Stock Factor Mining Based on Genetic Programming

Dongwei Li

Abstract

Through theoretical analysis and empirical research, this paper explores the application of genetic programming in factor mining of factor investing. Genetic programming is a heuristic formula evolution technique that simulates the process of genetic evolution in nature to generate formulas fitting a specific target. Genetic programming is well-suited for feature engineering and factor mining of factor investing. Since traditional factors are relatively simple technical indicators, genetic programming, enabling the construction of more complex factors, can bring new perspectives to factor mining.

This essay first discusses the underlying theory and technical detail of genetic programming. Then I build a framework for factor mining based on genetic programming and conduct empirical research. Finally, I analyze the effectiveness of mined factors based on factor analysis.

The framework proposed in this paper can be used in factor mining based on genetic programming, creating innovative insights in traditional factor investing.

1. Overview of Genetic Programming

Feature engineering plays an essential role in applied machine learning. However, traditional feature engineering methods require professional knowledge and understanding of the relevant field. Additionally, effective features with complex structure are difficult to discover. Genetic programming can effectively address these challenges.

1.1 Fundamental principle of Genetic Programming

Genetic programming simulates the process of natural selection, where in each generation all features evolve randomly like genetic mutations, and those best adapted are selected to survive into the next generation.

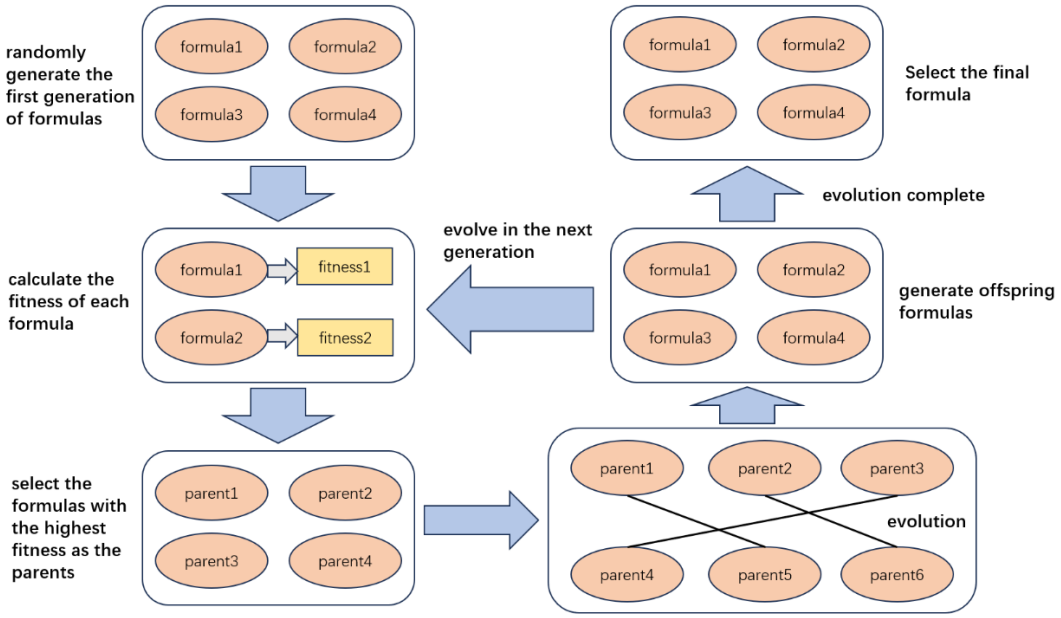


Fig. 1. The process of genetic programming

Specifically, genetic programming randomly generates a series of primitive formulas to be the first generation of formulas. For each generation, genetic programming calculates the fitness of each formula and selects suitable features as the parents for the next generation's evolution. These selected parents evolve through various methods to form different offspring formulas, which then undergo subsequent rounds of evolution. With an increasing number of generations, the formulas continuously reproduce, mutate, and evolve, thereby progressively approximating the true nature of the data distribution.

