join(('PUBLIC_ENHANCE_EXRET_MEAN',str(k)))] = X2.loc[:,'_'.join(('PUBLIC_ENHANCE_EXRET',str(k)))].rolling(60).mean()
    X2.loc[:,'_'.join(('PUBLIC_ENHANCE_EXRET_MEDIAN',str(k)))] = X2.loc[:,'_'.join(('PUBLIC_ENHANCE_EXRET',str(k)))].rolling(60).median()
    X2.loc[:,'_'.join(('PUBLIC_ENHANCE_EXRET_STD',str(k)))] = X2.loc[:,'_'.join(('PUBLIC_ENHANCE_EXRET',str(k)))].rolling(60).std()
    X2.loc[:,'_'.join(('PRIVATE_ENHANCE_EXRET',str(k)))] = fund_value['private enhance'].pct_change(k) - index_close['000905.SH'].pct_change(k)
    X2.loc[:,'_'.join(('PRIVATE_ENHANCE_EXRET_MEAN',str(k)))] = X2.loc[:,'_'.join(('PRIVATE_ENHANCE_EXRET',str(k)))].rolling(60).mean()
    X2.loc[:,'_'.join(('PRIVATE_ENHANCE_EXRET_MEDIAN',str(k)))] = X2.loc[:,'_'.join(('PRIVATE_ENHANCE_EXRET',str(k)))].rolling(60).median()
    X2.loc[:,'_'.join(('PRIVATE_ENHANCE_EXRET_STD',str(k)))] = X2.loc[:,'_'.join(('PRIVATE_ENHANCE_EXRET',str(k)))].rolling(60).std()
    X2.loc[:,'_'.join(('PUBLIC_ALPHA_RET',str(k)))] = fund_value['public alpha'].pct_change(k)
    X2.loc[:,'_'.join(('PRIVATE_ALPHA_RET',str(k)))] = fund_value['private alpha'].pct_change(k)
```

Figure 24: Python code to derive factors related to Alpha strategy and neutral strategy

3.3.4 Derived Factors Related to Trading volume and Open Interest

Trading data and position data can reflect the trading structure of stock index futures, thereby providing some predictive indication for the basis spread of stock index futures. Long-short net positions of top brokers can reflect the strength of investor's bullish and bearish forces, with decreasing values often go with an increase in short position hedging demand. Changes in open interest and trading volume can reflect the degree of long-short divergence and market speculative sentiment, thus indirectly influencing the basis spread of stock index futures. The trading volume-to-open interest ratio and the futures-to-spot trading ratio can reflect the strength of speculative forces in futures contract. An increase of volatility often accompanied by a rise in the trading volume-to-open interest ratio.

Factors related to volume and open interest are derived from the raw dataset, and the new derived factors are listed in the Table 12.

Variable	Variable Explanation	Parameter k Values
IC_VOLUME_SUM	Total Trading Volume of four CSI 500 Stock Index Futures	
IC_OI_SUM	Total Open Interest of four CSI 500 Stock Index Futures	
IC_VOLUME_OI	Trading-to-Open Interest Ratio of CSI 500 Stock Index Futures (ratio of trading volume to open interest)	
IC_CSI500_VOLUME_RATIO	Futures-to-Spot Trading Ratio of CSI 500 Stock Index Futures (ratio of futures trading volume to spot trading volume, multiplied by 10^6)	
NET_OI_TOP5	Long-Short Net Positions in CSI 500 Stock Index Futures - Top 5 brokers	
NET_OI_TOP10	Long-Short Net Positions in CSI 500 Stock Index Futures - Top 10 brokers	
NET_OI_TOP20	Long-Short Net Positions in CSI 500 Stock Index Futures - Top 20 brokers	
NET_OI_RATIO_TOP5	Long-Short Net Positions in CSI 500 Stock Index Futures as a Percentage of Total Open Interest - Top 5 brokers	
NET_OI_RATIO_TOP10	Long-Short Net Positions in CSI 500 Stock Index Futures as a Percentage of Total Open Interest - Top 10 brokers	
NET_OI_RATIO_TOP20	Long-Short Net Positions in CSI 500 Stock Index Futures as a Percentage of Total Open Interest - Top 20 brokers	
VOLUME_MA_k	Moving Average of Trading Volume of CSI 500 Stock Index Futures (over k days)	k=5,10,20,60
OI_MA_k	Moving Average of Total Open Interest of CSI 500 Stock Index Futures (over k days)	k=5,10,20,60
VOLUME_OI_MA_k	Moving Average of Trading volume -to-Open Interest Ratio of CSI 500 Stock Index Futures (over k days)	k=5,10,20,60
VOLUME_MA_CHG_k	Moving Average Percentage Change of Moving Average Trading Volume of CSI 500 Stock Index Futures over k days	k=5,10,20,60
OI_MA_CHG_k	Moving Average Percentage Change of Moving Average Open Interest of CSI 500 Stock Index Futures over k days	k=5,10,20,60

Table 12: Factors Related to Trading volume and Open Interest

We utilized Python to generate all these new variables related to trading volume and open interest, code provided in Figure 25.

```
# X3:Derived Factors Related to Trading volume and Open Interes
X3 = pd.DataFrame(index = Y.index)


X3['IC_VOLUME_SUM'] = ic_volume.sum(axis=1)
X3['IC_OI_SUM'] = ic_oi.sum(axis=1)
X3['IC_VOLUME_OI'] = X3['IC_VOLUME_SUM'] / X3['IC_OI_SUM']
X3['IC_CSI500_VOLUME_RATIO'] = ic_volume.sum(axis=1)/index_volume['volume']*1000000
X3['NET_OI_TOP5'] = broker_oi['long_oi_top5'] - broker_oi['short_oi_top5']
X3['NET_OI_TOP10'] = broker_oi['long_oi_top10'] - broker_oi['short_oi_top10']
X3['NET_OI_TOP20'] = broker_oi['long_oi_top20'] - broker_oi['short_oi_top20']
X3['NET_OI_RATIO_TOP5'] = (broker_oi['long_oi_top5'] - broker_oi['short_oi_top5'])/(broker_oi['long_oi_top5'] + broker_oi['short_oi_top5'])
X3['NET_OI_RATIO_TOP10'] = (broker_oi['long_oi_top10'] - broker_oi['short_oi_top10'])/(broker_oi['long_oi_top10'] + broker_oi['short_oi_top10'])
X3['NET_OI_RATIO_TOP20'] = (broker_oi['long_oi_top20'] - broker_oi['short_oi_top20'])/(broker_oi['long_oi_top20'] + broker_oi['short_oi_top20'])


for k in [5,10,20,60]:
    X3.loc[:,'_'.join(('VOLUME_MA',str(k)))] = X3['IC_VOLUME_SUM'].rolling(k).mean()
    X3.loc[:,'_'.join(('OI_MA',str(k)))] = X3['IC_OI_SUM'].rolling(k).mean()
    X3.loc[:,'_'.join(('VOLUME_OI_MA',str(k)))] = X3['IC_VOLUME_OI'].rolling(k).mean()
    X3.loc[:,'_'.join(('VOLUME_MA_CHG',str(k)))] = X3['IC_VOLUME_SUM'].rolling(k).mean().pct_change(k)
    X3.loc[:,'_'.join(('OI_MA_CHG',str(k)))] = X3['IC_OI_SUM'].rolling(k).mean().pct_change(k)
```

Figure 25: Python code to derive factors related to volume and open interest

3.3.5 Derived Factors Related to Basis

Stock index futures' basis exhibits certain trend and reversal features. In the short term, the basis can experience significant fluctuations or reach historical extremes, often followed by quick reversals. In the medium to long term, the basis tends to exhibit certain trends. Based on our observations of basis changes, we have selected features including the annualized basis rate after dividend adjustment, percentile rank, and correlation related to the basis and spot stock index.

Factors related to basis are derived from the raw dataset, and the new derived factors are listed in the Table 13.

Categories	Variable	Variable Explanation	Parameter k Values
Basis	BASIS_RATE_00	Annualized Basis Rate (Excluding Dividend Impact) of CSI 500 Stock Index Futures - Current Month vs. Spot	
	BASIS_RATE_01	Annualized Basis Rate (Excluding Dividend Impact) of CSI 500 Stock Index Futures - Next Month vs. Spot	
	BASIS_RATE_02	Annualized Basis Rate (Excluding Dividend Impact) of CSI 500 Stock Index Futures - Current Quarter vs. Spot	
	BASIS_RATE_03	Annualized Basis Rate (Excluding Dividend Impact) of CSI 500 Stock Index Futures - Next Quarter vs. Spot	
	BASIS_RATE_00_RANK_k	Percentile-Based Annualized Basis Rate (Excluding Dividend, k-year) of CSI 500 Stock Index Futures - Current Month vs. Spot	k=1,2
	BASIS_RATE_01_RANK_k	Percentile-Based Annualized Basis Rate (Excluding Dividend, k-year) of CSI 500 Stock Index Futures - Next Month vs. Spot	
	BASIS_RATE_02_RANK_k	Percentile-Based Annualized Basis Rate (Excluding Dividend, k-year) of CSI 500 Stock Index Futures - Current Quarter vs. Spot	
	BASIS_RATE_03_RANK_k	Percentile-Based Annualized Basis Rate (Excluding Dividend, k-year) of CSI 500 Stock Index Futures - Next Quarter vs. Spot	
	BASIS_RATE_00_MA_k	Moving Average-Based Annualized Basis Rate (Excluding Dividend, Moving Average of k Days) of CSI 500 Stock Index Futures - Current Month vs. Spot	k=5,10,20,60
	BASIS_RATE_01_MA_k	Moving Average-Based Annualized Basis Rate (Excluding Dividend, Moving Average of k Days) of CSI 500 Stock Index Futures - Next Month vs. Spot	
	BASIS_RATE_02_MA_k	Moving Average-Based Annualized Basis Rate (Excluding Dividend, Moving Average of k Days) of CSI 500 Stock Index Futures - Current Quarter vs. Spot	
	BASIS_RATE_03_MA_k	Moving Average-Based Annualized Basis Rate (Excluding Dividend, Moving Average of k Days) of CSI 500 Stock Index Futures - Next Quarter vs. Spot	
Correlation between Basis and Spot Stock Index	INDEX_BASIS02_CORR_k	Correlation Coefficient between CSI 500 Index and Current Quarter Annualized Basis Rate (k Days)	

Table 13: Factors Related to Basis

We utilized Python to generate all these new variables related to basis, code provided in Figure 26.

```
# X4:Derived Factors Related to Basis
X4 = pd.DataFrame(index = Y.index)

pctrank = lambda x:pd.Series(x).rank(pct=True).iloc[-1]

basis_rate = (ic_close.subtract(index_close['000905.SH'],axis=0) + dividend).divide(index_close['000905.SH'],axis=0) / daystoexpire * 365
basis_rate = basis_rate.replace([np.inf,-np.inf],0)

X4['BASIS_RATE_00'] = basis_rate['IC00']
X4['BASIS_RATE_01'] = basis_rate['IC01']
X4['BASIS_RATE_02'] = basis_rate['IC02']
X4['BASIS_RATE_03'] = basis_rate['IC03']

X4['BASIS_RATE_00_RANK_1Y'] = basis_rate['IC00'].rolling(250,min_periods=1).apply(pctrank)
X4['BASIS_RATE_01_RANK_1Y'] = basis_rate['IC01'].rolling(250,min_periods=1).apply(pctrank)
X4['BASIS_RATE_02_RANK_1Y'] = basis_rate['IC02'].rolling(250,min_periods=1).apply(pctrank)
X4['BASIS_RATE_03_RANK_1Y'] = basis_rate['IC03'].rolling(250,min_periods=1).apply(pctrank)

X4['BASIS_RATE_00_RANK_2Y'] = basis_rate['IC00'].rolling(500,min_periods=1).apply(pctrank)
X4['BASIS_RATE_01_RANK_2Y'] = basis_rate['IC01'].rolling(500,min_periods=1).apply(pctrank)
X4['BASIS_RATE_02_RANK_2Y'] = basis_rate['IC02'].rolling(500,min_periods=1).apply(pctrank)
X4['BASIS_RATE_03_RANK_2Y'] = basis_rate['IC03'].rolling(500,min_periods=1).apply(pctrank)

for k in [5,10,20,60]:
    X4.loc[:,'_'.join(('BASIS_RATE_00_MA',str(k)))] = X4['BASIS_RATE_00'].rolling(k).mean()
    X4.loc[:,'_'.join(('BASIS_RATE_01_MA',str(k)))] = X4['BASIS_RATE_01'].rolling(k).mean()
    X4.loc[:,'_'.join(('BASIS_RATE_02_MA',str(k)))] = X4['BASIS_RATE_02'].rolling(k).mean()
    X4.loc[:,'_'.join(('BASIS_RATE_03_MA',str(k)))] = X4['BASIS_RATE_03'].rolling(k).mean()
    X4.loc[:,'_'.join(('INDEX_BASIS02_CORR',str(k)))] = X4['BASIS_RATE_02'].rolling(k).corr(index_close['000905.SH'])
```

Figure 26: Python code to derive factors related basis

3.3.6 Derived Factors Related to Cross-Period Combination Features

Our target variable is specific yield difference between two contracts. The volume and price indicators of these two specific contracts will impact the spread in the following ways:

Differences in contract liquidity will affect the spread. We quantify this disparity using ratios or differences in trading volume, open interest, and trade-to-open interest ratio between the two contracts.

The number of days until the expiration of the contracts will influence the spread. As contracts approach expiration, the basis tends to converge more rapidly. If the forward contract doesn't follow this convergence, the spread will change. We measure this impact using the difference in days until expiration between the two contracts.

Spread changes exhibit certain seasonal patterns. We calculate the historical and future average returns of the spread for the corresponding months and expiration periods of the two contracts。

Greater differences in expiration periods between the two contracts often lead to higher spread volatility. We quantify this effect using the difference in expiration months between the two contracts.

Factors related to spread are derived from the raw dataset, and the new derived factors are listed in the Table 14.

Variable	Variable Explanation	Parameter k Values
SPREAD_RATE_near_far_MA_K	K-day Moving Average of Spread Rate (Annualized)	k=5,10,20,60 near_far=['00','01'],['00','02'],['00','03'],['01','02'],['01','03'],['02','03']
SPREAD_RATE_near_far_RANK_kY	Spread Rate k-year Percentile (Annualized)	k=1,2 near_far=['00','01'],['00','02'],['00','03'],['01','02'],['01','03'],['02','03']
SPREAD_RATE_near_far_MA5	5-day Moving Average of Spread Rate (Annualized)	near_far=['00','01'],['00','02'],['00','03'],['01','02'],['01','03'],['02','03']
SPREAD_RATE_near_far_MA60	60-day Moving Average of Spread Rate (Annualized)	
PAIR_RET_1D_near_far	Daily Return of Spread Combination	
PAIR_RET_5D_near_far	5-day Return of Spread Combination	
SPREAD_MA5_near_far	5-day Average Spread	
OI_RATIO_near_far	Near-month/Far-month Contract Position Ratio	
OI_SUB_near_far	Near-month/Far-month Contract Position Difference	
VOLUME_RATIO_near_far	Far-month/Near-month Trading Volume Ratio	
VOLUME_SUB_near_far	Far-month/Near-month Trading Volume Difference	
MATURITY_near	Days to Expiry for Near Month Contract	
OI_near	Near-month Contract Open Interest	
VOLUME_near	Near-month Contract Trading Volume	
VOLUME_OI_near	Near-month Contract Trading-to-Open Interest Ratio	
MATURITY_far	Days to Expiry for Far month Contract	
OI_far	Far-month Contract Open Interest	
VOLUME_far	Far-month Contract Trading Volume	
VOLUME_OI_far	Far-month Contract Trading-to-Open Interest Ratio	
MONTH_SUB_near_far	Maturity Difference in far Month-to-near Month Contract	

Table 14: Factors Related to Cross-period Combination Features

We utilized Python to generate all these new variables related to cross-period combination features, code provided in Figure 27.

