ATLANTIS PRESS

Advances in Economics, Business and Management Research, volume 211

Proceedings of the 2022 7th International Conference on Financial Innovation and Economic Development (ICFIED 2022)

A Model Combining LightGBM and Neural Network for High-frequency Realized Volatility Forecasting

Xiang Zhang*

Department of Information Science and Technology, East China University of Science and Technology, 200000, China

**Corresponding author. Email: 19001665@mail.ecust.edu.cn*

ABSTRACT

The financial market is a nonlinear and frequently changing complex dynamic. Volatility, as one of the important indicators to measure the return of financial assets, occupies an indispensable position in the field of financial measurement. With the development of machine learning and massive data technology, there is an increasing demand for volatility prediction. In this paper, an ensemble learning model mainly based on the LightGBM algorithm and supplemented with a neural network is constructed. The model achieves the prediction of high-frequency realized volatility using ultra-high frequency stock market data and through the method of moving windows in finance. The superiority of the LightGBM-NN model is verified by comparing it with the single LightGBM model. The LightGBM-NN model produces less error and has higher accuracy, precision, and F1 score. The lightGBM-NN model has advanced the application of LightGBM in the field of financial measurement, which brings new ideas on how to handle the massive data efficiently and fast in the stock market.

Keywords: *Realized volatility forecasting, Ensemble learning model, LightGBM, Neural network*

1. INTRODUCTION

The main method of stock investment is to obtain the maximum return under the premise of limited control and resistance to risks, or to obtain the minimized risk under the premise of maintaining a certain return, that is, to maximize the investment expectations. Therefore, because of the question of how to correctly choose the way and timing of stock price trading, finding an efficient and quantitative stock price investment strategy that is better suitable for investment is particularly important, so that it can not only fully meet investors' expectations for stock returns, but also effectively avoid investment risk and loss reduction.

With the further expansion of the size of the securities market and the further increase of the securities trading quota, it is no longer possible to analyze the huge information of the stock market accurately and effectively only by relying on traditional statistics knowledge and computer technology. In the face of the huge amount of financial data, what methods and algorithms should be used to extract and analyze effective information is a challenge for many investment companies.

Volatility, as a measure of uncertainty of financial asset returns, is of vital importance for stock portfolio selection, enhanced risk management, option pricing, etc. The estimation of volatility has long been of great interest in the field of financial econometric research. Benefiting from the much lower cost of acquiring and storing high-frequency data, the application of high-frequency data has become a new entry point in the field. The realized volatility that can be obtained using high-frequency data can be approximated as the actual volatility of financial asset returns.

Gradient Boosting Decision Tree (GBDT) is a long-lasting model in machine learning. Its main idea is to use a weak classifier (Decision Tree) to obtain the optimal model by iterative training. And Light Gradient Boosting Machine (LightGBM) is a framework that implements the GBDT algorithm. It supports efficient parallel and distributed training, which means it has a faster training speed. It also consumes less memory, making it possible to process massive data quickly. In addition, it has better accuracy. Since the advent of LightGBM, it has become popular in the industrial sector, but it has not been widely used in other fields. However, from its application in the industry, it can be seen that LightGBM can also do a lot in various fields and has a great research value.

As a highly complex nonlinear system, the neural network can identify more complex features in a sample. This means that neural networks can capture and model multiple features that interact in financial markets. Also, due to its inherent construction, a neural network is well suited to handle sequential problems.

Merton analyzed three equilibrium expected market return models that reflect the relationship between market returns and changes in the level of risk associated with the market. He formally introduced the concept of realized volatility, linking it to financial markets [1]. Andersen et al. formally introduced the concept of realized volatility. They pointed that realized volatility can significantly reduce the error and noise in the volatility estimation process compared to low-frequency volatility [2]. Andersen et al. established a formal link between realized volatility and the conditional covariance matrix using the theory of continuous-time arbitrage-free price processes and quadratic variance theory. They used linked recorded spot exchange rate data to integrate high-frequency data into the measurement and forecasting of relatively low-frequency volatility. They found that realized volatility can be approximated as actual volatility under the condition that the return series can approximately satisfy the assumption of zero means and the sample tends to infinity [3].

