Quantitative Equity Trading Strategies

Krutarth Haveliwala

DATASET

Source - Bloomberg Terminal

Pool of assets - SPDR and constituent ETFs

Frequency - Minute by Minute

Timeframe - 6 months

Technical Indicators

List of Technical Indicators used:

1) RSI
2) Bollinger Bands
3) MACD
4) Stochastic Oscillator
5) Money Flow Index

RSI - Relative Strength Index

- It is a momentum indicator.
- RSI measures the speed and magnitude of a security's recent price changes to evaluate overvalued or undervalued conditions in the price of that security.
- We have chosen a period of 14 for our signal.

$$RSI_{\text{step one}} = 100 - \left[\frac{100}{1 + \frac{\text{Average gain}}{\text{Average loss}}} \right]$$

MACD – Moving Average Convergence Divergence

Example: MACD=12-Period EMA − 26-Period EMA

The result of that calculation is the MACD line.

A nine-day EMA of the MACD line is called the signal line, which is then plotted on top of the MACD line, functions as a trigger for buy or sell signals.

Stochastic Oscillator

Formula:

$$\%K = \left(\frac{C - L14}{H14 - L14}\right) \times 100$$

where:

$C =$ The most recent closing price

$L14 =$ The lowest price traded of the 14 previous trading sessions

$H14 =$ The highest price traded during the same

The stochastic oscillator is range-bound, meaning it is always between 0 and 100. This makes it a useful indicator of overbought and oversold conditions.

Bollinger Band

- Bollinger Bands® are a technical analysis tool developed by John Bollinger for generating oversold or overbought signals.
- There are three lines that compose Bollinger Bands: A simple moving average (middle band) and an upper and lower band.
- The upper and lower bands are typically 2 standard deviations +/- from a 20-day simple moving average (which is the center line), but they can be modified.

$$\mathrm{BOLU} = \mathrm{MA}(\mathrm{TP}, n) + m * \sigma[\mathrm{TP}, n]$$

$$\mathrm{BOLD} = \mathrm{MA}(\mathrm{TP}, n) - m * \sigma[\mathrm{TP}, n]$$

where:

BOLU = Upper Bollinger Band

BOLD = Lower Bollinger Band

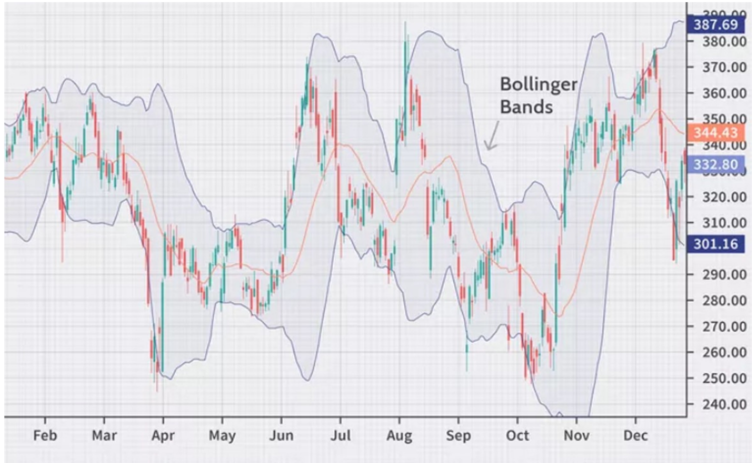
MA = Moving average

TP (typical price) = (High + Low + Close) $\div$ 3

n = Number of days in smoothing period (typically 20)

m = Number of standard deviations (typically 2)

$\sigma[\mathrm{TP}, n]$ = Standard Deviation over last n periods of TP



Money Flow Index

This indicator measures the flow of money into and out of a security over a specified period of time.

