

Using News to Predict Stock Movements - Live Kaggle Competition

Group 19

Charu Sreedharan

Anirban Sen

Tigmanshu Chaudhary

1. Abstract

- Stock movement prediction using news is a Kaggle Competition to predict how stocks will change based on the market state and news articles.
- The aim of this project is to analyze a long series of trading days. For each day, we receive an updated state of the market, and a series of news articles which were published since the last trading day, along with impacted stocks and sentiment analysis.
- This information is then used to predict whether each stock will have increased or decreased ten trading days into the future. So, ultimately, we have to predict a confidence value between -1 and 1. The target variable to predict here is: **'returnsOpenNextMktres10'** (market-residualized return 10 days into the future).

2. Introduction

The movements in the stock market are affected by multiple factors such as the industry's performance, the company's performance, the economic status of the country etc. Similarly news articles also play a major role in influencing the movement of these stock prices. News articles provide information regarding the company and the industry's activities, performance and status, it also provides an estimate or predictions of their performance in the future. As investors rely on past and current news while making investments, it can be said that news articles is one of the major factors that influence stock behavior.

3. Related Work

What we noticed in most of the work related to this topic is that research papers tend to focus on how to extract sentiments (positive, negative or neutral) from news articles.

But, we don't need to mine sentiments from news, as they are already part of the dataset provided to us.

1) Using News Articles to Predict Stock Price Movements - Győző Gidófalvi [1]

The author proposes that short-term stock price movements can be predicted using financial news articles. He assigns a label - up, down or unchanged to each news article according to the movement of the associated stock compared to its expected movement by aligning news articles to stock prices and scoring them based on the relative movement of the stock price during the window of influence. He then employs Bayesian text classifier to predict which class an article belongs to. Although he gets low classifier scores using this method, he finds a strong correlation between news articles and the stock price behavior within a 20-minute window around the news article's publication time.

2) Forecasting to Classification: Predicting the direction of stock market price using Xtreme Gradient Boosting - Shubharthi Dey, Yash Kumar, Snehanshu Saha, Suryoday Basak [2]

This research paper uses ensemble learning methods for stock prediction. The authors observe that ensemble learning models have been unexplored in stock forecasting. The work flow diagram the paper used is: Data Collection -> Exponential Smoothing -> Feature Extraction -> Ensemble Learning -> Stock Market Prediction . They use +1 for indicating the rise in stock valuation and -1 to indicate the fall in the prices in the future. XGBoost reported the highest accuracy of 88%. They infer that they got the highest accuracy with XGBoost because the problem is not linearly separable and hence the entire suite of SVM type classifiers or related machine learning algorithms did not work very well.

3) The Predicting Power of Textual Information on Financial Markets - Gabriel Pui Cheong Fung, Jeffrey Xu Yu, Hongjun Lu [3]

This paper designed a system which employed a t-test based split and merge segmentation algorithm to filter out the noise of the movement over the stock time series. The words in a news document act as features and they are weighted using term frequency (i.e. the number of times

f_i appears in D) and document frequency (i.e. the number of documents containing f_i), where f_i is the i th feature in the Document(D).

4) Using Text and Data Mining Techniques to extract Stock Market Sentiment from Live News Streams - Anurag Nagar, Michael Hahsler [4]

This paper proposed an automated text mining based approach to collect news stories from various sources and then filter it down to relevant sentences using Natural Language Processing (NLP) techniques. A sentiment metric is computed by measuring the count of positive and negative polarity words. They conclude that the time variation of News Sentiment has a very strong correlation with the observed stock price movement.

5) Predicting the Direction of Stock Market Price Using Tree Based Classifiers - Suryoday Basak, Snehanshu Saha, Saibal Kar, Luckyson Khaidem, Sudeepa Roy Dey [5]

The research paper employs tree-based classifiers to predict whether stock prices will increase or decrease. The traditional forecasting model is reformulated as a classification problem. Random Forests and XGBoost classifiers are used to build the predictive model and their model outperforms the models used in other research papers where ensemble learning methods are not used. Random Forest and XGBoost outperform SVM in terms of accuracy (close to 92% on an average, across different stocks).

