

AI capstone HW1

Writer : 111652040 吳泓諺

code is available at [NYCU_Artificial_Intelligence_Capstone_Labs](https://github.com/jimmy93029/NYCU_Artificial_Intelligence_Capstone_Labs)

(https://github.com/jimmy93029/NYCU_Artificial_Intelligence_Capstone_Labs), where I create dataset with teammate 111652017

1. intro

As the Trump government embrace the cypercurrency, the price of bit coin surge, indicating the new era of crypto.

In this project, we are interested in building a price prediction model for cypercurrency. We will first make our dataset with multiple technical indicators, then compare different algorithms for price prediction.

Finally, we aim to learn how to do quantitative trading from the experience.

1. Dataset



We take [Binance](https://developers.binance.com/docs/derivatives/quick-start) (<https://developers.binance.com/docs/derivatives/quick-start>), an raw data platform for trading developer, as the source of our dataset. Many developer extract their needed datas from this platform since it is reliable. However, these data is uncleaned, we may have to clean it later.

(1) How to get data

we collect BTCUSDT crypto from 2024-12-01 to 2025-02-01, sampling it every 15 minutes. By querying datas with python requests package, we will get a raw json data. We then reformat it with python pandas, turning it into csv file.

(2) What kind of data is in our dataset, klines_BTC_with_factors.csv

| | A | B | C | D | E | F | G | H | I | J | K | L | M | N |
|---|-----------|---------|---------|---------|---------|----------|-----------|-----------|-----------|------------|------------|-------------|-----------|-----------|
| 1 | open_time | open | high | low | close | volume | taker_buy | taker_buy | return | log_return | volatility | price_range | ma_7 | ma_25 |
| 2 | ##### | 96959.4 | 97000 | 96761.3 | 96761.3 | 1255.041 | 396.161 | 38389791 | -0.002042 | -0.002044 | 0.0027301 | 0.0024619 | 96666.157 | 97354.66 |
| 3 | ##### | 96761.3 | 96775.2 | 96560.1 | 96615.8 | 1232.785 | 474.37 | 45866815 | -0.001504 | -0.001505 | 0.0026873 | 0.002223 | 96675.586 | 97311.204 |
| 4 | ##### | 96615.9 | 96630 | 96420 | 96444.9 | 1416.754 | 666.18 | 64285913 | -0.001769 | -0.00177 | 0.0026922 | 0.0021736 | 96703.529 | 97272.404 |
| 5 | ##### | 96444.8 | 96444.9 | 96222.2 | 96278.5 | 2006.739 | 977.16 | 94105813 | -0.001725 | -0.001727 | 0.0026997 | 0.0023091 | 96668.629 | 97232.94 |
| 6 | ##### | 96278.6 | 96384.5 | 96221.1 | 96304.8 | 1895.229 | 973.346 | 93734826 | 0.0002732 | 0.0002731 | 0.0027009 | 0.0016972 | 96592.129 | 97192.548 |
| 7 | ##### | 96304.7 | 96433.7 | 96233 | 96354.9 | 1242.472 | 697.858 | 67244373 | 0.0005202 | 0.0005201 | 0.0026005 | 0.002084 | 96531.357 | 97153.512 |
| 8 | ##### | 96354.9 | 96507 | 96325.9 | 96349.9 | 1300.137 | 723.849 | 69809679 | -5.19E-05 | -5.19E-05 | 0.002599 | 0.0018795 | 96444.3 | 97102.004 |
| 9 | ##### | 96349.9 | 96538.4 | 96349.8 | 96437.8 | 984.703 | 511.367 | 49329859 | 0.0009123 | 0.0009119 | 0.0025896 | 0.0019574 | 96398.086 | 97058.536 |

To analysis stock price, it is significant to select the right index before analyzing. We consider different kind of index in our dataset, such as

Time step information : open_time, close_time

Candlestick shape (OHLC): open, high, low, close

Volume & Trade Activity : volume, quote_asset_volume, num_trades

Buyer-Side Transactions (Taker Volume) : taker_buy_base_asset_volume, taker_buy_quote_asset_volume