1.2 Genetic Programming and Factor Investing

In the field of factor investing, factor mining means creating factors that can explain future returns, which is similar to feature engineering. Traditional factors are typically technical indicators with underlying economic meaning. Traditional factor mining involves designing factors based on market laws and investment experience. Over time, factor investing has exhibited significant homogeneity issues. The core challenge is to discover novel and effective factors. Genetic programming creatively addresses this issue through feature engineering. Contrary to traditional factor mining methods, genetic programming breaks through the limitations of human thought, uncovering hidden factors with complex structures that are difficult to construct through human intellect.

1.3 Fitness of Genetic Programming

Fitness is the criterion used to judge whether a formula meets a given target and is a key reference in the evolution. The selection of fitness should be based on the specific model and target. For factor mining, IC is the most suitable fitness measure since it is a widely used in factor analysis.

1.4 Methods of formula evolution

Common evolution methods include crossover, subtree mutation, point mutation, and hoist mutation.

1) Crossover

Crossover selects two formula trees A and B and randomly chooses subtrees A1 and B1 from each. By replacing A1 with B1, the evolved next-generation formula tree A' is obtained.

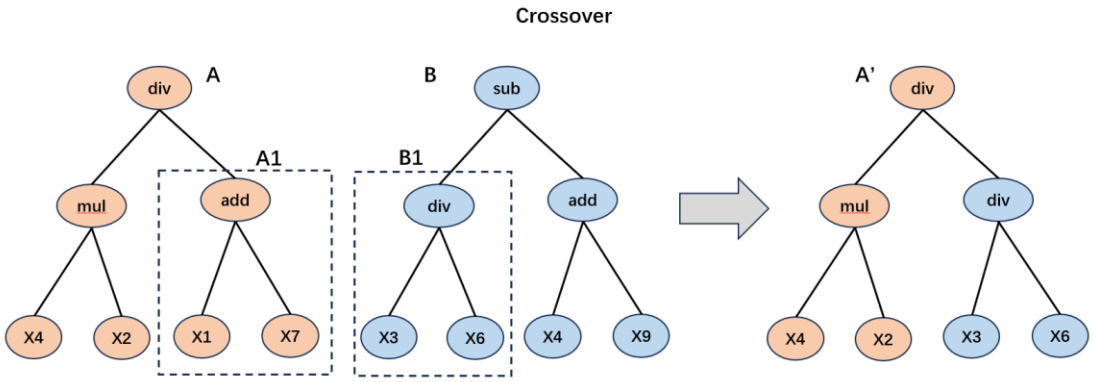


Fig. 2. crossover

2) subtree mutation

Subtree mutation selects a formula tree A and its subtree A1 randomly. By replacing A1 with a randomly generated formula tree C, the evolved next-generation formula tree A' is obtained.

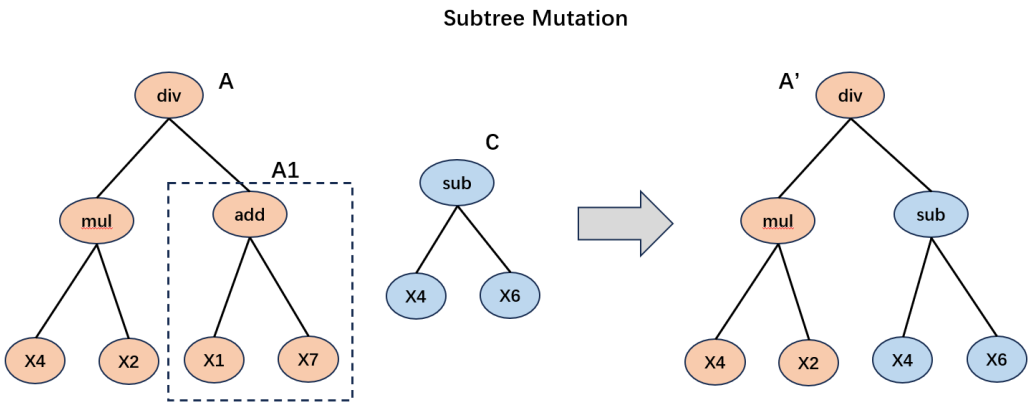


Fig. 3. subtree mutation

3) point mutation

Point mutation randomly selects a node in formula tree A and replaces it randomly to obtain the evolved next-generation formula tree A'.

