Kaggle Competition: 2sigma Using News to Predict Stock Movements

Ziwen Wang [1], Yufan Liu [1], Jingyu Zhang [1]

[1]Tsinghua University, China

Introduction

As a scientifically driven investment manager, Two Sigma has been applying technology and data science to financial forecasts for over 17 years. Now, they're eager to engage with Kagglers in analyzing news data to predict stock prices. By understanding the predictive power of the news, this could help predict financial outcomes and generate significant economic impact all over the world. A large set of daily market and news data from is provided for US-listed financial instruments. This data shall be used to predict future stock market returns.

In our project, we want to apply different supervised learning algorithms learned from the course to **predict stock returns**. It is a good opportunity to put various algorithms at use and improve our understanding of statistic learning.

Setting of the Problem

The original competition comprised two stages: In the first stage the predictions were tested against historical data. It ended early next year at which time the final submissions must be handed in. Those then was evaluated against future data for about six months to identify the best performing submission which disclosed on 7/15/2019. The submission is no longer be accepted, so we evaluate the prediction results by cross validation instead of future data.

The objective function: For each day t within the evaluation period the value x_t is calculated as

$$x_t = \sum_i \hat{y_{ti}} \gamma_{ti} u_{ti} \tag{1}$$

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

Where the score(mean divided by standard deviation of daily prediction) is original used to determine the competition winner.

Exploratory Data Analysis

The datasets are provided, one for market data and one for news data, for about 3700 assets from 2007 to 2016. We use read datasets with vaex, and finally merge two datasets on 'date', 'assetCode' and 'assetName' after data cleaning for model preprocessing.

The **market data** provided by Intrinio with more than 4 million rows has 15 features, this dataset contains financial market information such as opening price, closing price, trading volume, calculated returns, etc. Each asset is identified by an 'assetcode', and a single company may have multiple 'asset Code's. The set of included instruments changes daily and is determined based on the amount traded and the availability of information. The number of all tradable asset at a given day ranges from roughly 1300 to1800. 'returnsOpenNextMktres10' is a 10 day market-adjusted -residualized return as the target variable used in competition scoring for the prediction task.

The **news data** provided by Thomson Reuters with more than 9 million rows has 37 features, this dataset contains information at both the news article level and asset level. The news articles/alerts are published about assets, such as article details, sentiment, and other commentary. 'sentimentClass' is a sentiment score analyzed from news.

Methods

After data cleaning and preprocessing, 'returnsOpenNextMktres10' in merged dataset is transformed into label for modeling. We normalize numerical data an encode codes, and split the dataset into training data and testing data for cross validation. The following supervised learning algorithms are applied for prediction:

Logistic Regression (LR)

Logistic regression is a classic and simple method for classification, thus we apply it as the baseline for algorithms contrast.

$$l = \log_b \frac{p}{1-p} = BX, \ \forall \ p = P(Y = 1) \tag{3}$$

Fully Connected Neural Network (FCNN)

We implement FCNN with Keras. After adjusting parameters, we finally determine to use 5 hidden layers. These hidden layers contain 4 Rectified Linear Unit (ReLU) activation functions, and a sigmoid function as last one layer. An embedding layer learns the encoding of the asset codes which are then concatenated to the numerical data. Binary- crossentropy is used as loss function and Adam optimizer is implemented with $learning\ rate = 0.001, \beta_1 = 0.9, \beta_2 = 0.999$. The model has about 50000 trainable parameters.

$$ReLU(x) = max(x, f(x)) \tag{4}$$

$$Sigmoid(x) = \frac{1}{1+e^{-x}} \tag{5}$$

$$Loss = -\frac{1}{output_{size}} \sum_{i=1}^{output_{size}} y_i log\hat{y_i} + (1-y_i) log(1-\hat{y_i}) \tag{6}$$

Light Gradient Boosting Machine (LGBM)

We implement LGBM with LightGBM. Binary-crossentropy is used as loss function is used with $sampling\ rate = 0.9$. The depth is set to be unlimited.