6) Predicting Stock Price Movements Based on Different Categories of News Articles - Yauheniya Shynkevich, T.M. McGinnity, Sonya Coleman, Ammar Belatreche [6]

This research paper studies how financial forecasting can be improved with the simultaneous usage of different financial news categories. News articles are classified into five categories of relevance to a target stock and they are analyzed using separate kernels. Their results illustrate that the simultaneous use of five news categories improves the prediction performance when compared to methods involving a lower number of categories. The Multiple Kernel Learning (MKL) technique showed much better prediction performance when compared to other methods like SVM and KNN.

7) XGBoost: A Scalable Tree Boosting System - Tianqi Chen, Carlos Guestrin [7]

This paper focuses on what makes this scalable tree boosting system so popular. The above mentioned paper [Tianqi Chen] addresses the key problem in tree learning of determining the best split. It is called gradient descent as it uses a gradient descent algorithm to reduce to disparity of predicted values from actual values when adding successive models.

8) The Impact of News on Stock Market - Xiqian Zhao, Juan Yang, Lili Zhao & Qing L [8]

This paper gives us the conclusion that different news articles have a different impact on the stock movement, but all online news articles initially have a temporary negative impact on the price. It was also found that different stock exchanges are affected at different levels due to news articles, it is not uniform.

4. Dataset

We worked with two datasets: market and news.

The training sets of these two datasets have records between 2007 and 2016, while the test sets have records between 2017 and 2018.

The market training dataset has 4072956 records and 16 features. Market data is provided by Intrinio.

Market dataset has the following columns:

time	volume	returnsOpenPrevRaw1	returnsClosePrevMktres10
assetCode	close	returnsClosePrevMktres1	returnsOpenPrevMktres10
assetName	open	returnsClosePrevRaw10	returnsOpenNextMktres10
universe	returnsClosePrevRaw1	returnsOpenPrevRaw10	returnsOpenPrevMktres1

- assetCode,assetName: Identifiers for Companies and their different assets
- open/close-stock price at the start and end of that day
- prev/next:returns either looking into past or future days
- Mktrs/Raw:returns adjusted with some metrics or raw

The news training dataset has 9328750 records and 35 features. News data is provided by Thomson Reuters. The news dataset has
Following are the columns of News dataset:

provider	sentenceCount	sentimentNegative	noveltyCount5D
subjects	wordCount	sentimentNeutral	noveltyCount7D
audiences	assetCodes	sentimentPositive	volumeCounts12H
bodySize	assetName	sentimentWordCount	volumeCounts24H
companyCount	firstMentionSentence	noveltyCount12H	volumeCounts3D
headlineTag	relevance	noveltyCount24H	volumeCounts5D
marketCommentary	sentimentClass	noveltyCount3D	volumeCounts7D

- sentimentClass: points to the predominant sentiment class for that particular news item
- sentimentNegative/Positive: probability that sentiment belongs to either class
- noveltyCount: novelty of contents of news items over difference amounts of time (12hrs,3 days and so on)
- volumeCount: aggregate volume of news content belonging to that particular asset over different periods of time(12hrs,3 days and so on)
- provider: the news provider of the news item
- headlineTag: the headline tag for a particular news item
- marketCommentary: boolean value indicating certain market factors
- Urgency: classifies the news stories urgency (1:alert, 3 :article)
- Relevance:number indicating the relevance of news item to the assets.