For more investigation, we also consider the following technical indicators

Price-Based Indicators : Return, Log Return, Volatility, Price Range

MACD (Moving Average Convergence Divergence) : MACD Line, Signal Line, MACD Histogram

Bollinger Bands : Middle Band, Upper Band, Lower Band, Bollinger Width

others : Moving Averages (MA), RSI (Relative Strength Index)

(3) Data cleaning

As the saying goes, "garbage in, garbage out" (<https://medium.com/tej-api-financial-data-analysis/data-analysis-2-garbage-in-garbage-out-8b7197ff178c>). It is important to maintain data quality in our dataset. To detect noise data within our dataset, we try several unsupervised methods, aiming to clean our data.

We use

| column \ model | description | |
|-----------------|---|--|
| Autoencoder | Autoencoder consists two parts, encoder and decoder. It will compress data in encoder part and decompress it in decoder. Besides, with the mapping of neural network, it is able to capture concepet in high dimension. | |
| IsolationForest | Isolation forest is known for idnetifying anomalies. It will split datas into different groups until all datas are isolated. Moreover, data with the shallow depth is the anomaly | |

- reference : Pytorch layer, [scikit learn outlier detection](https://scikit-learn.org/stable/modules/outlier_detection.html) (https://scikit-learn.org/stable/modules/outlier_detection.html).

2. Algorithms

(1) Supervised learning 1

Decision trees based method is a known way of supervised learning. In the experiment, we would like to compare different methods, testing which is the best model

| model \ column | description | |
|----------------|--|--|
| decision tree | classifying datas based on colomuns | |
| random forest | random forest is a decision tree with Bagging. | |
| XGBoost | XGBoost is a random forest with Boosting. | |

- reference : [sklearn random forest](https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html) (<https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html>), [sklearn Random forest](https://scikit-learn.org/stable/modules/ensemble.html) (<https://scikit-learn.org/stable/modules/ensemble.html>), [XGBoost XGBRegressor](https://xgboost.readthedocs.io/en/stable/python/python_api.html) (https://xgboost.readthedocs.io/en/stable/python/python_api.html).

(2) Supervised learning 2

Since price prediction is a sequence to sequence problem, it is natural to use time-series forecasting method.

| model \ column | description | |
|----------------|--|--|
| LSTM | A LSTM is RNN with forget gate, aiming to fix vanishing gradient problem. | |
| Transformer | A Transformer is famous for its attention layer, which attends to partial keys when training | |

- reference : Tensorflow LSTM (https://www.tensorflow.org/api_docs/python/tf/keras/layers/LSTM), Pytorch Transformer (<https://pytorch.org/docs/stable/generated/torch.nn.TransformerEncoder.html>).

