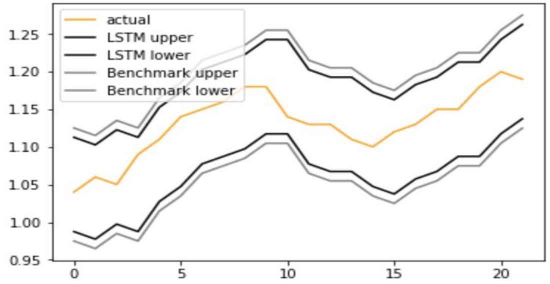
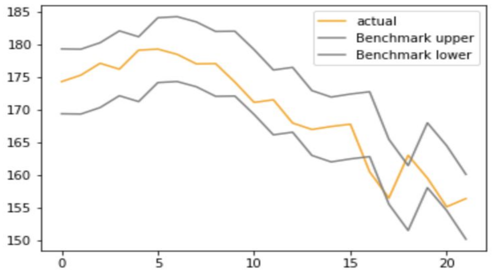
# Long Short Term Memory

Yufei Zhang



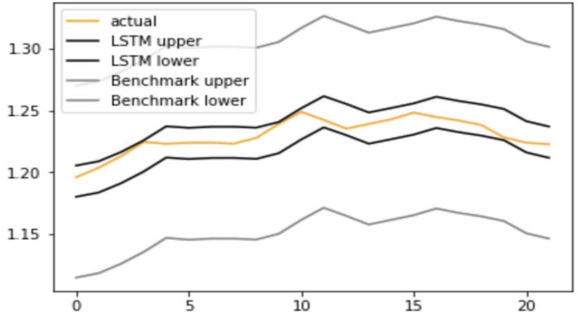

MAPE:1.45% RMSE: 0.020

MAPE:1.387% RMSE: 0.019



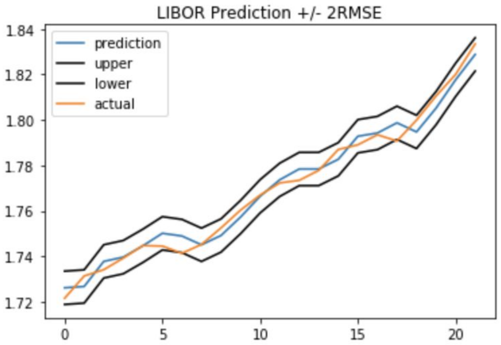

MAPE:1.315% RMSE: 2.961

MAPE:1.321% RMSE: 2.969




MAPE:0.419% RMSE: 0.006

MAPE: 0.406% RMSE: 0.006




MAPE:0.321% RMSE: 0.007

MAPE: 0.19% RMSE: 0.004

# Long Short Term Memory

Yufei Zhang

1. Load the dataset from CSV file.
2. Transform the dataset to make it suitable for the LSTM model, including:
   2.1. Transforming the data to be **stationary** : differencing
   2.2. Transforming the data to a supervised learning problem : **shifting**
   2.3. Transforming the data so that it has the **scale** -1 to 1: MinMaxScaler(min=-1, max=1)
   2.4. **Spliting** data to training set and testing set

X_std = (X - X.min) / (X.max - X.min)
X_scaled = X_std * (max - min) + min

| No. | Value |
|---|---|
| 0 | 10 |
| 1 | 16 |
| 2 | 20 |
| 3 | 15 |
| .. | ... |

Original Data

Values = [10,16,20,15...]

| No. | Diff |
|---|---|
| 0 | NA |
| 1 | 6 |
| 2 | 4 |
| 3 | -5 |
| .. | ... |

Step 2.1 Stationary

Diff = [6,4,-5,...]

| No. | X | Y |
|---|---|---|
| 0 | NA | 6 |
| 1 | 6 | 4 |
| 2 | 4 | -5 |
| 3 | -5 | -6 |
| ... | ... | ... |

Step 2.2 Shifting

X = [0,6,4,-5,...]
Y = [6,4,-5,...,0]

| No. | X_Scaled | Y-Scaled |
|---|---|---|
| 0 | 0 | 0.8 |
| 1 | 0.8 | 0.7 |
| 2 | 0.7 | -0.75 |
| 3 | -0.75 | -0.78 |
| ... | ... | ... |

Step 2.3 Scaling

X_Scaled = [0,0.8,0.7,-0.75,...]
Y_Scaled = [0.8,0.7,-0.75,...0]

| No. | |
|---|---|
| 0-1237 | Training Set |
| 1238-1259 | Testing Set |

Step 2.4 Spliting

X_Scaled_Train = [0,0.8,0.7,-0.75,...] (1238)
Y_Scaled_Train = [0.8,0.7,-0.75,...] (1238)
X_Scaled_Test = [...] (length = 22)
Y_Scaled_Test = [...0] (length = 22)

# Long Short Term Memory

Yufei Zhang

3. Fitting a stateful LSTM network model to the training data : **training**
4. Recoding training **RMSE**
5. Evaluating the static LSTM model on the test data : **testing;**
6. Applying **2*RMSE (95% confidence)** to prediction as upper and lower bounds.
7. Reverting scaling
8. Reverting differencing

| No. | X_Training _Scaled | Prediction in last Epoch |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0.8 | 0.4 |
| 2 | 0.7 | 0.6 |
| 3 | -0.75 | 0.1 |
| ... | ... | ... |
| 1237 | 0.5 | 0.4 |

Step 4 Training rmse

RMSE=rmse(X_Training_Scaled, Prediction in last Epoch)
*Suppose RMSE = 0.05*

| No. | X_Test | Prediction by Trained Model |
|---|---|---|
| 1238 | 0.8 | 0.9 |
| 1239 | 0.6 | 0.4 |
| 1240 | 0.5 | 0.3 |
| 1241 | -0.75 | 0.1 |
| ... | ... | ... |
| 1259 | 0.4 | 0.5 |

Step 5 Testing

test rmse = rmse (X_Test, prediction by trained model)

| No. | X_Test | Prediction by Trained Model | Lower Bound | Upper Bound |
|---|---|---|---|---|
| 1238 | 0.8 | 0.9 | 0.7 | 1.0 |
| 1239 | 0.6 | 0.4 | 0.3 | 0.5 |
| 1240 | 0.5 | 0.3 | 0.2 | 0.4 |
| 1241 | -0.75 | 0.1 | 0.0 | 0.2 |
| ... | ... | ... | | |
| 1259 | 0.4 | 0.5 | 0.4 | 0.6 |

Step 6 Applying Bounds

Lower bound = Prediction - 2*RMSE
Upper bound = Prediction + 2*RMSE
“If the data are roughly normal, then most of the residuals lie within about ± 2 RMSE of their mean (at zero)”

# Benchmark -- Persistence Model

Yufei Zhang

1. Load the dataset from CSV file.
2. Spliting data as LSTM
3. Make prediction : using observation of prior time stamp
4. Recording training RMSE
5. (For Random Walk Data only) Applying training rmse to testing data to get bounds.

“A good baseline forecast for a time series with a linear increasing trend is a persistence forecast.” (1)

| No. | Value |
|---|---|
| 0 | 10 |
| 1 | 16 |
| 2 | 20 |
| 3 | 15 |
| .. | ... |

Step 1 Original Data

Values = [10,16,20,15...]

| No. | |
|---|---|
| 0-1237 | Training Set |
| 1238-1259 | Testing Set |

Step 2 Spliting

X_Train = [...] (1238)
X_Test = [...] (length = 22)

| No. | X_Train | Benchmark prediction |
|---|---|---|
| 0 | 10 | 0 |
| 1 | 16 | 10 |
| 2 | 20 | 16 |
| ... | ... | ... |
| **1237** | 22 | 14 |

Step 3,4 Predicting

benchmark rmse = rmse (X_Train, Benchmark prediction)

| No. | X_Test | Benchmark prediction | Lower Bound | Upper Bound |
|---|---|---|---|---|
| **1238** | 15 | 18 | 17 | 19 |
| 1239 | 16 | 15 | 14 | 16 |
| ... | ... | ... | | |
| **1259** | 13 | 0 | -1 | 1 |

Step 5 Bounds

RMSE is getting from Step 4. Suppose rmse=0.5.
Lower bound = Prediction - 2*RMSE
Upper bound = Prediction + 2*RMSE

(1) Jason Brownlee, Time Series Forecasting with the Long Short-Term Memory Network in Python

# Random Walk Test usig MatLab econometrics toolbox

Yufei Zhang

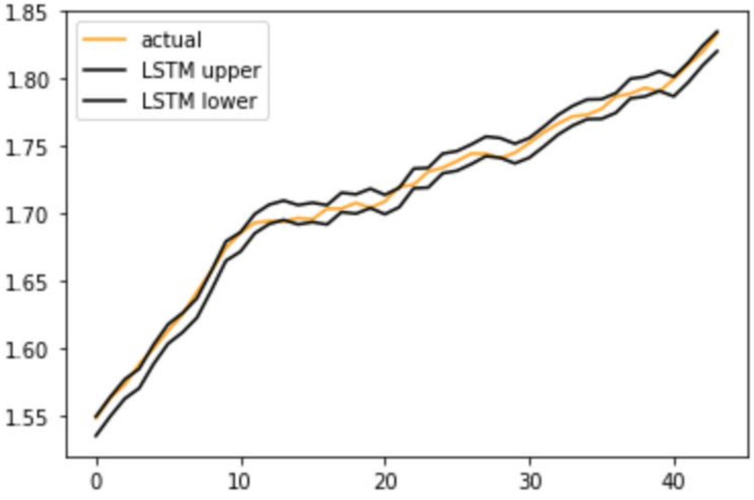
```
[h,pValue,stat,cValue,ratio] = vratiotest(data)
```

| | LIBOR | APPL | SWAP | DEXUSEU |
|---|---|---|---|---|
| h (logical) | 1 | 0 | 0 | 0 |
| Random Walk? | N | Y | Y | Y |

h=0 : vratiotest does not reject the hypothesis that a random walk is a reasonable model for the stock series.
h=1 : vratiotest reject the hypothesis that a random walk is a reasonable model for the stock series.

# Prediction on More days

Yufei Zhang




Data: LIBOR

Predict window : 44 days

Breaches : 3

Bounds: 2* training rmse (22-43 days use a different rmse from 0-21 days)

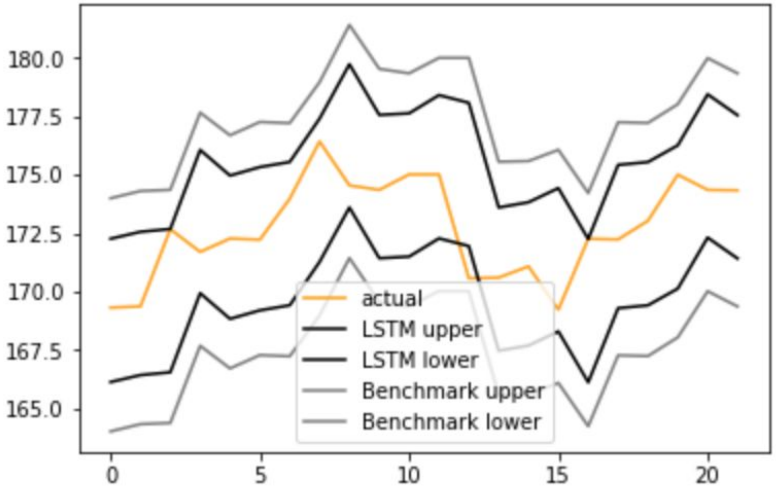
Training set: 22-43 days have a larger(22 days more)  training set.

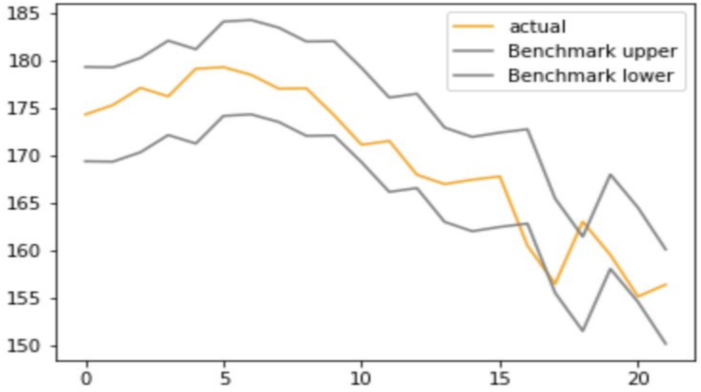
Still debugging the program that can automatcally backtesting. Once

completed, more days prediction can be provided.

# Prediction on More days APPL

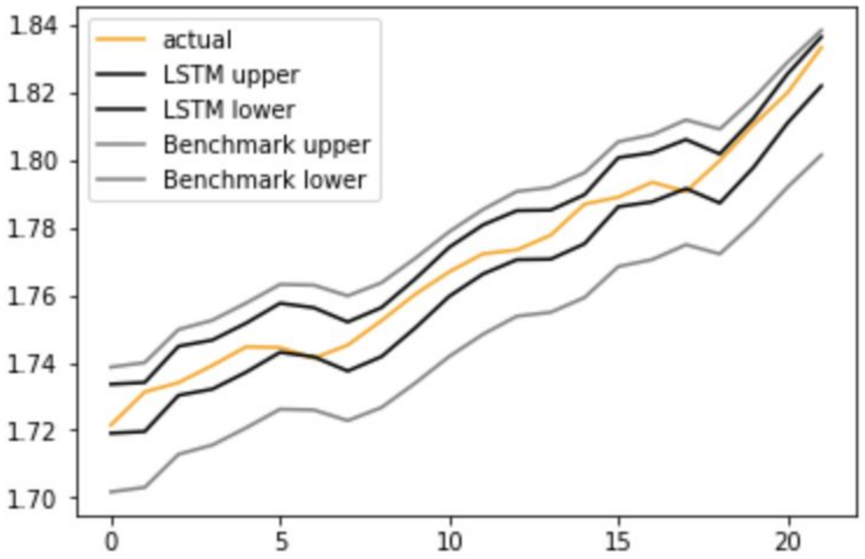
Yufei Zhang



# LIBOR

Yufei Zhang




| Accuracy | MAPE% | RMSE | Breach |
|---|---|---|---|
| Benchmark | 0.321 | 0.00667 | 0 |
| Before Stationary | 0.28 | 0.0060 | NA |
| After Stationary | 0.1927 | 0.00412 | 2 |

Benchmark model got less breaches, however, it's using a wider bounds.

