Using News to predict stock Movement

I.ABSTRACT

In this article, we use news data and market data to predict the trend of the stock market. We choose all the numerical features in these two datasets. Then we combine these two dataset in a innovative way and normalize these data. Finally we train logistic regression, decision tree, random forest and multiple layer perceptron with the training data. The results of those models are compared and analyzed in “Discussion” section. We found that the results doesn’t change much when we use different models.

Keywords:

News Data, Market Data, Two Sigma, Random Forest, Logistic Regression, Support Vector Machine, Multilayer Perceptron

II. INTRODUCTION

Can we use the content of news analytics to predict stock price performance? Recently, Two Sigma holds a Kaggle contest hoping to find out a good machine learning model to predict stock prices of ten days later. It is an infinite difficult problem because the quantitative relationship between stock price and news are vague. It would be very interesting whether a machine learning model could learn the trend of stock price using news data. The computation resources are very limited on the Kaggle platform, so it favors efficient models instead of time-consuming deep learning model. Our group test logistics regression, decision tree, random forests and multilayer perceptron model, SVM and their performance on stock prediction, which shed light on the complexity of this problem.

III. PROBLEM STATEMENT

It is a traditional classification problem. It would be interesting to see if machine learning model fit in this problem. Two data sets are given: market data and news data. They are both in pandas dataframe format.

The market data are comprise of time series, stock code information as key index for each entry. Other column is used for prediction and the label is returnsOpenNextMktres10. If it is positive, the classification of it is 1 otherwise is -1.

The news data have time series as its index keys and the related stock codes in the column serves as foreign keys to the market data. It is many-to-many relationship because a stock may have more than one entry of news on the same day.

The concatenated two data set is the input of the problem. The possibility of label for each required stock after 10 days instead of the label itself is the output of our model. And the performance is evaluated by the following formula:

$$x_t = \sum_i y_{ti} r_{ti} u_{ti}$$

where y_{ti} is the possible of our label (it would be negative for -1 label) for i stock at day t, r_{ti} is value in returnsOpenNextMktres10 column and u_{ti} is the universe (0 or 1). The score of model is given as:

$$score = \frac{avg(x_t)}{\sigma(x_t)}$$

where x_t is computed using above formula for each day. The score formula implies the problem is classification problem because if y_{ti} (our prediction possibility) is always correct for each stock on each day (at best 1 or -1), x_t is always positive large, so the deviation is small. The average of x_t is also large, as a result score is maximized. The falsified prediction makes x_t negative, so deviation is enlarged and average shrinks. Accordingly, the score would become smaller for more incorrect prediction.

IV. RELATED WORK

There are lot of works having been done in this field. Schumaker and Chen use SVM derivative to predict the discrete number stock prices twenty minutes after an article release and the result show that their model had a statistically significant impact on predicting future stock prices compared to linear regression[3].

Peng and Jiang applied the popular word embedding methods and deep neural networks to leverage financial news to predict stock price movements in the market and the results show that their methods can significantly improve the stock prediction accuracy on a standard financial database over the baseline system using only the historical price information [2].

Fiacc Larkin and Conor Ryan use Genetic program combined with the news sentiment data to find non-linear solutions for predicting large intraday price jumps on the S&P 500 up to an hour before they occur, the results show that GP was successfully able to predict stock price movement using these news *alone*, that is, without access to even current market price[1].

There are indeed a lot of work having been done in this field. But the problem have not yet been solved perfectly. The only thing people can make sure today is that there is some relation between the public media's information and the stock market movement. Many researchers just build various models to make use of the news data to predict the trend of the stock market.

Because of the development of the NLP technologies, some companies are able to refine more information from the news. And many companies also offer some tools or framework of machine learning technologies which we can employ very conveniently. And Two sigma just offered us the refined news data and market data. What we need to do is to choose the appropriate feature, preprocess the data, and build a model to predict the future trend of the stock market with the data.

