A volatility arbitrage trading neural network model in the options market

Instructor: Bernardo Pereira Nunes

Student: Xin Yang, u7174459

Date: 2022.10.25

Table of Content:

Abstract

Volatility, an important indicator in the financial market, could be used in risk management and even making profit. In the options market, for each option contract, the implied volatility could be extracted by the options analysis. In this project, using stocks and options' volatility data and companies' conference earning calls (text + audio) data, we design several trading models, trying to predict the future implied volatility and capturing the arbitrage opportunities.

We observe that the BNN model is highly appropriate to such tasks in which the input data is with high level of noise, by capturing the uncertainty in the training data. We also realize the Sortino ratio, a less frequently used indicator in finance, is quite useful in improving models' decision-making ability as a loss function. Finally, we probably do the first try in the industry as far as we know to use the NLP data into the real trading practice of implied volatility arbitrage.

1. Introduction

Predicting financial risk is necessary for investment in the secondary market, which is already shown in previous researches (Engle, 1992). While the stock volatility, and option implied volatility are both the important indexes, useful in financial risk management and hedging (Muzzioli, 2010). Successfully predicting the option's implied volatility value and making correct corresponding trading decision will be our ultimate target.

There are many different data set and models could be used in the implied volatility prediction depending on the skills in use. For statistics researchers, the time series models such as the GRACH model is popular (Huang and Clive, 2005). For traditional researchers in finance, the binary tree and backwardation methods are commonly in used (Dean and Paxson, 2003).

In this project, the neural network model will be our focus. In 1996, traditional neural network model has already proven that it is useful in implied volatility prediction using the historical volatility data as the input (Malliaris and Salchenberger, 1996), while the Bayesian Neural Network (BNN) model is with the advantage in showing the confidence in prediction (Jang, Huisu, and Lee, 2017). In recent years, researchers begin to build up the NLP model in prediction, fed by the unstrctured data like text, and audio (Kogan et al., 2009).

The purpose of this research is to use different types of data to try building up a practical trading-decision-making model in the option market with real practice. Data may include companies' conference earning call (transcript and audio as the unstructured data) and the traditional historical volatility data in the stock and option market.

Our innovation in this project is mainly in two parts. The first one is to integrate the trading simulation regime as the back-test into the loss function design as a part of the model framework. We recognize the evaluation of the model's success is to predict correctly for samples with confidence rather than predicting every samples correctly, which is different from the typical machine learning task. The second one is to build up an innovative NLP model, Multi-modal Deep Trading-decision-making Model, making use of the structured and unstructured data in prediction.

This paper will be arranged as follows. In Section 2, the literature review for relevant neural network models (BNN and NLP models) in prediction will be done to understand the updates and results from previous researches. Section 3 will introduce the background knowledge needed like the option analysis and the trading regime design. Section 4 will discuss in details on data preprocessing, algorithms, loss functions, and models design with formulas. Section 5 will show the back-test results for these designed model for testing, and try to explain and discuss the result with rationales. The potential limits for models will be discussed in Section 6. Section 7 will be our conclusion, with the potential future work for further researchers interested in.

2. Related Work

2.1 The Bayesian Neural Network model in financial application

One of the biggest advantages of using the Bayesian Neural Network model is to estimate the noise and the uncertainty, which provides users with the output as the distribution rather than a single value (MacKay, 1996). The parameters in the models are the posterior probability function, originated from the prior probability distribution, updated by the training samples (Neal, 1996). When more data is available to feed the BNN model, the model may show its higher confidence when the test sample is similar to the training set (Freedman, 1963). The final output distribution maybe highly complex because of activation function's participation and complex posterior distribution for each parameters in the model, therefore the Monte Carlo method becomes useful. It use the sampling method to get the necessary information of the output distribution (MacKay, 1992).

Actually few studies are about using the BNN model in the finance research. (Gencay and Qi, 2001) shows that the BNN model is useful in the derivative securities' pricing and hedging. In the stock price prediction, it is also effective in forecasting before and after the Covid-19 pandemic (Chandra and He, 2021). In addition, in the crypto currency aspect, quite popular in recent years, the BNN model has been shown its outstanding performance in price prediction (Huisu and Lee, 2017). Our project may bring the BNN model into a new application, volatility prediction in the option market.

2.2 The NLP model in financial application

2.2.1 Plain text data is useful in prediction

Many previous researches try to use different data resources and NLP models to predict financial market performance.

Using the Support Vector Regression (SVR) Model is with the advantage of the human-interpretable result and the plain text in the companies annual report maybe useful. Clipping the paragraph of management's discussion and analysis from the companies' annual report (MD&A), Kogan used simple bags of unigrams and bigrams to represent the text in the MD&A as one of the inputs in the model. The result shows that the SVR model with text representation and past volatility history as the input outperforms the model with past volatility history as the input only (Kogan et al., 2009).

Similarly, the SVR model again but using financial-specific lexicon as seeds from (Loughran and McDonald, 2011), Tsai expanded keywords by calculating cosine-distance with the seed words and represented words by the Continuous Bag-of-Words (CBOW) model. The result shows that Keywords' Expansion can help make better stock volatility's predictions than

the original financial lexicon (Tsai and Wang, 2014).

Except the companies annual report we have shown above, other data sources like public news (e.g. Wall Street Journal) and social media (e.g. the post on the forum) are both shown useful in financial prediction by (Paul, 2007) and (Ding et al., 2015).

Though these researches use different plain text data source, the preprocessing methods they used in common are the text vectorization, the pre-trained words' embedding matrices are used commonly like Glove (Nicholas and Christopher, 2018; Ramit et al., 2020; Yu and Yi, 2019), Word2Vec (Hu et al., 2018), and fastText etc., with widespread application in peers' article.

Other mature models in deep learning are also applied in prediction.

The attention model with discriministic network is popular in recent years. it may give weight to the atom unit (words or sentence) in the article, and average them with weight as output (textual features). The purpose is to figure out the important sentences/words which may impact the stock volatility. Such Textual feature output will work with Financial feature together, using discriminative network, to determine the target output (Ma et al., 2020).

2.2.2 Audio data is also useful in prediction

It has been proven that the vocal feature extracted from the audio data could transmit additional information. (Jiang and Pell, 2017) found that the confidence and doubt in the speech could be captured. Speaker may change their confidence level in the speech when the pitch for sentences changes comparing with the global average. It also shows the potential useful vocal

indicators which could check speakers' confidence, including the fundamental frequency, amplitude, speech rate, voice quality, and the measures of pause and fluency.

The source of the audio data in stock price prediction is mainly the audio record of companies conference earning call. (Tonin and Scherer, 2022) uses more than 1000 samples for the conference earning calls in the Brazil stock market as training data, showing that different participants in the earning call conference with their diverse vocal features maybe able to predict the fluctuation of the stock price after the meetings. The research result of (Mayew and Venkatachalam, 2012) has similar statements. It states that companies' executives communication in the important meeting like the conference calls maybe helpful to quantitatively predict the stock price's fluctuation whenever their communication is with high quality.

2.2.3 Plain text + audio as the input for multi-modal models

The combination between plain text and audio provides multi-modal models' training with firm grounding.

The analysis based on the sentence level is done by (Yu and Yi, 2019). Their input is from the earning conference call, extracting the transcript in plain-text and audio features. They clip the whole transcript document and the audio as a series of sentences, which are the input for the Contextual BiLSTM model respectively for features extraction because the BiLSTM Model is good for dealing with information with sequence. The text and audio extraction features will merge by sentences consequently. Such merged feature is the input for the final BiLSTM Model as multimodal learning. The result shows that such the Multimodal Deep Regression Model based on sentence-level analysis with multi-features is relevant to the stock price prediction,

performing better than other multi-modal models including the bc-LSTM model by (Poria et al., 2017).

Multi-tasks ensemble model is another way to complete the multi-modal learning. The research done by (Sawhney et al,. 2020) uses the text and audio data from the conference earning call to build up a multi-task, multi-modal ensemble model to predict the stock price and volatility at the same time. The connection between these two tasks may could mitigate the overfitting problem.

Overall, the earning conference call as public information is convenient for acquisition. It is the main opportunity for the executives to communicate with the (potential) shareholders and many front edge researches already prove its potentials. Therefore, it will be included in our models as the input data.

3. Background knowledge

3.1 Implied volatility and the Greeks in the options

In option analysis, the implied volatility and the Greeks are important indicators. They are extended from the Black-Scholes formulas and the option valuation model (Black, Myron 1973).

We assume σ is the underlying stock's volatility. $\mathcal{K}$ is the strike price of the stock. r is the risk-free interest rate. In here, the reference risk-free interest rate we select is the 1-year Treasury Bill in the U.S. $\mathcal{S}$ is the spot price of the stock. t is the time until maturity. $\Psi(\cdot)$ is the cumulative distribution function

of the normal distribution, $\mathcal{p}$ is the price of the put option, c is the price of the call option. $\mathcal{C}$ is the pricing function of the call option. $\mathcal{P}$ is the pricing function of the put option.

$$\Psi(\varkappa) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\varkappa} e^{-\frac{\Gamma^2}{2}} \, d\Gamma \tag{1}$$

The prices for call and put option are the following:

$$\mathcal{C}(\mathcal{S}, \mathcal{K}, r, t, \sigma) = \mathcal{S}\Psi(d_1) - \mathcal{K}e^{-rt}\Psi(d_2) = c \tag{2}$$

$$\mathcal{P}(\mathcal{S}, \mathcal{K}, r, t, \sigma) = -\mathcal{S}\Psi(-d_1) + \mathcal{K}e^{-rt}\Psi(-d_2) = \mathcal{p} \tag{3}$$

$$\text{which } d_1 = \frac{\ln\frac{\mathcal{S}}{\mathcal{K}} + (r + \frac{1}{2}\sigma^2)t}{\sigma\sqrt{t}} \tag{4}$$

$$\text{which } d_2 = d_1 - \sigma\sqrt{t} \tag{5}$$

Based on the trading data and quoted price in the option market, we could get the value of all factors except the σ, the underlying stock's volatility, so we need to build up two equations to get implied volatility value for call and put options correspondingly.

$$\mathcal{f}(\sigma) = 0 \Leftrightarrow \mathcal{S}\Psi(d_1) - \mathcal{K}e^{-rt}\Psi(d_2) = c = 0 \tag{6}$$

Fortunately, such complex partial derivative equation could be solved by computer easily using numerical approximation.

The next part is the Greeks. It includes five elements, Delta, Gamma, Vega, Theta and Rho. In this application, we may pay more attention on Delta, Gamma, Vega and Theta.