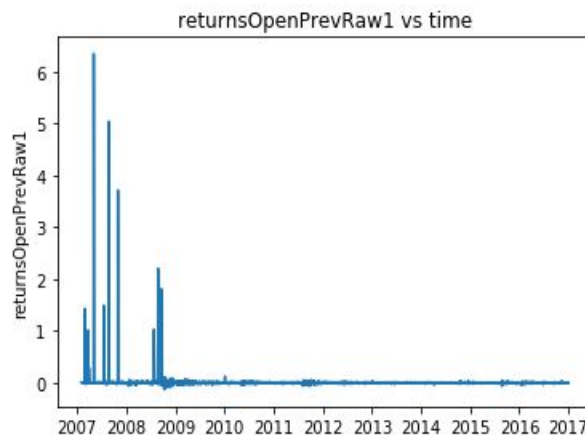
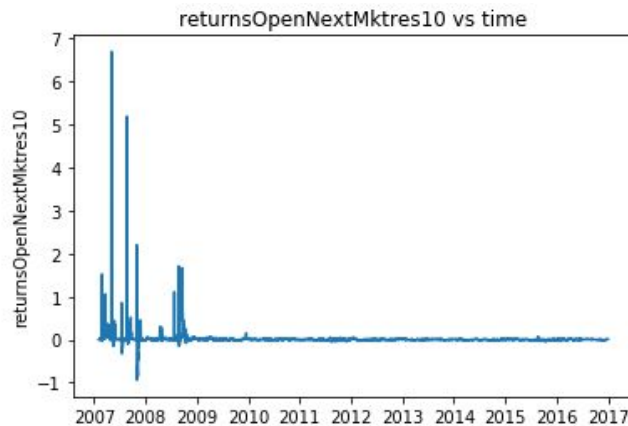
5. Solution

- Stocks can either go up or down. So, first, we predict whether the stock will go up or down. So the first part is basically binary classification [5]. After which we

compute the probability of the stock going up through 'predict_proba' function which comes with XGBoost (up probability). Then, we rescale the up probability which is between (0,1) to a new range (-1,1), which is the range of the target variable. If we know our model confidence for the stock to go up, then our new confidence value is:

$$\text{Final predicted confidence value} = (2 * \text{up probability}) - 1$$

6. Preprocessing



- Looking at the statistics, most data behave homogeneously after 2009 (volume increase, price increase, etc.). But, before 2009, due to the housing crisis that led to the financial crisis in 2008, the data behaves differently. So the question to make the right prediction for this problem is: Was there a financial crisis between 2017 and 2018 (Testset range)? As, the answer is No, we only consider records after January 2009.

	time	assetCode	assetName	volume	close	open	returnsClosePrevRaw1	returnsOpenPrevRaw1
627547	2008-09-29 22:00:00+00:00	BK.N	Bank of New York Mellon Corp	18718479.0	26.5	3288.1136	-0.271578	99.125262
1127598	2010-01-04 22:00:00+00:00	TW.N	Towers Watson & Co	223136.0	50.0	9998.9900	-0.058470	185.988360

- We clip records in the market dataset where the difference between open and close price is very high(Figure above). So, we only consider records where close price to open price ratio is in the range of 0.5 to 1.5. This is because, for instance, there might have been splits or mergers of a particular asset during the course of the day which might have caused an extreme close-to-open ratio. Thus, removing the high close-to-open ratio seems to be a good idea, so as not to confuse the model.

```

Check null data:

time                0
assetCode           0
assetName           0
volume              0
close               0
open                0
returnsClosePrevRaw1  0
returnsOpenPrevRaw1  0
returnsClosePrevMktres1 15980
returnsOpenPrevMktres1 15988
returnsClosePrevRaw10  0
returnsOpenPrevRaw10   0
returnsClosePrevMktres10 93010
returnsOpenPrevMktres10 93054
returnsOpenNextMktres10  0
universe            0
dtype: int64

```

- All null values in the market dataset are from market-adjusted columns, as can be seen above. So, we fill them up with raw values of the corresponding row. The difference between raw and market-adjusted returns is that the data is not adjusted against any benchmark in the case of the former. While market-residualized (Mktres) returns ensures that movement of the market as a whole has been accounted for.

