Application of Pre-trained BERT-base Model on Intraday Volatility Prediction of Bitcoin

Rongbing Liang
SID: 3032772810

Ranran Lei
SID: 3036460249

Minxun Xu
SID: 3036369868

Yusong Zhou
SID: 3037249727

October 9, 2021

1 INTRODUCTION

Neural networks for language modeling have been proven effective on several sub-tasks of natural language processing. Training deep language models, however, is time-consuming and computationally intensive. Pre-trained language models such as BERT are thus appealing since they yielded state-of-the-art performance, and they offload practitioners from the burden of preparing the adequate resources (time, hardware, and data) to train models. Most of pre-trained models are generic, pre-trained on large diversified document dataset. There are also some specialized pre-trained models for financial documents, such as FinBERT.

Therefore, pretrained language models such as RoBERTa and FinBERT look like a promising solution for textual financial polarity analysis and semantic representation, which indeed has gaining research popularity for analysis of traditional financial texts like new, analyst reports and official company announcements. Driven by the interest in their application on the quantitative finance domain, we want to step a little bit further to examine whether they could be applied to Twitter massage and enhance the performance of volatility forecasting of financial assets. Twitter post data has greater availability than core financial text data. For each hour, there are many thousands of tweets related to main force financial assets, which could potentially provide relevant information of the market, but it can also be totally noise. Although challenging, we still like to see whether popular sentiment analysis and sentence embedding approaches based on the BERT could work with large, noisy tweet datasets for financial tasks, specifically, predicting the volatility.

As for the targeting financial asset, we choose Bitcoin due to its growing popularity. Although Bitcoin and other cryptocurrencies exhibit significant volatility, the number of main force crypto derivatives is increasing every year. In 2020, CME launched the first exchange-traded Bitcoin option and ETH future on Feb.8, 2021. Therefore, it is increasingly important to understand the source of their volatility and develop an effective model for volatility forecasting. Also, the prices of cryptocurrencies are highly event-driven, which makes the text data of social media predictive for cryptocurrencies volatility.

2 LITERATURES

2.1 FinBERT: Financial Sentiment Analysis with Pre-trained Language Models

In this paper, the hypothesis that pre-trained language models can help with sentiment analysis in a specific domain, like financial sentiment analysis, is made. This is because pre-trained models require less labeled examples and can be further trained on domain-specific corpus.

The implications are interesting. It has been shown that without training using high computational power, the results for a specific task can be better with some complex but pre-trained models. This improved performance is accompanied by the increase in efficiency.

2.2 Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks

This paper uses fine-tuning techniques to BERT in a Siamese network architecture. The main target of the problem is to generate an embedding that could be used for common similarity measures, like the cosine-similarity. BERT performs pretty bad in this task, and so does the RoBERTa, another pre-trained version of BERT. It turns out that Sentecne-BERT is also computationally efficient.

The implication from this paper is the possible advantage of using fine-tuning to some pre-trained models, and it can outperform the original model for some certain tasks. Our project goal is to predict volatility, which is quite different from the original target of popular pre-trained models. Fine-tuning the existing language models might have the improvements needed.

3 DATA

3.1 Data Collection

From 2016 to the current time being, the average total number of Bitcoin-related tweets posted daily is 12,500. Unlike public news, there is no way that all those tweets would be equally representative and influential. Thus, we propose that rather than collect all those tweets, it would be more efficient to only focus on the tweets of top influencers in the Cryptocurrency community because they usually have a more significant impact on people's decision-making. And their tweets are likely to be seen by the public.

Firstly, we built a web scraper to retrieve the information of the top 200 Crypto and Blockchain on Twitter from www.openbusinesscouncil.org/top-200-blockchain-influencers- and https://blockinfluence.com/our-view/the-top-100-crypto-and-blockchain-influencers-on-tw We listed the top 100 influencers by intersection. And then, we gathered all their original tweets and retweets using Twitter Academic API. As all those influencers are specifically focused on cryptocurrency, we did not apply any hashtag or keyword filter. Raw tweet text, timestamps, public metrics were stored. As Twitter truncates tweets over 140 characters, we modified our scraper to collect a full-length version of those tweets. In the end, a total of 847,540 tweets were collected from 1 January 2016 to 1 August 2021. The volume of tweets collected for each date has a similar trend to the total volume of Bitcoin-related tweets, as shown in Figure 1 and 2.