Delta measures the impact of an unit of underlying stock price change to the option price. The following is the formula:

$$\text{Delta} = \mathcal{C}(\mathcal{S} + \Delta\, s, \mathcal{K}, r, t, \sigma) - \mathcal{C}(\mathcal{S} - \Delta\, s, \mathcal{K}, r, t, \sigma) \,/\, 2\,\Delta\, s \qquad (7)$$

Gamma measures the impact of a unit of underlying stock price change on the Delta value. The following is the formula:

$$\text{Gamma} = (\mathcal{C}(\mathcal{S} + \Delta\, s, \mathcal{K}, r, t, \sigma) + \mathcal{C}(\mathcal{S} - \Delta\, s, \mathcal{K}, r, t, \sigma) - 2\mathcal{C}(\mathcal{S}, \mathcal{K}, r, t, \sigma))/ (\Delta\, s)^2 \qquad (8)$$

Vega measures the change of 1% underlying stock price's implied volatility's increase to the option price. The following is the formula:

$$\text{Vega} = \mathcal{C}(\mathcal{S}, \mathcal{K}, r, t, \sigma + 1\%) - \mathcal{C}(\mathcal{S}, \mathcal{K}, r, t, \sigma - 1\%) \,/\, 2 \qquad (9)$$

Theta measures the impact of the change of maturity (1 day) to the theoretical option price. The following is the formula:

$$\text{Theta} = \mathcal{C}\left(\mathcal{S}, \mathcal{K}, r, t - \frac{1}{365}, \sigma\right) - \mathcal{C}(\mathcal{S}, \mathcal{K}, r, t, \sigma) \qquad (10)$$

Formulas for the put option are similar by changing $\mathcal{C}$ to $\mathcal{P}$, and we skip these formulas for avoiding repetition.

3.2 Trading Regime / Rules

The fundamental for a well-learned model is a good trading regime design. The intra-day trading is our trading regime. The information from open price (stock and option), and unstructured data (audio and text) may be the input for the decision model and the output is the trading decision. All positions and exposures will be closed at the end of the trading day.

Trading for price, volatility, and term structure are all available in the option market. **In this project, our trading regime is based on volatility.**

3.2.1 Informal trading regime: Long / Short Straddle

Long / Short Straddle is a common approach to trade for the volatility, which means to buy / sell the same amount of call and put options at the same time. It is appropriate to hold the Long Straddle position when we believe that the stock price may fluctuate sharply or investors may hold a higher expectation for severe stock price fluctuation. They are related to **realized volatility** and **implied volatility** respectively. The realized volatility means the price difference between the strike price and the stock price at maturity. If the gap is large enough, the profit will be made by the long straddle but the loss by the short straddle.

The change of **implied volatility** also leads to the profit and loss. Assuming all other factors are unchanged except implied volatility, and the option price change due to implied volatility linear (actually not), the profit and loss of long straddle is based on the formula:

$$\text{Return} \approx \text{Vega}_{\text{call}} \times \Delta\,\sigma_{\text{call}} + \text{Vega}_{\text{put}} \times \Delta\,\sigma_{\text{put}} \qquad (11)$$

3.2.2 Formal trading regime: Delta Hedge

Straddle as trading regime as an inspiration, but it still has a big problem making the model uneasy to learn well. As shown in the Straddle strategy, the realized volatility due to the stock price change will change our profit and loss result, which is undesirable because the stock price prediction is almost impossible and it may cause noise to the training samples. To get rid of the noise due to stock price movement, **Delta Hedge** is what we need.

Delta Hedge is a trading regime to make the option portfolio's Delta = 0, based on the following formula:

$$\text{Delta}_{\text{portfolio}} = 0$$

$$\Leftrightarrow \text{Delta}_{\text{call}} \times \text{position}_{\text{call}} + \text{Delta}_{\text{put}} \times \text{position}_{\text{put}} = 0 \qquad (12)$$

Recalling the definition of Delta, it means the impact of underlying stock price change to the option price. Therefore, a portfolio with Delta = 0 stands for the unchanged portfolio value when the stock price changes. We assume such relationship statically holds (actually it is dynamic with changing price). In a trading day, the factor with uncertainty leading to the change of portfolio value is the **implied volatility** only. **Long / Short such portfolio stands for long / short implied volatility.**

Therefore, the final output of the model is from -1 to 1 as the trading decision (**Short / Long portfolio**). '-1' stands for short a unit of this portfolio (the implied volatility), while '1' stands for long a unit of this portfolio (the implied volatility). '0.1' stands for long the implied volatility with uncertainty, so the model only buys 0.1 units of this portfolio. For each trading, the maximum

position could be only as much as 10% of the total asset (so '0.5' stands for 5% position), because the trading loss > 100% position holding is quite common in the option market.

3.3 Trading regime evaluation

Several evaluation indicators imported from the financial portfolio management will be introduced and they may be used in the loss function design. They are the **average return rate for each trading**, **Sharpe Ratio** and **Sortino Ratio**. These indicators show that evaluating a trading regime is complex. The best trading regime maybe not the one earning the most.

3.3.1 Evaluation indicator 1: Average return rate for each trades

This loss function is to calculate the mean return rate for each trades. It aims to make the model maximize the expected return rate for each trades, even though its return may be more unstable. Actually the trading evaluation is not based on the return rate only, but the combination of return and risk.

3.3.2 Evaluation indicator 2: Sharpe Ratio

In 1966, Economist William Sharpe created the Sharpe Ratio for the trading evaluation (William F, 1998). In the following 50 years until now, it is one of the most widely used trading evaluation indicators all around the world .

The main idea of Sharpe Ratio is clear that the additional risk needs to be compensated by the additional return. $\mathcal{T} = 250$ in here because return samples in our portfolio are calculated daily and the annual Sharpe Ratio is what we want. Here is the formula:

$$\text{Sharpe Ratio} = \frac{\mathrm{E}(\mathrm{Return}_{\mathrm{Portfolio}} - \mathrm{Return}_{\mathrm{Risk\ free\ rate}})}{\sigma_{\mathrm{Portfolio}}} \times \sqrt{\mathcal{T}} \qquad (13)$$

As a loss function in the model, it aims to make the model learn how to get modest but stable return, because risk and return could be enlarged simultaneously by using the leverage.

3.3.3 Evaluation indicator 3: Sortino Ratio

The problem for the Sharpe Ratio is also obvious. Trades with positive return higher than the mean may also lead to high standard deviation, causing lower Sharpe Ratio. As supplement, in 1970s, economist Frank Sortino created Sortino Ratio, with similar format to the Sharpe Ratio but only focusing on standard deviation of the downside (Rollinger et al., 2013). Here is the formula:

$$\text{Sortino Ratio} = \frac{\mathrm{E}(\mathrm{Return}_{\mathrm{Portfolio}} - \mathrm{Return}_{\mathrm{Risk\ free\ rate}})}{\sigma_{\mathrm{downside}}} \times \sqrt{\mathcal{T}} \qquad (14)$$

Actually it is not widely used in practice because its implication is not as much intuitive as the Sharpe Ratio. However, it seems to be a good principle for the machine to learn, and we may discuss it carefully in result & discussion.

4. Methods

Methods in general are in five steps. Raw data collection, preprocessing of the raw data, loss function design, model design, hyperparameter search to find the best model after fine-tuning and finally the back-test.

Overall, our ultimate goal of this project is to create a trading decision model in the U.S stock options market. The decision model may determine whether the opportunity of volatility arbitrage exists. To ensure the market is with high liquidity and is not under manipulation, we only focus on the stocks/options in the list of the Standard & Poors' 500 (Dhillon et al., 1991).

4.1 Data collection

Two different parts of data will be collected in this project, the structured traditional historical stock and option trading data and the unstructured companies' earning conference calls data (transcript and audio).

4.1.1 NLP data collection:

With the approval of the Seeking Alpha website[1], we collect more than 20,000 JSON files for S&P 500 companies' seasonal press conference transcripts corresponding with more than 5,000 audio files. The raw data for a press conference includes the complete audio and high-quality transcripts split by sentences, with additional information including the speaker and the type of sentence (presentation or Q&A[1]).

The following picture shows the transcript of the Apple company's conference earning call in 2018Q4. The transcript is a list in the JSON file and there are 4 properties for each elements in the list. The number represents the sequence number of the sentence in the whole document. The name represents the speaker. The characters means the sentence is from the presentation or Q&A part. The last element is the transcript.

