Kaggle Competition 2sigma
Using News to Predict Stock Movements

Barthold Albrecht (bholdia)

Yanzhou Wang (yzw)

Xiaofang Zhu (zhuxf)

1 Introduction

The 2sigma competition at Kaggle aims at advancing our understanding of how the content of news analytics might influence the performance of stock prices. For this purpose a large set of daily market and news data is provided for a subset of US-listed financial instruments. This data shall be used to train any kind of learning algorithm deemed useful in order to predict future stock market returns.

The competition comprises two stages with two different evaluation periods. In the first stage the predictions are tested against historical data of the period 1/1/2017 to 7/31/2018. This stage will be terminated early next year at which time the final submissions of the participating teams must be handed in. The latter will then be evaluated against future data for about six months to identify the best performing submission which will be disclosed 7/15/2019.

The objective function for this machine learning task is set the same for all participants in the competition and constructed as follows: for each day t within the evaluation period the value x_t is calculated as

$$x_t = \sum_i \hat{y}_{ti} r_{ti} u_{ti} \tag{1}$$

where for any financial asset $i \in \{1, ..., m\}$ the term $\hat{y}_{ti} \in [-1, 1]$ stands for the predicted confidence value that it's ten-day market-adjusted leading return $r_{ti} \in \mathbb{R}$ is either positive or negative. The universe variable $u_{ti} \in \{0, 1\}$ controls whether the asset i is included in the evaluation at the particular evaluation day t. Finally, the score which determines the position in the competition is composed of the mean and the standard deviation of the daily value x_t:

$$score = \frac{\bar{x_t}}{\sigma(x_t)} \tag{2}$$

with $score \equiv 0$ for $\sigma(x_t) = 0$.

We apply three different algorithms to this problem: logistic regression, neural network and gradient boosting tree.

2 Related work

There have been multiple attempts looking into the popular topic of forecasting stock price with techniques of Machine Learning. Based on the works we find, the focus of these research projects vary mainly in three ways.
(1) The text information used in prediction ranges from public news, economy trend to exclusive information about the characteristics of the company.[2][3]

(2) The targeting price change can be near-term (high-frequency, less than a minute), short-term (tomorrow to a few days later), and long-term (months later).[4][5]
(3) The set of stocks can be limited to particular stocks, to stocks in a particular sector or to generally all stocks.[6]
The competition we are managing is the kind of topic that predicts long-term price on generally selected stocks, using time series data of stock price and public data (news data). This kind of problem currently doesn't produce satisfactory prediction accuracy: Most researches in this domain have only found models with around 50 to 60 percent accuracy[5][7], comparing to the over-70-percent accuracy when only considering limited number of stocks or sticks in a particular industry and has the access to exclusive information of the company[6].

3 Datasets and features

3.1 Description

All the data used in the project is provided by Kaggle. Two sources of data are provided, one for market data and one for news data, both spanning from 2007 to the end of 2016. The market data contains various financial market information for 3511 US-listed instruments. It is comprised of more than 4 million samples and 16 features.

The "returnsOpenNextMktres10" Column indicates the market normalized return for the next 10 days and, thus, serves as the ground truth value for the prediction task. The news data contains information at both article level and asset level. There are more 9 million samples and 35 features. Most of the news features are either numerical or type indicators except the "headline" feature, which contains text. The news data provided is intentionally not normalized.

Both data sets can be joined by using either the time stamp, asset code or asset name.

3.2 Processing

As shown in Figure 1, the stock crashed in late 2008 due to the financial crisis. Thus the stock behaves differently before 2009. Since in the coming 1 year the stock is unlikely to crash, only data after 2009 is considered.

Figure 1: Closing prices by quantiles

A large number of samples have null values for features related to normalized market returns, they are filled with the corresponding raw market returns. All features from the market dataset is selected as input. The news dataset, however, are filtered based on feature correlations; highly correlated news features are removed from the training set. For example, sentenceCount and wordCount are highly correlated, so wordCount is removed and sentenceCount is kept. Moreover, outliers with extreme values are removed from market dataset. For example, if the open to close ratio for a single stock is greater than 2, the sample is discarded as an outlier. For the news dataset, most numerical features are clipped between 98 and 2 percentile. The dataset is split into 95% training data and 5%

validation data because of the large sample size. To make it a classification problem, the prediction labels (market residual return in next 10 days) are converted to binary values with 0 representing negative return and 1 representing positive return.

