FRE-GY 7251

Xinyu Zhang xz2202

Yuezhou You yy2156

Yuqing Wang yw3637

Instructor: Pierre-Yves Guillo

Dec 19th, 2019

High Frequency and Algo Trading Project

Alpha Strategy on JPY & Gold

Contents

1. Introduction

The purpose of this project is to come up with trading strategies and conduct back-testing for Gold and JPY assets, given 2 years of hourly prices and flow data for Gold (2014 -2015) and one of 1+ year of hourly prices for USD/JPY (2016). Our work fully covers the three parts of quantitative strategy workflow: data and factor preparation, model development and strategy backtesting.

As market microstructure theory indicates, the order flow affects price and asset price also reflects the regime of market and the dynamics of demand and supply. For data and factor preparation part, based on the given price and order flow data, we build more technical indicators including : adjusted flow, smoothed moving average, exponential moving average, volatility, correlation and relative strength index. In order to reduce overfitting and multicollinearity, we also apply extra trees classifier to select important factors and conduct correlation analysis.

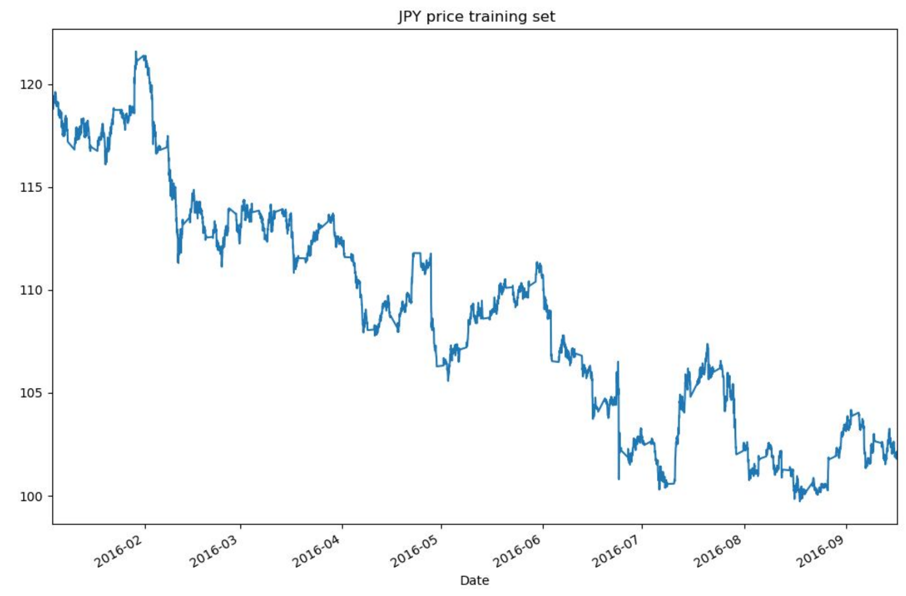
For alpha engine part, we compare the performance of three types of trading strategies: single factor model, trend following model and machine learning algorithms. The single factor model is based on the value range of relative strength index. The trend following model combines multiple factors. It detects the market regimes first and then adopt trend following strategy according to market conditions. In machine learning strategy, we utilize Light Gradient Boosting Machine (tree based learning algorithms) and Long short-term memory model (deep learning model developed for time series data analysis) to predict the price movement in the next period.

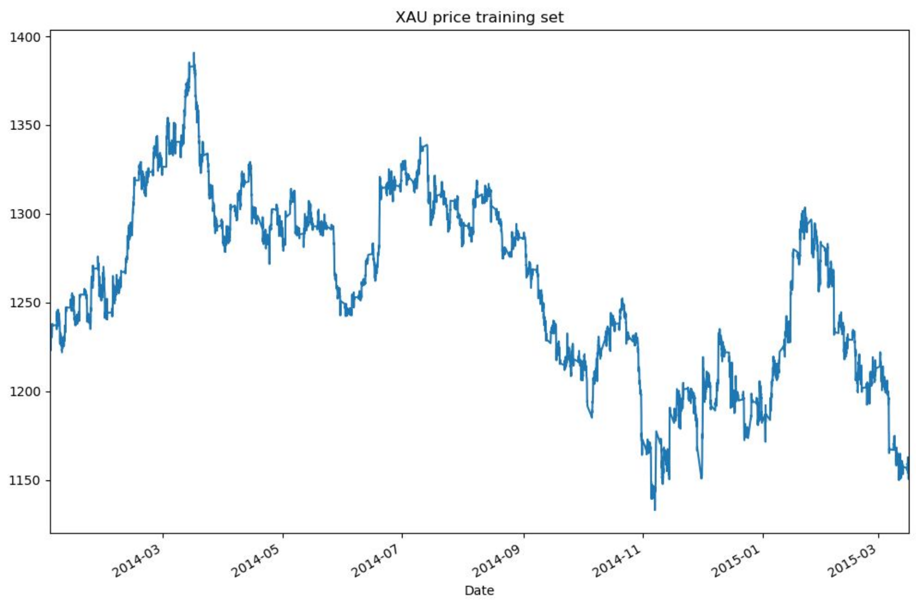
Creating a well defined back testing framework is of vital importance to evaluating performance of signals and trading strategies. In our back-testing part , we divide the data into training, validation and testing set to avoid looking ahead bias, and calculates stats including information ratio, annualized return, VAR and maximum drawdown. In addition, we also considered the influence of bid ask spread and execution time on our strategies to test the soundness and sturdiness of our alphas.

2. Data and Factor reparation

2.1. Data Description

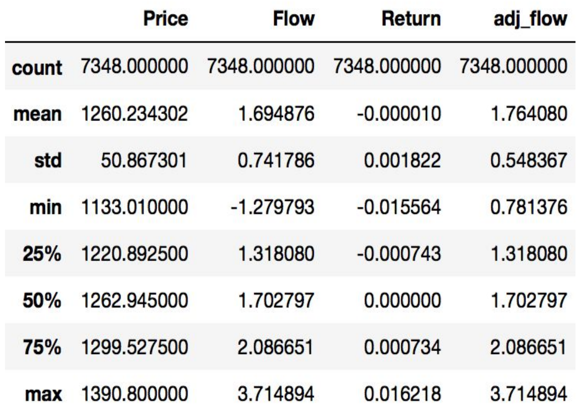
The data given for this project is nearly one-year hourly prices and flows for USD/JPY and two year hourly prices and flows for USD/JPY. Because our aim is to find out a profitable trading strategy, it is helpful to know our dataset and know how the prices for USD/JPY and XAU/USD move from the training set (for single factor and trending strategy, the train test is split by 3:2, for machine learning strategy, the splitting method is mentioned in the next part). Therefore, the visual descriptions of prices in training set are given as the following:





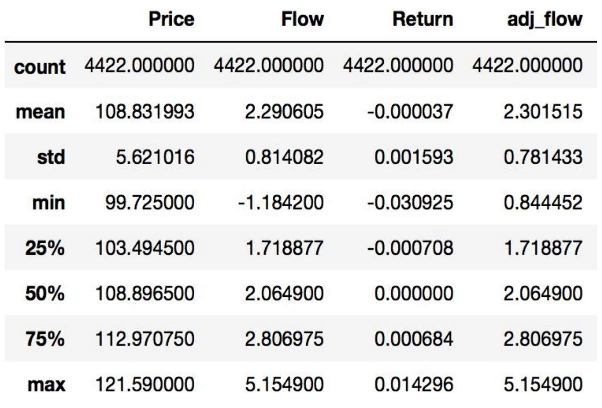
In addition, the basic statistics for the prices and flow in USD/JPY and XAU/USD are given below :

XAU/USD



	Price	Flow	Return	adj_flow
count	7348.000000	7348.000000	7348.000000	7348.000000
mean	1260.234302	1.694876	-0.000010	1.764080
std	50.867301	0.741786	0.001822	0.548367
min	1133.010000	-1.279793	-0.015564	0.781376
25%	1220.892500	1.318080	-0.000743	1.318080
50%	1262.945000	1.702797	0.000000	1.702797
75%	1299.527500	2.086651	0.000734	2.086651
max	1390.800000	3.714894	0.016218	3.714894

USD/JPY



	Price	Flow	Return	adj_flow
count	4422.000000	4422.000000	4422.000000	4422.000000
mean	108.831993	2.290605	-0.000037	2.301515
std	5.621016	0.814082	0.001593	0.781433
min	99.725000	-1.184200	-0.030925	0.844452
25%	103.494500	1.718877	-0.000708	1.718877
50%	108.896500	2.064900	0.000000	2.064900
75%	112.970750	2.806975	0.000684	2.806975
max	121.590000	5.154900	0.014296	5.154900

For the prices of XAU_USD and USD_JPY, the range are (1133,1390) and (99.7,121.59). The range, compared to many stocks, is not highly volatile. However, the hourly standard deviations for each are 75.61 and 5.62. Within moderate price range, the daily and hourly price changes a lot.