---

[1] www.seekingalpha.com

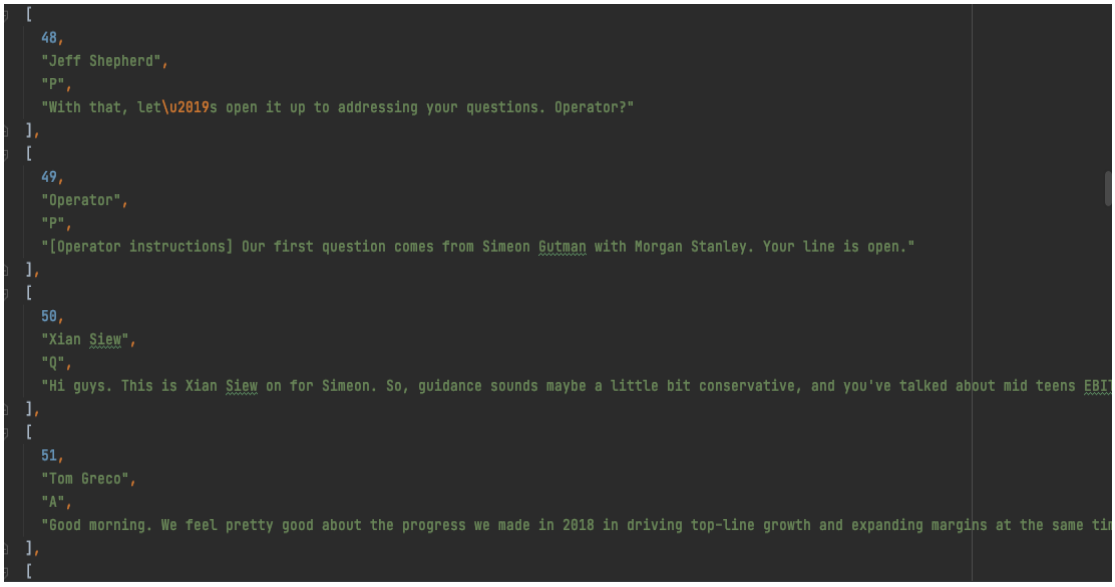
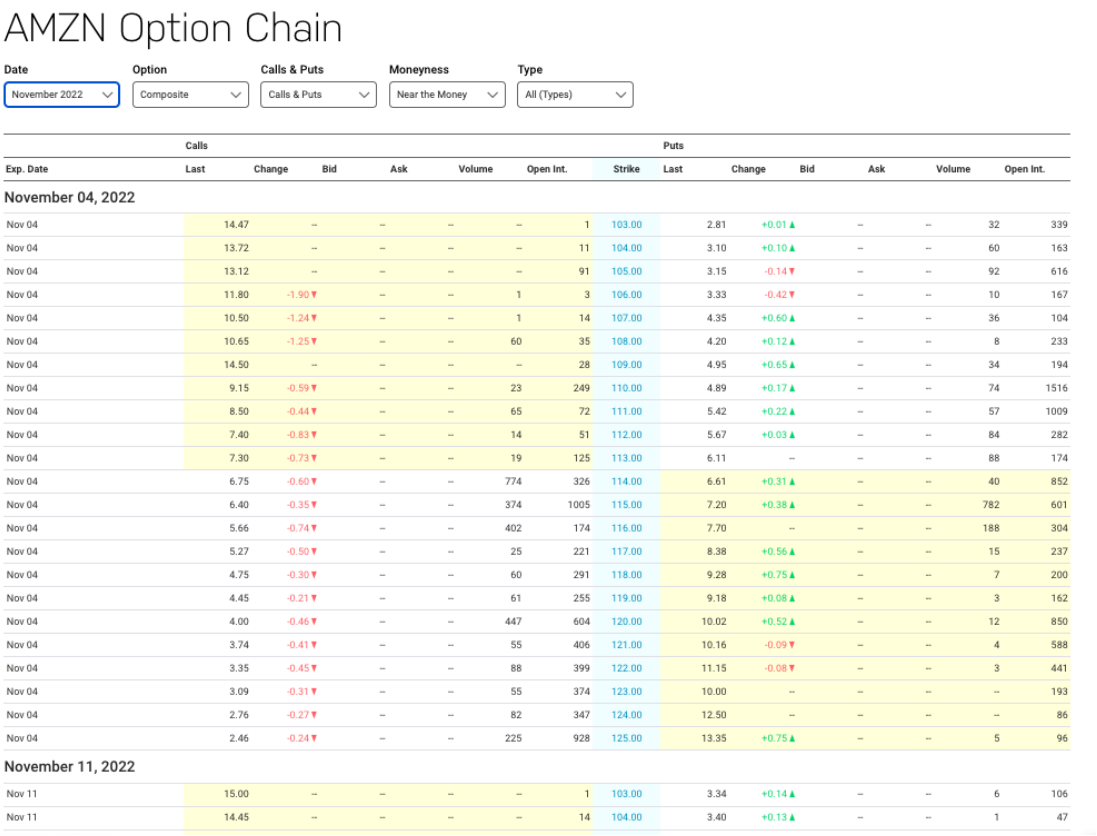


Figure 1 The transcript of conference earning call for Apple company in 2018Q4

4.1.2 Stock price and option price data collection:

To match the same time-line of the NLP data, we collect stocks and options trading data of the next closest trading date and the previous month's historical data. Data includes stocks' daily open price, closed price, trading volume, option's daily open price, and closed price.

For options' data, We select the closest monthly call and put contract (the closest date of the third Friday in the month as the maturity date), with the strike price closest to the stock price, because this contract stands for high liquidity and undistorted implied volatility.



AMZN Option Chain

Date: November 2022 | Option: Composite | Calls & Puts: Calls & Puts | Moneyness: Near the Money | Type: All (Types)

	Calls							Puts						
Exp. Date	Last	Change	Bid	Ask	Volume	Open Int.	Strike	Last	Change	Bid	Ask	Volume	Open Int.
November 04, 2022													
Nov 04	14.47	–	–	–	–	1	103.00	2.81	+0.01 ▲	–	–	32	339
Nov 04	13.72	–	–	–	–	11	104.00	3.10	+0.10 ▲	–	–	60	163
Nov 04	13.12	–	–	–	–	91	105.00	3.15	-0.14 ▼	–	–	92	616
Nov 04	11.80	-1.90 ▼	–	–	1	3	106.00	3.33	-0.42 ▼	–	–	10	167
Nov 04	10.50	-1.24 ▼	–	–	1	14	107.00	4.35	+0.60 ▲	–	–	36	104
Nov 04	10.65	-1.25 ▼	–	–	60	35	108.00	4.20	+0.12 ▲	–	–	8	233
Nov 04	14.50	–	–	–	–	28	109.00	4.95	+0.65 ▲	–	–	34	194
Nov 04	9.15	-0.59 ▼	–	–	23	249	110.00	4.89	+0.17 ▲	–	–	74	1516
Nov 04	8.50	-0.44 ▼	–	–	65	72	111.00	5.42	+0.22 ▲	–	–	57	1009
Nov 04	7.40	-0.83 ▼	–	–	14	51	112.00	5.67	+0.03 ▲	–	–	84	282
Nov 04	7.30	-0.73 ▼	–	–	19	125	113.00	6.11	–	–	–	88	174
Nov 04	6.75	-0.60 ▼	–	–	774	326	114.00	6.61	+0.31 ▲	–	–	40	852
Nov 04	6.40	-0.35 ▼	–	–	374	1005	115.00	7.20	+0.38 ▲	–	–	782	601
Nov 04	5.66	-0.74 ▼	–	–	402	174	116.00	7.70	–	–	–	188	304
Nov 04	5.27	-0.50 ▼	–	–	25	221	117.00	8.38	+0.56 ▲	–	–	15	237
Nov 04	4.75	-0.30 ▼	–	–	60	291	118.00	9.28	+0.75 ▲	–	–	7	200
Nov 04	4.45	-0.21 ▼	–	–	61	255	119.00	9.18	+0.08 ▲	–	–	3	162
Nov 04	4.00	-0.46 ▼	–	–	447	604	120.00	10.02	+0.52 ▲	–	–	12	850
Nov 04	3.74	-0.41 ▼	–	–	55	406	121.00	10.16	-0.09 ▼	–	–	4	588
Nov 04	3.35	-0.45 ▼	–	–	88	399	122.00	11.15	-0.08 ▼	–	–	3	441
Nov 04	3.09	-0.31 ▼	–	–	55	374	123.00	10.00	–	–	–	–	193
Nov 04	2.76	-0.27 ▼	–	–	82	347	124.00	12.50	–	–	–	–	86
Nov 04	2.46	-0.24 ▼	–	–	225	928	125.00	13.35	+0.75 ▲	–	–	5	96
November 11, 2022													
Nov 11	15.00	–	–	–	–	1	103.00	3.34	+0.14 ▲	–	–	6	106
Nov 11	14.45	–	–	–	–	14	104.00	3.40	+0.13 ▲	–	–	1	47

Figure 2 An example of different quoted prices for the Amazon

4.2 Preprocessing of data

4.2.1 NLP data preprocessing

Three different features as different modalities are extracted from these NLP data, including the text semantic feature, the vocal feature and the word embeddings feature. These three features are linked to each other because they are from the same sentence.

For the **transcript** consisting of a group of sentences, tokenizer is used to break the sentence as a list of tokens. Stopwords are removed to improve each sentence's information content. Then, two tools are used in here.

The first one is the financial-specific lexicon from McDonald (Loughran and

McDonald, 2011). It classifies words with special meanings into eight different semantics. Words in each sentence included in the lexicon will be counted by their semantics accordingly and summed up. The 8-dimensional vector of semantics for each sentence is the first modality output.

The second one is the Glove300 (Manning et al., 2014), a famous word embedding tool. It vectorizes words as word representation (Jeffery, 2014). Its regime is based on aggregated from word to word co-occurrence statistics. Each sentence may sum up and average the vector of each words. The 300-dimensional vector of sentence representations is the second modality output.

For raw **audio** files, there are two steps in total. The first step is to clip the audio file into slices to match the transcript of sentences. The package 'Anenas' is used in here with forced-alignment algorithm (Moreno et al., 1998). The second step is to extract the vocal features. The mean, variance and maximum range features in pitch, intensity and (Harmonic-to-noise-ratio) HNR are recorded respectively. Five features of Jitter and six features of Shimmer after PCA (Principle Component Analysis) are also recorded (Pettarin, 2017). Overall, there exist 20 vocal features in total for each sentence, therefore the 20-dimensional vector as vocal representation is the third modality output.

4.2.2 Stock/Option data preprocessing

For **stock** data, two types of data are acquired after preprocessing calculation. The first one is the fluctuation of price change based on the open price and closed price on the last trading day. The second one is the previous month's stock trading data, calculating its previous month's volatility shown in the annual volatility format. We assume the set of observed samples is $\mathcal{X}$,

with the number of elements $\mathbb{N}$. x_j and $\bar{x}$ stand for the j-th sample and the mean value of the set $\mathcal{X}$ respectively. Assume the total number of trading days in a year $\mathcal{T} = 250$, and then we get the volatility. The following is the formula.

$$\text{Volatility} = \sqrt{\frac{\sum_{j=1}^{\mathbb{N}} (x_j - \bar{x})^2}{\mathbb{N} - 1}} \times \mathcal{T} \tag{15}$$

For **options** data, we are more interested in its implied volatility and the Greeks. The calculation in details checks Section 3.1.

4.3 Loss functions' design

The loss function design is possibly the most important part in this project, because it determines what kinds of trading strategy you want for your neural network model to learn. Most of previous studies are used the common Mean Square Error (MSE) as the loss function in volatility prediction (Yu and Yi, 2019). Actually our application output is a little bit different with '**prediction**', but '**trading decision**'. Its difference is just like '**theorist**' and '**realizer**'. The model as the trading decision-maker, it is not necessary for the model to make trading decision all the time. Even the most intelligent investor would not have capacity to predict the result of every unknown trades. The noise in some trades makes such training samples messy, with no learning value, causing the machine performing badly due to learning from such useless samples.

For our special condition, I customize three loss functions as candidates for testing here. **Average return rate for each trading**, **Sharpe Ratio** and **Sortino Ratio**. The first one is quite common as our benchmark, to maximize

the return of each trading. The latter two of them are from the financial portfolio management and evaluation. Details is in the Section 3.3.