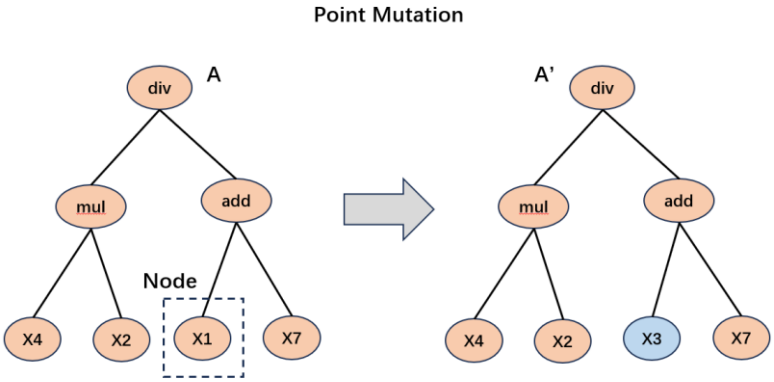


Fig. 4. point mutation

4) hoist mutation

Hoist mutation removes certain leaves or nodes from a formula tree A to simplify it.

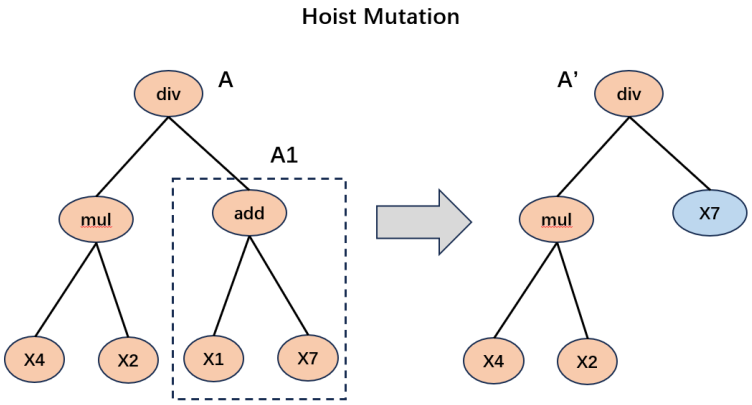


Fig. 5. hoist mutation

2. Factor Mining Based on Genetic Programming

2.1 Introduction to gplearn

The package gplearn is one of the most mature Python genetic programming packages which allows for flexible adjustment of hyperparameters and the customization of complex formulas.

2.2 Selection of Training and Testing Data

This paper selects the daily data of S&P 500 constituent stocks from 2017 to 2019 as the training set, and uses data from 2021 to 2022 as the test set to perform factor analysis on the mined factors to verify their effectiveness. The original features include OHLCV (Open, High, Low, Adjusted Close, Volume). The target for the model is 5-day

returns.

2.3 Data Processing and Model Setting

Firstly, given the strong time-series correlation of financial data, 1, 3, 5, 10, and 20-day lag values and rolling averages of initial features are added to the model. Observations with missing values are removed.

```
def _cube(data):
    return np.square(data)*data

def _square(data):
    return np.square(data)
```

```
cube = make_function(function=_cube, name='cube', arity=1)
square = make_function(function=_square, name='square', arity=1)
```

```
user_function = [square, cube]
```

Secondly, to capture the non-linear relationship between factors and returns, square and cubic operators are customized and added. The updated function set is shown in the following table.

Table 1 function set

function	formula
add	$X_1 + X_2$
sub	$X_1 - X_2$
mul	$X_1 \times X_2$
div	$X_1 \div X_2$
sprt	$\sqrt{X_1}$
log	$ln(X_1)$
inv	$1/X_1$
max	$\max(X_1, X_2)$
min	$\min(X_1, X_2)$
sqaure	${X_1}^2$
cube	${X_1}^3$

Lastly, the hyperparameters for the genetic programming are shown in the following table.