It incorporates volume while RSI only considers price.

$$\text{Money Flow Index} = 100 - \frac{100}{1 + \text{Money Flow Ratio}}$$

where:

$$\text{Money Flow Ratio} = \frac{\text{14 Period Positive Money Flow}}{\text{14 Period Negative Money Flow}}$$

$$\text{Raw Money Flow} = \text{Typical Price} * \text{Volume}$$

$$\text{Typical Price} = \frac{\text{High} + \text{Low} + \text{Close}}{3}$$

Testing Indicator Performance

```
total_buy = all_analysis[all_analysis['order_type'] == 'BUY'].shape[0]
print("Number of total times we go long - " + str(total_buy))
total_sell = all_analysis[all_analysis['order_type'] == 'SELL'].shape[0]
print("Number of total times we go short - " + str(total_sell))
total_buy_perc = (int(total_buy) / 33995) * 100
print("Percentage of total times we go long - " + str(total_buy_perc))
total_sell_perc = (int(total_sell) / 33995) * 100
print("Percentage of total times we go short - " + str(total_sell_perc))
```

```
Number of total times we go long - 24266
Number of total times we go short - 9729
Percentage of total times we go long - 71.3810854537432
Percentage of total times we go short - 28.618914546256804
```

I have tested the indicators on our dataset and have calculated the profit generated by each signal given by the indicators

```
In [180]: etf2_combined.iloc[15:25]
```

Out[180]:

	Strategy	close_datetime	open_datetime	order_type	profit	status	Ticker
2203	Bollinger Bands	10/25/22 15:26	10/10/22 9:43	SELL	-5289.219945	closed	VGT
1734	RSI	10/11/22 10:03	10/11/22 10:00	BUY	-112.571429	closed	VGT
2189	MACD	10/25/22 9:30	10/11/22 10:05	BUY	6347.063934	closed	VGT
1735	RSI	10/11/22 12:23	10/11/22 12:20	SELL	70.535714	closed	VGT
1736	RSI	10/11/22 13:16	10/11/22 13:15	BUY	-43.000000	closed	VGT
1737	RSI	10/11/22 13:17	10/11/22 13:16	BUY	47.642857	closed	VGT
1738	RSI	10/11/22 13:18	10/11/22 13:17	BUY	48.357143	closed	VGT
1739	RSI	10/11/22 14:45	10/11/22 14:44	BUY	-71.071429	closed	VGT
1740	RSI	10/11/22 14:46	10/11/22 14:45	BUY	-80.000000	closed	VGT
1741	RSI	10/11/22 14:48	10/11/22 14:46	BUY	-87.857143	closed	VGT

Merging and Processing Data

```
all_analysis
```

	Strategy	close_datetime	open_datetime	order_type	profit	status	Ticker
17209	RSI	10/10/22 10:05	10/10/22 10:03	SELL	-14.803571	closed	VIS
17210	RSI	10/10/22 10:06	10/10/22 10:05	SELL	-16.964286	closed	VIS
10780	RSI	10/10/22 10:30	10/10/22 10:29	SELL	10.289286	closed	VOX
0	RSI	10/10/22 11:01	10/10/22 10:37	BUY	-264.392857	closed	SPY
6662	RSI	10/10/22 10:38	10/10/22 10:37	BUY	-23.821429	closed	VCR
...	...	...	...	...	...	...	...
2870	RSI	9/9/22 9:47	9/9/22 9:46	BUY	69.607143	closed	VGT
10778	RSI	9/9/22 9:47	9/9/22 9:46	SELL	-26.660714	closed	VCR
10779	RSI	9/9/22 9:49	9/9/22 9:47	SELL	52.250000	closed	VCR
6661	Bollinger Bands	9/19/22 9:33	9/9/22 9:47	BUY	-1884.558130	closed	VHT
2871	RSI	9/9/22 9:57	9/9/22 9:48	BUY	86.392857	closed	VGT