4 Methods

4.1 Logistic regression

We chose logistic regression as a starting point for establishing a baseline score. The logistic regression takes in all the features as is, such that it does not include higher degree terms. Because of the large size of the training data, small regularization is used. The log likely-hood is

$$l(\theta) = \sum_{i}^{m} y^{(i)} \log h(x^{(i)}) + (1 - y^{(i)}) \log(1 - h(x^{(i)}))$$

4.2 Neural network

We implement a fully connected neural network with two inputs. Into the first input branch we feed all numerical values of the preprocessed dataset, while the second input branch encodes the categorical data "asset code" for each sample in a trainable embedding layer. After batch normalisation and two fully connected layers for the numerical part and one fully connected layer for the categorical part, both branches of the network are concatenated. The concatenated data is finally fed into one more fully connected layer followed by the output layer. All fully connected layers use relu activation except the output layer which has a sigmoid activation function.

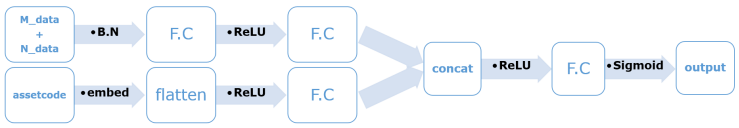

Figure 2: Architecture of the FCNN

The loss function we use is binary cross entropy with the Adam optimizer with default learning rate of 0.001 and batch size of 32. The model has 54.000 trainable parameter.

4.3 Gradient boosting

Gradient boosting is a technique that combines weak predicting models into an ensemble to produce a much stronger one. It is typically implemented on decision trees. Like other boosting algorithms, Gradient boosting is an iterative operation. At each iteration, the algorithm creates a new estimator that minimizes the loss with respect to the current model. This minimization can be approximated by fitting the new estimator to the gradient of loss such that :

$$f_k(x) = f_{k-1}(x) + h_k(x)$$
$$r_{ik} = -\Big[\frac{\partial L(y_i, f_{k-1}(x))}{\partial f_{k-1}(x)}\Big] \text{ for } i \in 1, ..., m$$

where

f_k is the ensemble model at kth iteration

r_{ik} is the gradient(residual) of the loss function with respect to f_{k-1} for ith data

h_k is the new model that fits r_{ik} for $i \in 1, ..., m$

L is the loss function (binary log loss function for this project)

It is similar to the normal gradient descent except that the gradient descent is performed on the output of the model instead of the parameters of each weak model.
The regularization is achieved through several ways: by slowly decreasing the learning rate, setting the number of minimum samples in a tree leaf, limiting number of leaves, or penalizing the complexity

of the tree model such as L2 regularization.
LightGBM library is used to implement this algorithm in this project. It converts continuous features into bins which reduces memory and boosts speed and grows each tree with the priority given to the leaf with maximum delta loss, leading to lower overall loss.

5 Results and discussion

Table 1: Validation accuracy, score and test score of the 3 models

Metric	LR	FC Neural Network	Light-GBM
Train-accuracy	0.503	0.561	0.554
Val-accuracy	0.485	0.557	0.538
Val-score	0.247	0.781	0.731
Competition score	0.259	0.645	0.644

Table 1 lists the result of the 3 models for the validation dataset and test dataset. The Fully-Connected Neural Network performs the best. And the Light-BGM performs almost as good as the Fully-Connected Neural Network. The poor result of the logistic regression is expected because the algorithm assumes linear relationships. Adding news data does not have noticeable effect on the performance. This can be explained by that the news data is more subjective and there are many noises and non-related features. The validation accuracy is similar for all three algorithms. The competition scores, however, are quite different. One explanation could be that the score calculation considers not only the binary prediction, but also market return and standard deviation of the prediction.

a. AUC curves
The AUC score for the Logistic Regression, the Fully Connected Neural Network Model and the Light-GBM Model is 0.5, 0.5799 and 0.5753 respectively, as shown by Figure 3, Figure 4 and Figure 5. The logistic regression behaves similar to a random guess, while the other 2 algorithms show slightly higher ability to predict the market returns.