- Market return should not exceed 50% or fall below 50%. If it does, it is more likely to be noise or extreme data that can confuse our model.

	time	assetCode	assetName	volume	close	open	returnsClosePrevRaw1	returnsOpenPrevRaw1
3787451	2016-05-19 22:00:00+00:00	EBRYY.OB	Unknown	0.0	1.96	1.96	0.0	0.0
3789272	2016-05-20 22:00:00+00:00	EBRYY.OB	Unknown	0.0	1.96	1.96	0.0	0.0
3791098	2016-05-23 22:00:00+00:00	EBRYY.OB	Unknown	0.0	1.96	1.96	0.0	0.0
3792928	2016-05-24 22:00:00+00:00	EBRYY.OB	Unknown	0.0	1.96	1.96	0.0	0.0
3794755	2016-05-25 22:00:00+00:00	EBRYY.OB	Unknown	0.0	1.96	1.96	0.0	0.0
3796586	2016-05-26 22:00:00+00:00	EBRYY.OB	Unknown	0.0	1.96	1.96	0.0	0.0
3798415	2016-05-27 22:00:00+00:00	EBRYY.OB	Unknown	0.0	1.96	1.96	0.0	0.0
3800244	2016-05-31 22:00:00+00:00	EBRYY.OB	Unknown	0.0	1.96	1.96	0.0	0.0
3802075	2016-06-01 22:00:00+00:00	EBRYY.OB	Unknown	0.0	1.96	1.96	0.0	0.0
3803903	2016-06-02 22:00:00+00:00	EBRYY.OB	Unknown	0.0	1.96	1.96	0.0	0.0

- We remove records where the asset code is 'EBRYY.OB', as the asset name for this particular asset code is almost always 'Unknown', as can be seen above.
- We clip extreme records in the news Dataset for columns like 'bodySize', 'sentenceCount', 'wordCount' by considering only those falling between the 2nd and 98th quantile. Those lying outside this range are very few in number and are likely to confuse the model.
- Relevance of a news item indicates the significance of the news item to an asset. News items can be of 2 types: alert or article. We assume that a news item published on a day will be relevant over multiple days based on the relevance of it. Range of the relevance column is between 0 and 1. If the news about a particular asset appears in the headline, then the relevance of it is maximum(1). If the news item is of type article, we convert the range of relevance from 0 to 1 to 0 to 7, then we replicate the news item over the number of days obtained from the new relevance.

While, if the news item is of type alert, we obtain relevance using the formula:

$$\text{Relevance} = (\text{sentenceCount} - (\text{firstMentionSentence} - 1)) / \text{sentenceCount}$$

where sentenceCount is the total number of sentences in the news item and firstMentionSentence is the first sentence, (starting with the headline), in which

the scored asset is mentioned.

Then, we convert the range of this newly obtained relevance from 0 to 1 to 0 to 10, and then replicate the news item over the number of days obtained from the new relevance. But we couldn't implement this code due to frequent kernel crashes as a result of memory failure.

- Irrelevant columns like 'audiences' and 'headline' are removed from the news dataset. This is because features like 'headline' are not useful to us, as the news sentiment has already been provided to us. We removed these features after comparing the accuracies before and after including these features.
- Categorical columns like 'headlineTag', 'provider' and 'sourcelid' are encoded into their numeric representations using the factorize function.
- News items having same 'date' and 'assetCodes' are grouped and aggregated on mean. This is done as we are going to merge news and market datasets on date and assetCodes in the next step, and grouping these records will help in merge.

7. Feature engineering

- Here, we merge market and news datasets on the columns date and assetCode. We use the function merge to do this and use only columns from the market dataset, as we need news information for only those records appearing in the market dataset. The merge is similar to a left outer join in SQL with the left dataframe being market and right dataframe news. We do this because we only need records in the news dataset whose assetCodes and Date are same as in the market dataset.