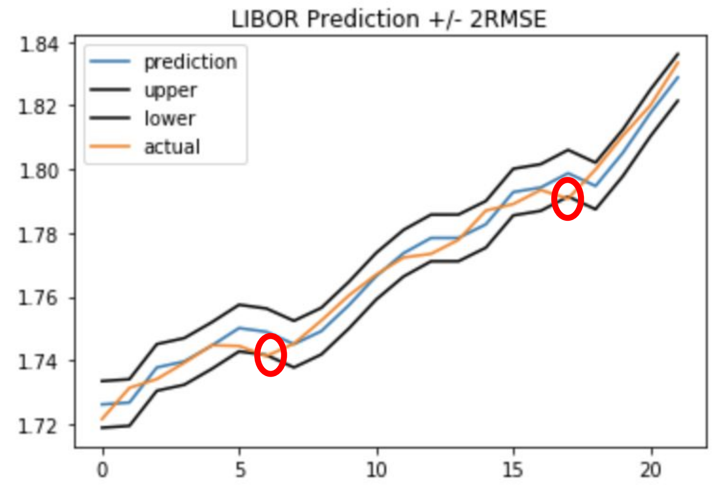
I suspect that benchmark model gives too wide bound to detect breaches. Still need real data with actual breaches to figure out which model is better.

RMSE: rmse got from model training process
Upper bounds: prediction + 2 * RMSE
Lower bounds: prediction - 2 * RMSE

# LIBOR Hyperparameters Tuning (Batch_Size = 1)

Yufei Zhang

| No | Epochs | Neurons | MAPE | RMSE | Breaches |
|---|---|---|---|---|---|
| 1 | 15 | 40 | 0.20450 | 0.00424 | 0 |
| 2 | 20 | 40 | 0.20349 | 0.00424 | 0 |
| 3 | 30 | 40 | 0.19677 | 0.00415 | 0 |
| 4 | 35 | 40 | 0.20030 | 0.00425 | 1 |
| 5 | 32 | 40 | 0.19435 | 0.00420 | 1 |
| 6 | 30 | 30 | 0.19888 | 0.00419 | 0 |
| 7 | 30 | 50 | 0.21470 | 0.00438 | 1 |
| 8 | 30 | 45 | 0.19271 | 0.04119 | 2 |




| Lower Bound | Actual | Upper Bound |
|---|---|---|
| 1.74158818901279 | 1.7413 | 1.75622873729980 |
| 1.791383221522132 | 1.7907 | 11.80602376980914 |

*Blue Numbers are best are currently best accuracy.
Orange numbers are chosen hyperparameters.

# APPL

Yufei Zhang

## Why prediction looks like a left move

- Random Walk Hypothesis

*"A random walk is one in which future steps or directions cannot be predicted on the basis of past history. When the term is applied to the stock market, it means that short-run changes in stock prices are unpredictable."*

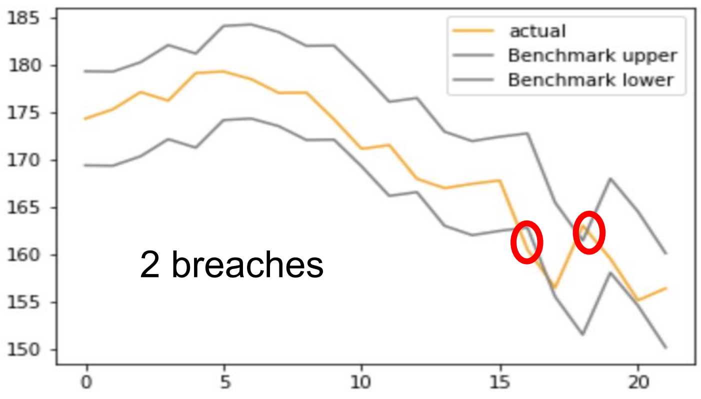
— Page 26, A Random Walk down Wall Street: The Time-tested Strategy for Successful Investing

| Accuracy | MAPE% | RMSE | Breach |
|---|---|---|---|
| Benchmark | 1.135 | 2.961 | 2 |
| Before Stationary | 1.28 | 2.99 | NA |
| After Stationary | 1.321 | 2.969 | 7 |

- Persistance Model provides the best source of reliable predictions.

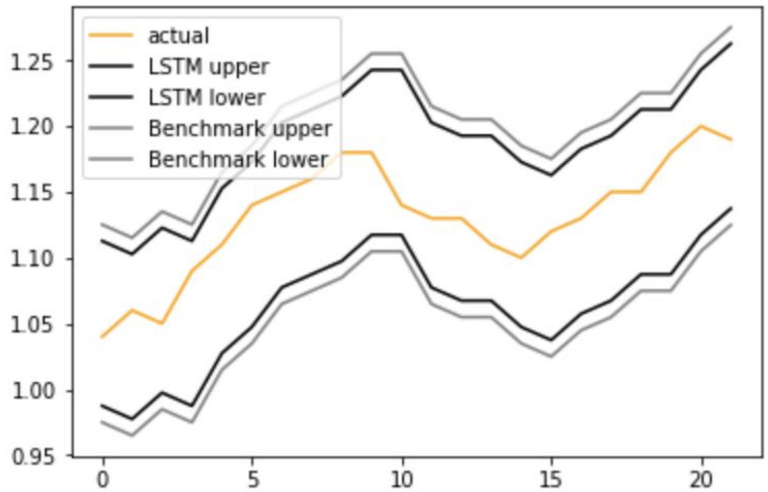
" This is key for time series forecasting. Baseline forecasts with the persistence model quickly flesh out whether you can do significantly better. If you can't, you're probably working with a random walk."

-- Jason Brownlee A Gentle Introduction to the Random Walk



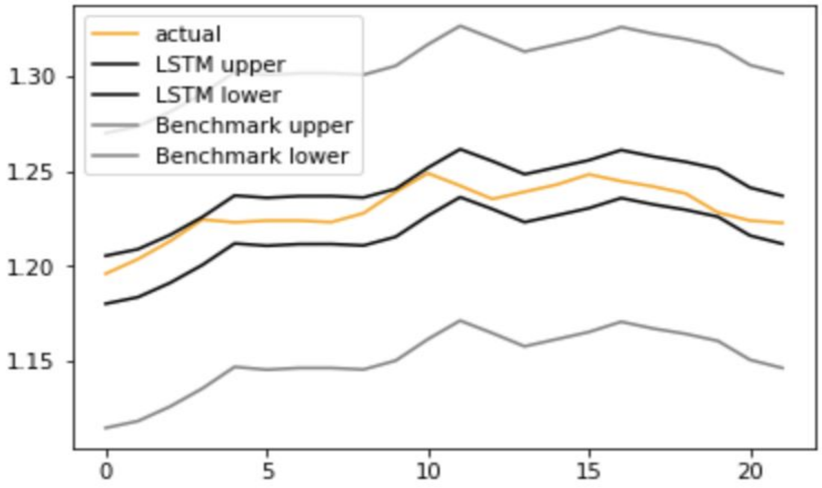
Yufei Zhang

## SWAP 30 epochs, 35 neurons




| Accuracy | MAPE % | RMSE | Breach |
|---|---|---|---|
| Benchmark | 1.45 | 0.0197714 | 0 |
| Before Stationary | 1.42 | 0.018 | NA |
| After Stationary | 1.449 | 0.01975 | 0 |

## DEXUSEU 60 epochs, 45 neurons




| MAPE% | RMSE | Breach |
|---|---|---|
| 0.419 | 0.00620484 | 0 |
| 0.57 | 0.0087 | NA |
| 0.419 | 0.00617969 | 0 |

# Summary

Yufei Zhang

- LIBOR is predictable. LSTM provides a significat improve on predicting over persistence model.
- SWAP, DEXUSEU and APPL are random walk, indicating they're **not predictable**.
- LSTM is still usful on random walk data by providing a **tight bound**.
    - bound is determined by training rmse
    - After rounds of training, LSTM can simulate **training** data better than persistence model
    - LSTM outputs a **smaller rmse** than persistence model

| Market Data | Random Walk? | Best performance model | Breaches detected by best performance model in 22 days |
|---|---|---|---|
| LIBOR | N | LSTM | 2 |
| APPL (Stock) | Y | Persistence Model | 2 |
| SWAP | Y | LSTM / Persistence Model | 0 |
| DEXUSEU (Exchange rate) | Y | LSTM / Persistence Model | 0 |

# Prediction looks like a left move

Yufei Zhang

- Random Walk Hypothesis

“*A random walk is one in which future steps or directions cannot be predicted on the basis of past history. When the term is applied to the stock market, it means that short-run changes in stock prices are unpredictable.*”
— Page 26, A Random Walk down Wall Street: The Time-tested Strategy for Successful Investing

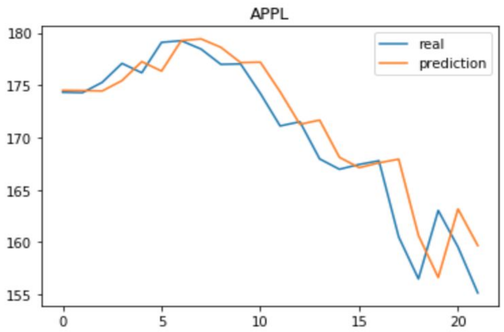
- Persistance Model provides the best source of reliable predictions.

“ This is key for time series forecasting. Baseline forecasts with the persistence model quickly flesh out whether you can do significantly better. If you can’t, you’re probably working with a random walk.”

-- Jason Brownlee https://machinelearningmastery.com/gentle-introduction-random-walk-times-series-forecasting-python/

# APPL Stock

Yufei Zhang




First Four days:

```
predicted:[[174.06352]], actual:[[174.33]]
predicted:[[174.0441]], actual:[[174.29]]
predicted:[[174.00528]], actual:[[175.28]]
predicted:[[174.96577]], actual:[[177.09]]
```

## Tuning the Epoch

One Epoch: Whole data set go through neural network once
Larger the epoch size, longer the calculation time

| Epochs | MAPE | RMSE |
|---|---|---|
| 100 | 1.15510211 | 2.7777774914911446 |
| 300 | 1.15351336 | 2.7631627514012953 |
| 500 | 1.21856932 | 2.9026333327139775 |
| 1000 | 1.19991636 | 2.685796528535209 |

*Instead of using 1000 epochs which takes a long time to run, I choose 300 epochs finally.

## Tuning the Training Batch size *Under epoch size=300*

Batch: subset of sample. Iteration in each epoch = sample size/ batch size
Smaller the batch size, longer the calculation time

| Batchs | MAPE | RMSE |
|---|---|---|
| 10 | 1.32042804 | 3.0935492293336155 |
| 100 | 1.16456492 | 2.7869327110175006 |
| 500 | 1.55472734 | 3.3439883563233956 |
| 1000 | 1.50658704 | 3.2703686091609 |

## Tuning the Num of Neurals *epoch=300, batch size=100*

The number of neurons affects the learning capacity of the network.
More neurons, longer the training time.
More learning capacity also creates the problem of potentially overfitting the training data.

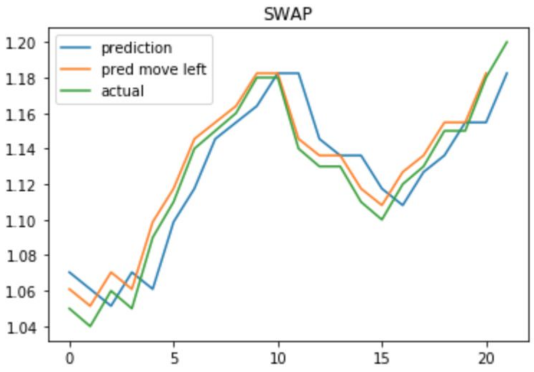
| Neurons | MAPE | RMSE |
|---|---|---|
| 128 | 1.16724945 | 2.7926793040158913 |
| 256 | 1.16456492 | 2.7869327110175006 |
| 512 | 1.55472734 | 3.3439883563233956 |
| 1000 | 1.50658704 | 3.2703686091609 |

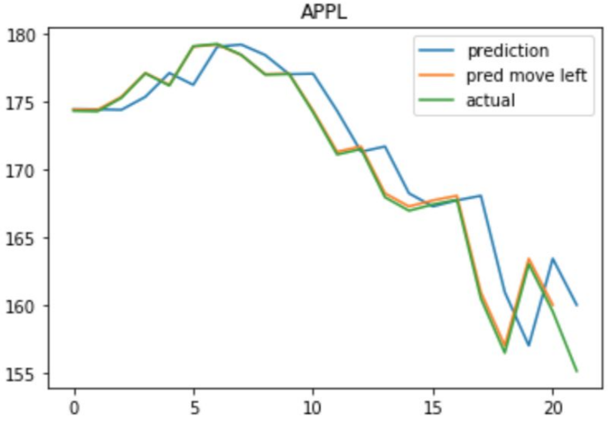
# Two concerns of should we continue using LSTM

1. It seems LSTM just use yesterday's value as the prediction for today ???
   a. Depends on nature of data. For some data set, it is true. For some, it isn't.
   b. Common across other one-day prediction algorithm
2. Impossible to get prediction interval

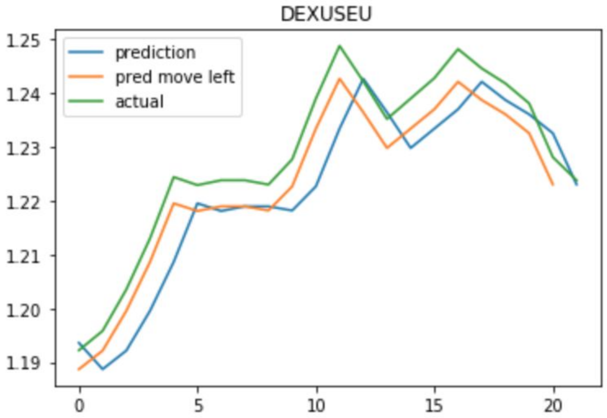
# Prediction=actual move right one day?

