FRE7871 News Analytics and Machine Learning: Project Report

Quantitative Equity Trading Signal Based on Financial News Headlines Sentiment Classification

Group Member: Siyang Huang (sh5958), Tianjing Cao (tc3385), Zhuoran Ma (zm2161)

I. Project Goal

Apply NLP techniques to building financial sentiment classification with various fine-tuned supervised machine learning models, then utilize the predicted signals in quant equity trading.

- First, label the financial headlines from Kaggle in two ways: using next-day stock daily returns from Yahoo Finance, and a pre-trained transformer pipeline of sentiment analysis.
- Second, after researching literature and pre-processing text, obtain 2 word embedding inputs with different NLP tools, mainly TF-IDF Vectorizer and BERT Transformer.
- Third, train different classifiers (Naïve Bayes & Random Forest) through cross validation and hyperparameter tuning with 2 types of target labels. (More on Logistic, SVM and TCN later).
- Fourth, evaluate all models (trained with 2 inputs of word embedding X and 2 types of sentiment label Y) on the validation set and report on test set results.
- Finally, implement a systematic trading strategy, using predicted sentiment labels from high-accuracy models as buying or selling stock signals, to test and compare profit and loss.

II. Project Motivation

Machine learning (ML) and Natural Language Processing (NLP) have been developing for wide use in many areas. In this project, we applied and compared various ML and NLP techniques through building sentiment classification models on recent financial news headlines data from Kaggle (2020). We referred to huggingface (2022) transformers pipeline and Sentence Transformer pretrained model (Reimers, 2022) to conduct NLP like text preprocessing, sentiment analysis, and word embedding using both TF-IDF (Term Frequency-Inverse Document Frequency) and BERT (Bidirectional Encoder Representations from Transformers). Apart from above, we also fine-tuned (Tran, 2020) different ML models, including Naive Bayes and Random Forest, to see how we can efficiently classify the sentiment of financial big data, and transform such sentimental results into quantitative trading signals.

III. Project Report

3.1 Data Preparation

After reading in the 'raw_partner_headlines.csv' file from Kaggle, we grouped the financial news headlines by year to find the starting date (2010-02-03) and ending date (2020-06-04), where we deleted one piece of outlier news from 1969.

In order to train supervised machine learning models, we labeled the headlines in two different ways for comparison. First, we used the headline's corresponding stock daily return on corresponding date as label, where we marked return > 0.1 as label 1 (positive) and return < -0.1 as label 0 (negative). Second, we simply implemented sentiment analysis on headlines using huggingface transformers pipeline, which is an encapsulated API dedicated to carry out NLP tasks and produce inference outputs.

For the first way (Label Type I), we needed to download the stock close price from Yahoo Finance and calculate the daily return of news' next day in the timeframe as presented in the notebook. We dropped the news headlines with relatively small next day return (between -0.1 and 0.1) because significant financial news would usually cause large fluctuations on stock returns, and therefore, we could also reduce computing amount and enhance accuracy.

For the second way (Label Type II), we used the pipeline of sentiment-analysis with FinBERT (Araci, 2019) which is a pre-trained NLP model. It is built by further training the BERT language model in finance domain, using a large financial corpus, and thereby fine-tuning it for financial sentiment classification. This process was quite slow so it would be much faster reading local file, where we had 'positive' and 'negative' sentiments and dropped 'neutral' labels to concentrate on stronger sentiments and decrease calculation. Finally, in this project, we split the whole data in actual date order as training, validation, and test sets with percentages about 60%, 13% and 27%, respectively.

3.2 Baseline Model: TF-IDF Vectorizer + Naive Bayes Classifier

When utilizing TF-IDF vectorizer, we first should preprocess raw text (Agrawal, 2021) by a few typical steps, including expanding contractions, lowercasing headlines, removing punctuations, removing digits, lemmatization, and removing extra whitespaces. Next, within TF-IDF Vectorizer, we analyzed words of 1 to 3 n-grams and removed English stop words to transform headline texts into vectors, for example, from X_train to X_train_tfidf.

As for Naive Bayes Classifier, we could tune the alpha smoothing parameter (from 0.1 to 10 by step 0.1) for multinomial Naive Bayes model, which is suitable for classification with discrete features such as TF-IDF, through comparing average AUC score (Area Under the ROC Curve) from cross-validation (KFold = 5) on training set.

With the best alpha determined, we evaluated the multinomial Naive Bayes model on validation set and produced ROC Curve (Receiver operating characteristic Curve), AUC score, and accuracy score. Last but not least, we applied the best model on test set to report confusion matrix and comprehensive classification results. This 3-step model training process: hyperparameter tuning, validation set evaluation and report on test set, would be carried out and compared for each following model in section 3.3 and section 3.4.