Secondly, for both USD/JPY and XAU/USD, there are significant trend patterns. For USD/JPY, the price first went down in the first half-year and then reverse in the following half. For XAU/USD, the wave amplitude is not as large, but the trend is also significant, the long term moving average went down for most times.

2.2. Factor Construction

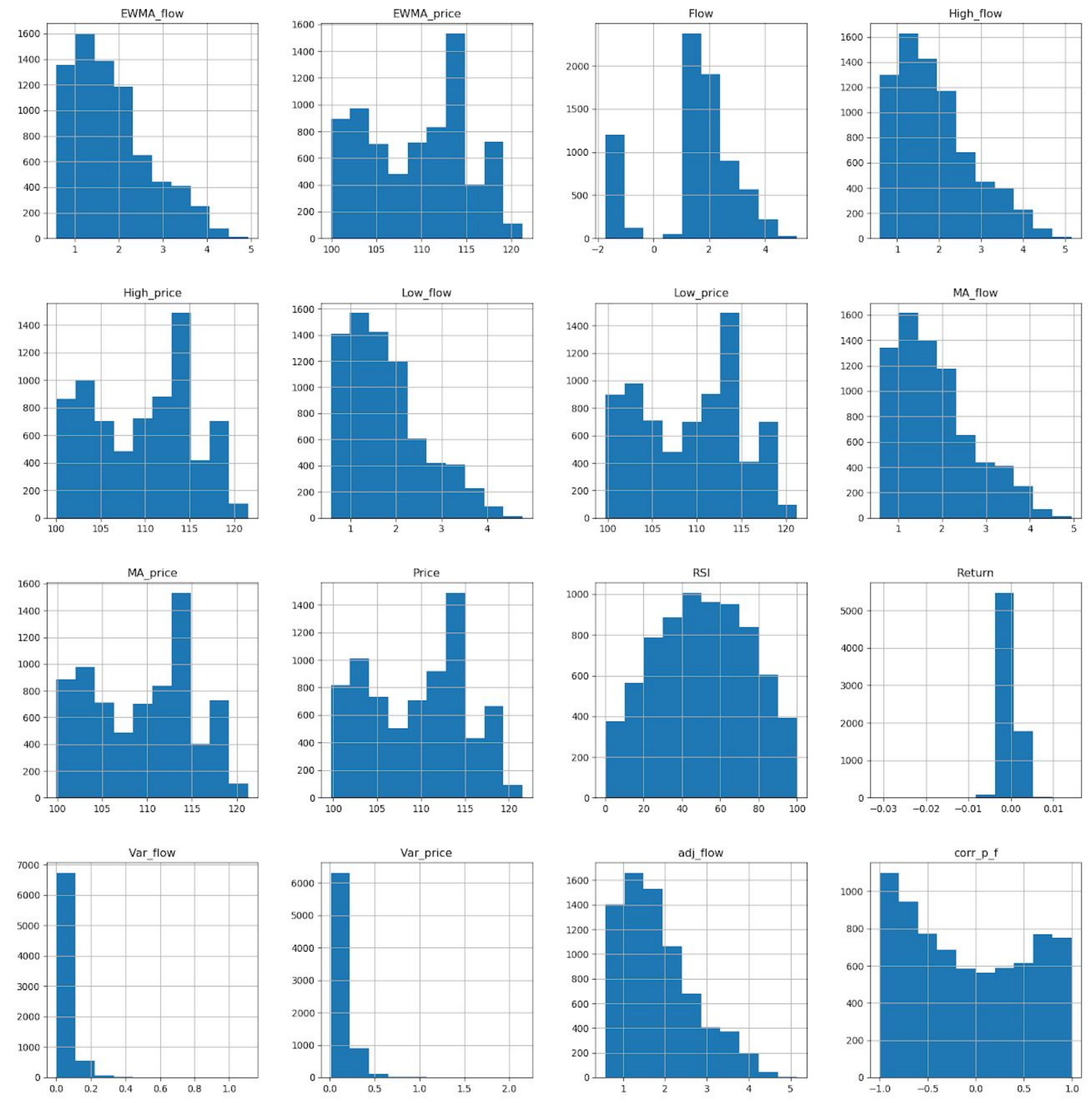
2.2.1. Factor introduction

In addition to price and flow, we also build the following factors:

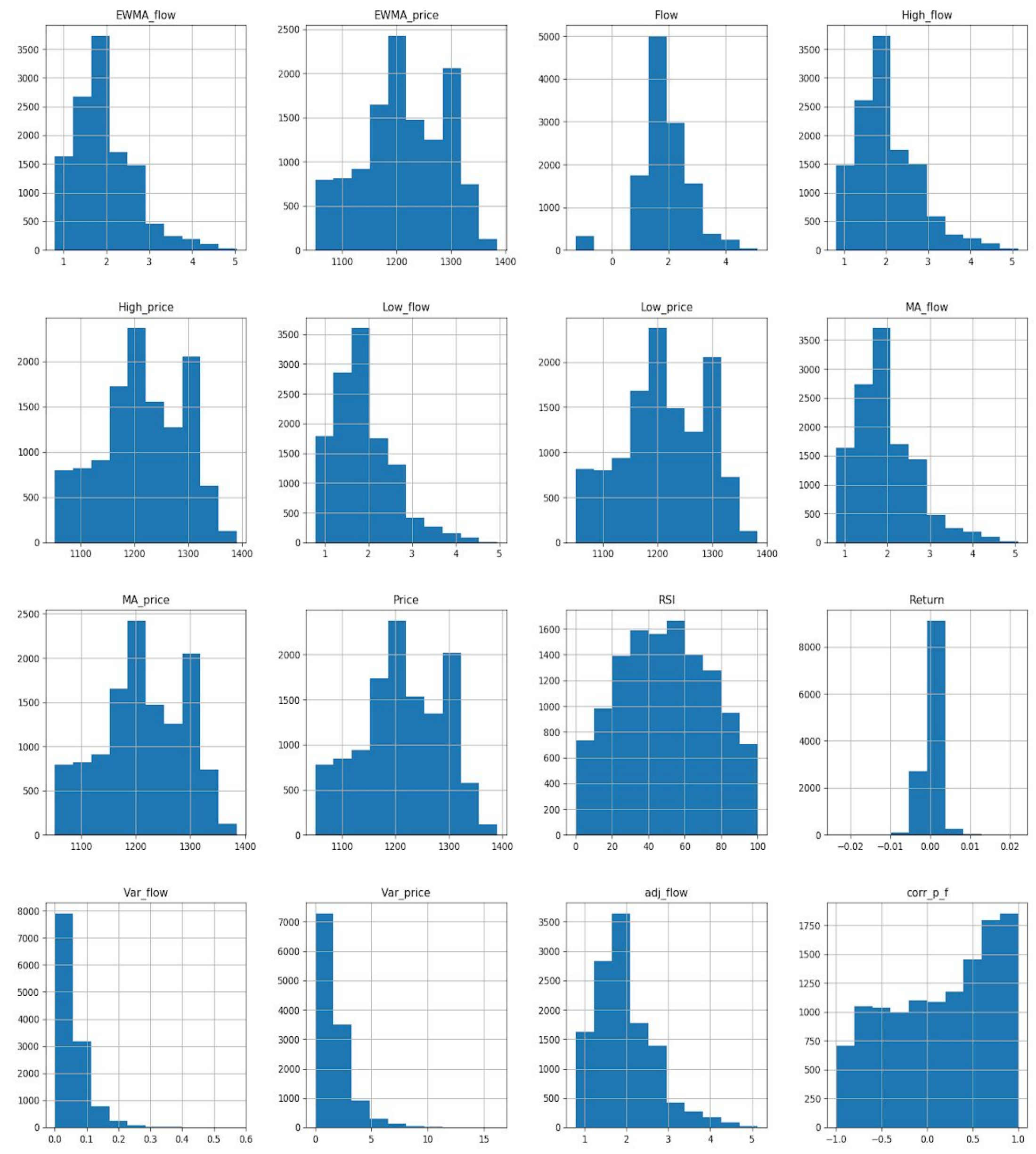
Factor	**Description**
return	daily percent price change
adj_flow	the flow adjusted to positive value by taking the negative reciprocal
MA_price MA_flow	the moving average for price and flow
Var_price Var_flow	the moving variance for price and flow
EWMA	Exponentially Weighted Moving Average
Corr_p_f	the correlation coefficient between price and flow
RSI	relative strength index

2.2.2. Factor visualization :

A. USD/JPY



B. XAU/USD



It is obvious that Vars for both price and flow, return, MAs, EWMAs, high and low for flow have a skewed distribution. Correlation coefficient and RSI have a seemly bell shape distribution. Basing on these distribution characteristics, we can have an intuition of alpha strategy construction.

