Cryptocurrency Market

machine learning for predicting price and trading strategy

Shan Xu
Junyuan Huang
Ting Shen
Jae Cho

Agenda

- Data Exploration

- Predicting Price Modelling
  - ARIMA
  - RNN LSTM
- Trading strategy
  - Pair trading
  - Google trend
  - Logistic Regression and SVM Classification

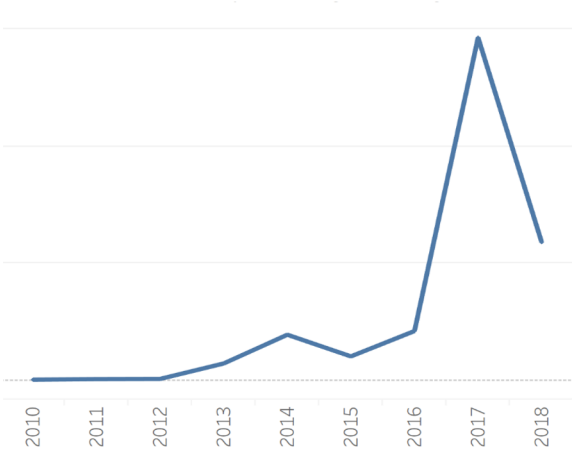
- Future further improvements

Bitcoin Price

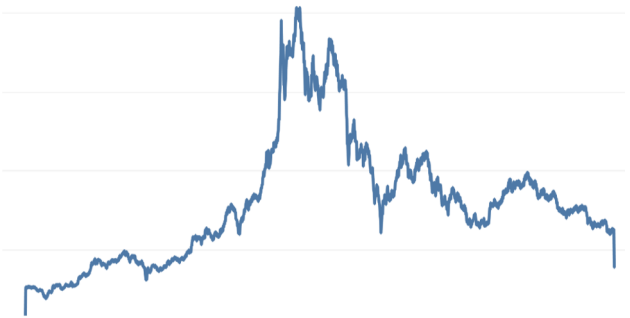
- Sudden surge beginning 2017
  - Idea of blockchain
  - Millionaires
- And decline
  - Regulation
  - Bubble

Data Exploration

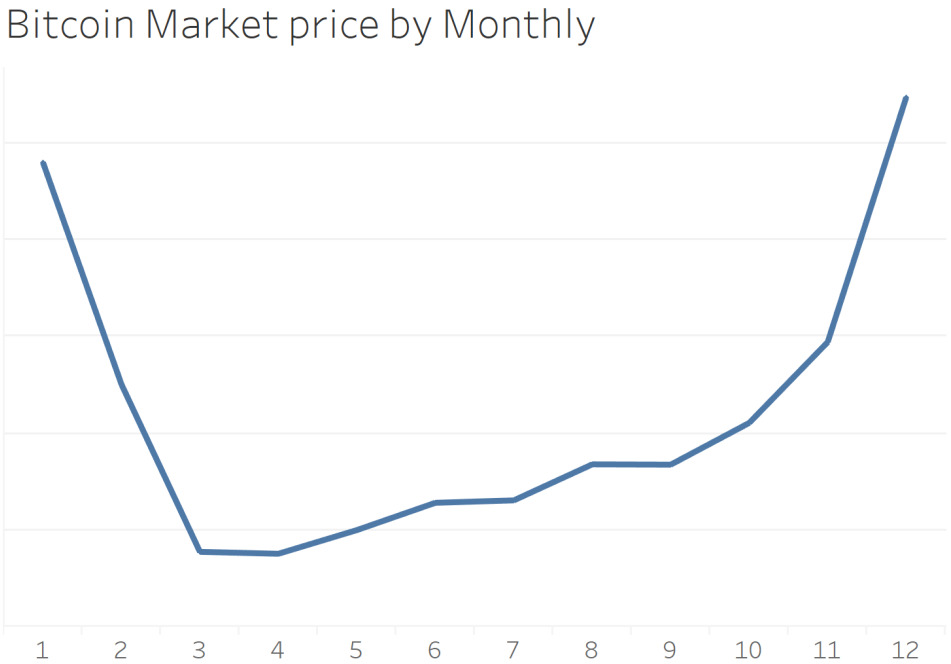
Bitcoin market price by yearly



Bitcoin closing price 10 minutes
2017~2018



Data Exploration

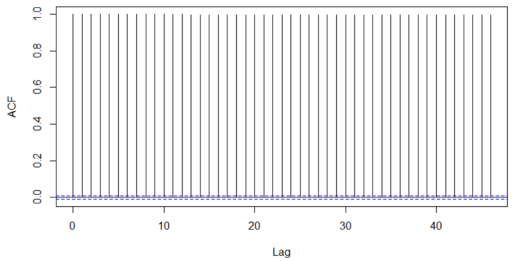


Data Exploration

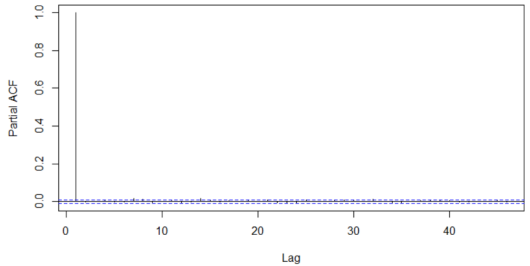
- Bitcoin historical data
- 511,576 rows
- Variables
  - Open- Bitcoin price in Currency units at time period open
  - High- Highest Bitcoin price in Currency units during time period
  - Low- Lowest Bitcoin price in Currency units during time period
  - **Close- Bitcoin price in Currency units at time period close**
    - *"**Closing price**" generally refers to the last **price** at which a **stock** trades during a regular trading session.*
  - Volume (BTC) - Volume of BTC transacted in time period
  - Volume (Currency)- Volume of Currency transacted in time period
  - Volume-weighted average price (VWAP)

