A Real-time System of Bitcoin Price Prediction

Zhao Cai

赵 才

Group 8

Bitcoin(~$3712.20)

- Bitcoin is a cryptocurrency, a form of electronic cash. It is a decentralized digital currency without a central bank or single administrator that can be sent from user-to-user on the peer-to-peer bitcoin network without the need for intermediaries (https://en.wikipedia.org/wiki/Bitcoin).

Method

- **Use the method of Bayesian regression and its efficacy for predicting price variation of Bitcoin**
- **This idea is from the paper published by MIT Professor Shah in Oct 2014. And I have done some optimizations basing on this method**
- **Reference: Devavrat Shah, Kang Zhang Bayesian regression and Bitcoin https://arxiv.org/abs/1410.1231**
- **Based on this price prediction method, they devise a simple strategy for trading Bitcoin. The strategy is able to nearly double the investment in less than 60 day period when run against real data trace.**

Method(from paper)

- 1. use the historic time series to generate three subsets of time-series data of three different lengths: S_1 of time-length 30 minutes, S_2 of time-length 60 minutes, and S_3 of time-length 120 minutes.
- 2. at a given point of time, to predict the future change Δp, use the historical data of three length: previous 30 minutes, 60 minutes and 120 minutes - denoted x^1 , x^2 and x^3
- 3.use x^j with historical samples S^j for Bayesian regression to predict average price change Δp^j for $1 \leq j \leq 3$.
- 4. calculate r = (vbid − vask)/(vbid + vask) where vbid is total volume people are willing to buy in the top 60 orders and vask is the total volume people are willing to sell in the top 60 orders based on the current order book data.
- 5.The final estimation Δp is produced as

$$\Delta p = w_0 + \sum_{j=1}^{3} w_j \Delta p^j + w_4 r, \tag{8}$$

where $\mathbf{w} = (w_0, \ldots, w_4)$ are learnt parameters.

Method(from paper)

Finding $S_j, 1 \leq j \leq 3$ and learning **w**

- 1. Utilize the first time period to find patterns S_j, $1 \leq j \leq 3$(previously divide the entire time duration into three, roughly equal sized, periods)
- 2. The second period is used to learn parameters **w** and the last third period is used to evaluate the performance of the algorithm.
- The learning of **w** is done simply by finding the best linear fit over all choices given the selection of S_j , $1 \leq j \leq 3$. Now selection of S_j, $1 \leq j \leq 3$.
- 3.take all possible time series of appropriate length (effectively vectors of dimension 180, 360 and 720 respectively for S_1,S_2 and S_3)
- Each of these form x_i and their corresponding label y_i is computed by looking at the average price change in the 10 second time interval following the end of time duration of x_i.
- 4. To facilitate computation on single machine with 128G RAM with 32 cores, clustered patterns in 100 clusters using k−means algorithm. From these, we chose 20 most effective clusters and took representative patterns from these clusters.

Method(from paper)

Finding S_j, $1 \le j \le 3$ and learning **w**

- 5. The one missing detail is computing 'distance' between pattern x and x_i , this is squared l_2-norm. use $\exp(c \cdot s(x,x_i))$ in place of $\exp(-||x - x_i||_2^2 * 0.25)$ with choice of constant c optimized for better prediction using the fitting data (like for **w**).

Problem and Task

- In the paper, they use 32 core machine with 128G RAM to train the model in in 1 second. That’s not slow, to some degree. Thanks to the powerful machine, they do not need to parallelize the computation
- But we do mot have this kind of machine. That is the problem.
- Using big data technologies can help to speed up the process of training the model (**Bayesian regression**), especially when we do not have powerful machines
- predict the price of Bitcoin using big data technologies

