Statistical Arbitrage on S&P 500 by Natural Language Processing (NLP)

Statistical Arbitrage on S&P 500 by Natural Language Processing (NLP)

1. Introduction

1.1 Background

In capstone 1, I only used daily price with machine learning methods to predict one-day movement of stock price. Besides that, analysts are dedicated to analyzing and attempt to quantify qualitative data from social media (such as twitter), breaking news (from Reuters), or public released files (SEC mandated reporting). Form 8-K is one of the most common forms filed with the SEC, after a significant event such as bankruptcy or a CEO design, a report must be filed on Form 8-K file within four business days to provide updated to previous Form 10-Q (quarterly reports) and Form 10-K (annual reports). Since there are huge amounts of 8-K files related to different kinds of stocks in decades, reading all information manually by human being will cost lots of time and may miss trading opportunity. With the help of NLP, we can use machine learning methods to analysis the opinion and ideas from many reports to predict the stock's movement and make a trading decision automatically.

1.2 Literature Review

Ryan (2017) developed a model with CNN-Glove to predict the future stock market performance of public companies in categories where a financial services partner of her company invested. She used SEC (Securities and Exchange Commission) 10-K reports and got 62% prediction accuracy. Yusuf (2018) compared machine learning methods MLP, CNN, RNN, CNN-RNN to predict stock movement based on SEC 8-K reports. He found CNN-RNN reached 64.5% accuracy on the validation data. Following Yusuf, Babbe et. al (2019), they trained a BERT + single layer MLP model and produced an out of sample accuracy of 71%. Joseph (2020) defined a stock price formula with the sentiment signal from 267 journal articles related to APPL to predict one day price of it.

1.3 Goal

This project will follow Yusuf (2018)'s work at first, collect the SEC 8-K reports related to S&P 500 companies from Jan 2005 to the end of Oct 2020, use MLP, CNN, RNN and CNN-RNN models to train for stocks with most articles by two cases (case 1 and case 2), then long (short) stocks with highest probabilities on file released date (or one day delay) based on case 2. The baseline will be all stocks be trained with above four models. The goals are to find the best model for every trading year (see if it can beat S&P 500 index) and see the performance of these methods in different economics situations (e.g. financial crisis in 2008, COVID-19 pandemic in 2020).

2. Datasets Extract, Transform and Load (ETL)

2.1 List of historical S&P 500 components

List of S&P 500 components download from Wikipedia (up to date 11/25/2020), stock tickers, GICS sector, and CIK (this is used to download SEC 8-K files) are used in further analysis.

2.2 Daily data of historical S&P 500 components

Based on the components from section 2.1, use Python yahoo finance module yahoo-finance-1.0.4 to download daily data of all components and S&P 500 index from 2005/01/01 to 2020/11/01 (15 years and 10 months data).

2.3 SEC 8-K Reports

Download 8-K files

From SEC website (https://www.sec.gov/edgar/searchedgar/companysearch.html) to download 8-K files of stocks from 2.1 with timestamp from 01-01-2005 to 10-31-2020.

3. Exploration Data Analysis (EDA)

3.1 Data split for formation and trading

Like capstone 1, I set the length of the in-sample formation window to 750 days (approximately three years) and the length of the subsequent out-of-sample trading window to 250 days (approximately one year). I move the formation-trading set forward by 250 days in a sliding-window approach, resulting in 13 non-overlapping trading batches (Table 1) to loop over my entire data set from begin of year 2005 until end of 2020 Oct.

Subset	form_start	form_start	trad_end	Subset	form_start	form_start	trad_end
0	1/1/2005	1/1/2008	1/1/2009	7	1/1/2012	1/1/2015	1/1/2016
1	1/1/2006	1/1/2009	1/1/2010	8	1/1/2013	1/1/2016	1/1/2017
2	1/1/2007	1/1/2010	1/1/2011	9	1/1/2014	1/1/2017	1/1/2018
3	1/1/2008	1/1/2011	1/1/2012	10	1/1/2015	1/1/2018	1/1/2019
4	1/1/2009	1/1/2012	1/1/2013	11	1/1/2016	1/1/2019	1/1/2020
5	1/1/2010	1/1/2013	1/1/2014	12	1/1/2017	1/1/2020	11/1/2020
6	1/1/2011	1/1/2014	1/1/2015				

Table 1: Formation and Trading periods for 13 non-overlapping trading batches

3.2 Data Cleaning

From download SEC 8-K files, there are total 108,347 files, after removing length of files more than 60,000 words, 107,348 files left. Text preprocessing includes - remove extra whitespace, tokenize and remove punctations, stop words and covert all words to lower case. After that 107,343 files are left. Set the length of text between 99 (0.05 quantile) and 1508 (0.99 quantile) causes 100,961 files as our final research target.

3.3 Data Description

Figure 1 and 2 display the length of files in training and trading periods, the shapes are close to possion distributions. For most of training and trading datasets, the average words are around 260 with standard deviation as 200. More words appear in recent years for both training and trading datasets. Figure 3 displays number of files for each GIS sector and year. Financials, industrials and health care are top three sectors and there are around 6,000 files for each year.