Close	High	Low	Open	Volume_(BTC)	Volume_(Currency)	Weighted_Price
296,540	296,558	296,016	296,127	1.159	343,244.14	296,257.67
296,260	296,260	296,011	296,215	2.735	810,100.81	296,221.52
296,621	296,859	296,621	296,636	8.186	2,429,080.19	296,733.40
296,709	296,880	296,666	296,666	2.082	617,838.34	296,785.11
295,911	296,154	295,842	295,842	2.613	773,264.43	295,929.75
295,996	296,080	295,777	296,080	3.488	1,032,515.89	295,993.71
294,387	294,525	294,250	294,250	11.575	3,406,793.63	294,334.03
294,400	294,800	294,237	294,778	18.201	5,359,863.26	294,481.80
295,159	295,200	295,159	295,200	0.830	245,014.77	295,198.52
295,335	295,345	294,911	295,112	0.144	42,459.29	295,022.12
295,800	295,800	295,494	295,600	8.548	2,527,633.91	295,703.03
295,395	295,550	295,108	295,200	1.090	321,926.25	295,252.53
294,179	294,261	293,978	294,160	6.236	1,833,737.63	294,035.21
293,639	293,811	293,362	293,811	3.531	1,036,307.27	293,455.07
292,362	292,554	292,362	292,400	4.951	1,447,849.51	292,427.96
290,600	290,600	290,040	290,442	27.820	8,078,513.66	290,381.86
292,186	292,431	291,748	292,183	11.641	3,399,319.80	292,018.18

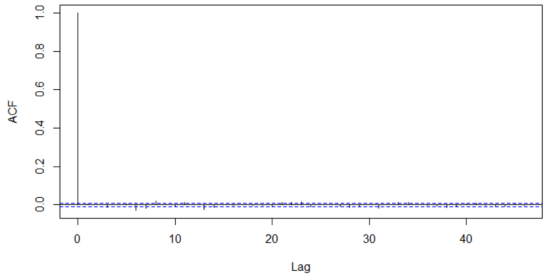
ACF

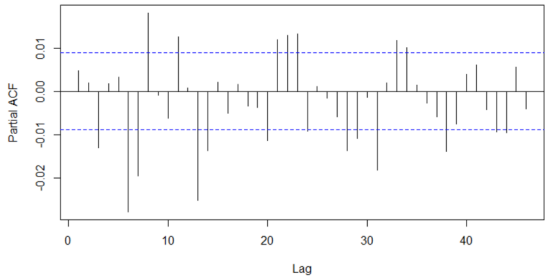


PACF



With Difference





Augmented Dickey-Fuller Test

```
        Augmented Dickey-Fuller Test

data:  ln_bit_price
Dickey-Fuller = -1.0975, Lag order = 36, p-value = 0.9238
alternative hypothesis: stationary
```

With Difference

```
        Augmented Dickey-Fuller Test

data:  dif_bit_price
Dickey-Fuller = -37.331, Lag order = 36, p-value = 0.01
alternative hypothesis: stationary
```

ARIMA Model Using R

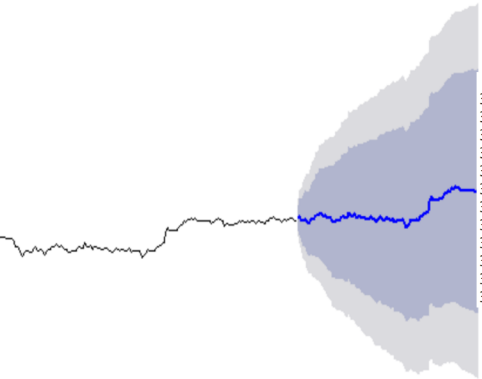
ARIMA Model

- auto.arima function
  - Enforce seasonality (D=1)

```
> auto.arima(ts_bitcoin, D=1)
Series: ts_bitcoin
ARIMA(0,1,1)(0,1,0)[144]
```

- RMSE = 0.007240218
- MSE = 5.242076e-05

Forecasted values



	Point Forecast	Lo 80	Hi 80	Lo 95	Hi 95
335.4306	13.63984	13.63054	13.64913	13.62562	13.65405
335.4375	13.64078	13.62760	13.65397	13.62062	13.66095
335.4444	13.63792	13.62175	13.65408	13.61320	13.66264
335.4514	13.63769	13.61902	13.65637	13.60913	13.66625
335.4583	13.63903	13.61815	13.65992	13.60709	13.67098
335.4653	13.63742	13.61453	13.66030	13.60242	13.67241
335.4722	13.63515	13.61042	13.65987	13.59734	13.67296
335.4792	13.63829	13.61186	13.66472	13.59786	13.67871
335.4861	13.63912	13.61108	13.66715	13.59624	13.68200
335.4931	13.63965	13.61010	13.66921	13.59445	13.68486
335.5000	13.64161	13.61060	13.67261	13.59419	13.68902
335.5069	13.64146	13.60908	13.67384	13.59194	13.69098

1. Mean
2. Exp

	Actual price	Forecasted Price
1	837768	838890.4
2	837500	839684.8
3	837076	837283.2
4	837702	837094.8
5	837500	838216.8
6	837999	836862.5
7	837128	834966.5
8	837122	837594.4
9	837800	838287.5
10	839265	838738.9
11	838932	840376.9
12	839250	840254.0
13	839500	841549.0
14	838501	840933.8
15	839200	840039.0
16	839892	840541.7
17	840500	838933.4
18	840522	838720.5
19	840850	838564.9
20	842552	836610.6
21	840267	836485.7

