**Detailed Breakdown of the "Alpha / Beta Multi-Time Analysis" Indicator Code**

This Pine Script code is a complex TradingView script that analyzes a stock's performance relative to a market benchmark over multiple timeframes. It calculates key financial metrics, such as Alpha and Beta, which help investors understand the risk and return profile of a stock compared to the market. Let's break down the script step-by-step, explain each part, and highlight why each section was written.

**1. Indicator Declaration**

pinescript

Copy code

//@version=5

//@Yaseen Khalil

indicator(title='Alpha / Beta Multi-Time Analysis', shorttitle='MTA αβ', overlay=false)

* **//@version=5**: Specifies that the script uses Pine Script version 5, the latest version with updated functionalities.
* **indicator(title='Alpha / Beta Multi-Time Analysis', shorttitle='MTA αβ', overlay=false)**: Creates an indicator named "Alpha / Beta Multi-Time Analysis" with a shorter title "MTA αβ." The overlay=false parameter ensures that the indicator appears in a separate pane rather than on the price chart.

**2. User Inputs Section**

This section defines inputs that users can adjust directly from the TradingView interface.

pinescript

Copy code

group\_gen\_inputs = 'Main Settings'

benchmark\_symbol = input.symbol('NSE:NIFTY', 'Market Benchmark', group=group\_gen\_inputs)

risk\_free\_rate = input.symbol('TVC:IN03MY', 'Risk-Free Rate', group=group\_gen\_inputs)

analysis\_period = input.int(252, 'Analysis Period (Days)', group=group\_gen\_inputs)

return\_type = input.string('Returns', 'Return Type', options=['Returns', 'Logarithmic Returns'], group=group\_gen\_inputs)

* **group\_gen\_inputs**: Sets the group name "Main Settings" for organizational purposes in the input panel.
* **benchmark\_symbol**: Allows users to select a market benchmark, with the default being 'NSE

' (Nifty 50 Index). This benchmark will be used to compare the stock's performance.

* **risk\_free\_rate**: Lets users choose a risk-free rate symbol, usually a government bond rate, which is crucial for calculating Alpha.
* **analysis\_period**: Defines the number of days to analyze (default is 252, representing a typical number of trading days in a year).
* **return\_type**: Offers two options: "Returns" and "Logarithmic Returns," which change how returns are calculated. Logarithmic returns are often used for more accurate performance assessments over time.

**3. Data Processing Functions**

This part of the code fetches data and performs calculations to analyze the stock and benchmark returns.

**Fetching Data**

pinescript

Copy code

fetch\_data(symbol, timeframe, price) =>

request.security(symbol, timeframe, price, barmerge.gaps\_off, barmerge.lookahead\_off)

* **fetch\_data**: A helper function that uses request.security to retrieve historical price data for the specified symbol, timeframe, and price type (e.g., close). The parameters barmerge.gaps\_off and barmerge.lookahead\_off are used to handle data gaps and avoid future-looking data errors.

**Calculating Returns**

pinescript

Copy code

calculate\_returns(price\_series) =>

return\_type == 'Returns' ? price\_series / price\_series[1] - 1 : math.log(price\_series / price\_series[1])

* **calculate\_returns**: Computes either simple returns or logarithmic returns based on the user's selection. The formula price\_series / price\_series[1] - 1 calculates simple returns, while math.log(price\_series / price\_series[1]) calculates logarithmic returns, which are preferred in financial modeling for their additive properties.

**4. Alpha and Beta Analysis Function**

This function calculates Alpha and Beta values, which are key metrics in finance.

pinescript

Copy code

analyze\_alpha\_beta(timeframe, period) =>

stock\_close = fetch\_data(syminfo.tickerid, timeframe, close)

index\_close = fetch\_data(benchmark\_symbol, timeframe, close)

risk\_free\_close = fetch\_data(risk\_free\_rate, timeframe, close)

stock\_return = calculate\_returns(stock\_close)

index\_return = calculate\_returns(index\_close)

std\_dev\_stock = ta.stdev(stock\_return, period)

std\_dev\_index = ta.stdev(index\_return, period)

correlation = ta.correlation(stock\_return, index\_return, period)

beta\_val = correlation \* (std\_dev\_stock / std\_dev\_index)

alpha\_val = stock\_return - (risk\_free\_close + beta\_val \* (index\_return - risk\_free\_close))

[beta\_val, alpha\_val]

* **Purpose**: This function calculates Alpha and Beta for a given timeframe and period.
* **Data Fetching**: It retrieves the closing prices for the stock, the market benchmark, and the risk-free rate using the fetch\_data function.
* **Returns Calculation**: It computes the returns of the stock and the benchmark using calculate\_returns.
* **Standard Deviation and Correlation**:
  + **std\_dev\_stock** and **std\_dev\_index**: Calculate the volatility (standard deviation) of stock and index returns.
  + **correlation**: Measures how closely the stock's returns move with the benchmark's returns.
* **Beta Calculation**:
  + **beta\_val = correlation \* (std\_dev\_stock / std\_dev\_index)**: Calculates Beta by adjusting the correlation with the ratio of stock volatility to index volatility.
* **Alpha Calculation**:
  + **alpha\_val = stock\_return - (risk\_free\_close + beta\_val \* (index\_return - risk\_free\_close))**: Measures how much the stock has performed compared to the expected return, considering its Beta and the benchmark's performance.