4.4 Model building

4.4.1 Notations

Before the model introduction, the notations will be introduced. The character in bold stands for a group of samples or a vector. Let $\mathcal{N}$ stands for the total number of samples and $\mathcal{M}$ is the number of sentences for the longest transcript in the sample set. For each sample, $\boldsymbol{\mathcal{D}}_j$ stands for the multi-features' data for the sample j and $1 < j < \mathcal{N}$. $\boldsymbol{\mathcal{D}}_j = [\boldsymbol{\mathcal{V}}_j, \boldsymbol{\mathcal{S}}_j, \boldsymbol{\mathcal{T}}_j, \boldsymbol{\mathcal{A}}_j]$, which means different modalities data for sample j. $\boldsymbol{\mathcal{V}}_j$ is a $1 \times v$ vector to represent the volatility data. $\boldsymbol{\mathcal{S}}_j$ is a $\mathcal{M} \times ds$ matrix to represent the financial semantics in each sentence, and ds represents the dimension of semantics features. $\boldsymbol{\mathcal{T}}_j$ is a $\mathcal{M} \times dt$ matrix to represent the word's embedding in each sentence and dt represents the dimension of embedding features. $\boldsymbol{\mathcal{A}}_j$ is a $\mathcal{M} \times da$ matrix to represent the vocal features in each sentence and da represents the dimension of vocal features.

4.4.2 Models:

(a) Basic Model:

The basic neural network model is our benchmark model, using 2 dense layers, with the activation function of 'tanh', to ensure the output is from -1 to 1. It uses the uni-modal data source $\boldsymbol{V}$ only.

(b) Alternative Uni-modal model: Bayesian Neural Network (BNN) Model

Bayesian Neural Network (BNN) is based on probabilistic implementation. Comparing with the Normal Neural Network, the main difference is that parameters in the layers are a distribution rather than the single value. Therefore, the output of the prediction in the BNN model is a random value in the posterior distribution (Carlin, 1997; Liang, 2005).

In the real implementation for this project, we follow the benchmark model's format, using the model structure of 2 layers, but what we use is the Dense-variational Layer rather than the Dense layer. In addition, the definition of prior distribution and posterior distribution are needed. We assume the prior distribution of parameters in the layer is the standard multivariate normal distribution with $\boldsymbol{\mathcal{k}}$ standing for the set of trainable parameters in the dense-variation layer. The following is the formula, where the small $\mathcal{k}$ stands for the dimension of the hidden layer. $\boldsymbol{\mu_0}$ and $\boldsymbol{\Sigma_0}$ are fixed.

$$\boldsymbol{\mathcal{P}}_{\text{prior}}(\boldsymbol{\mathcal{k}}) = (2\pi)^{-\frac{\mathcal{k}}{2}} \det(\boldsymbol{\Sigma_0})^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\boldsymbol{\mathcal{k}} - \boldsymbol{\mu_0})^{\mathrm{T}} \boldsymbol{\Sigma_0}^{-1} (\boldsymbol{\mathcal{k}} - \boldsymbol{\mu_0})\right) \qquad (16)$$

The following the is formula for posterior probability density function (P.D.F.). We assume that the posterior P.D.F. follows the multivariate normal distribution and n samples $\{\boldsymbol{\mathcal{k}_1}, \boldsymbol{\mathcal{k}_2}, \ldots, \boldsymbol{\mathcal{k}_n}\}$ after optimization are given:

$$\boldsymbol{\mathcal{P}}_{\text{posterior}}(\boldsymbol{\mathcal{k}} \mid \boldsymbol{\mu_n}, \boldsymbol{\Sigma_n}) = (2\pi)^{-\frac{\mathcal{k}}{2}} \det(\boldsymbol{\Sigma_n})^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\boldsymbol{\mathcal{k}} - \boldsymbol{\mu_n})^{\mathrm{T}} \boldsymbol{\Sigma_n}^{-1} (\boldsymbol{\mathcal{k}} - \boldsymbol{\mu_n})\right) \qquad (17)$$

So when the new sample $\boldsymbol{k}_{n+1}$ is given, the updated process is the following formulas:

$$\boldsymbol{\mu_{n+1}} = \boldsymbol{\Sigma_0}(\boldsymbol{\Sigma_0} + \frac{1}{n}\boldsymbol{\Sigma})^{-1}\,(\frac{1}{n}\sum_{i=1}^{n}\boldsymbol{k}_i) \;+\; \frac{1}{n}\boldsymbol{\Sigma}(\boldsymbol{\Sigma_0} + \frac{1}{n}\boldsymbol{\Sigma})^{-1}\,\boldsymbol{\mu_0} \tag{18}$$

$$\boldsymbol{\Sigma_{n+1}} = \boldsymbol{\Sigma_0}(\boldsymbol{\Sigma_0} + \frac{1}{n}\boldsymbol{\Sigma})^{-1}\frac{1}{n}\boldsymbol{\Sigma} \tag{19}$$

The same with the benchmark model, it uses the uni-modal data source $\boldsymbol{\mathcal{V}}$ only.

(c) Multi-modal model: Multi-modal Deep Trading-decision-making Model

Our Multi-modal Deep Trading-decision-making Model consists of three components with 5 times' training in total. The first component of model is to make use of data source $\boldsymbol{\mathcal{V}}$ to make a basic trading decision. The second component of model is to use the uni-modal NLP data source only ($\boldsymbol{\mathcal{S}}, \boldsymbol{\mathcal{T}}, \boldsymbol{\mathcal{A}}$) as input, to feed the Modified Contextual BiLSTM model (Poria et al., 2017) to generate its new trading decision, after combining the trading decision from basic model. The last component is a multi-modal model, using the transfer-learning skill at the same time. It extracts the hidden layer features for these 3 uni-modal separately using the model in the second part, and then concatenate them together as input, doing the training for the Modified Contextual BiLSTM Model again. The details will be given in the following.

Modified Contextual BiLSTM Model:

Our multi-modal model is with the modification based on the original multi-modal model: *Contextual LSTM* by (Poria et al.,2017). The main purpose

of this model is to classify the emotion of speakers for each sentence when they speak, using the data of text, audio and video as the multi-modal input. The original model consists of a LSTM layer, connecting with a time-distributed dense layer before the softmax output layer.

Due to the different application, what we want actually is based on the summary of the whole transcript / audio to make the final decision, therefore an Attention layer is also introduced after the Contectual BiLSTM Model as information summarization. Moreover, we also replace the BiLSTM layer to the original LSTM layer, which is with the advantage of its bi-directional broadcast (Hochreiter and Schmidhuber, 1997), an improved layer based on the model of recurrent neural network (RNN).

We give the details how to build the Contextual BiLSTM model in the following formulas:

$$g_i = \sigma(\mathcal{W}_g x_i + \mathcal{U}_g hidden_{i-1} + \mathcal{b}_i) \tag{20}$$

$$n_i = \sigma(\mathcal{W}_n x_i + \mathcal{U}_n \mathrm{hidden}_{i-1} + \mathcal{b}_n) \tag{21}$$

$$u_i = \sigma(\mathcal{W}_u x_i + \mathcal{U}_u \mathrm{hidden}_{i-1} + \mathcal{b}_u) \tag{22}$$

$$cell_i = g_i \cdot \mathrm{cell}_{i-1} + n_i \cdot \sigma(\mathcal{W}_c x_i + \mathcal{U}_c \mathrm{hidden}_{i-1} + \mathcal{b}_c) \tag{23}$$

$$\mathrm{hidden}_i = u_i \cdot \sigma(cell_i) \tag{24}$$

$$Output_i = ReLU(\mathcal{W}_{output} hidden_i + \mathcal{b}_{\mathrm{output}}) \tag{25}$$

From formulas 20 to 25, i means the i-th sentence for all over $\mathcal{M}$ sentences in the sample. Correspondingly, g_i, n_i, u_i mean the forget gate, input gate

and output gate for the i-th sentence respectively. x_{i} means the feature vector (semantic / embedding / vocal features) for the i-th sentence in the sample. $\mathcal{W}$ and $\mathcal{b}$ stand for the trainable vectors. $hidden_{\mathrm{i}}$ and $\mathrm{cell}_{\mathrm{i}}$ stand for the hidden state and cell state for the i-th sentence. Finally, the $\mathrm{Output}_{\mathrm{i}}$ means the final output of the Contextual BiLSTM Model from the time-distributed dense layer.

In our model, the Contextual BiLSTM Model training is completed 4 times in total. The first three times are for the uni-modal model performance checking and to produce reliable modal features in hidden layer as the input for the multi-modal model. The framework is in the following picture:

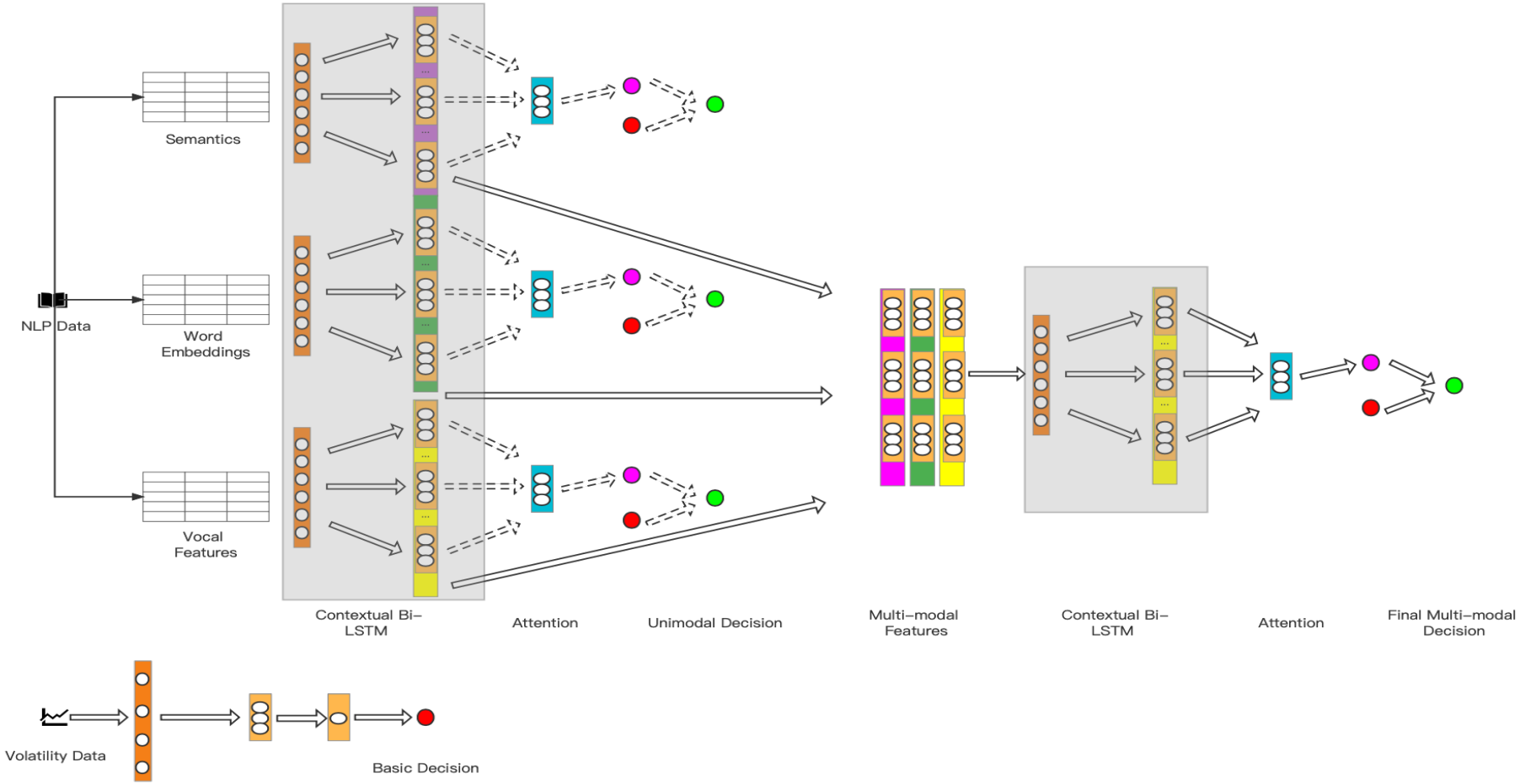


Figure 3 The Multi-modal Deep Trading-decision-making Model framework

The complete model regime starts from the feature extraction and vectorization which already shown in the Section 4.1. The number of sentences for each sample smaller than $\mathcal{M}$ will be masked by -1. Three different features $\boldsymbol{\mathcal{S}}$, $\boldsymbol{\mathcal{T}}$, $\boldsymbol{\mathcal{A}}$ as the uni-modal input to be trained separately in the Contextual BiLSTM + Attention model, with dimensions $d\mathcal{s}, d\mathcal{t}, d\mathcal{a}$ equal to 8, 300 and 20 respectively. Trading decisions $\boldsymbol{\mathcal{D}_S}$, $\boldsymbol{\mathcal{D}_T}$, $\boldsymbol{\mathcal{D}_A}$ will concatenate with the benchmark decision $\boldsymbol{\mathcal{D}_V}$ respectively as the input for the final dense layer to generate final decision $\boldsymbol{\mathcal{D}_{S_F}}$, $\boldsymbol{\mathcal{D}_{T_F}}$, $\boldsymbol{\mathcal{D}_{A_F}}$. The hidden layer's output from the Contextual BiLSTM Model $\boldsymbol{Output_S}$, $\boldsymbol{Output_T}$, $\boldsymbol{Output_A}$ (from Formula 25) will concatenate together as the multi-modal input, to complete the above training again, with the final trading decision $\boldsymbol{\mathcal{D}_{Multi_F}}$. Transfer learning thinking is used here while the uni-modal models' training makes features more representative.

4.5 Backtest

The backtest is our last step to check the quality of trading decisions. The initial value before the backtest is assumed to be 1. Our trading decision is based on the delta-hedge principle (Section 3.2 Formula 12). The maximum principal for each trading is 10% of our asset value (when $\mathcal{D} = 1 \text{ or } -1$), consisting of long / short $\text{position}_{\text{call}}$ and $\text{position}_{\text{put}}$ to make Formula 12 valid. The return rate depends on the trading decision and the difference between open and close price, with the following formula:

$$\text{Return Rate} = \frac{\text{position}_{\text{call}} \ \Delta \text{ Call Price} + \text{position}_{\text{put}} \ \Delta \text{ Put Price}}{\text{position}_{\text{put}} \cdot \text{Open Put Price} + \text{position}_{\text{call}} \cdot \text{Open Call Price}} \cdot \mathcal{D} - |\mathcal{D}| \cdot 0.5\% \quad (26)$$

The 0.5% of trading fee for each trades is taken into consideration.

4.5.1 Special setting for BNN model

For each single trade, actually the trading decision for the BNN model is a complex distribution rather than a single value. The Monte-Carlo method is used here. The simple BNN model will output a random value from the distribution based on their probabilities, while the BNN model with filter will make 100 repeated decisions from the distribution to get the decision set $\mathfrak{S}$. The mean $m_{\mathfrak{S}}$ and standard deviation $\mathrm{std}_{\mathfrak{S}}$ will be calculated. $\frac{|m_{\mathfrak{S}}|}{\mathrm{std}_{\mathfrak{S}}}$ is used to check the confidence for each trades. For example, if $\frac{|m_{\mathfrak{S}}|}{\mathrm{std}_{\mathfrak{S}}} = 2$, it means that in the range of 2 standard deviation, the trading system will keep the same trading direction, so the system is somehow confident for this trade. We may do the back-test and compare the result for the BNN model implementing the trading decisions of different confidence levels (e.g. 1-std, 2-std,...) .

5. Result and discussion

5.1 Result 1: Which loss function and model using traditional volatility data perform best?

The result in detail is shown in Figure 3 and Table 1.

Performance could be evaluated in different aspects. We do not use the total final return as the indicator to evaluate the model performance, because it does not consider the corresponding risk. A high-return model with high risk

actually is worse than a mid return model with low risk. Therefore, the indicators considering both the return and risk such as the Sharpe ratio and the Sortino ratio will be in our discussion.

a) Sharpe Ratio: The best performance indicator could be the one with highest Sharpe Ratio, because it stands for the highest return compensation for an unit of the risk. It shows that the Bayesian Neural Network model with filter (Only implementing the trades with at least 1-std confidence), the loss function equal to the simple average return rate for each trades, is with highest Sharpe Ratio, 5.192. This result surprises us because its Sharpe Ratio is even higher than the one using the Sharpe Ratio itself as the loss function.

The reason is probably that several single trades with high loss lower the average return rate, leading to more conservative trading decision making system.

b) Sortino Ratio: Sortino Ratio could also be recognized as the indicator of the best performance, because it stands for the highest return compensation for a unit of the downside risk. It shows that the BNN model with filter, combining with the Sortino ratio as the loss function performs best, with the value of 9.770.

In real practice, it maybe the most useful indicator to define the best trading model, because it represents the aggressive return and the minimal withdraw at the same time. The trading result for lines with red/ purple / brown colors in the Figure 4 could prove this result. The total return is highest using the Sortino ratio as the loss function for all models.

In conclusion, the combo (Model: BNN with filter (1-std) + Loss function:

Sortino ratio) maybe the most practical in real trading practice. Our next result will focus on the fine tuning of the filter in the BNN model.

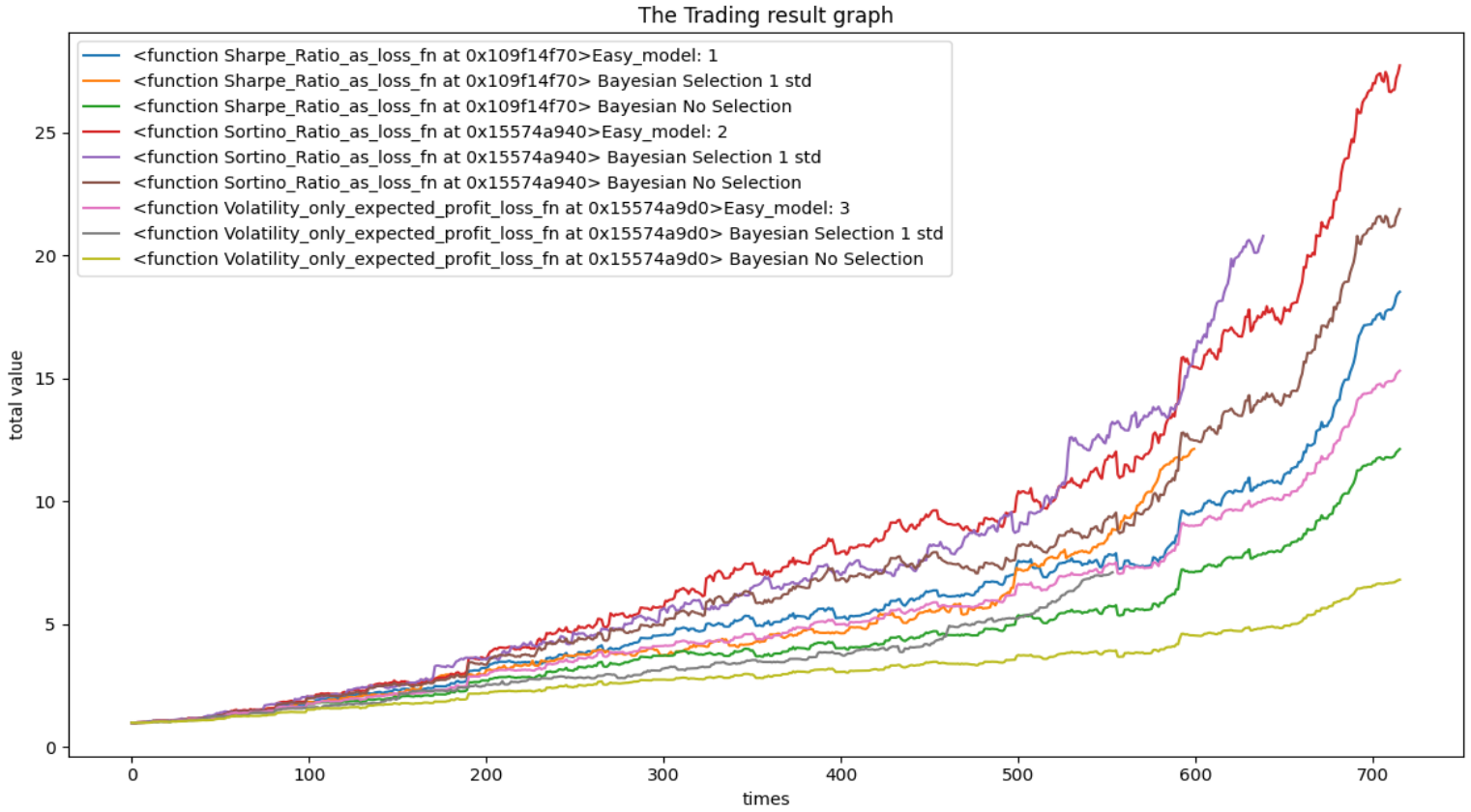


Figure 4 The back-test result for different models and loss functions

Model	Basic Neural Network			Bayesian Model with filter (1-std)			Bayesian Model without filter		
Loss Functions	Sharpe Ratio	Sortino Ratio	Return Rate for each trading	Sharpe Ratio	Sortino Ratio	Return Rate for each trading	Sharpe Ratio	Sortino Ratio	Return Rate for each trading
Final Value	18.52	27.73	15.31	12.13	20.79	7.11	12.12	21.89	6.81
Maximum Withdraw	9.59%	8.92%	6.23%	7.64%	8.87%	5.85%	8.18%	10.80%	6.59%
Sharpe Ratio	4.299	4.276	4.734	4.854	4.750	5.192	4.332	4.368	4.230
Sortino Ratio	6.752	8.400	8.498	8.327	9.770	9.009	7.461	9.138	7.015
Win	442	408	454	354	352	322	415	386	404
Lose	273	307	261	245	286	231	300	329	311
No action	0	0	0	116	77	162	0	0	0
Total	715	715	715	599	638	553	715	715	715
Winning Rate	61.82%	57.06%	63.50%	59.10%	55.17%	58.23%	58.04%	53.99%	56.50%
Buy	299	421	331	263	395	263	314	436	346
Sell	416	294	384	336	243	290	401	279	369
No action	0	0	0	116	77	162	0	0	0
Buying Rate	41.82%	58.88%	46.29%	43.91%	61.91%	47.56%	43.92%	60.98%	48.39%

Table 1 The summary of back-test for different models and loss functions

5.2 Result 2: Using the BNN Model, whether implementing trades with high confidence only lead to better trading regime?

The result in detail is shown in Figure 4 and Table 2.