- Percent error

$$\frac{|\text{Guess Number} - \text{Result Number}|}{|\text{Result Number}|}$$

- Mean Percent Error = -0.0364

Overview

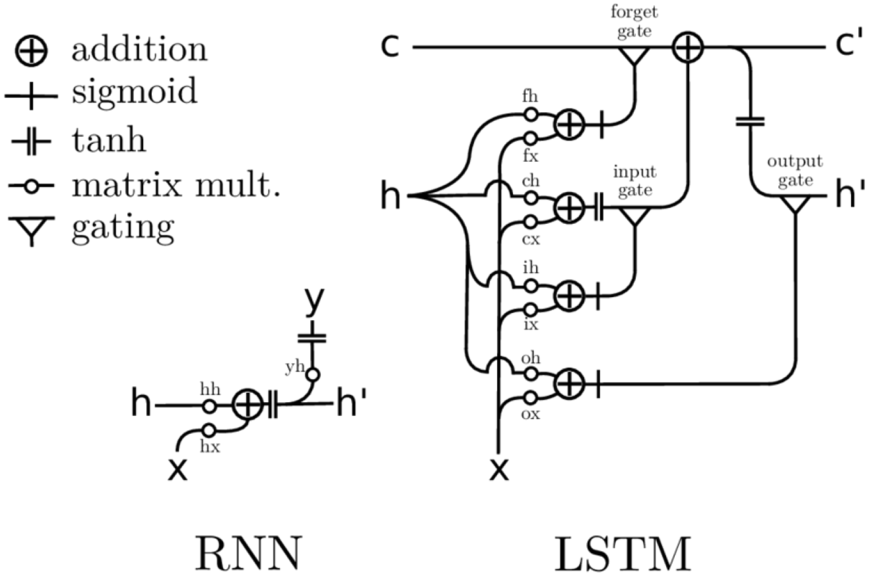
- Data

  Bitcoin historical data (512k x 8)
- Model

  Recurrent Neural Network
- Implementation

  Python code based on Sklearn, Keras, Tensorflow

RNN LSTM


addition
sigmoid
tanh
matrix mult.
gating
y
yh
h
hh
h'
hx
x
forget gate
c
c'
fh
fx
ch
cx
input gate
output gate
h
h'
ih
ix
oh
ox
x
RNN
LSTM

Steps

- Load file (bitflyerJPY_1-min_data_2017-07-04_to_2018-06-27.csv)
- Preprocessing raw data
- Define and training model with train data
- Testing with test data
- Prediction for future events

Preprocessing

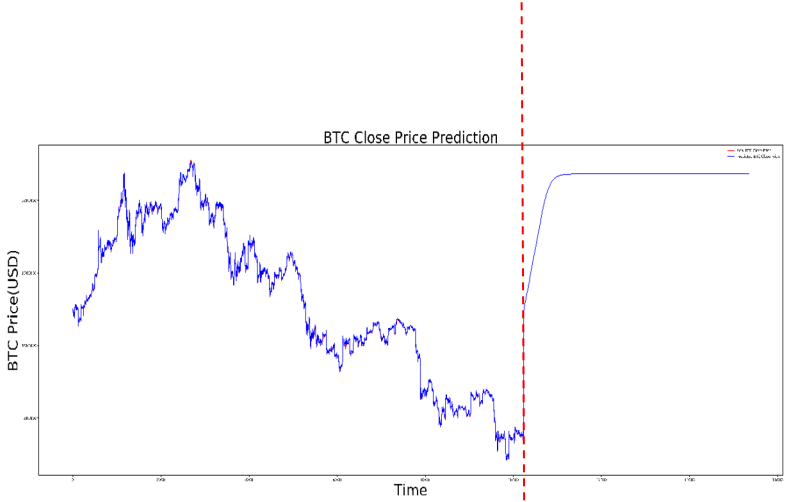
- Sampling the raw data (1 min → 10 min intervals, every 10th)
- Pick up the ‘Close’ column from raw ndarray
- Split data into train_data and test_data
- Normalization:Scale between (-1, 1)
- Reshape the input format

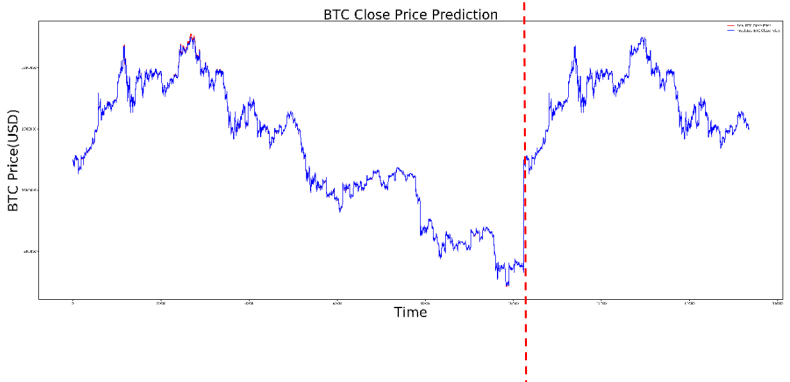
Define Model

- Using keras to simplify the building procedure
- Initialize the RNN with a Sequential model
- Adding the input layer, LSTM cell, output layer (Dense)
- Compile the RNN
- Using training set (x_train, y_train) to fit the model

Prediction

- Moving window algorithm
  - Predict each future step, update moving window
- Predict future based on multiple previous steps
  - Use whole test data
- Result
  - Good match for test data
  - Different patterns for prediction