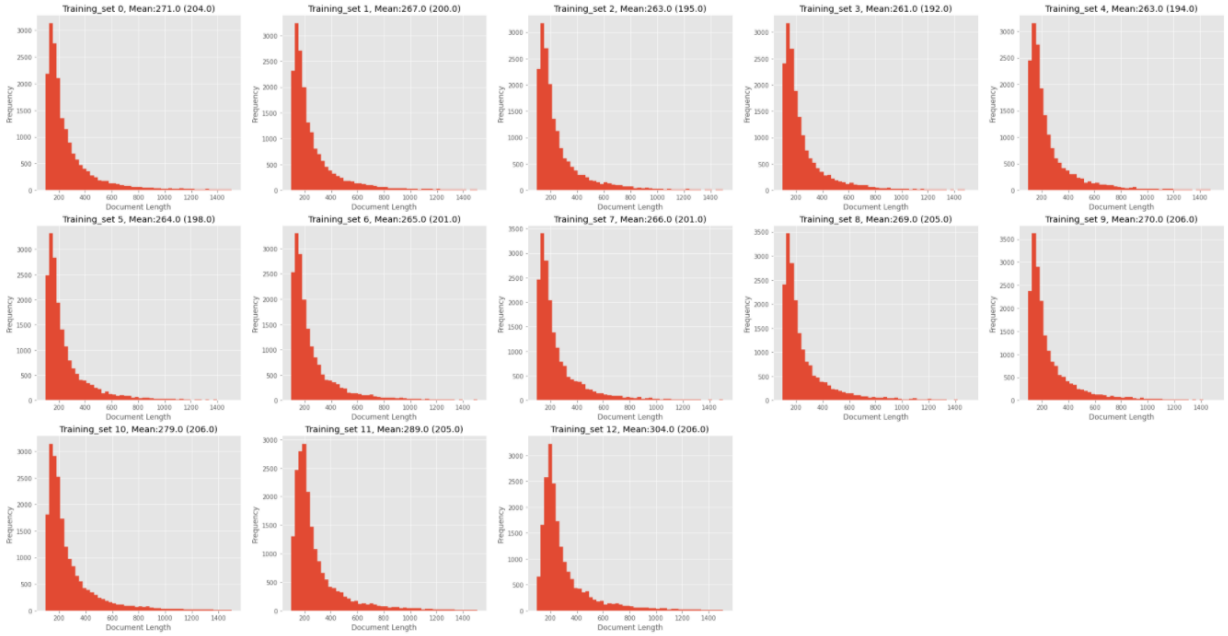


Figure 1. Length of SEC 8-K files in different training periods

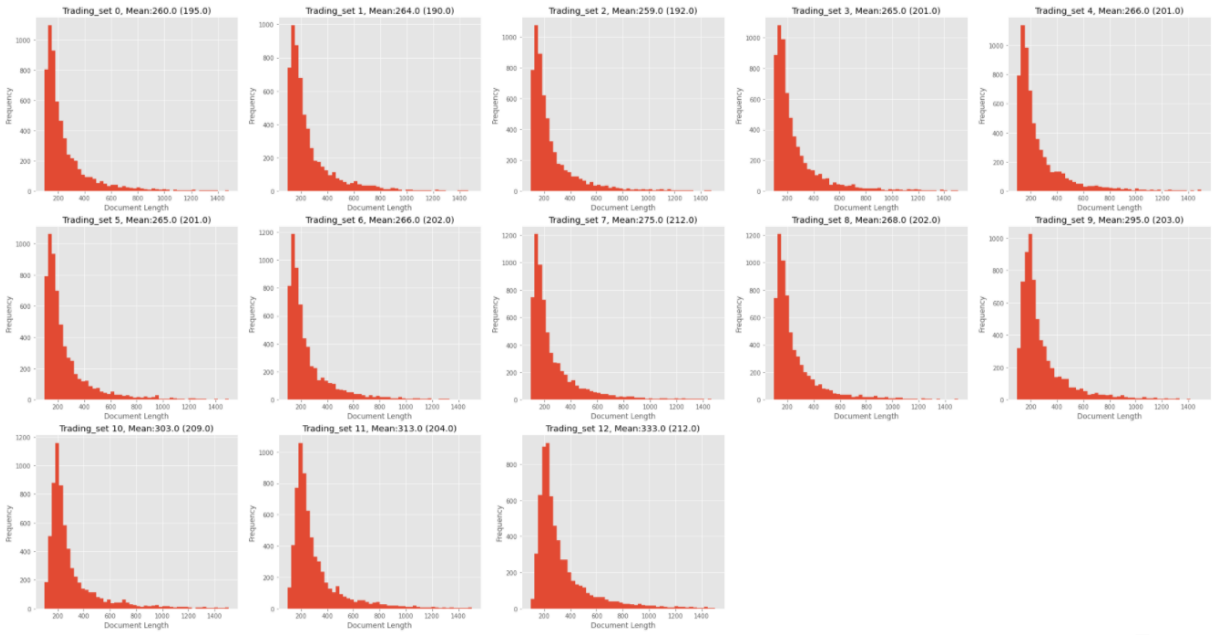


Figure 2. Length of SEC 8-K files in different trading periods

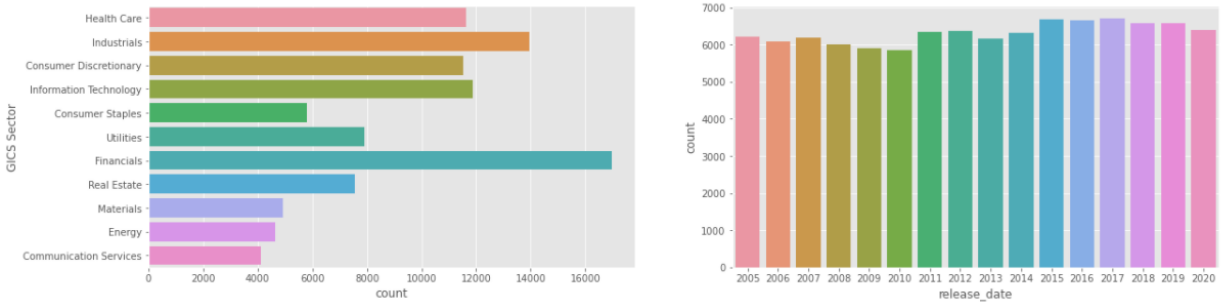


Figure 3. Number of SEC 8-K files for each GIS sector and release year

3.4 Feature & Target Generation

For each formation-trading set, the feature space (input) and the response variable (output) are created as follows:

Input:

- Let $P^s = (P_t^s)_{t \in T}$ denote the price process of stock s or financial indicator, with $s \in \{1, \ldots, n\}$. Define the simple return $R_{t,m}^s$ for each stock or indicator over m periods as $R_{t,m}^s = \frac{P_t^s}{P_{t-m}^s} - 1$
  Where $m \in \{1\ week,\ 1\ month, 1\ quarter\ and\ 1\ year\}$. The notation of features is f_m, m is defined as above which represents returns of different lags of periods.
- VIX (CBOE Volatility Index), choose open price if the file is released before the market open, otherwise choose close price.
- GIS section as categorical data

Output:

Construct a triple response variable $Y_t^s \in \{0,1,2\}$ for each stock s. Here index represents S&P 500 index.

Case 1: Immediate effect of released files

$$s_{\ price\_change}(t) = \begin{cases} s_{\ open}(t) - s_{\ close}(t-1), if\ 8-K\ is\ released\ before\ normal\ trading\ hours \\ s_{\ close}(t) - s_{\ open}(t), if\ 8-K\ is\ released\ during\ normal\ trading\ hours \\ s_{\ open}(t+1) - s_{\ close}(t), if\ 8-K\ is\ released\ after\ normal\ trading\ hours \end{cases}$$

$$Index_{\ price\_change}(t) = \begin{cases} index_{\ open}(t) - index_{\ close}(t-1), if\ 8-K\ is\ released\ before\ normal\ trading\ hours \\ index_{\ close}(t) - index_{\ open}(t), if\ 8-K\ is\ released\ during\ normal\ trading\ hours \\ index_{\ open}(t+1) - index_{\ close}(t), if\ 8-K\ is\ released\ after\ normal\ trading\ hours \end{cases}$$

Case 2: Delay effect of released files

$$s_{\ price\_change}(t) = \begin{cases} s_{\ close}(t) - s_{\ open}(t), if\ 8-K\ is\ released\ before\ normal\ trading\ hours \\ s_{\ open}(t+1) - s_{\ close}(t), if\ 8-K\ is\ released\ during\ normal\ trading\ hours \\ s_{\ close}(t+1) - s_{\ open}(t+1), if\ 8-K\ is\ released\ after\ normal\ trading\ hours \end{cases}$$

$$Index_{\ price\_change}(t) = \begin{cases} index_{\ close}(t) - index_{\ open}(t), if\ 8-K\ is\ released\ before\ normal\ trading\ hours \\ index_{\ open}(t+1) - index_{\ close}(t), if\ 8-K\ is\ released\ during\ normal\ trading\ hours \\ index_{\ close}(t+1) - index_{\ open}(t+1), if\ 8-K\ is\ released\ after\ normal\ trading\ hours \end{cases}$$

Y_t^S for both case 1 and 2 are constructed as follows.

$$spread_{\ price\_change}(t) \ = \ s_{\ price\_change}(t) - index_{\ price\_change}(t)$$

$$Y_t^s = \begin{cases} up\ (2), & if\ \ spread_{\ price\_change}(t) > 0.01 \\ stay\ (1), & if\ \ |spread_{\ price\_change}(t)| \leq 0.01 \\ down\ (0), & if\ spread_{\ price\_change}(t) < -0.01 \end{cases}$$

Case 1 is meaningful and easy to understand, but not easy to be processed in the trading since we only have daily data; case 2 tries to detect 8-K (delay) effect on trading day, it can be processed in trading.