```
#X5: Derived Factors Related to Cross-Period Combination Features
X5 = pd.DataFrame(index = Y.index)

for near,far in [['00','01'],['00','02'],['00','03'],['01','02'],['01','03'],['02','03']]:
    X5.loc[:,'_'.join(('SPREAD_RATE',near,far))] = -((ic_close['IC'+far] - ic_close['IC'+near]) / ( 0.5 * (ic_close['IC'+far] + ic_close['IC'+near])) * 12 / np.round(((daystoexpire
    for k in [5,10,20,60]:
        X5.loc[:,'_'.join(('SPREAD',near,far,'MA',str(k)))] = (ic_close['IC'+far] - ic_close['IC'+near]).rolling(k).mean()
        X5.loc[:,'_'.join(('SPREAD_RATE',near,far,'MA',str(k)))] = X5.loc[:,'_'.join(('SPREAD_RATE',near,far))].rolling(k).mean()

    X5.loc[:,'_'.join(('SPREAD_RATE',near,far,'RANK_1Y'))] =  X5.loc[:,'_'.join(('SPREAD_RATE',near,far))].rolling(250,min_periods= 60).apply(pctrank)
    X5.loc[:,'_'.join(('SPREAD_RATE',near,far,'RANK_2Y'))] =  X5.loc[:,'_'.join(('SPREAD_RATE',near,far))].rolling(500,min_periods= 60).apply(pctrank)


    X5.loc[:,'_'.join(('PAIR_RET_1D',near,far))] = ((ic_close['IC'+far] - ic_close['IC'+near]).diff(1).shift(-1) / (ic_close['IC'+far] + ic_close['IC'+near]) * 250).shift(1)
    X5.loc[:,'_'.join(('PAIR_RET_5D',near,far))] = ((ic_close['IC'+far] - ic_close['IC'+near]).diff(5).shift(-5) / (ic_close['IC'+far] + ic_close['IC'+near]) / 5 * 250).shift(5)


    X5.loc[:,'_'.join(('OI_RATIO',near,far))] = ic_oi['IC'+str(far)] / ic_oi['IC'+str(near)]
    X5.loc[:,'_'.join(('OI_SUB',near,far))] = ic_oi['IC'+str(far)] - ic_oi['IC'+str(near)]
    X5.loc[:,'_'.join(('VOLUME_RATIO',near,far))] = ic_volume['IC'+str(far)] / ic_volume['IC'+str(near)]
    X5.loc[:,'_'.join(('VOLUME_SUB',near,far))] = ic_volume['IC'+str(far)] - ic_volume['IC'+str(near)]
    X5.loc[:,'_'.join(('MATURITY',near))] = daystoexpire['IC'+str(near)]
    X5.loc[:,'_'.join(('OI',near))] = ic_oi['IC'+str(near)]
    X5.loc[:,'_'.join(('VOLUME',near))] = ic_volume['IC'+str(near)]
    X5.loc[:,'_'.join(('VOLUME_OI',near))] = ic_volume['IC'+str(near)] / ic_oi['IC'+str(near)]
    X5.loc[:,'_'.join(('MATURITY',far))] = daystoexpire['IC'+str(far)]
    X5.loc[:,'_'.join(('OI',far))] = ic_oi['IC'+str(far)]
    X5.loc[:,'_'.join(('VOLUME',far))] = ic_volume['IC'+str(far)]
    X5.loc[:,'_'.join(('VOLUME_OI',far))] = ic_volume['IC'+str(far)] / ic_oi['IC'+str(far)]
    X5.loc[:,'_'.join(('MONTH_SUB',near,far))] = np.round(((daystoexpire['IC'+far] - daystoexpire['IC'+near])/30))

X = pd.concat([X1,X2,X3,X4,X5], axis=1)
```

Figure 27: Python code to derive factors related cross-period combinations

3.4 Data selection for Derived Dataset

3.4.1 Training sample periods selection based on distribution characteristics

The stability of the distribution of the target variable is a prerequisite for establishing a predictive model. Hence, an examination of the distribution of the target variable is undertaken. In a sufficiently priced futures market, the spread between contracts with varying expiration dates should follow a normal distribution with a mean of zero. From historical data, the pricing of China's stock index futures has transitioned from being insufficiently priced to becoming relatively well-priced. We use Python to plot the distribution of 5-days return for 6 cross-period combinations from 2015-2016, and the code is provided in Figure 28.

```
fig = plt.figure(figsize = (20,20))
count=1
for i,year in enumerate(Y.index.year.unique().tolist()):
    for j,item in enumerate(Y.columns):
        plt.subplot(9,6,count)
        temp = Y[Y.index.year == year][item]
        sns.kdeplot(temp, shade = True, color = 'cornflowerblue')
        plt.ylim(0,10)
        plt.ylabel('')
        plt.xlabel('')
        plt.yticks([])
        count += 1
```

Figure 28: Code to plot the returns from 2015-2016 for 6 combinations using calendar-spread strategy

Considering the temporal dimension, the spread distribution of stock index futures has undergone substantial changes from 2015 to the present. The distributions of the returns are plotted in the Figure 29. During 2015-2016, stock index futures encountered trading limits due to a stock market crash, resulting in extremely insufficient futures pricing. The distribution of cross-period combination returns was dispersed and far from forming a normal distribution. Subsequently, after 2016, with the gradual relaxation of restrictions on stock index futures, their pricing efficiency improved, causing the distribution of cross-period combination returns to progressively improve. The distribution's tails became increasingly thinner over time, indicating a gradual reduction in arbitrage opportunities. It is evident that the distribution characteristics of the target variable during 2015-2016 significantly differ from those after 2017. Consequently, the starting point for our training samples is set from 2017 onwards. We selected records from 2017 for all variables.

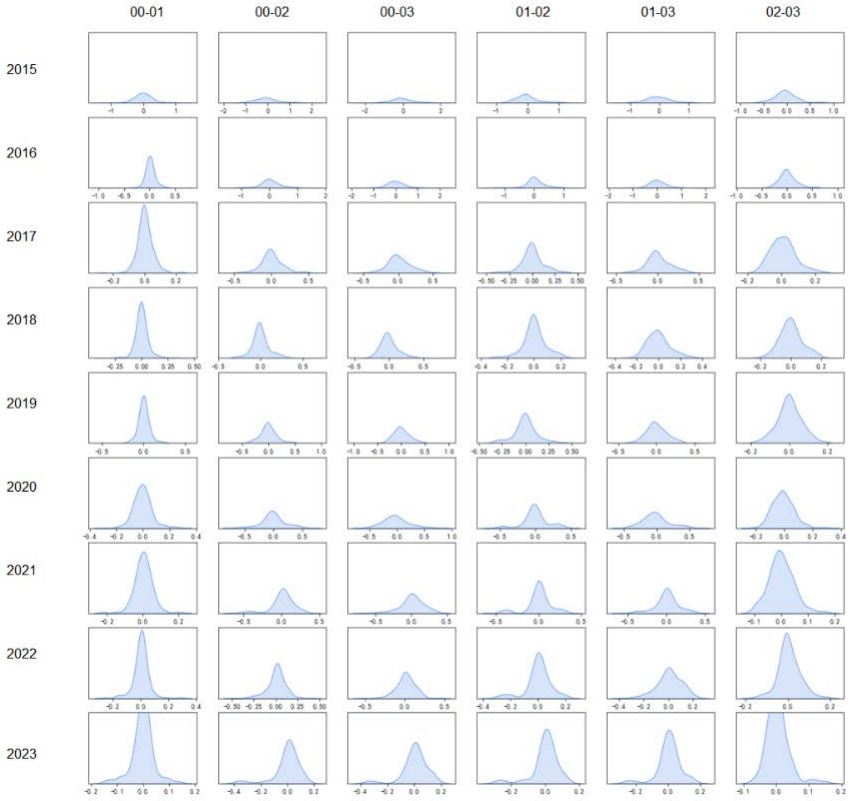


Figure 29: Distribution of 5-day Annualized Returns for Spread Combinations from 2015 to 2023

3.5 Data Integration and Cleaning for Derived Dataset

3.5.1 Data Integration

Initially, we integrate the five sub-tables (3.3.2-3.3.6) by matching them based on the "trading date."

3.5.2 Data Cleaning

In the data cleaning process, we begin by checking for any NULL data after deriving variables. The results are illustrated in Figure 30. We observe that "CSI500_VOL_1" has 1998 missing values. This is because one day return has no volatility, so we decide to drop this variable.

```
In [12]: na_detect = X.isnull().sum().sort_values(ascending = False)
    ...: na_detect[na_detect != 0]
Out[12]:
CSI500_VOL_1                       1998
PUBLIC_ENHANCE_EXRET_MEDIAN_60      119
STYLE_1000_300_RET_60_STD           119
OI_MA_CHG_60                        119
STYLE_300_50_RET_60_STD             119
                                   ...
PUBLIC_ALPHA_RET_1                    1
PRIVATE_ALPHA_RET_1                   1
STYLE_1000_300_RET_1                  1
STYLE_1000_50_RET_1                   1
PAIR_RET_1D_00_01                     1
Length: 248, dtype: int64
```

Figure 30: Checking Null value after deriving varaibles

Additionally, we exclude records that contain missing values. These missing values often arise during the computation of moving averages, especially when calculating the 60-day moving average of returns, where the initial sixty data points are insufficient for the entire calculation, shown in Figure 31.

Apart from missing values, we also encounter instances of infinite values. This occurs when the denominator is zero. For instance, when a contract reaches its exact expiration date, the time to maturity becomes zero, leading to the generation of infinite values. To address this, we replace these infinite values with zeros, shown in Figure 31.

Another thing that we consider is that, on the contract delivery date, the closing price tends to approach the contract settlement price, causing the closing basis spread and price spread to deviate significantly from normal levels. To ensure the robustness of our model and mitigate the impact of outliers, we opted to remove all records that on the contract delivery date, shown in Figure 31.

Finally, we align variables (X) with the target variables (Y) by trading date, shown in Figure 31.

```
#data intergration
X = pd.concat([X1,X2,X3,X4,X5], axis=1)

## data cleaning
na_detect = X.isnull().sum().sort_values(ascending = False)
na_detect[na_detect != 0]

X = X.drop(['CSI500_VOL_1'], axis=1)
X = X.dropna()

X.isnull().sum().sort_values(ascending = False)

X.isin([np.inf,-np.inf]).any().sort_values(ascending = False)

X = X.replace([np.inf,-np.inf],0)
X.isin([np.inf,-np.inf]).any().sort_values(ascending = False)

expire_dates = daystoexpire[daystoexpire['IC00'] == 0].index
Y = Y.drop(expire_dates)
Y = Y.dropna()
Y = Y[Y.index >= '2017-01-01']

len(X) == len(Y)

X = X[X.index.isin(Y.index)]
Y = Y[Y.index.isin(X.index)]

len(X) == len(Y)
```

Figure 31: Data integration and data cleaning process for the new derived X and Y

After integrating and cleaning the data, we are left with 328 X variables and 1494 records, as shown in Figure 32.

```
    ...: Y.info()
    ...: X.info()
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 1494 entries, 2017-01-03 to 2023-06-21
Data columns (total 6 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   00_01   1494 non-null   float64
 1   00_02   1494 non-null   float64
 2   00_03   1494 non-null   float64
 3   01_02   1494 non-null   float64
 4   01_03   1494 non-null   float64
 5   02_03   1494 non-null   float64
dtypes: float64(6)
memory usage: 81.7 KB
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 1494 entries, 2017-01-03 to 2023-06-21
Columns: 328 entries, CSI500_CLOSE to MONTH_SUB_02_03
dtypes: float64(299), int64(29)
```

Figure 32: Structured dataset after deriving new X and Y

3.6 Data Formation by Z-Score Transformation

In the process of data mining, it's important not only to consider the completeness and quality of the data but also to ensure that the distribution and scale of the data do not introduce biases or mislead models. Especially when dealing with datasets with multiple attributes, these attributes may have vastly different numerical ranges. Inputting these data directly into models might cause instability in model training or hinder model performance due to scale disparities.

To address this, we intend to utilize the z-score method for data normalization. The z-score, also known as the standard score, reflects how many standard deviations a value is from the mean of the entire dataset. The formula is defined as:

$$z = \frac{x - \mu}{\sigma}$$

, where x is a specific data point, μ represents the mean of the dataset, and $\sigma$is the standard deviation of the dataset.

Utilizing the z-score for normalization offers several benefits. Firstly, it ensures scale uniformity by centering all attributes at zero and giving them unit variance, which guarantees that every attribute is treated with equal significance during the modeling process. Secondly, it bolsters the stability of models, especially those employing gradient-based optimization techniques such as deep learning models, by facilitating quicker convergence. Lastly, z-scores enhance the interpretability of data, providing an intuitive understanding of a data point's position relative to the overall distribution, which is instrumental in spotting outliers.

However, it's important to note that we intend to divide the dataset into two parts: the "in-sample" and the "out-of-sample." The "in-sample" data will be used for model training, while the "out-of-sample" data will be used for model testing. During the model training phase, we will standardize the "in-sample" data. When it comes to testing the model, for the "out-of-sample" data, we will employ the mean and standard deviation of the "in-sample" data for the standardization of “out-of-sample” data, serving as input variables for testing model. Standardizing the entire dataset together is not appropriate because we cannot have access to future information standing at the current point. The standardization procedure is executed using Python, as illustrated in Figure 33. The results of standardization for both the training and test sets can be observed in Figure 34.