Table 2 hyperparameters

Hyperparameters	Explanation	Values
generations	The number of generations for evolution.	3
population_size	The number of formulas generated in each generation.	3000
tournament_size	The size of formulas which are selected in each generation.	30
metric	Fitness metric.	RankIC

p_crossover	The probability of crossover.	0.4
p_subtree_mutation	The probability of subtree mutation.	0.01
p_hoist_mutation	The probability of Hoist mutation.	0
p_point_mutation	The probability of point mutation.	0.01
p_point_replace	The probability of each parent node undergoing mutation in point mutation.	0.4

```
fields = X_train.columns
function_set = init_function + user_function
population_size = 3000
generations = 3
random_state= 42
est_gp = SymbolicTransformer(
                            feature_names=fields,
                            function_set=function_set,
                            generations=generations,
                            metric='spearman',   # i.e. RankIC
                            population_size=population_size,
                            tournament_size=30,
                            random_state=random_state,
                            verbose=2,
                            parsimony_coefficient=0.0001,
                            p_crossover = 0.4,
                            p_subtree_mutation = 0.01,
                            p_hoist_mutation = 0,
                            p_point_mutation = 0.01,
                            p_point_replace = 0.4,
                            n_jobs = 6
                           )
```

2.4 Model Results

This genetic programming consists of three generations. The first generation of formulas was randomly generated, with a longer average length and very low average fitness. The second generation of formulas saw a reduction in average length and a significant improvement in average fitness. The third generation of formulas continued to evolve based on the second generation, with a slightly longer average length and a decrease in average fitness.

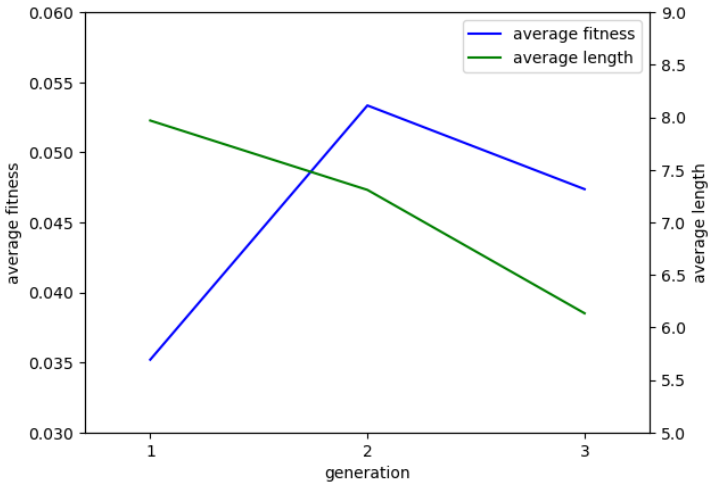


Fig. 6. the evolution of length and fitness

The top ten factors with the highest fitness after three generations of evolution in genetic programming are shown in the table below.

Table 3 factors

factors	fitness	expression
factor_1	0.0597	sub(div(Open_Lag1, Adj Close_Lag20), inv(High_Lag3))
factor_2	0.0591	sub(log(min(div(Open_Lag3, Adj Close_Lag20), min(High_Lag5, Open_Lag5))), inv(High_Lag3))
factor_3	0.0594	inv(div(Low_Lag5, sub(Low_Lag3, Adj Close_Lag20)))
factor_4	0.0592	log(min(div(Open_Lag3, Adj Close_Lag20), div(Open_Lag1, Adj Close_Lag20)))
factor_5	0.0593	sub(div(High_Lag1, Adj Close_Lag20), inv(High_Lag3))
factor_6	0.0588	sub(inv(sub(Open_Lag20, div(Open_Lag5, Adj Close_Lag20))), inv(div(Adj Close_Lag20, High_Lag1)))
factor_7	0.0588	log(min(div(Open_Lag3, Adj Close_Lag20), sub(div(High_Lag3, Adj Close_Lag20), inv(High_Lag3))))
factor_8	0.0590	inv(sub(div(Open_Lag3, Adj Close_Lag20), inv(High_Lag3)))
factor_9	0.0588	sub(sub(div(High_Lag3, Adj Close_Lag20), inv(High_Lag3)), inv(High_Lag3))
factor_10	0.0588	sub(div(High_Lag3, Adj Close_Lag20), inv(Open_Lag1))

The average fitness of these ten factors reached 5.91%. Therefore, they have been proven to be effective factors on the training set.