In the notebook, Baseline Model 1 and 2 stand for employing TF-IDF Vectorizer and Multinominal Naive Bayes Classifier on different Label Type I and II, whose model details and results are shown as below (Type I on the left and Type II on the right).

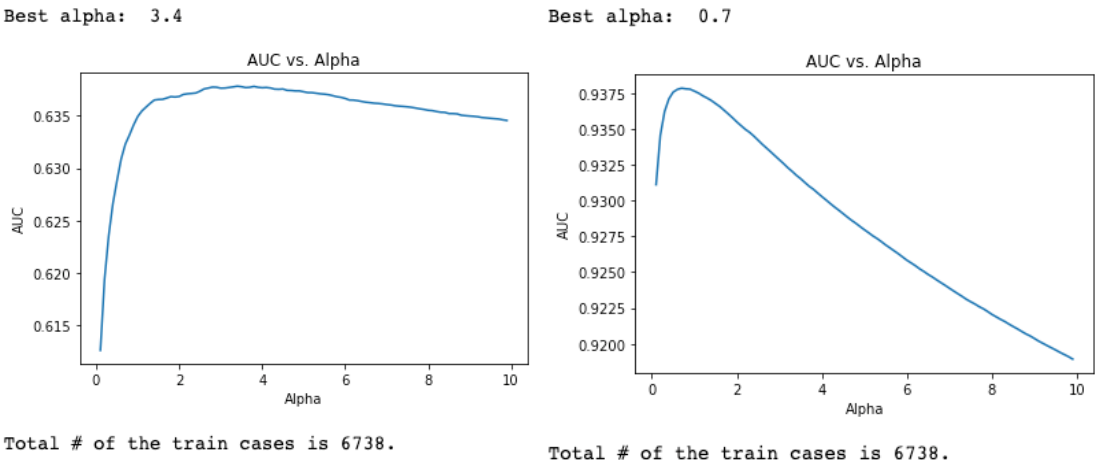


Figure 1 - MultinomialNB: Hyperparameter tuning of Alpha (Label Type I and II)

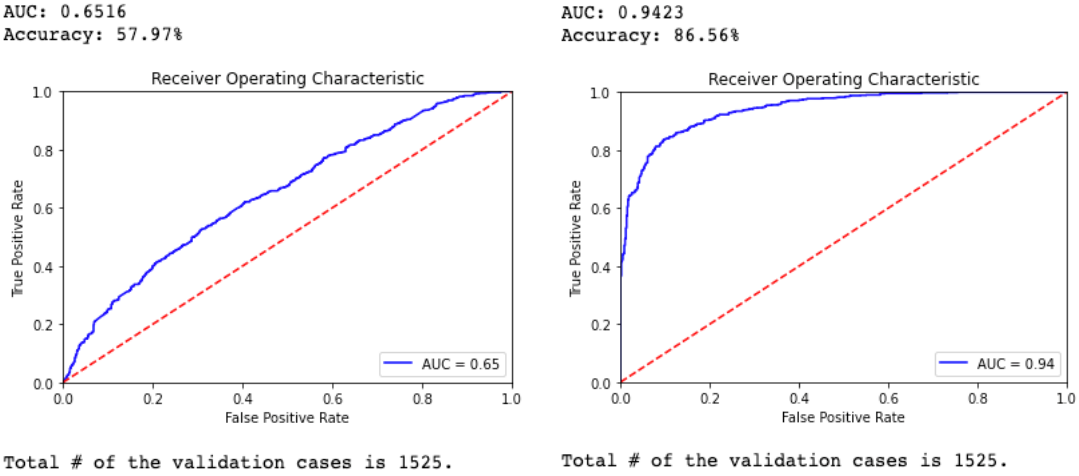
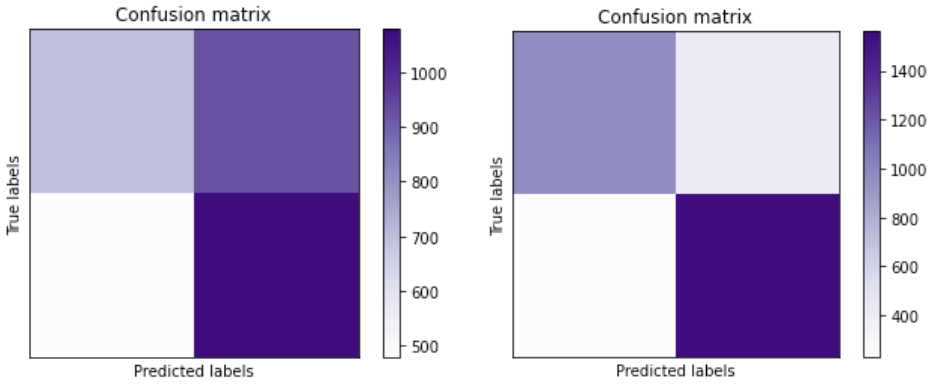