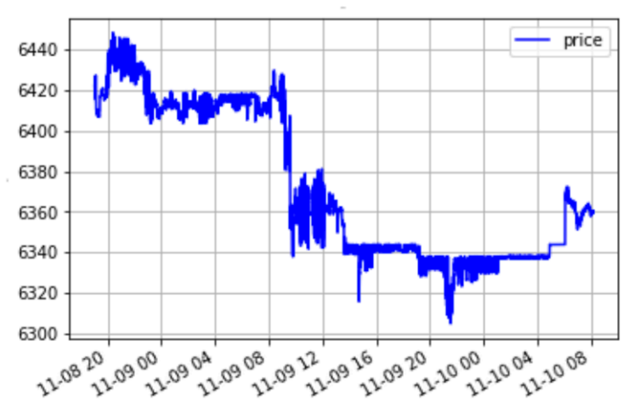
Data

- Bitcoin history price and order book are obtained from Okcoin.com every 10 second interval by its APIs
- OKCoin is a world leading digital asset trading platform. OKCoin currently provides fiat trading with major digital assets, including Bitcoin, Bitcoin Cash…
- run python program in Azure every day to request Bitcoin history price and order book from Okcoin.com and save data in MongoDB
- This real-time data collection mechanism allowed me to collect high-granularity Bitcoin price data and accumulate roughly 100,000 unique price points for use in our modeling step.

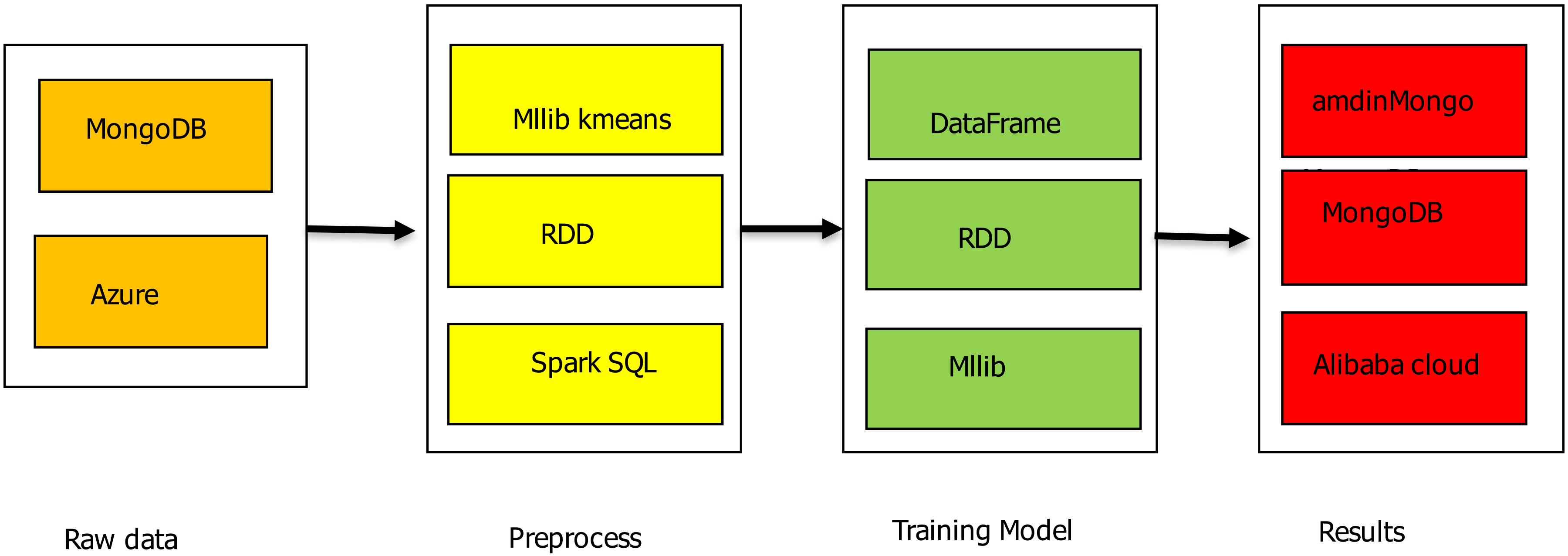
Data using rule

- Choose 40000 * 2/3 data points to train the model
- 2018-10-29~2018-11-08 20:00

- Choose last 40000 * 1/3 data points to test the model
- 2018-11-08 20:00~2018-11-10



