FRE-GY 7871 News Analytics

Final Report

Project title: Cryptocurrency Trading Strategy Based on News Text

Zhengxuan Yan

Yuxuan Wang

Chang Liu

Xiaolai Li

Contents

1. Goal:

We try to explore and implement machine learning and deep learning techniques to analyze the information in news text data and predict the performance of cryptocurrency as trading signals. We will use those signals to determine whether to buy, sell or hold cryptocurrency. We may create several models using different sources of data and combine those signals together to conclude decisions. Our goal is to beat the benchmark which we define as buy-and-hold and find "alpha". We also want to explore and learn deep learning techniques in natural language processing through this project.

2. Project Overview:

This part, we will include a high-level overview of what we plan to do in this project.

2.1. Data Collection:

We collected data from various sources, including news API, price and volume API and website scraping.

The following links are available data sources:

1) Price and volume data:

   Yahoo Finance: https://pypi.org/project/yfinance/ (Labeling)

2) News:

   Alpha Vantage: https://www.alphavantage.co/

2.2. Processing:

We performed data processing based on models we use. This process may include removing special characters, stop words, stemming and lemmatization. For BERT related models, we used BERT tokenizer to transform language into ids and attention masks for BERT use.

2.3. Models:

We used models such as naive bayes, multilayer perceptron, LSTM, Transformer and BERT to create trading signals based on text data.

2.4. Signals/Strategy:

We will use a relatively simple trading strategy here. We will either hold cash or cryptocurrency based on the signals created by our machine learning models.

2.5. Back testing:

We will back test our strategy in the historical data to see if our strategy can beat the benchmark and find "alpha". Some metrics such as Sharpe ratio and maximum drawdown will be calculated.

3. Data

Firstly, we used the yahoo-finance API to retrieve data of BTC and ETH cryptocurrencies. The frequency of the price volume data is hourly.

Secondly, we used the news data from Alpha Vantage. The frequency of the news data is hourly as well. The news data is composed of four different parts. The first part is the time of the news items when happening; the second part is the title of the news; the third part is the content and the details of the news; the last part is the sentimental scores of each news item.

We also used the price percentage change as the classification labels. To be specific, 1 represents a positive return in the next hour; 0 represents a negative return in the next hour. To build a model, we just need the news text strings and the hourly return classification labels. There is an example of our data:

	text	class
2022-03-06 14:00:00	Terra Is Now DeFi's Network of Choice After Et...	negative
2022-03-07 11:00:00	Here's the Cryptocurrency That Ethereum Whales...	positive
2022-03-07 15:00:00	OTC: DRCR, Swifty Global (Dear Cashmere Hold...	positive
2022-03-07 22:00:00	BRISE token soars to new highs after the offic...	negative
2022-03-08 19:00:00	What Is A Crypto Airdrop?	negative

4. Model and Strategy:

We start with a relatively basic model - naive bayes and we will gradually increase the complexity of our models. We will include BERT models with either fully connected layers or LSTM as classifiers for downstream text classification tasks. We hope neural networks models can capture more complicated relationships between news text and stock performance.

4.1. Baseline model: Naive bayes

1) Model introduction

   This model is mainly composed of two parts. The first part is the data processing, and second part is multinomial naïve bayes classifier. The idea is to process the text using bag of words method and put word counting vector into multinomial naïve bayes classifier to predict the next hour's stock returns (positive or negative).

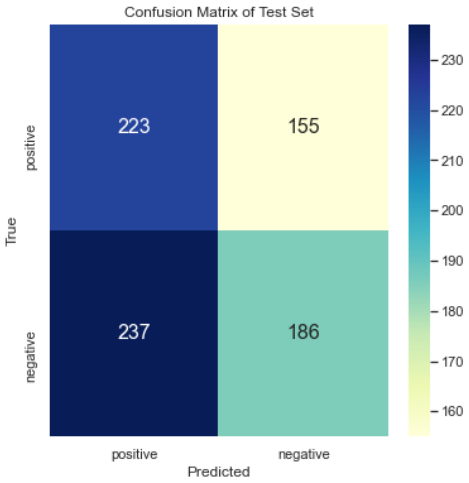
   Bag of Words: The bag of words method is to count the frequency of each word in a sentence and transform a sentence into a frequency vector. There will be vocabulary based on words in the sentences.

