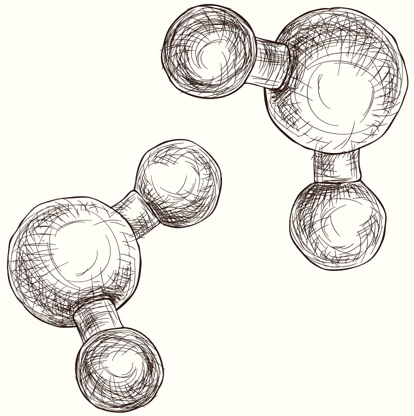
White Paper

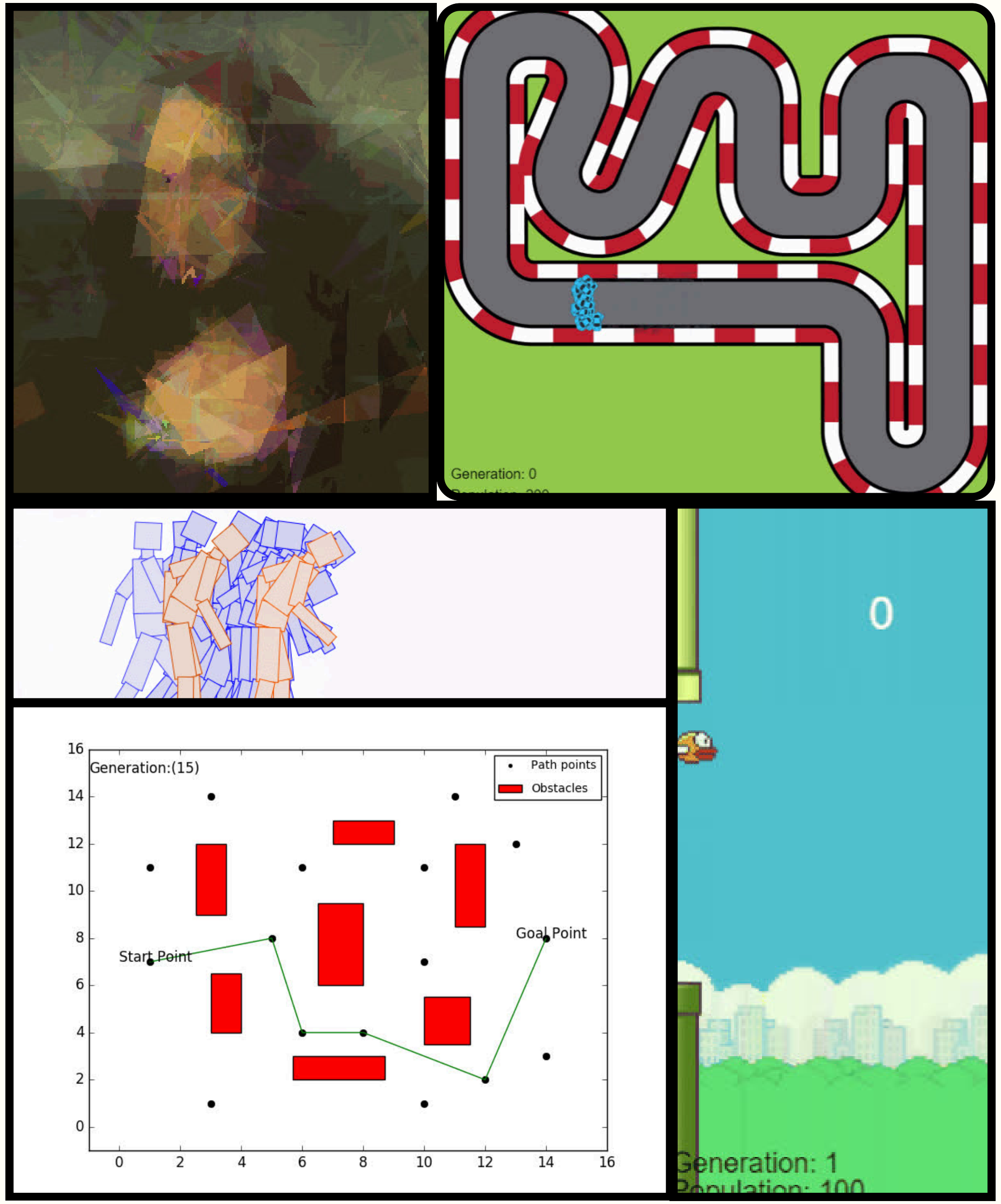
The αlpha Engine

QUANT CLUB

GENETIC ALGORITHM

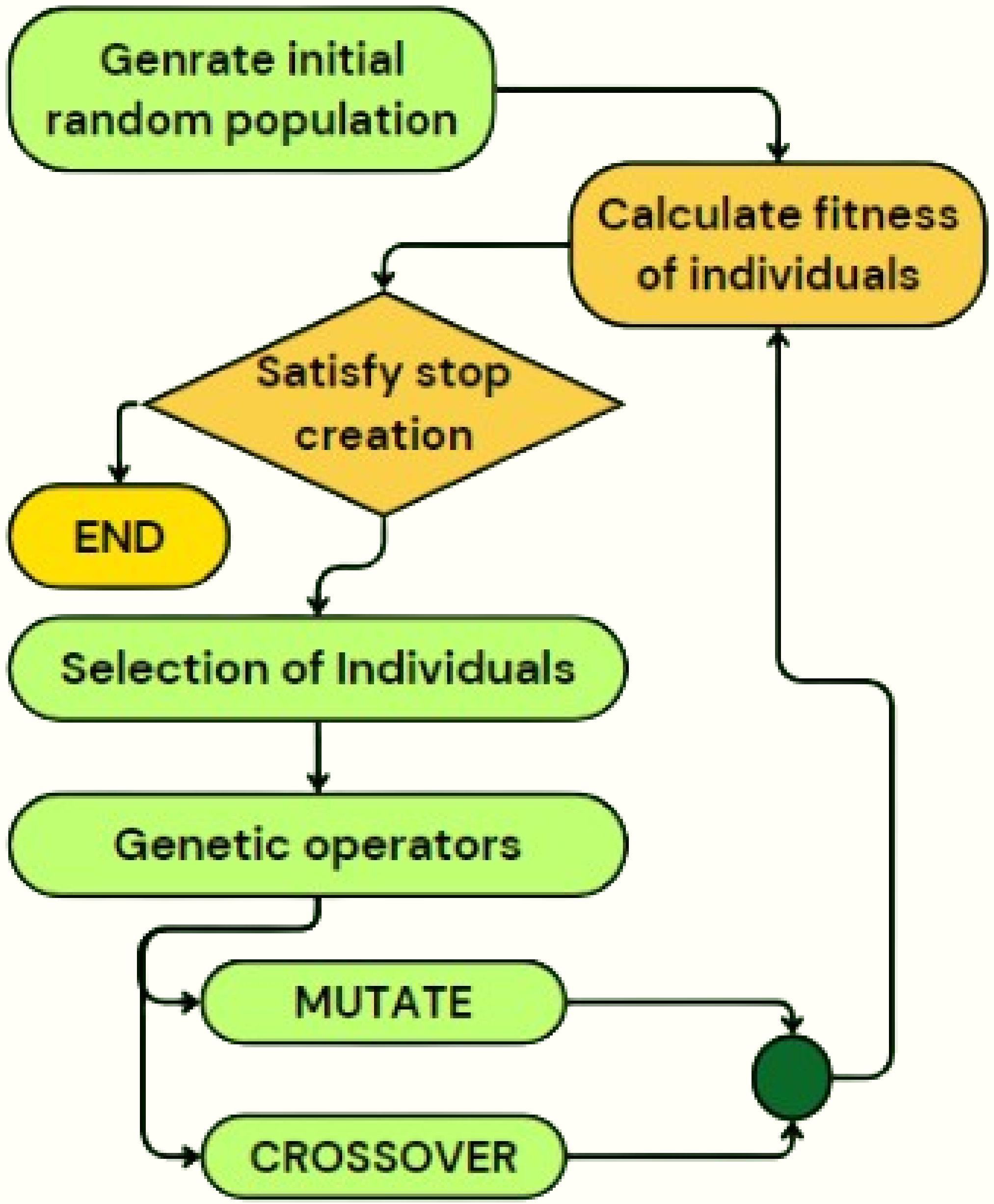


Genetic algorithms mimic natural selection to optimize solutions. Starting with a population of potential answers, fitter individuals are selected and their genetic information is combined through crossover and mutation. This iterative process leads to improved solutions for complex problems in various fields

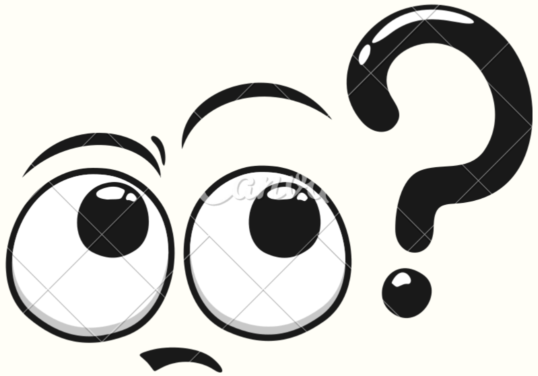


WORKFLOW

*In a genetic algorithm, a population of solutions is evaluated based on fitness, and the fittest individuals are selected for reproduction. Through crossover and mutation operations, their genetic information combines to form a new generation. This iterative process, akin to natural selection, refines the population, gradually converging towards optimal solutions. Just as the **fittest survive** in nature, the algorithm ensures the persistence of superior solutions, making it a powerful tool for optimization and search tasks*



GENETIC ALGORITHM + “FINANCE”



Genetic algorithms are extensively applied in finance, notably for ***optimizing portfolio weights*** *to balance returns and risk. These algorithms represent portfolios as individuals, evolving them through selection and recombination. In technical analysis,* ***they enhance indicator performance by dynamically adjusting look-back periods for improved trading signals****. Additionally, genetic algorithms contribute to* ***stock selection****, evaluating various criteria and evolving