33995 rows × 7 columns

Fitting Models

I trained and tested the combined data on the following set of classification models to predict buy/sell signals

```
best_model_portfolio = compare_models(n_select=3, sort = 'Precision', turbo = True)
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
ada	Ada Boost Classifier	0.7049	0.4787	0.0160	0.9061	0.0250	0.0078	0.0453	2.0720
lr	Logistic Regression	0.7125	0.6373	0.0300	0.7099	0.0573	0.0344	0.1013	5.4500
ridge	Ridge Classifier	0.7124	0.0000	0.0298	0.7063	0.0570	0.0340	0.1006	1.2470
lda	Linear Discriminant Analysis	0.7139	0.6400	0.0386	0.7030	0.0730	0.0440	0.1147	1.7050
nb	Naive Bayes	0.7160	0.6298	0.0597	0.6609	0.1095	0.0643	0.1335	1.9130
rf	Random Forest Classifier	0.7080	0.6020	0.0432	0.5933	0.0766	0.0357	0.0873	2.8700
et	Extra Trees Classifier	0.7047	0.6090	0.0290	0.4833	0.0504	0.0177	0.0468	2.3280
knn	K Neighbors Classifier	0.6973	0.6499	0.3275	0.4746	0.3874	0.1954	0.2013	2.0390
svm	SVM - Linear Kernel	0.5710	0.0000	0.2819	0.3491	0.2481	-0.0249	-0.0107	1.2820
qda	Quadratic Discriminant Analysis	0.4175	0.4953	0.7053	0.2888	0.3648	0.0035	-0.0066	1.6940
dt	Decision Tree Classifier	0.7039	0.5011	0.0124	0.0840	0.0183	0.0028	0.0006	1.7420
gbc	Gradient Boosting Classifier	0.7040	0.5303	0.0122	0.0338	0.0179	0.0029	0.0034	2.5820
lightgbm	Light Gradient Boosting Machine	0.7040	0.4784	0.0122	0.0338	0.0179	0.0029	0.0034	1.9210
dummy	Dummy Classifier	0.7074	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000	1.6810

Selecting the Best Models:

```
ada = create_model('ada')
```

Fold	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.6742	0.4931	0.1272	0.3463	0.1861	0.0342	0.0409
1	0.7078	0.4838	0.0018	1.0000	0.0036	0.0025	0.0356
2	0.7083	0.4539	0.0036	1.0000	0.0071	0.0051	0.0504
3	0.7092	0.4867	0.0054	1.0000	0.0107	0.0076	0.0618
4	0.7087	0.4717	0.0036	1.0000	0.0072	0.0051	0.0504
5	0.7081	0.4628	0.0018	1.0000	0.0036	0.0025	0.0357
6	0.7087	0.4956	0.0036	1.0000	0.0072	0.0051	0.0504
7	0.7081	0.4563	0.0018	1.0000	0.0036	0.0025	0.0357
8	0.7076	0.4913	0.0018	1.0000	0.0036	0.0025	0.0356
9	0.7087	0.4916	0.0090	0.7143	0.0177	0.0105	0.0562
Mean	0.7049	0.4787	0.0160	0.9061	0.0250	0.0078	0.0453
Std	0.0103	0.0153	0.0372	0.2051	0.0539	0.0091	0.0093

```
logreg = create_model('lr')
```

Fold	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.7188	0.6532	0.0520	0.8056	0.0976	0.0644	0.1564
1	0.7093	0.6138	0.0215	0.6000	0.0415	0.0217	0.0695
2	0.7088	0.6583	0.0161	0.6000	0.0314	0.0163	0.0601
3	0.7108	0.6400	0.0323	0.6000	0.0613	0.0324	0.0855
4	0.7144	0.6464	0.0395	0.7097	0.0748	0.0454	0.1180
5	0.7087	0.6301	0.0287	0.5333	0.0545	0.0254	0.0670
6	0.7150	0.6186	0.0305	0.8500	0.0589	0.0395	0.1263
7	0.7129	0.6666	0.0251	0.7778	0.0487	0.0310	0.1042
8	0.7134	0.6223	0.0269	0.8333	0.0521	0.0344	0.1160
9	0.7129	0.6238	0.0269	0.7895	0.0520	0.0333	0.1095
Mean	0.7125	0.6373	0.0300	0.7099	0.0573	0.0344	0.1013
Std	0.0030	0.0173	0.0094	0.1106	0.0173	0.0128	0.0289