```
#Standardization procedures
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(X_train)
X_train_standard = pd.DataFrame(scaler.transform(X_train))
X_train_standard.index = X_train.index
X_train_standard.columns = X_train.columns

X_test_standard = pd.DataFrame(scaler.transform(X_test))
X_test_standard.index = X_test.index
X_test_standard.columns = X_test.columns
```

Figure 33: Standardization process in Python

```
In [18]: X_train_standard
    ...: X_test_standard
Out[18]:
            CSI500_CLOSE  WINDA_CLOSE  ...  VOLUME_SUB_02_03
MONTH_SUB_02_03
Date                                   ...
2021-01-04      1.169959     2.454450  ...         -2.533400
0.0
2021-01-05      1.248903     2.580132  ...         -1.339101
0.0
2021-01-06      1.232836     2.599610  ...         -1.685280
0.0
2021-01-07      1.246663     2.647758  ...         -2.214246
0.0
2021-01-08      1.273669     2.618770  ...         -2.549079
0.0
...                  ...          ...  ...               ...
...
2023-06-14      0.561763     1.337917  ...         -1.079156
0.0
```

Figure 34: X variables in training and testing set after standardization

4 Data transformation

4.1 Variable Reduction

4.1.1 Variable Reduction Using Features Selection

In Section 3, we proceeded by considering the five potential aspects: A-share market risk-return characteristics, hedging demands, trading volume and open interest, contract basis, and cross-period combination features. From these aspects, we generated a preliminary set of 328 features by overlaying various parameters for individual indicators. Further refinement of feature selection is necessary. We aim to retain features for prediction that exhibit strong correlations with the target variable and possess reasonable economic significance.

The target variables are returns of six stock index futures cross-period combinations. Our goal is to construct individual predictive models for these six portfolios, with the aim of forecasting their respective returns. Despite all six strategies being based on the same underlying assets--CSI 500 Stock Index Futures, variations in contract maturities may lead to variations in important features.

Different maturities of CSI 500 Stock Index Futures contracts have varying feature selections for our arbitrage strategies. This distinction primarily arises from diverse market participants and trading behaviors for contracts with distinct maturities. For instance, shorter-term contracts typically have higher liquidity and trading volumes, potentially emphasizing features related to liquidity in the selection process. Conversely, longer-term contracts are more susceptible to factors such as interest rates, funding costs, seasonality, and macroeconomic influences, making these factors pivotal in predicting their returns. Although all contracts are based on the same underlying asset, the varying market participant behavior, liquidity, and economic conditions necessitate different feature sets to enhance the accuracy in return predictions.

We implement feature selection for each arbitrage strategy's return prediction model using Python code, as illustrated in Figure 35. Given our initial set of variables, which comprises 328 features, we opt to select the top 50 significant features to avoid discarding variables that may contain valuable information.

```
# Feature Selection   each important feature will be stored in var_filter
from sklearn.feature_selection import SelectKBest, f_regression
var_filter = {}
for pair in ['00_01','00_02','00_03','01_02','01_03','02_03']:
    selector = SelectKBest(f_regression, k=50).fit(X_train_standard, Y_train[pair])
    selected_mask = selector.get_support()
    filtered = X_train_standard.columns[selected_mask].tolist()
    var_filter.update({pair:filtered})

var_filter_df = pd.DataFrame.from_dict(var_filter, orient='index')
var_filter_df_transposed = var_filter_df.transpose()
var_filter_df_transposed.to_excel(r'C:\Users\keden\Desktop\rp\iteration3\data_code\var_filter_transposed.xlsx', sheet_name='Sheet1')
var_filter_df_transposed = pd.read_excel(r'C:\Users\keden\Desktop\rp\iteration3\data_code\var_filter_transposed.xlsx', sheet_name='Sheet1',
```

Figure 35: Feature selection in Python

Based on the outcome of selected features, we indeed observe slight differences in the variables influencing the returns of the six arbitrage combinations. The filtered features are displayed in Figure 36.

A	B	C	D	E	F	G
	00_01	00_02	00_03	01_02	01_03	02_03
0	CSI500_TURNOVER_1	CSI500_TURNOVER_1	CSI500_TURNOVER_1	CSI500_VOL_5	CSI500_TURNOVER_1	CSI500_TURNOVER_1
1	WINDA_TURNOVER_1	WINDA_TURNOVER_1	WINDA_TURNOVER_1	STYLE_1000_300_RET_5	WINDA_TURNOVER_1	WINDA_TURNOVER_1
2	STYLE_1000_300_RET_1_STD	CSI500_VOL_5	CSI500_VOL_5	CSI500_VOL_10	CSI500_VOL_5	STYLE_500_300_RET_1_STD
3	STYLE_1000_50_RET_1_STD	CSI500_VOL_10	CSI500_TURNOVER_5	PUBLIC_ENHANCE_EXRET_5	CSI500_TURNOVER_5	STYLE_500_50_RET_1_STD
4	STYLE_500_300_RET_1_STD	PRIVATE_ENHANCE_EXRET_5	WINDA_TURNOVER_5	PRIVATE_ENHANCE_EXRET_5	WINDA_TURNOVER_5	CSI500_VOL_5
5	STYLE_500_50_RET_1_STD	PUBLIC_ENHANCE_EXRET_10	CSI500_VOL_10	PUBLIC_ENHANCE_EXRET_10	CSI500_VOL_10	CSI500_TURNOVER_5
6	CSI500_VOL_5	BASIS_RATE_02	WINDA_TURNOVER_10	PUBLIC_ENHANCE_EXRET_20	WINDA_TURNOVER_10	WINDA_TURNOVER_5
7	CSI500_TURNOVER_5	BASIS_RATE_03	CSI500_VOL_20	OI_MA_CHG_10	CSI500_VOL_20	CSI500_VOL_10
8	WINDA_TURNOVER_5	BASIS_RATE_02_MA_5	CSI500_TURNOVER_20	INDEX_BASIS02_CORR_60	CSI500_TURNOVER_20	WINDA_TURNOVER_10
9	STYLE_1000_300_RET_5_STD	BASIS_RATE_03_MA_5	WINDA_TURNOVER_20	SPREAD_RATE_00_01	WINDA_TURNOVER_20	CSI500_VOL_20
10	CSI500_TURNOVER_10	SPREAD_RATE_00_01	BASIS_RATE_02_MA_5	SPREAD_00_01_MA_5	PRIVATE_ENHANCE_EXRET_5	CSI500_TURNOVER_20
11	WINDA_TURNOVER_10	SPREAD_00_01_MA_5	BASIS_RATE_03_MA_5	SPREAD_RATE_00_01_MA_5	PUBLIC_ENHANCE_EXRET_10	WINDA_TURNOVER_20
12	STYLE_1000_300_RET_10_STD	SPREAD_RATE_00_01_MA_5	BASIS_RATE_02_MA_10	SPREAD_RATE_00_01_MA_10	PUBLIC_ENHANCE_EXRET_20	CSI500_VOL_60
13	CSI500_TURNOVER_20	SPREAD_00_01_MA_10	BASIS_RATE_03_MA_10	OI_RATIO_00_01	BASIS_RATE_03_MA_5	PRIVATE_ENHANCE_EXRET_STD_1

Figure 36: Selected features of six combination

4.2 Variable projection

4.2.1 Dimensionality Reduction Using Principal Component Analysis (PCA)

4.2.1.1 Advantages and Limitations of Principal Component Analysis

PCA is a commonly used data dimensionality reduction technique aimed at transforming high-dimensional data into a lower-dimensional form. There are both advantages and limitation for using PCA.

Advantages:

Ensure variable Independence: All principal components derived from PCA are intrinsically orthogonal, ensuring their mutual independence. By transitioning from original predictor variables to these principal components, we can effectively mitigate multicollinearity among the variables. In linear regression, a fundamental assumption requires the independence of predictor variables. This assumption is inherently met when employing PCA.

Disadvantages:

- No Intuitive Interpretation of Principal Components: The interpretation of each principal component may not be intuitive. Explaining the practical significance represented by each principal component can become more challenging.
- Inability to Handle Missing Data: PCA has limited capabilities in handling datasets with missing values. It requires uniform data sample formatting and cannot accommodate missing value.

4.2.1.2 PCA on Influencing Factors of Stock Index Futures Returns

After the variable reduction process in Section 5.1, we have effectively reduced the initial 328 variables to 50. Nevertheless, the presence of 59 variables still remains relatively substantial. We intend to further employ Principal

Component Analysis (PCA) to achieve dimensionality reduction on these variables, aiming to extract crucial principal components.

We implement PCA using Python code, as illustrated in Figure 37.

```
#PCA
from sklearn.decomposition import PCA
pca = PCA().fit(X_train_standard)
plt.plot(pca.explained_variance_ratio_,'-')
plt.ylabel('Proportion of Variance Explained')
plt.xlabel('Principal Component')

plt.plot(pca.explained_variance_ratio_.cumsum())
plt.ylabel('Cumulative Proportion of Variance Explained')
plt.xlabel('Principal Component')
plt.axhline(0.95,linestyle = '--',c = 'red')
plt.axvline(np.where(pca.explained_variance_ratio_.cumsum() >= 0.95)[0].min(),linestyle = '--', c='red')
print('At least',np.where(pca.explained_variance_ratio_.cumsum() >= 0.95)[0].min(),'principals contribute over 95% variance.')