Runlog

```
Train on 34787 samples, validate on 6139 samples
Epoch 1/10
 - 77s - loss: 0.0113 - val_loss: 1.8967e-05
Epoch 2/10
 - 76s - loss: 5.3860e-05 - val_loss: 1.4431e-05
Epoch 3/10
 - 76s - loss: 5.3834e-05 - val_loss: 1.4258e-05
Epoch 4/10
 - 76s - loss: 5.3736e-05 - val_loss: 1.3904e-05
Epoch 5/10
 - 76s - loss: 5.3368e-05 - val_loss: 1.3489e-05
Epoch 6/10
 - 76s - loss: 5.3542e-05 - val_loss: 1.6597e-05
Epoch 7/10
 - 76s - loss: 5.3521e-05 - val_loss: 1.4016e-05
Epoch 8/10
 - 76s - loss: 5.3507e-05 - val_loss: 2.1722e-05
Epoch 9/10
 - 75s - loss: 5.3415e-05 - val_loss: 1.4737e-05
Epoch 10/10
 - 83s - loss: 5.3048e-05 - val_loss: 1.7105e-05
train MSE =  4.3e-05
test MSE =  0.000117
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_1 (LSTM)                (1, 4)                    96
_________________________________________________________________
dense_1 (Dense)              (1, 1)                    5
=================================================================
Total params: 101
Trainable params: 101
Non-trainable params: 0
```

Trading Strategy

Logistic Regression and SVM Classification
Google trend
Pair trading

Logistic Regression

```
In [125]: bit_train_7d = bit_train[[x for x in bit_train.columns if 'Close' in x]].iloc[7:]
```

```
In [126]: bit_train_7d.head(10)
```

Out[126]:

Date	Close	Close - 1d	Close - 2d	Close - 3d	Close - 4d	Close - 5d	Close - 6d	Close - 7d
2013-05-06	112.30	115.91	112.50	97.75	105.21	116.99	139.00	144.54
2013-05-07	111.50	112.30	115.91	112.50	97.75	105.21	116.99	139.00
2013-05-08	113.57	111.50	112.30	115.91	112.50	97.75	105.21	116.99
2013-05-09	112.67	113.57	111.50	112.30	115.91	112.50	97.75	105.21
2013-05-10	117.20	112.67	113.57	111.50	112.30	115.91	112.50	97.75
2013-05-11	115.24	117.20	112.67	113.57	111.50	112.30	115.91	112.50
2013-05-12	115.00	115.24	117.20	112.67	113.57	111.50	112.30	115.91
2013-05-13	117.98	115.00	115.24	117.20	112.67	113.57	111.50	112.30
2013-05-14	111.50	117.98	115.00	115.24	117.20	112.67	113.57	111.50
2013-05-15	114.22	111.50	117.98	115.00	115.24	117.20	112.67	113.57

```
In [129]: lm = linear_model.LogisticRegression(C=1000)
```

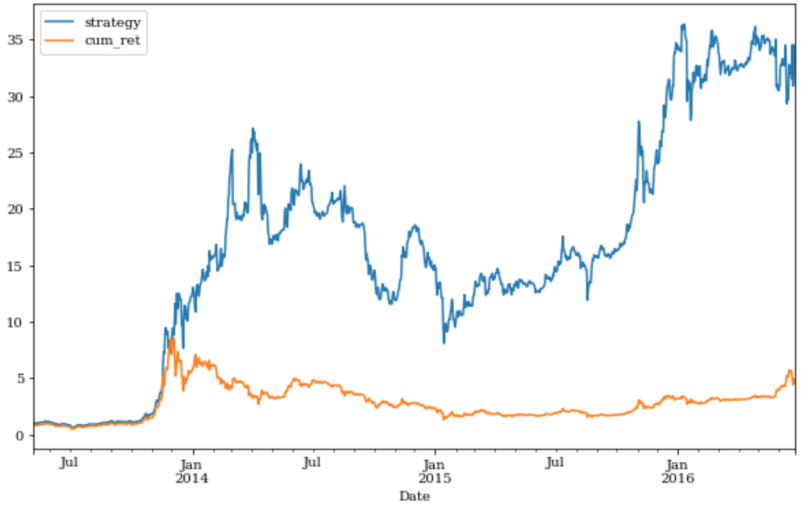
```
In [130]: # Get training set's labels;
          y_train = np.sign(bit_train_7d['Close'].pct_change().shift(-1))
          y_train.replace(to_replace= np.NaN, value = 0,inplace = True)
          y_train = y_train.reshape(-1,1)
          y_train[-10:]
```

```
Out[130]: array([[-1.],
                 [ 1.],
                 [ 1.],
                 [-1.],
                 [-1.],
                 [-1.],
                 [ 1.],
                 [ 1.],
                 [-1.],
                 [ 0.]])
```

Logistic Regression

Training:

```
In [137]: bit_train[['strategy','cum_ret']].dropna().plot(figsize=(10, 6))
Out[137]: <matplotlib.axes._subplots.AxesSubplot at 0x1a173e7860>
```