Figure 2 - MultinomialNB: ROC Curve Evaluation on Validation Set (Label Type I and II)

From *Figure 1* hyperparameter tuning and *Figure 2* evaluation results, we can see that the optimal alpha for Baseline Model 1 and 2 are 3.4 and 0.7, while their accuracy scores on validation set are around 58% and 87%, respectively. Moreover, the validation AUC of Baseline Model 2 is as high as 94%, while that of Baseline Model 1 is only about 65%.

From *Figure 3* confusion matrix and *Figure 4* classification reports, it can be inferred that the target labels in test set are more balanced for Label Type I than Type II. Therefore, for imbalanced data, we should also compare the AUC or weighted average scores rather than only focus on accuracy score. Nevertheless, Label Type II produces much better outcomes than Type I, which is logically reasonable. Since the stock return will be affected by many other factors, simply using the next day return as a sentiment label of today's news is too fluctuating and unstable.




```
====== Confusion Matrix ======
[[ 694  925]
 [ 479 1079]]
```

```
====== Confusion Matrix ======
[[ 963  423]
 [ 231 1560]]
```

Figure 3 - MultinomialNB: Confusion Matrix on Test Set (Label Type I and II)

====== Classification Report ======			Alpha	Label
Model: MultinomialNB			3.4	Type I
Labels & Scores	precision	recall	f1-score	support
0	59.16%	42.87%	49.71%	1619
1	53.84%	69.26%	60.58%	1558
accuracy	55.81%	55.81%	55.81%	55.81%
macro avg	56.50%	56.06%	55.15%	3177
weighted avg	56.55%	55.81%	55.04%	3177

====== Classification Report ======			Alpha	Label
Model: MultinomialNB			0.7	Type II
Labels & Scores	precision	recall	f1-score	support
0	80.65%	69.48%	74.65%	1386
1	78.67%	87.10%	82.67%	1791
accuracy	79.41%	79.41%	79.41%	79.41%
macro avg	79.66%	78.29%	78.66%	3177
weighted avg	79.53%	79.41%	79.17%	3177

Figure 4 - MultinomialNB: Classification Report on Test Set (Label Type I and II)

In conclusion, the quality of Label Type II is much better than Type I for training with TF-IDF vectorizer and Multinomial Naive Bayes model, reaching a test accuracy score of 79.41% compared with 55.81%. To see if BERT transformer could improve performance, we utilized random forest classifier in the next part because MultinomialNB model fails when features have negative values.

3.3 BERT Model: Sentence Transformer + Random Forest

Devlin, Chang, Lee and Toutanova (2019) introduced a new language representation model called BERT, which stands for Bidirectional Encoder Representations from Transformers. BERT is designed to pretrain deep bidirectional representations from unlabeled text by jointly conditioning on both left and right contexts in all layers. To transform text data (headlines) using BERT language model into vectors, we utilized the SentenceTransformer according to SBERT.net with pretrained 'all-MiniLM-L6-v2' model. (Note: We don't have to preprocess text here since BERT is pretrained on full text.) After BERT sentence encoding, we dumped the data as local pickle files to save time.

In this case, we chose Random Forest Classifier to fine-tune n_estimators (from 100 to 300 by step 50) and max_depth (from 5 to 50 by step 5) parameters in the classification model of Label Type I and II, denoted as BERT Model 1 and 2. Same as previous model, we continued to do validation set evaluation and test set report. From the results in *Figure 5* below, we set n_estimators=300 in both cases, while the max_depth is slightly different. Their accuracy scores on validation set are around 58% and 88%, respectively. Furthermore, the validation AUC of BERT Model 2 is much higher at 94.41%, while that of BERT Model 1 is only 63.80%, which confirms that Label Type II is more reliable.