pca = PCA(n_components = 0.95).fit(X_train_standard)
X_train_pca = pca.transform(X_train_standard)
X_test_pca = pca.transform(X_test_standard)
```

Figure 37: Python code to implement PCA

First, we fit PCA to the standardized training dataset (X_train_standard). PCA finds the most important principal components by projecting the data into a new feature space to explain the variance in the data. Then, we create two plots to analyze the results of PCA: The first plot, shown in Figure 38, displays the variance explained by each principal component. This helps us determine the contribution of each principal component to the total variance. Typically, we aim to select principal components that explain a significant portion of the variance.

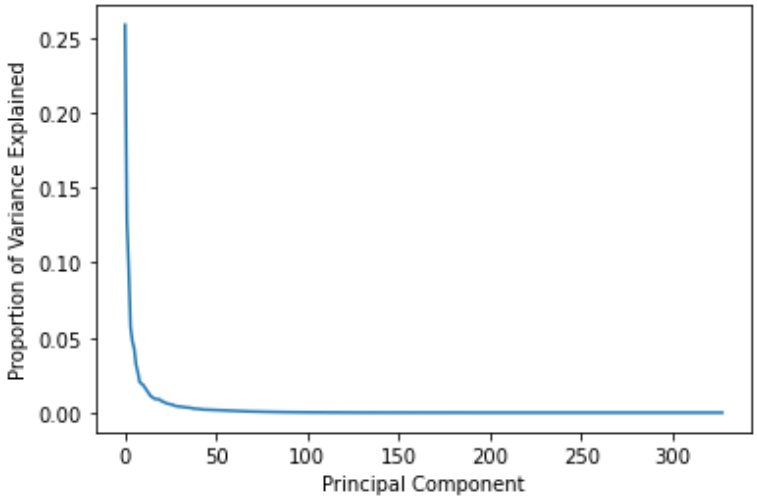


Figure 38: Proportion of variance explained

The second plot, as shown in Figure 39, shows the cumulative variance explained as the number of principal components increases. Our goal is to find the number of principal components that explain at least 95% of the total variance. A red dashed line indicates the threshold of 95% cumulative variance, and a vertical red dashed line represents the number of principal components required to reach that threshold, which is 44. We find that at least 44 principals contribute over 95% variance.

Finally, we use the reduced-dimensional principal components obtained through PCA to transform both the training and test dataset. This reduces the dimensionality of the data, simplifying the model while retaining over 95% of the total variance to ensure that the model retains crucial information.

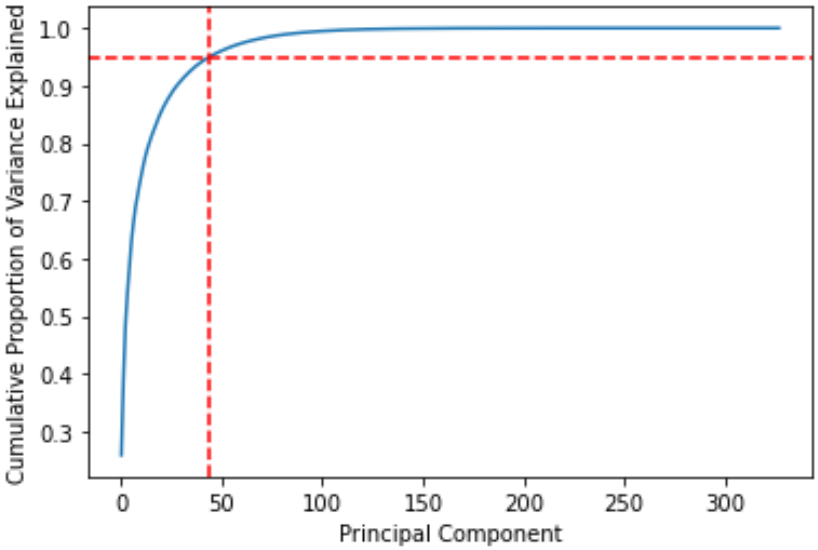


Figure 39: Cumulative proportion of variance explained

5 1. Data-mining method selection

5.1 Aligning Data-Mining Method with Data Mining Objective

Firstly, from the perspective of data mining objectives, our task involves predicting the cumulative returns of six cross-period combinations of stock index futures over the next 5 trading days. Following the forecasting of returns for these six combinations, our aim is to select the combination with the highest return as our rebalancing strategy for the next five days. This is a predictive problem. A predictive model estimates the output data size based on the input data provided. Adopting a predictive model, the objective is to estimate the cumulative returns of six cross-period stock index futures combinations over the next five days, using the input data. Utilizing the pre-processed and dimensionally reduced data as input, our aim is to forecast the cumulative returns of each combination over the next five days. Based on these predictions, we can identify the combinations with the highest expected returns for subsequent rebalancing operations. The core focus of this endeavor aligns with the selection and implementation of a prediction model, finely meet our data mining goal.

Additionally, when considering the nature of our dataset, it's evident that it falls under the category of time-series data. This type of data encompasses a series of continuous data points that include stock index futures prices and trading volumes tracked over time. Given the inherent temporal dependencies and trends present in time-series data, it becomes imperative to select a data mining model designed specifically for time-series analysis. Furthermore, it's crucial to note that our dataset has no labeled data, meaning that both the independent and dependent variables are continuous numerical values. This situation leads us to adopt supervised learning methods, where the fundamental principle revolves around mapping input features to the target variable. In our case, the specific target is to predict the continuous numerical variable representing stock index futures returns over the subsequent five trading days. Lastly, our dataset comprises approximately 1500 records, which falls within the range of moderately sized data. This data size is well-suited for employing models known for their effectiveness within this size range.

Based on the considerations above, we have the option to choose from supervised learning regression models, including linear regression, regression trees, ensemble methods such as random forest regression trees, and deep learning algorithms like neural networks. In the following chapter, we will delve into a discussion of the strengths and weaknesses of each algorithm to arrive at our final model selection.

5.2 Model Evaluation and Selection

In Section 5.1, we initially laid the groundwork by analyzing the objectives of our data mining goal and the characteristics of our dataset. In particular, we focused on the predictive nature of our problem and highlighted that our task aligns well with supervised learning regression methods. Specifically, four potential models were considered viable for our task: linear regression, regression trees, random forest regression trees, and neural networks. In this chapter, we will undertake an in-depth comparative analysis, shedding light on the strengths and weaknesses of each model.

5.2.1 Linear Regression

Linear regression often serves as a starting point for regression modeling because of its simplicity and interpretability. In the context of arbitrage strategies for the CSI 500 stock index futures, a basic linear regression model can help us get an initial sense of the relationship between the independent variables and the target variable. However, this linear approximation often falls short of capturing the complexities in financial markets. Besides, linear regression comes with a set of assumptions, such as the independence, homoscedasticity, and normality of error terms, which often do not hold true in real-world financial data. These assumptions limit the model's applicability, especially when dealing with data that exhibits heteroscedasticity or autocorrelation. Moreover, linear regression assumes causal relationship between explanatory and dependent variables. In financial markets, variables often interact with each other in a more complex, bidirectional manner. Lastly, linear models are ill-suited for capturing the impact of sudden market shocks, such as political crises or abrupt economic changes, which can result in sharp fluctuations in stock prices.

In simpler words, while linear regression is a good starting point for understanding relationships between variables, it often isn't sufficient for capturing the complexities of financial markets. It operates under a set of assumptions that don't always hold true in real-world financial data.

Therefore, while linear regression has advantages in terms of interpretability and computational efficiency, its limitations make it potentially less suitable for predicting arbitrage strategies for the CSI 500 stock index futures. This is particularly true when the data is highly complex and exhibits nonlinear characteristics, where more advanced models could be a better fit.

5.2.2 Regression Trees

Regression trees excel at handling nonlinear and intricate relationships, making them valuable for predicting complex financial markets like CSI 500 stock index futures. They not only capture intricate patterns in the data but also present these patterns in an intuitive manner, which is useful for explaining and communicating investment strategies. However, individual regression trees tend to overfit, particularly when the data contains a lot of noise or outliers. Overfitting leads to poor performance on new, future financial data. Additionally, regression trees can be sensitive to small changes in the input data, affecting their robustness.

In this context, Random Forests serve as an ensemble method that combines the predictions of multiple decision trees to offer more accurate and robust results. Random Forests inherently include regularization features, minimizing the risk of overfitting, which is particularly crucial when predicting financial time-series data-the five-day cumulative returns, the target variable of this iteration. Random Forests are also more adept at handling high-dimensional data and the interaction between variables, which is important considering the nature of multiple arbitrage strategies and many attributes in our project. Compared to individual decision trees, Random Forests usually provide more robust and accurate predictions by averaging the forecasts from multiple trees, thus reducing model variance.

Therefore, while single regression trees offer the advantage of capturing nonlinear patterns and providing intuitive insights, they are limited by their propensity for overfitting and sensitivity to minor data changes. These limitations are particularly significant for predicting arbitrage strategies in the volatile and complex CSI 500 stock index futures market. In contrast, ensemble methods like random forests mitigate these drawbacks by aggregating predictions from multiple trees, offering a more robust and accurate model that is better suited for this application.

5.2.3 Random Forest

Given the complexity of our data mining goal, which involves predicting returns of six combinations with various contract expiration dates in the arbitrage strategies, a highly flexible model is essential to accurately forecast future returns. Random forests, an ensemble method combining multiple decision trees, excel at capturing intricate nonlinear relationships in the data. This is particularly beneficial for dealing with the noisy and outlier-ridden nature of financial market data. By averaging predictions from numerous trees, random forests reduce the risk of overfitting inherent in single models, thus offering more robust forecasts.

Moreover, random forests can rank the relative importance of each feature, thereby identifying key factors that influence arbitrage strategies. This adds an extra layer of interpretability and can deepen our understanding of the stock index futures market. The ensemble nature of random forests also improves the model's generalization capabilities, making it more reliable when confronted with unseen data in the ever-changing financial markets.

Considering our dataset involves a high number of independent variables, the ability of random forests to perform effective dimensionality reduction is highly advantageous. This addresses the 'curse of dimensionality,' making it a particularly fitting choice for your project.

By integrating these strengths, random forests offer a robust approach, adapting well to the multifaceted challenges posed by the prediction of arbitrage strategies in CSI 500 stock index futures.

5.2.4 Neural Network

In the context of our project, which aims to predict arbitrage opportunities for the CSI 500 stock index futures, the use of Neural Networks presents several significant advantages.

First, unlike linear regression models, Neural Networks have the capacity to capture complex, non-linear relationships between variables. This is crucial in financial markets where outcomes are influenced by lots of interconnected factors such as price, trading volume, and market depth. Second, Neural Networks are particularly effective in dealing with time-dependent data. This capability is augmented using specialized structures like Recurrent Neural Networks (RNNs) and Long Short-Term Memory Networks (LSTMs), which offer even more nuanced understandings of time dependencies, a feature often lacking in single regression trees. Finally, Neural Networks can also handle high-dimensional data well, which means that we can consider a broader range of variables that might influence stock index futures returns without being overly constrained by dimensionality.

Despite these advantages, it's important to acknowledge the computational complexity of Neural Networks, which generally require more resources and training time than simpler models like linear regression or individual regression trees. However, given that our dataset comprises only about 1500 entries, the computational burden is not a major concern at least for now. The principal drawback is the "black box" nature of Neural Networks, making their outcomes less interpretable. To mitigate this issue, we intend to use Random Forests as a supplementary tool to identify key features affecting arbitrage opportunities.

In summary, given the ability to model complex relationships and time dependencies, along with our moderate dataset size and the supplementation of Random Forests for interpretability, we conclude that Neural Networks are also well-suited for our objectives.

5.3 Model Selection

In conclusion, based on our in-depth analysis of different models in Section 5.2, we have decided to employ two algorithm approaches, utilizing Random Forests and Neural Networks, to address our predictive problem.

6 Data-mining Algorithm Selection

6.1 Algorithm Selection

6.1.1 Algorithm Selection for Random Forest

Random Forest algorithms essentially come in two flavors: Classifier and Regressor. Each serves different types of problems, and selecting the appropriate one is crucial for achieving our data mining objectives.

- Random Forest Classifier:

Random Forest Classifiers are predominantly used for classification tasks. In this applications, each decision tree within the forest makes a class classification, and the final forecast is the most frequently occurring class among all the trees. While excellent for tasks requiring labeled data to categorize new data points into predefined classes, this approach is not aligned with the objectives of our project. Our focus is not on categorizing data into distinct classes; rather, we aim to predict a continuous variable. Moreover, our dataset has no labeled data, which is often a key requirement for classification tasks.

- Random Forest Regressor:

On the other hand, Random Forest Regressors are used for regression tasks. In these applications, each tree predicts a continuous value, and the final prediction is the average of these individual forecasts.

Random Forest Regressors are particularly suitable for scenarios like our project for several reasons. Firstly, Random Forest Regressors are designed to predict a continuous output variable, aligning directly with our data mining goal of predicting cumulative returns of stock index futures. Secondly, Random Forest Regressors excel at capturing complex, non-linear relationships, which are prevalent in financial time-series data. Our data falls under the realm of time-series, and Random Forest Regressors are well-suited to capture the inherent temporal dependencies. Thirdly, Random Forest Regressors are generally resistant to overfitting, particularly beneficial when the dataset includes noise or outliers Moreover, Random Forest Regressors are scalable and can efficiently handle moderately sized datasets. With approximately 1500 records in our dataset, the Random Forest Regressor strikes a balance between computational efficiency and predictive accuracy.

6.2 Algorithm Selection for Neural Networks

MLP (Multi-Layer Perceptron) and RBF (Radial Basis Function) are different types of algorithms in neural networks. In MLP, weights and parameters adjust through backpropagation like stochastic gradient descent, affecting the whole network. In contrast, RBF tweaks individual radial basis functions' centers and widths for localized adaptation. These functions connect closely with specific data points or areas, capturing local patterns, but might lag behind MLP in understanding broader patterns.

Considering our data mining goals, we opted for the MLP algorithm due to these reasons:

- Prediction of Continuous Values:

MLP excels at predicting continuous values. Forecasting cumulative returns over the next five days involves continuous value prediction. MLP's flexible output layer activation yields precise outcomes. In comparison, RBF networks might struggle with such continuous value tasks.

- Capturing Complex Patterns:

MLP excels in decoding intricate nonlinear patterns. Financial data's complex interactions benefit from MLP's architecture, boosting prediction quality. RBF networks may lack MLP's strength with intricate nonlinear patterns.

- Support for Multiple Outputs:

MLP inherently handles diverse output tasks. In our project, predicting both continuous returns and classification ranking benefits from MLP's tailored output layer. On the other hand, while RBF theoretically accommodates multiple outputs, it leans towards single outputs. RBF's utility lies more in specific areas like localized approximations, less suited for various output needs.

In conclusion, our choice of MLP aligns with our data mining objectives. It effectively handles continuous value prediction and intricate pattern recognition, while also accommodating diverse outputs and classification tasks.

6.3 Parameter Selection in Model Building

6.3.1 Parameter Selection in Random Forest

In Random Forest model, we consider to adjust certain parameters to optimize our model performance. Specifically, we consider the following parameter settings, shown in Figure 40.

1. n_estimators (Number of Estimators) set to 200: Increasing n_estimators appropriately can enhance model performance. However, an excessively high value for n_estimators can also lead to model overfitting and longer training times. For our datasets with around 1500 records, starting with a relatively high value like 200 oensures a sufficiently complex model.

2. max_depth (Maximum Tree Depth)set to 20: Financial market data often exhibits complexity, and setting a very high max_depth can make the model very sensitive to noisy data, impacting generalization. Restricting the maximum depth to around 20 strikes a balance, controlling tree complexity while preserving predictive capabilities.

3. min_samples_split (Minimum Samples for Splitting)set to 2: Given a relatively small dataset, setting min_samples_split to a small value, like 2, allows the tree to split on smaller nodes, capturing subtle data features and relationships. We should not to set it too low in order to avoid overfitting.

4. min_samples_leaf (Minimum Samples per Leaf)set to 2: In financial market data, important patterns may exist in leaf nodes with few samples. Setting min_samples_leaf to 2 enables effective capture of these critical but uncommon patterns.

5. max_features (Maximum Features)set to “None”: For financial market data prediction, we deicide to consider all features due to potential complex nonlinear relationships. Randomness in random forests arises from random feature selection for trees, mitigating overfitting risk.

In conclusion, these parameter settings aim to balance model complexity and performance, taking into account the characteristics of financial market datasets.

```
#Random forest  ——  Filtered vars RF
from sklearn.ensemble import RandomForestRegressor
n_estimators = 200
max_depth = 20
min_samples_split =2
min_samples_leaf = 2
max_features = None

rf = RandomForestRegressor(
    n_estimators=n_estimators,
    max_depth=max_depth,
    min_samples_split=min_samples_split,
    min_samples_leaf=min_samples_leaf,
    max_features=max_features
)
```

Figure 40: Parameter Settings for random forest model in Python

6.3.2 Parameter selection in Neural Networks

We consider the following parameter settings in Neural Networks Model, shown in Figure 41.

1. Solver set to: 'adam':

We set solver to 'adam'. This is suitable for moderately sized datasets like ours with around 1500 records. 'adam' excels on larger datasets and can effectively harness the neural network's potential, which is valuable when dealing with complex financial market data. 'adam' serves as an adaptive learning rate optimization algorithm, adept at handling the inherent complexity and variability in financial market data. This enhances the neural network's fitting ability, leading to more accurate predictions of cumulative returns in stock index futures.

2. Hidden Layer Structure (hidden_layer_sizes) set to (100, 100):

The financial market data often contains intricate nonlinear relationships. Therefore, incorporating multiple hidden layers with 100 neurons each enhances the model's capacity to capture these complexities. Predicting cumulative returns in stock index futures requires considering intricate interactions among numerous variables (50 in our project). By increasing the number of hidden layers and neurons, neural network becomes more proficient at learning these nonlinear relationships, ultimately improving predictive performance.

3. Activation Function set to 'relu':

'relu' stands as a commonly used activation function, particularly suitable for deep neural networks. It effectively addresses nonlinear price fluctuations and trading behaviors prevalent in financial market data. Employing 'relu' helps the neural network better capture these nonlinear relationships, thereby enhancing predictive capabilities.

4. Regularization Parameter Alpha Set to 1e-5:

Regularization aids in preventing overfitting, and a small alpha value allows for a controlled level of model complexity while retaining its fitting capabilities. In financial market data prediction, overfitting can lead to models performing well on training data but poorly on new data. By setting a small alpha value, neural network model generalizes better to future stock index futures data, making it more practical.

5. Maximum Iterations (max_iter) Set to 1000:

Increasing the maximum iteration count ensures that the model adequately fits the data during training, especially for larger neural network structures. In our project, raising the maximum iteration count provides sufficient training time for neural network model to comprehensively learn patterns and trends within financial market data. This, in turn, improves accuracy in predicting cumulative returns.