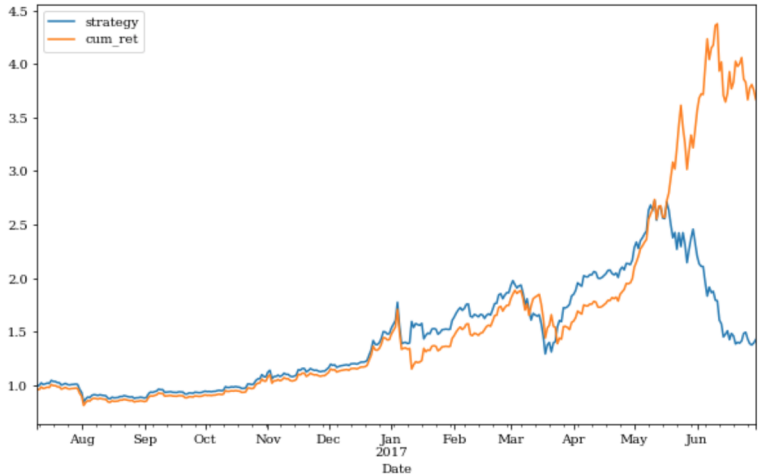
Logistic Regression

Testing:

```
In [158]: bit_test['strategy'] = (bit_test['prediction'].shift(1) * bit_test['returns'] + 1).cumprod()
          bit_test['cum_ret'] = (bit_test['returns']+1).cumprod()

In [159]: bit_test[['strategy','cum_ret']].dropna().plot(figsize=(10, 6))

Out[159]: <matplotlib.axes._subplots.AxesSubplot at 0x1a17a42d68>
```



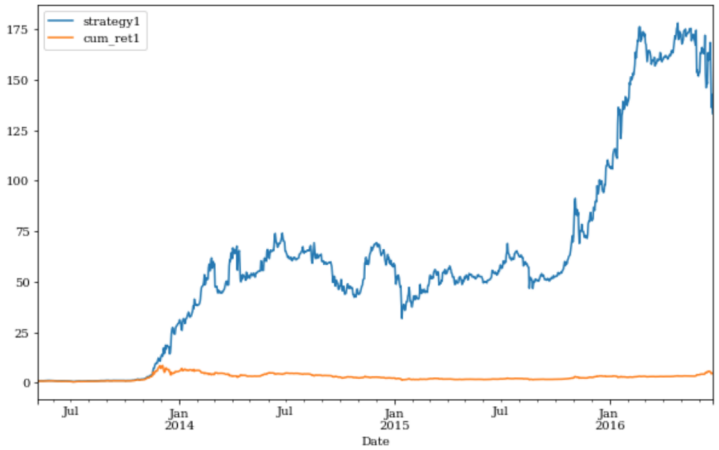
SVM

Training:

```
In [143]: bit_train['strategy1'] = (bit_train['prediction1'].shift(1) * bit_train['returns'] + 1).cumprod()
          bit_train['cum_ret1'] = (bit_train['returns']+1).cumprod()

In [144]: bit_train[['strategy1','cum_ret1']].dropna().plot(figsize=(10, 6))

Out[144]: <matplotlib.axes._subplots.AxesSubplot at 0x1a17712518>
```



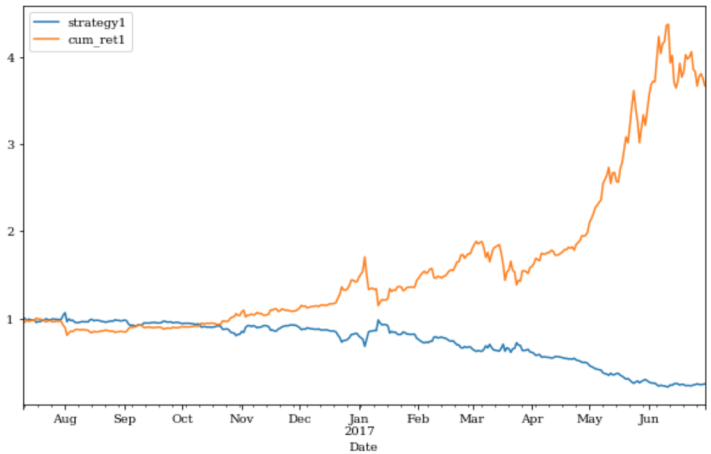
SVM

Testing

```
In [162]: bit_test['strategy1'] = (bit_test['prediction1'].shift(1) * bit_test['returns'] + 1).cumprod()
          bit_test['cum_ret1'] = (bit_test['returns']+1).cumprod()

In [163]: bit_test[['strategy1','cum_ret1']].dropna().plot(figsize=(10, 6))

Out[163]: <matplotlib.axes._subplots.AxesSubplot at 0x1a17c17780>
```