The correlation coefficients of these ten factors are shown in the following graph. Some factors have similar formulas, resulting in a higher overall correlation among these ten factors.

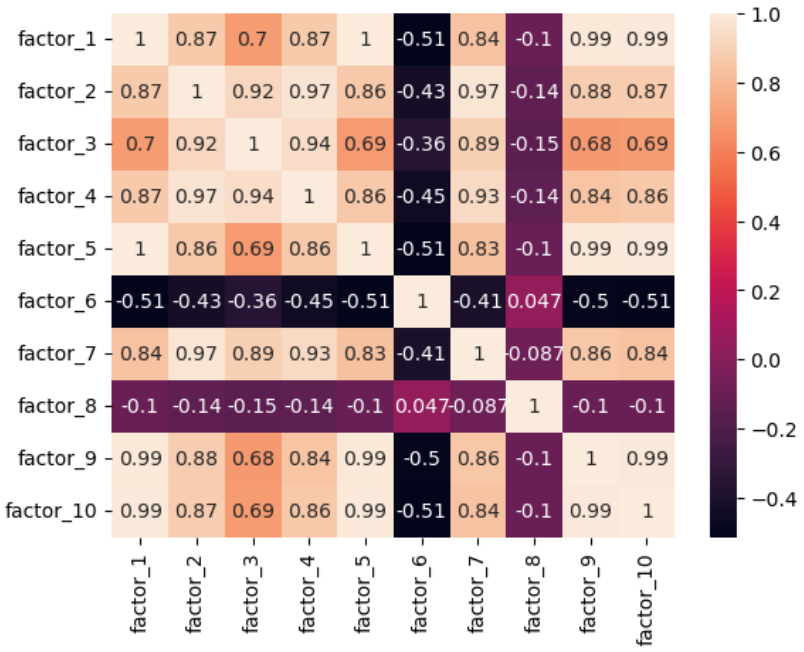


Fig. 7. factor correlation

This paper selected factor1, factor3, and factor6, three factors with significant structural differences and low correlation as final results.

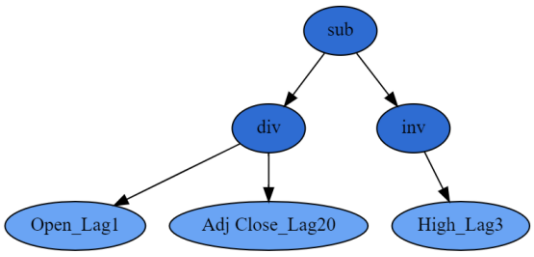
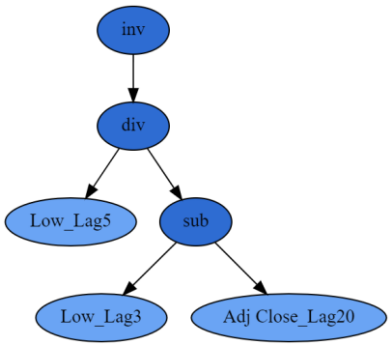
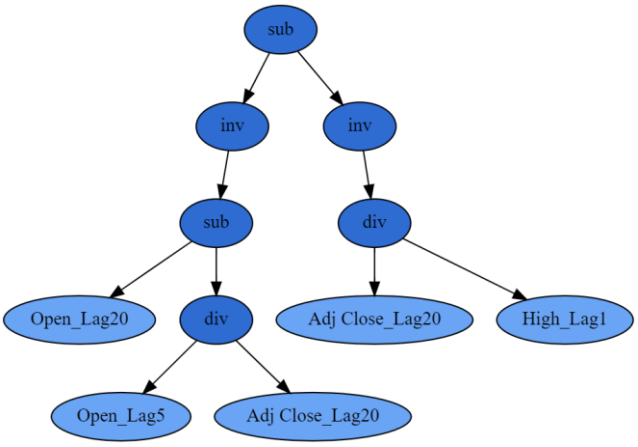
3. Factor Analysis

This section conducts factor analysis on factor1, factor3, and factor6 within the backtesting period of 2021-2022, including Information Coefficient and group backtesting.

