

SRStockAlertBot Report

This is a summarized description of the purpose and functioning of this mini analysis project, how the analysis is done, loopholes/drawbacks in this analysis, and future scope.

CONTENTS:

Purpose

Functioning

Code Logic:

Evaluation of Alerts - Logic:

Drawbacks:

Loopholes:

Future scope

Results (As on 25 Nov, 2021)

Bullish Alerts 🐮 vs 🐻 Bearish Alerts:

Purpose

I wanted to make sure if the stock alerts given on this telegram page are reliable or not, and to what extent, which would allow me to make an informed decision based on my risk profile.

Functioning

Folder structure:

- main.py: Code that is responsible for fetching the data from telegram servers, parsing, cleaning, and organizing the data into excel, and finally evaluating the alerts.
- utils.py: Function that contains the logic to evaluate the stock alerts
- last_id.py: Used to store and fetch the ID of the last message that was retrieved & evaluated since main.py was last executed. This allows the script to only evaluate the recent alerts that were produced on the channel since main.py last ran. Hence allowing

the function in utils.py to avoid evaluating the alerts that have already been evaluated in the past.

- log.txt: Used for maintaining the accuracy metrics, updates whenever main.py is executed.
- backup stock data.xlsx: Self-explanatory. In case the admin changes the format of the alerts/messages, the code isn't robust enough to adapt.
- Result.xlsx: Self-explanatory.

Code Logic:

1. Connect to telegram server and fetch messages from channel:
<https://t.me/SRStockAlertBot>. Only the messages which have not been retrieved in the past will be fetched using last_id.py. If main.py is run for the first time, will fetch all the messages since the inception of this channel.
2. Filter out NCASH and F&O messages.
3. Filter out required parameters from the alerts such as [What, Time, Stock, Trigger Price, SL1, SL2, Target] and create a data frame.
4. Create a backup of Results.xlsx up until this point. Evaluate the recent alerts (data frame) using the function in utils.py and append the result onto Result.xlsx (i.e Results found on previous alerts). Update Result.xlsx.
5. Update last_id.py and log.txt.

Evaluation of Alerts - Logic:

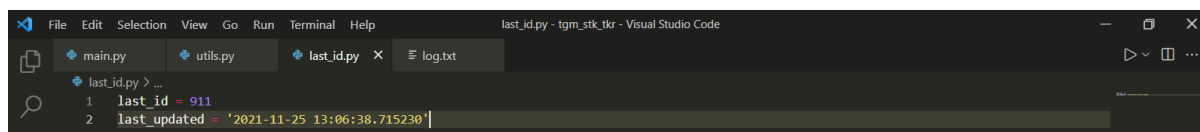
1. The time frame for the data of each stock upon which it will be evaluated is 15 **working** days starting from the Trigger date.
2. 3 Categories, either of which, a stock will fall into after 15 **working** days :
 - a. Target achieved (Only Target 1 is evaluated)
 - b. SL Hit (If SL is hit within these 15 days)
 - c. Sideways (If neither of a or b)
3. For Bullish stocks, the target (T1) is compared with the *high* of each day. And SL1 i.e 'Stoploss on closing basis' is compared with *close* of each day.

4. For Bearish stocks, the target (T1) is compared with the *low* of each day. And SL1 is compared with *close* of each day.
5. The result of the stock i.e [Target Achieved, SL Hit, or Sideways] is stored in a subsequent column.
6. '15Day Close' i.e the *closing price* of the stock as on 15th day is stored in a subsequent column.
7. NoD i.e Number of days it took for the stock to achieve the target (T1) or hit SL1 is calculated for each stock and stored in a subsequent column.

Drawbacks:

1. Doesn't evaluate the degree of target achieved i.e if the stock hits Target 1, it will be dumped into the 'Target Achieved' class. Won't check if it reached T2/T3/T4/T5.
2. Doesn't take into account SL2 i.e 'System Stoploss'. If SL1 (on closing basis) hits, it will immediately be dumped into 'SL Hit' class.
3. last_id.py and Result.xlsx will need to be initialized if main.py is being run for the first time.

Format of last_id.py:

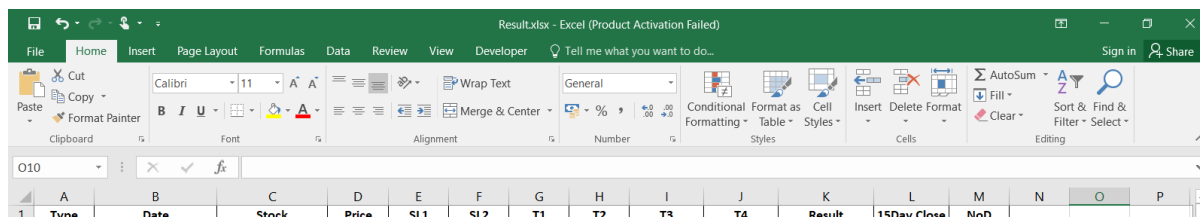


```

1 last_id = 911
2 last_updated = '2021-11-25 13:06:38.715230'

```

Format of Result.xlsx:



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	Type	Date	Stock	Price	SL1	SL2	T1	T2	T3	T4	Result	15Day Close	NoD			

Loopholes:

None that I know of that might be affecting the accuracy.

Future scope

1. Evaluate the degree of target achieved. Would help in better decision-making. (In progress)
2. Detailed analysis of the parameters of alert messages, such as to identify patterns among parameters that have given better results in the past (Maybe using a ML model)

Results (As on 25 Nov, 2021)

(Alerts considered for evaluation : since inception i.e May 27 - Nov 25, 2021)

% of alerts that hit T1 = 54.25%

% of alerts that hit SL1 = 39.88%

% of alerts that went sideways = 5.87%

Average number of days required to hit T1 = 2.59

NOTE: All the alerts are from the free channel only, which contains only a sample of the total population alerts which are available in the paid service. Therefore, a margin of error can be assumed for the true accuracy of the underlying model.

Approximate accuracy of the true model $\sim (54.25 \pm 5)\%$

Bullish Alerts 🐮 vs 🐻 Bearish Alerts:

Total alerts = 494

total bullish alerts = 420

total bearish alerts = 74

bullish alerts achieved = 241

bearish alerts achieved = 27

accuracy of bullish alerts = 57.38%

accuracy of bearish alerts = 36.48%

BULLISH ALERTS HAVE PERFORMED BETTER!