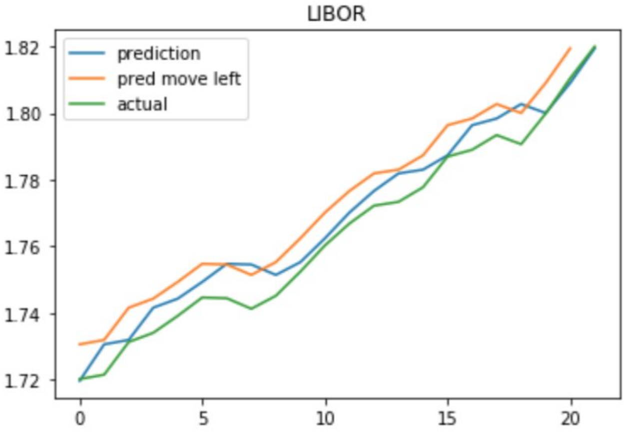
Yufei Zhang




seems using yesterday's value



# Two concerns of should we continue using LSTM

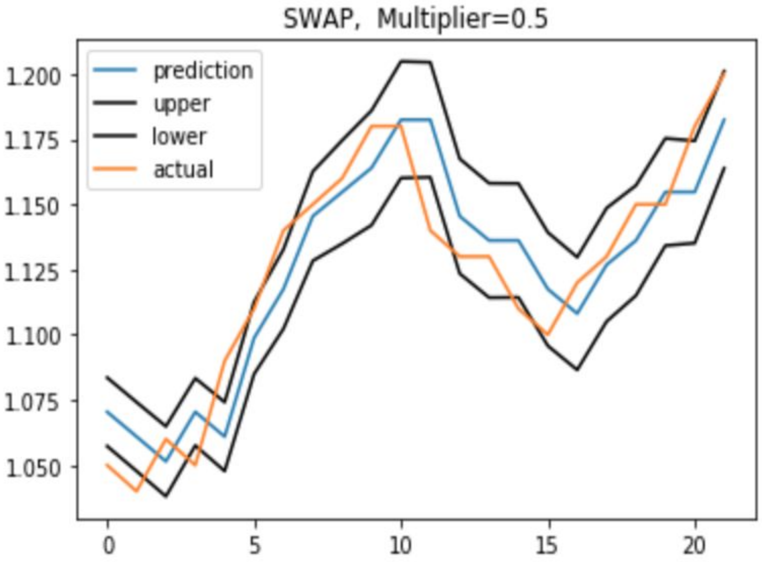
1. It seems LSTM just use yesterday’s value as the prediction for today ???
2. Impossible to get prediction interval
    a. But we can always +/- a std of past N days to prediction value to get bound like Bollinger Bands:
        i. upper bound= LSTM prediction + (std of past 20 days * multiplier)
        ii. lower bound= LSTM prediction - (std of past 20 days * multiplier)

| | Muptilpier=0.5 | Muptilpier=1 | Muptilpier=2 (BB) |
|---|---|---|---|
| SWAP | 8 | 1 | 0 |
| APPL | 10 | 4 | 0 |
| US-EU | 10 | 4 | 0 |
| LIBOR | 1 | 0 | 0 |

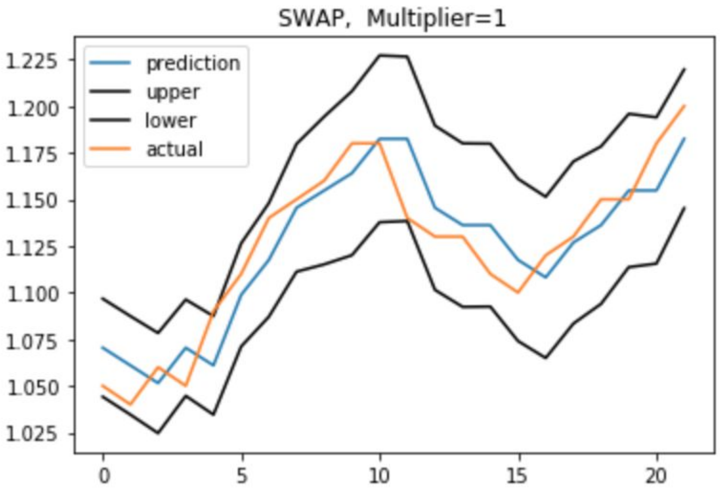
Number of breaches using different multiplies on different dataset