**5. Calculate Alpha and Beta for Various Timeframes**

The script calculates Alpha and Beta across multiple timeframes, from minutes to years.

pinescript

Copy code

[beta\_5min, alpha\_5min] = analyze\_alpha\_beta('5', analysis\_period)

[beta\_30min, alpha\_30min] = analyze\_alpha\_beta('30', analysis\_period)

[beta\_hour, alpha\_hour] = analyze\_alpha\_beta('60', analysis\_period)

[beta\_day, alpha\_day] = analyze\_alpha\_beta('D', analysis\_period)

[beta\_week, alpha\_week] = analyze\_alpha\_beta('W', analysis\_period)

[beta\_month, alpha\_month] = analyze\_alpha\_beta('M', analysis\_period)

[beta\_3month, alpha\_3month] = analyze\_alpha\_beta('3M', analysis\_period)

[beta\_6month, alpha\_6month] = analyze\_alpha\_beta('6M', analysis\_period)

[beta\_year\_week, alpha\_year\_week] = analyze\_alpha\_beta('W', 52)

[beta\_year\_month, alpha\_year\_month] = analyze\_alpha\_beta('M', 12)

[beta\_2year\_week, alpha\_2year\_week] = analyze\_alpha\_beta('W', 104)

[beta\_2year\_month, alpha\_2year\_month] = analyze\_alpha\_beta('M', 24)

[beta\_4year\_month, alpha\_4year\_month] = analyze\_alpha\_beta('M', 48)

[beta\_4year\_year, alpha\_4year\_year] = analyze\_alpha\_beta('12M', 4)

* **Purpose**: These lines compute Alpha and Beta for different timeframes, ranging from short-term (5 minutes) to long-term (4 years), giving users insights into performance across various time scales.

**6. Performance Evaluation and Table Display**

**Evaluating Performance**

pinescript

Copy code

evaluate\_performance(alpha, beta) =>

alphaEval = ""

betaEval = ""

if (alpha > 0)

alphaEval := "Alpha > 0: Outperforming, excess reward for risk."

else if (alpha == 0)

alphaEval := "Alpha = 0: Adequate return for risk taken."

else

alphaEval := "Alpha < 0: Underperforming, too risky for the return."

if (beta == 1)

betaEval := "Beta = 1: Volatile as the index."

else if (beta > 2)

betaEval := "Beta > 2: Trending Stock, high risk/reward."

else if (beta > 1)

betaEval := "Beta > 1: More volatile, higher risk/reward."

else if (beta > 0)

betaEval := "Beta < 1: Less volatile, safer."

else if (beta == 0)

betaEval := "Beta = 0: No correlation to the index."

else

betaEval := "Beta < 0: Negatively correlated to the index."

alphaEval + "\n" + betaEval

* **Purpose**: This function interprets Alpha and Beta values, providing textual insights about a stock's performance compared to the benchmark. It helps users understand whether the stock is performing well and whether it's risky or safe.

**Displaying Results in a Table**

pinescript

Copy code

if display\_table and barstate.islast

var table data\_tbl = table.new(table\_position, 15, 10)

// Group Headers and Labels

table.cell(data\_tbl, 1, 0, text='Short TF', text\_color=color.gray, text\_size=table\_text\_size, bgcolor=color.new(color.gray, 90))

table.cell(data\_tbl, 6, 0, text='Medium TF', text\_color=color.gray, text\_size=table\_text\_size, bgcolor=color.new(color.gray, 90))

table.cell(data\_tbl, 11, 0, text='Long TF', text\_color=color.gray, text\_size=table\_text\_size, bgcolor=color.new(color.gray, 90))

// Displaying Timeframe Data

table.cell(data\_tbl, 1, 2, text='5 min', text\_color=color.gray, text\_size=table\_text\_size)

table.cell(data\_tbl, 2, 2, text=str.tostring(alpha\_5min, '#.##'), text\_color=color.new(alpha\_5min > 0 ? color.green : color.red, 0), text\_size=table\_text\_size)

table.cell(data\_tbl, 3, 2, text=str.tostring(beta\_5min, '#.##'), text\_color=color.new(beta\_5min > 1 ? color.blue : color.red, 0), text\_size=table\_text\_size)

// Similar code for other timeframes

* **Purpose**: This section builds a table to display the calculated Alpha and Beta values for various timeframes, along with performance evaluations. It provides a visual representation of the data, making it easy for traders to quickly assess stock performance at a glance.