```
# Neural network ——  Filtered vars NN
from sklearn.neural_network import MLPRegressor
nn = MLPRegressor(solver='adam', random_state=1,
                  hidden_layer_sizes=(100, 100), activation='relu',
                  alpha=1e-5, max_iter=1000)
```

Figure 41: Parameter Settings for neural network model in Python

7.1 Data Segmentation for Model Training and Backtesting

Our dataset covers the timeframe from January 2017 to June 2023. In order to facilitate subsequent model testing, we divide this dataset into two distinct parts: the "in-sample" and "out-of-sample" sets. Specifically, the in-sample dataset, spanning from 2017 to 2021, serves as the basis for training our models. Conversely, the out-of-sample dataset, which covers the period from 2021 to 2023, is reserved for evaluating the performance of our trained models. This data separation approach is consistently applied to each individual model (Since we need to predict returns for six combinations of stock index futures, we will develop six separate models, each focused on predicting a specific combination). This ensures the creation of well-defined training and testing datasets. Figure 42 shows how we split the data into the training set (in-sample) and the testing set (out-of-sample in our model.

```
## In & Out of Sample
Y_train = Y.loc[:'2021-01-01',:]
Y_test = Y[~Y.index.isin(Y_train.index)]

X_train = X.loc[:'2021-01-01',:]
X_test = X[~X.index.isin(X_train.index)]
```

Figure 42: Data partition for each model

7.2 Model Execution

7.2.1 Prediction Model Using Random Forest

Before constructing our model, we define its configuration. The reasoning behind these configuration choices has been extensively covered in Section 6.3.1. In this section, we solely present the results, shown in Figure 43.

7.2.1.1 Model Configuration

1. n_estimators: The number of decision trees in the ensemble is set to 200.

2. max_depth: The maximum depth of each tree is limited to 20, controlling the complexity of individual trees.

3. min_samples_split: The minimum number of samples required to split an internal node is set to 2, allowing for finer splitting.

4. min_samples_leaf: The minimum number of samples required to be at a leaf node is set to 2, potentially capturing smaller patterns.

5. max_features: It is set to None, meaning it considers all features when making splits, which is suitable for this financial data where complex relationships among features might exist.

```
from sklearn.ensemble import RandomForestRegressor
n_estimators = 200
max_depth = 20
min_samples_split =2
min_samples_leaf = 2
max_features = None

rf = RandomForestRegressor(
    n_estimators=n_estimators,
    max_depth=max_depth,
    min_samples_split=min_samples_split,
    min_samples_leaf=min_samples_leaf,
    max_features=max_features
)
```

Figure 43: Model Configuration of Random Forest Model

7.2.1.2 Model Training and Model Prediction:

We start by training separate Random Forest models for different stock index futures combinations, referred to as 'pair.' Each 'pair,' like '00_01,' represents a trading strategy involving long positions in forward contracts and short positions in near-term contracts. For example, '00_01' means we go long on the contract expirating in the next month and short the contract expiring in the current month. These models use a training dataset that has 50 independent variables ('var_filter[pair]') . These 50 independent variables have been selected by feature selection algorithm from the original 328 variables, which has been discussed in Section 4. After training, the model is used to make predictions on the testing dataset for each 'pair.' These predictions are stored in the 'Y_pred' DataFrame.

```
Y_pred = pd.DataFrame(columns = Y.columns)
for pair in ['00_01','00_02','00_03','01_02','01_03','02_03']:
    rf.fit(X_train_standard[var_filter[pair]],Y_train[pair])
    pred = rf.predict(X_test_standard[var_filter[pair]])
    Y_pred.loc[:,pair] = pred

Y_pred.index = Y_test.index
```

Figure 44: Model Training and Model Prediction of Random Forest Model

7.2.1.3 Model Visualization:

We generate a visual representation in Figure 46 to facilitate a more intuitive comparison between our model's predictions and the actual values for each pair of stock index futures contracts. This visualization allows us to assess the effectiveness of our predictions by examining the differences between them and the actual values, as well as identifying overall trends.

```
plt.figure(figsize = (20,10))
count = 1
for item in Y_test.columns:
    plt.subplot(6,1,count)
    pred_temp = Y_pred[item]
    Y_test_temp = Y_test[item]
    pred_temp.plot(c = 'firebrick')
    Y_test_temp.plot(c = 'navy')
    count += 1
```

Figure 45: Visualization of Random Forest Model

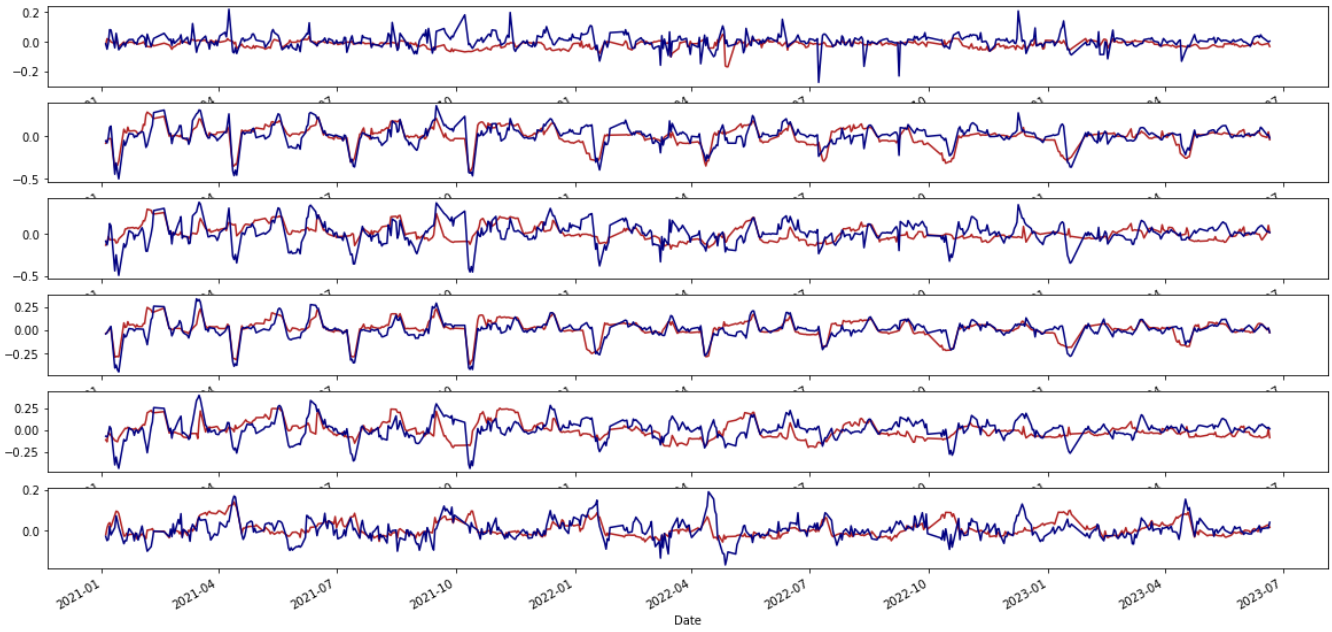


Figure 46: Visualization of the prediction result using Random Forest Model compared to the actual values

7.2.1.4 Model Evaluation:

We then calculate the Mean Squared Error (MSE) for each pair's predictions, shown in Figure 47. MSE measures the average squared difference between predicted and actual values, providing insight into the model's accuracy.

Furthermore, we compute an accuracy rate by comparing the predicted rankings of the six combinations with the actual rankings of the strategy's returns. This accuracy rate measures how often the model precisely predicts the stock index futures with the highest returns. The accuracy of exact ranking prediction is 20%.

```
metrics.mean_squared_error(Y_test[item], Y_pred[item])
   ...: mse
Out[8]:
00_01    0.002975
00_02    0.010411
00_03    0.019291
01_02    0.005415
01_03    0.017319
02_03    0.002084
dtype: float64
```

Figure 47: MSE result for each prediction model using random forest algorithm

```
In [9]: accuracy = (Y_test.idxmax(axis=1) ==
Y_pred.idxmax(axis=1))
   ...: accuracy_rate = accuracy.sum()/len(accuracy)
   ...: accuracy_rate
Out[9]: 0.20246478873239437
```

Figure 48: Accuracy rate for prediction using random forest model

7.2.1.5 Feature Importance:

One of the advantages of employing Random Forest is its capacity to rank the importance of features, thereby improving our comprehension of the distinct set of features that impact the performance of each of the six arbitrage strategies (six combinations).

After training the model, we calculate the feature importance using the 'feature_importances_' attribute of the Random Forest model. This attribute contains the importance scores assigned to each feature based on how much they contribute to the model's predictive performance. The feature importance are sorted in ascending order, and the top 20 most important features are selected for analysis. These are the features that have the highest impact on predicting the returns for the '00_01' pair. The process is shown in Figure 49. Finally, a horizontal bar chart is created to visualizes the feature importance, with the most important features displayed at the top. The plot is shown in Figure 50. This plot provides a clear view of which features are the most influential when predicting returns for the '00_01' pair.

```
# Create an empty DataFrame to aggregate feature importances
importance_combined = pd.DataFrame(columns=['pair', 'feature', 'importance'])

# 6 pair'00_01','00_02','00_03','01_02','01_03','02_03'
pairs = ['00_01', '00_02','00_03','01_02','01_03','02_03']

for pair in pairs:
    # Create an DataFrame to store feature importances
    importance = pd.DataFrame()
    importance['feature'] = X_train_standard.columns

    #
    rf = RandomForestRegressor()

    # train model
    rf.fit(X_train_standard, Y_train[pair])

    # get imporatant features
    importance['importance'] = rf.feature_importances_

    # get top 20
    importance = importance.sort_values('importance', ascending=True)
    importance = importance[-20:]

    #
    importance['pair'] = pair

    #
    importance_combined = pd.concat([importance_combined, importance], axis=0)

# save to excel file
importance_combined.to_excel(r'C:\Users\keden\Desktop\rp\iteration3\data_code\combined_feature_importance.xlsx', index=False)

# plot the feature imporatance
for pair in pairs:
    importance_pair = importance_combined[importance_combined['pair'] == pair]
    plt.barh(importance_pair['feature'], importance_pair['importance'], color='navy')
    plt.title('Feature Importance for Pair: ' + pair)
    plt.show()
```

Figure 48: Getting top 20 important features for each combination in random forest model

Taking "00_01" as an example, the top 20 variables that most influence the returns of the "00_01" combination are depicted in the Figure 49.

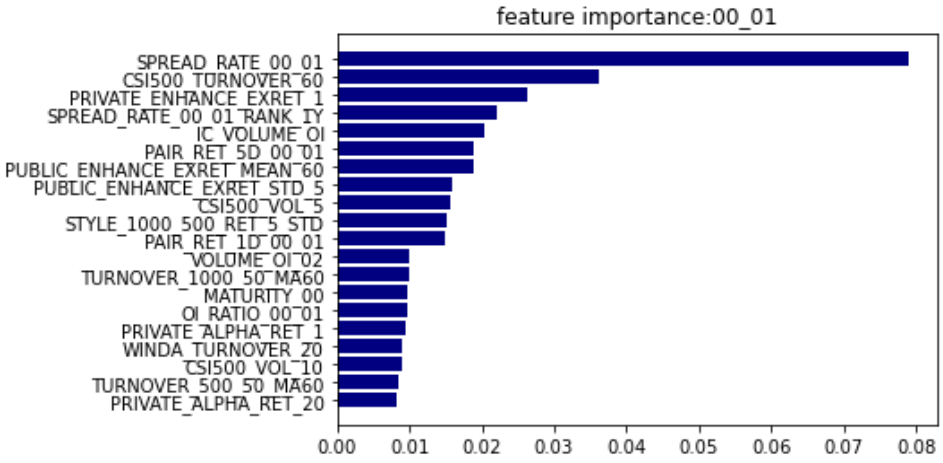


Figure 49: Top 20 important feature that most influence the returns of the "00_01" combination

7.2.2 Prediction Model Using Neural Networks

Before constructing our neural network model, we define its configuration, which has been discussed in Section 6.3. 2. Here, we present the key configuration settings for our neural network model, shown in Figure 50.

7.2.2.1 Model Configuration

1.Solver: We choose the 'adam' solver.

2. Hidden Layer Structure: The neural network architecture consists of two hidden layers, each containing 100 neurons.

3.The activation function we used is 'relu,' which is suitable for handling non-linearity and reducing the risk of gradient vanishing.

4. Regularization Parameter (alpha): We set the regularization parameter alpha to 1e-5.

5. Maximum Iterations (max_iter): The maximum number of iterations is set to 1000. This ensures that the model has enough training time to capture patterns and trends in the financial data effectively.

```
# Neural network ——  Filtered vars NN
from sklearn.neural_network import MLPRegressor
nn = MLPRegressor(solver='adam', random_state=1,
                  hidden_layer_sizes=(100, 100), activation='relu',
                  alpha=1e-5, max_iter=1000)
```

Figure 50: Model Configuration of Neural Networks Model

7.2.2.2 Model Training and Model Prediction:

We proceed by training individual neural network models for different stock index futures combinations, referred to as 'pair.' Each 'pair,' such as '00_01,' represents a trading strategy that involves long positions in forward contracts and short positions in near-term contracts. For instance, '00_01' corresponds to going long on the contract expiring in the next month and shorting the contract expiring in the current month. The neural network models utilize a training dataset that has been preprocessed and filtered down to 50 important independent variables ('var_filter[pair]'). These 50 variables have been selected through feature selection techniques, as discussed in Section 4. After training, the neural network models are

utilized to make predictions on the testing dataset for each 'pair.' The resulting predictions are stored in the 'Y_pred' DataFrame.

```
Y_pred = pd.DataFrame(columns = Y.columns)
for pair in ['00_01','00_02','00_03','01_02','01_03','02_03']:
    nn.fit(X_train_standard[var_filter[pair]],Y_train[pair])
    pred = nn.predict(X_test_standard[var_filter[pair]])
    Y_pred.loc[:,pair] = pred

Y_pred.index = Y_test.index
```

Figure 51: Model Training and Model Prediction of Neural Networks Model

7.2.2.3 Model Visualization:

To provide a clear comparison between our model's predictions and the actual values for each pair of stock index futures contracts, we create a visual representation. This visualization helps us to assess the effectiveness of our predictions by allowing us to visualize the differences between predicted and actual values and identifying overall trends between those two. The visualization is presented in Figure 53.

