Document

Preprocess analysis:

data completion:

- the oral data is provided in .csv file.
- the data contain 5 features: time, Bid price, Bid size, Ask price, Ask size as described in the project description.
- the table looks like this:

Time	BidPrice	BidSize	AskPrice	AskSize
2014-03-20 09:20:00.163	76.2	15200	76.25	24000
2014-03-20 09:30:00.065	NA	NA	76.25	27200
2014-03-20 09:30:00.112	76.2	9600	NA	NA
2014-03-20 09:30:00.138	76.15	400	NA	NA
2014-03-20 09:30:00.299	76.2	400	NA	NA

- Strategy: The first row is completed as the market is open. We run through each row of the data: if the value is "NA" then assign it as the same value as the value in the same column last row:

  $P_{i,j} = P_{i-1,j}$

- structure data: here we add time slot as integer to replace the first column of the data frame. the feature matrix is: $\boldsymbol{X}_i = [t_i, Bidsize, Asksize$; the response is $\boldsymbol{y}_{bid} = Bidprice$ and $\boldsymbol{y}_{ask} = Askprice$

Sample the data into minute unit (option).

according to the first column of the data frame, sample the original data into minute unit.

This can largely cut the computation time.

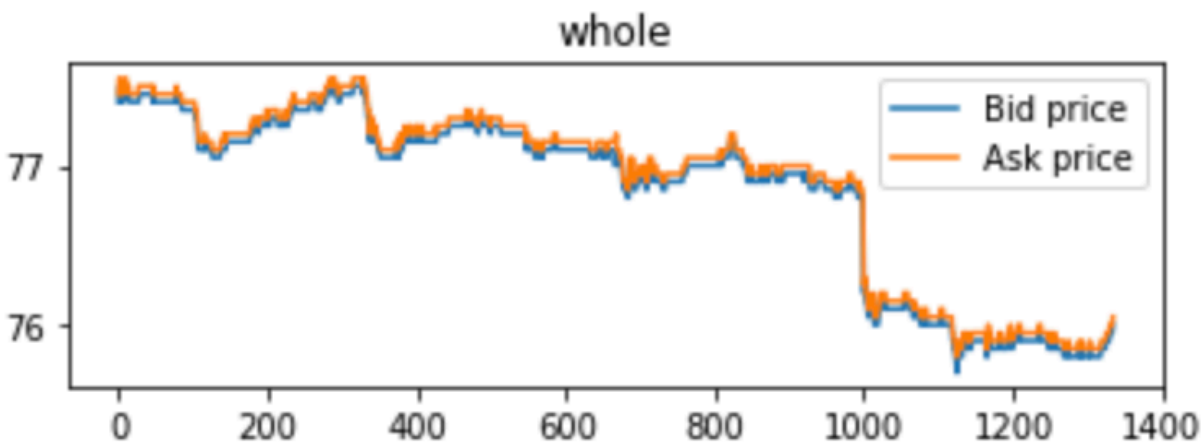
Profit Strategy Analysis:

- `Strategy i: Hold the Stock:`
  - $Bid_j > Ask_i, for\ j > i$. i.e. $Bid_{i+k} - Ask_i > 0, given\ k > 0$