a population of potential investments for optimal choices. Their adaptability and optimization make them crucial tools in addressing complex financial challenges,* ***offering solutions for portfolio management, technical analysis parameter tuning, and intelligent stock selection***

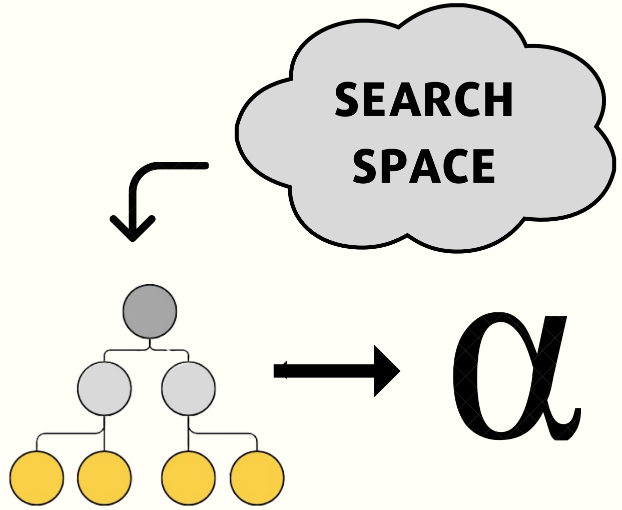
S&P BSE SENSEX

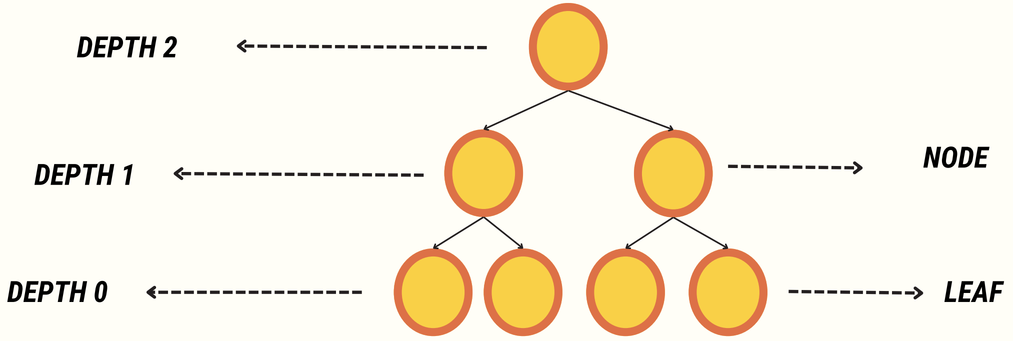
Nifty50

BUT HOW GENETIC FOR ALPHA GENRATION!

Alpha *is a finance metric using market indicators to predict security or portfolio performance. It assigns weights based on perceived contribution to returns, guiding portfolio allocation. Investors utilize these weights to outperform the market or a benchmark index.*

In our approach, we leverage genetic algorithms to optimize alphas in trading strategies. We construct a search space comprising operators and data fields, forming trees that represent alpha expressions. Through mutation and crossover, we explore this space to refine alphas, mimicking natural selection processes to efficiently identify optimal trading strategies.





LETS DEEP DIVE INTO THE POCESS

SEARCH SPACE

terminal_values = ["close", "open", "high", "low", "vwap", "adv20", "volume", "cap", "returns", "dividend"]

ts_operators = ["ts_zscore", "ts_rank", "ts_arg_max", "ts_arg_min", "ts_backfill", "ts_delta", "ts_ir", "ts_mean","ts_median", "ts_product", "ts_std_dev"]

binary_ops= ["add", "subtract", "divide", "multiply", "max", "min"]

unary_ops = ["rank", "zscore", "winsorize", "normalize", "rank_by_side", "sigmoid", "pasteurize", "log"]

ts_ops_values = ["20", "40", "60", "120", "240"]