   Multinomial Naïve Bayes: This model is based on bayes formula and "naïve" assumptions about data distributions. Here we use the word counting vector as the input and next hour's stock returns (positive or negative) as output.

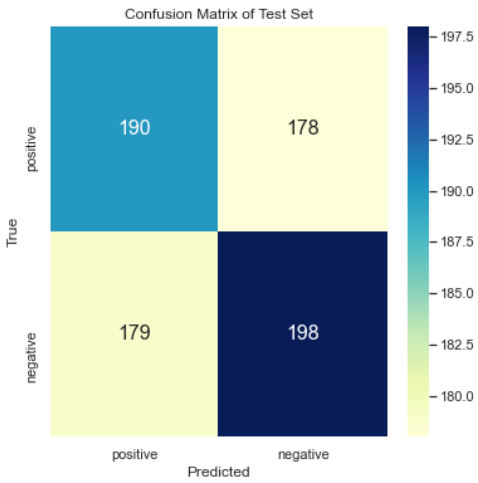
2) Confusion matrix

After we applied the model to predicting the hourly return direction of crypto currencies, we can look at the confusion matrix to see the performance. The following are the results of BTC and ETH as an example.

BTC



ETH



3) Back testing:

For back testing, we used the predicted signals of ‘positive’ to represent ‘long’ and ‘negative’ to represent ‘short’. And for each hour, we either all-in our asset which starts from 100% (under ‘positive’) or clear all our positions and hold non-crypto cash (under ‘negative’).

Then we can plot two net asset value curves, one for the simple long and hold strategy as benchmark, the other for our strategy.

The nav curves for BTC and ETH are as follows:

BTC

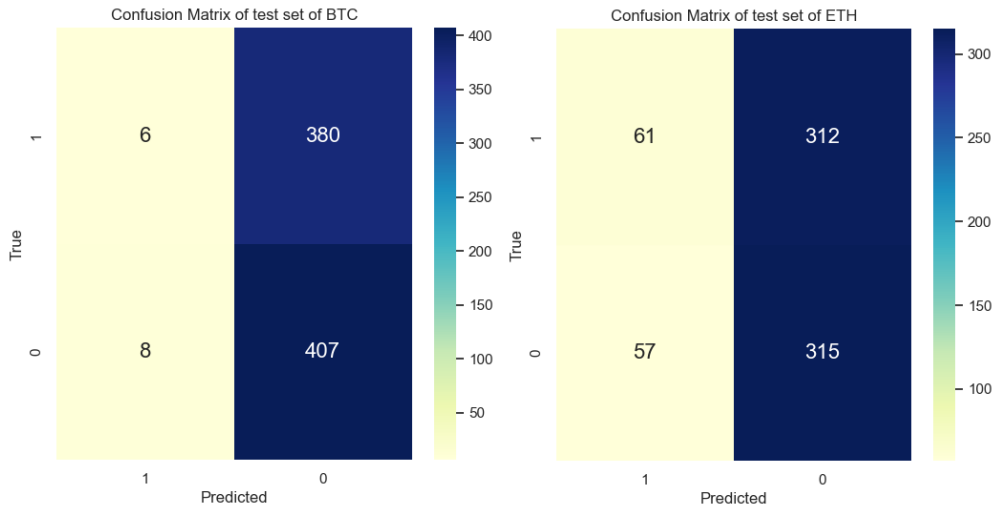
ETH

4) Results
   From both examples, our strategy outperformed the benchmark. For ETH, we can even get a positive cumulative return in a bearish market condition. We also calculated some back testing metric for further compare our strategy with the benchmark.

Although both ETH and BTC are in a bearish market condition, our strategy can achieve higher return (lower loss) and lower volatility compared with buy-and-hold strategy. For ETH, we are even able to get a positive return.