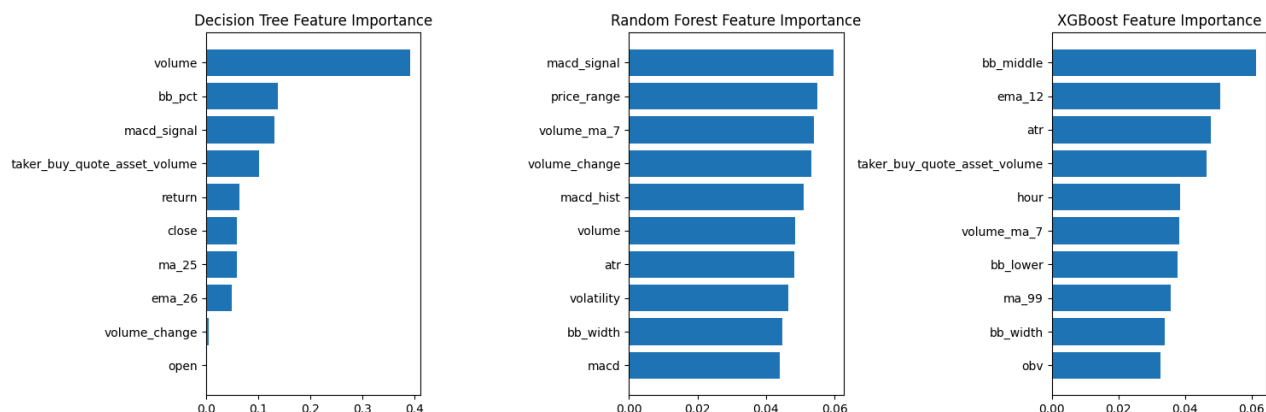
(3) Unsupervised learning

| column \ model | description | |
|------------------------|---|--|
| Guassian mixture model | Guassian mixture model aims to find which group does the data belongs to. To do so, it will perform EM algorithm | |
| Hidden markov model | Hidden markov model assumes the sequential data is a markov chain. It will try to find the hidden state in each data. To do so, it also leverage EM algorithm | |

- reference : sklearn GMM (<https://scikit-learn.org/stable/modules/mixture.html>), hmmlearn HMM (<https://hmmlearn.readthedocs.io/en/latest/api.html#hmmlearn-hmm>).

3. Analysis

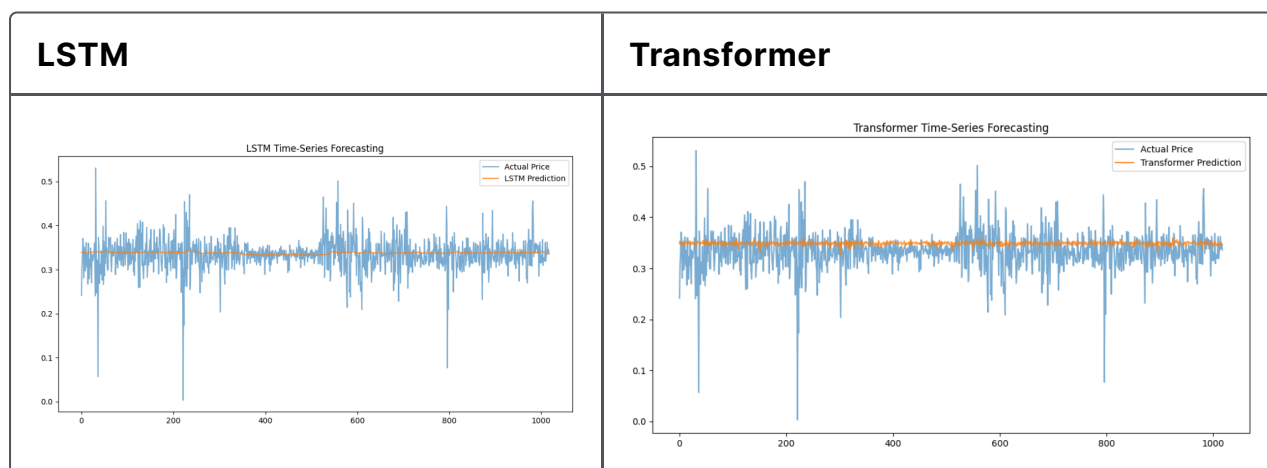
Supervised learning 1 : decision tree vs. random forest vs. XGBoost



| | Decision tree | Random Forest | XGBoost |
|-----|---------------|---------------|---------|
| mse | 0.0.0276 | 0.0.0286 | 0.0329 |

It is interesting that different model view different feature as important. For decision tree, it value volume feature. For Random Forest, it value Moving Average Convergence / Divergence more. For XGBoost, it value Bollinger Bands more.