V. DATASET

The DataSet comprises of two type of Data.
Market Data: About 3 million rows.
News Data: About 9 million rows.
The explanation for the features are attached to the appendix.A

VI. APPROACH

The Overall process contains five stages, just like the following figures shows:

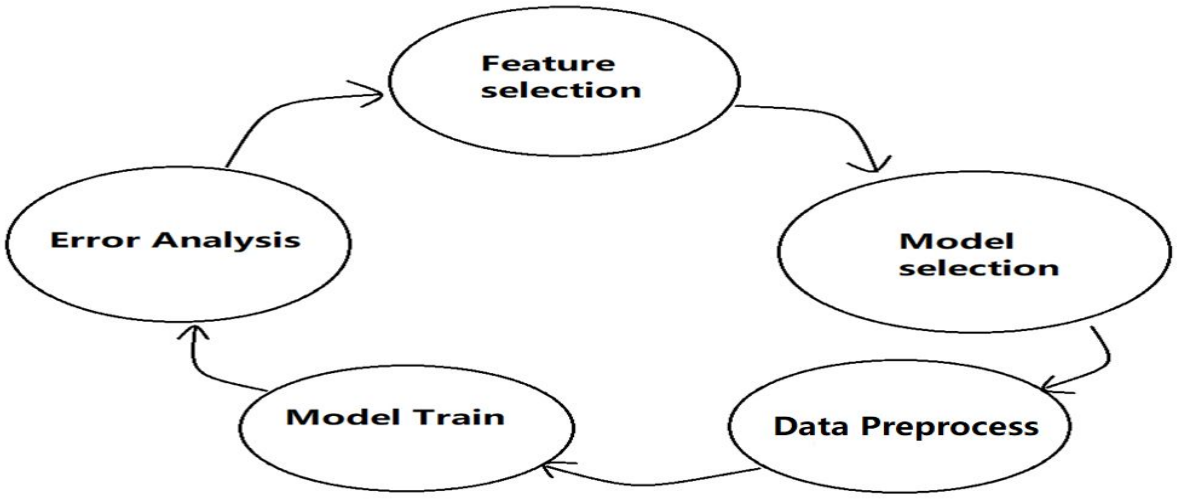

Figure 1

We iterate around this circle util the result cannot be improved.

Feature selection:

We decide to choose all the numerical features from the news data for simplicity reason. And we choose all the features of the market data.

Model selection:

Because of the restriction of the kaggle contest:

16 GB Memory, 9 Hours total runtime, 4 CPU cores

We decide to choose a very simple model which is very fast to train and requires small extra memory usage: Logistic regression.

Preprocess the data:

Preprocess the market Data:

1. Extract the market data whose “time” value is after Jan.1/2012 as the new market data.This is a way to reduce the amount of data for the reason of restricted memory resources.
2. Check the data and eliminate the the rows that contain exceptional data.

For each feature, sort the data value of that feature, eliminate all the corresponding rows that contain the value which is not in the normal range of that feature.

3. Extract the value of “returnsOpenNextMktres10” as the label set of the problem. If this value is bigger or equal to 0, transfer it to +1, else, transfer it to -1.

Preprocess the news Data:

1. Extract the news data whose “time” value is after Jan.1/2012 as the new market data.
   This is a way to reduce the amount of data for the reason of restricted memory resources.
2. Eliminate the features that we don’t need. But remains the “AssetName” and “Time” feature.

Combine the market data and the news data:

There are “AssetName” and “Time” features in both dataset. But for market data, there is at most one row for a specific combination of “AssetName” and “Time”. While for news data, there may be multiple rows. As the figure below shows:

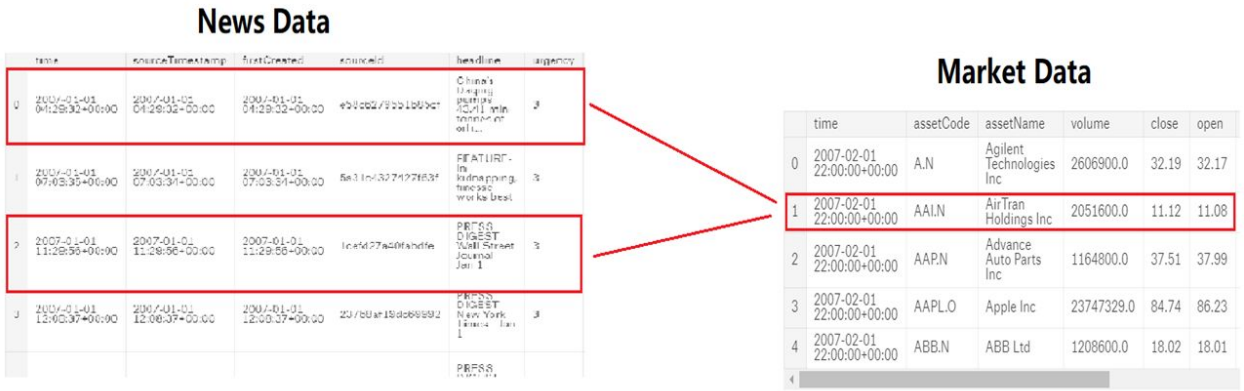

Figure 2

In this scenario, we combine the multiple rows to one row in the way, that we discard the raw data and remain the maximum value, minimum value, standard deviation of that group of values. Then we combine the row from news data and the row from market data.

Extract the value of “month” and “week” from the “time” value. Then eliminate the “Assetname” “Assetcode” and “Time” features.Normalize the data(standard score) and then replace the value “NaN” or “Infinity” with “0”.

Split the dataset into two parts: Training Data (80%), Validation Data (20%)

Model Train:

Our model is Logistic Regression. So we use the “sklearn.LogisticRegressionClassifier”, to build our model. At first, we decided to set the penalty parameter C as 1.

Here is part of the code:
model = LogisticRegression(C = 1).fit(X, y)

And here is part of the result:

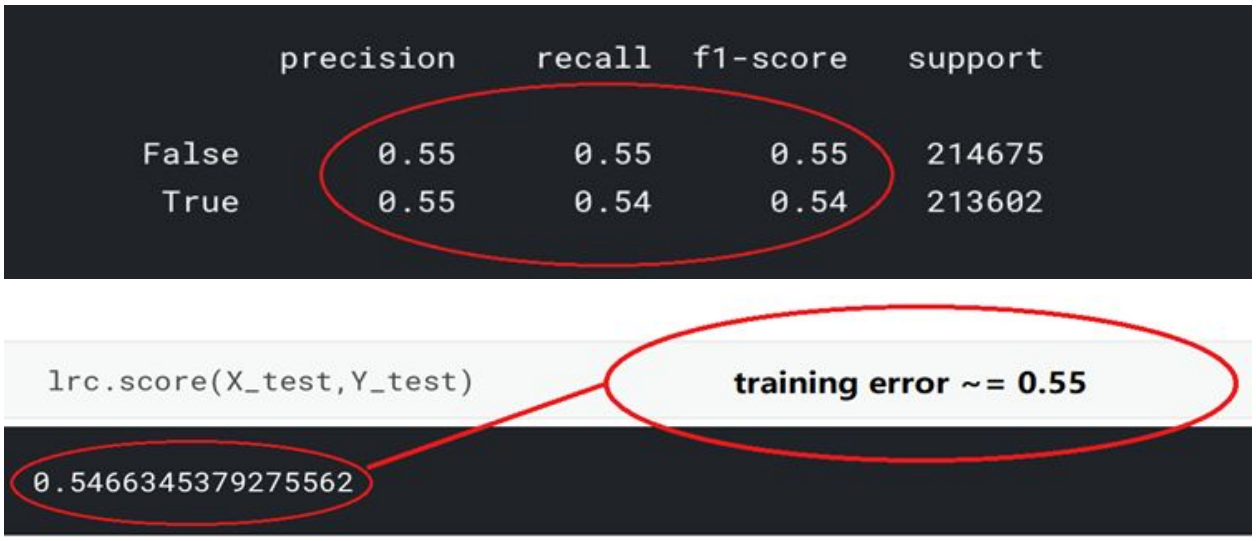

Figure 3

Then we change the value of C, and retrain the model. The results are similar to each other. Because there are plenty of data, and the performance of the model doesn’t change when the C changes. We think maybe this model itself is too simple for this problem. So, we decided to try some more complex model. These models are introduced in the “EXPERIMENTS” section. And the comparison among these models are made in “DISCUSSION” section.