4.2. Embedding + Fully connected layer (sentence embedding)

1) Model introduction
   This model is mainly composed of two parts. The first part is the sentence embedding module which is used to convert each news data item into a 384-d vector and the second part is the fully connected layer which is further used to make classification using the sentence vectors.
   Sentence Embedding: The sentence embedding is based on Bert module. A typical method is to add a pooling layer after Bert to transfer the word embedding vectors to a single sentence embedding vector for each sentence. By using the sentence embedding method, each news item can be transferred as a vector to be fed into the classification model.
   Fully connected layer: This layer is a simple model to make a binary classification to predict the price for the next hour. To prevent overfitting, we add a dropout layer to provide some regularization.
2) Confusion matrix
   After we applied the model to predicting the hourly return direction of crypto currencies, we can look at the confusion matrix to see the performance. The following are the results of BTC and ETH as an example.



3) Back testing

For back testing, we continue to use the predicted signals of '1' to represent 'long' and '0' to represent 'short'. And for each hour, we either all-in our asset which starts from 100% (under '1') or clear all our positions and hold non-crypto cash (under '0').

Then we can plot two net asset value curves, one for the simple long and hold strategy as benchmark, the other for our strategy.

The nav curves for BTC and ETH are as follows:

4) Results

From both examples, we clearly figure out that our strategies can outperform the benchmarks, especially for ETH. We also calculated out some back testing metric for further compare our strategy with the benchmark.

When comparing the results, we find that both BTC and ETH are in a bearish market during the back testing period. However, both our strategies show better rate of return (or lower rate of loss), lower volatility, better Sharpe performance and lower maximum drawdown.

4.3.Embedding + LSTM + Fully Connected Layer (word embedding)

1) Model introduction

This model is to use the pretrained BERT model to get word embeddings of a sentence and use the word embeddings of sentences as the input of downstream classification model. The classifier model we use here has two stacked LSTM layers followed by two fully connected layers and sigmoid function to create probability output.

BERT Embedding: The word embedding is from the pretrained BERT model. For each word in a sentence, it will create a 1-dimension vector so we can transform each sentence into a 2-dimension representation.

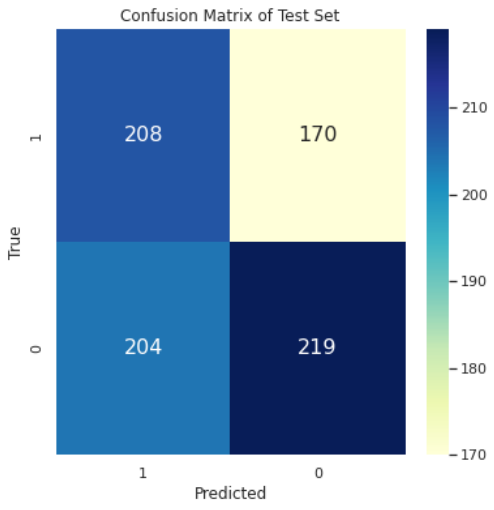
LSTM: LSTM is a standard RNN has both “long-term memory” and “short-term memory”. It can use series data to make predictions or produce hidden states.

Fully Connected Layer: Fully connected layer is the basic neural network type. It’s a linear function plus an activation function. We will use sigmoid function in the output layer to produce a probability.

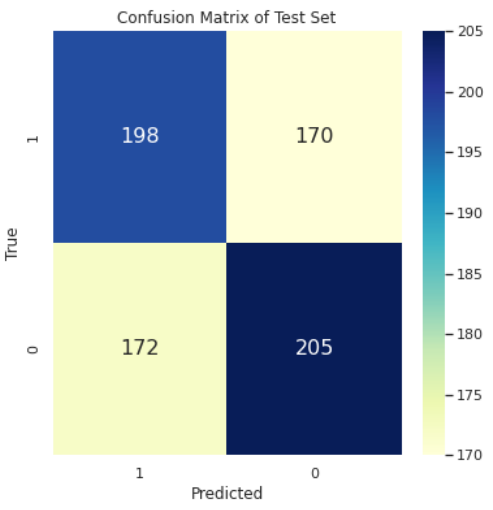
2) Confusion matrix

After we applied the model to predicting the hourly return direction of crypto currencies, we can look at the confusion matrix to see the performance. The following are the results of BTC and ETH as an example.