3.5 Feature Description

Figure 3 and 5 display heatmap correlation between created features vix, 1-week lag (rm_week), 1-month lag (rm_month), 1-quarter lag (rm_qtr), 1-year lag (rm_year) of stock prices for case 1 and 2 defined in section 3.3. There is no obvious pairs correlation between these features. Figure 4 and 6 display frequency of signals in training and trading sets for case 1 and 2. From those plots, we can find there are more "stay" signals than "up" and "down" signals, so we will oversample in our training datasets.

- Case 1

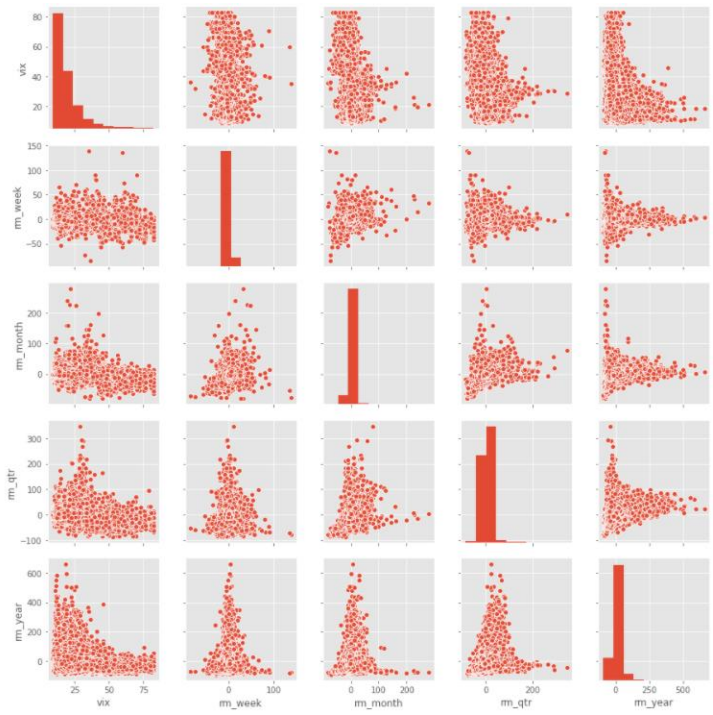


Figure 3. Heatmap correlation between new created features vix, 1-week lag (rm_week), 1-month lag (rm_month), 1-quarter lag (rm_qtr), 1-year lag (rm_year) of stock prices for case 1