In the Bayesian Neural Network's trading regime, the confidence of the trading decision could be shown in two ways.

The first indicator is the size of trading positions. +1 or -1 both stand for highest confidence because they maximize the principal that they could use in a single trade, while the second indicator is the $\frac{|m_{\mathfrak{S}}|}{\mathrm{std}_{\mathfrak{S}}}$ introduced in the Section 4.5 using the Monte-Carlo method.

Trades are with the uncertainty. What we are interested in is that whether the BNN model could capture such uncertainty. If so, the BNN model only implements the high confident trades should perform better than the one implementing all trades.

We notate the indicator $\frac{|m_{\mathfrak{S}}|}{\mathrm{std}_{\mathfrak{S}}}$ as the confidence level, and we separately do the back-test for 11 trading regimes. They are with the same BNN model but implementing trades with different confidence level, from 0-std (Implementing all trades), to 5-std (Implementing trades with confidence level ≥ 5 only).

The result is surprising, and it maybe the most important result in my project. It shows that the BNN models implementing the trades with higher confidence level only, will improve their performance measured by all indicators for evaluation, including the Sharpe ratio, the Sortino ratio and the winning rate.

The Sharpe ratio increases gradually from 4.532 (no filter (0-std)) to 5.799 (5-std). So as the Sortino ratio, increasing from 10.777 (no filter (0-std)) to 14.016 (5-std). The winning rate is also rising by about 3% to 57.52% for the BNN model with 5-std confidence level.

The final value is highest for the BNN model with the confidence level (1.5-std), at 22.302 while the one with 5-std confidence level is with the final value 19.328. The main reason is that though the model implementing the most confident trade could improve the efficiency, a part of trading opportunities will be given up, leading to the lower total return.

In total, the BNN (5-std) model will take no action for 216 trading opportunities over 715 samples, while the BNN (1.5-std) model may give up 93 trading chances.

In conclusion, it provides us with a clear inspiration that for the problem with the noisy data set, while at the same time it is not compulsory to predict each samples, it maybe a good idea to use the BNN model, only implementing the prediction for samples with high confidence. The noise / uncertainty in the data set could be detected.

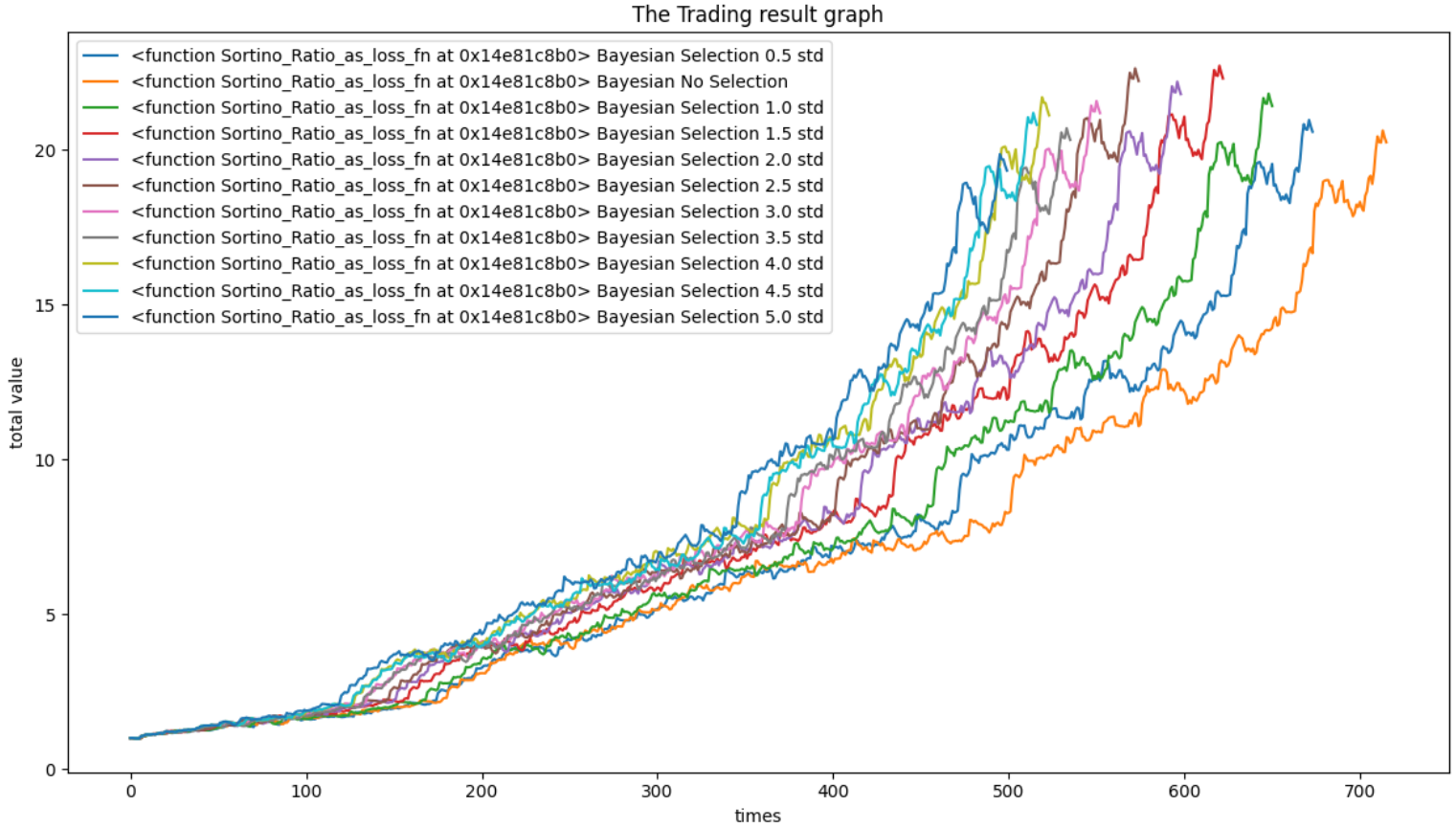


Figure 5 The result of trading decisions based on different confirmation conditions

Model			Bayesian Model with filter								
Loss Functions			Sortino Ratio								
Confidence Level	0	0.5 std	1 std	1.5 std	2 std	2.5 std	3 std	3.5 std	4 std	4.5 std	5 std
Final Value	19.8	20.58	21.415	**22.302**	21.797	22.22	21.184	20.321	21.112	20.806	19.328
Maximum Withdraw	8.83%	7.96%	7.95%	7.98%	7.94%	8.02%	7.99%	8.01%	7.95%	**7.91%**	8.52%
Sharpe Ratio	4.532	4.865	5.016	5.216	5.309	5.495	5.597	5.631	5.78	5.796	**5.799**
Sortino Ratio	10.777	11.526	11.943	12.43	12.665	13.46	13.499	13.641	13.94	13.946	**14.016**
Win	392	371	360	346	333	322	311	302	298	296	287
Lose	323	302	290	276	265	252	241	233	225	220	212
No action	0	42	65	93	117	141	163	180	192	199	216
Total	715	673	650	622	598	574	552	535	523	516	499
Winning Rate	54.83%	55.13%	55.38%	55.63%	55.69%	56.10%	56.34%	56.45%	56.98%	57.36%	**57.52%**
Buy	442	414	400	387	368	358	342	333	324	319	308
Sell	273	259	250	235	230	216	210	202	199	197	191
No action	0	42	65	93	117	141	163	180	192	199	216
Buying Rate	61.82%	61.52%	61.54%	62.22%	61.54%	62.37%	61.96%	62.24%	61.95%	61.82%	61.72%

Table 2 The result of trading decisions based on different confirmation conditions

5.3 Result 3: Is NLP data helpful in trading decision making?

The result in detail is shown in Figure 5 and Table 3.

Unfortunately, the result shows that the multi-modal model and all unimodal models cannot beat the benchmark model, while the multi-modal model also cannot beat the uni-modal model. Therefore, it cannot prove the NLP data could help improve the trading decision-making regime, making better trading decisions.

From the Table 3, it shows that no matter whether the evaluation indicator is the Sharpe ratio or the Sortino ratio, the benchmark model is keeping highest compared with all of the NLP models.

Among all of the NLP models, the unimodal NLP model using the word embedding data only performs best, with the highest winning rate, Sharpe ratio (3.862), and the Sortino ratio (7.716). However, actually, the performances are pretty close among these NLP models, with not much difference.

Moreover, what surprises us is that the multi-model actually performs worst among all models, and this result is actually different from previous research totally (Ramit et al., 2020). Previous researches mentions its relevance, however, there is still a long run for making such data into the real application.

One of the potential explanations is that the NLP data is with too much noise, far more than the useful information that it carries. This assumption is reasonable, because even for a human, it is hard to use the vocal feature or the semantic feature only, extracting the information from these features and

making trading decisions.

The word embedding feature may carry more information than the other two features, and that is why the NLP model using the word embedding feature only has better performance than the one with semantic / vocal features only.

Assuming the NLP data is useful for the prediction model, more delicate features' extraction / data preprocessing should be done as the potential solution.

For example, for the vocal feature extraction, we could identify the speakers as different entities, to classify and group their vocal features separately. Moreover, from the literature review, we have already known that speakers' emotion transmitted by the voice feature actually is with a pretty complex pattern, such as the voice intensity and pitch fluctuation in a sentence (Jiang and Pell, 2017). Therefore, a more delicate pre-trained vocal feature extraction model may be needed.

On the other hand, for the word embedding vector extraction, preprocessing could include not only the stop words' removal, but also the lemmatization and stemming. In addition, the model could be fed by the most important speaker's speaking only to improve their relativeness.