VII. EXPERIMENTS

All the models that we tried are listed below. And the results are merge into a graph.

1. **Decision tree**

   dtc=DecisionTreeClassifier()
   dtc=dtc.fit(X_train,Y_train.ravel())

2. **MLP**

   MLP-2 :

   mlp = MLPClassifier(solver='lbfgs', alpha=1e-5,hidden_layer_sizes=(5,5), random_state=1)

mlp.fit(X_train,Y_train)

MLP-3:

mlp = MLPClassifier(solver='lbfgs', alpha=1e-5,hidden_layer_sizes=(20,20,20), random_state=1)

mlp.fit(X_train,Y_train).

3. **Random Forest tree**

rfc = RandomForestClassifier(n_estimators=100, max_depth=2,random_state=0)

rfc.fit(X_train,Y_train)

4. **SVM**

Trained very slow and consumed too much memory, no result.

5. **Final results:**

SUMMARY BY CATEGORY

Category	Precision	Recall	f1-score	Accuracy
Logistics Regression	0.55	0.55	0.55	0.547
Decision Tree	0.52	0.52	0.52	0.523
Random Forest	0.54	0.64	0.59	0.549
MLP-2	0.54	0.69	0.6	0.539
MLP-3	0.55	0.59	0.57	0.557

Figure 4

Figure 5

There are not much difference among the validation results of these models. Then We finish the first round of the overall process. In the next section we analyze these results and give more insights about this problem.

This is the score of our kernel in this kaggle contest:

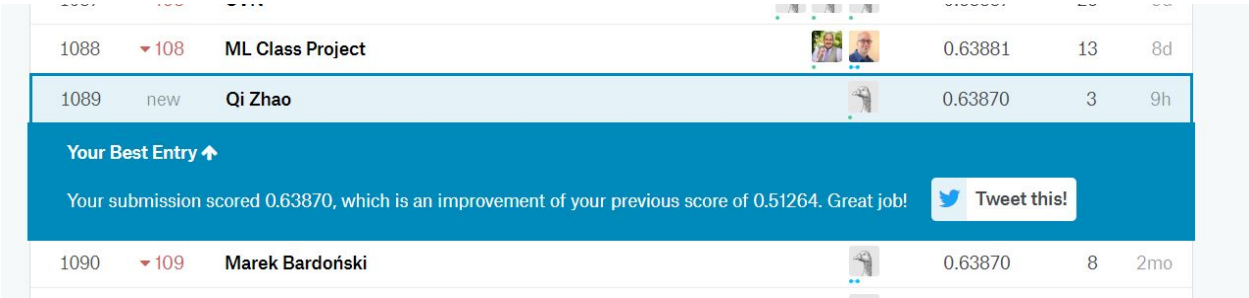

Figure 6

IX. DISCUSSION

Firstly, we should talk about the feature selection. For the experiment we have done above, we just choose features manually based on some economy background. This should make sense in a way because economy background can provide us with the correct way to select features among all, it can tell us which feature should be the most important factor that influence a stock from a finance perspective.

However, from a computer science perspective or from a machine learning perspective, there's another way that may be more appropriate to do feature selection whose name is Lasso algorithm. Lasso, the acronym of "least absolute shrinkage and selection operator" is a regression analysis method that performs both variable selection and regularization in order to enhance the prediction accuracy and interpretability of the statistical model it produces. Though we have tried using lasso algorithm, and result is we finally choose 2 features from all the 31 features, and the accuracy didn't improve with the same classifier. Though the result didn't meet our expectation but what I want to emphasize is that in the feature selection process, maybe sometimes the results of a pure mathematical calculation are more reliable than those based on economy background knowledge.

Secondly, as we can see from the above pictures, the accuracy of Decision Tree classifier is: 52.3%, the accuracy of Multi-Layer perceptron classifier is: 55.7%, the accuracy of Logistic Regression classifier is: 54.7%, and the accuracy of Random Forest classifier is: 54.9%.