$$F = \sum_k F_k(x) \tag{7}$$

$$r_k = -[\frac{dL(y, F_{k-1}(x))}{dF_{k-1}(x)}] \tag{8}$$

$$L = \sum_i l(y_i, \hat{y_i}) + \sum_k \Omega(F_k) \tag{9}$$

eXtreme Gradient Boosting (XGBoost)

We implement XGBoost with xgboost. It is similar to LGBM, they both come from the gradient boosted trees. The difference between XGBoost and LGBM is the definition of objective function in (9). LGBM uses a novel technique of Gradient-based One-Side Sampling (GOSS) to filter out the data instances for finding a split value while XGBoost uses pre-sorted algorithm and histogram-based algorithm for computing the best split.

Result and discussions

We run the model with market data only and merged data to compare results in different model. Compare the LR results as baseline, we can find that accuracy is same in two data as score is better in merged data. Generally, FCNN, LGBM and XGBoost perform better than LR. The poor results of LR is expected because the algorithm assumes linear relationships. Adding news data doesn't improve the results. This may come from the reason news data has many noises and non-related features. Besides, too much missing value in the news features in merged data after joining two datasets. Although adding news data doesn't have noticeable effect on the performance, the results excepted LR in merged data are about or over 0.5. As for accuracy and score, scores are less than accuracies in LR as baseline, it may be that the score calculation considers not only accuracy, but also market return and standard deviation of the prediction. In practice, we need to evaluate stocks by score instead of accuracy. Comparing LR,FCNN,LGBM and XGBoost models, we can get two conclusions:

(1) In our situation, XGBoost model performs best. However, we don't have the original test data and only split the train and test data among the original train data, so XGBoost model's performance may be over estimated.

(2) The positive impact of news data remains unknown. In LR and XGBoost models, The news data seems to be of no effect. In FCNN model, the news data seems to have negative effect. While in LGBM model, he news data seems to have positive effect.

Table 1: Model contrast with market data only.

	LR	FCNN	LGBM	XGBoost
Accuracy	0.538	0.557	0.548	0.609
Score	0.325	0.671	0.403	1.957

Table 2: Model contrast with merge data.

	LR	FCNN	LGBM	XGBoost
Accuracy	0.538	0.553	0.56	0.609
Score	0.327	0.623	0.499	1.957

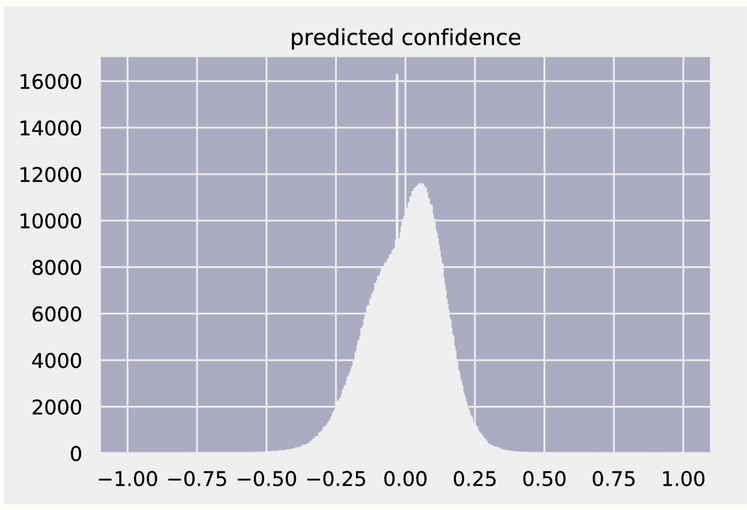

Figure 1: Predicted Confidence of FCNN

Summary and conclusions

Although we can improve the classification results in this project, the score isn't good enough. In the future, we need to implement more feature engineering to filter out outliers and useless features. Time-related model, such as LSTM, may be applied because the stock data is time series. Besides, we can also increase text information in news data by web crawler and analyze much on NLP. There are some ways that may help improve the performance.