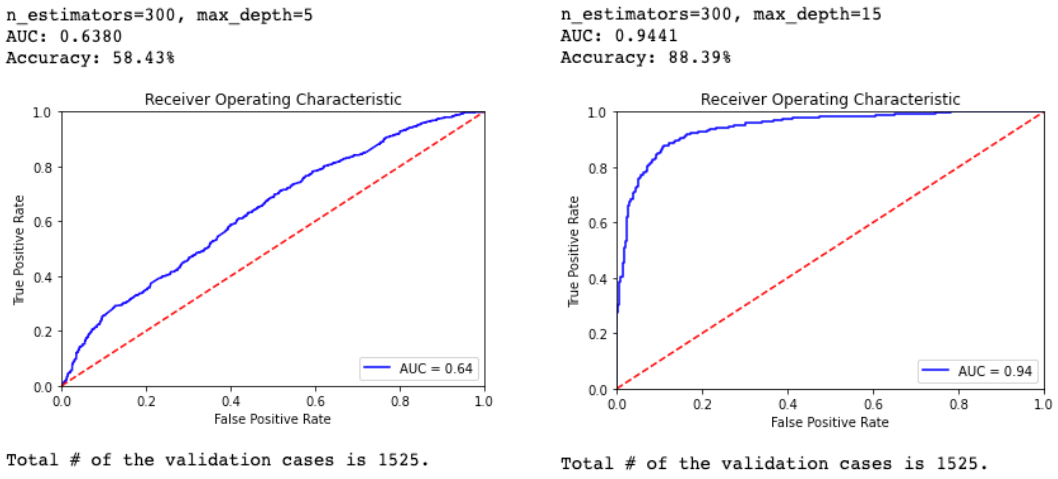


Figure 5 - Random Forest: ROC Curve Evaluation on Validation Set (Label Type I and II)

=== Classification Report ===		n_estimator	max_depth	Label
Model: Random Forest		300	5	Type I
Labels & Scores	precision	recall	f1-score	support
0	58.08%	43.30%	49.61%	1619
1	53.40%	67.52%	59.64%	1558
accuracy	55.18%	55.18%	55.18%	55.18%
macro avg	55.74%	55.41%	54.62%	3177
weighted avg	55.78%	55.18%	54.53%	3177

=== Classification Report ===		n_estimator	max_depth	Label
Model: Random Forest		300	15	Type II
Labels & Scores	precision	recall	f1-score	support
0	81.01%	80.95%	80.98%	1386
1	85.27%	85.32%	85.29%	1791
accuracy	83.41%	83.41%	83.41%	83.41%
macro avg	83.14%	83.13%	83.14%	3177
weighted avg	83.41%	83.41%	83.41%	3177

Figure 6 - Random Forest: Classification Report on Test Set (Label Type I and II)

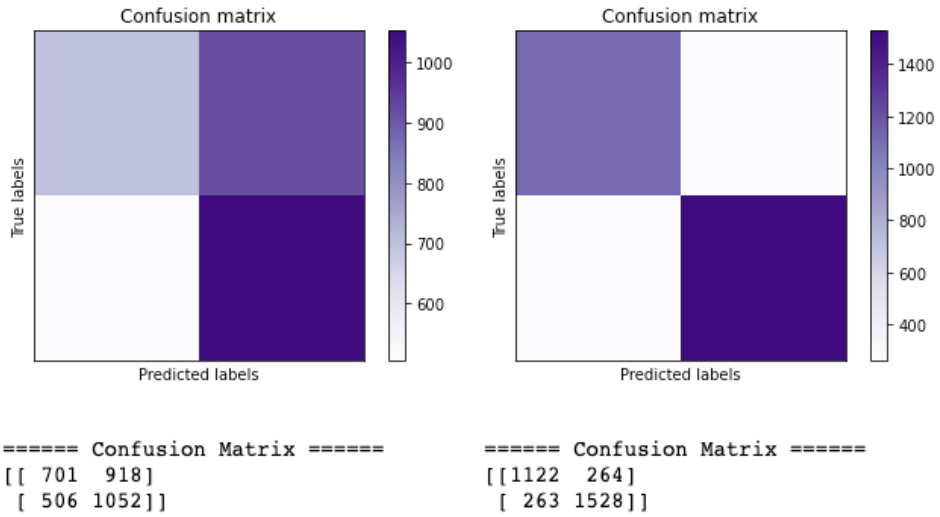


Figure 7 - Random Forest: Confusion Matrix on Test Set (Label Type I and II)

From the classification reports above in *Figure 6*, we can tell that Label Type I prediction accuracy on test set becomes a bit worse than applying TF-IDF vectorizer, but Label Type II test accuracy improved by 4% with BERT transformer. However, for BERT Model 2 on Label Type II, the recall and f1-score of 'negative' class (label=0) are still under-performing than 'positive' class (label=1), similar to the result in previous Baseline Model 2. Therefore, further adjustments can be made to decrease the probability of wrongly recognizing negative news as positive, as revealed in the confusion matrix of *Figure 7*, to enhance model performance.