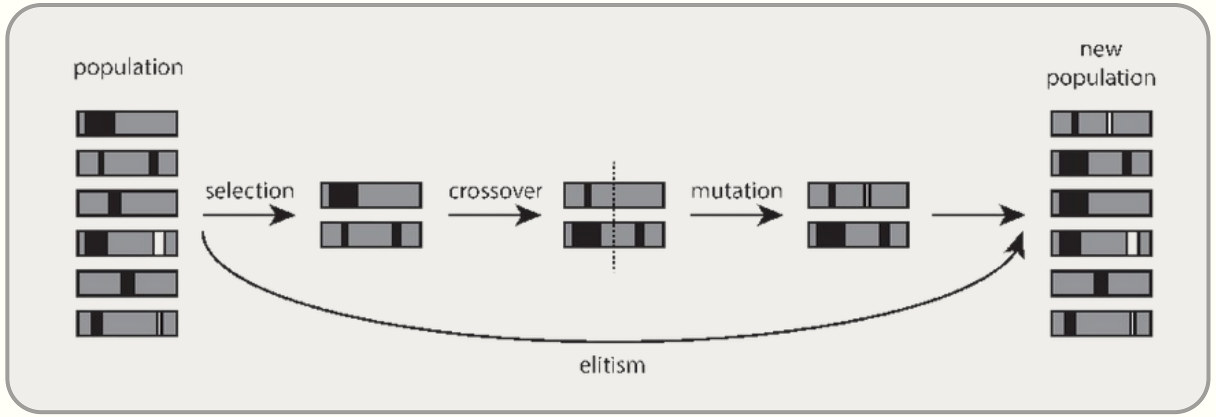
When creating an alpha tree, we randomly sample values from the search space to initialize node values, while leaf values are set using terminal values. This process establishes an initial structure for the tree, incorporating diverse solutions and ensuring exploration across the search space

ELITISM

The usage of the concept of elitism can help reduce bloat in populations and thus speed up the process of finding high quality candidates. The idea is quite simple: we choose the best performing candidates of the previous generation and add them to the next generation that we are trying to generate.

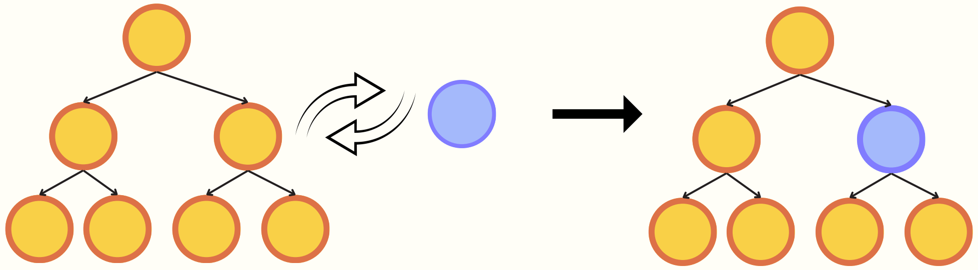
It is important that we do not change the candidate at all. By doing this we make sure that our evolutionary process does not accidentally forget about the best performing candidates.

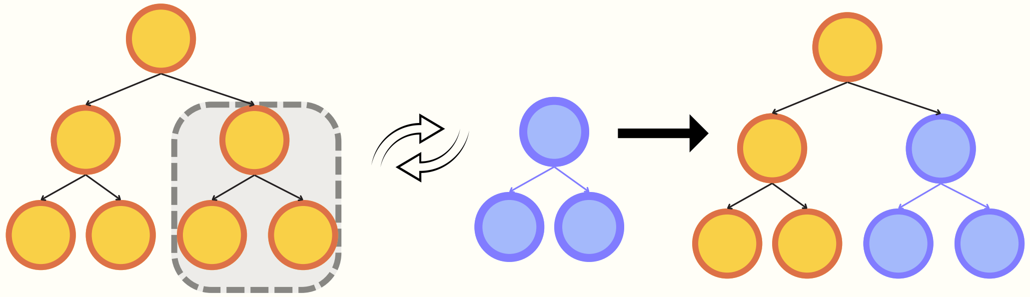
In our Parent vs Off-spring approach of evolving a generation, we only make use of elitism in cases where we can't come up with candidates that are not contained in our next generation ourselves. We know that the candidates we introduce into the next generation with the elitism approach will not be worse (in terms of fitness) than all the candidates in the last generation.



MUTATION AND CROSSOVER

In the course of our evolutionary process we will mutate and crossover different functions to create new off-springs ***that might perform better than their predecessors.***





MUTATION :

A mutation follows the idea of genetic mutations as they happen in the real world . With a set probability we introduce some random changes in the genetic representation of a function

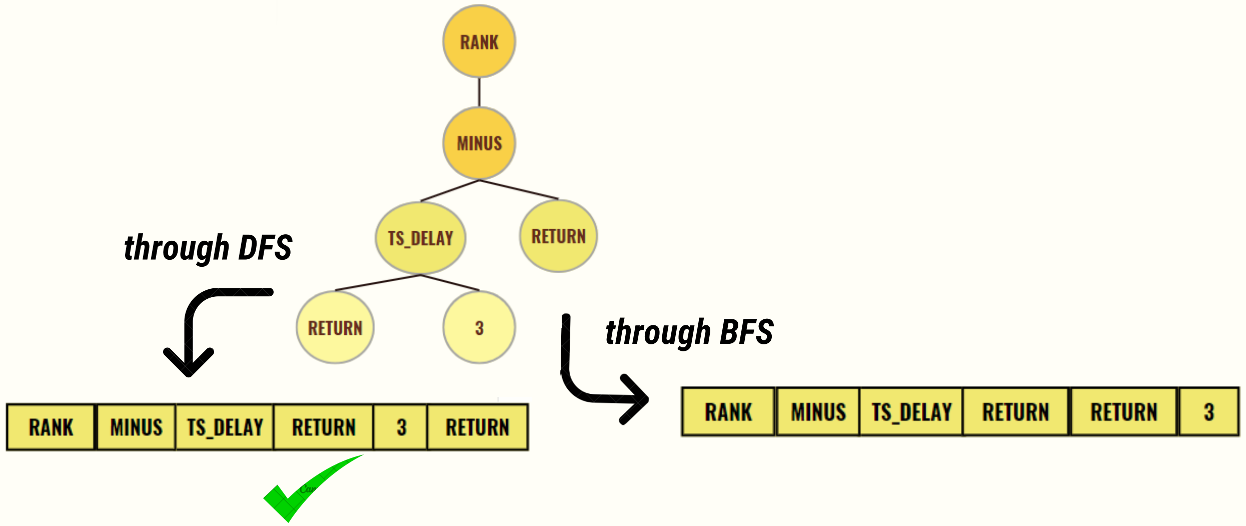
CROSSOVER :

Genetic programming swaps random sub-trees between high-performing parent and donor candidates to produce offspring, aiming for improved performance through trait combination akin to genetic crossover.