System architecture



Data stored in MongoDB

- date : the timestamp of requesting data from okcoin
- price : the last price (close price)of Bitcoin at that time
- v_bid is total volume people are willing to buy in the top 60 orders and v_ask is the total volume people are willing to sell in the top 60 orders based on the current order book data.

date	price	v_ask	v_bid
2018-10-24 15:57:37	6414.87	131.07739999999998	82.9433
2018-10-24 15:58:07	6414.79	127.13759999999999	81.1229
2018-10-24 15:58:17	6414.79	126.60749999999999	84.08560000000001
2018-10-24 15:58:27	6415.19	127.80949999999999	84.08560000000001
2018-10-24 15:58:37	6415.19	127.80949999999999	84.08560000000001
2018-10-24 15:58:42	6415.19	127.01939999999999	84.08560000000001
2018-10-24 15:58:47	6415.19	127.01939999999999	84.08560000000001
2018-10-24 15:58:52	6415.19	127.01939999999999	84.08560000000001
2018-10-24 15:58:57	6415.19	127.55479999999999	84.08560000000001

Some Details of implementation(1)

- 1. use pyspark.sql to load data from MongoDB
- 2. pyspark –packages org.mongodb.spark:mongo-spark-connector_2.11:2.2.0

In [2]:
```
# connect to mongodb
my_spark = SparkSession \
    .builder \
    .appName("myApp") \
    .config("spark.mongodb.input.uri", "mongodb://127.0.0.1/okcoindb.historical_data") \
    .config("spark.mongodb.output.uri", "mongodb://127.0.0.1/okcoindb.historical_data") \
    .getOrCreate()
df = my_spark.read.format("com.mongodb.spark.sql.DefaultSource").load()
```

In [3]:
```
df = df.select('date', 'price', 'v_ask', 'v_bid')
df.show()
```

```
+-------------------+-------+------------------+------------------+
|               date|  price|             v_ask|             v_bid|
+-------------------+-------+------------------+------------------+
|2018-10-07 20:22:23|6555.92| 49.39590000000001|57.832999999999984|
|2018-10-07 20:22:33|6555.92|49.740100000000005| 57.64739999999998|
|2018-10-07 20:22:43|6552.61|49.740100000000005| 57.64739999999998|
|2018-10-07 20:22:53|6552.61|49.740100000000005| 57.64739999999998|
|2018-10-07 20:23:03|6552.61| 49.47540000000001|57.832999999999984|
```

Some Details of implementation(2)

- 2. use pyspark.mllib.clustering import KMeans

```
def find_cluster_centers(timeseries, k, flag='pyspark'):
    """Cluster timeseries in k clusters using k-means and return k cluster centers.

    Args:
        timeseries: A 2-dimensional numpy array generated by generate_timeseries().
        k: An integer representing the number of centers (100).
        flag: indicate which KMeans to use if flag='pyspark' use pyspark_KMeans
    Returns:
        A 2-dimensional numpy array of size k x num_columns(timeseries). Each
        row represents a cluster center.
    """
    if(flag != 'pyspark'):
        #from sklearn.cluster import KMeans
        k_means = sklearn_KMeans(n_clusters=k)
        k_means.fit(timeseries)
        # print('k_means.cluster_centers_', k_means.cluster_centers_)
        return k_means.cluster_centers_
    else:
        # from pyspark.mllib.clustering import KMeans
        # print(flag)
        rdd = sc.parallelize(timeseries)
        model = pyspark_KMeans.train(rdd, k)
        return model.clusterCenters
```

Some Details of implementation(3)