Supervised learning 2 : LSTM vs. Transformer

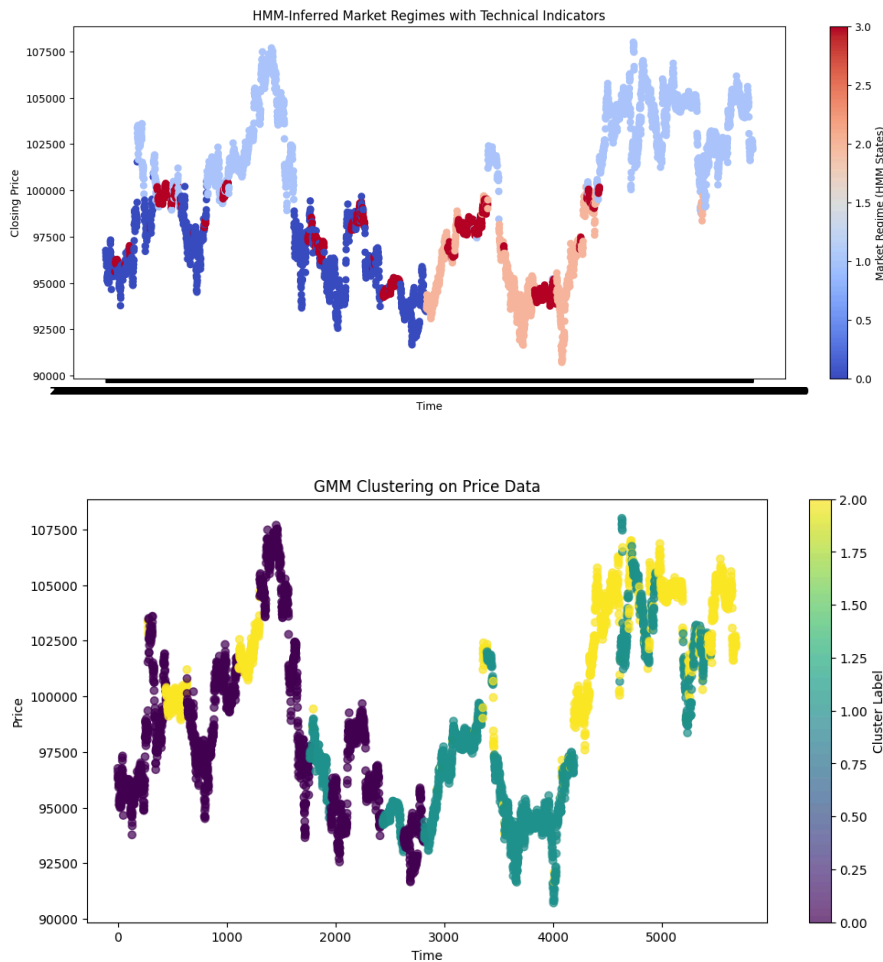


| | LSTM | Transformer |
|-----|--------|-------------|
| MSE | 0.0014 | 0.0017 |

There are 2 findings:





1. LSTM performs slightly better than Transformer. It is because Transformer need large dataset for training, therefore Transformer couldn't learn well with our limited datas
2. Both LSTM and Transformer model output a horizontal line. It turns out that the best policy for trading is to do nothing.

Unsupervised learning : GMM vs. HMM



HMM Cluster Averages Interpretation

The **Hidden Markov Model (HMM)** has segmented the data into four clusters, each representing unique market conditions. Since it has the capacity to learn hidden states, HMM capture the trend of data, determining whether the phase is bearish or bullish.

| Feature | Bullish Market (Green) | Bearish Market (Yellow) |
|---|--------------------------|-------------------------|
|  Price Levels | ~103,293 (Open & Close) | ~94,133 (Open & Close) |
|  Volume | ~1,846 units | ~1,484 units |
|  Volatility | 0.2389% | 0.224% |
|  Returns | 0.0016% (Highest) | 0.0037% |