# SWAP

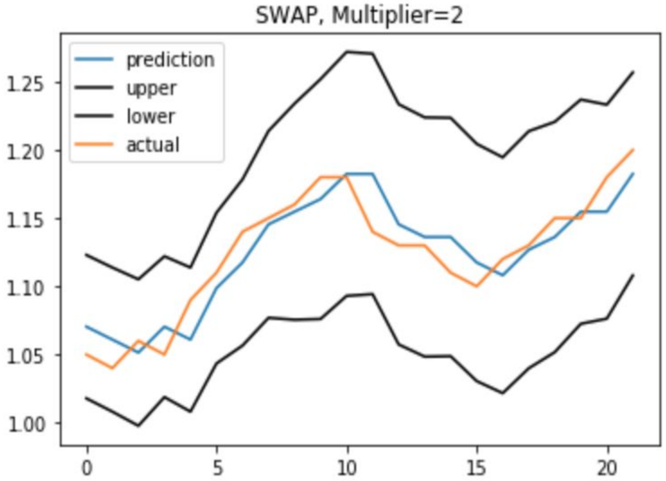
Yufei Zhang




breach=8



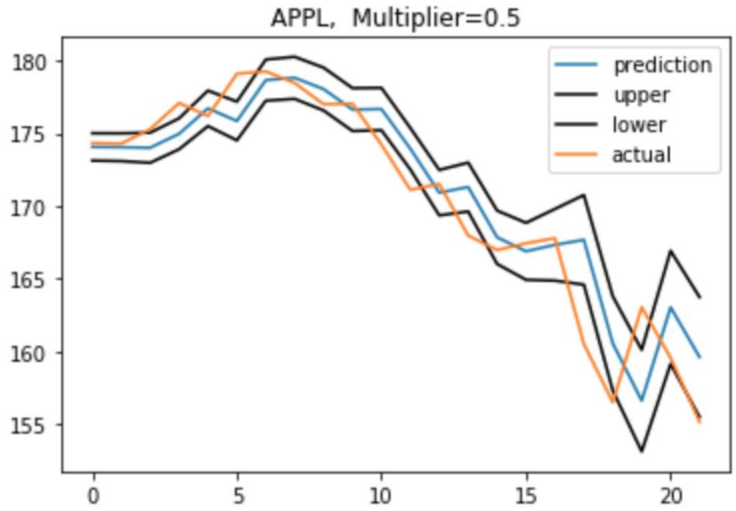

breach=1



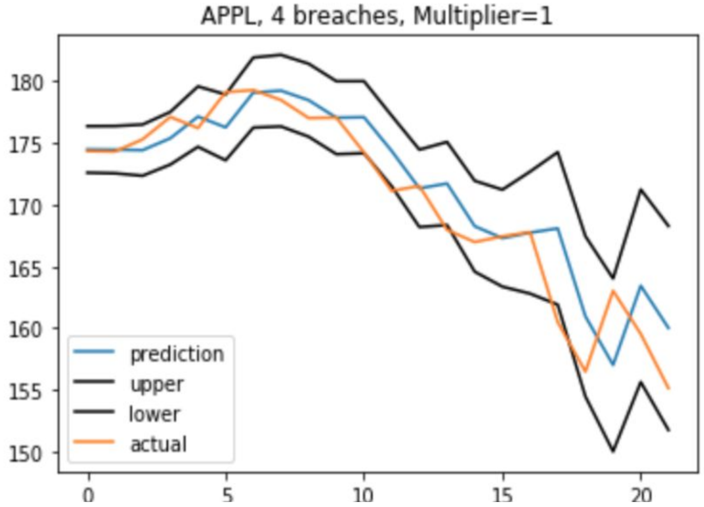

breach=0

# APPL

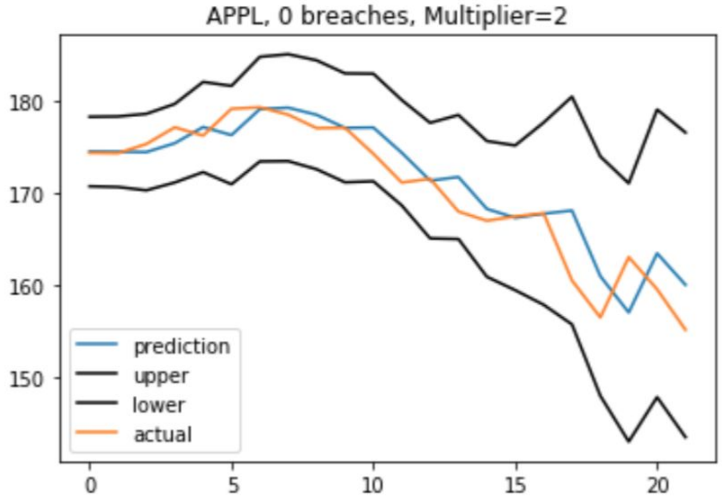
Yufei Zhang




breach=10



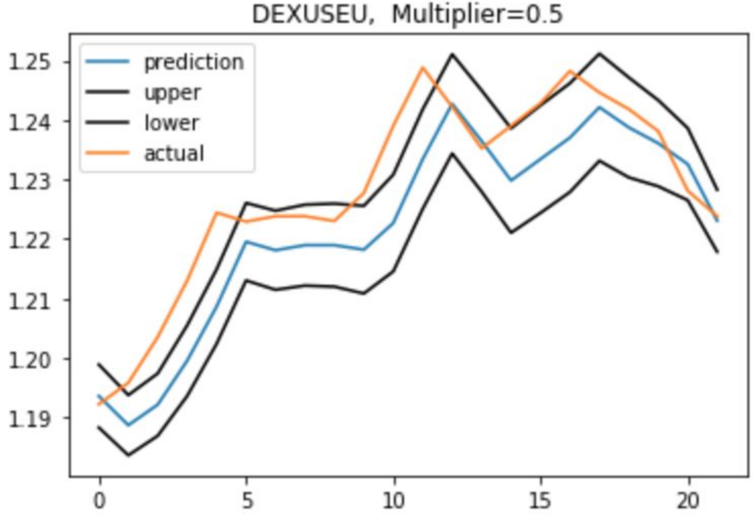

breach=4



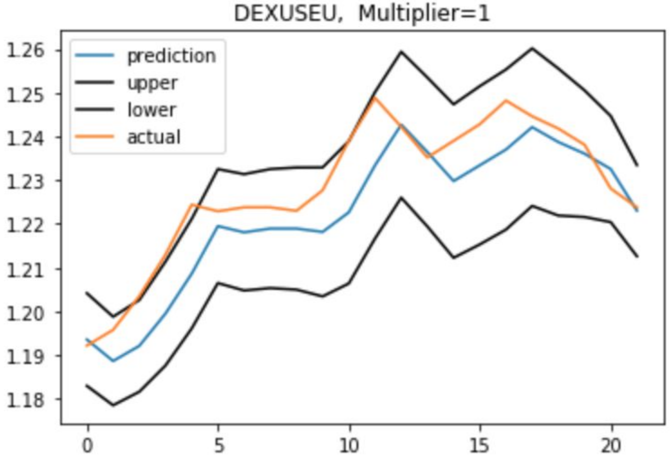

breach=0

# DEXUSEU

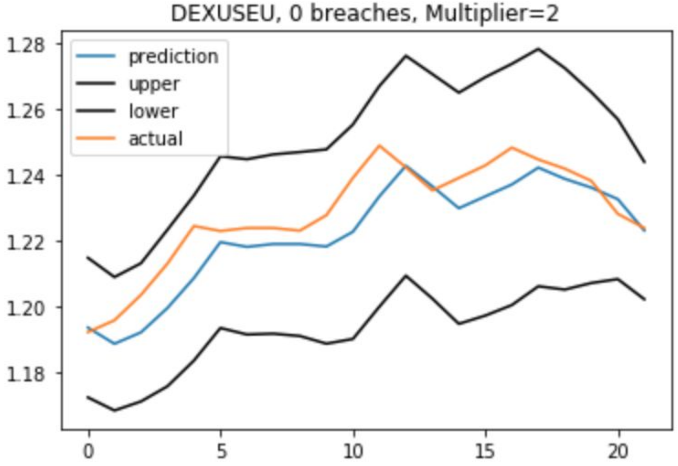
Yufei Zhang




breach=10



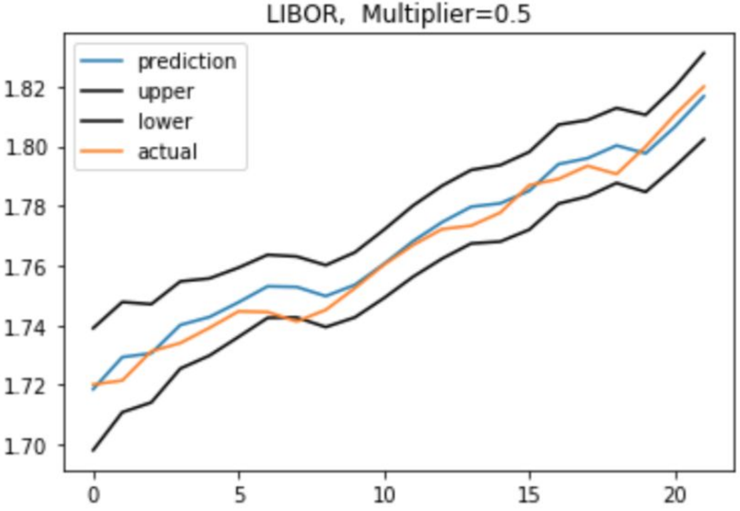

breach=4



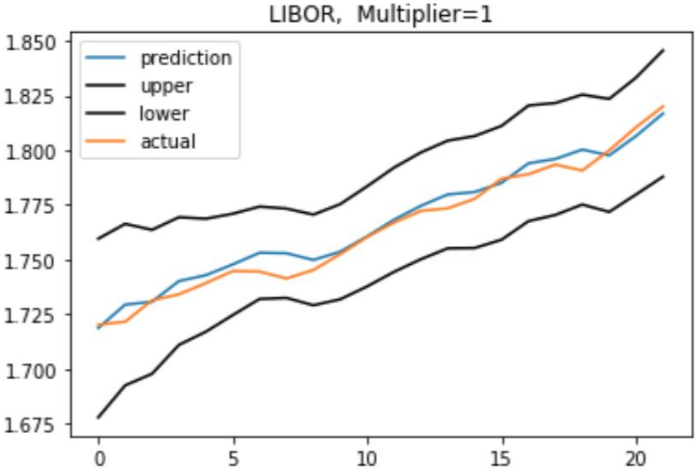

breach=0

# LIBOR

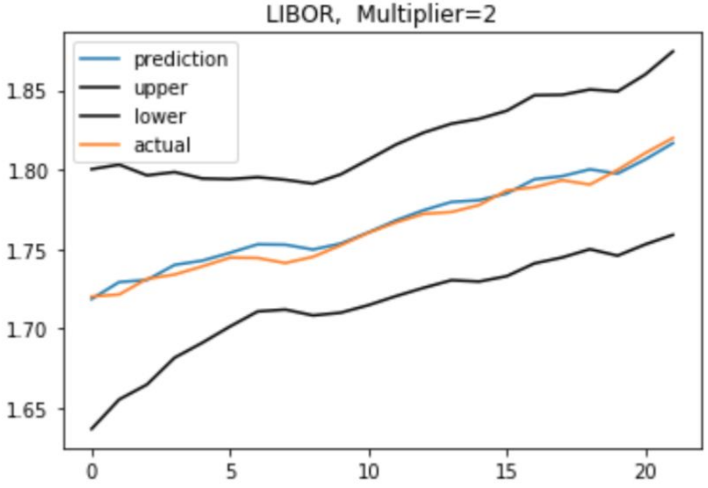
Yufei Zhang




breach=1




breach=0




breach=0

Yufei Zhang

# Literature Review of Time Series Data Analysis

| METHOD | DETAIL | PROS | CONS |
|---|---|---|---|
| SIMPLE MOVING AVERAGE | Adding up the last ‘n’ period’s values and then dividing that number by ‘n’.<br>Moving average value is considering as the forecast for next period | • Quickly detect trend<br>• A moving average is used to smooth out irregularities (peaks and valleys) to easily recognize trends. | • Not accurate compared with other methods |
| EXPONENTIAL SMOOTHING | Exponential Smoothing assigns exponentially decreasing weights as the observation get older. | • Can “smoothing” out the data by removing much of the “noise” (random effect) from the data by giving a better forecast. | • Restrict to data with no trend or seasonal pattern |
| ARIMA | The parameters used in the ARIMA is (P, d, q) which refers to the autoregressive, integrated and moving average parts of the data set, respectively. | • Simple<br>• Achieve great performance on many data set | • Fail to detect **complex** time series patterns that cannot be determined by simple parametric models. |
| ARITIFICIAL NEURAL NETWORK | Machine learning approach that models human brain and consists of a number of artificial neurons | • Flexible<br>• Detect complex patterns | • Takes long time to run<br>• Often require more data than other models |

Reference:
https://www.bistasolutions.com/resources/blogs/5-statistical-methods-for-forecasting-quantitative-time-series/
https://datascience.stackexchange.com/questions/12721/time-series-prediction-using-arima-vs-lstm

# Random Walk Test usig MatLab econometrics toolbox

Yufei Zhang

```
[h,pValue,stat,cValue,ratio] = vratiotest(data)
```

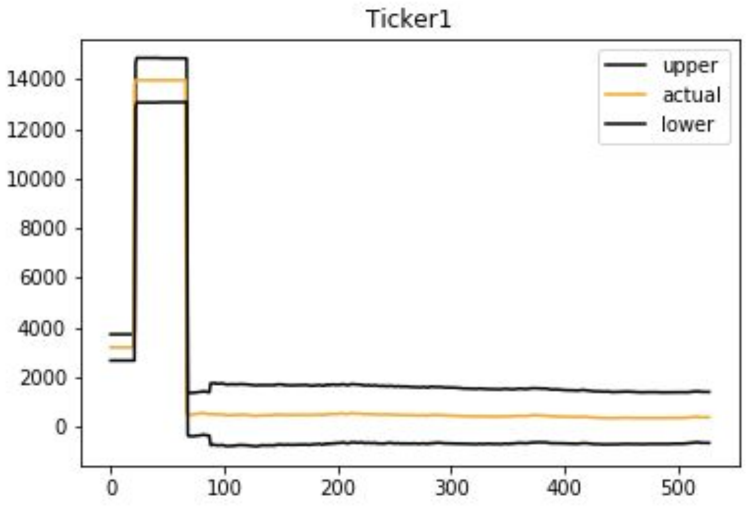
| | 1 | 2 | 3 |
|---|---|---|---|
| h (logical) | 0 | 1 | 0 |
| Random Walk? | Y | N | Y |

| 2.1 | 2.2 |
|---|---|
| 1 | 0 |
| N | Y |

h=0 : vratiotest does not reject the hypothesis that a random walk is a reasonable model for the stock series.
h=1 : vratiotest reject the hypothesis that a random walk is a reasonable model for the stock series.

# Ticker 1

Yufei Zhang




Random Walk: Y

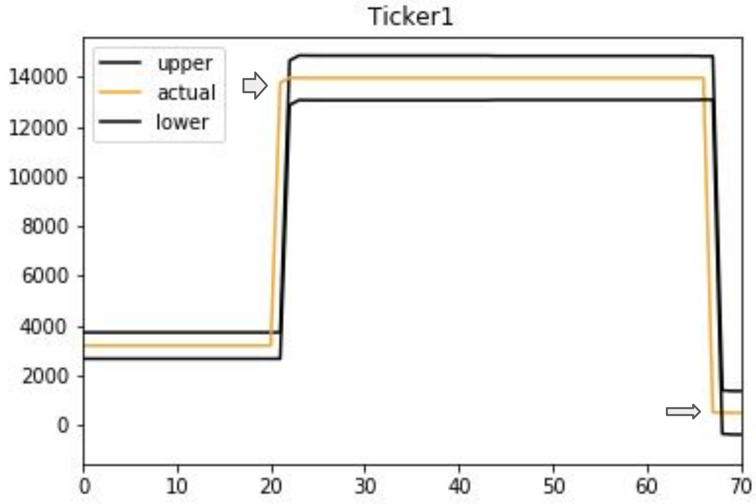
Train : 819 days

Test : 528 days

Model: Persistence Model

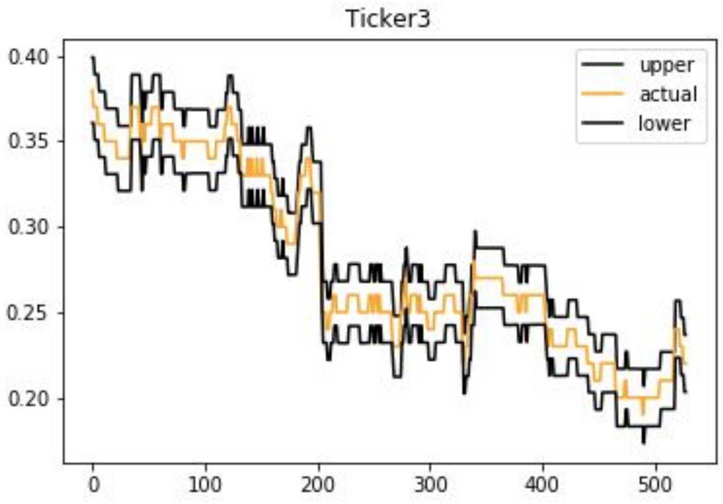
Breachs : 2

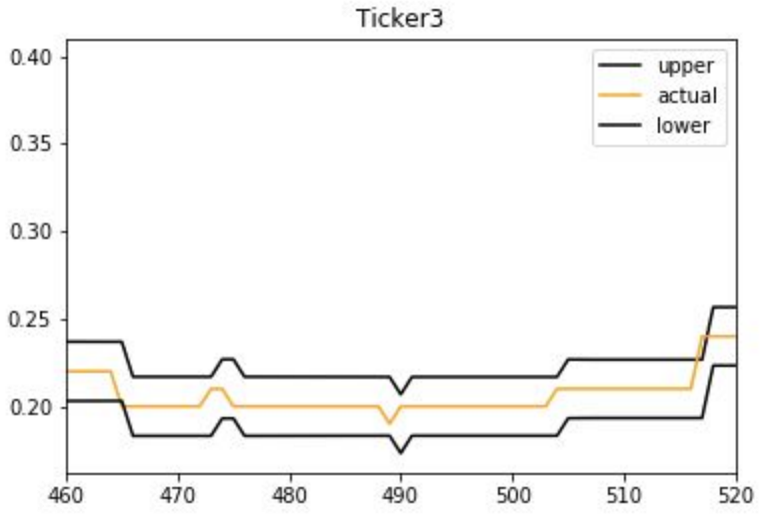
Breach Position: 21, 67



# Ticker 3

Yufei Zhang




Random Walk: Y

Train : 819 days

Test : 528 days

Model: Persistence

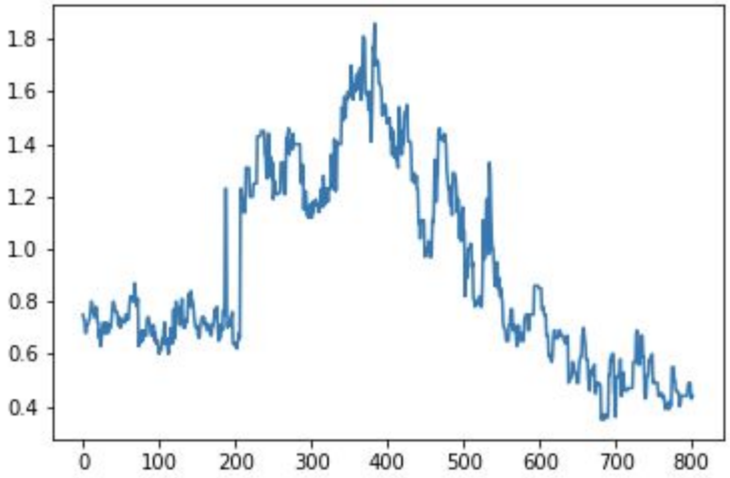
Breachs : 13

Breach Position: 34, 43, 44, 60, 132, 203, 204, 274, 329, 336, 339, 465,517

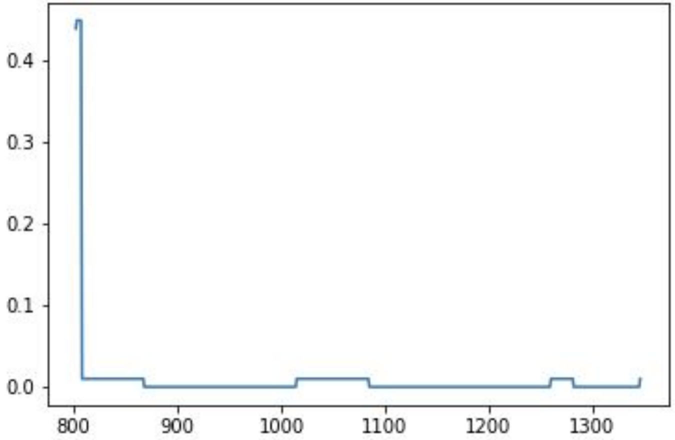
I tried LSTM for ticker 3. Number of breaches and position of breaches were exactly same with persistence model.