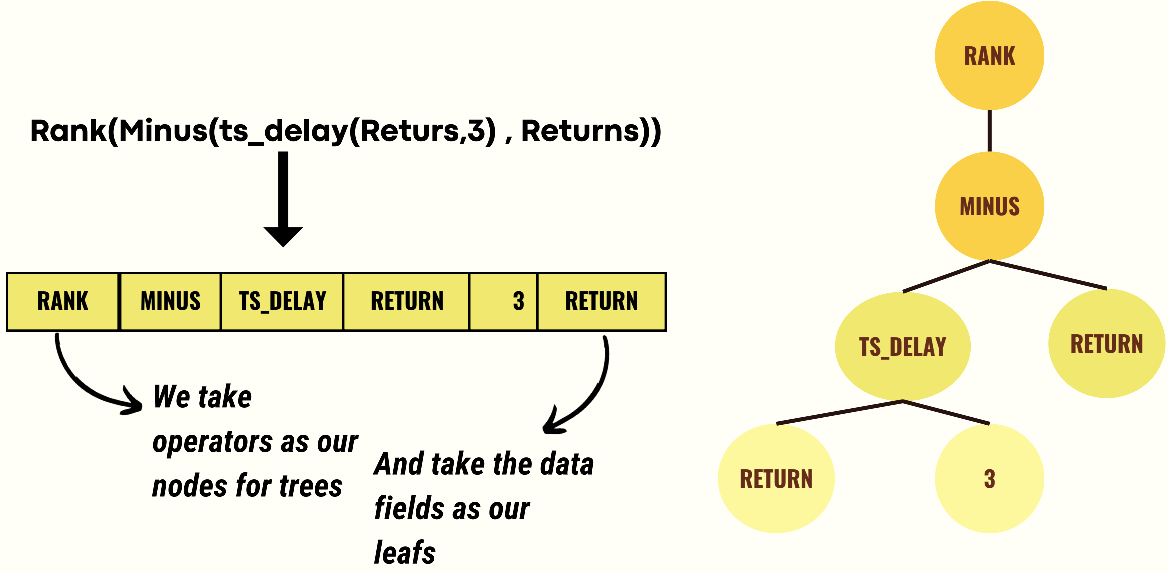
In our context, mutation and crossover aid in exploring the data field and diverse operators, enhancing the search for optimal solutions through genetic variation and recombination.

CONVERTING AN TREE TO ALPHA

For converting a tree-structured alpha into a string, I tried both the BFS(Breadth First Search) and DFS (Depth First Search) algo.



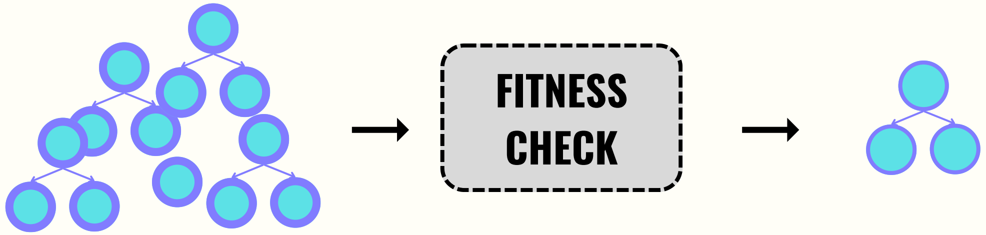
CONVERTING AN ALPH TO TREE



FITNESS FUNTION

The fitness function evaluates the performance of potential solutions in a genetic algorithm, influencing their likelihood of reproduction and shaping the evolution towards better solutions.

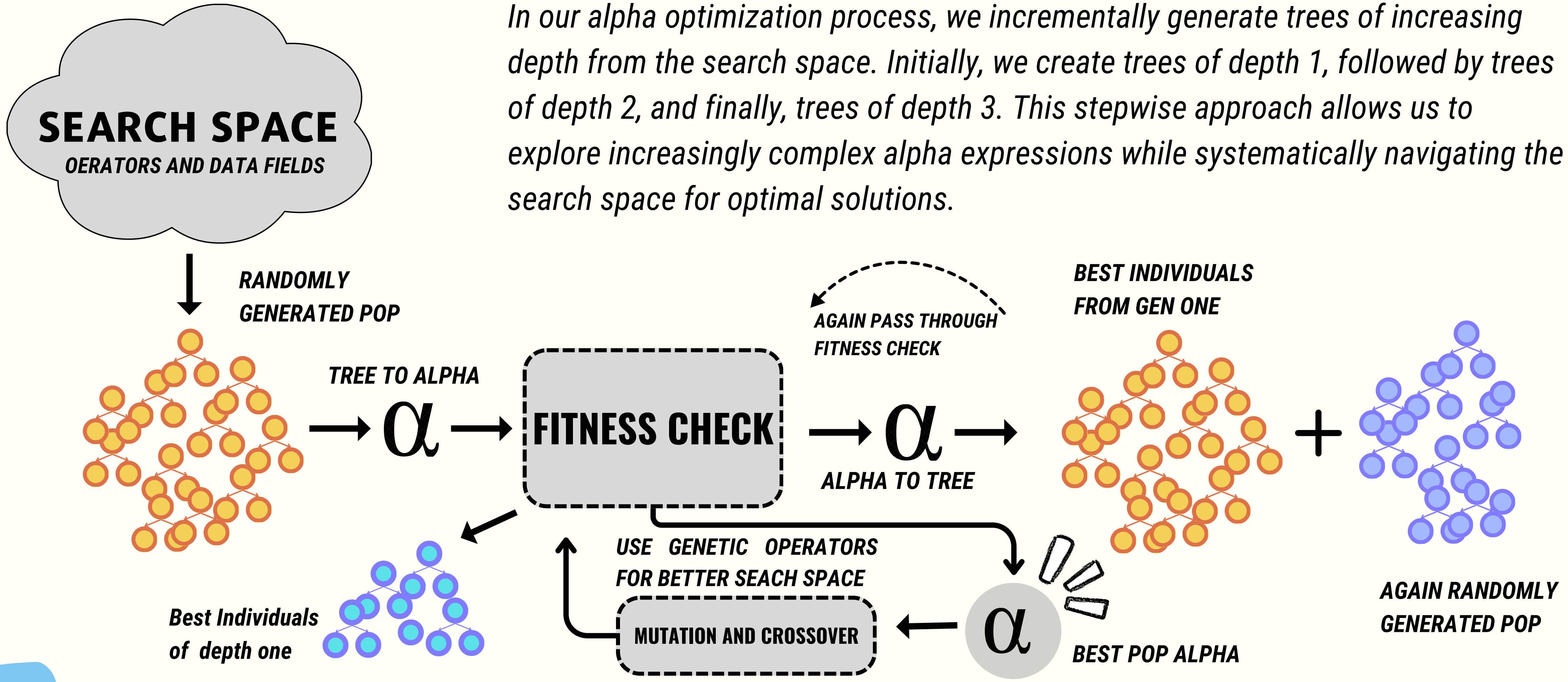
$$\text{FITNESS FUNTION} = \frac{\text{SHARPE} * \text{FITNESS} * \text{RETURNS}}{\text{DRAWDOWN} * \text{TURNOVER}^2 + 0.001}$$