marketCommentary	sentenceCount	wordCount	assetCodes	assetName	firstMentionSentence	relevance	sentimentClass
False	11	275	{'0857.HK', '0857.F', '0857.DE', 'PTR.N'}	PetroChina Co Ltd	6	0.235702	-1
False	55	907	{'STAN'}	Travelers Companies Inc	8	0.447214	-1
False	15	388	{'WMT.DE', 'WMT.N'}	Wal-Mart Stores Inc	14	0.377964	-1
False	14	325	{'GOOG.O', 'GOOG.OD', 'GOOGa.DE'}	Google Inc	13	0.149071	-1

- After performing the above merge operation, we observed that there were many null values in the news columns of records. This is due to the mismatch in the

assetCode and assetCodes of the market and news datasets respectively. As can be seen above, the news dataset is not normalized. To solve this, we tried to normalize the news dataset by unstacking the assetCodes into 1 assetCode per row so that the null entries in news-related columns would decrease. But we couldn't implement this code unfortunately due to frequent kernel crashes as a result of memory failure. So, we had to remove records with null values after merging, as removing them gave us a higher accuracy when compared to imputing them with mean, median or using Mice algorithm.

8. Building Model

- Listed below is the different models we used:
 - **Linear Discriminant Analysis**:0.538
 - **Quadratic Discriminant Analysis**:0.512
 - **AdaBoost(n_estimators=300)**:0.541
 - **Logistic Regression (c=1, penalty=l2)** :0.53
 - **XGBoost(n_jobs=4,n_estimators=250,max_depth=8,eta=0.1)** :
0.62926
- We use XGBoost(**eX**treme **G**radient **B**oosting) classifier as our final model, as it has the highest accuracy.
- XGBoost is an algorithm which gave the highest accuracy in many of the research papers we read.
- XGBoost is an implementation of gradient boosting decision tree algorithm designed for speed and performance that is dominating competitive machine learning.
- XGBoost was created by Tianqi Chen and its primary benefits are reduced time complexity and better memory utilization. Gradient boosting is an approach where new models try to reduce the error residuals of the prior models. The new weak learners focus on areas where the previous learners performed poorly. After repetitive iterations, the model will be able to fit the data better. Iterations are done until the error residuals are zero or close to zero. Its aim is to minimize the loss when adding successive models. It is called gradient boosting because it uses a gradient descent algorithm to minimize the loss

when adding new models. This approach supports both regression and classification predictive modeling problems. [7]

- “Gradient boosting is an improvement on decision trees, where each tree is approximated as an aggregate of many regressor functions $f_i(x)$. This is different from the traditional idea of decision trees, where a Gini impurity based splitting is directly used to find the best split in the feature space. Each successive function f_i is built such that the misclassification rate successively decreases. This is done by trying to better classify the residuals, or the misclassified samples of the i th iteration in the $i + 1$ th iteration. Hence, the error in classification successively decreases. This step-wise aggregation of functions approximates a node in a tree, and eventually, the entire tree is approximated. Once each tree has been optimally approximated, the structure scores and gain are calculated, based on which the best split is determined. There are obvious advantages of doing this: since each tree is built carefully, a lot of random trees need not be used in classifying a random sample, substantially decreasing the number of trees required in the forest of classifiers. XGBoost is a framework and library which parallelizes the growth of gradient boosted trees in a forest. The idea of gradient boosting [9] comes from the principle of gradient descent: a greater number of the misclassified samples of the i th learner should be classified correctly by the $i + 1$ th learner, and so on. This implies that the error in classification reduces as more number of regressors are constructed in each node. More specifically, in the case of gradient boosted decision trees (GBDT), the $i + 1$ th regression function is expected to rectify the mistakes of the i th function. Since the error goes on decreasing as a tree is approximated by a larger number of functions, gradient boosting is considered to be a convex optimization problem. XGBoost [7] aims to minimize the time required to grow trees. This makes GBDTs more practical to use. Here, too, a subset of the features is used to build each tree. However, in the algorithmic description of XGBoost, the set of features being used to grow the tree is deliberately excluded: the assumption is that a subset of features is drawn out and fed into the algorithm.” [5]

- **XGBoost.XGBClassifier(n_jobs=4,n_estimators=250,max_depth=8,eta=0.1**

The parameters used are:

n_jobs – Number of parallel threads used to run xgboost.





n_estimators - Number of boosted trees to fit

max_depth – Maximum tree depth for base learners.

eta: learning_rate: shrinks the feature weights to make the boosting process more conservative.