3.4 Different models on Label Type II

In this section, we also tried another 4 machine learning models on the better-performing Label Type II, including Logistic Regression, Support Vector Machine (SVM), and Temporal Convolutional Network (TCN), to compare model accuracy, ROC Curve and AUC in *Figure 8 and Figure 9.*

Model Test Accuracy Score on **Label Type II**			
TF-IDF Vectorizer	MultinomialNB	RandomForest	Logistic
	79.41%	Worse Result	80.17%
	GaussianNB	SVM	TCN
	70.32%	81.78%	~ 80%
BERT Transformer	MultinomialNB	RandomForest	Logistic
	Not Suitable	83.41%	86.81%
	GaussianNB	SVM	TCN
	64.75%	87.72%	~ 85%

Figure 8 - Accuracy Scores on Test Set for Label Type II with both TF-IDF and BERT word embedding

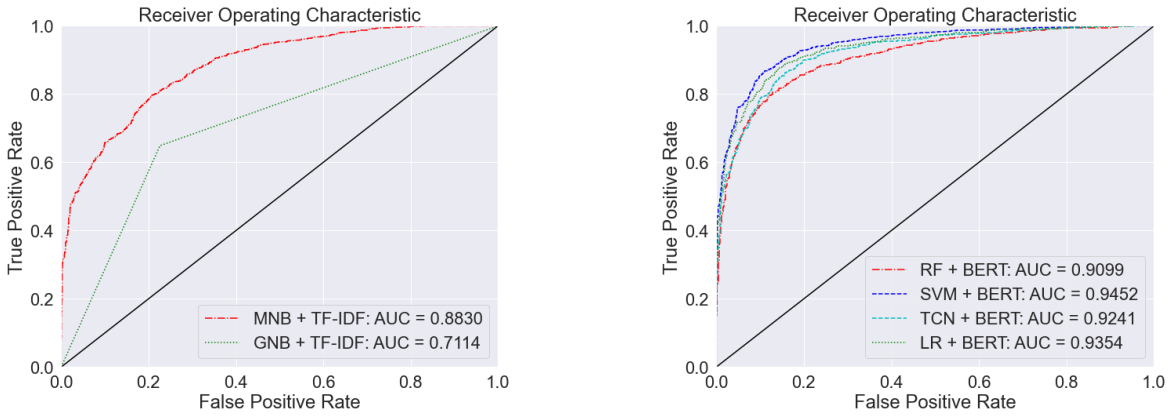


Figure 9 - ROC Curve and AUC on Test Set for Label Type II with better-performing word embedding

For the Logistic Regression, Gaussian Naive Bayes and SVM, we simply used their default parameters, such as solver = 'lbfgs' and kernel='rbf'. For TCN with BERT word embedding, we set the input shape (6738, 1, 384) and create a sequential model with TCN layer, Dense layers with Sigmoid activation, and Dropout layer with a rate of 0.5 to resolve overfitting. Next, we compile the model with 'binary_crossentropy' loss function, 'adam' optimizer and accuracy metrics.

To conclude from *Figure 8*, simple Naive Bayes models would be more effective when applying TF-IDF vectorizer than BERT transformer, but both have a lower accuracy compared to other classifiers. With pretrained BERT transformer, more complex models like Logistic, Random Forest, SVM and TCN could improve the accuracy score by a few percent. From the perspective of AUC, which is a more appropriate choice in evaluating imbalance targets, most models in *Figure 9* have similar results except for Gaussian Naive Bayes. It is noteworthy that SVM with BERT transformer reaches the highest for both AUC of 94.52% and accuracy of 87.72%, leaving much space to modify.

Test Example Headlines	TF_IDF + LR	BERT + SVM	Manual
Yelp EPS misses by $0.05, beats on revenue	0	0	0
Dow Jones Industrial Average Pops After The Latest Trade Move By Chin	0	1	1
China responds to new U.S. tariff threat	0	0	0
JOYY (YY) Q4 Earnings Beat Estimates, Revenues Rise Y/Y	1	1	1
Will Portfolio Strength & User Growth Aid MSCI Q2 Earnings?	1	1	-
Buy 6 Top Stocks Set to Beat on Earnings in Q2 Today	1	1	1
Zynga (ZNGA) Q4 Earnings Miss Estimates	0	0	0
Zumiez Curbs Costs & Extends Store Closure Amid Coronavirus	0	0	0
Zumiez (ZUMZ) Q3 Earnings and Revenues Beat Estimates	1	1	1
American Eagle (AEO) Slumps Despite Q2 Earnings & Sales Beat	1	0	0

Figure 10 - Test Examples: <TF-IDF + Logistics Regression> & <BERT + SVM>

In addition, we presented some test examples in *Figure 10* above, using TF-IDF + Logistic Regression and BERT + SVM model to predict the sentiment label on financial news headlines. Considering manual neutral news as wrong prediction, it can be implied that 7 out of 10 in Logistic Regression are accurate, and SVM outperforms with 9 out of 10 in accuracy for these ten examples.