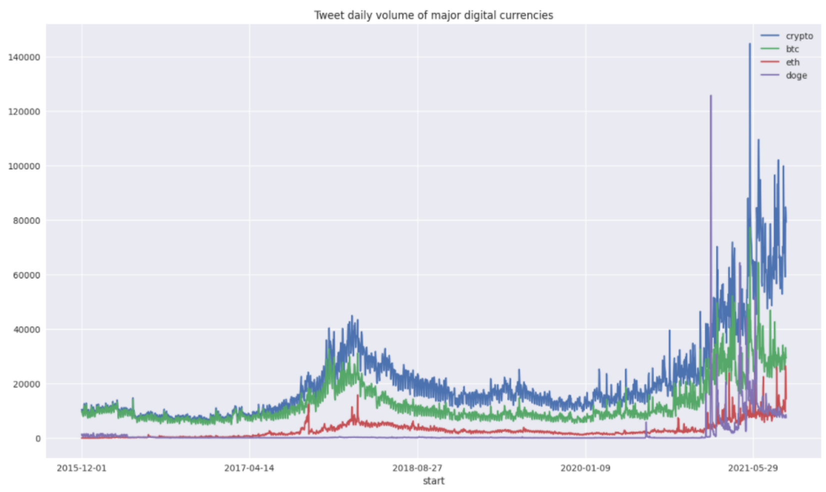


Figure 1: Tweet daily volume of major digital currencies

As for the Bitcoin price data, we download minute level price and volume data of Bitcoin from the Gemini exchange. A total of 2,993,765 minute-level data points were collected from 2015-12-01 to 2021-08-10. To avoid microstructure noise, we resample the data to a 5-minute level. Due to the lack of availability of implied volatility data, we calculated realized volatility of different bin sizes by summing the square of the 5-minute return series. Among them, we chose the 3-hour realized volatility as the proxy of intraday volatility and our research target.

3.2 Pre-processing

Preprocessing was performed on the raw text from each tweet. The cleaning functions are composed of two parts. One managed the removal of tweet-specific

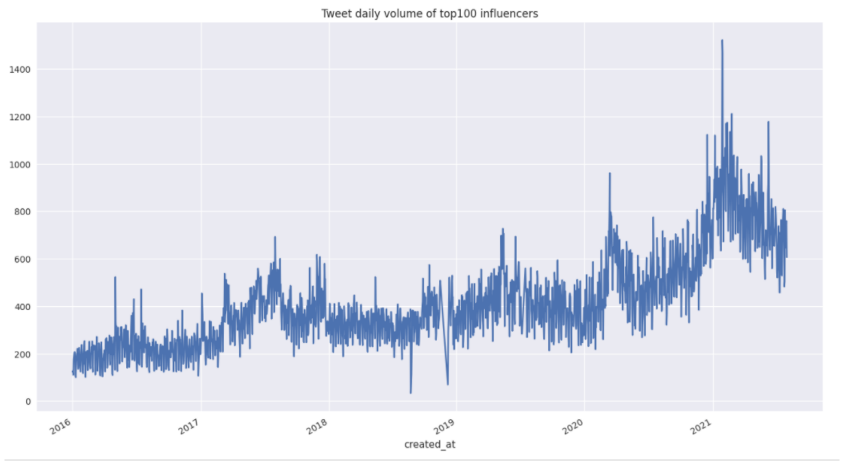


Figure 2: Tweet daily volume of top100 influencers

syntax, and the other was responsible for regular tokenization, including stemming, lemmatization, normalization, and stopwords removal. Since most pretrained language models have their own tokenization function, we only apply the second part to the baseline model.

3.2.1 Tweet-specific Parsing

We used the first 'shallow clean' function to remove unwanted characters and words used specifically on Twitter's platform, such as hyperlinks, numbers, and emojis.