Obviously, the MLP win the competitions among all the classifiers. A multilayer perceptron is a class of feedforward artificial neural network. An MLP consists of, at least, three layers of nodes: an input layer, a hidden layer and an output layer. For our experiment we have tried twice MLP with 3 hidden layers of which each hidden layer has 20 nodes and 2 hidden layers of which each hidden layer has 5 nodes. The accuracy showing above is the result of the MLP with 3 hidden layers.

What's more, as we can see, the accuracy results of all the classifiers are below 60%, that's not a satisfactory accuracy. We should find out the essential reason cause of this low accuracy. There are two main reasons that may cause this results. First, the prediction results of stock prediction based on the data Kaggle provided can only reach at most 55% to 60%, since the increase or decrease of one stock can't be influenced by only 31 features, it's very hard to

do stock prediction in a high accuracy. If we can do stock prediction with our program and reach an accuracy as high as 80%, then we all can be next Warren Buffet, however we all know that's impossible. From another perspective, may be the pre-processing of the data is not perfect, we may should pay more attention on data pre-processing, like doing more normalization or do some convolution based on some specific features. If we have more time, we can try those ways to calculation new accuracy and compare them to the result we have now.

X. CONCLUSION

Our study focuses on test performance of logistics regression, random forests, SVM and MLP models on the stock prediction problem. In the course of analysis, we have demonstrated that the accuracy of all the machine learning model is slightly above 50% and MLP wins among these classifiers. The quantitative relationship between stock price and news can hardly be learned with single simple machine learning model. The study concludes that more complicated model is needed for stock prediction problem. Additional research should focus more on deep-learning model for stock prediction problem although computation resources are limited on the Kaggle platform.

REFERENCES

1.Larkin, F., & Ryan, C. (2008, March). Good news: Using news feeds with genetic programming to predict stock prices. In *European Conference on Genetic Programming* (pp. 49-60). Springer, Berlin, Heidelberg.

2.Peng, Y., & Jiang, H. (2015). Leverage financial news to predict stock price movements using word embeddings and deep neural networks. *arXiv preprint arXiv:1506.07220*.

3.Schumaker, R., & Chen, H. (2006). Textual analysis of stock market prediction using financial news articles. *AMCIS 2006 Proceedings*, 185.

Appendix A

There are total 15 features:

- `time`(datetime64[ns, UTC]) - the current time (in marketdata, all rows are taken at 22:00 UTC)
- `assetCode`(object) - a unique id of an asset
- `assetName`(category) - the name that corresponds to a group of `assetCodes`. These may be "Unknown" if the corresponding `assetCode` does not have any rows in the news data.
- `universe`(float64) - a boolean indicating whether or not the instrument on that day will be included in scoring. This value is not provided outside of the training data time period. The trading universe on a given date is the set of instruments that are avilable for trading (the scoring function will not consider instruments that are not in the trading universe). The trading universe changes daily.
- `volume`(float64) - trading volume in shares for the day
- `close`(float64) - the close price for the day (not adjusted for splits or dividends)
- `open`(float64) - the open price for the day (not adjusted for splits or dividends)
- `returnsClosePrevRaw1`(float64) - see returns explanation above
- `returnsOpenPrevRaw1`(float64) - see returns explanation above
- `returnsClosePrevMktres1`(float64) - see returns explanation above
- `returnsOpenPrevMktres1`(float64) - see returns explanation above
- `returnsClosePrevRaw10`(float64) - see returns explanation above
- `returnsOpenPrevRaw10`(float64) - see returns explanation above
- `returnsClosePrevMktres10`(float64) - see returns explanation above
- `returnsOpenPrevMktres10`(float64) - see returns explanation above
- `returnsOpenNextMktres10`(float64) - 10 day, market-residualized return. This is the target variable used in competition scoring. The market data has been filtered such that `returnsOpenNextMktres10` is always not null.