To sum up, we can say that using stock return as a sentiment label of financial news headlines (Label Type I) is impractical. Besides the method of utilizing sentiment analysis pipeline to obtain Label Type II in our project, more reliable label marking techniques or manual tagging could be researched and implemented. Nevertheless, all above models could be further fine-tuned to enhance performance on expanded available data source with more computing power and time.

3.5 Quantitative Equity Trading Signal

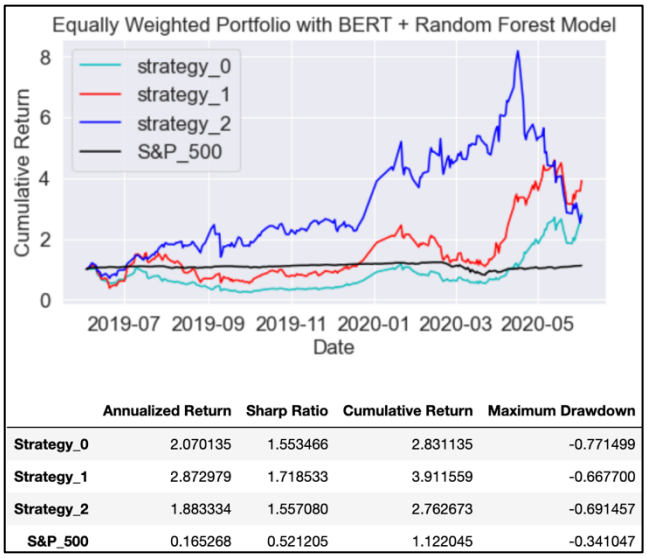
For the last part of our project, we implemented a systematic equity trading strategy on test set headlines trained on Label Type II, and stocks with the 1-year investment horizon from 2019-06-02 to 2020-06-02, using the predicted positive sentiment label from high-accuracy models as buying stock signal (+1), and negative as selling stock signal (-1), to test and compare profit and loss.

First, we needed to determine the trading signal for each stock since there could be multiple headlines for the same stock every day. Through voting or adding up all headline signals for each stock today, we obtained the "**Signal_Sum**" as the sentiment reference for the next day. From here, we designed 3 signals of strategy and compare them with S&P 500 index benchmark. **Signal_0** is to buy on the next day if news appears today for a stock, which is the simple baseline strategy. **Signal_1** is to buy when "Signal_Sum" is positive but not long nor short otherwise. **Signal_2** is to long if "Signal_Sum" is positive and short if negative, no trading when "Signal_Sum" equals to 0. All three strategies used today's sentiment label as tomorrow's trading signal and cleared position on a daily basis, and each day, we trade only the stocks with news meaning the signals are discrete.

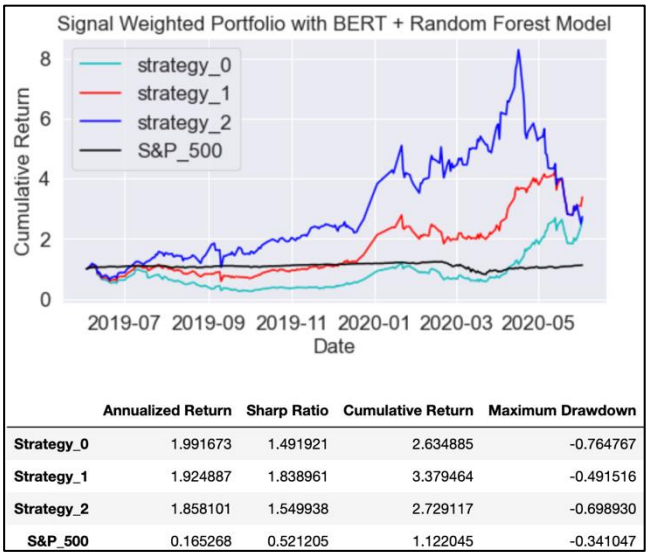
Second, after combining signals with daily returns, we calculated the weight of each stock in the strategy portfolio with two methods. **Equally weighted** means to buy or sell the stocks at the same weight on the same day, totaling 100% every day. **Signal weighted** assumes that more news indicates stronger sentiment, so we will long or short stocks weighted on "Signal_Sum" to capture extra returns, where the daily stock weights still sum up to 100%.