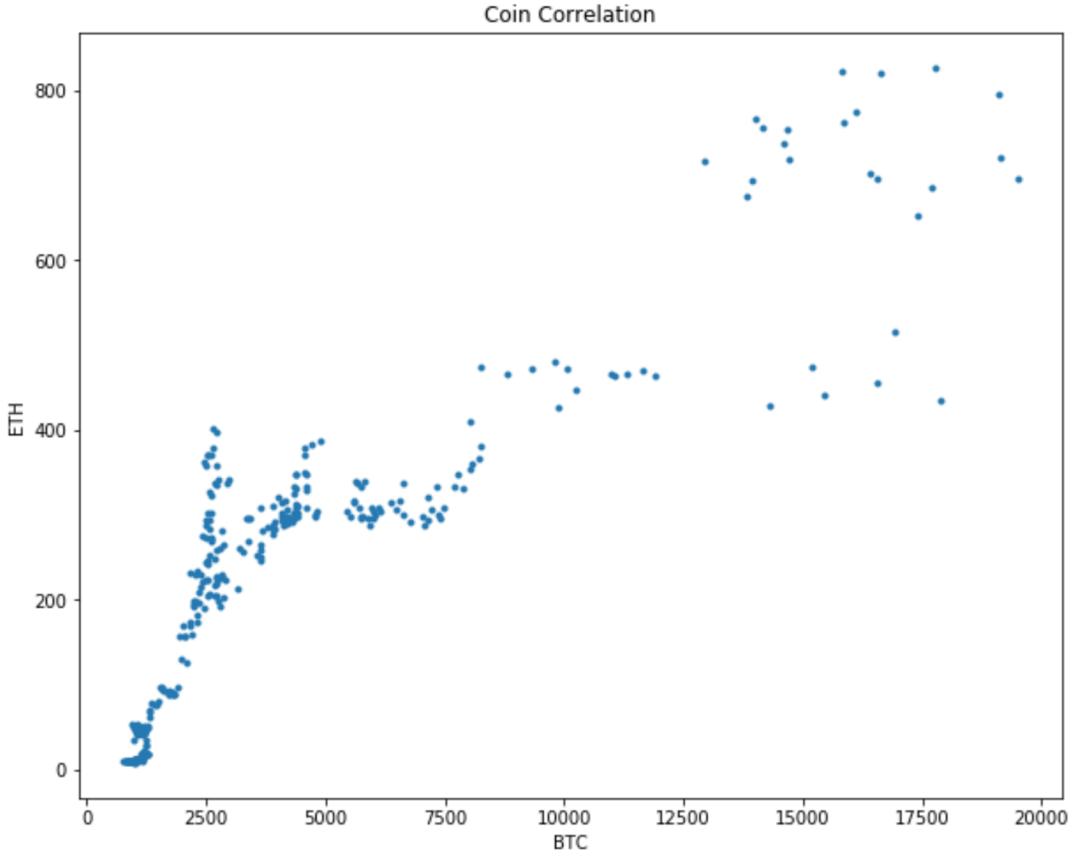
Convergence, 1952 by Jackson Pollock

Pair Trading —— cointegration

- Two time series are non-stationary
- The linear combination of the two time series is stationary
- The target of investing is not one, but two highly related assets

  1, Pepsi and Coca-Cola ;
  2, Goldman Sachs (GS) and Morgan Stanley (MS);
  3, Bitcoin and Ethereum

- Based on mean-reverting character


Coin Correlation
800
600
400
200
0
ETH
0
2500
5000
7500
10000
12500
15000
17500
20000
BTC

In [121]:
```
[slope, intercept] = np.polyfit(data.iloc[:,0], data.iloc[:,1], 1).round(2)
slope,intercept
```

Out[121]:
```
(0.040000000000000001, 60.469999999999999)
```

In [122]:
```
data['spread'] = data.iloc[:,1] - (data.iloc[:,0]*slope + intercept)
```

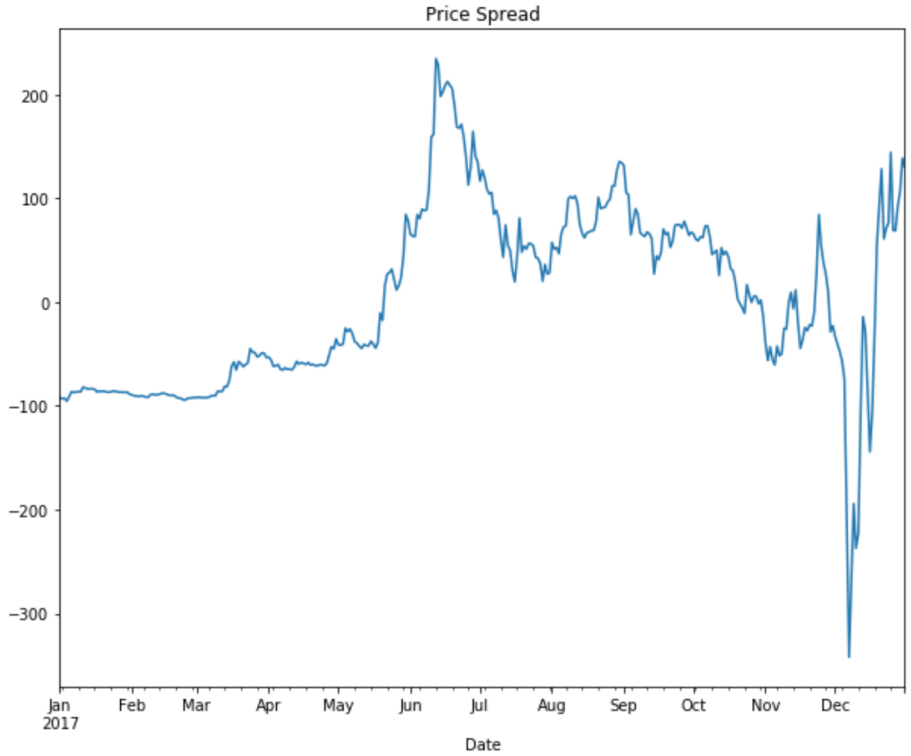
In [123]:
```
data.head()
```

Out[123]:

Date	BTC	ETH	spread
2017-01-01	998.33	8.17	-92.2332
2017-01-02	1021.75	8.38	-92.9600
2017-01-03	1043.84	9.73	-92.4936
2017-01-04	1154.73	11.25	-95.4092
2017-01-05	1013.38	10.25	-90.7552

```
In [124]: data['spread'].plot(figsize = (10,8),title = 'Price Spread')
```

```
Out[124]: <matplotlib.axes._subplots.AxesSubplot at 0x1a1e88d518>
```



In [125]:
```
data['zscore'] = (data['spread'] - data['spread'].mean())/data['spread'].std()
```

In [126]:
```
data.head()
```

Out[126]:

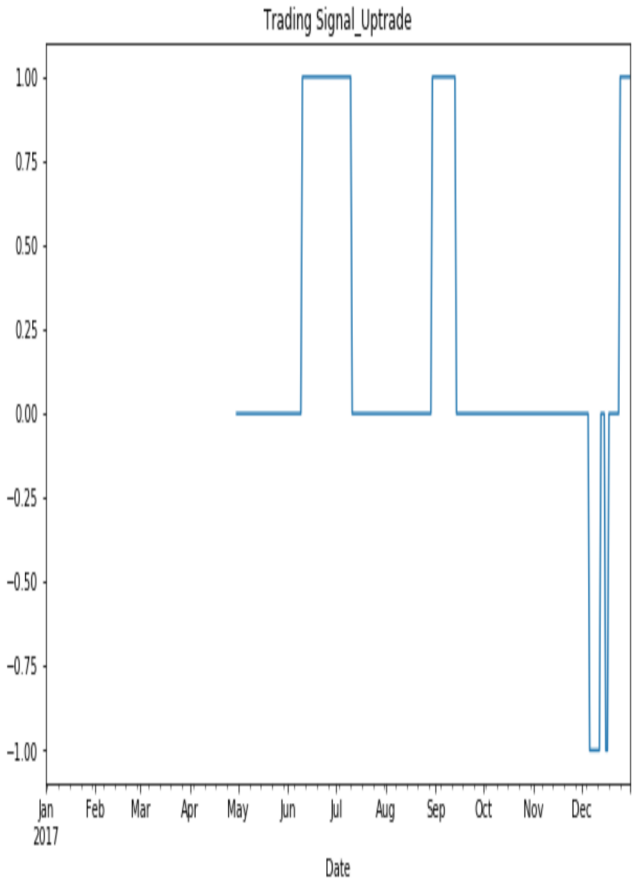
Date	BTC	ETH	spread	zscore
2017-01-01	998.33	8.17	-92.2332	-1.101312
2017-01-02	1021.75	8.38	-92.9600	-1.109699
2017-01-03	1043.84	9.73	-92.4936	-1.104317
2017-01-04	1154.73	11.25	-95.4092	-1.137962
2017-01-05	1013.38	10.25	-90.7552	-1.084257

In [127]:
```
data['zscore'].plot(figsize = (10,8),title = 'Z-score')
plt.axhline(1.5)
plt.axhline(0)
plt.axhline(-1.5)
```

Out[127]: <matplotlib.lines.Line2D at 0x1a1ee9f550>

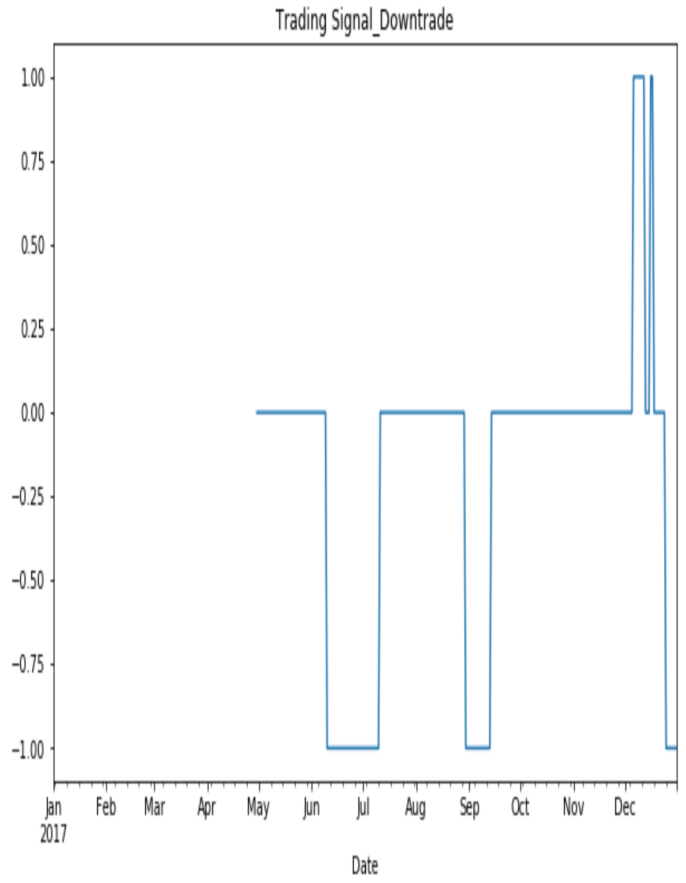
In [130]:
```
data['position_1'].plot(ylim=[-1.1, 1.1], figsize=(10, 6),title = 'Trading Signal_Uptrade')
```

Out[130]: <matplotlib.axes._subplots.AxesSubplot at 0x1a1eaeee80>



In [132]:
```
data['position_2'].plot(ylim=[-1.1, 1.1], figsize=(10, 6),title = 'Trading Signal_Downtrade')
```