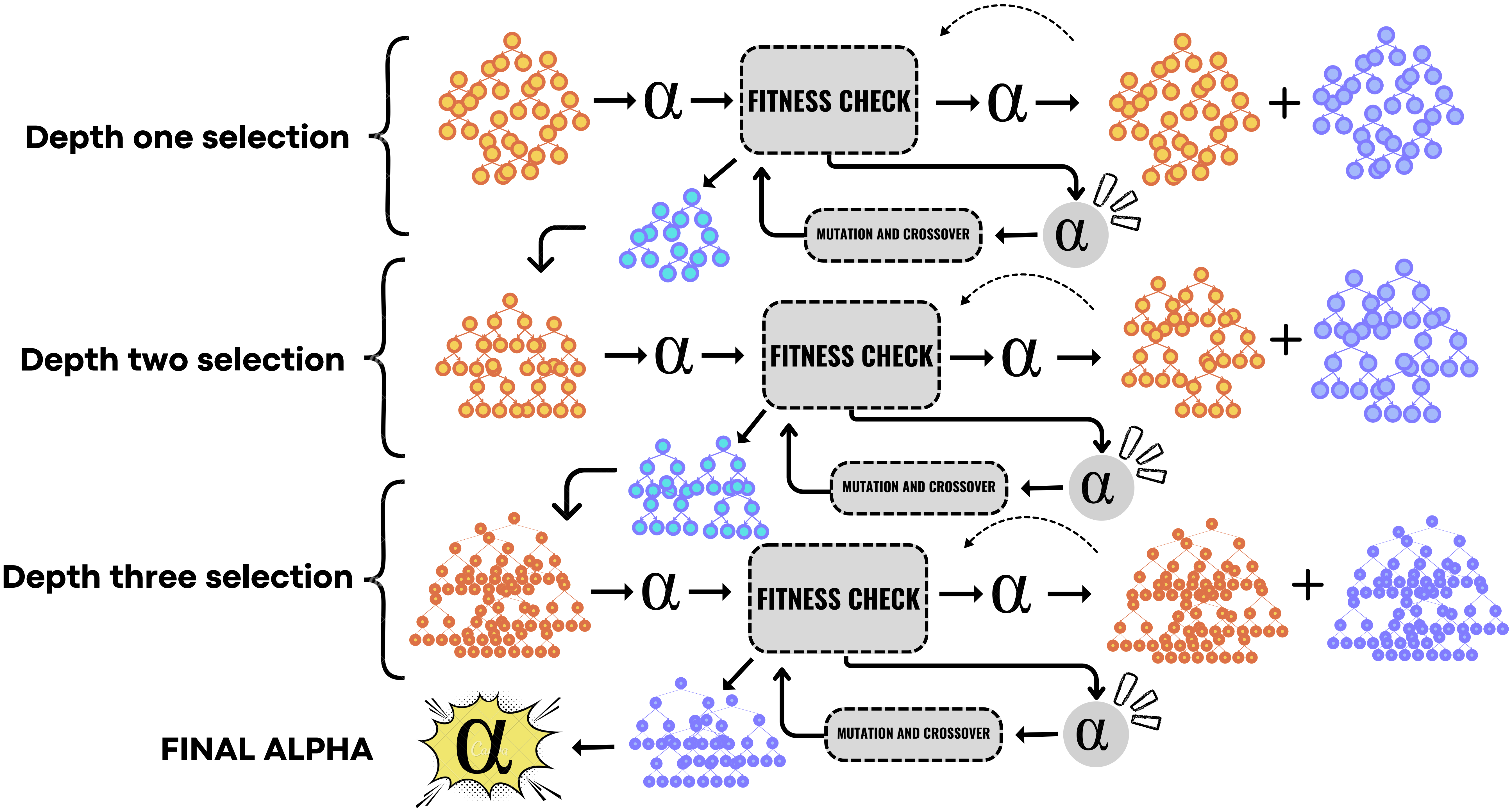


In our case, the fitness function selects the top individuals from the population by considering parameters essential for success in the test, tailored to the requirements an alpha must meet. It serves as a criterion for evaluating and prioritizing solutions, ensuring those with the most promising traits advance in the evolutionary process.

THE PROCESS OF FINDING THE BEST ALPHA

In our alpha optimization process, we incrementally generate trees of increasing depth from the search space. Initially, we create trees of depth 1, followed by trees of depth 2, and finally, trees of depth 3. This stepwise approach allows us to explore increasingly complex alpha expressions while systematically navigating the search space for optimal solutions.



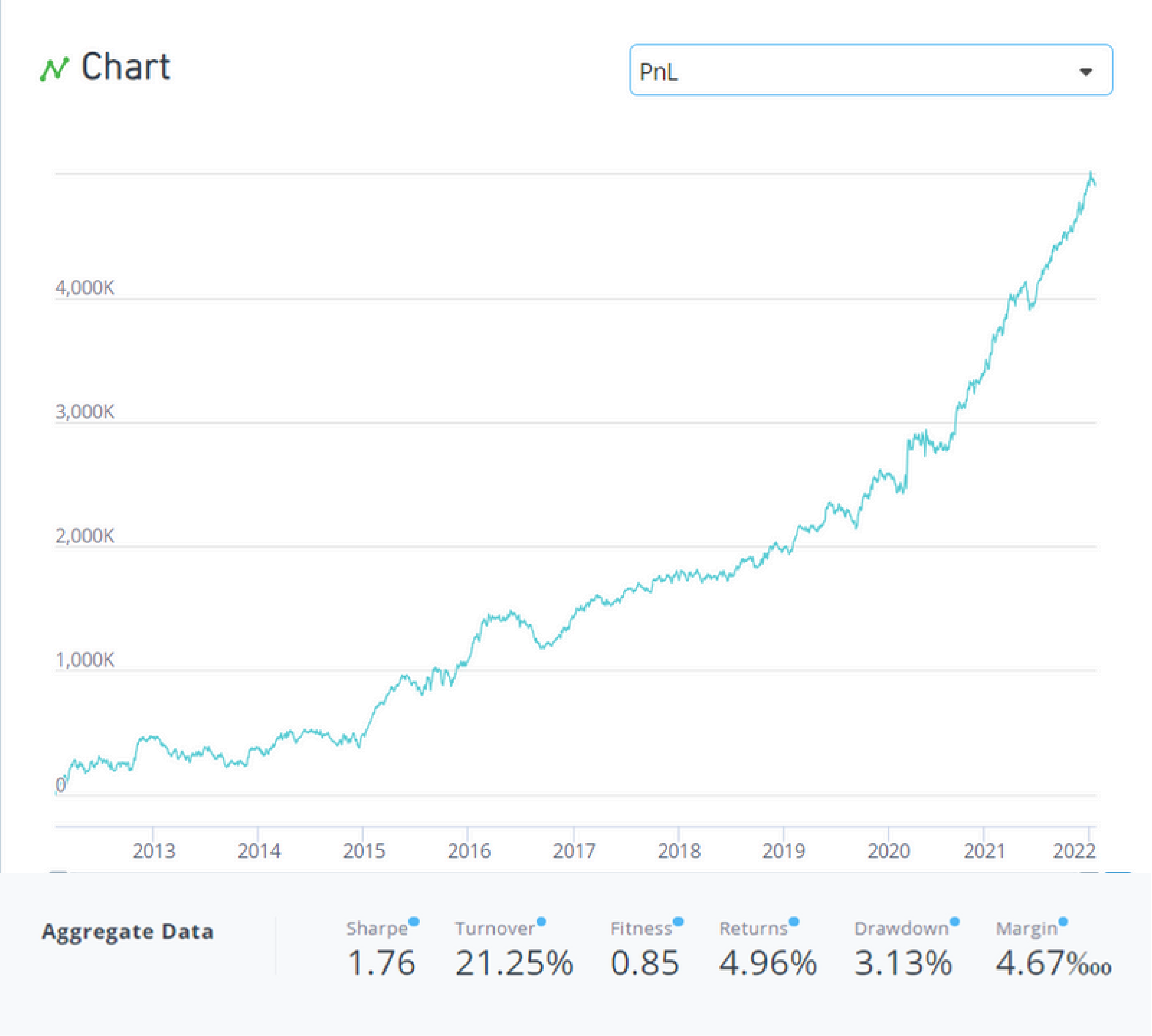


Depth one selection
α
FITNESS CHECK
α
+
MUTATION AND CROSSOVER
α
Depth two selection
α
FITNESS CHECK
α
+
MUTATION AND CROSSOVER
α
Depth three selection
α
FITNESS CHECK
α
+
MUTATION AND CROSSOVER
α
FINAL ALPHA
α

RESULTS

Rank(Minus(ts_delay(Returs,3) , Returns))

Hypothesis:
The expression seems to aim at ranking the difference between current returns and returns from 3 periods ago. This could be used to identify assets whose recent performance has significantly deviated from their performance 3 periods ago, providing a basis for potential trading signals or portfolio adjustments.