# Ticker 2

Yufei Zhang



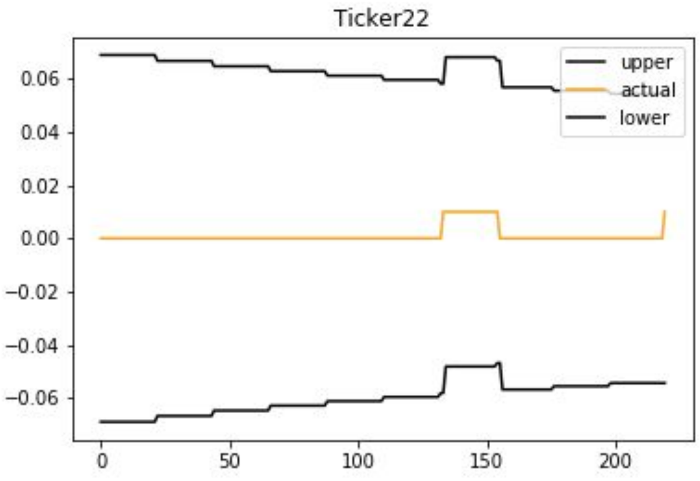

0 - 819 Day




820 ~

# Ticker 2.2

Yufei Zhang




Random Walk: Y

Train : 819 days

Test : 528 days

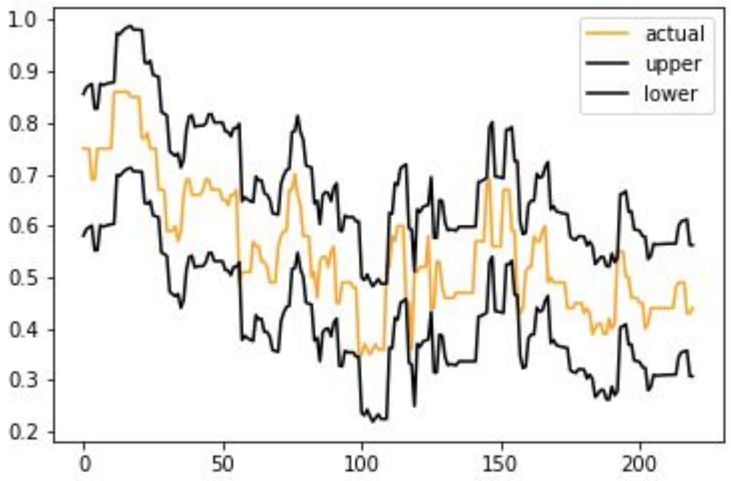
Model: Persistence

Breachs : 0

Breach Position: NA

# Ticker 2.1

Yufei Zhang




Random Walk: N

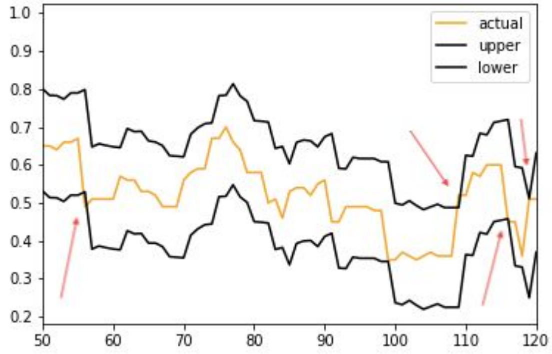
Train : 580 days

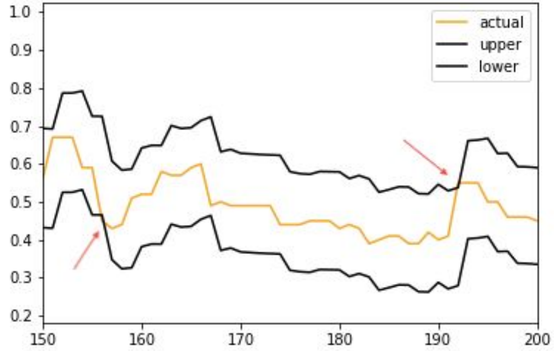
Test : 220 days

Model: LSTM

Breachs : 6

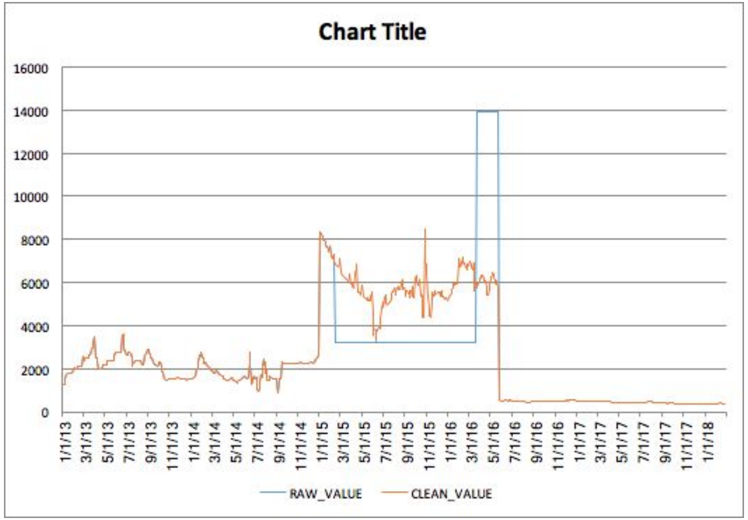
Breach Position: 56, 109, 116, 119, 156, 192

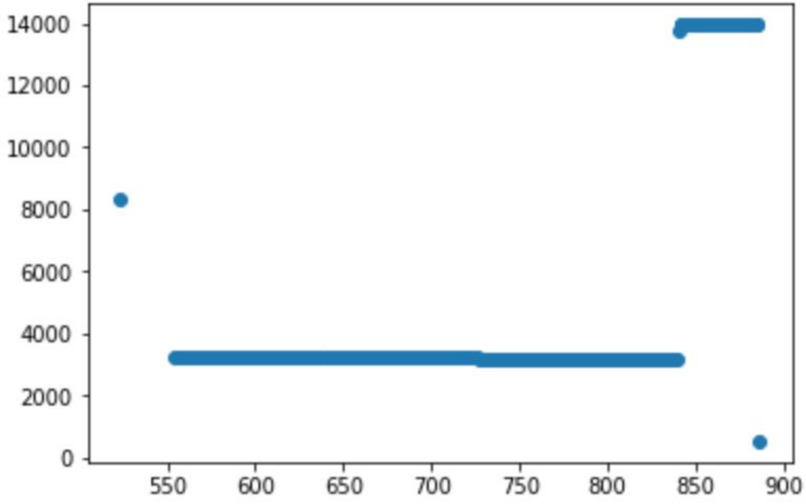


# Ticker 1

Yufei Zhang




Random Walk: Y

Train : 520 days

Test : 528 days

Model: Persistence Model

Breachs : 2

Breach Position: 21, 67

# Ticker 2

Yufei Zhang


Random Walk: N

Train : 520 days

Test : 528 days

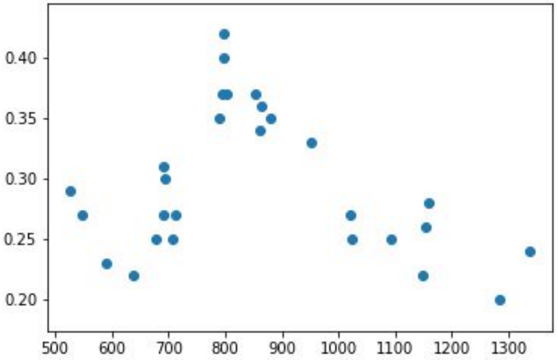
Model: Persistence Model

Breachs : 2

Breach Position: 21, 67

# Ticker 3

Yufei Zhang


| Predict\Actual | Wrong Input | Correct Input | Sum |
|---|---|---|---|
| Wrong Input | 1 | 27 | 28 |
| Correct Input | 1 | 798 | 799 |
| Sum | 2 | 825 | 827 |

# Ticker 4

Yufei Zhang


Random Walk: N

Train : 520 days

Test : 528 days

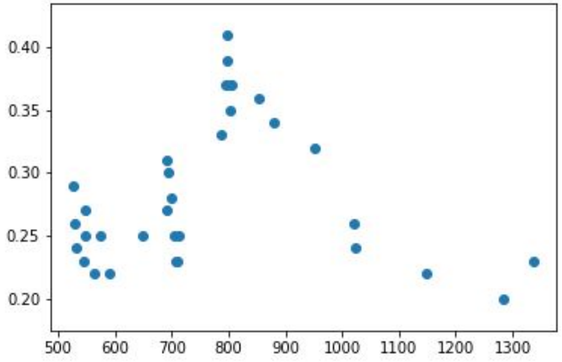
Model: Persistence Model

Breachs : 2

Breach Position: 21, 67

# Ticker 5

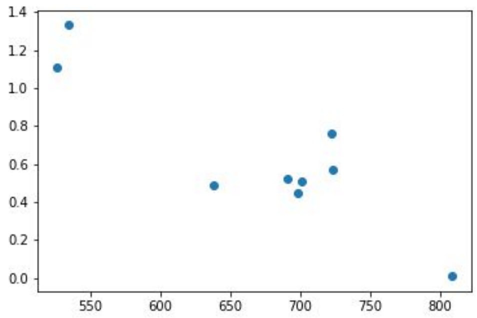
Yufei Zhang


| Predict\Actual | Wrong Input | Correct Input | Sum |
|---|---|---|---|
| Wrong Input | 1 | 32 | 33 |
| Correct Input | 1 | 793 | 794 |
| Sum | 2 | 825 | 827 |

# Ticker 6

Yufei Zhang

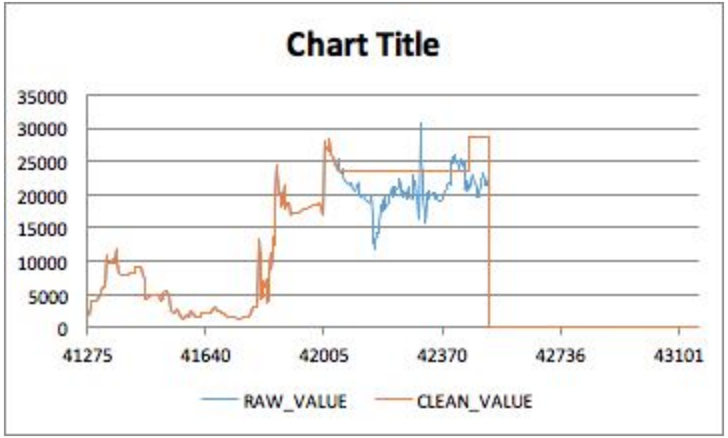




| Predict\Actual | Wrong Input | Correct Input | Sum |
|---|---|---|---|
| Wrong Input | 1 | 8 | 9 |
| Correct Input | 0 | 818 | 818 |
| Sum | 1 | 826 | 827 |