R.French et al. examined the relation between stock returns and stock market volatility. They use two different statistical approaches, the autoregressive-integrated-moving-average (ARIMA) models and generalized-autoregressive-conditional-heteroskedasticity (GARCH) models. Through the experimental studies, they find that the expected return on a stock portfolio is positively related to the predictable volatility of stock returns, and unexpected stock market returns are negatively related to the unexpected change in the volatility of stock returns [4]. White pointed that the conclusions of econometric model determination based on out-of-sample prediction results are more reliable and practical than those based on in-sample prediction results. This provides the idea of using out-of-sample prediction results with dynamic rolling time windows instead of in-sample prediction results based on a single static sample for model predictive power testing [5]. Ke et al. had proposed a novel GBDT algorithm called LightGBM to deal with a large number of data instances and a large number of features respectively. LightGBM contains two novel techniques: Gradient-based One-Side Sampling and Exclusive Feature Bundling. They used five different public datasets to perform both experimental studies and theoretical analysis on these two techniques. The datasets were relatively large and contained sparse and dense features, covering real businesses. The experimental results show3e that with the help of GOSS and EFB, LightGBM performs much better than XGBoost and SGB on computational speed and memory consumption [6]. Dauod investigated and compared the efficiency of three gradient methods, XGBoost, LightGBM, and CatBoost. He chose the home credit dataset and used several techniques to rank and select the best features. The experimental results showed that the LightGBM performs better than XGBoost and Catboat in speed and accuracy [7]. Hassan et al. used daily stock prices as the input features of the neural network and the output of the network as the input to a Hidden Markov model to predict stock prices for a single day. The optimization of the parameters of the Hidden Markov model was done by a genetic algorithm. Their experiment showed that this fusion model can achieve similar prediction performance as the ARIMA model, demonstrating the applicability of machine learning in dealing with price time-series data. The applicability of machine learning in processing time-series data was demonstrated [8]. Ballings et al. investigated the performance of integrated learning algorithms to construct and combine multiple classifiers of the same type to predict the annual trend of asset prices. They demonstrated that, with appropriate parameters, the classification results of the integrated learning algorithm outperformed those of a single algorithm [9]. Baral et al. investigated the integration of different types of single classifiers (including SVM, NN, and decision trees), using a method based on dataset clustering and waiting for classifier accuracy to determine the number and type of base classifiers. In addition, by comparing three integration learning methods, which were Bagging, Boosting, and Ada-Boost, they demonstrated that the models integrated using the Bagging method achieved the highest prediction accuracy [10].

The objective of this paper is to explore the application of LightGBM in financial markets. An ensemble model mainly based on LightGBM is constructed to forecast realized volatility over 10-minute periods.

2. METHOD

2.1.The introduction of LightGBM

2.1.1.The motivation of LightGBM

LightGBM is an algorithm based on Gradient Boosting Decision Tree (GBDT) designed by Microsoft in 2017. In the GBDT algorithm, entire training data needs to be traversed multiple times in each iteration, so the size of training data is limited by memory when the entire training data is loaded into the memory. If data is not loaded, it will consume a lot of time to read the data repeatedly. The main reason LightGBM put forward is that the ordinary GBDT algorithm cannot meet needs, especially in the face of industrial-grade massive data. LightGBM can help solve this problem so that GBDT can be used in industrial practice better and faster.

2.1.2.Principles of LightGBM

LightGBM uses a decision tree algorithm based on the Histogram. For the Histogram, an algorithm based on feature discretization, the two most outstanding advantages are a smaller memory footprint and lower computational cost. The algorithm is implemented as Fig 1. First, the continuous floating-point eigenvalues are discretized into k integers and a histogram with a width of k is constructed. The discretized values are used as the indexes to accumulate statistics in the histogram when traversing the data. After it, the histogram accumulates statistics, and then according to the discrete value of the histogram, the optimal split point can be found by traversing. The segmentation points are not very accurate after the features are discretized. However, the results on different data sets show that the discretized segmentation points do not have a great impact on the final accuracy, and sometimes the result is even better. The reason is that the decision tree is inherently a weak model, and it is not too important whether the split points are accurate. Even if the training error of a single tree is greater than that of the precise segmentation algorithm slightly larger, it does not have much impact under the framework of Gradient Boosting. In addition, the thicker split points also have a regularization effect, which can effectively prevent overfitting.