```
[ ] ridge = create_model('ridge')
```

Fold	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.7172	0.0000	0.0466	0.7879	0.0880	0.0572	0.1444
1	0.7093	0.0000	0.0215	0.6000	0.0415	0.0217	0.0695
2	0.7067	0.0000	0.0143	0.4706	0.0278	0.0107	0.0371
3	0.7108	0.0000	0.0323	0.6000	0.0613	0.0324	0.0855
4	0.7144	0.0000	0.0395	0.7097	0.0748	0.0454	0.1180
5	0.7087	0.0000	0.0287	0.5333	0.0545	0.0254	0.0670
6	0.7160	0.0000	0.0323	0.9000	0.0624	0.0430	0.1376
7	0.7134	0.0000	0.0269	0.7895	0.0521	0.0334	0.1097
8	0.7150	0.0000	0.0305	0.8947	0.0589	0.0404	0.1327
9	0.7123	0.0000	0.0251	0.7778	0.0486	0.0309	0.1041
Mean	0.7124	0.0000	0.0298	0.7063	0.0570	0.0340	0.1006
Std	0.0033	0.0000	0.0085	0.1414	0.0159	0.0125	0.0333

Creating a new model

I have combined the previous 3 best performing models into one

```
[ ] blend1= blend_models(estimator_list=[ada, logreg, ridge], optimize= 'Precision')
```

Fold	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.7177	0.0000	0.0466	0.8125	0.0881	0.0582	0.1493
1	0.7083	0.0000	0.0179	0.5556	0.0347	0.0167	0.0564
2	0.7083	0.0000	0.0143	0.5714	0.0280	0.0138	0.0527
3	0.7108	0.0000	0.0323	0.6000	0.0613	0.0324	0.0855
4	0.7150	0.0000	0.0395	0.7333	0.0750	0.0465	0.1226
5	0.7081	0.0000	0.0269	0.5172	0.0512	0.0229	0.0615
6	0.7150	0.0000	0.0287	0.8889	0.0557	0.0380	0.1281
7	0.7123	0.0000	0.0233	0.7647	0.0453	0.0285	0.0985
8	0.7139	0.0000	0.0269	0.8824	0.0522	0.0355	0.1229
9	0.7123	0.0000	0.0251	0.7778	0.0486	0.0309	0.1041
Mean	0.7122	0.0000	0.0282	0.7104	0.0540	0.0323	0.0982
Std	0.0031	0.0000	0.0090	0.1314	0.0168	0.0127	0.0317

Accuracy of the Final Model

The Final Model yields an accuracy of 72.27% on the training dataset and 73.97% on the testing dataset

	Model	Accuracy
0	Voting Classifier	0.7227

```
from pycaret.classification import *
#Loading the stored model
loading = load_model("model2");
#Making Predictions
prediction = predict_model(loading,data = prediction_data)
```

```
Transformation Pipeline and Model Successfully Loaded
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	Voting Classifier	0.7397	0.5016	0	0	0	0.0046	0.0293

Final Results of the Model

```
results_predicted.tail(15)
```

	Ticker	Strategy	order_type	prediction_label
6759	VCR	RSI	SELL	BUY
6760	VHT	RSI	BUY	BUY
6761	VCR	RSI	SELL	SELL
6762	VHT	RSI	BUY	BUY
6763	VHT	RSI	BUY	BUY
6764	VPU	RSI	BUY	BUY
6765	VFH	RSI	SELL	SELL
6766	VCR	RSI	SELL	BUY
6767	VCR	RSI	SELL	BUY
6768	VGT	RSI	BUY	BUY
6769	VCR	RSI	SELL	BUY
6770	VGT	RSI	BUY	BUY
6771	VHT	Bollinger Bands	BUY	SELL
6772	VCR	RSI	SELL	BUY
6773	VGT	RSI	BUY	BUY

Performance

Profit : 8.85% on Notional

Sharpe : 2.44

Thank you!