9. Results and Future Improvements

- We got the highest accuracy of 62.926% with the **XGBoost** model.

972	▼ 116	RenatoBMLR		0.62971	2	2mo
973	▼ 116	dsp2109		0.62931	2	2mo
974	▲ 335	Tigmanshu	  	0.62926	16	5m
Your Best Entry ↑						

- XGBoost model was chosen for it's ability to perform incremental learning, a technique where the algorithm steps through the dataset training on a number of observations without retraining.
- Some possible reasons for the low accuracy are:
 - There might be a strong correlation between the behavior of stock prices and the time the news articles become publicly available as [1] suggested. As our classifier model predicts stock returns 10 days into the future, the dependence of the news article on the stock may have diminished.
 - Significant reduction in the number of records in the combined dataset after merging due to the null values in the news information of records. This was due to the mismatch in the assetCode and assetCodes columns in the market and news datasets respectively.
 - Discrepancies in the dataset like the large number of null values in the market dataset.
 - Inability to run some code due to the sheer size of the dataset which caused frequent kernel crashes. We were permitted to work only on Kaggle's internal kernel and thus could not interface with outside computing resources.
 - Also, news is only one of the factors impacting stock behavior. There are many other factors which also influence stock movements like economic factors, industry performance and investor sentiment.
- As future improvements, we could dig more into the dataset and try engineering new features.

- We could also try other models like Recurrent Neural Networks(LSTM) or LightGBM.

10. References

- [1] Gidofalvi, Gyozo. Using News Articles to Predict Stock Price Movements. Department of Computer Science and Engineering, University of California, San Diego. 2001.
- [2] Shubharthi Dey, Yash Kumar, Snehanshu Saha, Suryoday Basak. Forecasting to Classification: Predicting the direction of stock market price using Xtreme Gradient Boosting
- [3] Fung, Gabriel, et. al. The Predicting Power of Textual Information on Financial Markets. IEEE Intelligent Informatics Bulletin. Vol. 5. No. 1. June 2005.
- [4] Anurag Nagar, Michael Hahsler, Using Text and Data Mining Techniques to extract Stock Market Sentiment from Live News Streams, IPCSIT vol. XX (2012) IACSIT Press, Singapore.
- [5] Suryoday Basak, Snehanshu Saha, Saibal Kar, Luckyson Khaidem, Sudeepa Roy Dey, Predicting the Direction of Stock Market Price Using Tree Based Classifiers.
- [6] Yauheniya Shynkevich, T.M. McGinnity, Sonya Coleman, Ammar Belatreche, Predicting Stock Price Movements Based on Different Categories of News Articles, 2015 IEEE Symposium Series on Computational Intelligence.
- [7] XGBoost: A Scalable Tree Boosting System -Tianqi Chen,Carlos Guestrin
- [8] Xiqian Zhao, Juan Yang, Lili Zhao & Qing Li, The Impact of News on Stock Market: Quantifying the content of internet based financial news.
- [9] Di, X. (2014), Stock Trend Prediction With Technical Indicators using SVM. Stanford University. Friedman, J. H. (2000). Greedy Function Approximation: A Gradient Boosting Machine. Annals of Statistics. Vol. 29, p.1189–1232.
- [10]Data Engineering kernel from public Kernels (Two Sigma Competition)<https://www.kaggle.com/dmdm02/complete-eda-voting-lightgbm>
- [11] Getting Started Kernel from Two Sigma <https://www.kaggle.com/dster/two-sigma-news-official-getting-started-kernel>
- [12] XGBoost Baseline public Kernel from Kaggle.<https://www.kaggle.com/jannesklaas/lb-0-63-xgboost-baseline>