Finally, each of the 3 portfolios' daily returns is calculated as the weighted average return of individual stocks that had news in one day. Then, we plotted the cumulative return, obtained annualized return, Sharpe ratio and maximum drawdown to evaluate performance. In the followings, we implemented BERT + Random Forest and BERT + SVM models with relatively higher test accuracy to predict news sentiment and backtest quantitative equity trading signals as discussed above.

3.5.1 BERT + RF and BERT + SVM Model with Discrete Signal



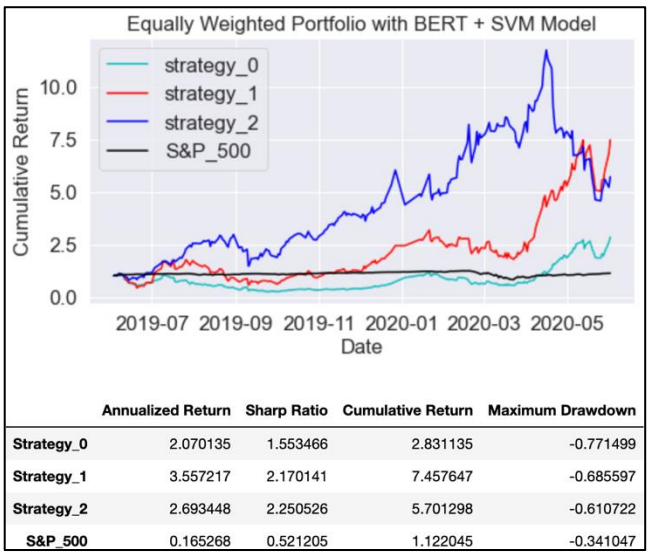
	Annualized Return	Sharp Ratio	Cumulative Return	Maximum Drawdown
Strategy_0	2.070135	1.553466	2.831135	-0.771499
Strategy_1	2.872979	1.718533	3.911559	-0.667700
Strategy_2	1.883334	1.557080	2.762673	-0.691457
S&P_500	0.165268	0.521205	1.122045	-0.341047



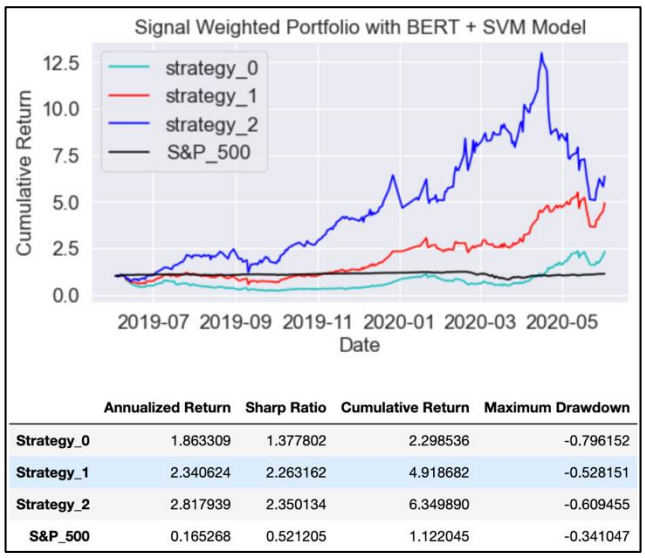
	Annualized Return	Sharp Ratio	Cumulative Return	Maximum Drawdown
Strategy_0	1.991673	1.491921	2.634885	-0.764767
Strategy_1	1.924887	1.838961	3.379464	-0.491516
Strategy_2	1.858101	1.549938	2.729117	-0.698930
S&P_500	0.165268	0.521205	1.122045	-0.341047

Figure 11 - Equally and Signal Weighted Portfolios with BERT + RF Model

With BERT + Random Forest Model, the equally weighted portfolios perform better overall than signal weighted ones, except for the Sharpe and MDD of equally weighted Strategy_1 which is more volatile. In both cases, Strategy_1 is the best-performing strategy over the one-year investment with Sharpe at about 1.7 and a cumulative return higher than 3, followed by Strategy_0. The shorting selling Strategy_2 here is slightly weaker than Strategy_0, and all three strategies beat S&P 500.