- 3. use rdd to compute the average price change, but pyspark does not support bigfloat.exp, only supports math.exp, this may be a bug

```python
def predict_dpi(x, s, flag='pyspark'):
    """Predict the average price change Δp_i, 1 <= i <= 3.

    Args:
        x: A numpy array of floats representing previous 180, 360, or 720 prices.
        s: A 2-dimensional numpy array generated by choose_effective_centers().
        flag: indicate which method to calculate average price change Δp_i if flag='pyspark',
              use mapreduce method
    Returns:
        A big float representing average price change Δp_i.
    """
    num = 0
    den = 0
    if(flag != 'pyspark'):
        #print('flag',flag)
        for i in range(len(s)):
            y_i = s[i, len(x)]
            x_i = s[i, :len(x)]
            exp = bg.exp(-0.25 * norm(x - x_i) ** 2)
            num += y_i * exp
            den += exp
        return num / den
    else:
        len_x = len(x)
        rdd_s = sc.parallelize(s)
        tmp_s = rdd_s.map(lambda p: (p[len_x],p[:len_x]))
        # maybe a bug pyspark do not support bigfloat.exp
        den_rdd = tmp_s.map(lambda p: (math.exp(-0.25 * norm(x - p[1])**2),p[0]))
        num = den_rdd.map(lambda p: (p[0] * p[1])).reduce(lambda x,y: x+y)
        den = den_rdd.map(lambda p: p[0]).reduce(lambda x,y: x+y)
        # print(num / den)
        return num / den
```

Some Details of Implementation(4)

- 3. use pyspark.mllib.regression import LinearRegressionWithSGD to find parameters w

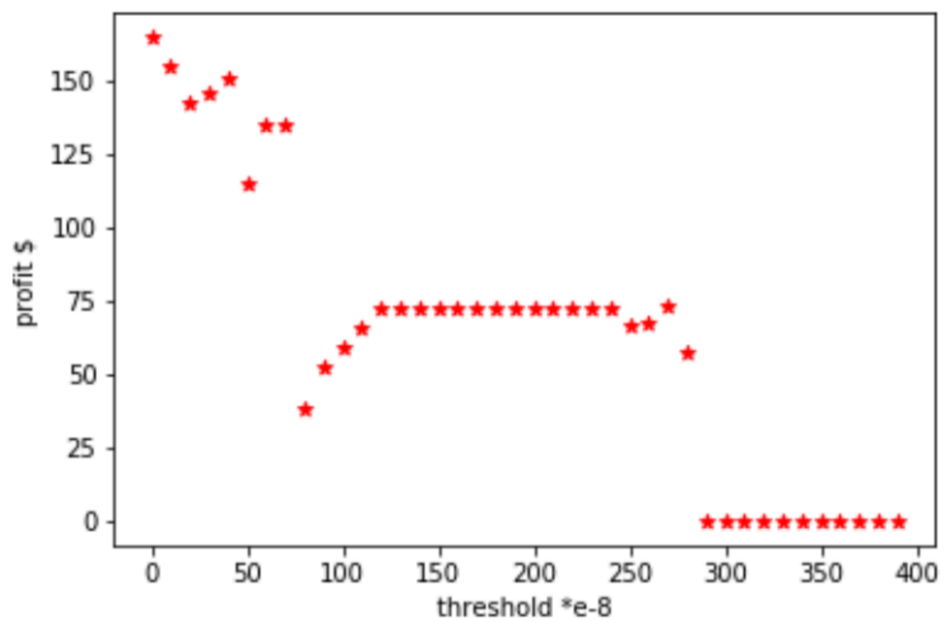
```
def find_parameters_w_spark(Dpi_r, Dp):
    train_pd = pd.DataFrame(Dpi_r)
    train_pd['label'] = Dp
    df_spark = spark.createDataFrame(train_pd)
    #df_spark.show()
    # Define the `input_data`
    parsedData = df_spark.rdd.map(lambda x: LabeledPoint(x[-1], (x[0:-1])))
    lgs = LinearRegressionWithSGD.train(parsedData, iterations=100000, step=0.001)
    w0 = lgs.intercept
    w1, w2, w3, w4 = lgs.weights
    return w0, w1, w2, w3, w4
```

Trading Strategy (from paper)

- The trading strategy is very simple: at each time, we either maintain position of +1 Bitcoin, 0 Bitcoin or −1 Bitcoin (do not consider transaction fees).
- if $\Delta p > t$, a threshold, and current bitcoin position is ≤ 0. then we buy a bitcoin
- if $\Delta p < -t$, and current position is ≥ 0, then we sell a bitcoin
- else do nothing.