```
plt.figure(figsize = (20,10))
count = 1
for item in Y_test.columns:
    plt.subplot(6,1,count)
    pred_temp = Y_pred[item]
    Y_test_temp = Y_test[item]
    pred_temp.plot(c = 'firebrick')
    Y_test_temp.plot(c = 'navy')
    count += 1
```

Figure 52: Visualization of Neural Networks Model

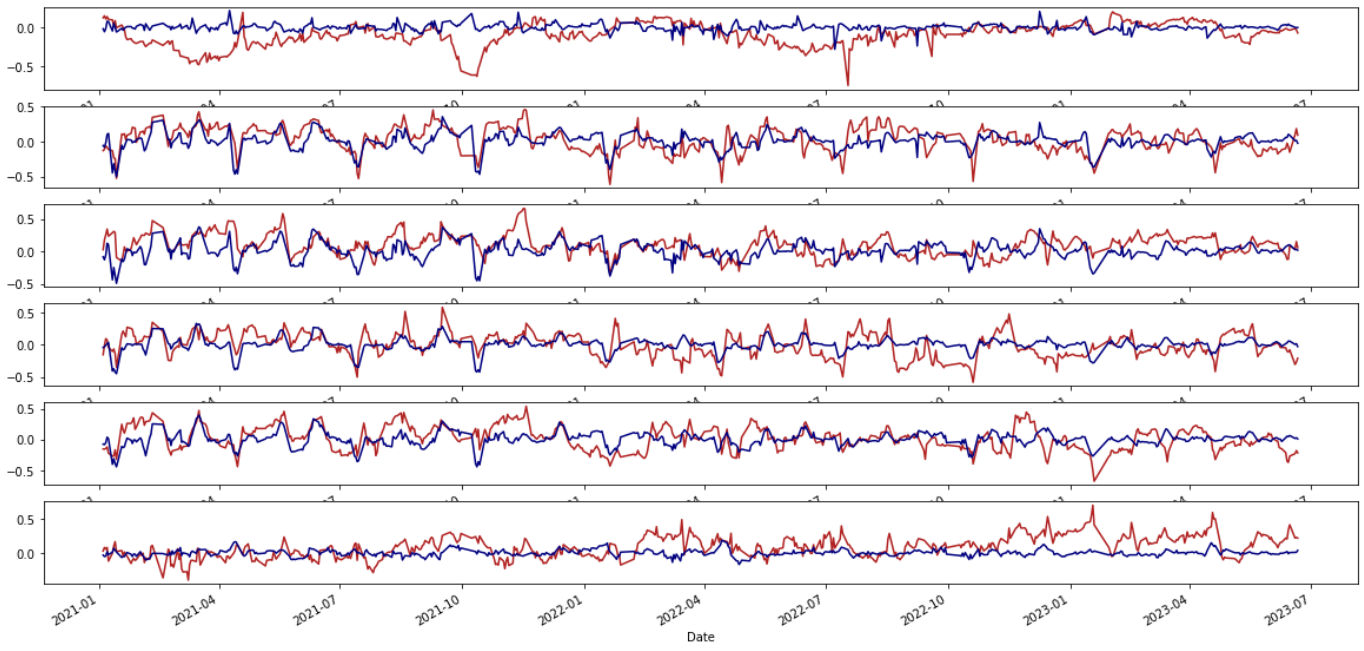


Figure 53: Visualization of the prediction result using Neural Networks Model compared to the actual values

7.2.2.4 Model Evaluation:

We proceed to evaluate the neural network model's performance. Firstly, we calculate the Mean Squared Error (MSE) for each pair's predictions The results are depicted in Figure 54.

Additionally, we compute an accuracy rate by comparing the predicted rankings of the six combinations with the actual rankings of the strategy's returns. This accuracy rate indicates how often the model accurately predicts the stock index futures with the highest returns, providing a measure of the model's ranking prediction accuracy. The accuracy rate is determined to be 19.5%, shown in Figure 55.

```
In [13]: from sklearn import metrics
    ...: mse = pd.Series()
    ...: for item in Y_test.columns:
    ...:     mse[item] =
metrics.mean_squared_error(Y_test[item], Y_pred[item])
    ...: mse
Out[13]:
00_01    0.032266
00_02    0.026197
00_03    0.042152
01_02    0.031845
01_03    0.040064
02_03    0.033732
dtype: float64
```

Figure 54: MSE result for each prediction model in Neural Networks

```
In [14]:
    ...: accuracy = (Y_test.idxmax(axis=1) ==
Y_pred.idxmax(axis=1))
    ...: accuracy_rate = accuracy.sum()/len(accuracy)
    ...: accuracy_rate
Out[14]: 0.1954225352112676
```

Figure 55: Accuracy rate for prediction in Neural Networks

7.3 Pattern Discovery and Discussion

7.3.1 Comparing MSE Performance: Random Forest vs. Neural Network

In Figure 56, the left side displays the MSE of the Random Forest model, while the right side shows the MSE of the Neural Network model. Based on the results, it is evident that the Random Forest model consistently exhibits lower MSE values for the predictions of cumulative returns across all six strategies compared to the Neural Network model. This observation suggests that the Random Forest model provides more accurate predictions for these specific combinations, as its predictions are closer to the actual values.

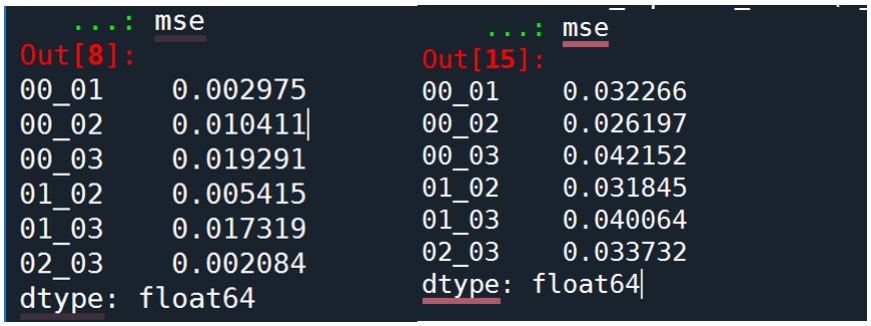


Figure 56: MSE comparison between two models

7.3.2 Comparing Accuracy: Random Forest vs. Neural Network

In Figure 57, the upper part displays the accuracy of the Random Forest model, while the lower side shows the accuracy of the Neural Network model. This accuracy rate indicates how often the model accurately predicts the stock index futures with the highest returns, providing a measure of the model's ranking prediction accuracy.

The accuracy of exact ranking prediction for the Random Forest model was 20%. The accuracy rate for the Neural Network model was determined to be 19.5%. This suggests that the Random Forest model was slightly better at precisely identifying the combinations with the highest returns compared to the Neural Network. One thing we consider is that while the MSE values suggested that the Neural Network had more significant prediction errors in terms of absolute returns, it performed reasonably well in terms of ranking accuracy. Specifically, the Neural Network sstill has the ability to maintain the relative ranking of returns across the six combinations, even if it did not precisely predict the absolute returns. This indicates that the model had a good grasp of the relative performance of the different combinations, showing that it understand the relative profitability of each combination during the same time frame.

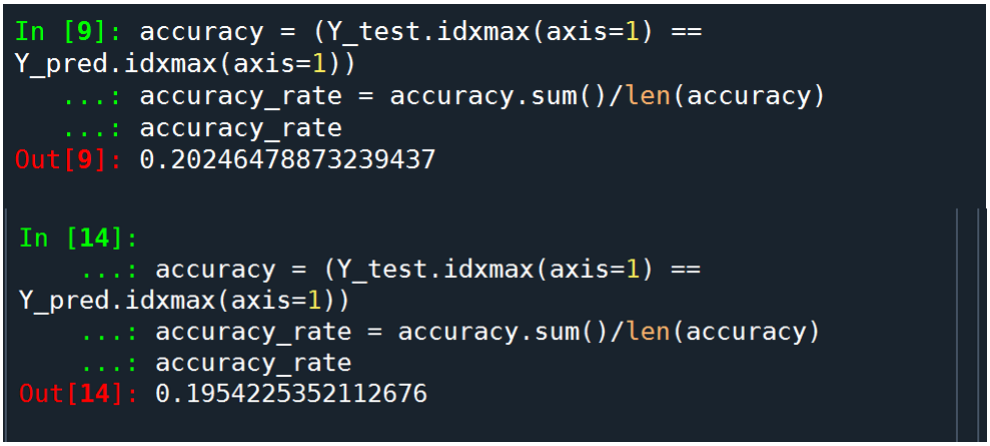


Figure 57: Accuracy comparison between two models

7.3.3 Comparing Prediction Performance in Visualization: Random Forest vs. Neural Network

Figure 58 exhibits results consistent with the MSE findings in Section 7.3.1. In the plot, the red line represents our model's predictions, while the blue line represents the actual returns. From the plot, we can observe that in the Random Forest model, the predicted trends for cumulative returns in all six combinations closely align with the actual return trends. There is minimal deviation in the absolute values of returns, and there is no significant gap between the actual and predicted values.

However, in the Neural Network model's chart, there are notable disparities between the actual and predicted values. For instance, for the '00_01' combination, the Neural Network model's predictions exhibit suboptimal performance during

the periods from January 2021 to April 2021, October 2021 to November 2021, and for '02_03,' during the period from January 2023 to June 2023.

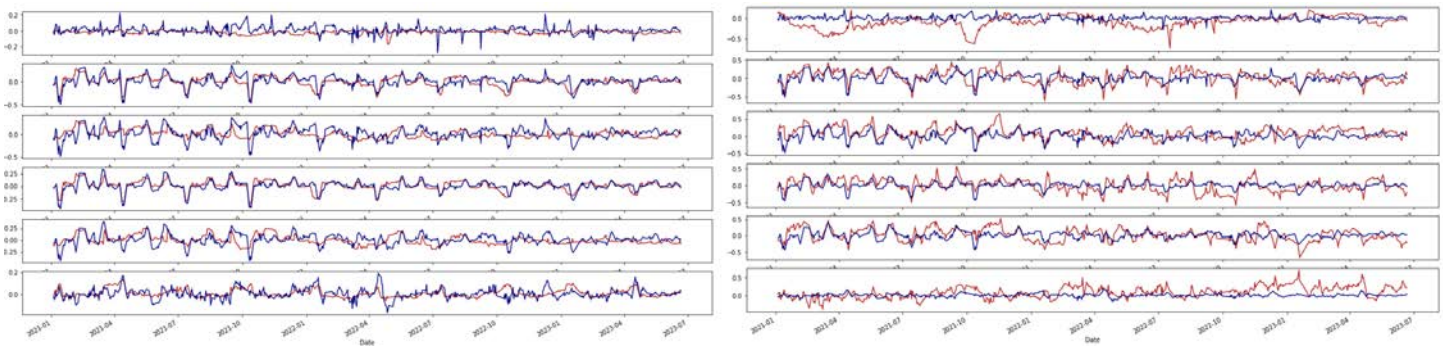


Figure 58: Comparing Prediction Performance in Visualization

8 Interpretation

8.1 Further Discussion and Reflection about the Pattern

In Section 7.3, we conducted an initial comparative analysis of two distinct modeling approaches. First, we examined the Random Forest model's performance in predicting returns for six different combinations of stock index futures. Second, we evaluated the Neural Network model's predictions for each of the six combinations. Our evaluation criteria encompassed metrics such as Mean Squared Error (MSE) and model absolute accuracy, enabling a comprehensive assessment of the predictive capabilities of both models. Additionally, we employed visualizations to depict the trends of actual versus predicted values for a clearer and more intuitive understanding of each model's performance between random forest model and neutral network across all pairs (a total of 2*6=12 models).

From our analysis in Section 7.3, we arrive at several key conclusions and reflections:

1. Regardless of whether we assess based on Mean Squared Error (MSE) for absolute returns or accuracy rates for relative return rankings, Random Forest consistently outperforms Neural Network.

2. While the MSE values consistently favored the Random Forest model, indicating its superior predictive accuracy in terms of absolute returns, the Neural Network model also exhibited almost similar level of ranking accuracy. This suggests that even though the Neural Network may not have precisely predicted the absolute returns, it retained the ability to gauge the relative profitability of each combination during the same time frame.

Taking a step further, the divergence in performance between these two models underscores a critical consideration in practical model application: the need to align model evaluation metrics with the specific objectives of investors or researchers.

In the context of financial decision-making, investors, specifically, speculators, may seek strategies that maximize their absolute returns, aiming to identify investments that offer the highest overall profitability. This perspective involves comparing the returns of stock index futures to those of various alternative asset classes and investment opportunities. From this viewpoint, a model's accuracy in predicting absolute returns becomes the primaryfor decision-making. From our modeling process, we found random forest algorithm did a good job in this aspect. Conversely, some investors, often hedgers, may have a specific mandate or preference for using stock index futures as part of a hedging portfolio. In such cases, the focus shifts towards analyzing the relative rankings and relative returns of different stock index futures hedging strategies within the portfolio. Hedgers adopting this approach are less concerned with the precise prediction of absolute returns and are more interested in ensuring that the chosen strategies consistently outperform others within the same portfolio. Therefore, the capacity of a model to accurately gauge the relative profitability of each strategy during the same time frame holds

greater significance. In this aspect, the performance of both random forest algorithm and NN are not significantly different from each other.

In essence, the choice between prioritizing absolute returns or relative rankings in financial modeling depends on investors' categories and thus their investment goals. This highlights the importance of aligning model evaluation with specific investment objectives and considering these nuanced aspects when selecting and deploying models in financial applications.

At this point, our analysis has delved into model assessments for each algorithm and model comparisons between two algorithms, as discussed in Sections 7.3 and 8.1. In these sections, we conducted a comprehensive exploration of the extracted patterns, encompassing data, initial model outputs, observed patterns, and data visualizations. We also offered reflections on these findings.

8.2 Interpretations on Important features

As we move forward, our focus shifts towards a deeper examination of the models' outputs, with a particular emphasis on the significance of features generated by the Random Forest model. This exploration is aimed at uncovering valuable insights into the key factors that influence arbitrage strategies. We're particularly interested in how these factors differ across strategies with varying maturity dates for stock index futures. By identifying these distinctions and understanding their economic implications, we aim to gain deeper insights into the stock index futures market.

We first extract the top 20 influential features on returns for each six combinations in the Random Forest algorithms, as illustrated in Figure 59.