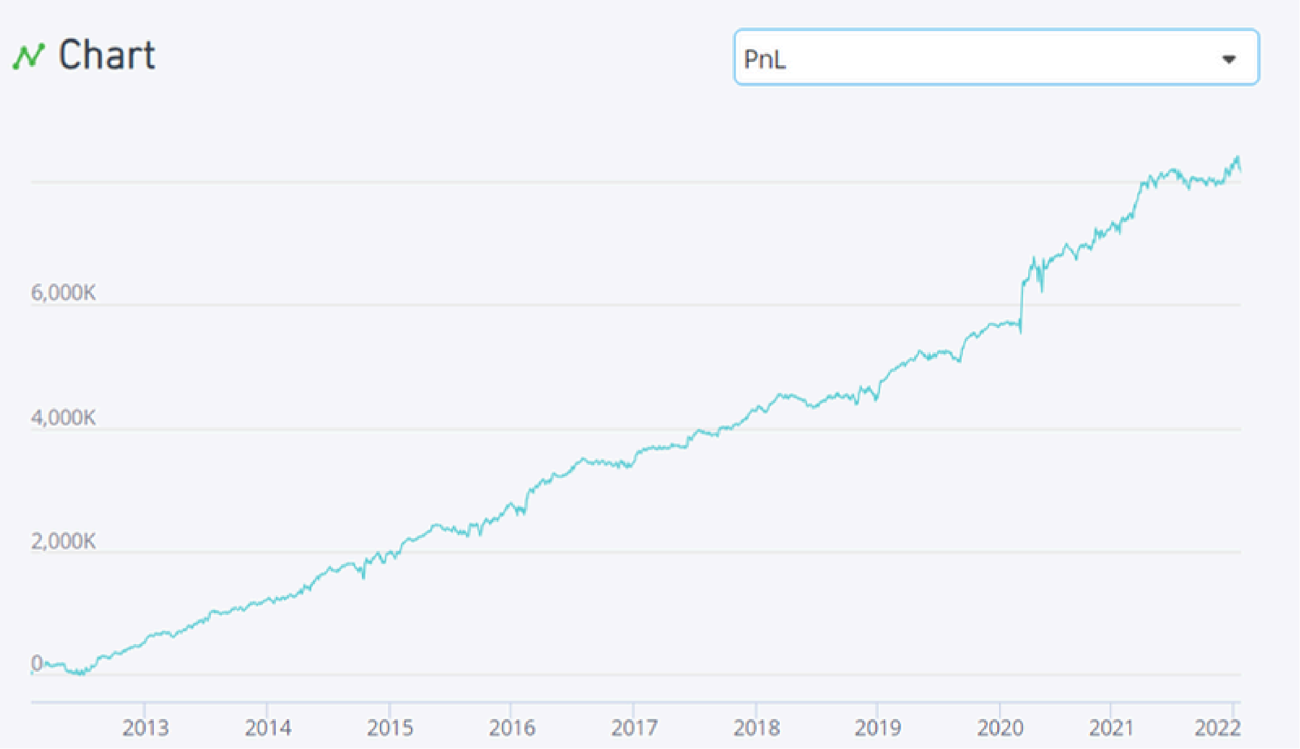
RESULTS

SUBMITTABLE

ts_backfill(ts_zscore(star_val_piv_sector_rank,60),240)

Hypothesis:

The expression aims to normalize the rankings of sectors based on star_val_piv_sector_rank by computing z-scores over a 60-period window and then filling in missing values using backfilling over a 240-period window. This could help in ensuring a smooth and continuous time series for analysis or modeling purposes.



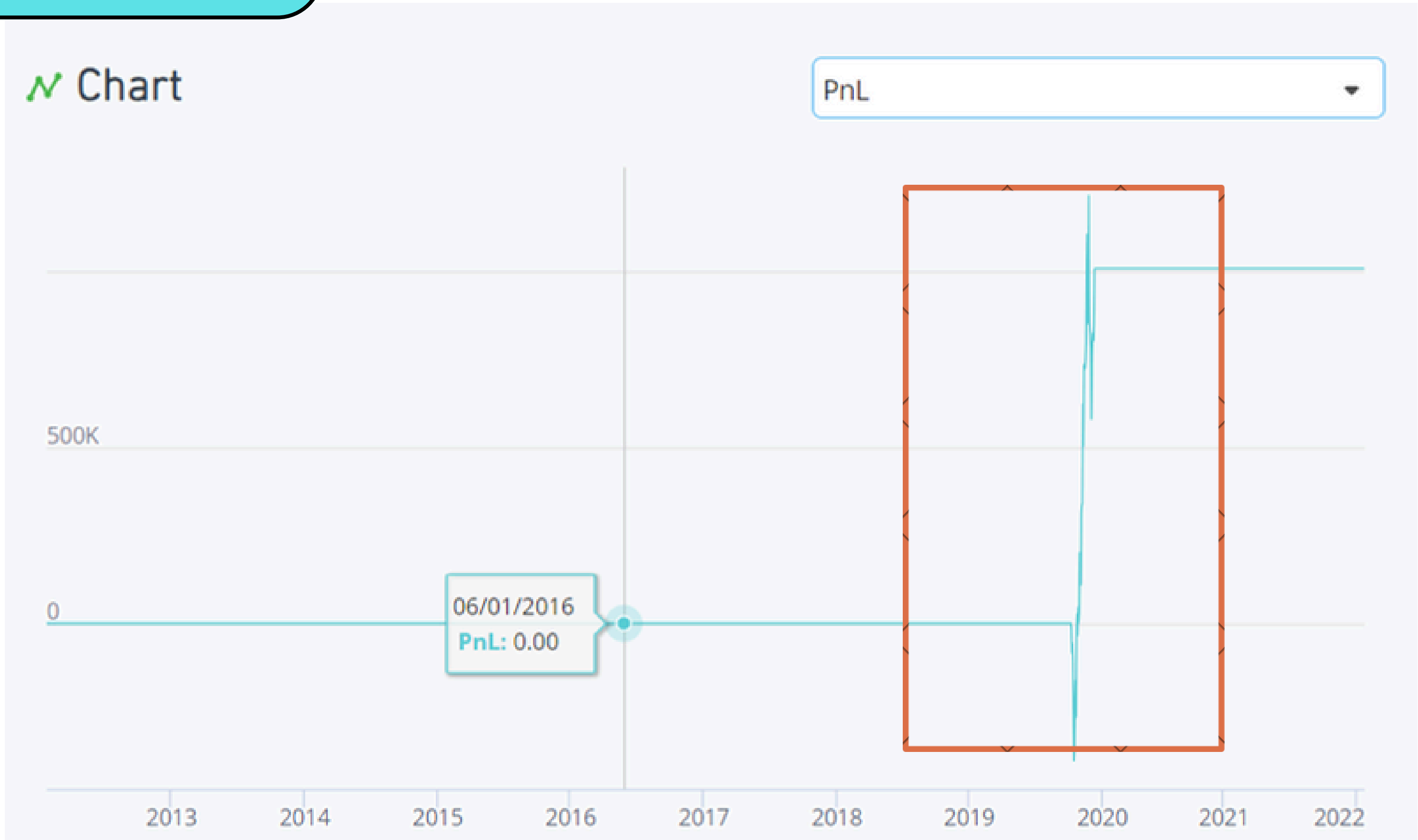
Aggregate Data	Sharpe	Turnover	Fitness	Returns	Drawdown	Margin
	1.91	28.78%	1.03	8.29%	5.80%	5.76‱