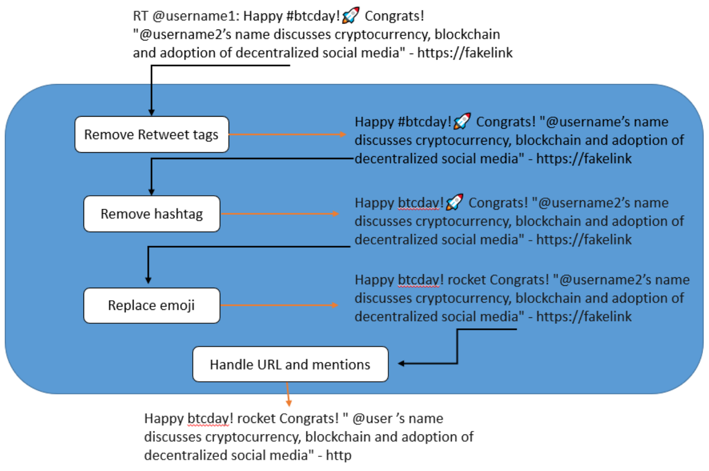

RT @username1: Happy #btcday!🚀 Congrats! "@username2's name discusses cryptocurrency, blockchain and adoption of decentralized social media" - https://fakelink
Remove Retweet tags
Happy #btcday!🚀 Congrats! "@username's name discusses cryptocurrency, blockchain and adoption of decentralized social media" - https://fakelink
Remove hashtag
Happy btcday!🚀 Congrats! "@username2's name discusses cryptocurrency, blockchain and adoption of decentralized social media" - https://fakelink
Replace emoji
Happy btcday! rocket Congrats! "@username2's name discusses cryptocurrency, blockchain and adoption of decentralized social media" - https://fakelink
Handle URL and mentions
Happy btcday! rocket Congrats! " @user 's name discusses cryptocurrency, blockchain and adoption of decentralized social media" - http

Figure 3: Tweet-specific parsing over sample text

3.2.2 Regular Tokenization

- Lemmatization
- Stemming
- Stop-word removal
- Normalization: Handle typos, numbers and email address and other noisy sources

3.3 Data Analysis

3.3.1 Empirical Study

Dep. Variable:	rvol_+3h	**R-squared:**	0.431
Model:	OLS	**Adj. R-squared:**	0.430
Method:	Least Squares	**F-statistic:**	265.0
Date:	Mon, 13 Sep 2021	**Prob (F-statistic):**	3.07e-219
Time:	05:19:39	**Log-Likelihood:**	29301.
No. Observations:	12843	**AIC:**	-5.859e+04
Df Residuals:	12838	**BIC:**	-5.855e+04
Df Model:	4		

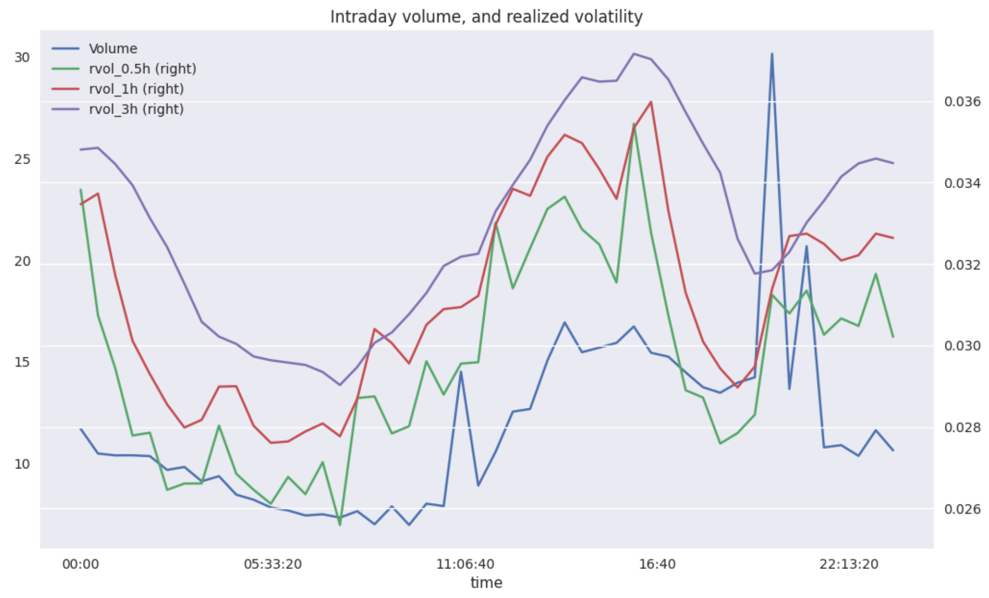


Figure 4: Intraday pattern

	coef	std err	z	P> |z|	[0.025	0.975]
const	0.0113	0.001	12.400	0.000	0.010	0.013
tweet_count	0.0011	0.000	3.735	0.000	0.000	0.002
norm_volume	0.0461	0.012	3.980	0.000	0.023	0.069
hour	-0.0010	0.001	-1.448	0.148	-0.002	0.000
rvol_3h	0.6002	0.033	18.348	0.000	0.536	0.664