```
# Create an empty DataFrame to aggregate feature importances
importance_combined = pd.DataFrame(columns=['pair', 'feature', 'importance'])

# 6 pair'00_01','00_02','00_03','01_02','01_03','02_03'
pairs = ['00_01', '00_02','00_03','01_02','01_03','02_03']

for pair in pairs:
    # Create an DataFrame to store feature importances
    importance = pd.DataFrame()
    importance['feature'] = X_train_standard.columns

    #
    rf = RandomForestRegressor()

    # train model
    rf.fit(X_train_standard, Y_train[pair])

    # get imporatant features
    importance['importance'] = rf.feature_importances_

    # get top 20
    importance = importance.sort_values('importance', ascending=True)
    importance = importance[-20:]

    #
    importance['pair'] = pair

    #
    importance_combined = pd.concat([importance_combined, importance], axis=0)

# save to excel file
importance_combined.to_excel(r'C:\Users\keden\Desktop\rp\iteration3\data_code\combined_feature_importance.xlsx', index=False)

# plot the feature imporatance
for pair in pairs:
    importance_pair = importance_combined[importance_combined['pair'] == pair]
    plt.barh(importance_pair['feature'], importance_pair['importance'], color='navy')
    plt.title('Feature Importance for Pair: ' + pair)
    plt.show()
```

Figure 59: Extracting the top 20 influential features in Random Forest Model

The most important independent variables of each combination are listed as below, shown in Table 15.

00_01	00_02	00_03	01_02	01_03	02_03
WINDA_RET_1	SPREAD_RATE_01_02	TURNOVER_1000_50_MA60	SPREAD_RATE_01_03_RANK_2Y	SPREAD_01_02_MA_10	SPREAD_RATE_02_03_MA_20
PRIVATE_ENHANCE_EXRET_STD_60	VOLUME_OI_MA_10	SPREAD_00_02_MA_5	SPREAD_00_01_MA_20	VOLUME_RATIO_01_02	SPREAD_RATE_00_02_MA_60
OI_RATIO_00_01	CSI500_RET_10	SPREAD_RATE_02_03_MA_10	WINDA_RET_5	OI_SUB_00_01	SPREAD_RATE_02_03_RANK_1Y
STYLE_1000_500_RET_5_STD	SPREAD_00_01_MA_10	SPREAD_01_02_MA_5	SPREAD_02_03_MA_20	SPREAD_RATE_02_03_MA_10	CSI500_VOL_10
TURNOVER_500_50_MA60	WINDA_TURNOVER_20	OI_RATIO_02_03	CSI500_VOL_5	SPREAD_RATE_02_03	PAIR_RET_1D_00_02
SPREAD_RATE_00_01_RANK_2Y	CSI500_VOL_5	TURNOVER_500_50_MA60	SPREAD_RATE_00_03	SPREAD_RATE_01_02_MA_60	BASIS_RATE_00_RANK_1Y
OI_MA_CHG_20	SPREAD_RATE_00_01_MA_5	MATURITY_03	WINDA_TURNOVER_60	SPREAD_RATE_00_02_MA_10	SPREAD_RATE_01_02_MA_60
VOLUME_OI_02	SPREAD_RATE_02_03_MA_20	MONTH_SUB_00_03	VOLUME_01	OI_RATIO_00_03	SPREAD_RATE_02_03
TURNOVER_1000_50_MA60	VOLUME_OI_MA_60	CSI500_VOL_10	VOLUME_OI_00	MONTH_SUB_01_03	CSI500_VOL_5
SPREAD_RATE_00_01_RANK_1Y	OI_RATIO_00_01	MATURITY_02	OI_RATIO_00_03	SPREAD_00_02_MA_5	INDEX_BASIS02_CORR_10
PUBLIC_ENHANCE_EXRET_STD_5	BASIS_RATE_00_RANK_2Y	SPREAD_RATE_00_02	IC_VOLUME_OI	SPREAD_RATE_00_02_MA_5	BASIS_RATE_00
PRIVATE_ALPHA_RET_1	VOLUME_OI_MA_5	CSI500_VOL_5	SPREAD_RATE_02_03_RANK_2Y	TURNOVER_500_50_MA60	WINDA_RET_5
PAIR_RET_1D_00_01	PUBLIC_ENHANCE_EXRET_STD_5	SPREAD_01_02_MA_10	MATURITY_02	CSI500_VOL_10	SPREAD_RATE_02_03_MA_10
IC_VOLUME_OI	MATURITY_01	SPREAD_RATE_00_03	VOLUME_OI_MA_10	MATURITY_02	CSI500_RET_5
PAIR_RET_5D_00_01	SPREAD_RATE_00_02	SPREAD_RATE_00_01	MATURITY_01	MATURITY_03	PAIR_RET_1D_02_03
CSI500_VOL_5	SPREAD_RATE_00_01_MA_10	VOLUME_OI_00	SPREAD_RATE_01_02	SPREAD_RATE_00_03	BASIS_RATE_01_MA_60
PUBLIC_ENHANCE_EXRET_MEAN_60	NET_OI_RATIO_TOP5	OI_RATIO_00_01	NET_OI_RATIO_TOP5	OI_RATIO_00_01	OI_SUB_00_02
PRIVATE_ENHANCE_EXRET_1	SPREAD_RATE_00_01	SPREAD_RATE_00_02_MA_10	SPREAD_00_01_MA_10	VOLUME_OI_00	PAIR_RET_5D_02_03
CSI500_TURNOVER_60	MATURITY_03	MATURITY_00	MATURITY_03	VOLUME_RATIO_00_01	MATURITY_02
SPREAD_RATE_00_01	MATURITY_00	VOLUME_RATIO_00_01	MATURITY_00	MATURITY_00	MATURITY_03

Table 15: Top 20 influential features in Random Forest Model

The common factors influencing the returns of the six stock index futures combinations are as follows:

1. Stock Index Return: When the corresponding stock indices perform well and exhibit a rising trend, it can positively impact long positions in far-term stock index futures and vise versa.

2. Stock Index Return Volatility: High market volatility can create trading opportunities for both long and short positions. Traders may use far-term contracts to hedge against market fluctuations, while also capitalizing on the increased price swings in near-term contracts. Therefore, when volatility rises, it can provide favorable conditions for the profit potential of the combinations.

3. Stock Index Turnover: Trading volume in the stock market reflects market activity and liquidity. This increased liquidity can lead to lower transaction costs and better execution prices, contributing to the overall performance of the strategies.

4. Open Interest (OI) of stock index future: An increase in Open Interest can suggest growing interest and participation in the stock index futures market. It signals a bullish or bearish sentiment.

5. Trading Volume stock index future: Increasing trading volume suggests increased market activity. Higher volume can lead to narrower bid-ask spreads and improved liquidity, which means investors could execute their positions with

less slippage and lower transaction costs. In markets with higher trading volume, price movements can become more volatile. This volatility can create both opportunities and risks for our strategies

6. Spread rate: We consider the calendar-spread arbitrage portfolio as an asset, and therefore, the spread rate can be regarded as the price of this asset. Consequently, the relationship between the model's target variable, which is the future return of the intertemporal portfolio, and the price of this asset is straightforward and intuitive. When the asset price increases, we can generate profits, and if the asset price declines, we face potential losses. Hence, this variable can be seen as a variant of the target variable, implying a strong correlation between the target variable and this variable. The feature importance ranking also confirms this observation, as the intertemporal spread rate plays a significant role across all model groups.

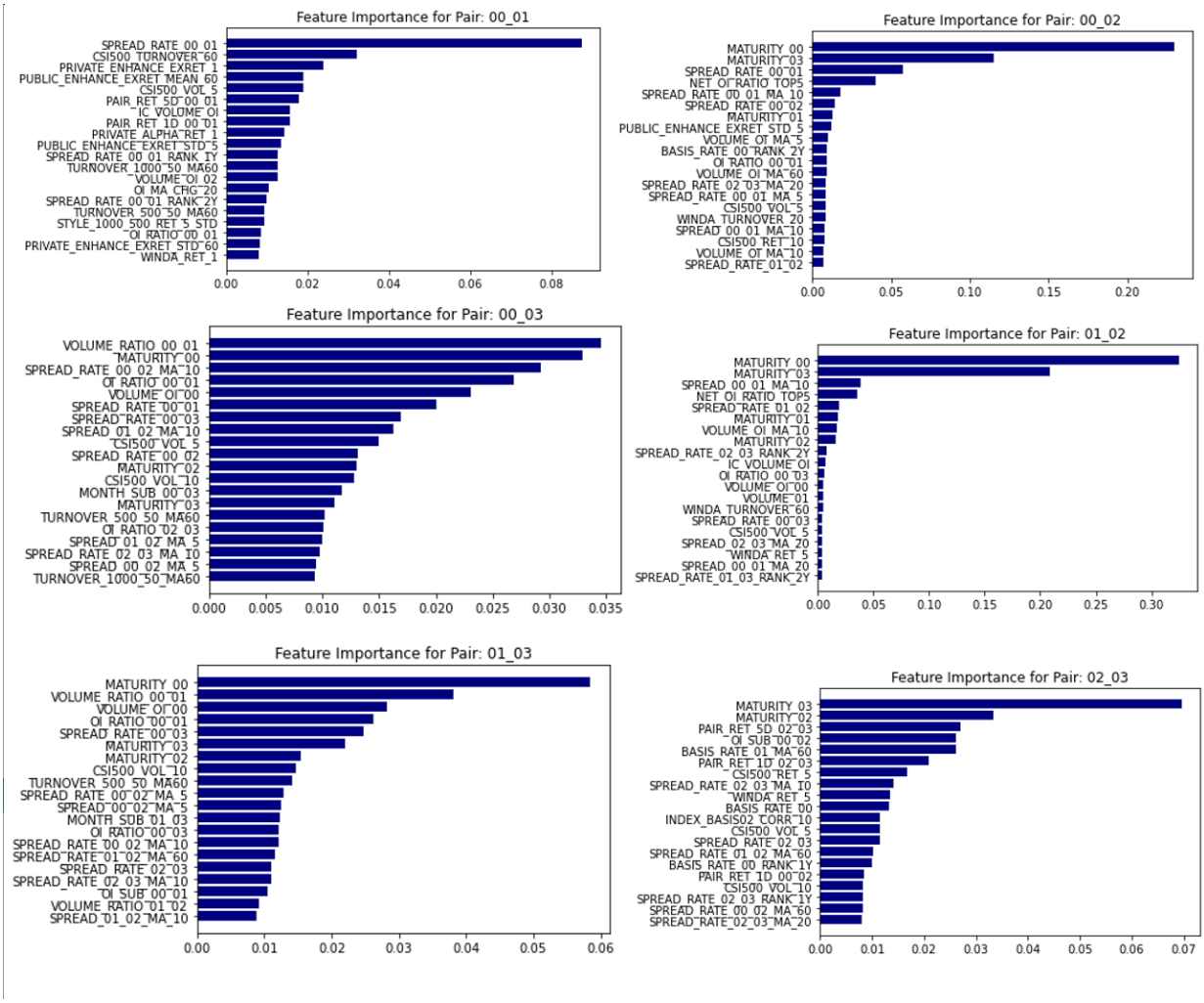


Table 60: Visualizing Top 20 influential features in Random Forest Model for each combination

However, upon closer examination, we notice some differences in the factors affecting the returns of the '00_01' combination (which is buying the long contract expiring in the next month and shorting the contract expiring in the next month) compared to other combinations. In this case, the prominent factors are more focused on short-term market indicators, such as WINDA_RET_1, PRIVATE_ENHANCE_EXRET_STD_60, STYLE_1000_500_RET_5_STD, TURNOVER_1000_50_MA60, and so on. These indicators lean towards measuring recent market sentiment and style, aligning well with our intuition that the returns of near-month contracts are significantly influenced by short-term market dynamics. This observation highlights the sensitivity of the '00_01' combination to short-term market variables, reinforcing the idea that near-month contracts are particularly influenced by recent market sentiment and style trends.

Recapping our iteration 2, when we used the C5.0 decision tree model to predict the ranking of returns for the six combinations in the next 5 days, we also examined the top ten nodes that partition the ranking, as shown in Table 16. It became evident that the features selected by each partition node of the decision tree somewhat differed from the important feature variables identified using the random forest model in Table 15. This discrepancy may arises from the nature of the C5.0 decision tree model, where the features chosen at each node can be considered important for making decisions within the context of that particular subtree. However, not all features selected at each node are globally significant for the entire model. Some features may only be utilized for making local decisions within specific branches of the tree.

00-01	00-02	00-03	01-02	01-03	02-03
MATURITY_03	MATURITY_03	OI_RATIO_00_01	MONTH_SUB_00_03	VOLUME_RATIO_00_01	MATURITY_03
MATURITY_01	VOLUME_RATIO_00_01	STYLE_1000_50_RET_60_STD	MATURITY_02	MATURITY_01	OI_RATIO_00_01
VOLUME_RATIO	PAIR_RET_1D_00_01	MATIRUTY_01	METURITY_01	PAIR_RET_1D_00_01	MATURITY_01
PUBLIC_ENHANCE_EXRET_MEDIAN	SPREAD_RATE_00_02	SPREAD_RATE_02_03	SPREAD_RATE_00_01_MA_5	CSI500_RET_10	OI_01
PRIVATE_ALPHA_RET	SPREAD_RATE_00_01_MA5	VOLUME_01	OI_RATIO_00_01	MATURITY_03	SPREAD_RATE_00_01
OI_00	STYLE_1000_300_RET_60	BASIS_RATE_02_RANK_2Y	INDEX_BASIS02_CORR_20	BASIS_RATE_02_RANK_1Y	SPREAD_RATE_00_01_RANK_2Y
PAIR_RET_1D	PRIVATE_ALPHA_RET_60	PAIR_RET_1D_00_01	WINDA_TURNOVER_20	MONTH_SUB_00_03	WINDA_TURNOVER_20
PUBLIC_ENHANCE_EXRET_STD	OI_02	VOLUME_SUB_00_01	SPREAD_RATE_02_03	INDEX_BASIS02_CORR_20	IC_CSI500_VOLUME_RATIO
BASIS_RATE	STYLE_500_50_RET_60	NET_OI_RATIO_TOP20	MONTH_SUB_00_02	STYLE_1000_50_RET_60_STD	OI_20
VOLUME_MA_CHG			PAIR_RET_1D_00_01		MATURITY_02

Table 16: Top 10 influential features in Decision Tree C5.0 in Iteration 2

8.3 Assessment and Evaluations

Based on the model outputs presented in Section 7.2.1.4 and 7.1.2.4, the accuracy rates for the Random Forest model and the Neural Network model are 20.2% and 19.5%, respectively. The result is shown in Figure 60.