Train : 520 days

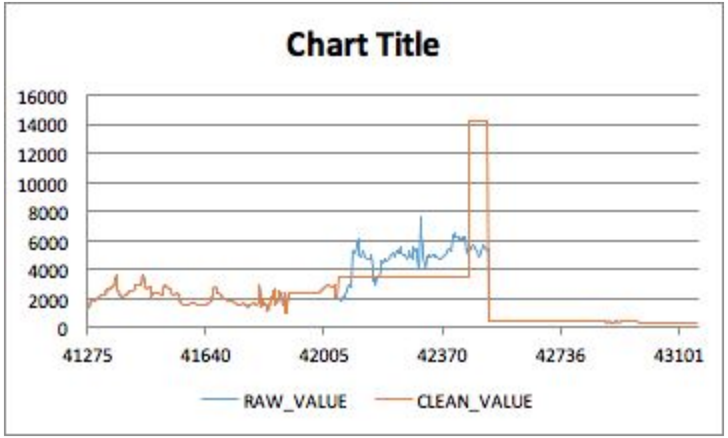
Test: 827 days

## Ticker 7

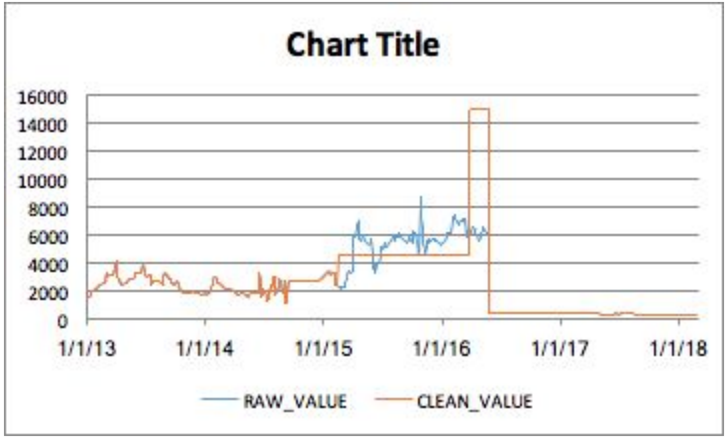



## Ticker 8

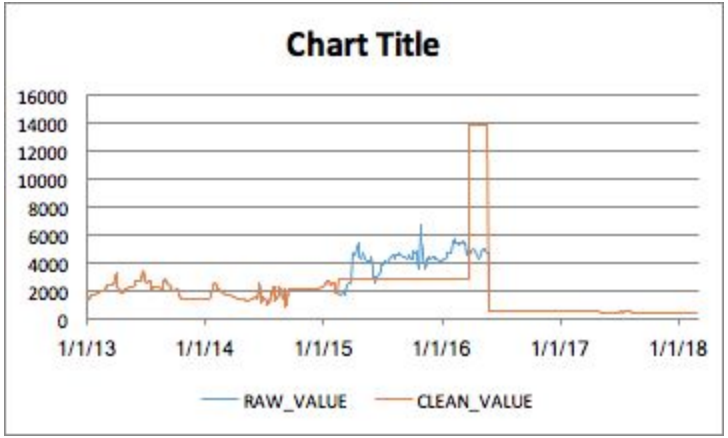
Yufei Zhang




## Ticker 9

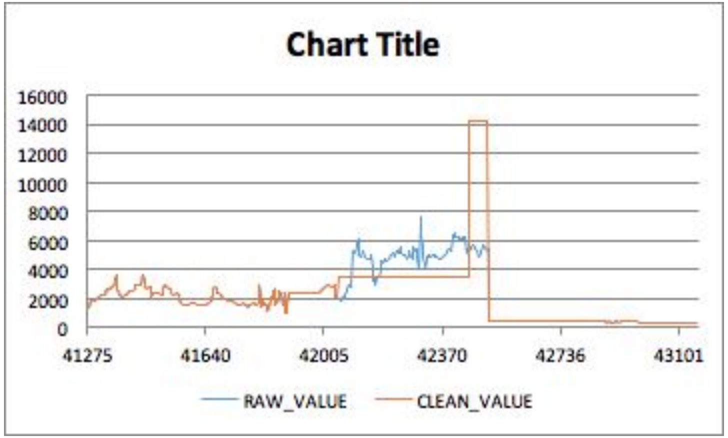



## Ticker 10



# Ticker 8

Yufei Zhang




Random Walk: Y

Train : 520 days

Test : 528 days

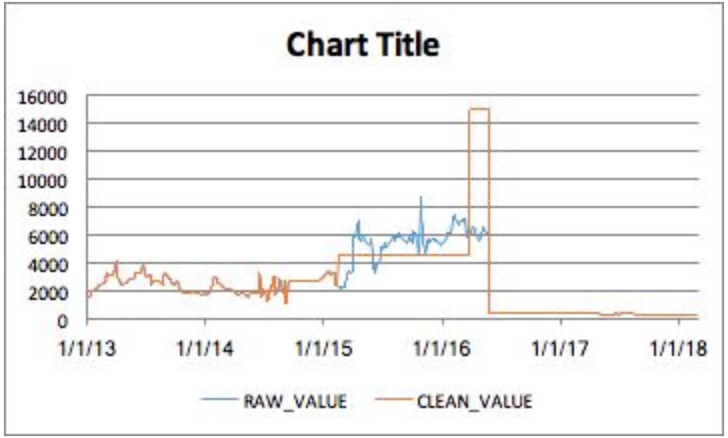
Model: Persistence Model

Breachs : 2

Breach Position: 21, 67

# Ticker 9

Yufei Zhang




Random Walk: Y

Train : 520 days

Test : 528 days

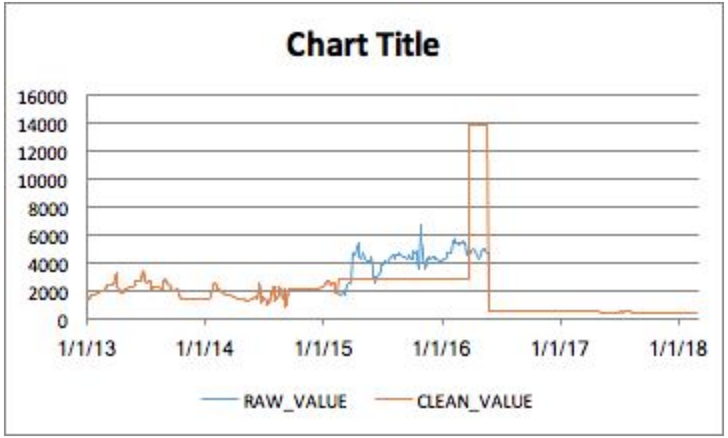
Model: Persistence Model

Breachs : 2

Breach Position: 21, 67

# Ticker 10

Yufei Zhang




Random Walk: Y

Train : 520 days

Test : 528 days

Model: Persistence Model

Breachs : 2

Breach Position: 21, 67

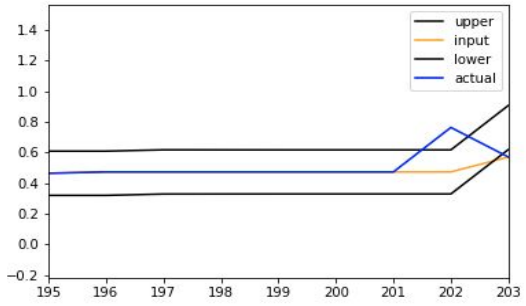
# Persistence Model

Train : 520 Test :827

Yufei Zhang

## Problem Group 1: Fail to detect flat.

From model's view, the "anomaly" is very consistent  with previous days.



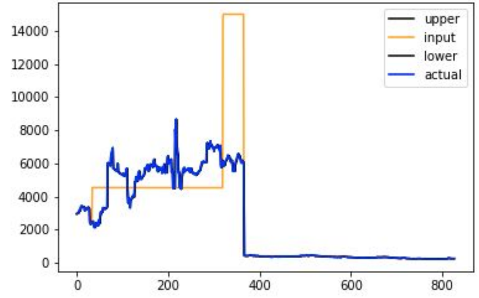

| Ticker | Multiplier | Missed Anomalies | Total Anamalies | Recall | Precision |
|---|---|---|---|---|---|
| 2 | 2 | 1 | 1 | 0 | 0 |
| 3 | 1.5 | 1 | 2 | 0.5 | 0.0238 |
| 4 | 2 | 1 | 1 | 0 | 0 |
| 5 | 1.5 | 1 | 2 | 0.5 | 0.0192 |
| 6 | 2 | 0 | 1 | 0 | 0 |

Yufei Zhang

# Problem Group 2: Input is flat, while real data is not.

Extremely small multiplier can increase recall. But the problem is caused by wrong flat input, and should not be solved in this way.

Ticker 1 and 14 can use a normal multiplier just because real data doesn't come across the flat frequently.

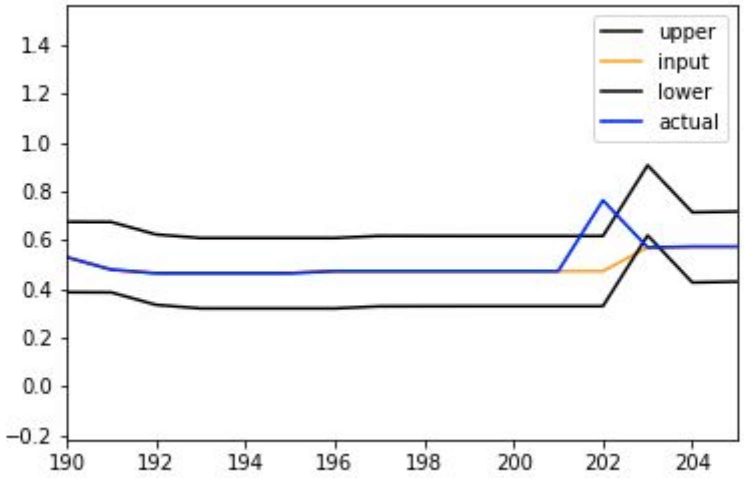


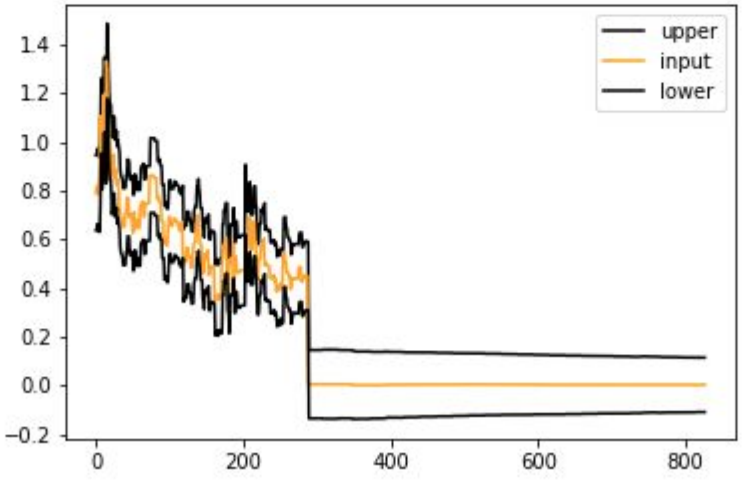

| Ticker | Multiplier | Missed Anomalies | Total Anamalies | Recall | Precision |
|---|---|---|---|---|---|
| 1 | 1 | 1 | 332 | 0.9969 | 0.9792 |
| 7 | 0.0028 | 0 | 334 | 1 | 0.9002 |
| 8 | 0.05 | 0 | 332 | 1 | 0.8447 |
| 9 | 0.000001 | 36 | 750 | 0.952 | 0.9107 |
| 12 | 0.005 | 1 | 332 | 0.9969 | 0.8422 |
| 13 | 0.5 | 1 | 332 | 0.9969 | 0.9792 |
| 14 | 2 | 0 | 332 | 1 | 0.9910 |

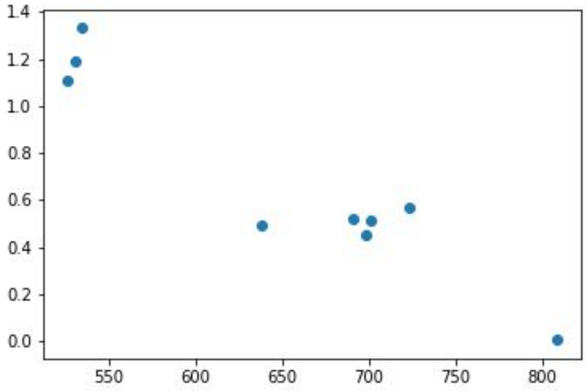
Yufei Zhang

# Ticker 2

Missed Anomalies : 1



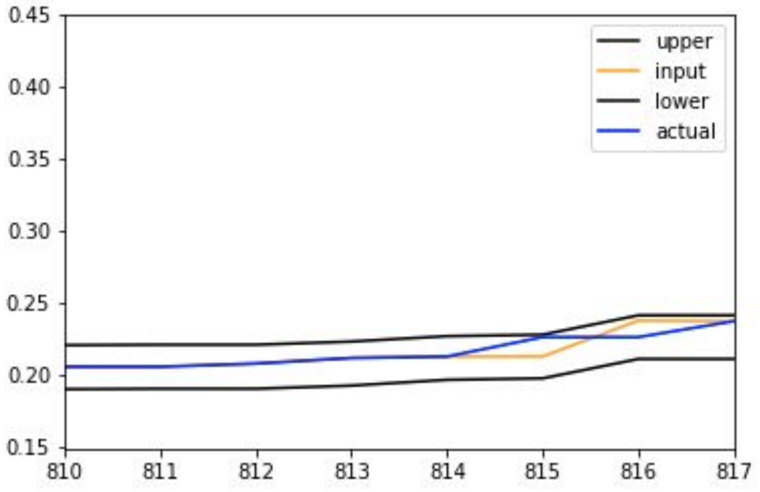

From models's view, input of No722 day is very normal.

Flat Problem!

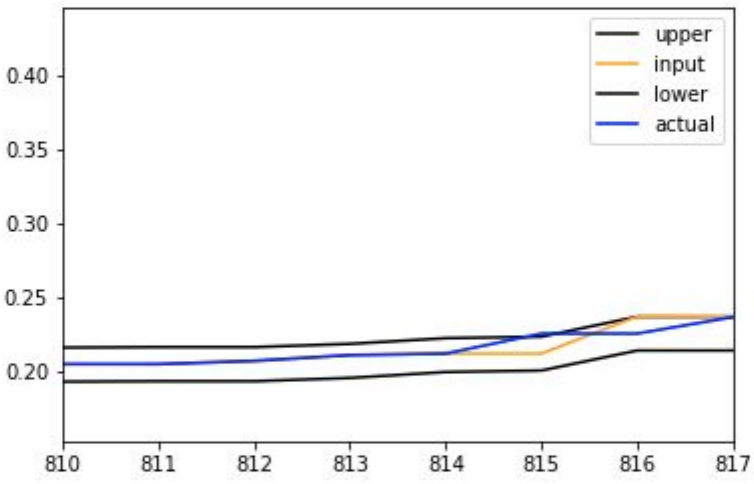
Yufei Zhang

# Ticker 3

Missed Anomalies : 1




Multiplier=2




Multiplier=1.5

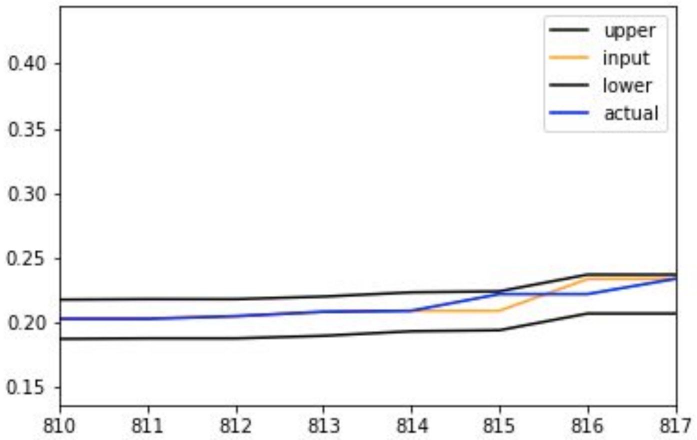
Decreasing Multiplier can solve the second anomaly

For the first anomaly : Flat Problem!