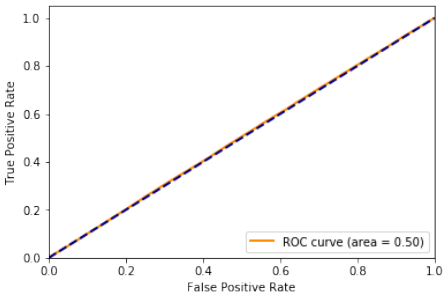

Figure 3: ROC curve for Logistic Regression

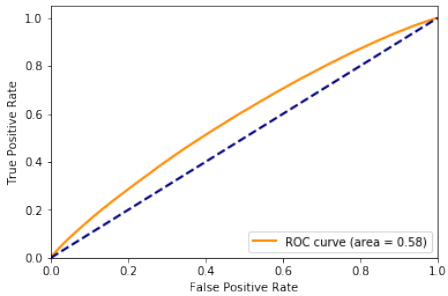

Figure 4: ROC curve for the FC Neural Network

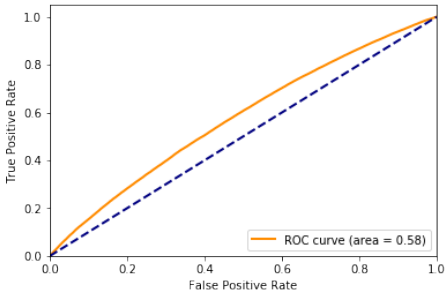

Figure 5: ROC curve for the LGBM

b. Confusion matrics

Table 2: Confusion matrix for LR

	Pred Class 0	Pred Class 1
Class 0	0	1
Class 1	0	1

Table 3: Confusion matrix for FCNN

	Pred Class 0	Pred Class 1
Class 0	0.458	0.542
Class 1	0.347	0.653

Table 4: Confusion matrix for LGBM

	Pred Class 0	Pred Class 1
Class 0	0.495	0.504
Class 1	0.389	0.611

c. Output analysis
The first model, the Logistic Regression model fails because the model assumes the features and the result are linearly correlated, which is obviously too much simplify the situation. And as expected, the model gets not so good performance.
To analyse on the output performance on the three models, as the table 1 showing the training accuracy and the validation accuracy of the three models, they are not overfitting. From the table, we see that the accuracies on training set and validation set are close to each other, which means the models perform similarly on the training set and the validation set.

6 Conclusion and Future work

Out of the three attempted algorithms, the neural network performs the best followed closely by the gradient boosting tree, while the logistic regression behaves almost like a random guess. As the logistic regression is fairly a simple algorithm with linear mappings to each feature dimension, its incapability to capture the complex relationship is expected. On the other hand, both the neural network and gradient boosting tree are powerful non-linear algorithms with a large degree of flexibility and control, making them competent to model complex situations.

For future work, deeper dive into the feature engineering is needed. It is also worth exploring to combine neural network and gradient boosting tree in an ensemble fashion to produce a stronger model. One of the news features is text based, thus natural language processing can be implemented to extract useful information from it. Given the large parameter sets for the neural network and the gradient boosting tree, achieve the optimum parameters is both difficult and time consuming. However, there is still possible room to make improvement by further tuning the parameters. Lastly, choosing a more powerful baseline such as the support vector machine instead of the simple logistic regression should be considered.

7 Contributions

As a group working on this collaborated project, we contributed equally overall. Barthold Albrecht has additional contribution on establishment of the Logistic Regression model and the fully-connected Neural Network model. Yanzhuo Wang has additional contribution on establishment of the Logistic Regression model and the LGBM model. Xiaofang Zhu has additional contribution on establishing the fully-connected Neural Network model.

8 Source Code

https://drive.google.com/open?id=1MnF5oPuzbDvotKXL6sJKgLPQaBCN8v9x

References

[1] XingYu Fu, JinHong Du, YiFeng Guo, MingWen Liu, Tao Dong XiuWen Duan. *(2018) A Machine Learning Framework for Stock Selection*. arXiv: 1806.1743

[2] Yauheniya Shynkevich, TM McGinnity, Sonya Coleman, and Ammar Belatreche. *Stock price prediction based on stock-specific and subindustry-specific news articles*. In 2015 International Joint Conference on Neural Networks (IJCNN), pages 1–8. IEEE, 2015.

[3] Robert P Schumaker and Hsinchun Chen. *Textual analysis of stock market prediction using breaking financial news: The azfin text system*. ACM Transactions on Information Systems (TOIS), 27(2):12, 2009.

[4] Salim Lahmiri. *Wavelet low- and high-frequency components as features for predicting stock prices with backpropagation neural networks*. Journal of King Saud University, 2014.

[5] Milosevic, Nikola. *Equity forecast: Predicting long term stock price movement using machine learning*. arXiv preprint arXiv:1603.00751 (2016).

[6] Nazish Nazir1 , Mudasirahma Dmutto2. *Stock Market Prediction Using Artificial Neural Network*. International Journal of Advanced Engineering, Management and Science (IJAEMS) Infogain Publication. 2016.

[7] Michael Hagenau, Michael Liebmann, and Dirk Neumann. *Automated news reading: Stock price prediction based on financial news using context-capturing features*. Decision Support Systems, 55(3):685–697, 2013.