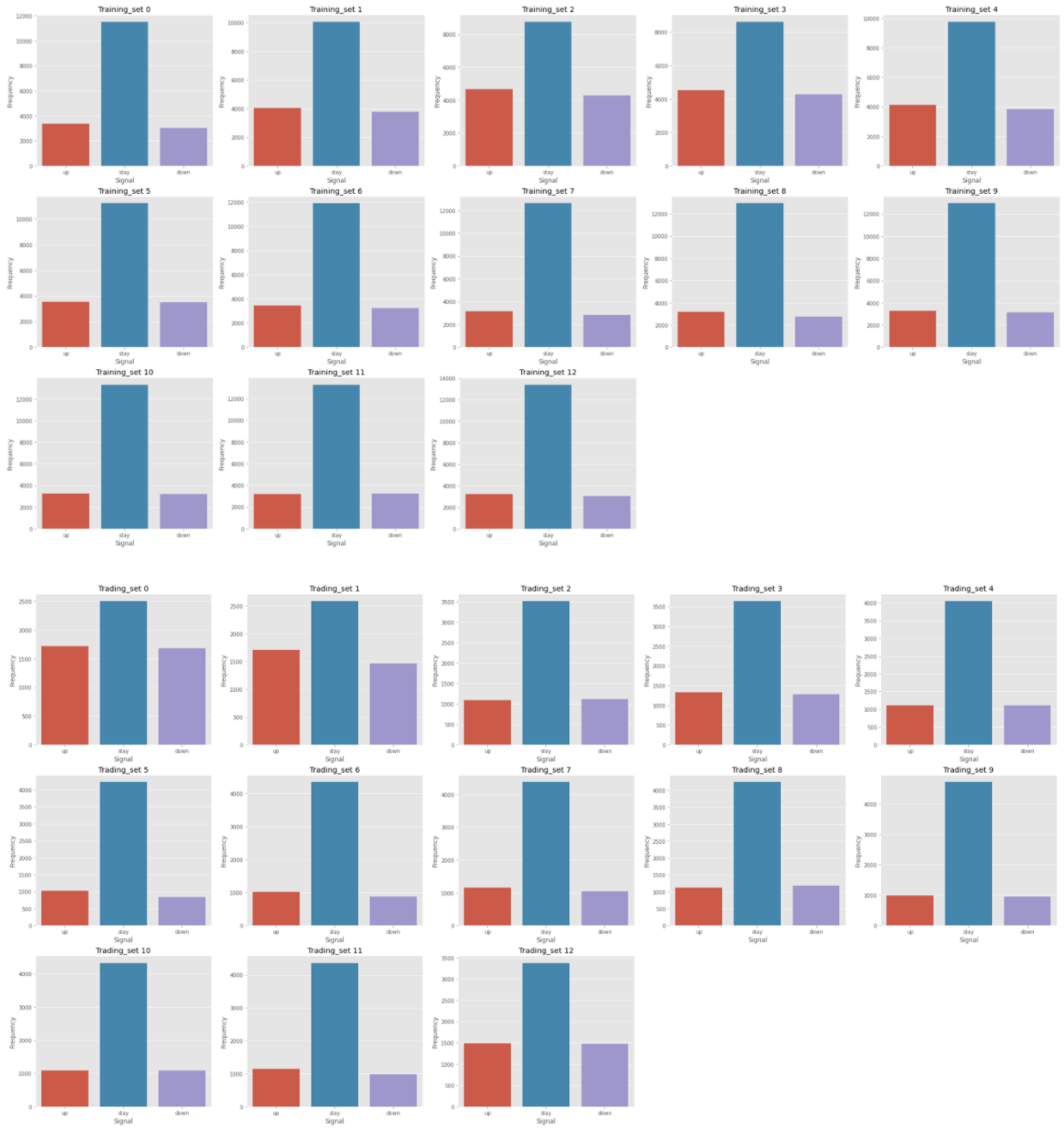


Figure 4. Frequency of signals in training and trading sets for case 1

- Case 2

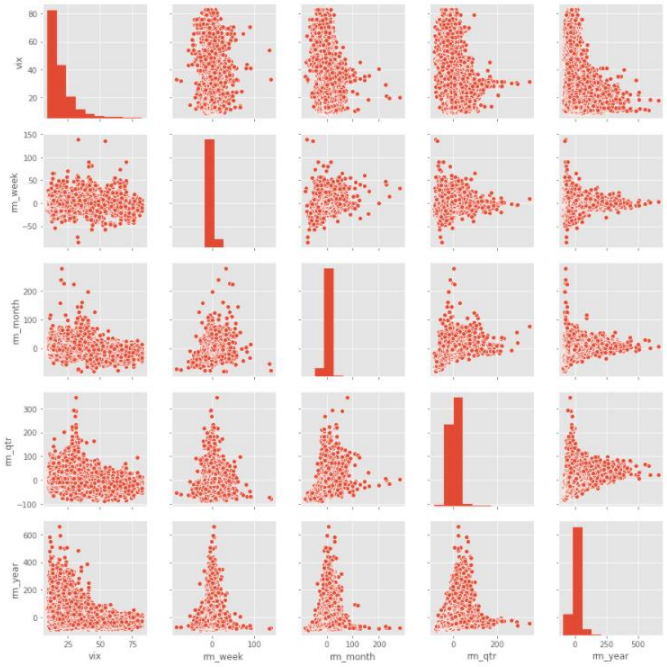
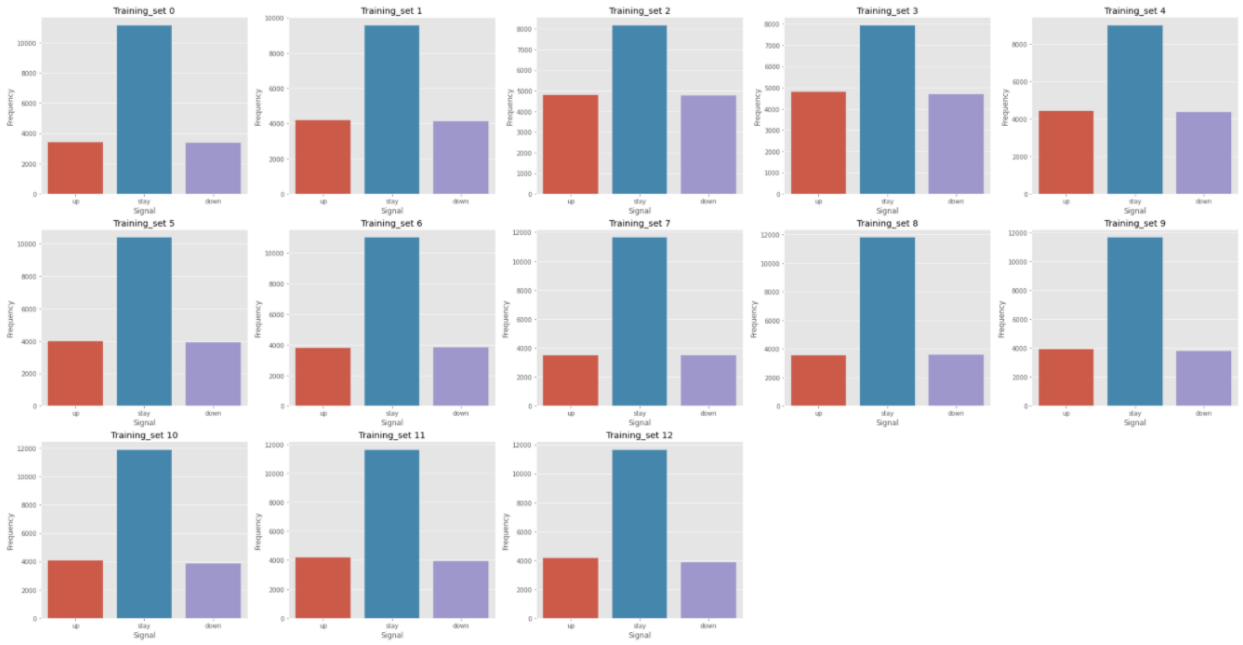


Figure 5. Heatmap correlation between new created features vix, 1-week lag (rm_week), 1-month lag (rm_month), 1-quarter lag (rm_qtr), 1-year lag (rm_year) of stock prices for case 2