Omnibus:	14825.189	**Durbin-Watson:**	2.320
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	6167145.959
Skew:	5.514	**Prob(JB):**	0.00
Kurtosis:	109.785	**Cond. No.**	49.2

Notes:
[1] Standard Errors are heteroscedasticity and autocorrelation robust (HAC) using 5 lags and without small sample correction

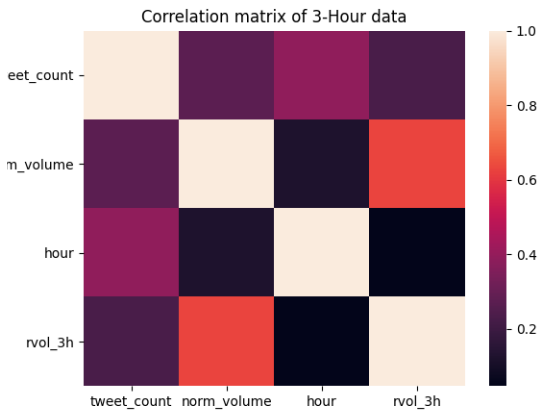


Figure 5: Intraday cross-correlation between possible features

4 Methodology

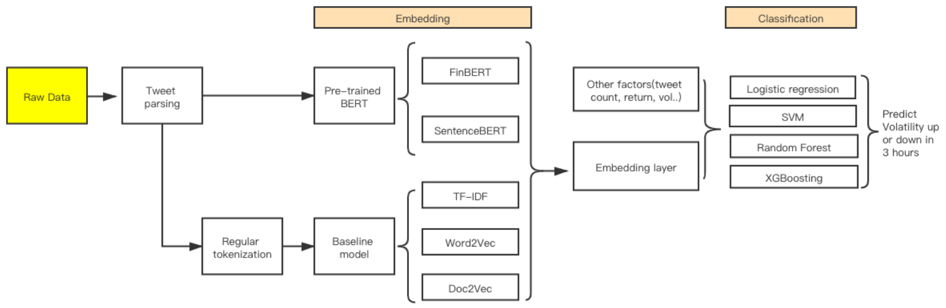


Figure 6: work flow

We will evaluate the pretrained language model by its performance on intraday volatility prediction. To simplify, we will focus on the classification of up and down of intraday volatility of Bitcoin. For the benchmark, we will compare it from two perspectives. First, it will be compared to classical NLP methods to demonstrate its performance in information extraction. Roughly speaking, with the goal to predict the up and down of volatility, if by using pretrained model we gain a significantly better accuracy and F1-score than the baseline methods. Then we may conclude that it does outperform the classical methods.

Secondly, we will study whether sentence embeddings can bring additional edge for us to predict the volatility of Bitcoin. Thus, here we would construct a baseline classification model to incorporate some common stylized factors for intraday volatility, such as autocorrelation and intraday seasonality. And compare its prediction performance with and without all Twitter related features, including sentiment score and metadata. If we also see a significant improvement here, then we will conclude that tweets of top crypto influencers do provide additional information. Also, it is probably feasible to apply a pre-trained NLP model to extract such information so that we have better forecasting on intraday volatility of Bitcoin

We will use four regression models to evaluate the prediction power of different embeddings, including baseline models and BERT-based models. Theseregression models areLogistic Regression,SVM,Random Forest,XGBoost.

4.1 Baseline Method

4.1.1 TF-IDF vectorizer

TF-IDF (term frequency-inverse document frequency) is a statistical measure that evaluates how relevant a word is to a document in a collection of documents. This is the product of two statistics. The intuition behind this statistics is about

$$tf(t,d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}}$$

where $f_{t,d}$ is the raw count of a term in a document, i.e., the number of times that term t occurs in document d. There are various other ways to define term frequency.

$$idf(t,D) = log\frac{N}{count(d \in D : t \in d)}$$

N: total number of documents in the corpus
$count(d \in D : t \in d)$: number of documents where the term t appears

One of the challenges we faced when dealing with TF-IDF is that it does not capture position in text, semantics, co-occurrences in different documents. And for our data set, we have almost 800,000 documents and 200,000 unique words. This will definitely generate dimension explosion. To reduce dimension problem, our group decided to try Sparse PCA and SVD(singular value decomposition) two typical methods. However, compared with Sparse PCA, SVD took less time to reduce dimension. After finishing the embedding, we use three baseline classification methods to test 'TF-IDF' method.

