2021 IEEE 6th International Conference on Intelligent Computing and Signal Processing (ICSP 2021)

Jane Street Stock prediction model based on LightGBM

Fangzhou Ye[1],
Fudan University,
Shanghai, China,
18300200004@fudan.edu.cn,

Jingru Wang[1]
University of Maryland
Maryland, United States
Jwang144@termail.umd.edu

Zhenning Li[2],
CentraleSupélec
Paris, France
zhenning.li@student-cs.fr

Zheng Jihan[3]
Jiangsu University of Science and Technology
Zhenjiang,China
983282619@qq.com

Congmin Yang*
The University of Sheffield
Sheffield, United Kingdom
Danielyangcm@163.com

2021 6th International Conference on Intelligent Computing and Signal Processing (ICSP) | 978-1-6654-0413-6/20/$31.00 ©2021 IEEE | DOI: 10.1109/ICSP51882.2021.9408851

***Abstract*—In recent years, the application of deep learning prediction models in predicting stock price trends has attracted extensive attention from researchers, which can help investors make better investment decisions. Therefore, we aims to use the data set provided by Jane Street to establish a stock forecast model and develop a trading strategy. In this paper, we proposed a stock prediction model based on LightGBM. Experiments show that our model has better predictive power and higher returns compared with other machine learning models. Specifically, our LightGBM-based stock prediction model has a return rate of 24.7% higher than that of the Xgboost model and 8.9% higher than that of the Nerual Network.**

***Index Terms*—Sales Forecast,LightGBM, Machine learning algorithms, Data Mining,**

I. INTRODUCTION

In the stock market, price fluctuation is related to the national macroeconomic development, the formulation of laws and regulations, the operation of the company, and the confidence of stockholders, so stock forecasts are very difficult. The stock price refers to the price at which the stock is sold on the market. Its fluctuations are subject to various factors including economic and political, and are affected by people's investment psychology and trading techniques. In summary, the factors that affect the stock market price and its fluctuations are mainly divided into two categories: one is the basic factor; the other is the technical factor.

Because of the complexity of the factors affecting share prices, manual forecasting is difficult. The machine learning algorithm can organize various features efficiently, which is very suitable for the task of stock prediction. In this paper, we used the data set provided by Jane Street to build a stock prediction model based on LightGBM. Based on the relevant information, the model can give advice on whether to conduct stock operation.

We first preprocessed the data set, including normalization and standardization of the data, and then supplement the missing values in the data. Since the target value given in the data set is not a single value, our target value is based on the integration of several different features. After the data processing, we built a model based on LightGBM, and then conducted model training.

In summary, the contributions of this paper are as follows:

●We proposed a stock prediction model based on LightGBM, which can make suggestions on whether to operate stocks based on the information given.

●In the experimental process, we carried out detailed feature engineering processing, including data standardization and normalization, important feature selection and target feature processing .

●We evaluated the LightGBM method on the data set provided by Jane Street. Experiments show that our method is better than the XGBoost algorithm and the Nerual Network method. The cumulative returns of these methods on different time scales are summarized and compared.

A. Related Work

The XGBoost [1] algorithm supports both pre-classification algorithm and histogram algorithm. In order to reduce the size of the training data, we first sampled the data. For example, in [2], if the weight of the data is less than a fixed threshold, they will be filtered. The specific operation of the SGB algorithm is to use a random subset to train a weak classifier in each iteration. Although SGB can be used in the GBDT algorithm, it usually reduces the accuracy of the model, and is not a good choice. At the same time, in order to reduce the number of features during data preprocessing, we usually delete some unimportant features through principal component analysis [3，4]

978-1-6654-0413-6/21/$31.00 ©2021 IEEE

GBDT is the overall model of the decision tree, which will be trained in the order of [5]. In each iteration, GBDT learns a decision tree by fitting a negative gradient. The most time-consuming part of the GBDT model is finding the optimal splitting point. Algorithms based on histograms [6，7，8] are one of the better-performing methods.

The rest of this paper is organized as follows. The second part introduces the method of data preprocessing. The third part mainly introduces the model LightGBM we used in this paper, as well as the design idea and characteristics of the model. Then, in the fourth part, we conducted experiments on the data set provided by Jane Street and compared our model with other classic models. Finally, the conclusion of this paper and the future work are expounded.

II. Feature Engineering

Feature engineering is a technology that can present data like art. Good feature engineering combines domain knowledge, intuition, and basic mathematics. Essentially, the data presented to the algorithm should have the relevant structure or properties of the basic data. In data modeling, data cannot be found well if all attributes of the original data are taken into account. If the data are preprocessed by feature engineering, the noise interference can be reduced during modeling so that the trend can be better detected.