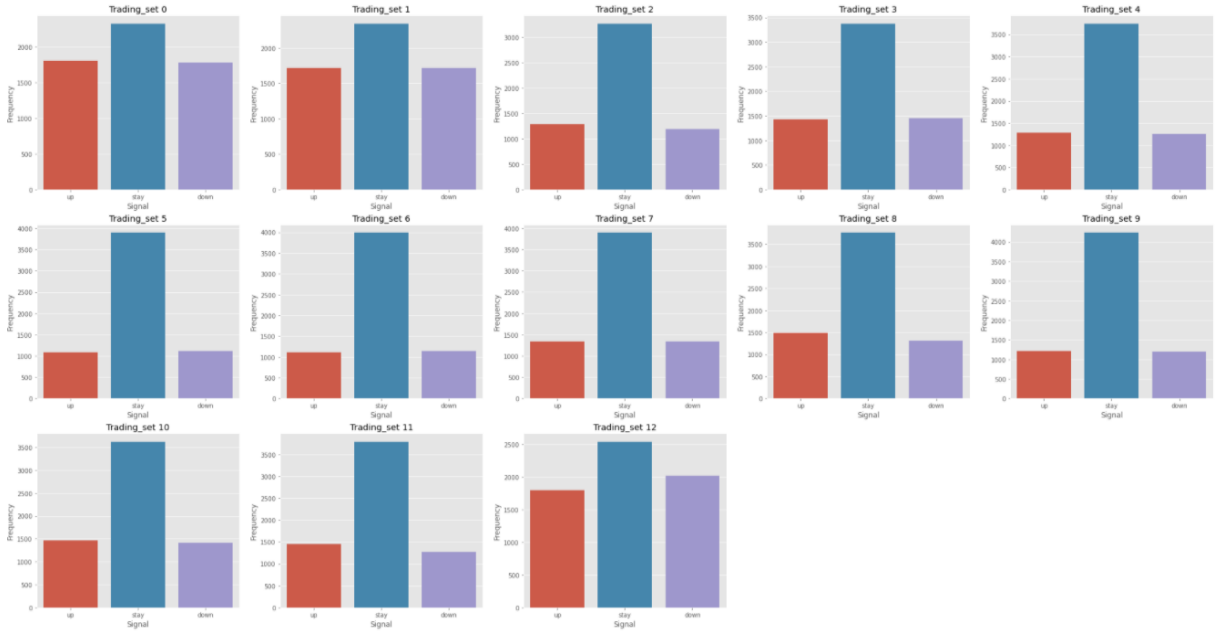


Figure 6. Frequency of signals in training and trading sets for case 2

3.6 Embedding Words

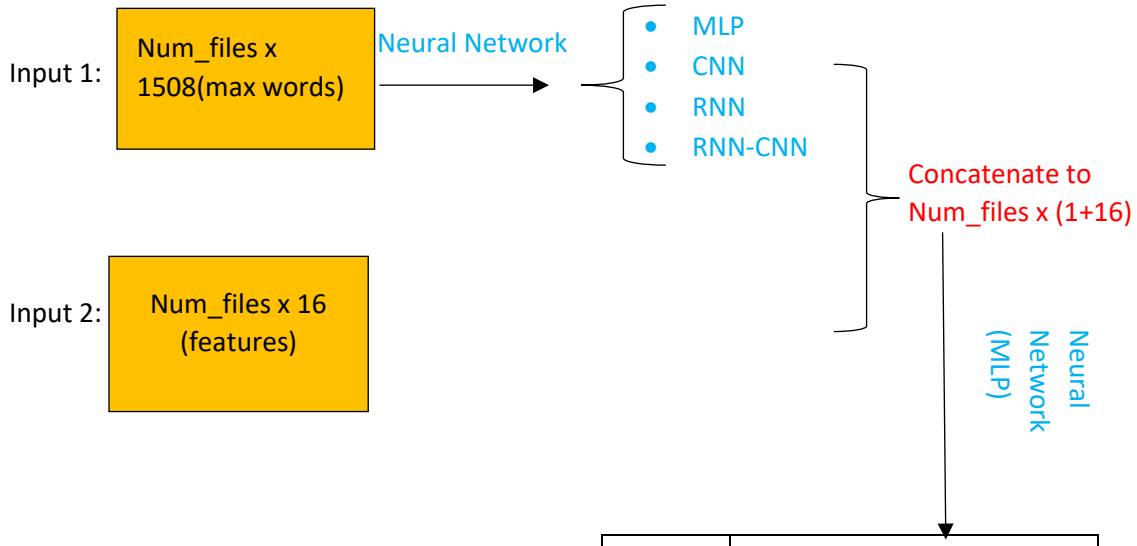
Different words are embedded from GloVe 100 dimension (https://nlp.stanford.edu/projects/glove/). “GloVe is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space.”

4. Deep Learning models

4.1 Design of deep learning models

I used deep learning models –MLP, CNN, RNN, RNN+CNN. Figure 7 shows the design of the two inputs and neural networks. Figure 8 shows detail neural network design for MLP,

CNN, RNN, RNN-CNN. Up to 50 epochs are used for each method with early stopping (5 steps patience) on Keras Tensorflow 1.14 - monitor of accuracy.



Layers	Design
1	(None,32), relu, dropout (0.2)
2	(None,32), relu
3	(None,32), relu
4	Sigmoid

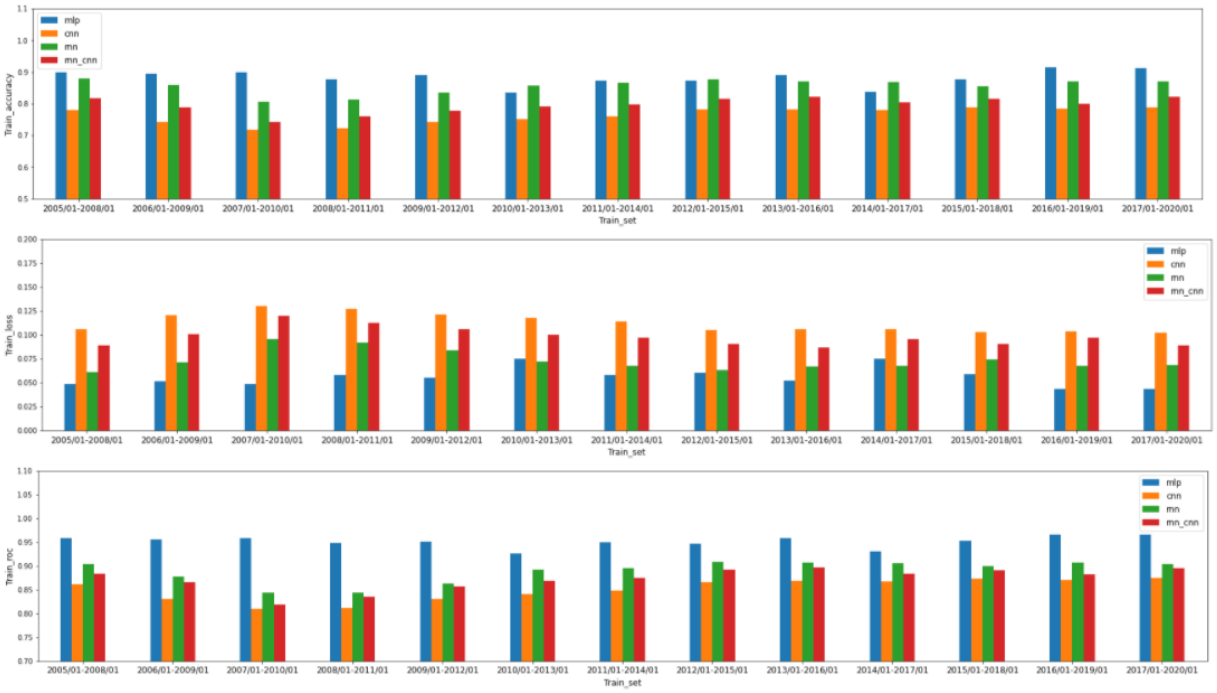
Figure 7. Design of two inputs and neural networks

Common Layers for MLP, CNN, RNN, RNN+CNN	Design
c1	(Num_files,1508)
c2	(1508, vocab_size)
c3	(vocab_size, 100)
Layers for MLP	Design
1	(None,32), relu, dropout (0.2)
2	Dense
Layer for CNN	Design
1	(64,3), stride=1, relu, padding =same
2	Max_pool, size=3, dropout (0.2)
3	(32,3), stride=1, relu, padding =same
4	Dense
Layers for RNN	Design
1	CuDNNGRU (64)
2	Dense
Layers for CNN-RNN	Design
1	(64,5), stride=1, relu, padding =same
2	Max_pool, size=2, dropout (0.2)
3	CuDNNGRU (32)
4	Dense

Figure 8. Design of neural networks for MLP, CNN, RNN CNN-RNN (vocab_size represents number of words found in GloVe 100)

4.2 Results for Deep Learning

Figure 9 and 10 display the train and test accuracy, loss and roc for case 1 and case 2 by four deep learning methods. The pattern looks similar for both case 1 and 2 for train datasets – the performance from best to worst is MLP, RNN, CNN-RNN, CNN. For accuracy in test (trade) datasets, all methods cannot reach the highest probability of the most categorical ("stay") since all datasets are unbalanced. Analysis by different years, MLP performs best in year 2008, 2009, 2013,2014, 2015, 2018 and 2019 and in year 2008, 2010, 2012, 2013, 2014 and 2016 for case 1 and case 2, respectively; CNN performs best in year 2011, 2017 and in year 2011, 2019 for case 1 and case 2, respectively; RNN performs best in year 2020 Oct and in year 2015, 2017 for case 1 and case 2, respectively; CNN-RNN performs best in year 2010, 2012 and year 2018, 2020 Oct for case 1 and case 2, respectively. The pattern of roc is close to accuracy and loss except years 2012, 2020 Oct for case 1, year 2017 for case 2. My results are different from Yusuf (2018) - his method CNN-RNN reached 64.5% accuracy on the validation data. I think there are two reasons: (1) he didn't split dataset and used files from year 2011 and 2018; (2) he used all 8-K files (with EX-99.1) and he had more unbalanced data on "up" signal while I have more unbalanced data on "stay" signal.