.

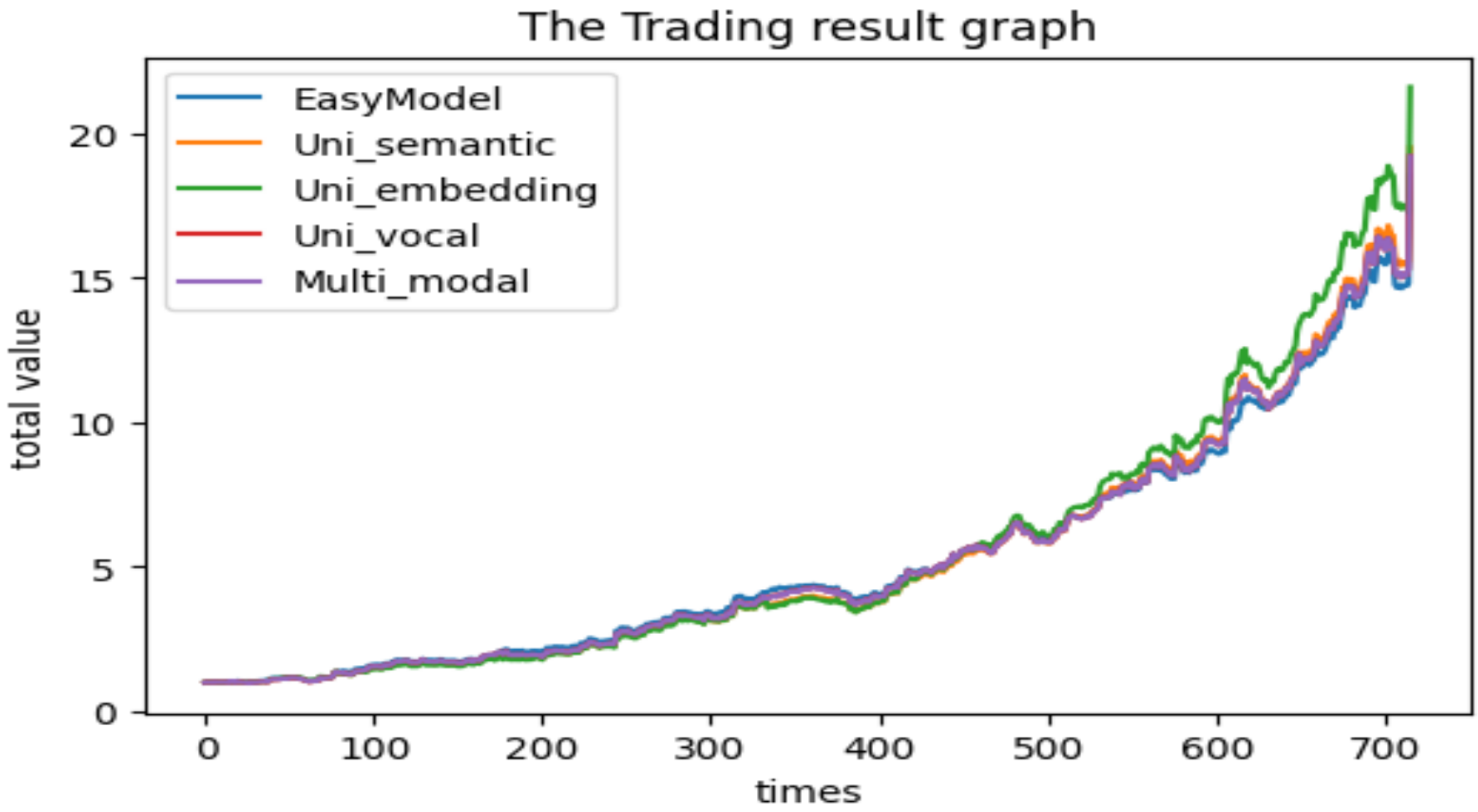


Figure 6 The trading results of Unimodal/Multi-modal models

Model	Easy Model	Uni-modal Model (Semantic Feature)	Uni-modal Model (Embedding Feature)	Uni-modal Model (Vocal Feature)	Multi-modals Deep Learning Decision Making Models
Loss Functions	Sortino Ratio				
Final Value	18.54	19.51	**21.61**	19.21	19.21
Maximum Withdraw	**12.50%**	13.93%	13.33%	14.13%	14.13%
Sharpe Ratio	**4.014**	3.727	3.862	3.717	3.716
Sortino Ratio	**9.921**	7.702	7.716	7.661	7.689
Win	386	390	400	390	391
Lose	329	325	315	325	324
No action	0	0	0	0	0
Total	715	715	715	715	715
Winning Rate	53.99%	54.55%	**55.94%**	54.55%	54.69%
Buy	422	389	364	408	408
Sell	293	326	351	307	307
Buying Rate	59.02%	54.41%	50.91%	57.06%	57.06%

Table 3 The summary of back-test for different NLP models

5.4 More Inspiration for similar questions and future work

Because our result is from the data with a high level of noise, it is not convincible that the result is representative. Our success in the backtest may be due to luck and randomness.

As a check, all data are shuffled randomly 50 times separately and the model is trained separately. Their back-test results are shown in Figure 7. It shows that for these 50 times back-test, final values are mostly between 10-25, with the mean final value of 17.99, the mean maximum withdrawal of 12.23%, the mean Sharpe ratio of 4.41, and the mean Sortino ratio of 8.56, excluding the times when the model cannot converge (several times). Therefore, based on Bernoulli's law of large numbers and Chebyshev's theorem, we could conclude that the back-test result and the data set both are actually sample from the uncertain distribution. If the number of times for this repeated experiment is large enough, the average result will converge to the expectation (Vinogradov, 2021).

In the future, many similar problems could be investigated using similar models and analytic skills. For these potential problems, two properties should be satisfied. The first one is that **the dataset is with high uncertainty**, while the second one is that **the prediction is not compulsory for each samples in the test set.**

Therefore, the potential applications may be special and unique, such as football game betting, casino games' design and reversed optimization, and the recommendation system design.

6. Limitations

6.1 Factors not in consideration

Though our model performs well in the back-test, some factors may be ignored and not in consideration may make it fail in real practice.

Potential Factor 1: Over-fitting maybe a limit

The data set that we are collecting is from the Standard & Poors' 500 list only, while the trading date is also only the date after the company releases its quarter / annual financial performance and conference call. Therefore the trading dates and companies are actually highly limited, which may potentially cause the over-fitting problem. So the real practice maybe worse than the back-test. The potential solution is to enlarge the stock list in the model training, testing the model in different economic conditions.

Potential Factor 2: Time horizon maybe a limit

The result of intra-day trades is also based on investors' expectation and their behavior patterns. It may change with time, and actually even the economic principles may change. For example, when the economic condition is at peak or in recession, options' volatility may increase when new information releases, because the market is expected to the high fluctuation. In our examples, only samples in recent 3 years are collected, when the economics are quite stable. It makes our back-test results not representative enough and the real practice result may be totally different when it is in different economic conditions. To test this potential problem, the model could be used to test the result in earlier time period like 20 years ago. It could also be used to test the ongoing market (today).

Potential Factor 3: Delta Hedge and Liquidity as a limit

In the back-test result, our trading regime assumed that the Delta-Hedge pre-condition could be satisfied, so that our model's profit / loss is based on the volatility change only. However, to satisfy the Delta-Hedge, needs the market to be in the high liquidity condition which is hard to be satisfied for most companies' option trades.

For example, assuming the model shows a 'buy' signal for a trade, consisting of 65% of the 'call' option position and 35% of the 'put' option position to satisfy the Delta-Hedge pre-condition. The minimal unit for trades in the options market is 1, corresponding to 100 shares of the stock. Therefore, we at least need to buy 7 units' put option and 13 units' call option. However, excepting the biggest companies in the S&P 500, it may be hard to complete such trades due to liquidity issues. Lots of companies' trading volumes in the options market are lower than $ 1 million per day, which makes this strategy hard to complete.

Giving up the Delta-Hedge precondition may be an alternative selection. The price is that our trading result may be impacted by the stock price and implied volatility together, leading to higher risk because a new risk factor is introduced. The result is that the Sharpe ratio, and the Sortino ratio will decrease while the maximum withdrawal will rise up.

In conclusion, for the gap between the success in the back-test and the success in real practice, these three factors may be the biggest issues.