4.1.2 Word2Vec

Word2Vec is a neural network model to learn word associations from a large corpus of text, it can be used to detect similar words or suggest additional words for an incomplete sentence. Technically, a Word2Vec model will map a word to a vector of certain dimension. We use a pretrained model called Glove-twitter-200, glove stands for *global vectors for word representation*, it is trained on twitter data and outputs a vector of 200 dimensions.

We will take several steps to get the features from Word2Vec model. First, we simply apply the model to each words in the sentences, and average them to get a single, 200-dimension vector for each sentence, at each timestamp. Next, we resample the data by time, at a frequency of 3 hours, and take the mean, maximum and minimum of each column for every 3 hour window, after this, we would have a vector of dimension 600, because we perform 3 resample methods for the 200 vector. After that, we perform a PCA for columns, and get an output of vector with dimension 50, for each 3 hour window. And this would be the feature we will use to represent the original texts. To test this model, we will run different classification models, just as the TF-IDF baseline model.

4.1.3 Doc2Vec

The third baseline embedding is dco2vec. In general, doc2vec is very similar to word2vec, But doc2vec tends to capture the feature that word2vec cannot capture, which is the meaning of the whole document. And the whole logistics is the same as what we do in the word2vec except that there is no pre-trained doc2vec embedding. so we train embedding using our own training dataset.

4.2 BERT Method

Compute sentiment score using pretrained RoBERTa base model, ideally, it should be previously trained on Twitter data. Aside from the polarity analysis, we would also experiment with some other approaches to construct features in order to enhance final prediction performance. Here we will also use a pre-trained Sentence-BERT based model and and pre-trained FinBERT model to represent the semantic for each tweet.

5 Result

Below table summarizes the results of three baseline models and BERT-based models. In general XGBoost has the best overall performance with almost more than 60% accuracy for all embeddings. And the basic linear model has a quite robust performance across these embeddings.

	Logistic Regression	Random Forest	SVM	XGBoost
TF-IDF	0.5969	0.5816	0.5603	0.6102
Word2Vec	0.6112	0.5924	0.5864	0.5980
Doc2Vec	0.5842	0.6024	0.6073	0.6045
SentenceBERT	0.5875	0.5756	0.5110	0.6147
FinBERT	0.6020	0.6073	0.5110	0.5976

6 Other Results

Table 1 lists the top 20 influencers, with related statistics regarding public metrics. We can see Elon Musk ranks No.1 with only 22 related tweets. As is implied by following records, count of related tweets is not a key factor in the public metrics.

	count	mean	std	min	25%	50%	75%	max
elon musk	22.0	12.88	0.93	10.03	12.54	13.01	13.43	14.26
davidgerard	393.0	10.67	1.24	5.09	10.02	10.84	11.54	13.66
cz_binance	803.0	8.50	1.29	3.99	7.49	8.16	9.34	12.55
thomaspower	993.0	8.37	2.72	0.00	6.97	9.00	10.22	14.26
APompliano	1690.0	8.28	1.89	1.39	7.35	8.74	9.46	12.39
maxkeiser	1246.0	8.09	1.32	2.83	7.32	8.07	8.88	12.38
justinsuntron	1380.0	8.09	1.38	1.79	7.55	8.22	8.93	12.08
NickSzabo4	1290.0	7.96	2.19	0.69	6.40	8.31	9.70	12.59
twobitidiot	327.0	7.90	2.27	0.00	6.83	7.95	9.35	12.72
haydentiff	1029.0	7.87	2.16	0.00	6.67	7.82	9.13	14.01
prestonjbyrne	297.0	7.81	1.59	0.69	7.01	7.98	8.73	11.81
PeterMcCormack	652.0	7.76	1.41	2.48	6.83	7.69	8.77	12.40
TheCryptoDog	1450.0	7.62	1.54	0.00	6.95	7.80	8.57	12.47
DrJDrooghaag	246.0	7.54	0.77	6.12	7.14	7.33	7.66	12.37
ProfFaustus	516.0	7.47	2.56	0.00	5.74	7.55	9.27	13.10
naval	1337.0	7.37	1.98	0.00	6.05	7.39	8.84	12.36
matthew_d_green	1847.0	7.37	1.70	0.69	6.28	7.32	8.52	12.39
balajis	1088.0	7.32	1.79	0.00	6.44	7.61	8.51	12.30
officialmcafee	1575.0	7.32	1.88	0.00	5.96	7.84	8.72	11.68
brian_armstrong	100.0	6.88	1.31	3.50	6.06	6.86	7.69	10.19