2.3. Factor Selection

2.3.1. Rank factor importance (Extra Trees Classifier)

A. Extra trees classifier introduction:

Comparing with random forest, which pick best from random subset of features, extra trees classifier picks randomly form random subset of feature. That's why this model can show the importance of different features based on the information gain.

B. Method:

Using Extra Trees to estimate the importance of features on training set.(treat the moving direction of next period as target data). The model gives an importance score for each attribute, the larger score the more important the attribute. Delete several unimportant features based on the important scores to avoid overfitting.

C. Benefits:

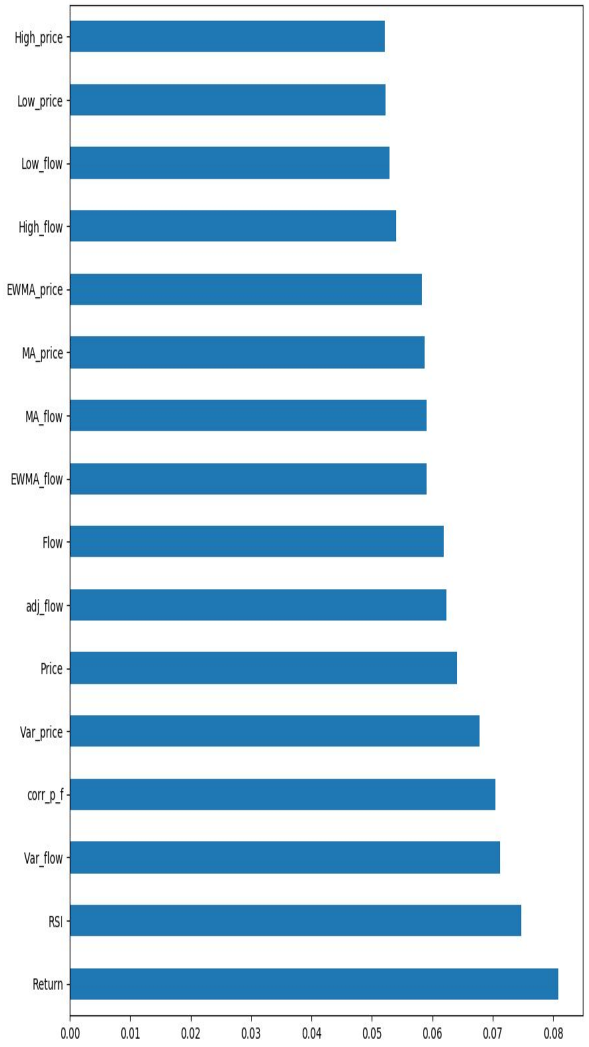
Reduces Overfitting: Less redundant data means less opportunity to make decisions based on noise.

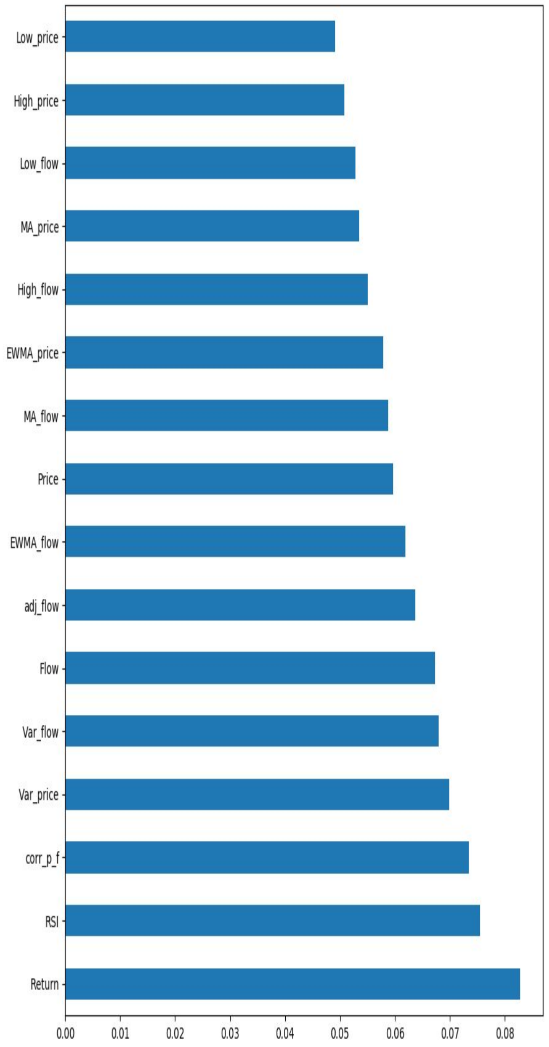
Improves Accuracy: Less misleading data means modeling accuracy improves.

Reduces Training Time: Less data means that algorithms train faste

D. Result:

The two figures below show the rank of importance for all factors in USD/JPY and XAU/USD:

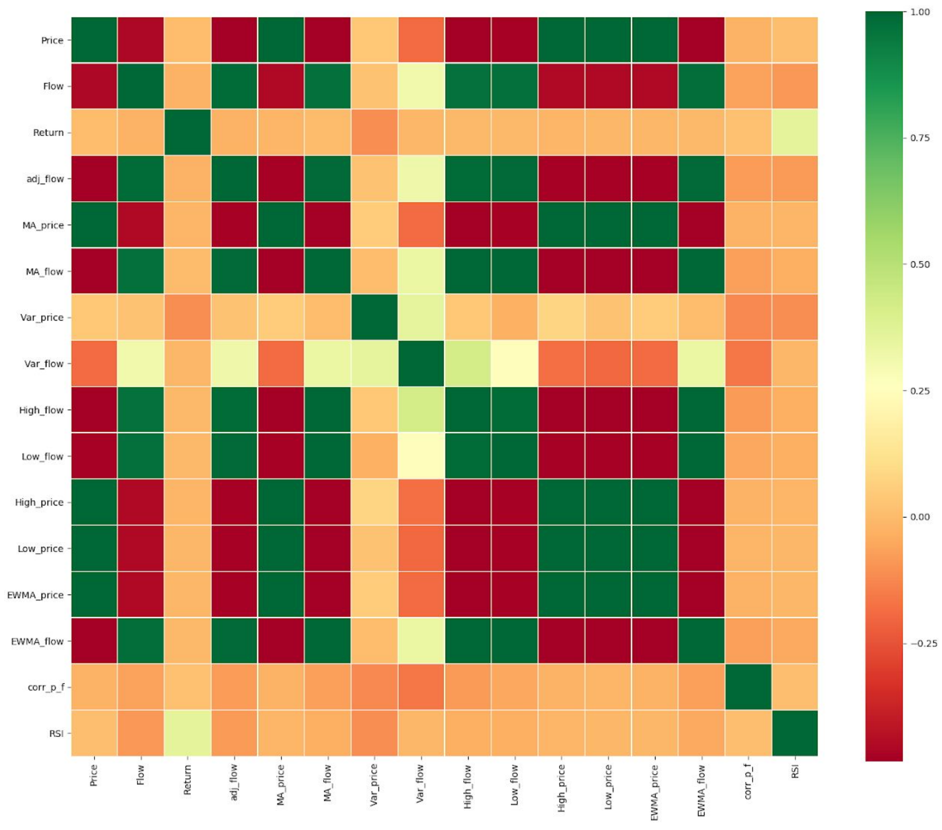


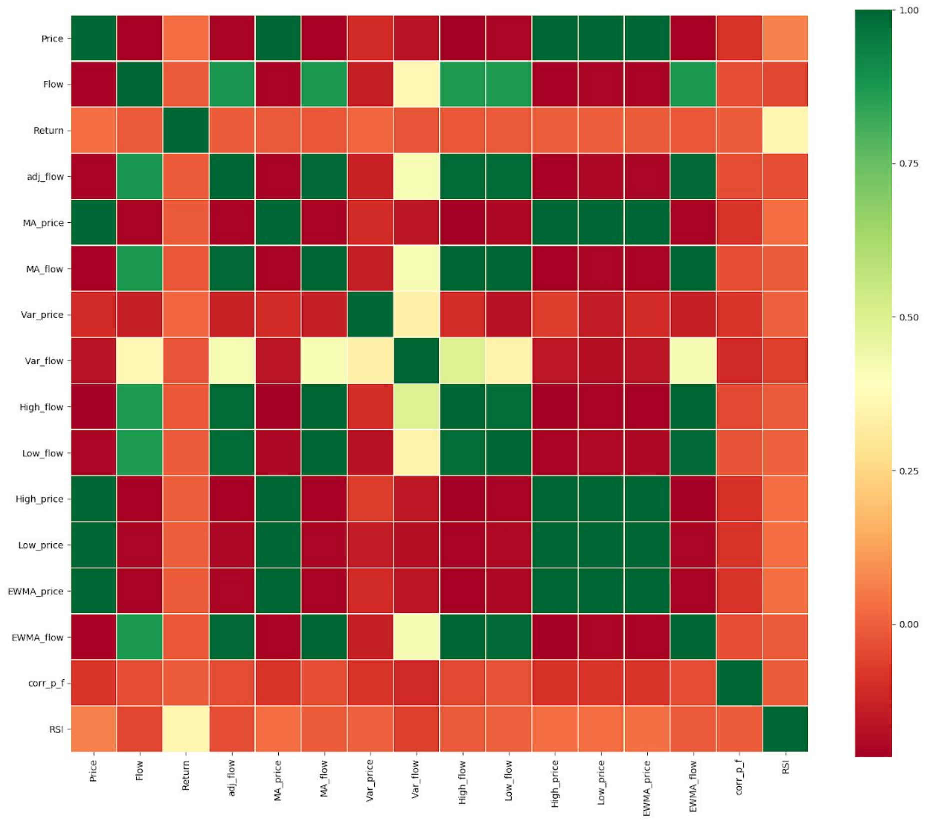


As we can infer from the chart above: RSI and return contributes the highest factor importance. Corr_p_f, price, Var for price and flow have relatively high importance. The rest shares lower differences and lower importances. Therefore, we decide to explore the models that use only RSI and factors including RSI, price and variance.

2.3.2. Correlation test

To avoid multicollinearity, we use the coefficient heat map to view the relationship between factors of USD/JPY and XAU/USD, and drop some highly correlated factors:





In these two graphs, the green shades represent there is a high positive correlation. Unsurprisingly, the collinearity exists among high, low, EWMA and MA for both price and flow. However, as for RSI, corr_p_f and variance, they are relatively independent from prices and flows. Thus, it is statistical reasonable to use one or more factors among RSI, corr_p_f, variance, with price and flow in one model.

3. Backtesting Framework

3.1. Data Split

3.1.1. Japanese Yen

Data of Japanese Yen starts from 17:00 Jan 1st, 2016 and ends at 15:00 Mar 3rd, 2017. There are 7390 data in total. For validation and testing, we split them into five periods as follows:

Period Number	**Start Time (including)**	**End Time (including)**
1	1/1/2016 17:00	3/9/2016 15:00
2	3/9/2017 16:00	6/23/2016 1:00
3	6/23/2016 2:00	9/16/2016 15:00
4	9/16/2016 16:00	12/13/2016 23:00
5	12/14/2016 00:00	3/9/2017 15:00

3.1.2. XAU

Data of XAU starts from 1:00 Jan 2nd, 2014 and ends at 13:00 Dec 31st, 2015. There are 12263 data in total. For validation and testing, we split them into five periods as follows:

Period Number	**Start Time (including)**	**End Time (including)**
1	1/2/2014 1:00	5/27/2014 3:00
2	5/27/2014 4:00	10/17/2014 9:00
3	10/17/2014 10:00	3/16/2015 8:00
4	3/16/2015 9:00	8/6/2015 23:00

5	8/7/2015 00:00	12/31/2015 13:00

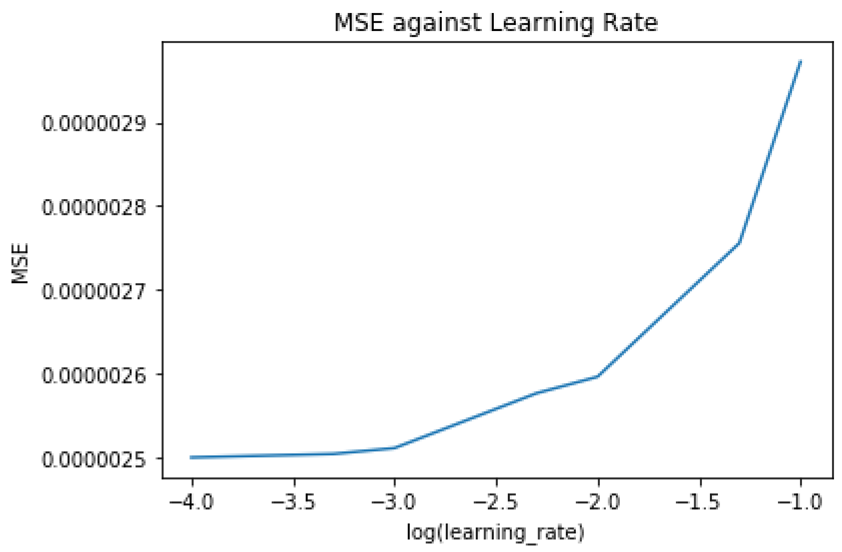
3.2. Backtesting Method

To do validation on time series model, future data could not be used to train models that are validated on previous data. Thus, for backtesting, models are trained with data from the first period and validated by data from the second period. Then we train models on the data from the second period and validate them on the third period. And we train models on the data from the third period and validate them on the fourth period. Finally we calculate the mean of error from three validation sets and select the best parameters and do backtesting on the fifth period to see its performance.

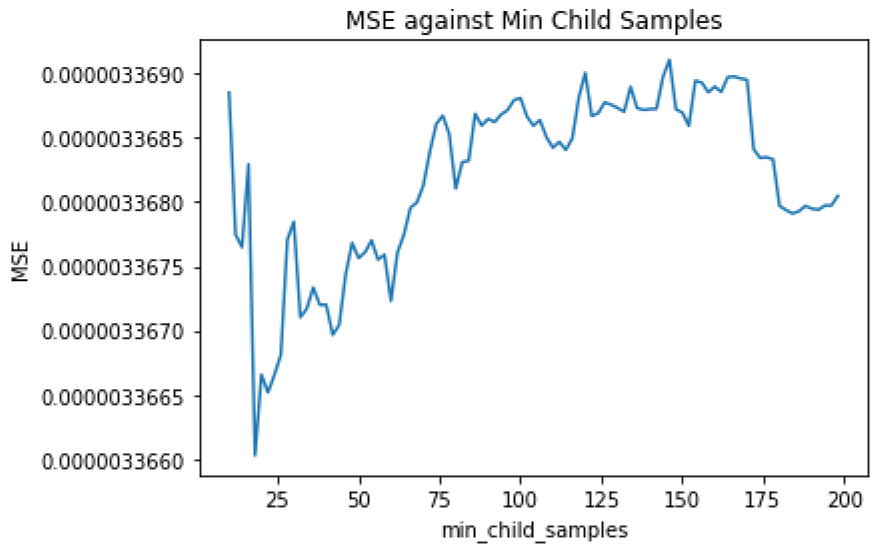
3.3. Backtesting Applied in Parameter Adjustment -- LGBM on XAU as an Example

Among all the parameters in LGBM model, learning rate and min child samples, which is the same as min sample split are two of the most important ones. Learning rate is boosting learning rate, determining the length of step of gradient descent. Min child samples is the minimum sample to divide, a parameter to avoid overfitting. Many other parameters, like max depth, play the same role as min child samples. So we do not need to adjust every one.

The following graph shows how MSE changes against learning rate:



We could find that the larger learning rate is, the larger MSE on validation set is. But the smaller the learning rate is, the less the MSE will decrease and the more time we need to spend on training the model.



This graph shows how min child samples affect MSE on validation set: when it is too small, the MSE will be large as a result of overfitting and when min child samples are too big, LGBM becomes weak learners.