The methods commonly used in feature engineering are as follows:

(1) Timestamp processing

Timestamps typically need to be divided into multiple dimensions such as year, month, day, hour, minute, and second. But many applications do not require a lot of information, so the data provided with time must be sure to be what the model requires, and the impact of time zones cannot be ignored.

(2) Discrete variable processing

For the processing of discrete variables, the most common method is to convert each variable value into a multi-dimensional number represented by 0 and 1, which is often referred to as one-hot code.

(3) Zoning and binning

Sometimes, it makes more sense to convert continuous variables into categories, while enabling the algorithm to reduce noise interference. This division makes sense only when the domain knowledge base of the variable is understood and the attribute can be divided into a concise range, that is, all the values belonging to a partition can present common characteristics. In practical applications, partitioning can avoid overfitting. Binning can also reduce the impact of errors. The specific operation is to divide a given value into the nearest block. If the number of division ranges is close to all possible values, or if accuracy is important to us, binning is not appropriate at this time.

(4) Cross feature

Cross feature is one of the most important methods in feature engineering. Its specific operation is to combine two or more category attributes into a single feature. This is a very useful technique when combined features are superior to individual features. Mathematically, this is the cross-multiplication of all the values of the category feature .

(5) Feature selection

In order to get a better model, some algorithms are used to automatically select a subset of the original features. In this process, features are not added or removed, but are modified in order to reduce noise and redundancy.

Feature selection algorithms use scoring methods to rank and select features, such as relevance or other methods to determine the importance of features. Further methods may require trial and error to find a subset of features.

There is another method of constructing auxiliary models. Stepwise regression is an example of automatic feature selection algorithms during model construction. There are also regularization methods such as Lasso regression and ridge regression included in feature selection by adding additional constraints. Alternatively, penalty terms can be added to the existing model to prevent over-fitting and improve the generalization ability.

(6) Feature zoom

It can be noted that sometimes certain features have a much higher span value than others. Certain models, such as ridge regression, require us to scale the eigenvalues to the same range of values. Feature scaling can prevent some features from obtaining very different weight values.

(7) Feature extraction

Feature extraction involves a series of algorithms that automatically generate some new feature sets from the original attributes, and dimensionality reduction algorithms belong to this category. Feature extraction is a process of automatically reducing the dimension of observations to a small data set sufficient for modeling. For tabular data, the available methods include some projection methods, such as principal component analysis and unsupervised clustering algorithms. For graphic data, it may include some straight line monitoring and edge detection, and there are different methods for different fields.

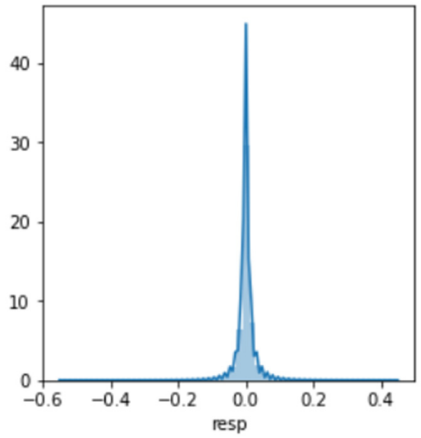

Figure1.feature distribution

III. LIGHTGBM

As one of the three famous GBDTS implementations, LightgBM not only has the advantage of fast running speed, but also has high prediction accuracy. Specifically, LightGBM does not need to calculate information gain through all samples, and has built-in feature dimensiona reduction technology, which makes the speed faster. The aforementioned XGBoost also uses the histogram method to speed up. It only needs to traverse a few possible split nodes, without having to calculate all the samples. Although, XGBoost only needs to traverse a few possible split nodes each time, then compare the information gain of each split node, and select the largest one for splitting, the comparison needs to consider the information gain of all samples. In contrast, LighttGBM only needs a small number of samples to calculate the information gain, which is much faster.

However, the above method will bring a problem. The tradeoff selects a small number of samples to calculate information gain and minimize information loss, The optimization method adopted by LightGBM is to keep some small gradient samples randomly while keeping large gradient samples, and at the same time amplify the information gain brought by small gradient samples. The specific process is to sort the samples according to the gradient, select the a% samples with the largest gradient, and then select b% samples randomly from the remaining small gradient data. When calculating the information gain, b% small samples will be selected. The information gain of the gradient samples is expanded by 1-a / b times. This will avoid changes to the data distribution. In addition to calculating information gain through partial samples, LightGBM also has built-in feature dimension reduction technology. The specific operation method is to merge sparse features with fewer conflicts. If the features are abstracted as points in the graph, and conflicts between features are regarded as edges in the graph, then the problem is converted to finding the communities in the graph and minimizing the number of communities in the graph. A greedy strategy is proposed in LightGBM, which sorts all the points in the graph by authority, and then merges the features into a community whose degree is less than a certain threshold or creates a separate community.