GMM Cluster Averages Interpretation

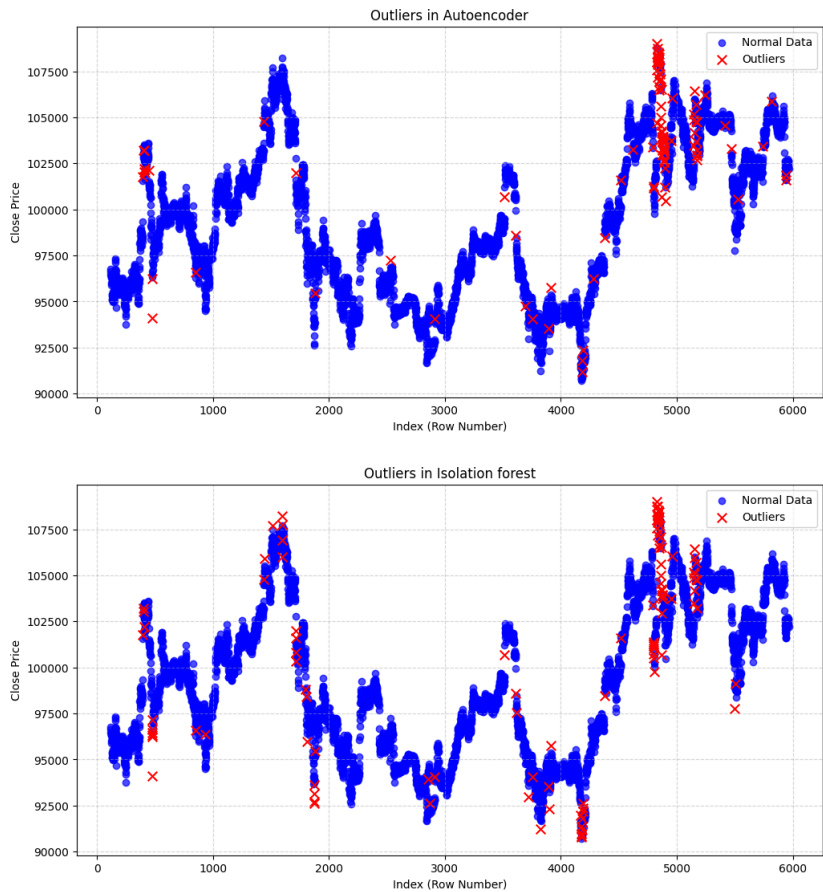
The **Gaussian Mixture Model (GMM)** has segmented the data into three clusters, each representing unique market conditions. Due to its ability of observing variance, it sort the data based on its volatility.

| Feature | Volatile Market | Calm Market |
|---|------------------------------------|-------------------------|
|  Price Levels | ~98,673 (Open & Close) | ~99,935 (Open & Close) |
|  Volume | 3,083 units (Highest) | ~1,085 units (Lower) |
|  Volatility | 0.3417% (Highest) | 0.1739% (Lowest) |
|  Returns | -0.0006859% (Most negative) | -0.0000787% |

4. Experiment

(1) What is the effect of anomaly data

I. Detect the anomaly datas



II. Compare performance of model between cleaned and non-cleaned dataset

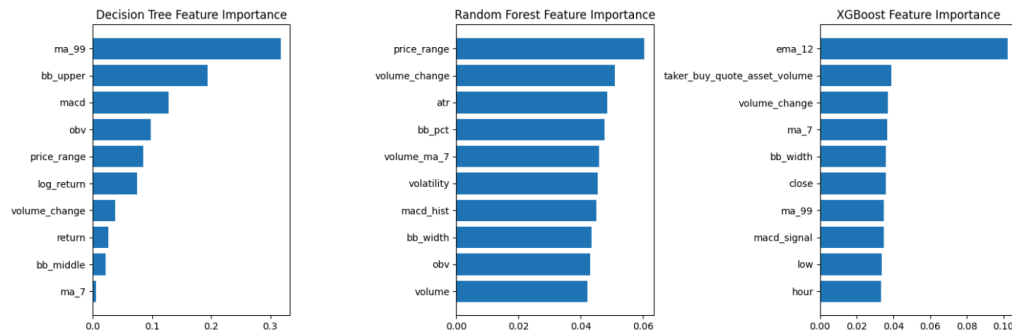
| mae\model | Decision tree | Random Forest | XGBoost |
|------------|---------------|---------------|---------|
| un-cleaned | 0.0317 | 0.0324 | 0.0397 |
| cleaned | 0.0276 | 0.0286 | 0.0329 |

| mse\model | LSTM | Transformer |
|--------------|--------|-------------|
| un-cleaned | 0.0017 | 0.0020 |
| cleaned data | 0.0014 | 0.0017 |