BTC



ETH



3) Back testing

For back testing, we used the predicted signals of '0' to represent 'long' and '1' to represent 'short'. And for each hour, we either all-in our asset which starts from 100% (under '1') or clear all our positions and hold non-crypto cash (under '0').

Then we can plot two net asset value curves, one for the simple long and hold strategy as benchmark, the other for our strategy.

The nav curves for BTC and ETH are as follows:

BTC

ETH

4) Results

From both examples, our strategy outperformed the benchmark. For both BTC and ETH, we can get positive cumulative returns in a bearish market condition. We also calculated some back testing metric for further compare our strategy with the benchmark.

Although both ETH and BTC are in a bearish market condition, our strategy can achieve higher return and lower volatility compared with buy-and-hold strategy. For BTC and ETH, we are even able to get positive returns. This model also performs better than our baseline naïve bayes model.

4.4.Embedding + Fully connected layer (sentence embedding + sentiment labels)

1) Model introduction

This model reproduced the same deep neural network framework as model 4.3: Embedding + LSTM + Fully Connected Layer but with different text preprocessing method and labels.

Since we also have sentiments i.e., Bearish, Somewhat-Bearish, Neutral etc., model training based on sentiments could also be expected. To calculate sentiment score for every 1-hour bar. We transformed sentiment to numerical scores such that Bearish -> -1, Somewhat-Bearish -> -0.5, Neutral -> 0, Somewhat-Bullish -> 0.5, Bullish -> 1.

Since we grabbed texts of one-hour frequency, each observation might have several news within an hour. Sentiment for each observation would be the average of those scores.

Structured data is shown as the following:

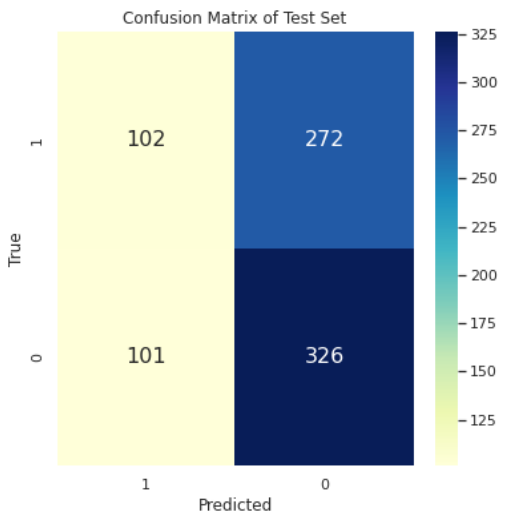
	text	sentiments	returns	class
2022-03-01 13:00:00	geopolitical risk returns global markets inves...	0.0	0.000856	0
2022-03-02 03:00:00	asian shares slip oil surges russia sanctions ...	0.0	0.004782	0
2022-03-02 08:00:00	business highlights lobbyists leaving rate hik...	-0.5	0.000519	0
2022-03-02 09:00:00	business highlights lobbyists leaving rate hik...	-0.5	0.000185	0
2022-03-02 11:00:00	millions crypto start ups real names necessary...	0.0	-0.000722	0

2) Confusion matrix

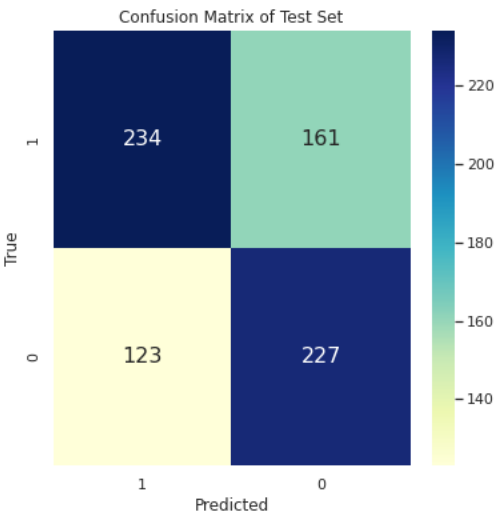
After we applied the model to predicting the hourly return direction of crypto currencies, we can look at the confusion matrix to see the performance.

The following are the results of BTC and ETH as an example. We can see that our model has a much better classification performance on ETH than BTC.