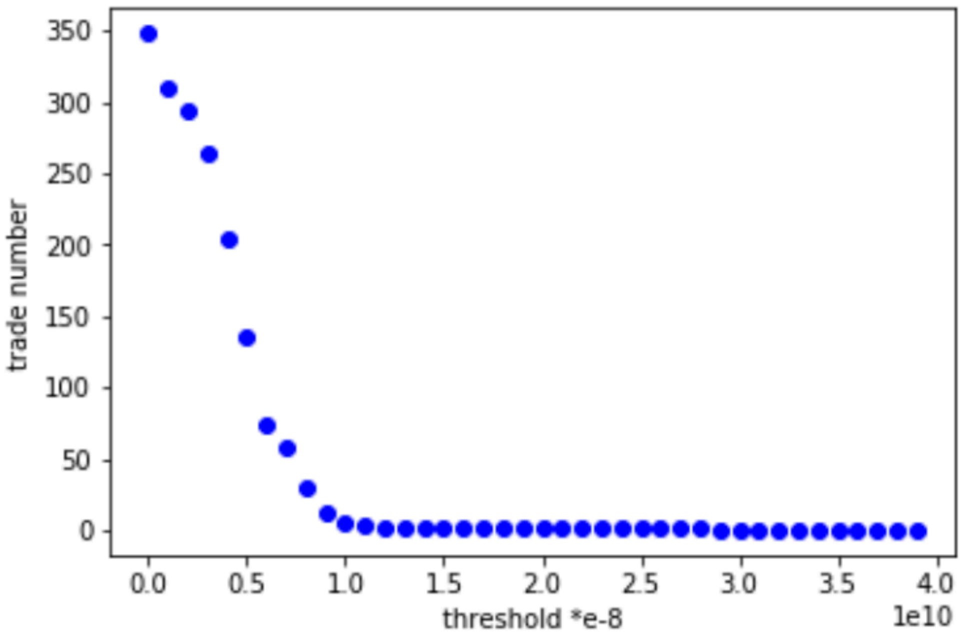
Results(1)

- The effect of different threshold and profit basing on the strategy
- Attention the threshold in the figure times 10^8
- Max profit 164.75 $ when threshold set properly



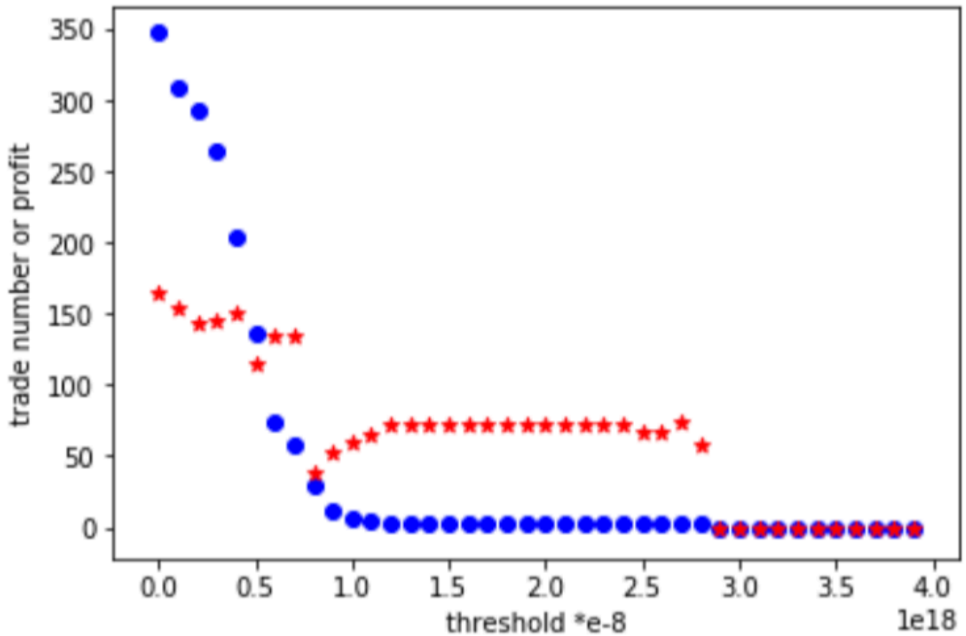
Results(2)