Table 1: Top20 Influencers ranked by public metrics

	followers_count	listed_count	tweet_count	retweet_count	reply_count	like_count
PC						
1	170160.36	3248.57	8811.57	1317.00	3.25	43.23
2	3733324(208559)	7888(3529)	9408(9994)	3493(863)	1659(14)	19491(98)
3	92776.00	2245.00	16248.00	598.22	0.54	3.09
4	190655.00	4204.00	23894.00	911.95	0.87	11.56
5	466695.43	5596.55	7553.40	300.38	24.18	279.97

Table 2: Principal components Group Characteristics including or (excluding) Elon Musk

Dep. Variable:	rvol_+3h	**R-squared:**	0.513
Model:	OLS	**Adj. R-squared:**	0.513
Method:	Least Squares	**F-statistic:**	492.6
Date:	Fri, 24 Sep 2021	**Prob (F-statistic):**	0.00
Time:	05:29:51	**Log-Likelihood:**	37264.
No. Observations:	15701	**AIC:**	-7.451e+04
Df Residuals:	15690	**BIC:**	-7.442e+04
Df Model:	10		

	coef	**std err**	**z**	**P> |z|**	**[0.025**	**0.975]**
const	0.0030	0.001	2.526	0.012	0.001	0.005
tweet_count	0.0014	0.000	4.395	0.000	0.001	0.002
norm_volume	0.0507	0.010	5.255	0.000	0.032	0.070
rvol_3h	0.3365	0.024	13.990	0.000	0.289	0.384
log_impact	0.0004	0.000	3.153	0.002	0.000	0.001
author_pct	-0.0111	0.004	-2.654	0.008	-0.019	-0.003
L1.rvol_3h	0.1001	0.014	7.215	0.000	0.073	0.127
L2.rvol_3h	0.1218	0.021	5.682	0.000	0.080	0.164
L3.rvol_3h	0.0741	0.017	4.325	0.000	0.041	0.108
L4.rvol_3h	0.1046	0.027	3.819	0.000	0.051	0.158
L5.rvol_3h	0.0611	0.018	3.463	0.001	0.027	0.096

Omnibus:	21086.709	**Durbin-Watson:**	2.009
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	12617909.959
Skew:	7.294	**Prob(JB):**	0.00
Kurtosis:	141.110	**Cond. No.**	550.

Table 3: OLS Regression Results

Notes:
[1] Standard Errors are heteroscedasticity and autocorrelation robust (HAC) using 6 lags and without small sample correction

7 Conclusion

For our three baseline NLP models, Word2Vec and Doc2Vec have no significant difference over simple TF-IDF, and these models have their corresponding weaknesses in terms of our task. First, it is hard for TF-IDF method to handle extreme large corpus, therefore, specific dimension reduction techniques for sparse matrix such as truncated SVD is required. Also, inappropriate aggregation and resampling may result in underperformance of Word2Vec. And the self-trained doc2vec does not have improved performance at the document level.

For classification models, logistic regression and XGBoost have the overall best performance among the four classification models. But the performance of linear model is more robust and stable. One possible explanation is that the distribution of the data is not good.

For feature importance in the classification task, due to the persistence of volatility, the lag term of realized volatility is always the most important one among all features. Some principal components from embedding vectors are as important as other common features such as trading volume and tweet volume, implying that these embeddings have certain degrees of prediction power over the future volatility, thus illustrating that at least some part of the embeddings extract valuable information from texts.