BTC



ETH

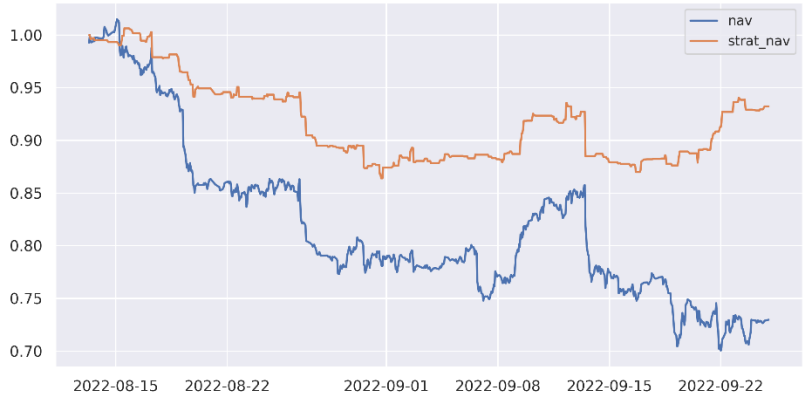


3) Back testing

For back testing, we continue to use the predicted signals of '1' to represent 'long' and '0' to represent 'short', since we consider 1 as positive sentiment whereas 0 corresponds to negative sentiments.

For each hour, we either all-in our asset which starts from 100% (under '1') or clear all our positions and hold non-crypto cash (under '0').

BTC



ETH

4) Results

From the above result we can see that our model outperformed the market using BTC, but failed to outperform the market using ETH. At least our model lowered the volatility.

Though this result is counterintuitive, since the model we trained based on ETH has a much higher generalization performance. From a technical point of view, since our training set and testing set are both within a bearish market, higher generalization abilities mean there would be lower False Negative and higher Recall, with more positive / long trading signals. This may lead to a higher loss within a bearish market.

5. Conclusion

5.1. Results comparisons:

Overall, for all strategies coming from different models, most of them outperformed the benchmark, and some of them even showed a positive return in the bearish back testing period. To figure out the source of the excessive return, we can investigate from the angle of the confusion matrix.

Firstly, we can see that in terms of accuracy, precision and recall, our strategies can get at least 2 of the 3 over 50%. That makes those strategies have an overall hit rate of more than 50% percent.

Secondly, some of the models have relatively much higher recall or much higher precision than others, making them good at catching bullish periods or avoiding bearish periods.

Lastly, since the back testing period is not long enough to cover a complete market cycle. We trained our model in the bearish market and tested it in the bearish market as well. This may let the models memorize some characteristics that are unique to recent market mode, making the results biased to the positive side.

5.2. Improvements:

1) Since the data we extracted started from Mar 2022, which was the start of the bear market of cryptocurrencies. Since the training data and testing data we used were all within the bear market, this might be the source of alpha of our four strategies.

   Technically, this involves the problem of our low recall scores. This means there are a lot of false negatives. Many supposed long trading signals were predicted as selling trading signals, since we were in the bear market, selling overall should always be better than buying.

   For further improvement, though, we can focus more on Recall during the bearish market whereas focus more on Precision during the Bullish market, to make our models have a much better generalization prediction abilities, we should use a more balanced training dataset, which can cover both bullish and bearish markets.

   Besides, since we used several deep learning models, more highly qualified data should be used to train models, so that the problem of ‘garbage-in-garbage-out’ could be avoided. Due to the limited time, more details should be considered when we develop our codes (design of layers using either PyTorch or Keras) to build deep neural networks.

2) Deep learning training and hyperparameter fine tuning requires a lot of GPU computing resources and we can’t iterate over all possible hyperparameter combinations and sometimes can’t guarantee a perfect training process. There is also some randomness in the training process and the randomness can affect the strategy performance and make the strategy unstable. We may get good back testing results on some parameters but still can’t guarantee those results will continue in the future.

Those problems still need to be solved through reading related papers about training optimization, fine tuning skills and trading strategy robustness analysis.

3) Our trading strategy is long-only and somehow aggressive. We didn’t include any hedging or long-short strategy. We can build more sophisticated trading or hedging strategies in the future.