```
In [9]: accuracy = (Y_test.idxmax(axis=1) ==
Y_pred.idxmax(axis=1))
   ...: accuracy_rate = accuracy.sum()/len(accuracy)
   ...: accuracy_rate
Out[9]: 0.20246478873239437

In [14]:
    ...: accuracy = (Y_test.idxmax(axis=1) ==
Y_pred.idxmax(axis=1))
    ...: accuracy_rate = accuracy.sum()/len(accuracy)
    ...: accuracy_rate
Out[14]: 0.1954225352112676
```

Figure 60: Accuracy comparison between two models

The observed accuracy rate, which is notably below our ideal benchmark of 50%, prompts an important question: must our model achieve perfect predictions to be considered effective? Consider a scenario where, among the six combinations, the top-ranked combination yields an actual annualized return of 1.01%, while the second-ranked combination offers a return of 1.00%. The annualized returns between these two combinations is not significantly different. According to our model, if we mistakenly place the second-ranked combination in the top position, the impact on our actual returns remains insignificant. Hence, we consider whether it is necessary to adjust our definition of predictive success.

8.3.1 Redefining "Accuracy Rate" to “Success Rate” in Dynamic Financial Market Environment

Given the considerable uncertainty prevailing in China's financial market and the profound transformations catalyzed by policies and global circumstances in recent years – encompassing alterations in trading systems and shifts in investor composition – utilizing financial market data spanning 2017 to 2021 as a training dataset to forecast returns from 2022 onwards introduces a pertinent issue: it fails to keep pace with the evolving market landscape. This dynamic makes the precise identification of the highest-yielding combinations in each prediction instance a challenging task. Aligned with our business objectives, we aim to demonstrate the value of proactive portfolio adjustments facilitated by our model, with the aim of surpassing passive strategies (e.g., prolonged holding without intervention) in terms of generating returns.

This objective suggests that we can redefine our model's success metric as the actual returns achieved from the calendar-spread portfolio selected by the model. These returns should surpass the median of returns obtained from the six benchmark portfolios. With this new benchmark, we have shifted our focus from accuracy rate to success rate. In other words, as long as the returns from our model-selected portfolio exceed the median returns of the six portfolios, we consider the model's selection successful. Therefore, after we change our benchmark, our success rate of Random Forest Model and neural network are now 55.1%, 54.75% respectively. The Python code used to compute the success rate is provided in Figure 61. The result of success rate for each algorithm is shown in Figure 62 and Figure 63. It means we have over about 55% probability to win the average performance of passive strategy.

```
success = 0
for date in Y_test.index:
    if Y_test.loc[date,Y_pred.idxmax(axis=1).loc[date]] > Y_test.loc[date,:].median():
        success += 1
success_rate = success/len(Y_test)
success_rate
```

Figure 61: Python code used to compute the success rate

```
   ...: success = 0
   ...: for date in Y_test.index:
   ...:     if Y_test.loc[date,Y_pred.idxmax(axis=1).loc[date]] >
Y_test.loc[date,:].median():
   ...:         success += 1
   ...: success_rate = success/len(Y_test)
   ...: success_rate
Out[4]: 0.551056338028169
```

Figure 62: Success rate for prediction in Random Forest Model

```
In [8]: success = 0
   ...: for date in Y_test.index:
   ...:     if Y_test.loc[date,Y_pred.idxmax(axis=1).loc[date]] >
Y_test.loc[date,:].median():
   ...:         success += 1
   ...: success_rate = success/len(Y_test)
   ...: success_rate
Out[8]: 0.5475352112676056
```

Figure 63: Success rate for prediction in Neutral Network

8.3.2 Backtesting of Model-Selected Strategy

In this section, we intend to further assess our model by evaluating how the model-selected strategy performs. Specifically, we will assess the performance of the model-selected portfolios for the duration spanning from 2022 to 2023 using an out-of-sample backtesting approach.

Our methodology involves rebalancing the portfolio every five trading days, with position adjustments made in accordance with the specific calendar spread pair recommended by our models. To access the effectiveness of our model and the performance of the selected portfolio combinations during this timeframe, we will employ six key performance indicators:

$$Cumulative\ return = V_{t+n} - V_t$$

$$Annualized\ return = Cummulative\ return \times \frac{50}{total\ weeks}$$

$$Annualized\ volatility = std(daily\ return) \times \sqrt{50}$$

$$daily\ return = V_{t+1} - V_t$$

$$MaxDrawdown = \max\left(abs(V_t - \max(V_1, \cdots, V_t))\right)$$

$$Sharpe\ ratio = \frac{Annualized\ return}{Annualized\ volatility}$$

$$Calmar\ ratio = \frac{Annualized\ return}{MaxDrawdown}$$

Cumulative return measures the total returns of a portfolio over a specific period. Annualized return calculates the average return of a portfolio per year. Maximum drawdown quantifies the largest loss a portfolio experiences from its highest point to its lowest point during a specified period. It assesses the worst-case scenario for a portfolio, indicating the maximum potential loss. Sharp ratio evaluates risk-adjusted returns by comparing the average excess return of the portfolio to its risk. The Calmar ratio assesses the annualized return achieved by a portfolio relative to the worst-case scenario (maximum drawdown).

These metrics are calculated as illustrated in Figure 64. Our approach entails a specific strategy based on the predicted combination returns. When the predicted return for a combination is positive, we take a long position in that combination, meaning we go long in the far-term contract and short the near-term contract. Conversely, if our prediction indicates a negative return, we take a short position in that combination to achieve a positive return. This entails going long in the near-term contract and shorting the far-term contract. By employing this strategy, we assess the performance of the investment strategies selected by the model during the backtesting period from 2022 to 2023.

As shown in Figure 64, we firstly define a function called performance_stat that computes those key performance metrics for a portfolio, which are cumulative return, annualized return, annualized volatility, maximum drawdown, Sharpe ratio, and Calmar ratio. After making predictions using a random forest model, we plot the backtesting curve for the portfolio by calculating the cumulative return.

```
#redefine performance metric
def performance_stat(value_data):
    daily_return = value_data.pct_change()
    cum_return = value_data[-1]/value_data[0] - 1
    annualized_return = cum_return * 50 / len(value_data)  #Simple interest
#     annualized_return = np.power(cum_return + 1, 250 / len(value_data)) - 1 # compound
    annualized_vol = daily_return.std() * np.sqrt(50)
    sharpe = (annualized_return - 0) / annualized_vol
    # maximum drawdown
    MaxDD = 0
    try:
        for i in np.arange(1,len(value_data)):
            MaxDD = min(MaxDD, value_data[i]/max(value_data[:i+1]) - 1 )
    except:
        MaxDD = None

    calmar = (annualized_return - 0) / abs(MaxDD)

    return pd.DataFrame([cum_return,annualized_return,annualized_vol,MaxDD,sharpe,calmar],
                        index = ['Cumulative return','Annual return','Annual vol','MaxDrawDown','Sharpe','Calmar'],
                        columns = ['Summary'])

## RF calculating backtestinbg curve
max_pair = Y_pred.abs().idxmax(axis=1)
date_list = []
ret_list = []
for i,date in enumerate(Y_test.index):
    if i%5 == 0:
        date_list.append(date)
        if Y_pred.loc[date,max_pair.loc[date]]>0:
            ret_list.append(Y_test.loc[date,max_pair.loc[date]]/50)
        if Y_pred.loc[date,max_pair.loc[date]]<0:
            ret_list.append(-Y_test.loc[date,max_pair.loc[date]]/50)
backtest = pd.DataFrame(index = date_list)
backtest['return'] = ret_list
backtest['value'] = 1 + np.cumsum(backtest['return'])
backtest.value.plot()

performance_stat(backtest.value)
```

Figure 64: Calculating the measures of the performance and plot the cumulative return curve in backtesting

8.3.2.1 Random Forest

Based on the steps outlined above, we have successfully computed six performance indicators for the portfolio under the Random Forest model. The result is shown in Figure 65. Additionally, we plot the model-selected forecasted results for the cumulative returns during the period from January 2022 to June 2023, as shown in Figure 66.

If our position is not leveraged, the Net Asset Value (NAV) of our strategy shows a steady upward trend, with the cumulative return reaching 8% from 2021 to June 2023. The annualized return is at 3.6%. During the out-of-sample period, the maximum drawdown is only -0.597%, which is quite favorable.

We can observe that the returns were more pronounced from 2021 to mid-2022, but the gains were relatively limited from mid-2022 to July 2023. This trend aligns with the conditions in the Chinese financial market during that period, characterized by low market confidence, reduced trading volumes, deteriorating liquidity, and suboptimal performance across various asset classes. Achieving objective returns became more challenging under these circumstances.

According to the trading rules and trading experiences, we can leverage our position up to 5x, which means the profit and loss are five times larger. After 5x leveraged, the annualized return can be as large as 18%. It satisfied our business goal stated at the very beginning.

```
    ...: performance_stat(backtest.value)
Out[25]:
                    Summary
Cumulative return  0.082527
Annual return      0.036196
Annual vol         0.016345
MaxDrawDown       -0.005970
Sharpe             2.214497
Calmar             6.062820
```

Figure 65: Performance Metrics in Random Forest Model

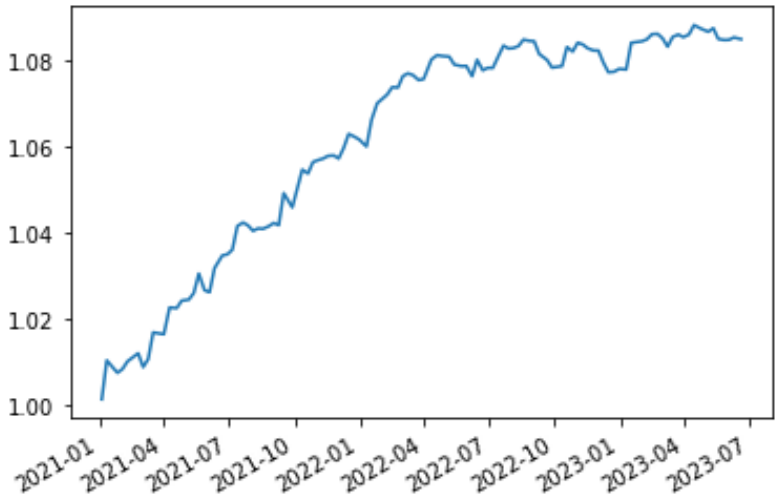


Figure 66: Cumulative return curve of the prediction in Random Forest Model

8.3.2.2 Neutral Network

Like we did in 8.3.2.1, we now compute six performance indicators for the portfolio under Neutral Network. The result presents in Figure 67. The cumulative return plot is presented in Figure 68.

From Figure 68, it's evident that the neural network's return curve exhibits significant fluctuations, indicating that the predictive performance of the neural network model is less stable. While the overall trend remains upward, similar to the random forest model, there are periods of noticeable declines in returns, such as from April 2022 to July 2022, and around April 2023. If our position is not leveraged, the Net Asset Value (NAV) of our strategy shows a cumulative return of 4.00% from 2021 to June 2023. The annualized return stands at 1.75%. During the out-of-sample period, the maximum drawdown is only -1.17%.

According to the trading rules and trading experiences, we can leverage our position up to 5x, which means the profit and loss are five times larger. After 5x leveraged, the annualized return can be as large as 8.75%. It also satisfied our business goal stated at the end.

```
    ...: performance_stat(backtest.value)
Out[22]:
                    Summary
Cumulative return  0.040044
Annual return      0.017563
Annual vol         0.015314
MaxDrawDown       -0.011708
Sharpe             1.146834
Calmar             1.500044
```

Figure 67: Performance Metrics in Neutral Network

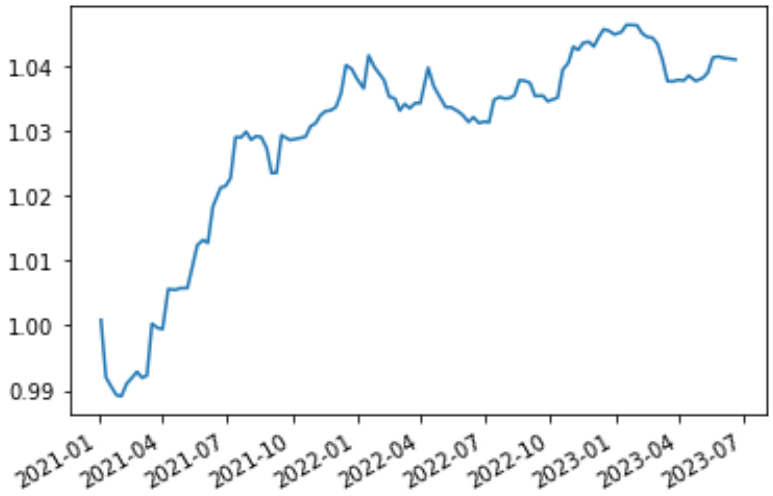


Figure 68: Cumulative return curve of the prediction in Neutral Network

8.3.2.3 Comparison Analysis on Backtesting Result Between Two Model

Based on the cumulative performance of the strategies, it is evident that the Random Forest model has achieved higher returns with a smaller maximum drawdown, indicating lower risk. The strategy implemented with this model shows a relatively smooth return curve. Even during the latter half of 2022, when market conditions became challenging due to

decreased market confidence, as reflected in declining trading volumes and volatility, the strategy continued to generate stable positive returns.

In contrast, the Neural Network model exhibits lower returns and greater volatility in its return curve. Notably, the performance of this model appears less stable, particularly in the past year. It is clear that, in our case, the Random Forest model outperforms the Neural Network model.

9 References

[1] Wang, H., & Kong, H. (2007). Utilizing cointegration to enhance the success rate of cross-period arbitrage in stock index futures [R]. Joint Securities Research Report.

[2] Alsayed, H., & Mcgroarty, F. (2014). Ultra-High-Frequency Algorithmic Arbitrage Across International Index Futures. Journal of Forecasting, 33(6si), 391-408.

[3] Tu, A. H., Hsieh, W. G., & Wu, W. (2016). Market Uncertainty, Expected Volatility And The Mispricing Of S&P 500 Index Futures. Journal of Empirical Finance, 35, 78-98

10 Disclaimer

I acknowledge that the submitted work is my own original work in accordance with the University of Auckland guidelines and policies on academic integrity and copyright.

I also acknowledge that I have appropriate permission to use the data that I have utilized in this project. (For example, if the data belongs to an organization and the data has not been published in the public domain, then the data must be approved by the rights holder.) This includes permission to upload the data file to Canvas. The University of Auckland bears no responsibility for the student's misuse of data.