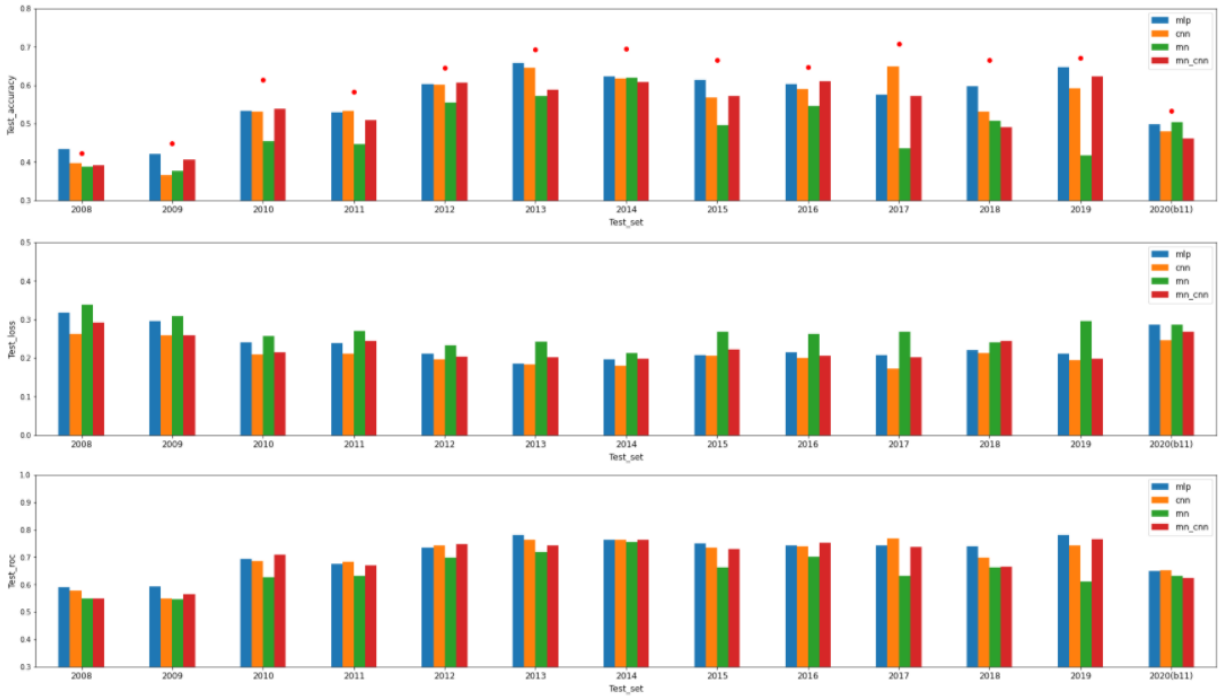
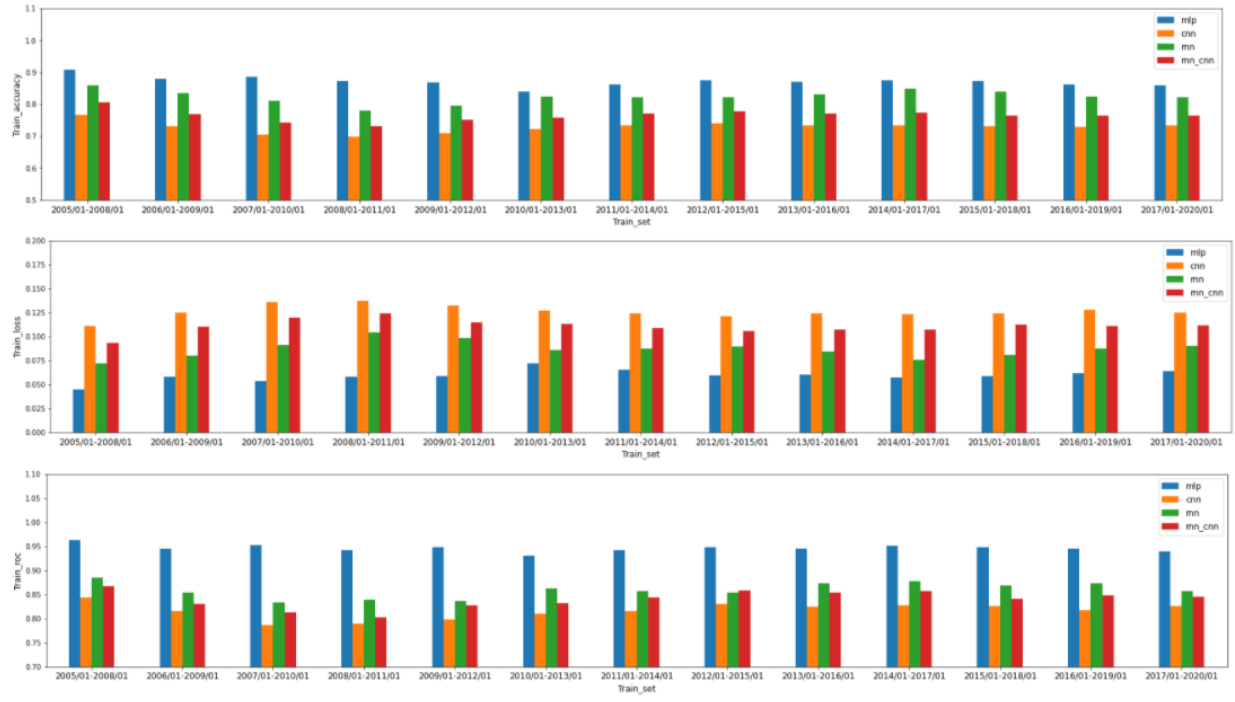