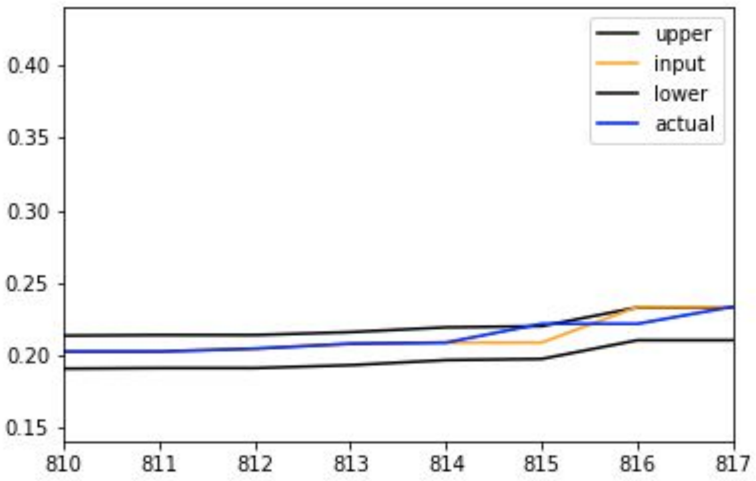
Yufei Zhang

# Ticker 5

Missed Anomalies : 1




Multiplier=2




Multiplier=1.5

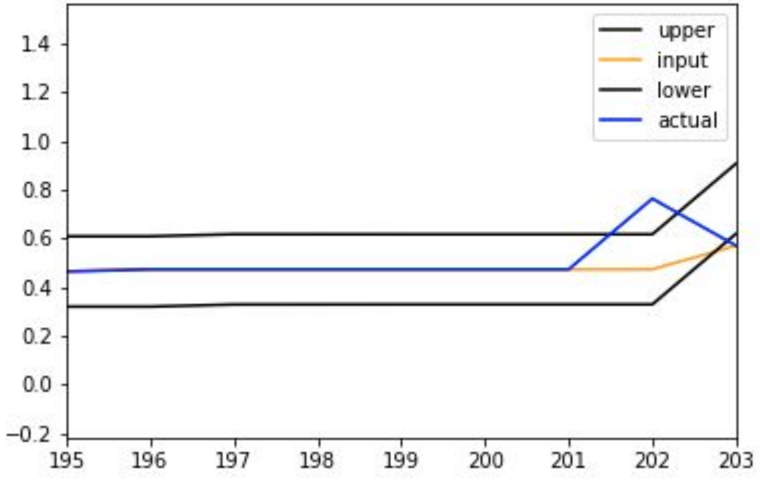
Decreasing Multiplier can solve the second anomaly

For the first anomaly : Flat Problem!

Yufei Zhang

## Ticker 4

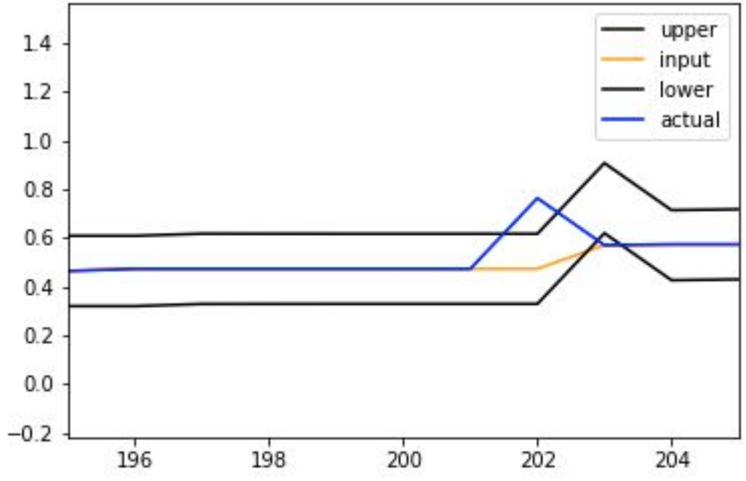
Missed Anomalies : 1




Flat Problem!

## Ticker 6

Missed Anomalies : 1

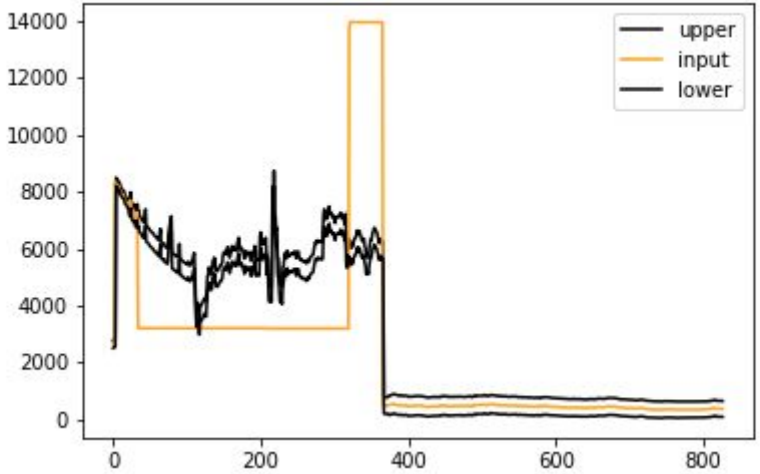


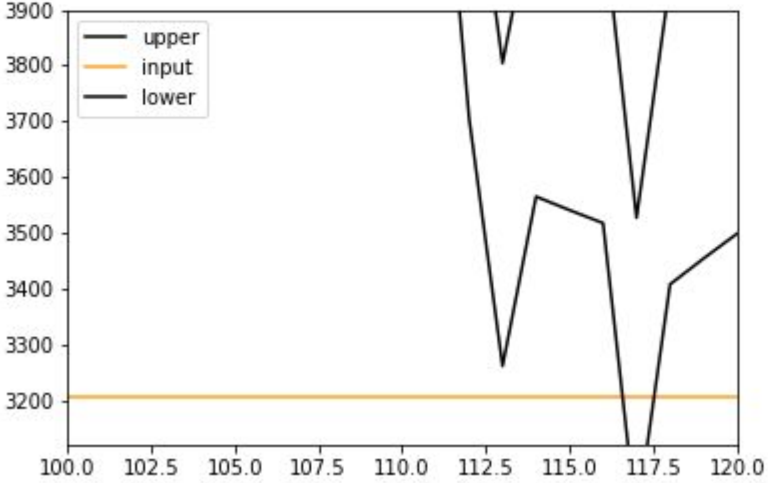

Flat Problem!

Yufei Zhang

# Ticker 1

## Missed Anomalies : 1

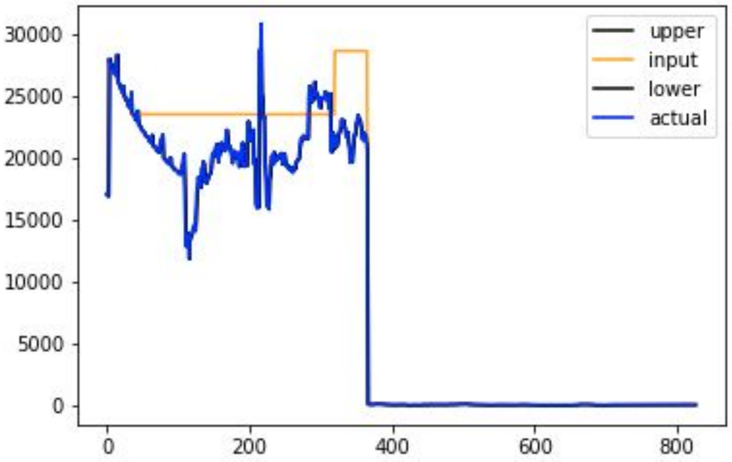


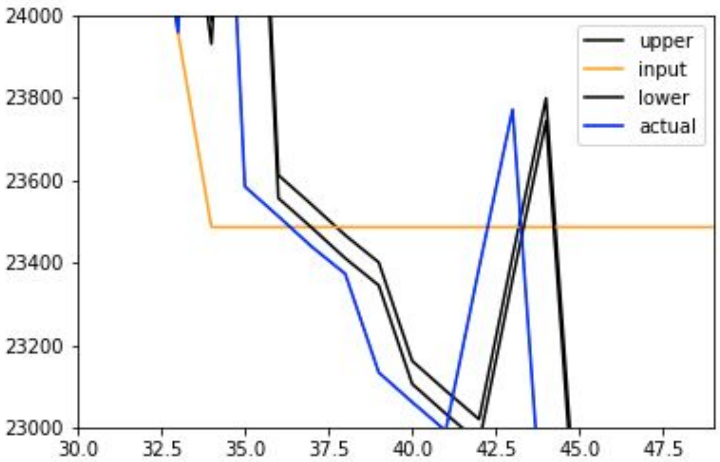



Yufei Zhang

# Ticker 7

Missed Anomalies : 0  Multiplier : 0.0028



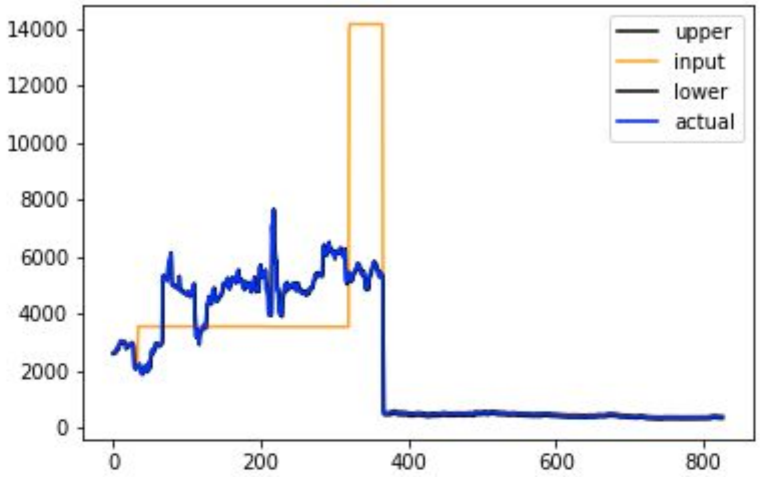



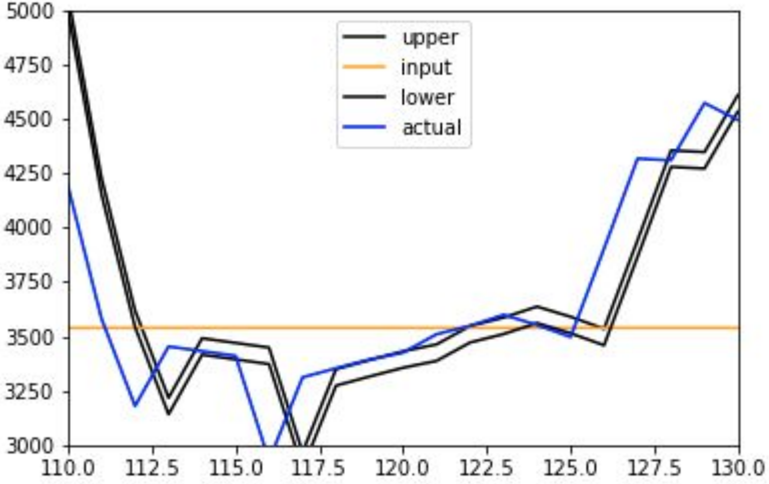

when Multiplier=0.003, Missed Anomalies : 0

Reason : Input happens to be in the range

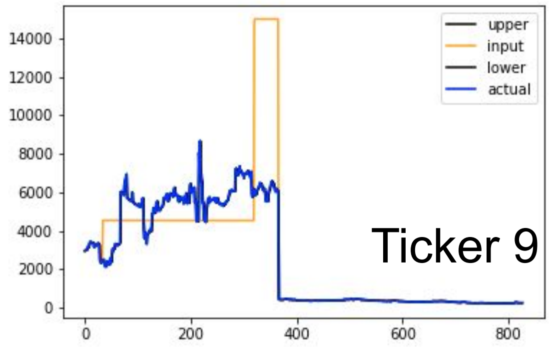
Yufei Zhang

# Ticker 8

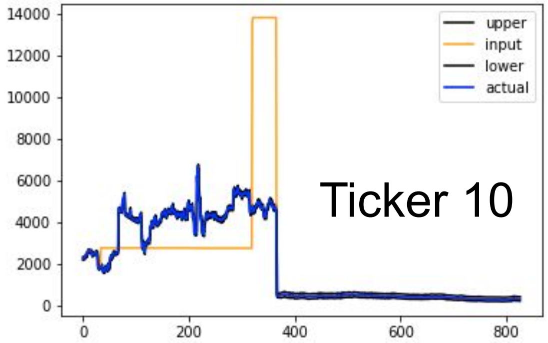


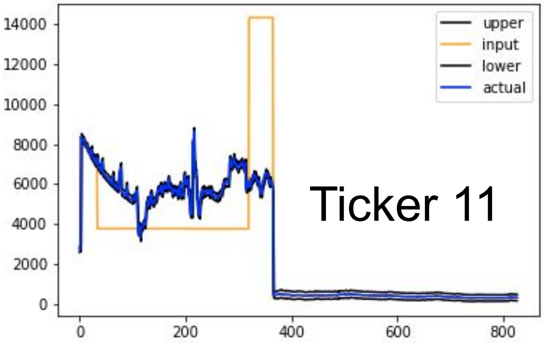
Yufei Zhang



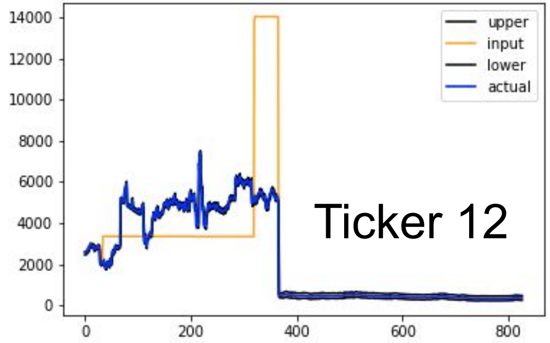

Ticker 9



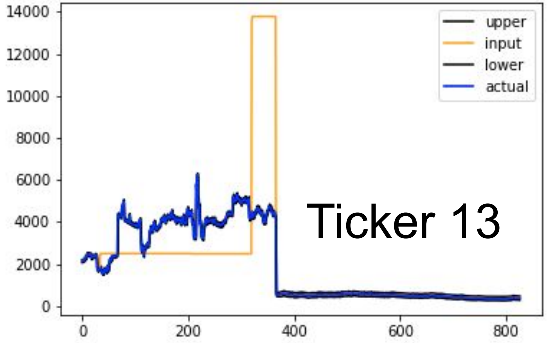

Ticker 10



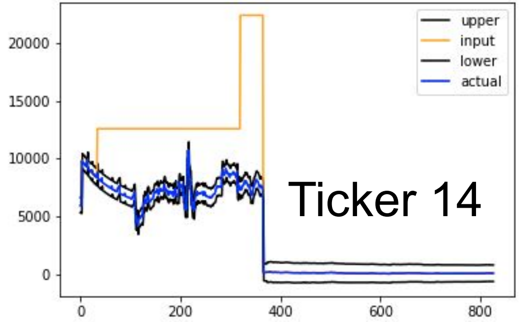

Ticker 11




Ticker 12




Ticker 13




Ticker 14

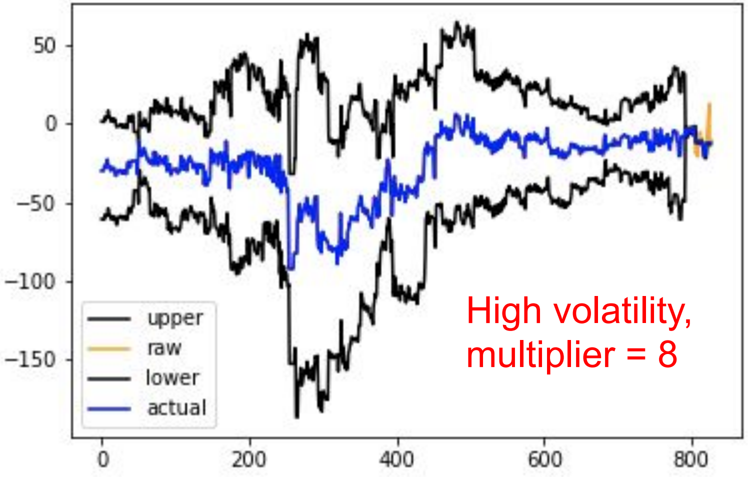
Lucky! Multiplier = 2 can achieve recall =1