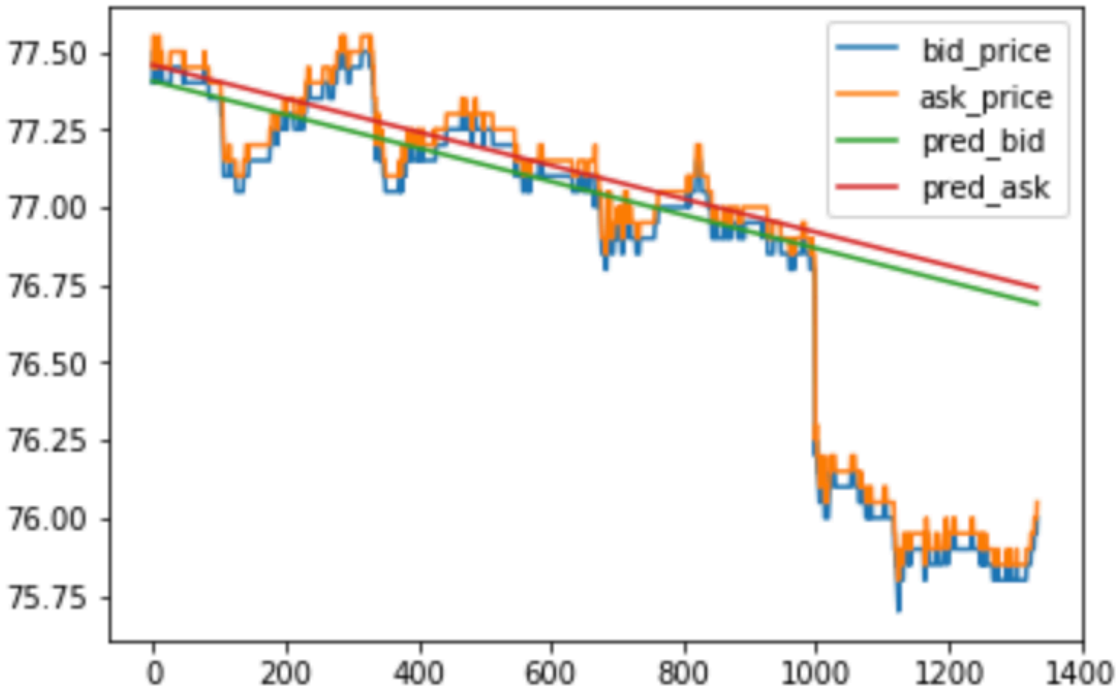
- `Strategy ii: Short Selling:`
  - $Ask_j \leq Bid_i, for\ j \geq i$ e.g. $Ask_{i+k} - Bid_i \leq 0, given\ k > 0$

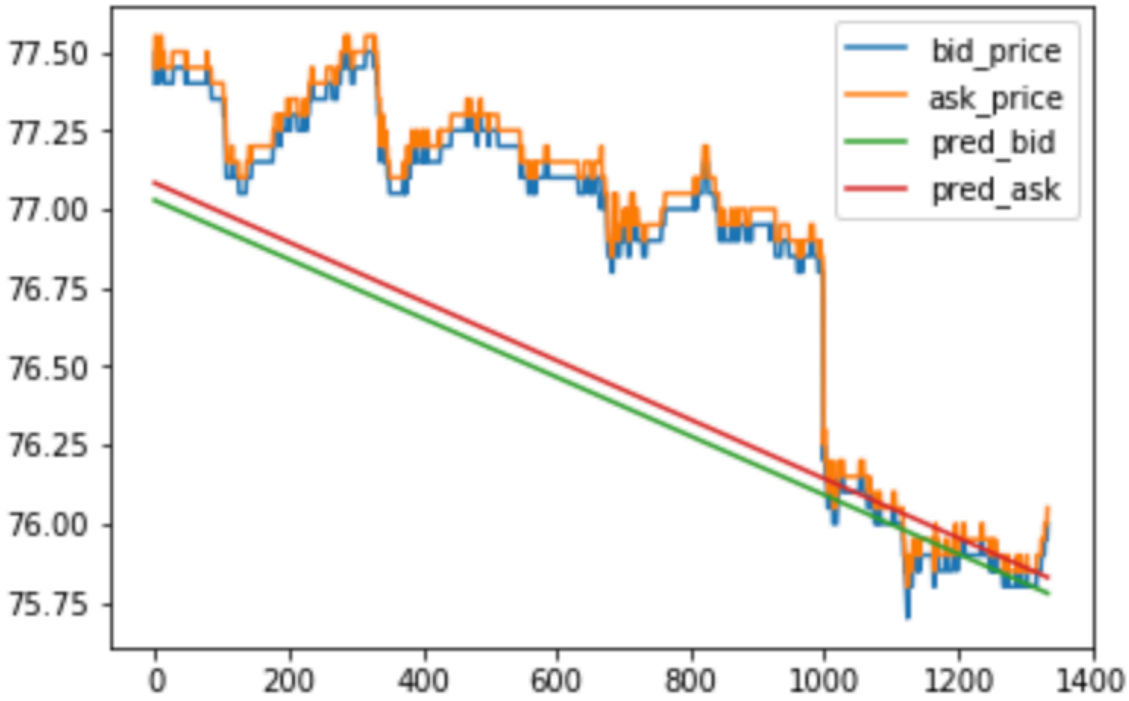
Whole Trend (with conclusion):