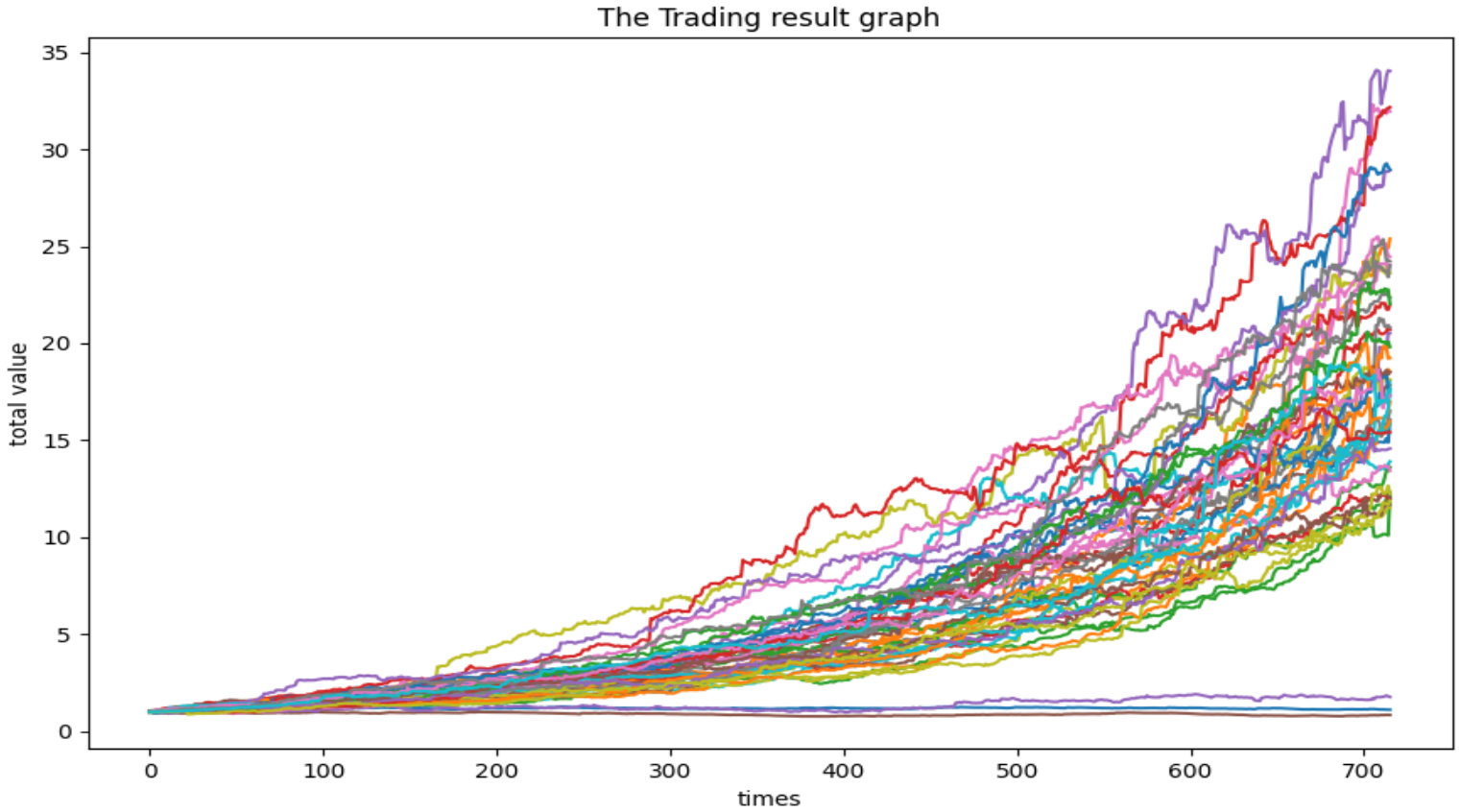


Figure 7 Different backtest results due to data shuffle

7. Conclusion

In this project, we build up several trading models to discover volatility arbitrage opportunities in the options market. Firstly, we innovatively combine the back-test process into loss functions, and realize the Sortino ratio, a less frequently used indicator in finance, actually is quite useful in machine learning and even the real practice.

Secondly, we observe that the prediction and trading decision are two different type's of problems in machine learning and our innovation is to recognize the high level of uncertainty in the data and realize that the BNN model is able to capture such uncertainty. Implementing trades with high confidence only could lead to better result.

Finally, it maybe the first try in industry as far as we know to use the NLP data into the real trading practice. Though many researches show the NLP data is relevant to the stock performance, we figure out that it is still a long way to go to make such data helpful in making better trading decisions.

8. Bibliography

1. Alberto Pettarin. (2017). “Anenas” https://github.com/readbeyond/aeneas

2. Black, Fischer; Myron Scholes (1973). "The Pricing of Options and Corporate Liabilities". Journal of Political Economy. 81 (3): 637–654. doi:10.1086/260062. S2CID 154552078. [1] (Black and Scholes' original paper.)

3. Carlin BP, Louis TA (1997). Bayes and empirical Bayes methods for data analysis. Statistics and Computing. 7(2):153–154.

4. Chandra R, He Y (2021) Bayesian neural networks for stock price forecasting before and during COVID-19 pandemic. PLoS ONE 16(7): e0253217.

5. C.R. Harvey, R.E. Whaley. (1992). Market volatility prediction and the efficiency of the S&P 100 index option market Journal of Financial Economics, 31 (1), pp. 43

6. Dhillon, Upinder, and Herb Johnson. (1991). "Changes in the Standard and Poor's 500 List." Journal of Business: 75-85.

7. Freedman DA. (1963). On the Asymptotic Behavior of Bayes’ Estimates in the Discrete Case. The Annals of Mathematical Statistics. 34(4):1386–1403.

8. Jang, Huisu, and Jaewook Lee. (2017). "An empirical study on modeling and prediction of bitcoin prices with bayesian neural networks based on blockchain information." Ieee Access 6: 5427-5437.

9. Jongwoo Lee & Dean A. Paxson (2003) Confined exponential approximations for the valuation of American options, The European Journal of Finance, 9:5, 449-474, DOI: 10.1080/1351847032000082796

10. Joyce Menezes da Fonseca Tonin, Luciano Marcio Scherer, (2022). MARKET REACTION TO THE TONES OF EARNINGS CONFERENCE

CALLS, Revista de Administração de Empresas, 10.1590/s0034-759020220107x, 62.

11. Liang F (2005). Bayesian neural networks for nonlinear time series forecasting. Statistics and computing. 15(1):13–29.

12. MacKay DJ. (1996). Hyperparameters: Optimize, or integrate out? In: Maximum entropy and Bayesian methods. Springer; p. 43–59.

13. MacKay DJC. (1992). A Practical Bayesian Framework for Backpropagation Networks. Neural Comput. 4(3):448–472.

14. Mary Malliaris, Linda Salchenberger, (1996). Using neural networks to forecast the S&P 100 implied volatility, Neurocomputing, Volume 10, Issue 2, Pages 183-195

15. Moreno, Pedro J., et al. (1998). "A recursive algorithm for the forced alignment of very long audio segments." ICSLP. Vol. 98.

16. Neal RM. (1996). Bayesian Learning for Neural Networks. vol. 118. Springer New York;

17. Nicholas Dingwall and Christopher Potts. (2018). Mittens: an Extension of GloVe for Learning Domain-Specialized Representations. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers). Association for Computational Linguistics, New Orleans, Louisiana, 212–217. https://doi.org/10.18653/v1/N18-2034

18. Paul C. Tetlock. (2007). Giving content to investor sentiment: The role of media in the stock market. Jour_x0002_nal of Finance, 62(3):1139–1168.

19. Pedro J Moreno, Chris Joerg, Jean-Manuel Van Thong, and Oren Glickman. (1998). A recursive algorithm for the forced alignment of very long audio segments. In In proceedings of ICSLP.

20. Pennington, Jeffrey, Richard Socher, and Christopher D. Manning. (2014).

"Glove: Global vectors for word representation." Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP).

21. PRoFET: Predicting the Risk of Firms from Event Transcripts Christoph Kilian Theil , Samuel Broscheit and Heiner Stuckenschmidt Data and Web Science Group, University of Mannheim, Germany {christoph, broscheit, heiner}@informatik.uni-mannheim.de

22. P. Ser-Huang, G. Clive. (2005). Practical issues in forecasting volatility Financial Analysts Journal, 61 (1), pp. 45-56

23. Qin, Yu., & Yang, Yi. (2019). What You Say and How You Say It Matters: Predicting Stock Volatility Using Verbal and Vocal Cues. Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics.

24. Ramit Sawhney, Puneet Mathur, Ayush Mangal, Piyush Khanna, Rajiv Ratn Shah, and Roger Zimmermann. (2020). Multimodal Multi-Task Financial Risk Forecasting. In Proceedings of the 28th ACM International Conference on Multimedia (MM '20), October 12–16, 2020, Seattle, WA, USA. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3394171.3413752

25. R. Gencay and M. Qi, (2001). "Pricing and hedging derivative securities with neural networks: Bayesian regularization early stopping and bagging", IEEE Trans. Neural Netw., vol. 12, no. 4, pp. 726-734.

26. R.F. Engle and M. Rothschild, (1992). Statistical models for financial volatility, J. Econometrics 52 1-4.

27. R.F. Engle, M.E. Sokalska. (2012). Forecasting intraday volatility in the us equity market. multiplicative component GARCH Journal of Financial Econometrics, 10 (1), pp. 54-83

28. Rollinger, Thomas N., and Scott T. Hoffman. (2013). "Sortino: a 'sharper'ratio." Chicago, Illinois: Red Rock Capital

29. Sepp Hochreiter and Jurgen Schmidhuber. (1997). ¨ Long short-term

memory. Neural computation, 9(8):1735–1780.

30. Sharpe, William F. (1998). "The sharpe ratio." Streetwise–the Best of the Journal of Portfolio Management: 169-185.

31. S. Muzzioli (2010) Option-based forecasts of volatility: an empirical study in the DAX-index options market, The European Journal of Finance, 16:6, 561-586, DOI: 10.1080/13518471003640134

32. Shimon Kogan, Dimitry Levin, Bryan R. Routledge, Jacob S. Sagi, and Noah A. Smith. (2009). Predicting risk from financial reports with regression. In In Proceedings of NAACL, pages 272–280.

33. S.J. Koopman, B. Jungbacker, E. (2005). Hol Forecasting daily variability of the S&P 100 stock index using historical, realised and implied volatility measurements Journal of Empirical Finance, 12 (3), pp. 445-475

34. Soujanya Poria, Erik Cambria, Devamanyu Hazarika, Navonil Majumder, Amir Zadeh, and Louis-Philippe Morency. (2017). Context-dependent sentiment anal_x0002_ysis in user-generated videos. In In Proceedings of ACL, volume 1, pages 873–883.

35. Tim Loughran and Bill McDonald. (2011). When is a liability not a liability? textual analysis, dictionaries, and 10-ks. The Journal of Finance, 66(1):35–65

36. Vinogradov, O.P. On Chebyshev's Theorem and Bernoulli's Law of Large Numbers. Moscow Univ. Math. Bull. 76, 135–138 (2021). https://doi.org/10.3103/S0027132221030086

37. William J Mayew and Mohan Venkatachalam. (2012). The power of voice: Managerial affective states and future fifirm performance. The Journal of Finance, 67(1):1–43.

38. William Yang Wang and Zhenhao Hua. (2014). A semi-parametric gaussian copula regression model for predicting financial risks from earnings calls. In In Proceedings of ACL, volume 1, pages 1155–1165.

39. Xiao Ding, Yue Zhang, Ting Liu, and Junwen Duan. (2015). Deep learning for event-driven stock prediction. In In Proceedings of IJCAI, pages 2327–2333.

40. Xiaoming Jiang and Marc D Pell. (2017). The sound of confidence and doubt. Speech Communication, 88:106–126.

41. Y. Zhang, T. Yao, L. He, R. (2018)Ripple Volatility forecasting of crude oil market: Can the regime switching GARCH model beat the single-regime GARCH models? International Review of Economics & Finance, 59

42. Zhiqian Ma, Chong Wang, Grace Bang, and Xiaomo Liu. (2020). Utilization of Deep Learning to Mine Insights from Earning Calls for Stock Price Move- ment Predictions. In ACM International Conference on AI in Finance (ICAIF '20), October 15–16, 2020, New York, NY, USA. ACM, New York, NY, USA, 8 pages.

43. Ziniu Hu, Weiqing Liu, Jiang Bian, Xuanzhe Liu, and Tie-Yan Liu. (2018). Listening to Chaotic Whispers: A Deep Learning Framework for News-oriented Stock Trend Prediction. In WSDM 2018: The Eleventh ACM International Conference on Web Search and Data Mining , February 5–9, 2018, Marina Del Rey, CA, USA. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3159652.3159690