Figure 9. Train and Test Accuracy for machine learning methods for case 1



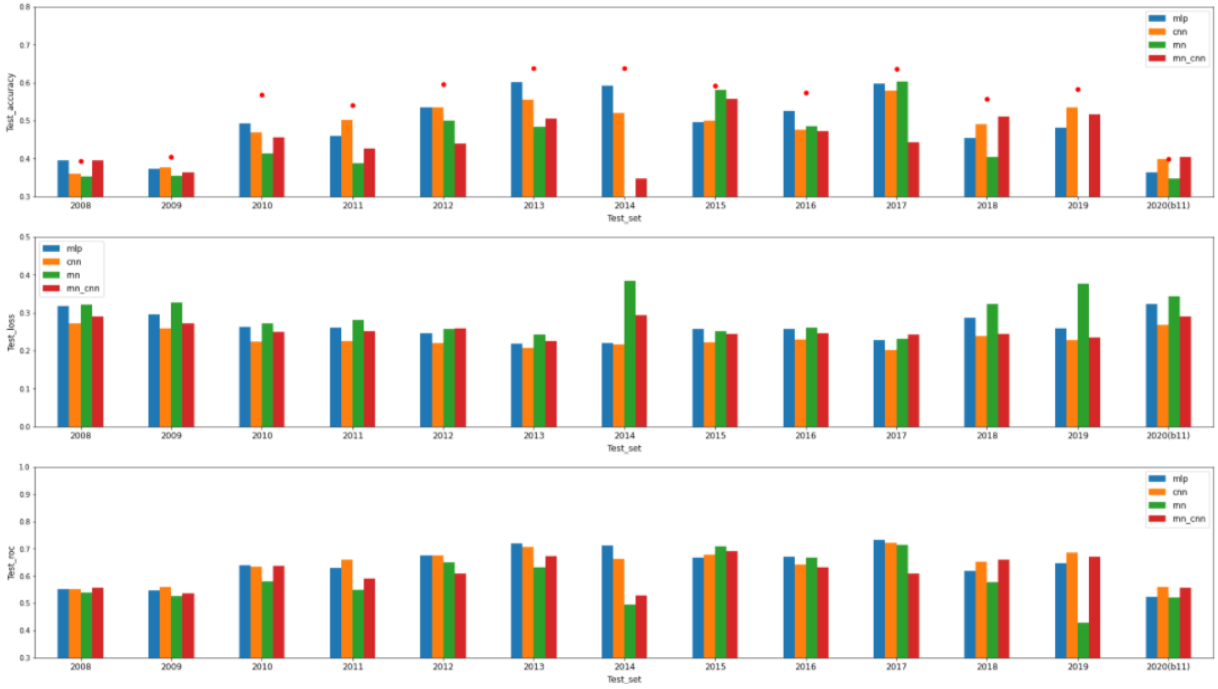


Figure 10. Train and Test Accuracy for deep learning methods for case 2

5. Trading

5.1 Trading Methods

Momentum method was studied by Jegadeesh and Titman (1993) earlier in the trading process. The main idea is to buy the past winners and short the losers. I sort all stocks over the trading period in descending order with the prediction probabilities for up or down signals which are reached by deep learning methods designed in figure 7. Trading process is to long the highest ("expected winner") maximum number of k probabilities' stocks (if it exists), short S&P 500 index with "up" signal and short the lowest ("expected loser") maximum number of k probabilities' stocks (if it exists), long S&P 500 index with "down" signal to make profits.

5.2 Trading Result (K=3)

The comparison of cumulated daily trading returns to S&P 500 for 3 stocks/pair (figure 11) indicates in year 2008, 2009, 2011, 2015, 2018 and 2020 Oct my method (stacking ensemble) outperformed the benchmark of S&P 500 without transaction cost. Beside year 2012, 2013 and 2014, my method achieved positive cumulated daily returns in other 7 years.

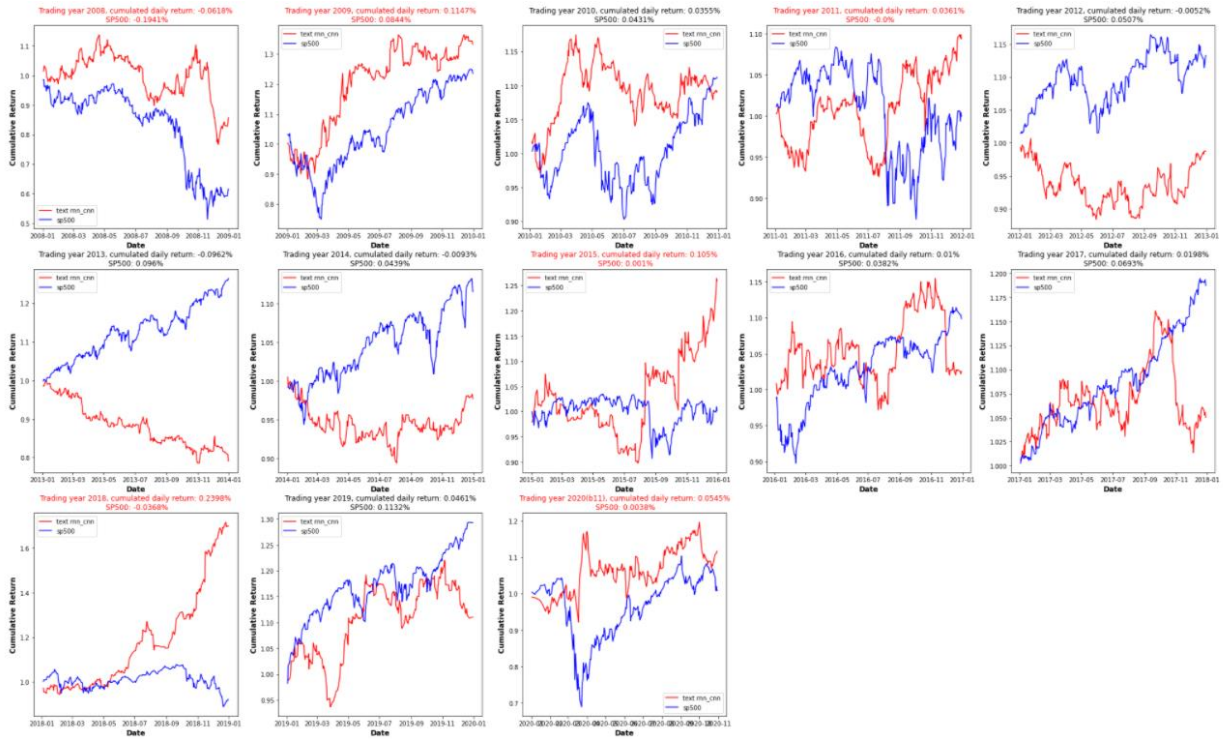


Figure 11. Cumulated Daily Trading Returns vs S&P 500 for 3 stocks/pair with CNN-RNN method

5.3 More Trading Result (K=1,2,3,4,5, all)

To check if the number of stocks in each pair will affect the trading performance, I increase k from 1 to 5 and all possible pairs in figure 12. Except year 2012 and 2017, at least one of my methods (with appropriate k) performed better than S&P 500. This result is better can capstone 1. (Recall in capstone 1, “In years 2013, 2014, 2016, 2017 and 2019, even the best performance of my method got less return than S&P 500”). Performance can be different by different methods in a trading period, such as in year 2020 Oct, mlp with different k(s) cannot beat S&P 500 while CNN, RNN and CNN-RNN can beat S&P 500 with most of k(s). This also confirms in COVID-19 pandemic period, SEC 8-K can be used to predict stock movements and do trading with appropriate method. Good performance of my methods also happened in the last financial crisis (year 2008), most of my method with k(s) can beat S&P 500. Since there is few literatures related to this kind of research with trading process, so we cannot compare the results to others. One thing still needs to recall is in capstone 1, “Krauss et al. (2017) got 0.25% daily return, most of their returns are realized before year 2008 (especially from year 1993 to 2000). In year 2010 to Oct 2015, loss of their capitalization was over 50% with after taking the transaction cost.” Select appropriate methods can decrease the probability of the hypothesis in capstone 1 - “profits are declining in recent years”. I also confirmed there is a severe challenge to the semi-strong form of market efficiency.