4. Single Factor Model (Relative Strength Index)

4.1. Strategy logic

Based on the analysis of factor importance, it is suggested that RSI contributes a significant high contribution to the factor importance. Therefore, we consider using only RSI as an independent variable and solely decide the signal.

Since the definition of RSI is a momentum indicator that measures the magnitude of recent price changes to evaluate overbought or oversold conditions in the price of a stock or other asset. When RSI is larger than 70, it is usually considered as overbought, when RSI is lower than 30, it is usually considered as oversold.

4.2. Empirical analysis

However, our data analyze implies that, for most of the time, there exists a pattern of trending. Therefore, it is smarter to follow the trend of market instead bet on the other side. For this strategy, the only problem is to pick profitable levels of RSI deciding when to take a long/short position.

Therefore, this is model is trained based on the first 4/5 months performance and the test sample are the last 1/5 of the time; test sample is valued by IR and accumulated return.

The algorithm of this strategy is simple: when the RSI is above certain levels, which means the price is strongly going up, take a short position, and vice versa.

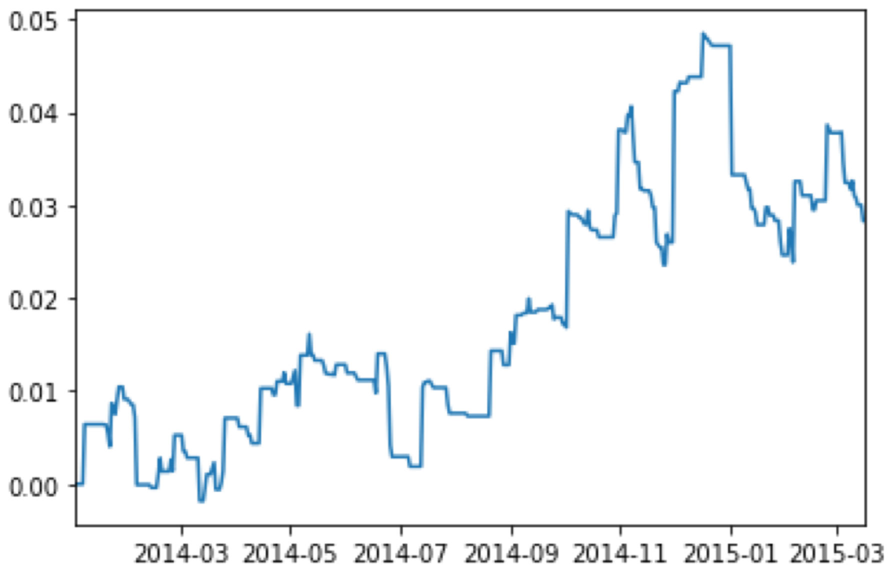
RSI (low/high)	**XAU/USD**		**USD/JPY**	
	IR	annu_r	IR	annu_r
30 / 70	-0.146	-0.019	0.908	0.081
20 / 80	-0.043	-0.07	0.605	0.039
15 / 85	0.037	0.045	0.524	0.029
10 / 90	0.663	0.043	0.064	0.005
5 / 95	0.318	0.015	0.064	0.006
3/ 97	0.70	0.293	-0.299	-0.08

Therefore, the best short/long level for XAU/USD is 3/97: taking short position when RSI is below 3 and taking long position when RSI is above 97. For USD/JPY, it is 30/70: taking short position when RSI is below 30 and taking long position when RSI is above 70.

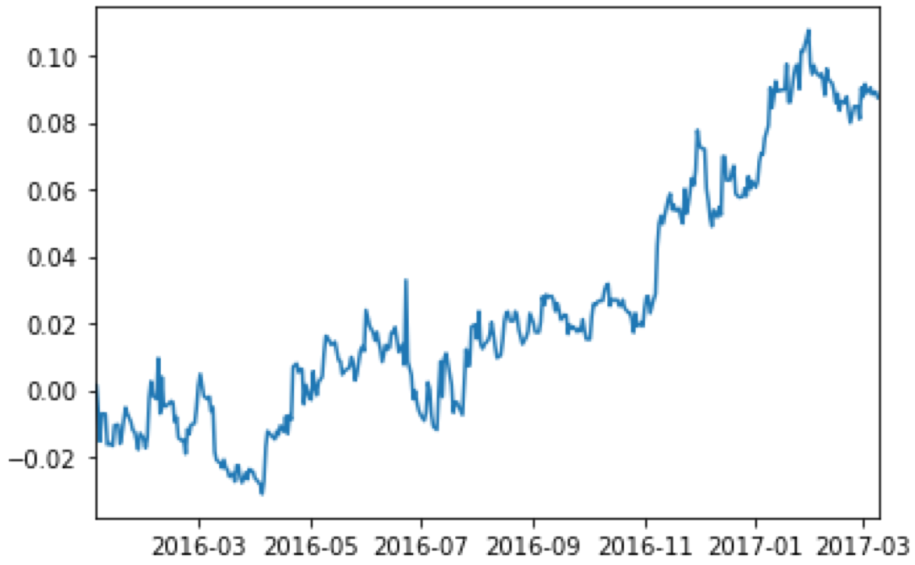
4.3. Testing result

The plot below depicts the PnL of RSI strategy of XAU_USD and USD/JPY.

XAU/USD --- Short/Long when RSI is below/above 3/97 (b/o = 2bips TTE=0):



0.05
0.04
0.03
0.02
0.01
0.00
2014-03
2014-05
2014-07
2014-09
2014-11
2015-01
2015-03



0.10
0.08
0.06
0.04
0.02
0.00
−0.02
2016-03
2016-05
2016-07
2016-09
2016-11
2017-01
2017-03

5. Trend Following Models

The general idea of the trend model is to take a long position when the market is bully and take a short position when the market is tank. Thus, there are two separate steps that contributes to this method.

5.1. Trend Model: Market Regime Recognition

The first step is to recognize the market regime. In the beginning, we tried to investigate whether the classic mean reversion strategy is profitable or not. However, the mean reversion strategy shows a bad performance (IR) on JPY/USD instrument. We think the reason might be that, during the given year, both JPY and XAU has relatively high long-term volatility and significant trend. Thus, we decide to use a trend strategy.
There are 3 parts in the algorithm in deciding the market regime. First, the flow in within certain range: 6-hour MA plus or minus 0.7* variance of flow. Secondly, the price should be within a several hour's price range. Thirdly, the volatility should also be within a certain MA range. If 2 of the 3 factors meet the requirement, we think the market is relatively peaceful and it is time to take position.

5.2. Trend Model: Signal Generator

When the market, as described in 5.1, meets with the requirement. The rest of the strategy is simple. When the current price is above the longer-term moving average, follow the trend and take a long position, and vice versa.

5.3. Result:

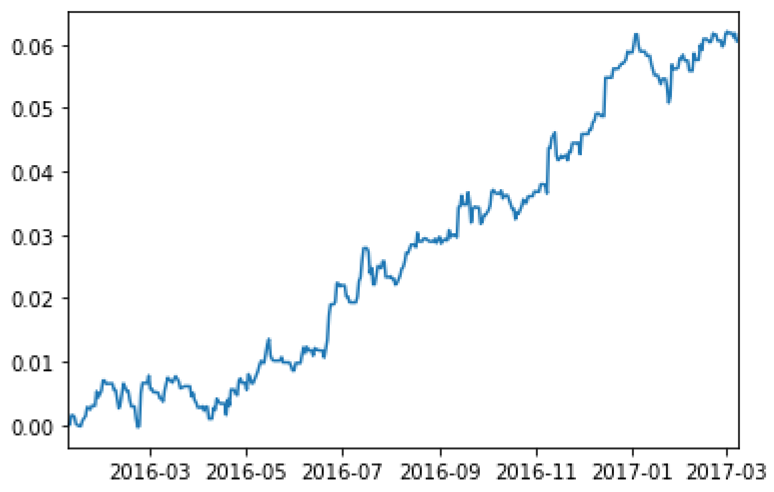
The sensitivities are given below:

Time Window (Hour)	JPY_IR	XAU_IR
4	0.006416	0.156706
6	0.104208	0.172423
8	0.060141	0.132608
10	0.098482	0.030286
12	0.003353	0.028135
24	-0.089494	0.124373

The best model with highest IR is:

Time window = 6 hours (calculating 6 hours moving average of price in deciding the signal)

JPY parameters = 0.7,0.3


0.06
0.05
0.04
0.03
0.02
0.01
0.00
2016-03
2016-05
2016-07
2016-09
2016-11
2017-01
2017-03

6. Machine Learning Models

6.1. Light GBM

6.1.1. Brief Introduction of Light GBM

LightGBM, Light Gradient Boosting Machine, is a gradient boosting framework that uses tree based learning algorithms. A series of trees are grown one by one by refitting the residual of the previous model.