- The effect of different threshold and the number of trades basing on the strategy
- Attention the threshold in the figure times 10^8



Results(3)

- The effect of different threshold , profit and the number of trades basing on the strategy
- Attention the threshold in the figure times 10^8

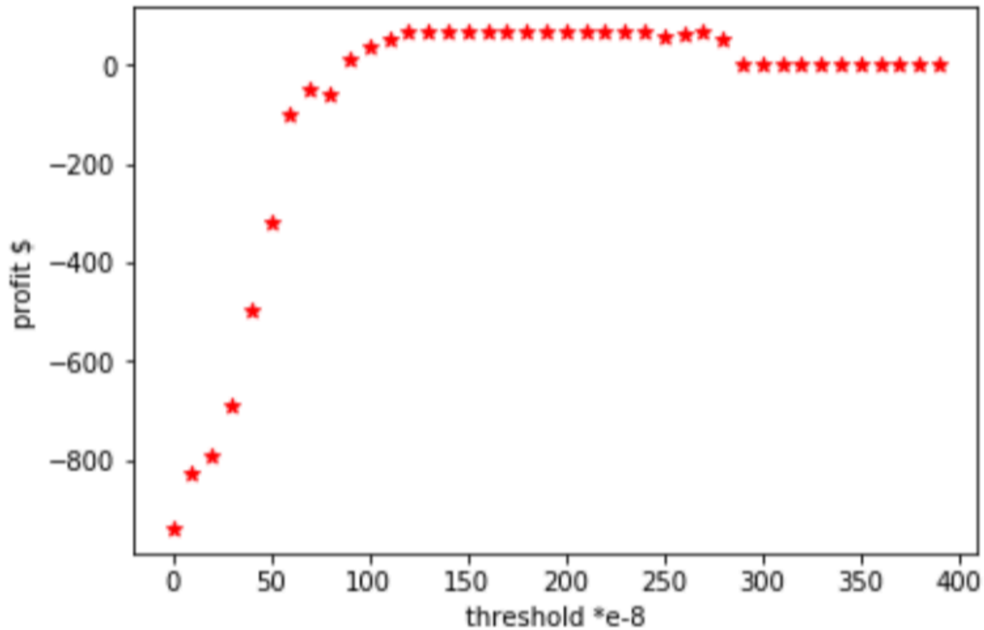


Trading Strategy (modified)

- The trading strategy is very simple: at each time, we either maintain position of +1 Bitcoin, 0 Bitcoin or −1 Bitcoin (consider transaction fees 0.0005).
- if $\Delta p > t$, a threshold, and current bitcoin position is ≤ 0. then we buy a bitcoin, and minus transaction fees
- if $\Delta p < -t$, and current position is ≥ 0, then we sell a bitcoin , and minus transaction fees
- else do nothing.