3.1 Structure Analysis

The detailed calculation formula for the three factor is shown in the following figures.

Table 4 factors

factor	formula structure
factor1	
factor3	
factor6	

It's evident that both factor1 and factor3 are designed to capture the momentum effect of stock prices. However, the structure of factor6 is more complex, making it difficult to obtain an intuitive economic interpretation.

3.2 IC analysis

The IC for the three factors is shown in the table below. It's evident that factor1 and factor2 are inverse factors, while factor6 is a positive factor. The absolute values of short-term IC are relatively small, but they steadily increase over time. This indicates that all three factors are effective in describing long-term returns.

Table 5 IC

factor	factor1				factor2				factor3			
term	1D	5D	10D	21D	1D	5D	10D	21D	1D	5D	10D	21D
IC Mean	-0.003	-0.008	-0.011	-0.03	-0.002	-0.007	-0.011	-0.026	0.002	0.009	0.012	0.031
IC Std.	0.214	0.198	0.188	0.188	0.214	0.2	0.194	0.195	0.212	0.196	0.186	0.185
Risk-Adjusted IC	-0.012	-0.041	-0.061	-0.159	-0.008	-0.034	-0.055	-0.134	0.009	0.047	0.065	0.169
t-stat(IC)	-0.254	-0.866	-1.296	-3.4	-0.172	-0.726	-1.182	-2.865	0.186	0.995	1.38	3.612
p-value(IC)	0.799	0.387	0.196	0.001	0.864	0.468	0.238	0.004	0.853	0.32	0.168	0
IC Skew	-0.03	-0.056	-0.213	-0.128	-0.01	-0.093	-0.216	-0.162	0.062	0.076	0.242	0.143
IC Kurtosis	-0.091	-0.165	-0.037	-0.277	-0.019	-0.216	-0.231	-0.277	-0.043	-0.162	-0.02	-0.228

3.3 Group backtesting

This part conducts backtesting by dividing stocks into five groups based on the factor values. The group backtesting returns for each factor are shown in the following graphs.