RESULTS

HIGHLY OPTIMISED ONY FEW TRADES IN 9 YEARS

ts_zscore(ts_zscore(mdl38_iv_year_div,40),120)

Hypothesis:
The expression aims to further normalize and smooth the mdl38_iv_year_div time series by applying two levels of z-scoring with different window sizes (40 and 120). This could help in identifying and emphasizing significant deviations or trends in the data, potentially for trading or analysis purposes.



Aggregate Data	Sharpe	Turnover	Fitness	Returns	Drawdown	Margin
	2.67	27.18%	3.84	56.09%	6.36%	41.27‱

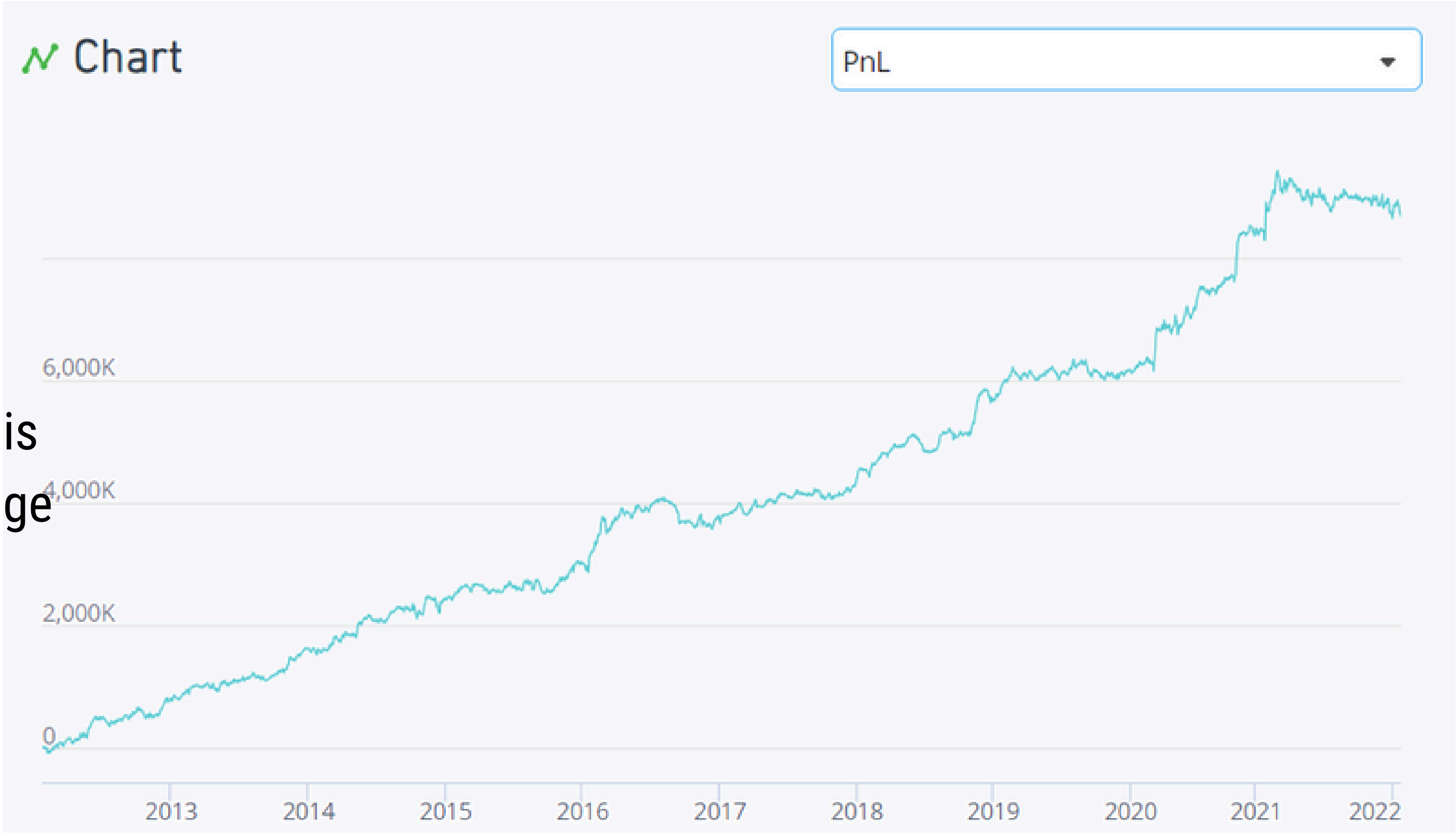
RESULTS

VERY HIGH TURNOVER

ts_delta(ts_delta(divide(vwap,close),240),120)

Hypothesis:

The expression aims to capture the second-order difference in the ratio of VWAP to closing price over two different time periods (240 and 120 periods). This could be used to identify changes in the rate of change of this ratio, potentially indicating shifts in market dynamics or trends.