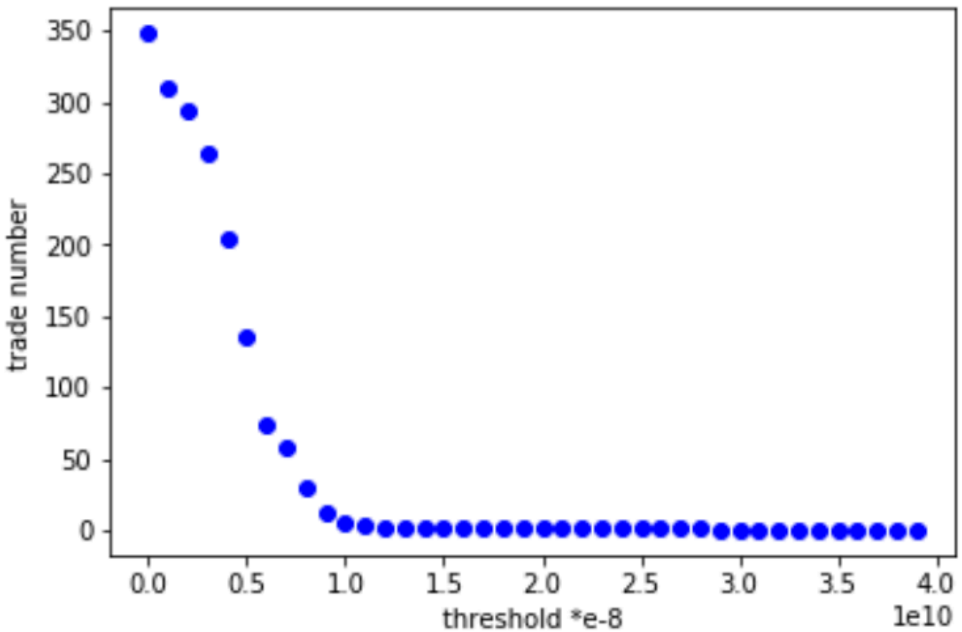
Results(1) considering fees

- The effect of different threshold and profit basing on the strategy
- Attention the threshold in the figure times 10^8
- Max profit 66.8932150000004 \$ when threshold set properly



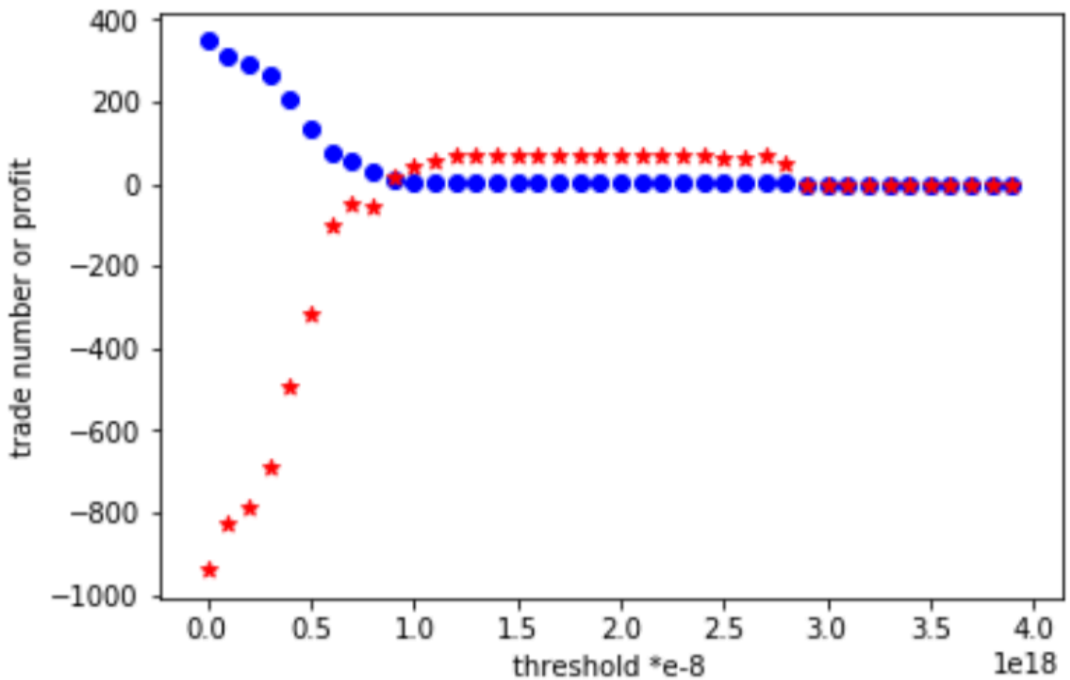
Results(2) considering fees

- The effect of different threshold and the number of trades basing on the strategy
- Attention the threshold in the figure times 10^8