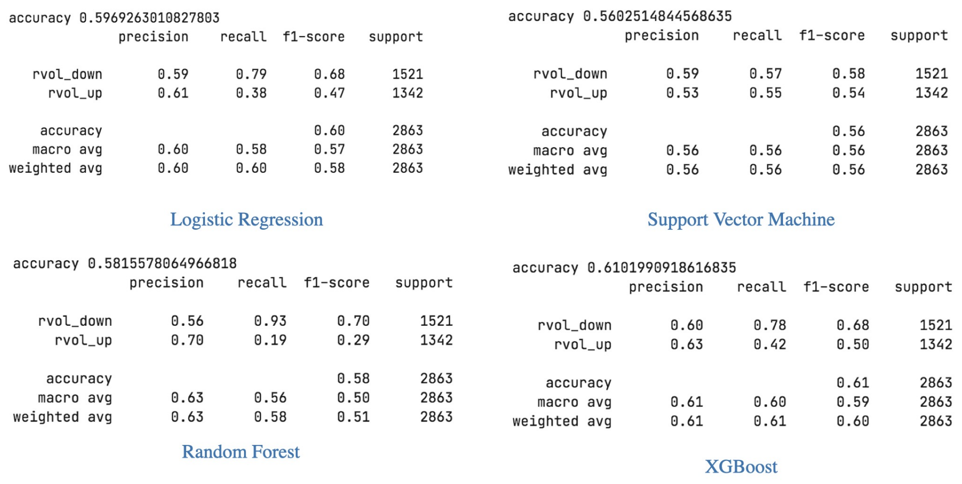
For BERT-based models, SentenceBERT does not outperforms our baselines for every classification models. Actually it has bad accuracy for Random Forest and SVM. However, the highest accuracy in our result is achieved by Sentence-BERT with XGBoost. So to some extent this embedding does have the strongest prediction power.

8 REFERENCES

Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. ArXiv Preprint ArXiv:1908.10084.

Araci, D. (2019). FinBERT: Financial Sentiment Analysis with Pre-trained Language Models. ArXiv, abs/1908.10063.

A Baseline performance



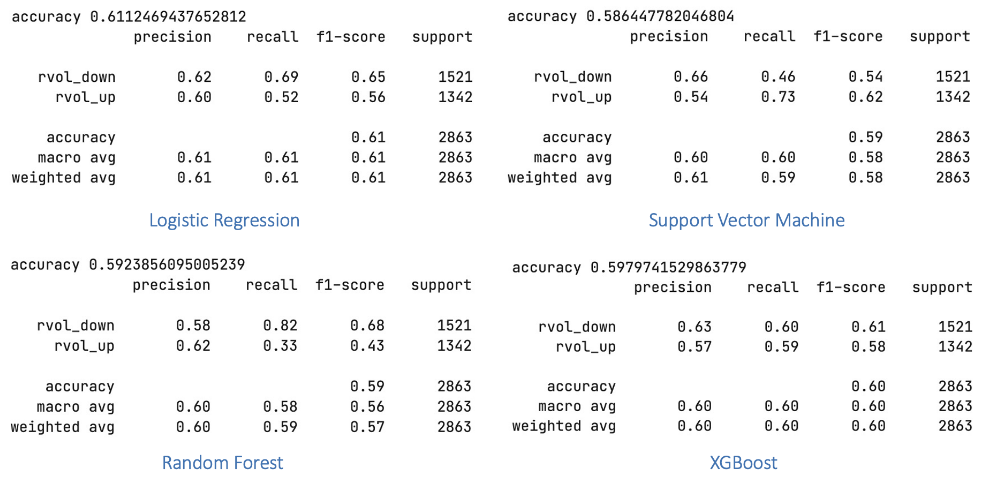
```
accuracy 0.5969263010827803
              precision    recall  f1-score   support

   rvol_down       0.59      0.79      0.68      1521
     rvol_up       0.61      0.38      0.47      1342

    accuracy                           0.60      2863
   macro avg       0.60      0.58      0.57      2863
weighted avg       0.60      0.60      0.58      2863
```

```
accuracy 0.5602514844568635
              precision    recall  f1-score   support

   rvol_down       0.59      0.57      0.58      1521
     rvol_up       0.53      0.55      0.54      1342

    accuracy                           0.56      2863
   macro avg       0.56      0.56      0.56      2863
weighted avg       0.56      0.56      0.56      2863
```

```
accuracy 0.5815578064966818
              precision    recall  f1-score   support

   rvol_down       0.56      0.93      0.70      1521
     rvol_up       0.70      0.19      0.29      1342

    accuracy                           0.58      2863
   macro avg       0.63      0.56      0.50      2863
weighted avg       0.63      0.58      0.51      2863
```

```
accuracy 0.6101990918616835
              precision    recall  f1-score   support

   rvol_down       0.60      0.78      0.68      1521
     rvol_up       0.63      0.42      0.50      1342

    accuracy                           0.61      2863
   macro avg       0.61      0.60      0.59      2863
weighted avg       0.61      0.61      0.60      2863
```