- Analysis the overall trend:
- 4 days data plot:



- Simple linear regression on 2 part(divided by the big drap):
- $\boldsymbol{y}_{bid} = \boldsymbol{X} \cdot \beta + \epsilon$
- $\boldsymbol{y}_{ask} = \boldsymbol{X} \cdot \beta + \epsilon$

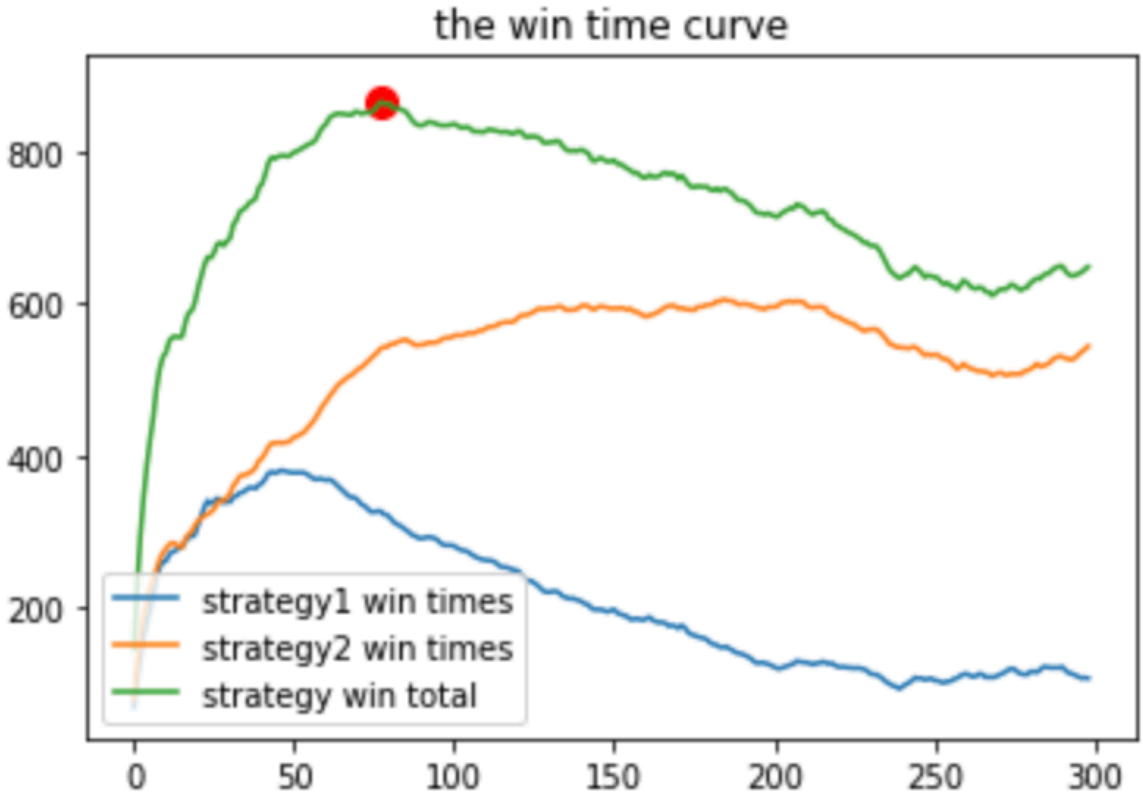


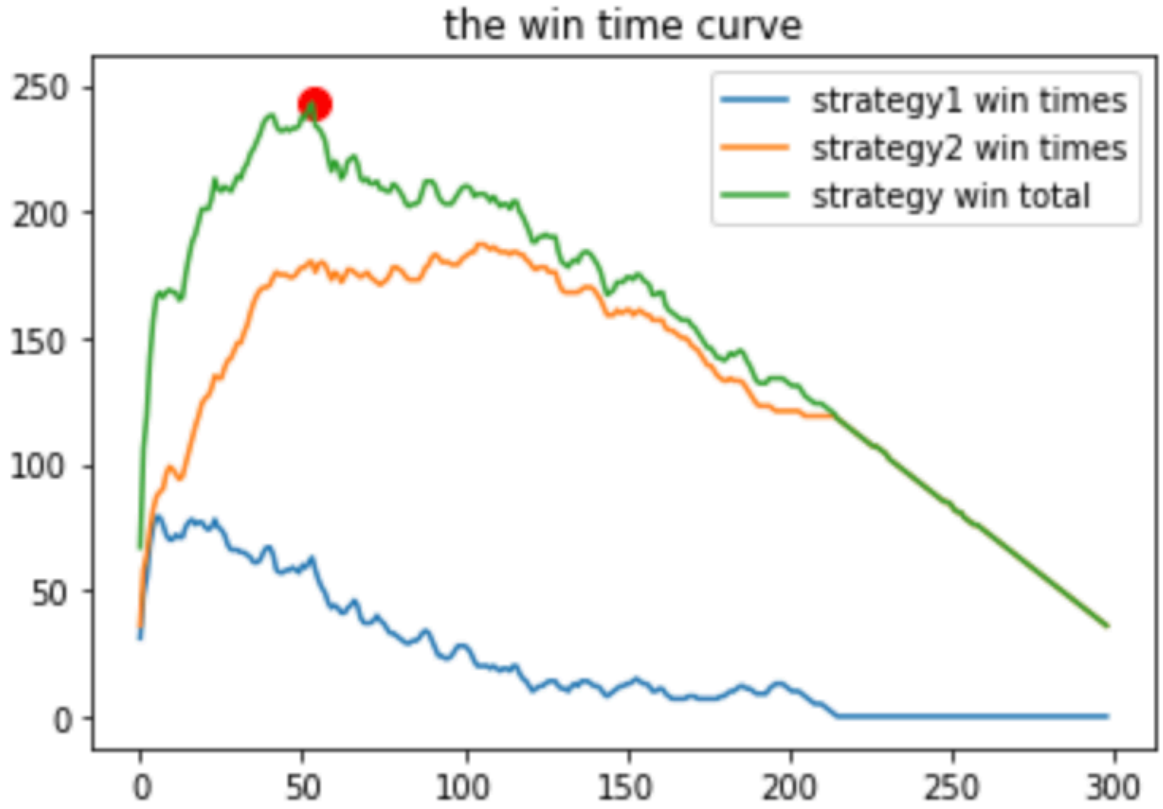


- Conclusion: This overall pridiction suggest that we should use `strategy ii` more often, or we should say use `statrgy i` more cautious.

Simple Trade Simulation:

- Definition: buy in one share with `strategy i` or `strategy ii` at time t_i exercise the share at t_{i+k}.
- Grid search to find the optimal k. For this kind of simulation.
- $arg\max_{k}\ (profit\ times)$
- perform the simulation on training data set(divided by big drop):





Conclusion:

```
The optimal k is: 70.0
the weight for the first strategy(hold the stock) is: 0.346690457155121
the weight for the Second strategy(short selling) is: 0.653309542844879
```

This conclusion can be as a verification of previous section.