The performance of model does drop a little bit. However, due to the limited data in our dataset, the drop isn't obvious

(2) Augmentation of dataset

For augmentation of dataset, we do so by collecting more stock history data. We collect datas from 2024-05-01 to 2025-02-01, which is 3 times bigger than original dataset

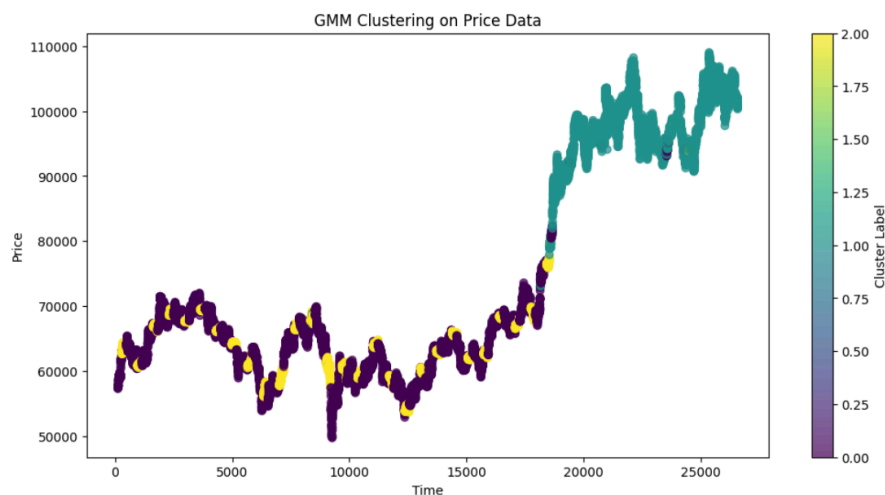


| mae\model | Decision tree | Random Forest | XGBoost |
|-----------|---------------|---------------|---------|
| cleaned | 0.0276 | 0.0286 | 0.0329 |
| augmented | 0.03687 | 0.02496 | 0.0524 |

| mse\model | LSTM | Transformer |
|--------------|--------|-------------|
| cleaned data | 0.0014 | 0.0017 |
| augmented | 0.0014 | 0.0013 |

It is interesting that larger data range deteriorate the precision of decision-tree based model. Besides, price range and volume change turns out to be a more important feature.

Moreover, feeding more data does improve the performance of Transformer. Transformer performs better than LSTM now.



For unsupervised learning, it turns out that a larger price range leads to more variation in clustering. Therefore, the position of bullish market change.

5. Discussion

(1) Based on your experiments, are the results and observed behaviors what you expect?

No, the results is out of my expectation. For exmaple, in Supervised learning, it is suprired to see that decision tree works beter than random forst. It is also suprired that LSTM performs better than Transformer, either.

(2) Discuss factors that affect the performance, including dataset characteristics.

I observe 3 factors which affect the performance.

1. the choosing of model
2. how cleaned is the dataset
3. the size of dataset

It is noteworthy that although the Transformer works well in LLM, it doesn't perform better than LSTM in our experiment.

(3) Describe experiments that you would do if there were more time available.

If there are more time available, I will reproduce some paper in our homework. I am intersted in what is the mainstream way to train price prediction model

(4) Indicate what you have learned from the experiments as well as your remaining questions.

Building our own dataset is an intriguing experience.

I learn that

1. how to maintain data quality in the experience :
Before this experience, I didn't notice that the anomaly datas affect the performance of model a lot.
2. I also learn how to implement different ML method in the experience:
It is suprired to see that some model didn't performance well in real experience.