Out[132]: <matplotlib.axes._subplots.AxesSubplot at 0x1a1ee46da0>



In [133]:
```
data['returns_1'] = np.log(data['BTC'] / data['BTC'].shift(1))
```

In [134]:
```
data['returns_2'] = np.log(data['ETH'] / data['ETH'].shift(1))
```

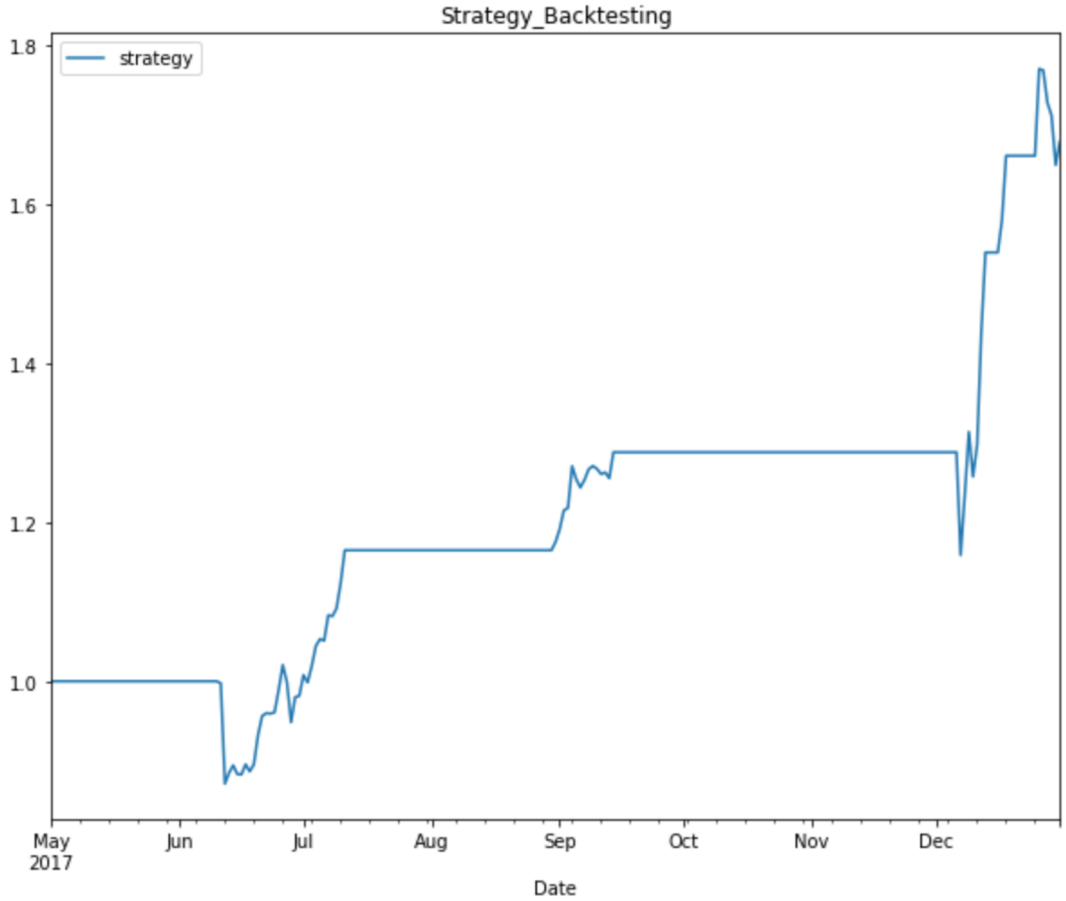
In [135]:
```
data['strategy'] = 0.5*(data['position_1'].shift(1) * data['returns_1'])+0.5*(data['position_2'].shift(1) * data['retur
```

In [136]:
```
data[['returns_1','returns_2','strategy']].dropna().cumsum().apply(np.exp).plot(figsize=(10, 8),title = 'Strategy_Backt
```

Out[136]: <matplotlib.axes._subplots.AxesSubplot at 0x1a1f248940>

In [137]:
```
data[['strategy']].dropna().cumsum().apply(np.exp).plot(figsize=(10, 8),title = 'Strategy_Backtesting')
```

Out[137]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a1f45f6a0>
```



The limitation of Pair trading

The Spread might not converge.

The Spread might not deviate.

The coefficients of regression must be updated frequently.

Google trend

Initial idea: weather “bitcoin” word popularity related to the bitcoin price.

Check the pattern of bitcoin price and the google trend of the word “bitcoin”

Quantifying Trading strategy :

If the Google “bitcoin” search volume for the week > average for the past three weeks, then long bitcoin and hold the position for one week.

If the Google “bitcoin” search volume for the week < average for the past three weeks, then short bitcoin and hold the position for one week.

Google trend

nature.com > scientific reports > articles > article

MENU

SCIENTIFIC REPORTS

Article | OPEN | Published: 25 April 2013

Quantifying Trading Behavior in Financial Markets Using *Google Trends*

Tobias Preis, Helen Susannah Moat & H. Eugene Stanley

Scientific Reports **3**, Article number: 1684 (2013) | Download Citation

Abstract

Crises in financial markets affect humans worldwide. Detailed market data on trading decisions reflect some of the complex human behavior that has led to these crises. We suggest that massive new data sources resulting from human interaction with the Internet may offer a new perspective on the behavior of market participants in periods of large market movements. By analyzing changes in *Google* query volumes for search terms related to finance, we find patterns that may be interpreted as “early warning signs” of stock market moves. Our results illustrate the potential that combining extensive behavioral data sets offers for a better understanding of collective human behavior.

Summary and further improvement

- RNN LSTM Models has better performance than ARIMA
- Features, Predictors, factors, ,regressors, X

CCI = (Typical Price - 20-period SMA of TP) / (.015 x Mean Deviation)
Rate of Change (ROC) the % between the most recent price and the price “n” day’s ago.
Bollinger Bands, Etc….

- Cross validation for better hyper parameter
- Using some active investment indicator like “ Sharpe ratio, information ratio” to measure or monitor
- Be awe in the market
  - a feeling of reverential respect mixed with fear or wonder.

References

https://www.researchgate.net/figure/RNN-and-LSTM-A-graphical-representation-of-the-RNN-and-LSTM-networks-are-shown-The_fig13_304346489

https://towardsdatascience.com/bitcoin-price-prediction-using-lstm-9eb0938c22bd

http://intelligentonlinetools.com/blog/2018/04/03/machine-learning-stock-market-prediction-lstm-keras/

https://www.nature.com/articles/srep01684

Thank you