System Construction:

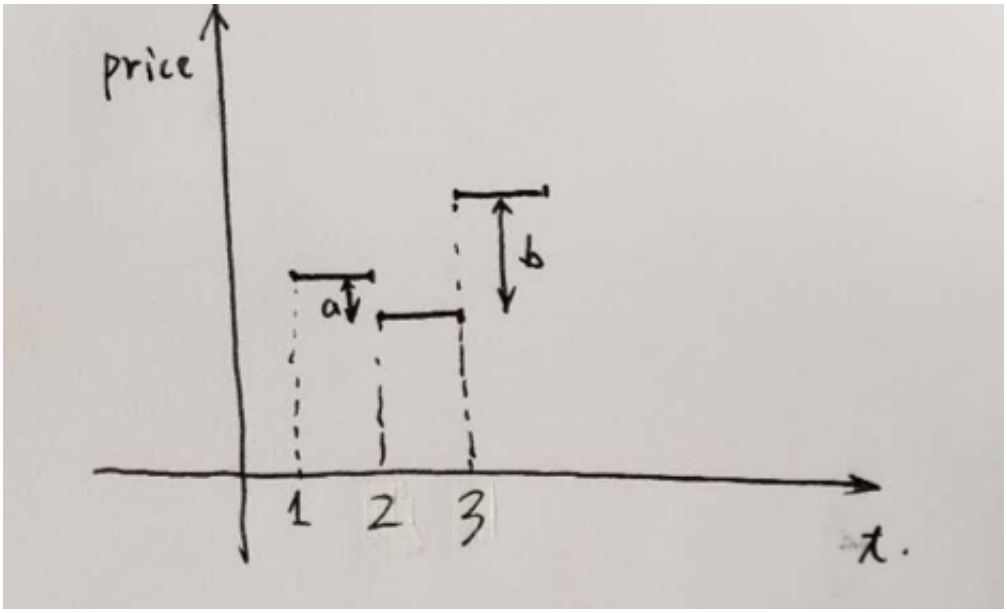
Machanism:

- process though the incoming time slot: For each incoming time point t we do regression once. Updating the regression model for each time point. $Model_t$ for each time t.

Definitions:

- Cut: 0 means keep the original data size, 1 means sample in minute unit.
- prediction window p: at t predict $\hat{\boldsymbol{y}}_{bid}(t)\cdots\hat{\boldsymbol{y}}_{bid}(t+p)$ and $\hat{\boldsymbol{y}}_{ask}(t)\cdots\hat{\boldsymbol{y}}_{ask}(t+p)$
- training window s: each $Model_t$ train on the data points $\boldsymbol{x}_{t-s}\cdots\boldsymbol{x}_t$
- Random seed rng: seed.
- split d : how many grid we investigate. Spread money over time, not buy in at once. Following picture, we can see buy in stock at $t_2$is better than buy in at t_1. for each t we most buy 1 time stock and 1 time short sell. The money total money investigated in the $\frac{Money_{current}}{d}$. As the analysis previously, 30% money goes to `strategy i` 70% money goes

to `strategy ii`. Difine each time of exercise as one piece of stock or short selling.



- Resampling times B: resampling time to the training data for each regression model
- Max inventory level L: how many piece of stock or short selling we could hold.
- Max hold time H: how long we can hold each piece of stock or short selling
- Initial money M: 100
- Cautious coeffience ρ: How much cautious we want when buy in stock.

When to buy in the stock or process short selling:

- use training data fit the regression model.
- At t, make a prediction ahead of size p. We get $\hat{\boldsymbol{y}}_{bid}(t) \cdots \hat{\boldsymbol{y}}_{bid}(t+p)$ and $\hat{\boldsymbol{y}}_{ask}(t) \cdots \hat{\boldsymbol{y}}_{ask}(t+p)$
- Find $\hat{\boldsymbol{y}}_{bid\ max}$ and $\hat{\boldsymbol{y}}_{ask\ min}$
- if $\rho \cdot \hat{\boldsymbol{y}}_{bid\ max} > \boldsymbol{y}_{ask}(t)$: then buy in one piece of stock. In this case, we expect the `strategy i` could make profit
- if $\hat{\boldsymbol{y}}_{ask\ min} < \boldsymbol{y}_{bid}(t)$: then exercise one piece of short selling. we expect the `strategy ii` could make profit

When to exercise the hold:

Review:

- Profit Condition: `strategy i` or `strategy ii`
- `Strategy i: Hold the Stock:`
  - $Bid_j > Ask_i, for\ j > i$. i.e. $Bid_{i+k} - Ask_i > 0, given\ k > 0$
- `Strategy ii: Short Selling:`
  - $Ask_j \leq Bid_i, for\ j \geq i$ e.g. $Ask_{i+k} - Bid_i \leq 0, given\ k > 0$

- `Plan i` : At each time t, go though all the holden stocks and short selling. Once the profit condition meet exercise the holden piece.
- `Plan ii` :

  If $[\hat{\boldsymbol{y}}_{bid}, y_{bid}(t)]_{max} = y_{bid}(t)$: that means $\hat{\boldsymbol{y}}_{bid}$ will not go up an more, we get the highest $\hat{\boldsymbol{y}}_{bid}$, go through the stock and exercise ones meets the profit condition.

  If $[\hat{\boldsymbol{y}}_{ask}, y_{ask}(t)]_{min} = y_{ask}(t)$: that means $\hat{\boldsymbol{y}}_{ask}$ will not go down anymore, we get the lowest $\hat{\boldsymbol{y}}_{ask}$ we can get, go through the short selling and exercise ones meets the profit condition.
- In practice `Plan i` and `Plan ii` no big difference.

Safe Gaurd:

- since we will use short selling, in the process we may have lots of money in the packet. To prevent invest too much into the market. We use a saft gaurd switch. Set an upper bound to how much we can invest once. To prevent a snowball effect.

Regression Part (Machine Learning Part):

Bagging & AdaBoostRegression framework(Main part):

- Use B Bootstrap resampling
- AdaBoost with Regreesion Tree as the fundamental model
- Forward Stagewise Model

Algorithm 1: AdaBoost & Bagging based Regression

Input: $\boldsymbol{X}_{tr}$, $\boldsymbol{y}_{tr}$, $\boldsymbol{X}_{te}$, B

Output: $\hat{\boldsymbol{y}}$

for *b from 1 to B* **do**

  $\boldsymbol{X}_b, \boldsymbol{y}_{tr} = BoostrapSample(\boldsymbol{X}_{tr}, \boldsymbol{y}_{tr})$

  $AdaBoost_b(RegressionTree(MaxDepth = 2), SubModelNumber = 300).fit(\boldsymbol{X}_b, \boldsymbol{y}_b)$

  $\hat{\boldsymbol{y}}_b = AdaBoost_b.pred(\boldsymbol{X}_{te})$

end

$\hat{\boldsymbol{y}} = \frac{1}{B} \cdot \sum_1^B \hat{\boldsymbol{y}}_b$

Tried and waived out Regressor:

- Weighted least square: polynomial with each data assign a count down weight γ^l where l is the distance from the current data point, means a diminishing weight of the data point, intruduce power p to cross-validation, not good in prectice.
- Gaussian Process(kernel selection): was waived out due to comlexity of the kernel selection and sophisticated package dependency.

Performance:

- simulation 1 setup (Low risk):
    - training: Day 1,2,3. Testing: Day4.
    - Cut = 0
    - Safe Gaurd = 1; split d = 10
    - training window s = 200
    - prediction window p = 20
    - Random seed rng = 0
    - Resampling times B = 2
    - Cautious coeffience ρ = 0.999
    - Max inventory level L = 50
    - Max hold time H = inf
    - Initial money M = 100
    - Money at the end of the day:

```
At the end of the day: 101.76384960595328
```

- simulation 2 setup (High Risk):
    - training: Day 1,2,3. Testing: Day4.
    - Cut = 0
    - Safe Gaurd = 0; split d = 5
    - training window s = 200
    - prediction window p = 20
    - Random seed rng = 0
    - Resampling times B = 2
    - Cautious coeffience ρ = 0.999
    - Max inventory level L = 50
    - Max hold time H = inf
    - Initial money M = 100
    - Money at the end of the day:

```
At the end of the day: 176.62273258767527
```

In the End:

This project somehow put a difficulty in the Regression part.

The result may needs some sort of tuning the hyper-parameters I introduced in the model.