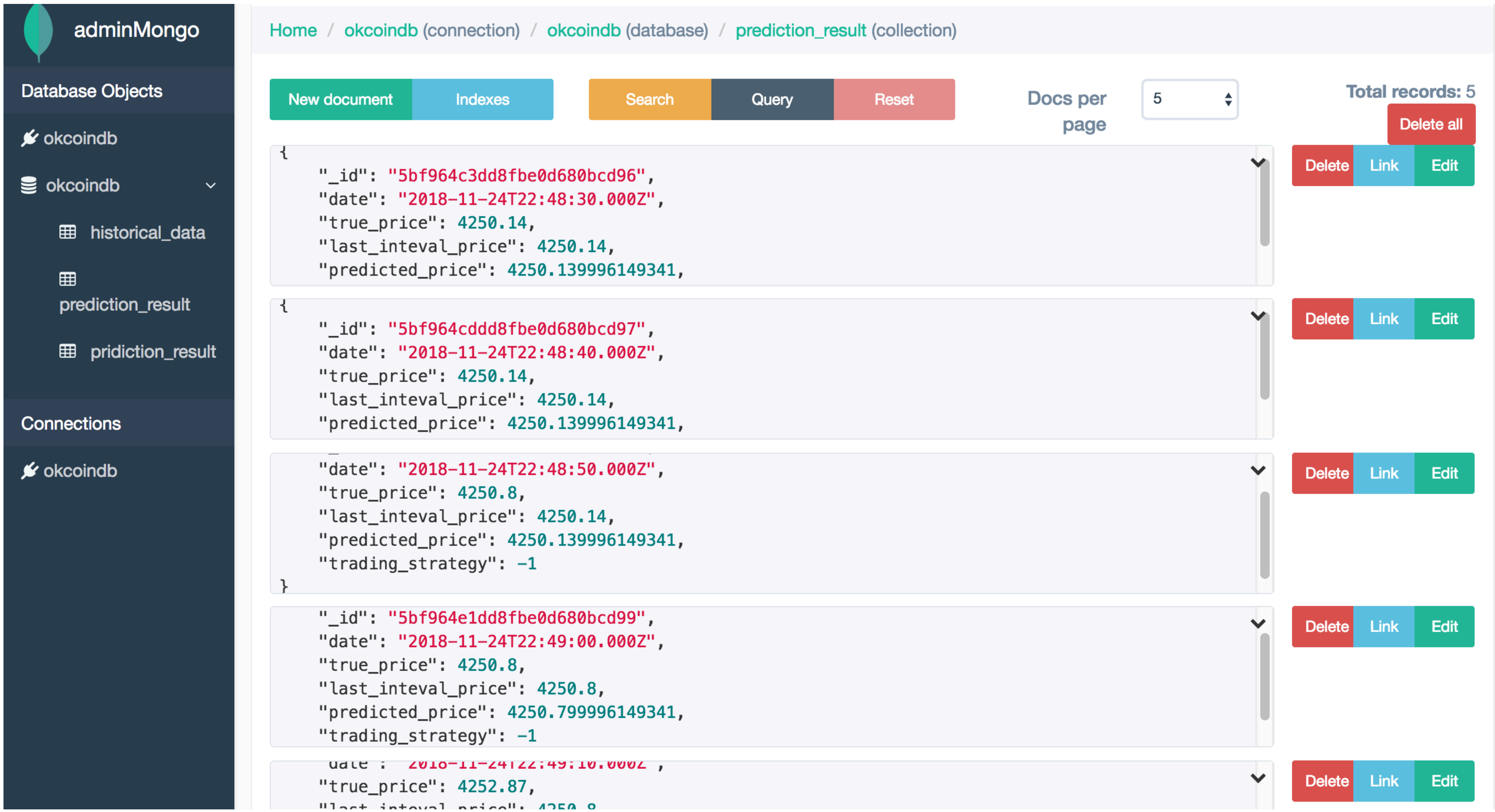
Results(3)

- The effect of different threshold , profit and the number of trades basing on the strategy
- Attention the threshold in the figure times 10^8



Results Saved in MongoDB

- The results of real-time prediction are saved in MongoDB
- Use adminMongo a visualiztion tool of MongoDB can monitor MongoDB



Benefits of using big data technologies(1)

- Azure can help to collect data in a real-time
- Mongo-Spark-Connector allows users to load data from MongoDB into a DataFrame in spark, which providing a faster data loading procedure.
- Spark Sql can help preprocess the training data quickly

Benefits of using big data technologies(2)

- RDD makes the computation faster
- Mllib can reduce the time of training the model
- adminMongo is a good visualization tool For MongoDB
- Overall, Using big data technologies can help to speed up the process of prediction

Future Work

- Apply this method in the prediction of other cryptocurrency' price