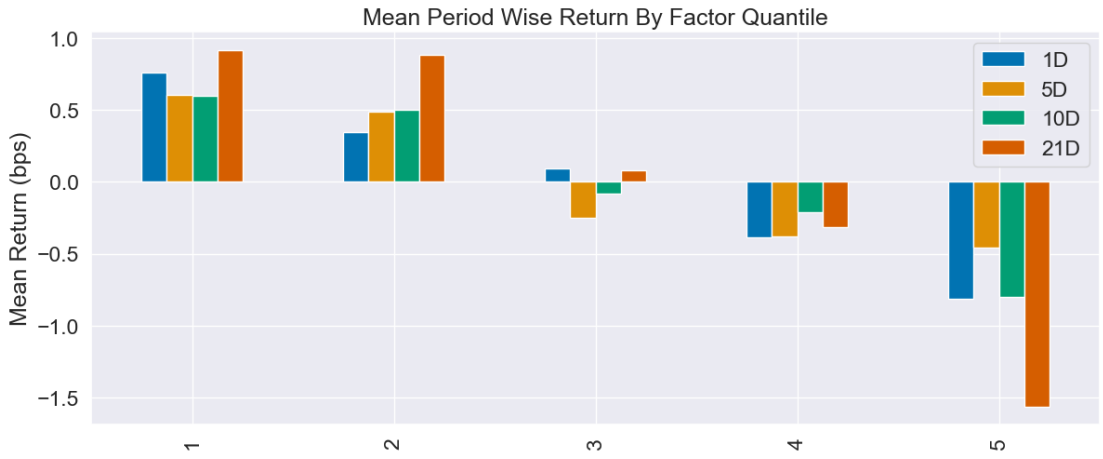


Fig. 8. group backtesting of factor1

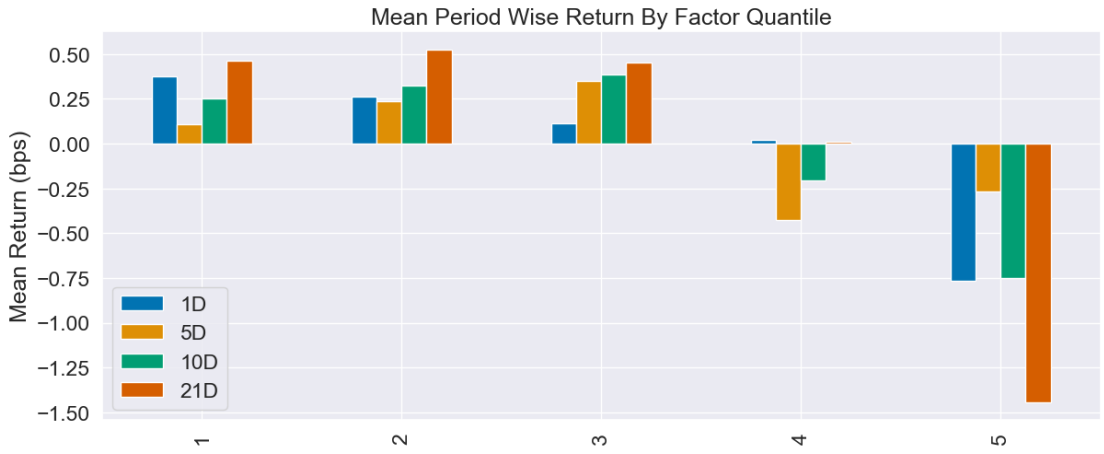


Fig. 9. group backtesting of factor2

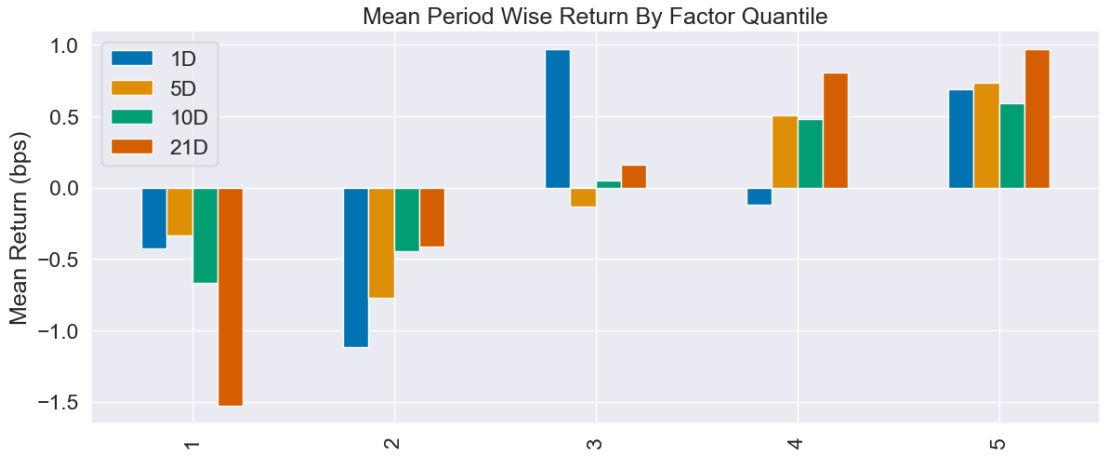


Fig. 10. group backtesting of factor3

It's evident that factor1 has the most stable performance in both the short and long-term. Factor3 and factor6 exhibit fluctuations in short-term performance. However, all three factors can generate stable long-term returns, which is consistent with the results of the IC analysis.

4. Conclusion

This paper employs genetic programming for factor mining in factor investing. Factor analysis is carried out using IC and group backtesting methods, which validate the effectiveness of the three mined factors.

Genetic programming is a flexible methodology that can discover a variety of novel factors by using new features and adding new operators. This paper demonstrates the detailed process and constructs a foundational framework of using genetic programming in factor mining.

Appendix

The code for this essay can be found in Colab:

https://colab.research.google.com/drive/1MFJg_bsAgZYmuT1dZSQyVhQEfr56ODjc?usp=sharing