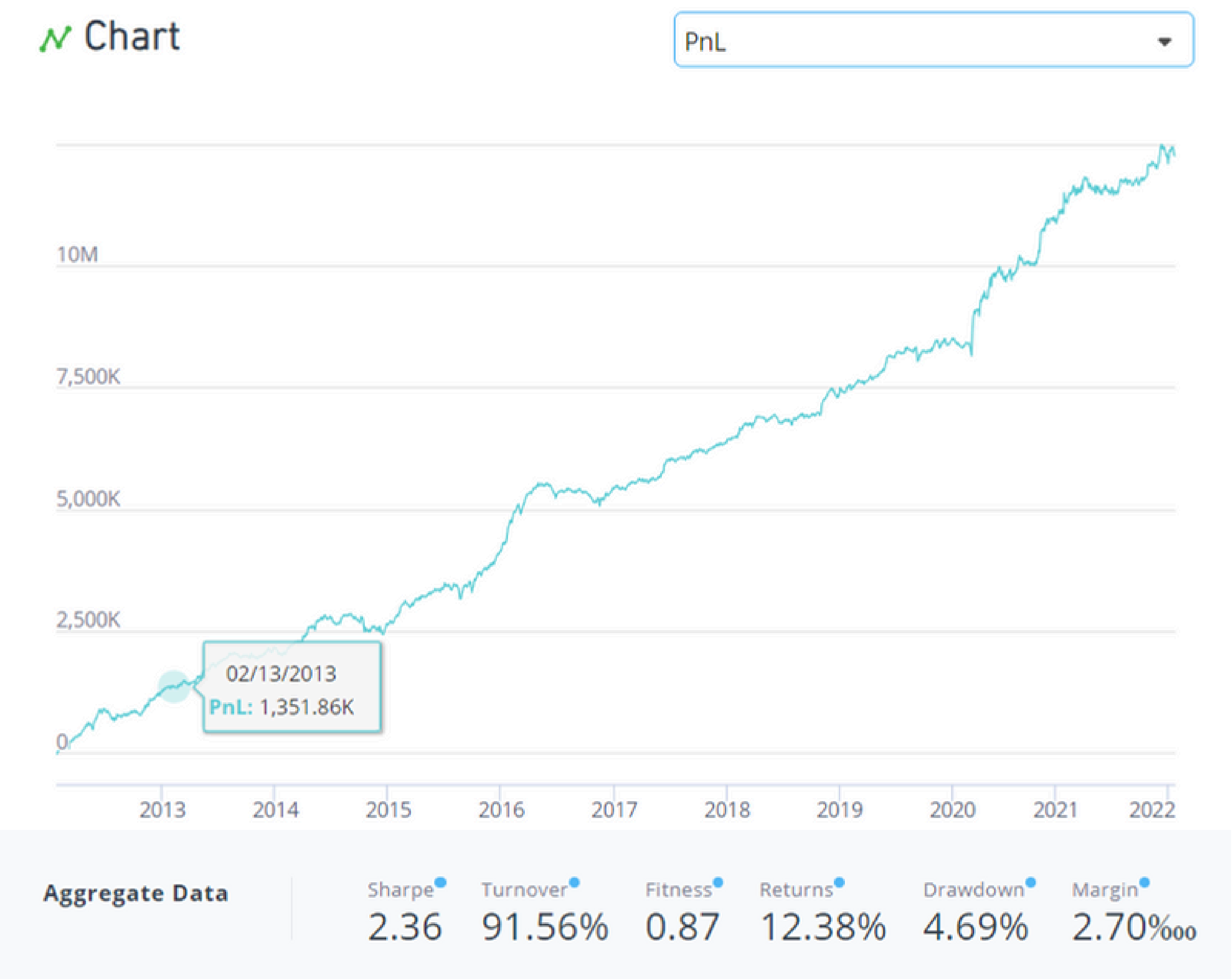


Aggregate Data	Sharpe	Turnover	Fitness	Returns	Drawdown	Margin
	1.63	143.14%	0.40	8.73%	7.91%	1.22‱

RESULTS

ts_zscore(ts_backfill(divide(vwap,close),240),120)

Hypothesis:
This alpha aims to identify deviations from the typical relationship between the VWAP and the closing price of an asset over a certain period. By calculating the ratio of VWAP to closing price and then normalizing it using a z-score, it attempts to identify periods where this ratio significantly differs from its historical average. The backfill and moving average are used to smooth out short-term fluctuations and highlight more significant deviations from the norm.

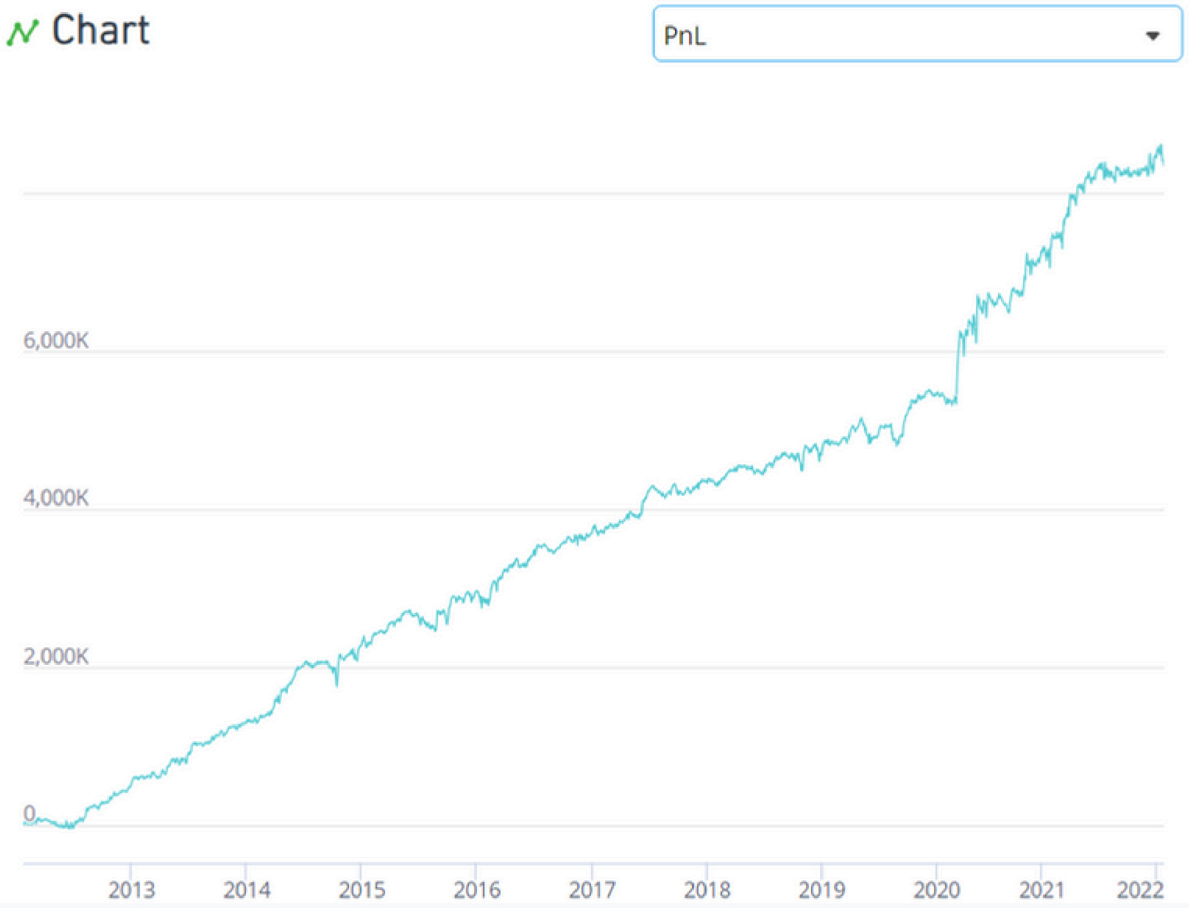


RESULTS

ts_rank(multiply(star_val_piv_region_rank,star_val_piv_region_rank),20)

Hypothesis:

The expression aims to rank the product of the rankings of star_val_piv_region_rank for different regions over a 20-period window. This could be used to identify regions that consistently perform well relative to others, or to detect changes in relative performance over time.



Aggregate Data	Sharpe	Turnover	Fitness	Returns	Drawdown	Margin
	1.75	28.94%	0.95	8.53%	3.63%	5.89‱

THANK YOU