News Data:

- `time`(datetime64[ns, UTC]) - UTC timestamp showing when the data was available on the feed (second precision)
- `sourceTimestamp`(datetime64[ns, UTC]) - UTC timestamp of this news item when it was created
- `firstCreated`(datetime64[ns, UTC]) - UTC timestamp for the first version of the item
- `sourceId`(object) - an Id for each news item
- `headline`(object) - the item's headline
- `urgency`(int8) - differentiates story types (1: alert, 3: article)
- `takeSequence`(int16) - the take sequence number of the news item, starting at 1. For a given story, alerts and articles have separate sequences.
- `provider`(category) - identifier for the organization which provided the news item (e.g. RTRS for Reuters News, BSW for Business Wire)

- `subjects`(category) - topic codes and company identifiers that relate to this news item. Topic codes describe the news item's subject matter. These can cover asset classes, geographies, events, industries/sectors, and other types.
- `audiences`(category) - identifies which desktop news product(s) the news item belongs to. They are typically tailored to specific audiences. (e.g. "M" for Money International News Service and "FB" for French General News Service)
- `bodySize`(int32) - the size of the current version of the story body in characters
- `companyCount`(int8) - the number of companies explicitly listed in the news item in the subjects field
- `headlineTag`(object) - the Thomson Reuters headline tag for the news item
- `marketCommentary`(bool) - boolean indicator that the item is discussing general market conditions, such as "After the Bell" summaries
- `sentenceCount`(int16) - the total number of sentences in the news item. Can be used in conjunction with firstMentionSentence to determine the relative position of the first mention in the item.
- `wordCount`(int32) - the total number of lexical tokens (words and punctuation) in the news item
- `assetCodes`(category) - list of assets mentioned in the item
- `assetName`(category) - name of the asset
- `firstMentionSentence`(int16) - the first sentence, starting with the headline, in which the scored asset is mentioned.
    - 1: headline
    - 2: first sentence of the story body
    - 3: second sentence of the body, etc
    - 0: the asset being scored was not found in the news item's headline or body text. As a result, the entire news item's text (headline + body) will be used to determine the sentiment score.
- `relevance`(float32) - a decimal number indicating the relevance of the news item to the asset. It ranges from 0 to 1. If the asset is mentioned in the headline, the relevance is set to 1. When the item is an alert (urgency == 1), relevance should be gauged by firstMentionSentence instead.
- `sentimentClass`(int8) - indicates the predominant sentiment class for this news item with respect to the asset. The indicated class is the one with the highest probability.
- `sentimentNegative`(float32) - probability that the sentiment of the news item was negative for the asset
- `sentimentNeutral`(float32) - probability that the sentiment of the news item was neutral for the asset
- `sentimentPositive`(float32) - probability that the sentiment of the news item was positive for the asset
- `sentimentWordCount`(int32) - the number of lexical tokens in the sections of the item text that are deemed relevant to the asset. This can be used in conjunction with wordCount to determine the proportion of the news item discussing the asset.
- `noveltyCount12H`(int16) - The 12 hour novelty of the content within a news item on a particular asset. It is calculated by comparing it with the asset-specific text over a cache of previous news items that contain the asset.

- `noveltyCount24H`(int16) - same as above, but for 24 hours
- `noveltyCount3D`(int16) - same as above, but for 3 days
- `noveltyCount5D`(int16) - same as above, but for 5 days
- `noveltyCount7D`(int16) - same as above, but for 7 days
- `volumeCounts12H`(int16) - the 12 hour volume of news for each asset. A cache of previous news items is maintained and the number of news items that mention the asset within each of five historical periods is calculated.
- `volumeCounts24H`(int16) - same as above, but for 24 hours
- `volumeCounts3D`(int16) - same as above, but for 3 days
- `volumeCounts5D`(int16) - same as above, but for 5 days
- `volumeCounts7D`(int16) - same as above, but for 7 days

The market data are provided by Intrinio.
The news data are provided by Thomson Reuters.

- Where is your data in the overall process?
- Two labels on top of each other on feature selection slide (Profs are already cognitively loaded, think of the grumpy reviewer syndrome, you need to explain the strengths of your project with straightforward results and presentation (occam's razor))
- Why did *you* choose features? That's what ML is good for! (You could have justified the feature choice by quantifying feature importance.)
- Combining news with market data: - random combination (multi rows in the other domain, they randomly picked one correspondance to merge with) - well explained preprocessing of this part
- What are the random forest params? MLP: Spell out name before acronym Check how things vary with different params.