Goal is to optimize a likelihood:

$$\widehat{\beta}, \widehat{\gamma} = argmin_{\{\beta_m, \gamma_m\}_{m=1}^{M}} \sum_{i=1}^{N} L(y_i, \sum_{m=1}^{M} \beta b(x_i, \gamma_m))$$

Here, $\widehat{\beta} \in R^M$ is the estimated weight of each tree model. $\widehat{\gamma}$ is a set of parameters of each tree model. b is a series of tree models we aim to train. L is the loss function measuring the difference between the actual output y_i and estimated output given input x_i. The overall goal is to find parameters of each tree and the weights between trees to minimize the loss over the input data set.

The paragraph above introduced the framework of addictive tree models and this paragraph will talk about how gradient boosting method works to find optimal parameters. In general optimization is done jointly over all parameters $\{\beta_m, \gamma_m\}_{m=1}^{M}$. But for gradient boosting method, instead think of $f(x)$ as a sequence of approximations:

$$f(x) = \sum_{m=1}^{M} \beta b(x_i, \gamma_m) = f_1(x) + f_2(x) + ... + f_M(x)$$

such that$f_m(x) = f_{m-1}(x) + \beta b(x, \gamma_m)$ where

$$\beta_m , \gamma_m = argmin_{\beta, \gamma} \sum_{i=1}^{N} L(y_i, f_{m-1}(x) + \beta b(x_i, \gamma))$$

For a general loss, we want to optimize (and Taylor expanding):

$$\mathcal{J}(\beta, \gamma) = \sum_{i=1}^{N} L(y_i, f_{m-1}(x) + \beta b(x_i, \gamma)) = \sum_{i=1}^{N} L(y_i, f_{m-1}(x)) + \beta b(x_i, \gamma)\partial_f L(y_i, f_{m-1}(x_i))$$

So, our strategy starts with $f_0(x) = argmin_\gamma \sum_{m=1}^{M} L(y_i, \gamma)$. Then we fit the base model $b(x_i, \gamma\ )$ to the residuals, i.e. find the base model $b(x_i, \gamma\ )$ that best approximates $r_{im} =- \partial_f L(y_i, f_{m-1}(x_i))$ to produce the biggest decrease in the loss function until it satisfies stopping condition.

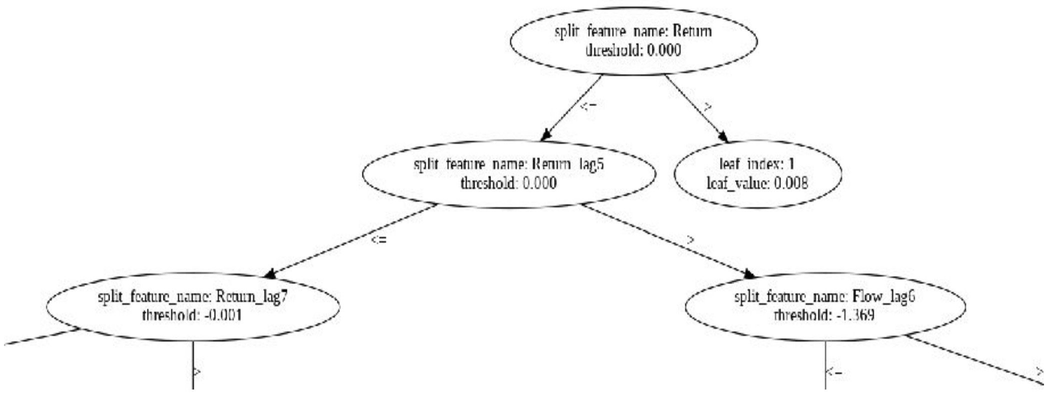
6.1.2. Light GBM Parameters and Visualization

We used LGBMClassifier Class as our models and we are delighted to share parameters to reproduce our result:

Parameters	Values
Random_state	42
Boosting_type	'gbdt'
Learning_rate	0.001
Min_child_samples	20
Num_leaves	31

And here is the first tree of JPY model:

This is the beginning part of the tree: The whole tree is too large to show. It shows that the return of the last period is the first justment: if it is larger than zero, we predict the next return will still be larger than zero; if it is smaller than zero, we need to look at the return 5 hours before.

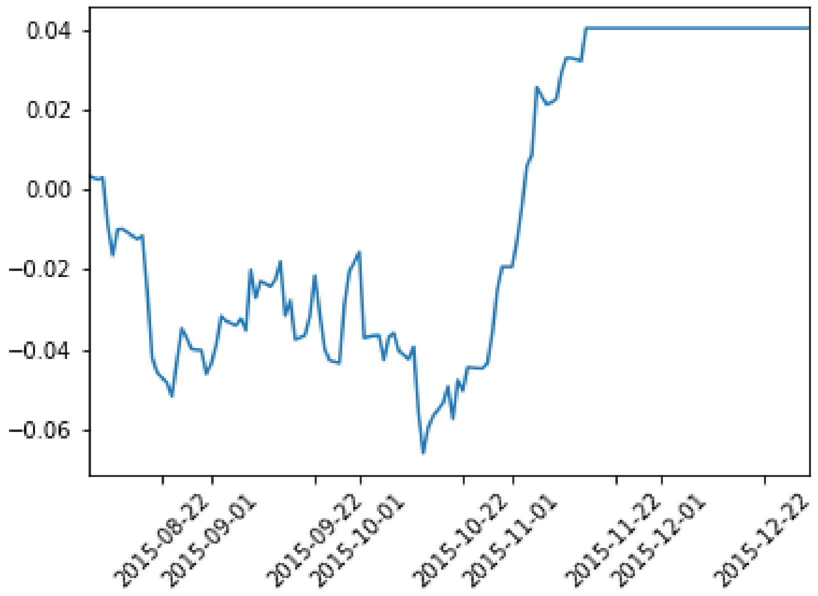

split_feature_name: Return
threshold: 0.000
<=
>
split_feature_name: Return_lag5
threshold: 0.000
leaf_index: 1
leaf_value: 0.008
<=
>
split_feature_name: Return_lag7
threshold: -0.001
split_feature_name: Flow_lag6
threshold: -1.369
>
<=
>

6.1.3. Light GBM Backtesting Result

The results of backtesting of Japanese Yen on the test set with no B/O spread and immediate execution are shown below:

Statistics	dailly_r_mean	dailly_r_var	anu_sharpe	max_drawdown
Values	0.000262	0.0029	1.435403	0.008733

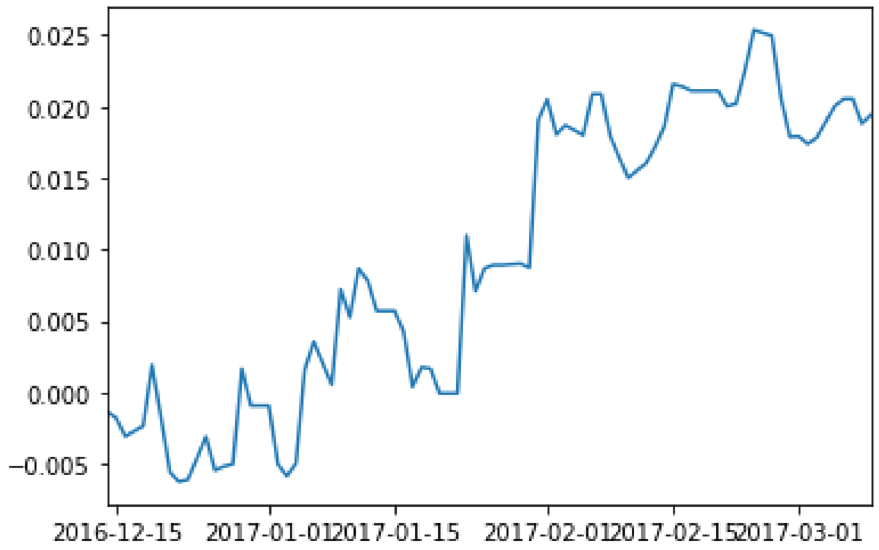
And here is the graph of cumulative return:



The results of backtesting of XAU on the test set with no B/O spread and immediate execution are shown below:

Statistics	dailly_r_mean	dailly_r_var	anu_sharpe	max_drawdown
Values	0.000319	0.00587	0.863676	0.069232

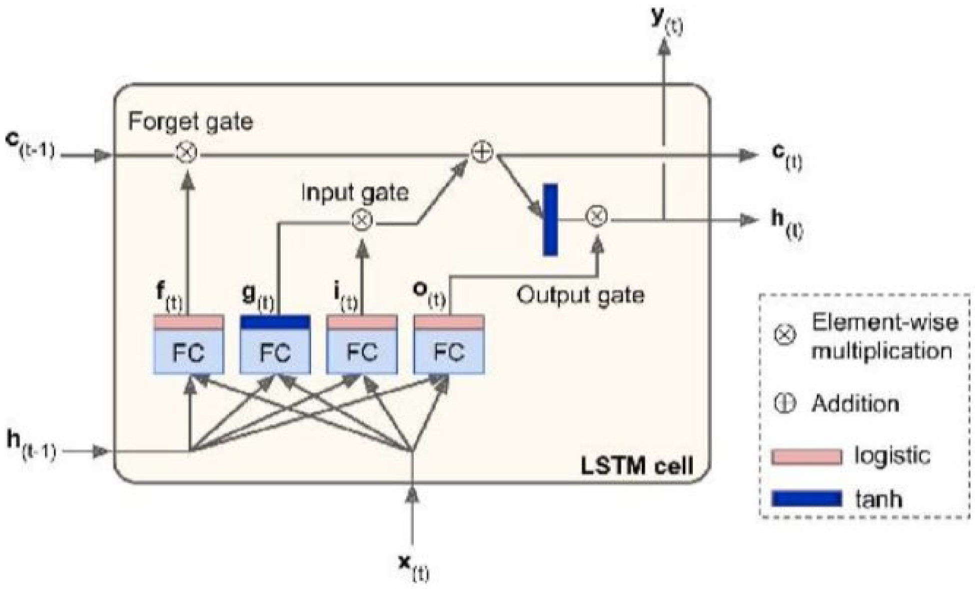
And here is the graph of cumulative return:



6.2. Deep Learning Method -- LSTM

6.2.1. Introduction of LSTM Cell

LSTM, Long short-term memory, is specific deep learning model developed for time series data analysis. LSTM cell replaces ordinary neuron cells in deep learning method. Here is a picture show what LSTM cell looks like:



Except for x_t as normal input, it needs c_{t-1}, the long term signal from previous cells and h_{t-1}, short term signal from the previous cells. There are three gates: forget gate, input gate and output gate to make lstm works. And it generates two signals: c_t, the long term signal and h_t, the short term signal. The short term signal is also the prediction of y_t.

6.2.2. Our LSTM Model

Our model looks like this:

```
np.random.seed(42)
tf.random.set_seed(42)

model = keras.models.Sequential([
    keras.layers.LSTM(20, return_sequences=False, input_shape=[None, 2]),
    keras.layers.Dense(1,activation='sigmoid')
])
optimizer = keras.optimizers.Adam(learning_rate=0.00001)
model.compile(loss="binary_crossentropy", optimizer=optimizer)
```

There are two layers of our deep learning model. The first layer is a LSTM layer. Assuming we use 13 periods to do our prediction. Then 'None' in the input shape equals to 13. There are a total of 13 (period) × 20 (units per period) lstm cells. The second layer is a regular dense lay with sigmoid activation function in order to make output between 0 and 1 interpreted as probability. For each period, there are 20 output from lstm layer which would be the input of the regular dense layer and 20 × 2 (c_t and h_t) outputs from lstm layer of this period which would be the input of lstm layer of the next period.

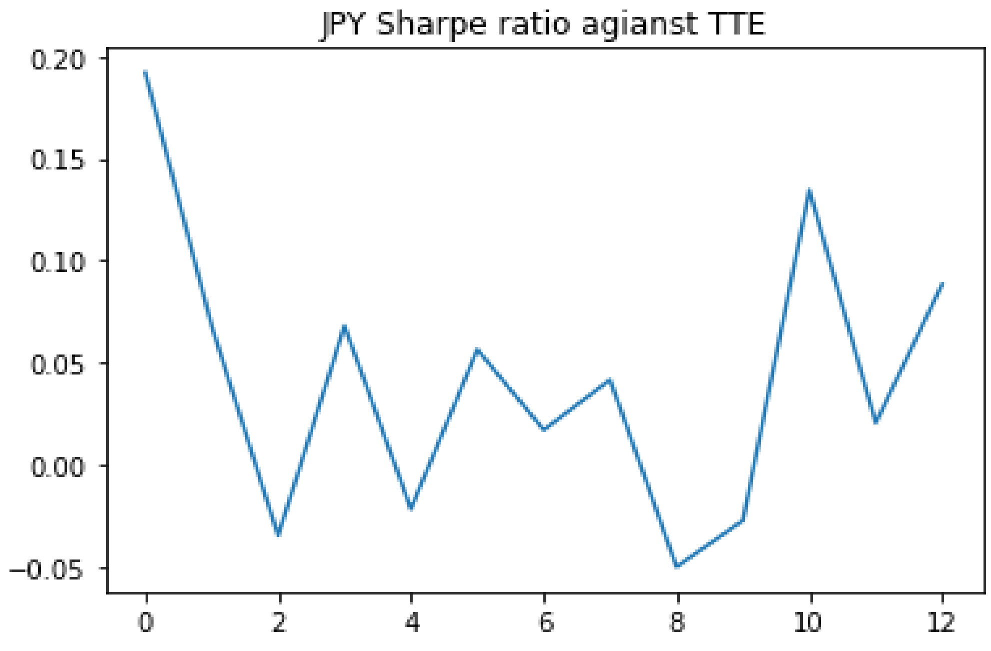
We settled down the learning_rate to be 0.00001to make sure the gradient descends at a suitable pace and use 'binary_crossentropy', which is the log loss as loss function.

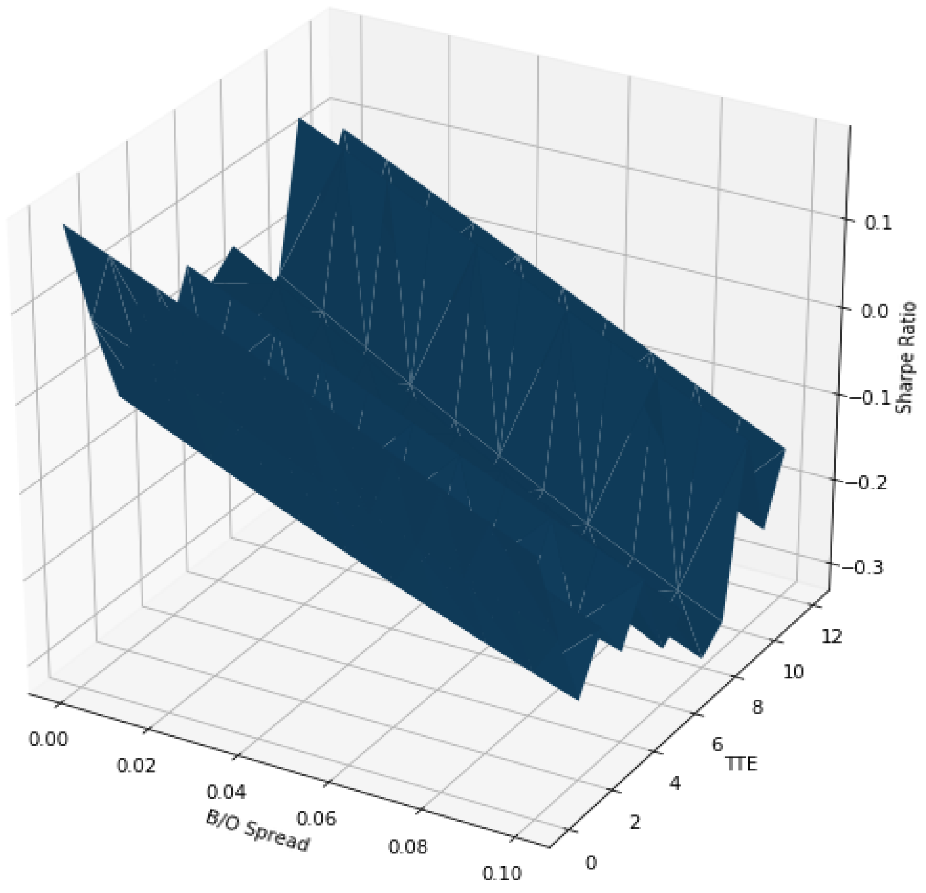
6.2.3. LSTM Model Backtesting Result

The results of backtesting of Japanese Yen on the test set with no B/O spread and immediate execution are shown below:

Statistics	dailly_r_mean	dailly_r_var	anu_sharpe	max_drawdown
Values	0.000653	0.003419	3.029932	0.015367

And here are the graphs of the daily Sharpe Ratio against different B/O spread and TTE:

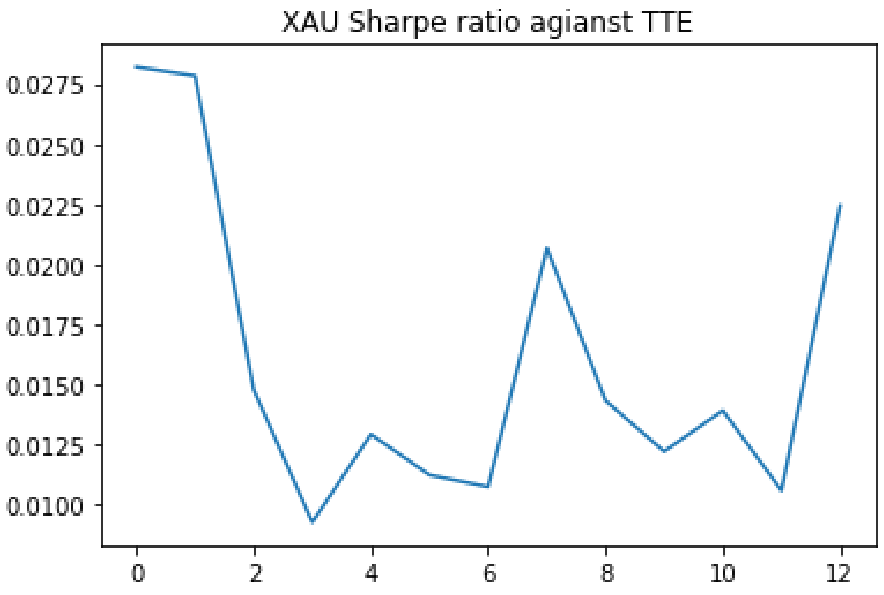

JPY Sharpe ratio agianst TTE
0.20
0.15
0.10
0.05
0.00
−0.05
0
2
4
6
8
10
12

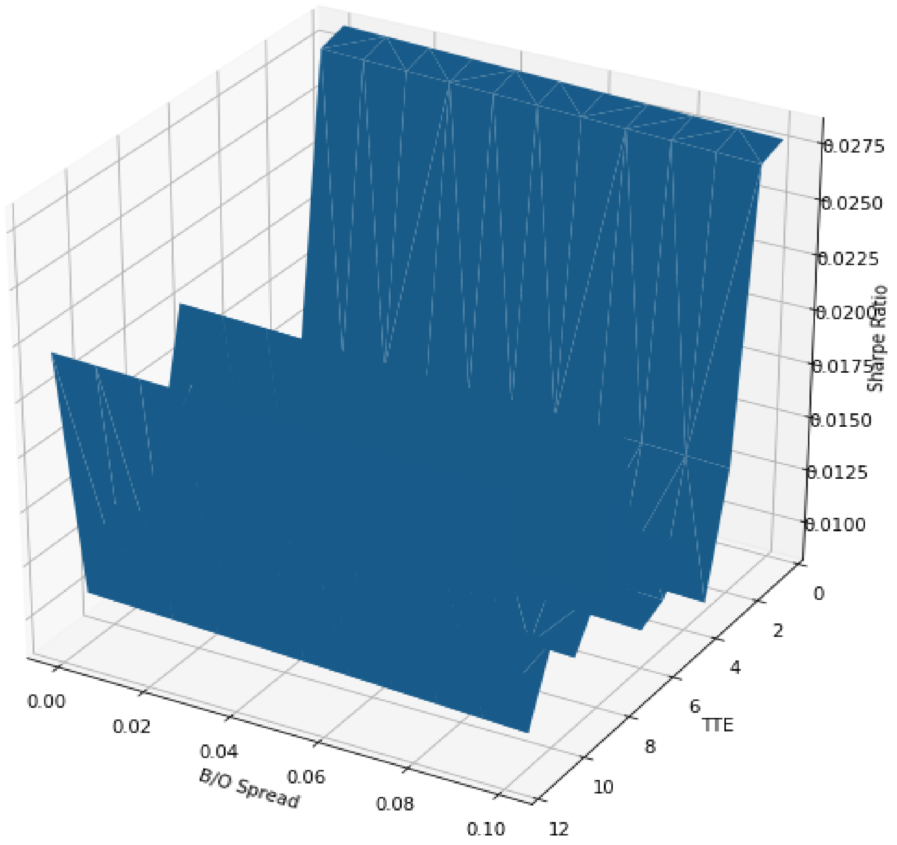

0.1
0.0
−0.1
−0.2
−0.3
Sharpe Ratio
0.00
0.02
0.04
0.06
0.08
0.10
B/O Spread
0
2
4
6
8
10
12
TTE

The results of backtesting of XAU on the test set with no B/O spread and immediate execution are shown below:

Statistics	dailly_r_mean	dailly_r_var	anu_sharpe	max_drawdown
Values	0.000221	0.007872	0.445894	0.085921

And here are the graphs of the Sharpe Ratio against different B/O spread and TTE:




0.0275
0.0250
0.0225
0.0200
0.0175
0.0150
0.0125
0.0100
Sharpe Ratio
0
2
4
6
8
10
12
TTE
0.00
0.02
0.04
0.06
0.08
0.10
B/O Spread

7. Conclusion

In this project, we build complete workflow for alpha engine in algorithm trading field: data and factor preparation, model (alpha signal) development and strategy back-testing.

For data and factor preparation part, based on the given price and order flow data, we build more technical indicators, apply extra trees classifier to select important factors and conduct correlation analysis to reduce collinearity. This help us select effective factors including: RSI, correlation of adjusted flow and price, variance and EWMA, which have relatively higher importance and independence.

For alpha engine part, we compare the performance of three types of trading strategies, single factor model, trend following model and machine learning algorithms :

- The first one is single factor model, which is based on the range of relative strength index. This factor acts as a momentum oscillator, measuring the velocity and magnitude of directional price movements. It's performance is very sensitive to predefined threshold of RSI values, and these thresholds vary largely in different asset classes. And it's also highly influenced by the execution delay, since it's a strategy based on following the recent action of institutions and large trades. But the advantage her is that the factor has a very clear financial meaning, easy to interpret and implement.

- The trend following model combines multiple factors, including variance, adjusted flow, moving average price and RSI. It detects the market regimes first and then adopt trend following strategy according to market conditions. This strategy add more filters to detect trend and thus become more strict in generating trading signals. As a result, it effectively reduce the risk but also have a relatively lower IR, for avoiding potential loss caused by wrong market judgement and losing some trading opportunities at the same time.

- In machine learning strategies, we utilize Light Gradient Boosting Machine (tree based learning algorithms) and Long Short-Term Memory model (deepfor time series data analysis) to predict the price movement in the next period. These two strategies have the highest IR, and not very sensitive to execution delay time. Machine learning models can accelerate the search for effective algorithmic trading strategies by automating what is often a tedious, offer the ability to move from finding associations based on historical data to identifying and adapting to trends as they develop. Therefore, they can capture more information from the market data and continuously adapt to the new market conditions. However, the limit is that the model is hard to interpret and the their performance highly depend on the hyperparameters, which make it hard to explain to our strategy to clients and we also need to improve the models' generality.

8. Appendix : Instruction For Use of Python Files

8.1. **Data_Factor_prepare.ipynb**

- Preprocessing our raw data
- Constructing factors
- Doing factor importance selection
- Correlation analysis

8.2. **RSI_Trend_Strategy.ipynb**

- Constructing single factor model (RSI)
- Constructing trend following strategy
- Tuning parameter

8.3. **LSTM. ipynb, LGBM.ipynb**

- Constructing machine learning models
- Trian, validation, test set splitting
- Tuning hyperparameters
- Plotting 3-D figures of IR change with bid ask spread and execution delay

8.4. **Backtest.py**

- The back-testing framework for strategies
- Calculating trading signals and optimal holding
- Calculating PnL, return, volatility, maximum drawdown
- Calculating all performance stats changing with bid ask spread and execution delay