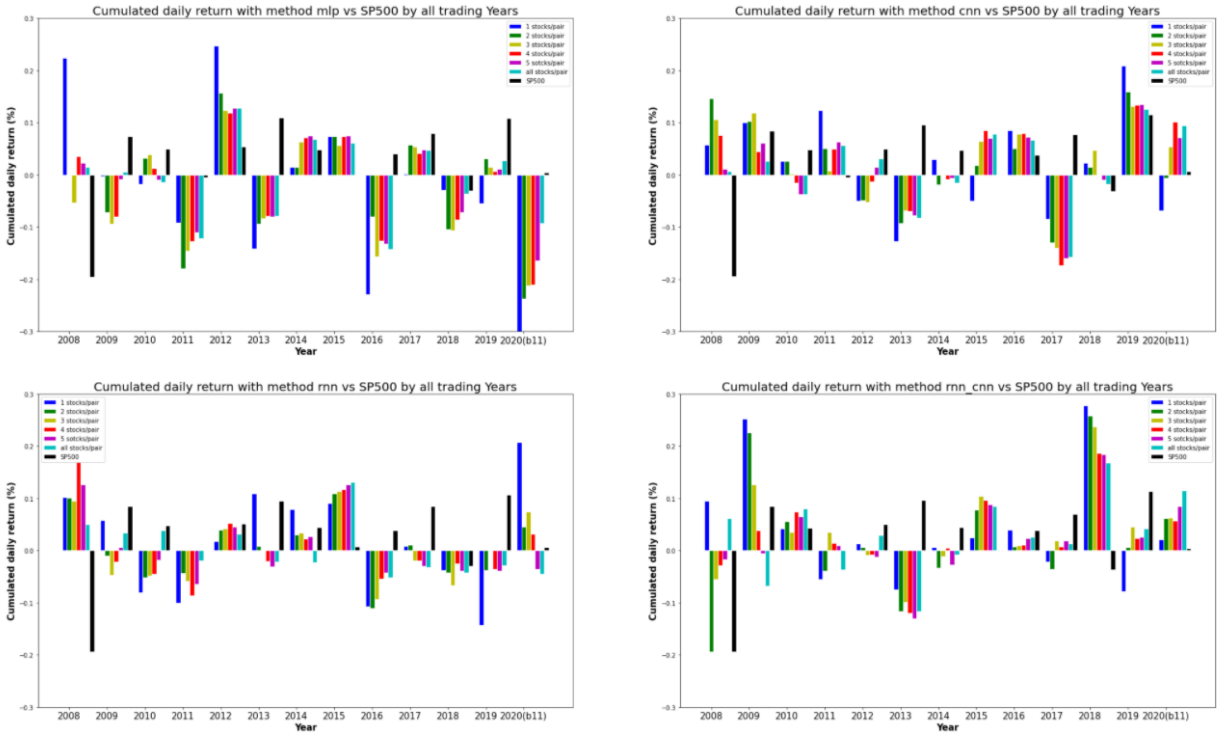


Figure 12. Result of Momentum with four methods by 1,2,3,4, 5 and all possible stocks/pair

5.4 Cumulative Return (CNN-RNN)

We plot the cumulative return without transaction cost (from 1 unit investment at the beginning of year 2008) to the end of 2020 Oct with CNN-RNN method by k=1, 2, 3, 4, 5 and all pairs vs cumulative return with buy and hold S&P 500 in figure 13. The trend for both of my method and buy and hold S&P 500 are upward. We can beat S&P 500 with k=1, 3 and all pairs. For k=4 pairs, cumulative return of CNN-RNN is close to S&P 500. For k=2 and 5, my method is a little bit fall behind S&P 500. For the whole process, except k=1, my method is always above S&P 500, there are several crossover points between my method and S&P 500 for other selections of k.

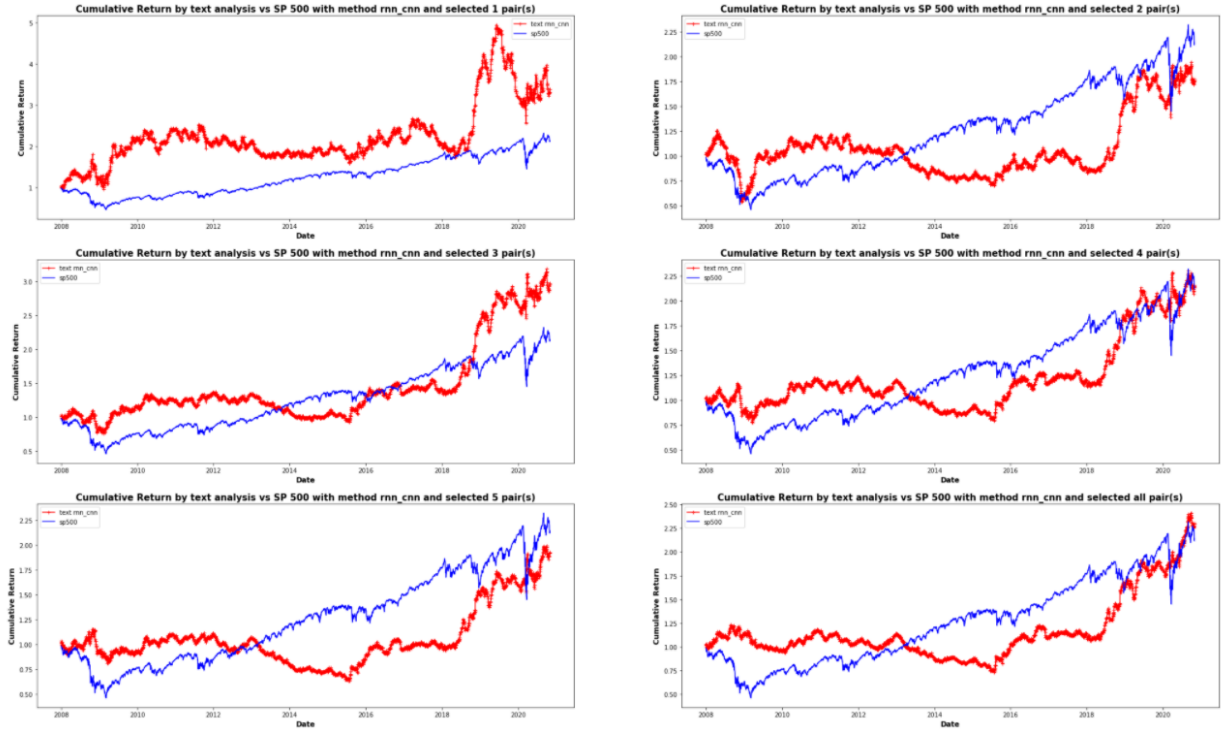


Figure 13. Cumulative return with CNN-RNN method in 15 years and 10 months by 1,2,3,4, 5 and all possible stocks/pair

6. Future Work

There are lots of work can be considered in the future:

- Use more accurate S&P 500 historical components (more time to clean and download data)
- Use other SEC files, such as 10-Q or 10-K.
- For embedding words, use other GloVe files, such as 50, 200, 300 dimensions.
- Try other models such as BERT, GPT 2 or GPT3
- Use different datasets such the Financial Times Stock Exchange (FTSE) 100 or Asian market data with news files.
- Analysis detailly with more criteria such as maximum drawdown and reasons why in the years 2012,2013 and 2014, my method didn't perform well.
- Use ensemble methods with cross validation in train datasets, tune and find the "optimal" parameters, then get the accuracy and trade results in trading datasets.
- Consider transaction fee and leverage.

Reference

Patty Ryan (2017). Stock Market Predictions with Natural Language Deep Learning.

Yusuf Aktan (2018). Using NLP and Deep Learning to Predict Stock Price Movements.

Mark Babbe et al. (2019). BERT is the Word: Predicting Stock Prices with Language Models.

Trist'n Joseph (2020). An investigation into NLP using sentiment analysis to predict Apple stock price movements.