Figure 7: Classification Results with TF-IDF vectorizer



```
accuracy 0.6112469437652812
              precision    recall  f1-score   support

   rvol_down       0.62      0.69      0.65      1521
     rvol_up       0.60      0.52      0.56      1342

    accuracy                           0.61      2863
   macro avg       0.61      0.61      0.61      2863
weighted avg       0.61      0.61      0.61      2863
```

Logistic Regression

```
accuracy 0.586447782046804
              precision    recall  f1-score   support

   rvol_down       0.66      0.46      0.54      1521
     rvol_up       0.54      0.73      0.62      1342

    accuracy                           0.59      2863
   macro avg       0.60      0.60      0.58      2863
weighted avg       0.61      0.59      0.58      2863
```

Support Vector Machine

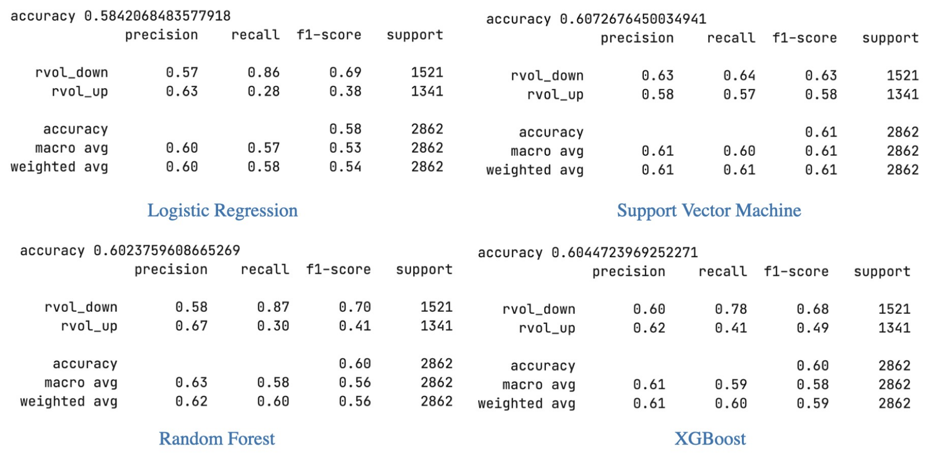
```
accuracy 0.5923856095005239
              precision    recall  f1-score   support

   rvol_down       0.58      0.82      0.68      1521
     rvol_up       0.62      0.33      0.43      1342

    accuracy                           0.59      2863
   macro avg       0.60      0.58      0.56      2863
weighted avg       0.60      0.59      0.57      2863
```

Random Forest

```
accuracy 0.5979741529863779
              precision    recall  f1-score   support

   rvol_down       0.63      0.60      0.61      1521
     rvol_up       0.57      0.59      0.58      1342

    accuracy                           0.60      2863
   macro avg       0.60      0.60      0.60      2863
weighted avg       0.60      0.60      0.60      2863
```

XGBoost

Figure 8: Classification Results with Word2Vec Embeddings



```
accuracy 0.5842068483577918
              precision    recall  f1-score   support

   rvol_down       0.57      0.86      0.69      1521
     rvol_up       0.63      0.28      0.38      1341

    accuracy                           0.58      2862
   macro avg       0.60      0.57      0.53      2862
weighted avg       0.60      0.58      0.54      2862
```

Logistic Regression

```
accuracy 0.6072676450034941
              precision    recall  f1-score   support

   rvol_down       0.63      0.64      0.63      1521
     rvol_up       0.58      0.57      0.58      1341

    accuracy                           0.61      2862
   macro avg       0.61      0.60      0.61      2862
weighted avg       0.61      0.61      0.61      2862
```

Support Vector Machine

```
accuracy 0.6023759608665269
              precision    recall  f1-score   support

   rvol_down       0.58      0.87      0.70      1521
     rvol_up       0.67      0.30      0.41      1341

    accuracy                           0.60      2862
   macro avg       0.63      0.58      0.56      2862
weighted avg       0.62      0.60      0.56      2862
```

Random Forest

```
accuracy 0.6044723969252271
              precision    recall  f1-score   support

   rvol_down       0.60      0.78      0.68      1521
     rvol_up       0.62      0.41      0.49      1341

    accuracy                           0.60      2862
   macro avg       0.61      0.59      0.58      2862
weighted avg       0.61      0.60      0.59      2862
```

XGBoost

Figure 9: Classification Results with Doc2Vec Embeddings