The reason LightGBM is more accurate than XGBoost we thought is that it uses Adaboost's idea of changing the weight of samples to select samples with large gradients for feature segmentation to generate trees. Each tree has a better ability to divide specific training samples, resulting in greater heterogeneity between each tree. Weight models have similar effects, but greater heterogeneity tends to lead to greater improvements.

IV. EXPERIMENTS

In order to comprehensively evaluate the performance of our proposed method, we used the same evaluation indicators and data sets to conduct experiments on classic algorithms and models, and compared their experimental results. In this paper, the calculation formula we used to evaluate the performance of the model is as follows:

$$p_i = \sum_j (weight_{ij} * resp_{ij} * action_{ij})$$

$$t = \frac{\sum p_i}{\sqrt{\sum p_i^2}} * \sqrt{\frac{250}{|i|}}$$

where |i| is the number of unique dates in the test set. The utility is then defined as:

$$u = min(max(t, 0), 6) \sum p_i.$$

Table 1 shows the experimental results of our proposed neural network model compared with other models.

Table 1. The results of different models used for stock prediction tasks.

Models	Unity
LightGBM	**7487**
Xgboost	6005
Nerual Network	6876

It can be seen that using the neural network model to make a decision at each step of the tradable node will get the highest return value.

The experimental results show that compared with the Xgboost algorithm and the Nerual Network algorithm, the stock prediction model we proposed based on LightGBM has better performance, and the investment suggestions given by our model will get higher returns.

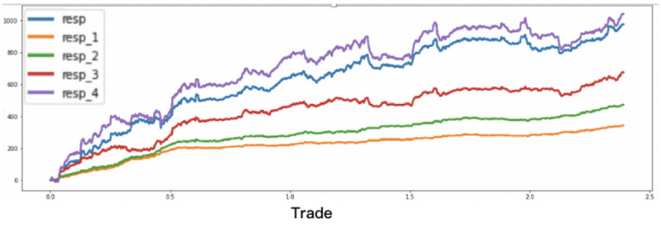

Figure 2. Cumulative returns on different time scales

V. CONCLUSIONS

In this paper, we propose a new method of stock prediction based on LightGBM. We use the data set provided by Jane Street to build stock prediction models and develop trading strategies. During the experiment, we performed detailed feature engineering processing, such as data standardization and normalization, important feature selection, and target feature processing. Experiments show that compared with the Xgboost model and the Nerual Network model, the proposed method can effectively mine features of different dimensions and has better performance. In addition, we studied the cumulative returns of the same model over different time scales, and obtained some interesting and constructive guidance.

REFERENCES

[1] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In Proceedings of the 22Nd ACM SIGKDD International

Conference on Knowledge Discovery and Data Mining, pages 785–794. ACM, 2016.

[2] Jerome Friedman, Trevor Hastie, Robert Tibshirani, et al. Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). The annals of statistics, 28(2):337–407, 2000.

[3] Ian Jolliffe. Principal component analysis. Wiley Online Library, 2002.

[4] Luis O Jimenez and David A Landgrebe. Hyperspectral data analysis and supervised feature reduction via projection pursuit. IEEE Transactions on Geoscience and Remote Sensing, 37(6):2653–2667, 1999.

[5] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. Annals of statistics, pages 1189–1232, 2001.

[6] Sanjay Ranka and V Singh. Clouds: A decision tree classifier for large datasets. In Proceedings of the 4th Knowledge Discovery and Data Mining Conference, pages 2–8, 1998.

[7] Ruoming Jin and Gagan Agrawal. Communication and memory efficient parallel decision tree construction. In Proceedings of the 2003 SIAM International Conference on Data Mining, pages 119–129. SIAM, 2003.

[8] Ping Li, Christopher JC Burges, Qiang Wu, JC Platt, D Koller, Y Singer, and S Roweis. Mcrank: Learning to rank using multiple classification and gradient boosting. In NIPS, volume 7, pages 845–852, 2007.

[9] Jerome Friedman, Trevor Hastie, Robert Tibshirani, et al. Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). The annals of statistics, 28(2):337–407, 2000..