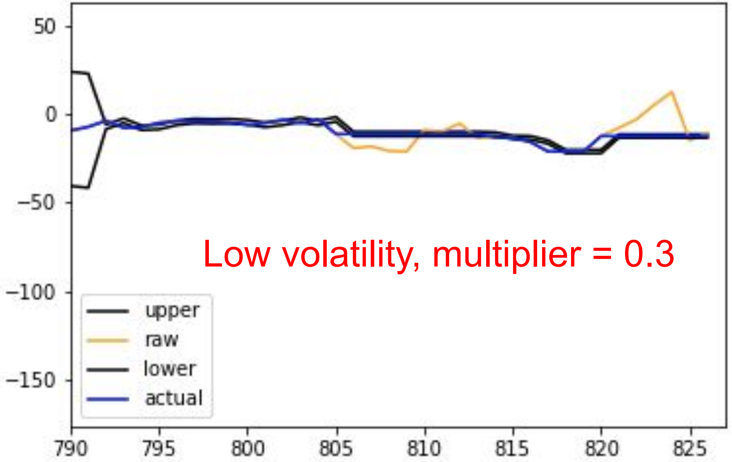
| Ticker | Multiplier | Total Breaches | Total Anamalies | # False Positive | Missed Anomalies | Recall | Precision |
|---|---|---|---|---|---|---|---|
| 15 | 1.3 | 364 | 307 | 57 | 0 | 1 | 0.8434 |
| 16 | 5 for Day 521-961<br>0.05 for the rest | 345 | 317 | 28 | 6 | 0.9810 | 0.9014 |
| 17 | 10 for Day 521-1313<br>0.3 for the rest | 26 | 17 | 9 | 3 | 0.8235 | 0.5384 |

# Ticker 17

| CLEAN_DATE | BEFORE_ADJUSTMENT | ADJUSTMENT | CLEAN_VALUE |
|---|---|---|---|
| 8/1/16 | -46.605 | -47.1376 | -47.1376 |
| 5/18/17 | -21.0302 | -21.0238 | -21.0238 |
| 5/19/17 | -21.0302 | -21.0174 | -21.0174 |
| 1/31/18 | -19.1105 | -11.0808 | -11.0808 |
| 2/1/18 | -18.1049 | -11.0808 | -11.0808 |

Missing anomalies.
Changes were too small to detect.

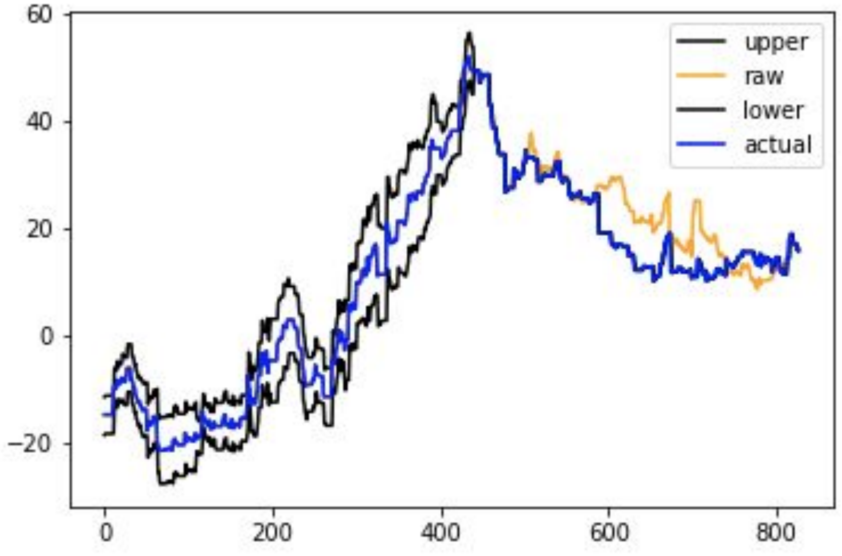


Yufei Zhang

# Ticker 16

| Day | Multiplier | Total Breaches | Total Anamalies | # False Positive | Missed Anomalies | Recall | Precision |
|---|---|---|---|---|---|---|---|
| 521-961 | 5 | 345 | 317 | 28 | 6 | 0.9810 | 0.9014 |
| 962- | 0.05 | | | | | | |



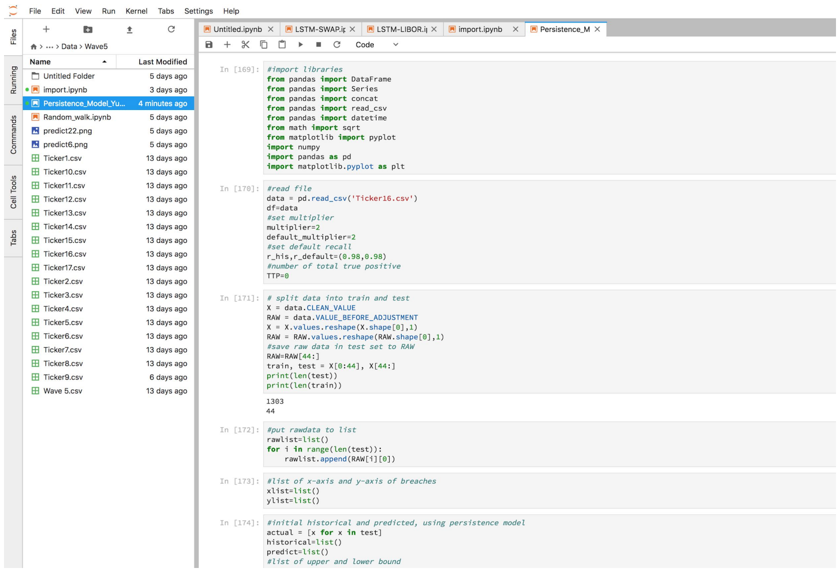
# Persistence Model -- Iteratively Test Multiplier

Yufei Zhang

**Environment**: Jupyter Notebook / Jupyter Lab. Free.

**Files**

- import.ipynb
  - Input: a csv file with all data (various tickers, raw value, clean value, adjustments…)
  - Output: csv files. Number of output files = number of tickers. Each csv file contains data of one ticker.
  - Function: Import and pre-processing data. Can be used to split different tickers to seperate files.
  - How to use: save with sourcing csv file (e.g. wave5.csv). Run from the first cell.
- Persistence_Model.ipynb
  - Input: Output of import.ipynb, ticker csv files.
  - Output: Multiplier, precision, recall, breaches.
  - Function: Predict values and calculate bounds. Logic can be found in next slide. Text anotations can be found in code.
  - How to use: save with sourcing csv file (e.g. ticker1.csv). Run from the first cell.



# Persistence Model -- Iteratively Test Multiplier

Yufei Zhang

## Initial Work:

Initial Train : 44 days

Test : Iteratively train and test every 22 days

1. Split

Apply Persistence model on initial Train set and get RMSE

2. Initialize RMSE

Set initial and default multiplier as 2.
Set initial and defaultRecall as 0.98.

3 Initialize variables

## Iteratively Work:

Read next 22 days and see if any adjustment happens.

4. Read Next

No adjustment

Use default multiplier . Return to step 4.

else

Apply multiplier (initial as 2) and RMSE to the 22 days. Calculate P and R.

5. Test on 22 days. Get P and R

If Recall decrease or less than default (0.98), decrease multiplier. Otherwise, increase multiplier.

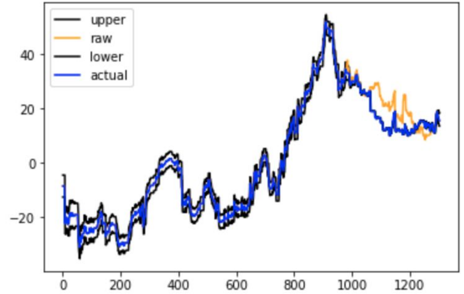
6. Adjust Multiplier

# Output of Persistence_Model.ipynb

1

| | | | |
|---|---|---|---|
| 37 | NA | NA | 2 |
| 38 | NA | NA | 2 |
| 39 | NA | NA | 2 |
| 40 | NA | NA | 2 |
| 41 | NA | NA | 2 |
| 42 | NA | NA | 2 |
| 43 | NA | NA | 2 |
| 44 | 0.5 | 1.0 | 0.8235294117647058 |
| 45 | 0.68 | 1.0 | 0.3843137254901961 |
| 46 | 0.64 | 1.0 | 0.15824682814302193 |
| 47 | 0.77 | 1.0 | 0.07384851980007691 |
| 48 | 1.0 | 1.0 | 0.08369498910675383 |
| 49 | 1.0 | 1.0 | 0.09485432098765433 |
| 50 | 1.0 | 1.0 | 0.10750156378600823 |
| 51 | 1.0 | 1.0 | 0.12183510562414265 |
| 52 | 1.0 | 1.0 | 0.13807978637402835 |
| 53 | 1.0 | 1.0 | 0.15649042455723214 |
| 54 | 1.0 | 1.0 | 0.17735581449819643 |
| 55 | 1.0 | 1.0 | 0.20100325643128927 |
| 56 | 1.0 | 1.0 | 0.22780369062212782 |
| 57 | 1.0 | 1.0 | 0.25817751603841155 |
| 58 | 0.92 | 0.8 | 0.106308388956993 |
| 59 | NA | NA | 0.106308388956993 |

2



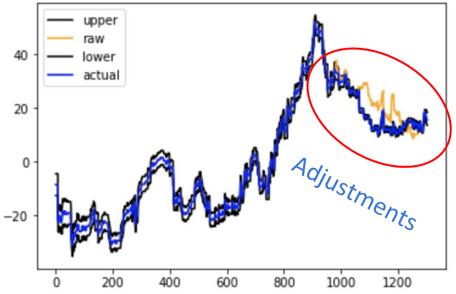

3

```
average recall: 0.9009324009324009
average precision: 0.9866666666666667
total true positive: 282
```

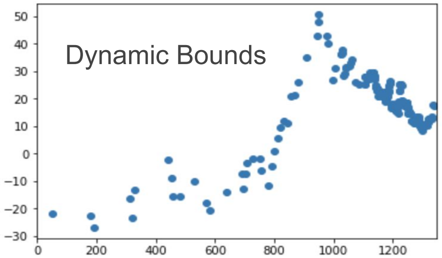
1. Table of p, r and multiplier after adjustment in each period. IF there's no adjustment in that period, p and r will be "NA".
2. Visualization of data and bounds.
3. Average recall and average precision in 60 periods, and number of total true positives.

Yufei Zhang

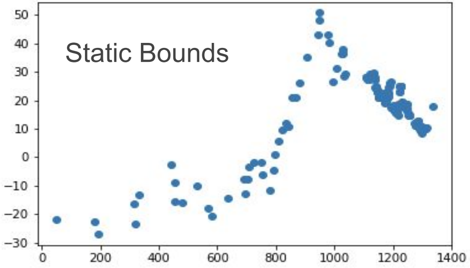
- Objective
  - Detecting anomalies using Prediction + Dynamic Bound
  - Predicting one day forward using Persistence Model
  - Automatically adjusting bound by iteratively train and test on given data
- Outcome
  - Prediction -- Using observation of (N-1)th day as prediction to Nth day.
    - Comparison between LSTM and Persistence model showing that Persistence Model is more suitable for market data, which is mostly a random walk.
  - Dynamic Range -- Adjusting range after every testing.
    - Taking every 22 days as a train/test slice.
    - Testing: Applying current range to a test slice. Using precision and recall as result of test.
    - Training: Adjusting bound when recall is not satisfying.
- Conclusions
  - Persistence model is suitable for providing a prediction of market data.
  - Dynamic bound can learn from the past pattern and self-adjust to fit the ticker.




Bounds and adjustments.
# Total Adjustments = 317




Scatter plot of breaches using Dynamic Bounds.
# Well-detected = 282




Scatter plot of breaches using Static Bounds.
# Well-detected = 189