	Annualized Return	Sharp Ratio	Cumulative Return	Maximum Drawdown
Strategy_0	2.070135	1.553466	2.831135	-0.771499
Strategy_1	3.557217	2.170141	7.457647	-0.685597
Strategy_2	2.693448	2.250526	5.701298	-0.610722
S&P_500	0.165268	0.521205	1.122045	-0.341047



	Annualized Return	Sharp Ratio	Cumulative Return	Maximum Drawdown
Strategy_0	1.863309	1.377802	2.298536	-0.796152
Strategy_1	2.340624	2.263162	4.918682	-0.528151
Strategy_2	2.817939	2.350134	6.349890	-0.609455
S&P_500	0.165268	0.521205	1.122045	-0.341047

Figure 12 - Equally and Signal Weighted Portfolios with BERT + SVM Model

For BERT + SVM Model, Strategy_1 outperforms other portfolios when equally weighted. Nevertheless, short selling Strategy_2 becomes the best signal weighted one with Sharpe over 2.3 and cumulative return over 6. Similarly, all three strategies outperform S&P 500 benchmark, while Strategy_1 and Strategy_2 both surpass the baseline Strategy_0 more significantly with SVM classifier.

In conclusion, seeing from the indicators on return and risk, Strategy_1 is a more stable choice and BERT + SVM model could greatly improve strategy performance. On the other hand, the plots suggest Strategy_2 had been the best-performing strategy, reaching a cumulative return of over 10. Although SVM classifier caught the rising trend in the last few trading days, such a dramatic fall in April and May of 2020, when Strategy_2 incorrectly shorted growing stocks, led to irreparable losses.

3.5.2 BERT + RF and BERT + SVM Model with Continuous Tracking

Built on the above discrete signals (Singnal_0, Singnal_1, Singnal_2), we tried another strategy which is continuous tracking on individual stock level to evaluate the performance of classification models statistically. **Strategy_0** as a baseline is where we buy and hold 1 share of stock for the whole one-year investment period. Incorporating financial news sentiment classification into Strategy_0, we have **Strategy_1** which is to close the long position on negative Signal_1, and **Strategy_2** which is selling 1 share of stock on negative Signal_2.

BERT + RF Compared to Strategy_0	Strategy_1	Strategy_2
% of Stocks Return Enhanced	8.1336	8.1578
% of Stocks Return the Same	84.5074	84.5074
% of Stocks Return Worsen	7.3590	7.3348

BERT + SVM Compared to Strategy_0	Strategy_1	Strategy_2
% of Stocks Return Enhanced	8.3757	8.3999
% of Stocks Return the Same	84.0232	84.0232
% of Stocks Return Worsen	7.6011	7.5769

Figure 13 - Continuous Individual Stock Tracking with BERT + RF / SVM Model

With BERT + Random Forest Model, we found that out of 4131 stocks with available return data, around 8.14% of the one-year cumulative returns are enhanced by Strategy_1 and Strategy_2, compared to Strategy_0 baseline. Most stocks (about 84.5%) remained the same under Strategy_1 and Strategy_2, while nearly 7.34% of the cumulative returns became worse than buy and hold. It means that such trading signals from news sentiment classification could correctly detect dropping stock prices but also might wrongly miss or make losses on rising stocks.

As for BERT + SVM Model, both the percentages of stocks improved and deteriorated increased by over 0.2%, while the percentage of stock unchanged decreased by 0.5%. The difference between these two models is actually minor when comparing the effectiveness of continuous tracking on individual stock here. However, since BERT + SVM Model has much better back-testing results in Section 3.5.1, the return enhancement brought by Strategy_1 and Strategy_2 is more significant than the return downside caused by wrongly predicted signals. Therefore, we should notice that it is very critical to further reduce the error rate so as to improve portfolio performance.

IV. Reference

[1] Araci, D. (2019). Finbert: Financial sentiment analysis with pre-trained language models. *arXiv preprint arXiv:1908.10063*.

[2] Agrawal, R. (2021). Must Known Techniques for text preprocessing in NLP. Retrieved 28 September 2022, from https://www.analyticsvidhya.com/blog/2021/06/must-known-techniques-for-text-preprocessing-in-nlp/

[3] Kaggle (2020). Daily Financial News for 6000+ Stocks. Retrieved 28 September 2022, from https://www.kaggle.com/datasets/miguelaenlle/massive-stock-news-analysis-db-for-nlpbacktests?resource=download&select=raw_partner_headlines.csv

[4] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

[5] Huggingface. (2022). Pipelines . Retrieved 28 September 2022, from https://huggingface.co/docs/transformers/main_classes/pipelines

[6] Reimers, N. (2022). Quickstart — Sentence-Transformers documentation. Retrieved 28 September 2022, from https://www.sbert.net/docs/quickstart.html

[7] Tran, C. (2020). Tutorial: Fine-tuning BERT for Sentiment Analysis - by Skim AI. Retrieved 28 September 2022, from https://skimai.com/fine-tuning-bert-for-sentiment-analysis/