LightGBM discards the Level-wise decision tree growth strategy used by most GBDT tools and uses the Leaf-wise algorithm with depth limitation. Fig 2 shows the difference between the Level-wise and the Leaf-wise. The Level-wise algorithm splits all the leaves of the level in the process of traversing the data once, while the Leaf-wise algorithm chooses the leaf with the greatest splitting gain from the leaves of the level each time, and then splits, and so on. Therefore, in the case of the same number of splits, Leaf-wise can reduce more errors and guarantee better accuracy. It also saves unnecessary computational overheads. However, the Leaf-wise algorithm may grow deeper decision trees and produce overfitting. So LightGBM adds a maximum depth limit on top of Leaf-wise to prevent overfitting while ensuring high efficiency.

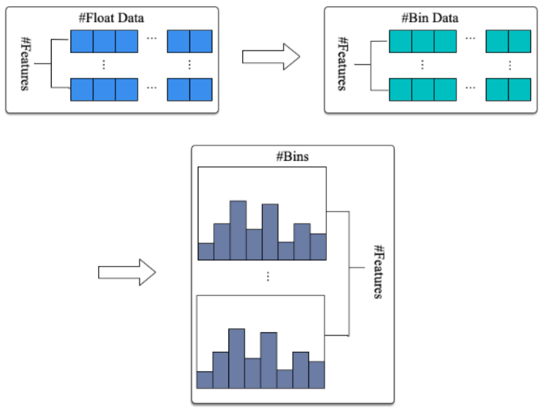

Figure 1 Histogram algorithm

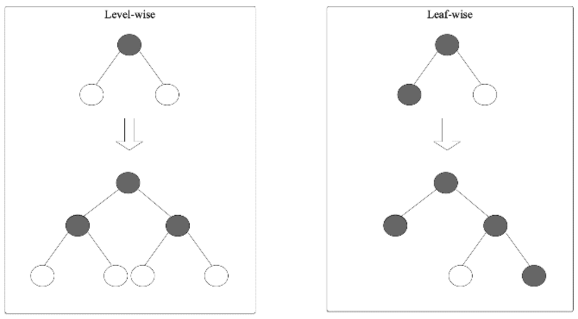

Figure 2 Comparison of Level-wise and Leaf-wise

LightGBM uses GOSS as the sampling algorithm to discard some samples that do not help compute the information gain and leave the helpful part. GOSS can pay more attention to the under-trained samples without changing the distribution of the original data set too much. It is worth mentioning that in the LightGBM algorithm, the memory is saved by not having to save the sorted results.

EFB is a lossless method that can reduce the dimensionality of sparse high-dimensional data. EFB can reduce the number of features by fusing and binding them. By adding a bias constant to the eigenvalue, the values of different features can be divided into different bins in the bundle when running the Histogram-based algorithm.

On top of EFB, LightGBM proposes a more efficient non-graph sorting strategy, sorting the features according to the number of non-zero values.

2.2.LightGBM-NN

2.2.1.Model introduction

The model constructed in this paper combines the LightGBM algorithm and neural network. The ensemble process is shown in Fig 3.

After importing the dataset, the order book data and trade data are processed separately for feature engineering. Data preprocessing is parallelized and looped to save runtime and improve efficiency. The k-means clustering is achieved by using the features of the training data and obtaining the clustering centers. Then the training set and the test set are transformed by the obtained clustering centers to get the classification results of each piece of data. The result of the clustering is merged into the original data and becomes a created feature. Before adopting the model for formal training, data consolidation is needed to match the model. LightGBM and neural networks are fused in a summative manner into a model called LightGBM-NN. And the LightGBM-NN model is used to predict realized volatility in this paper.

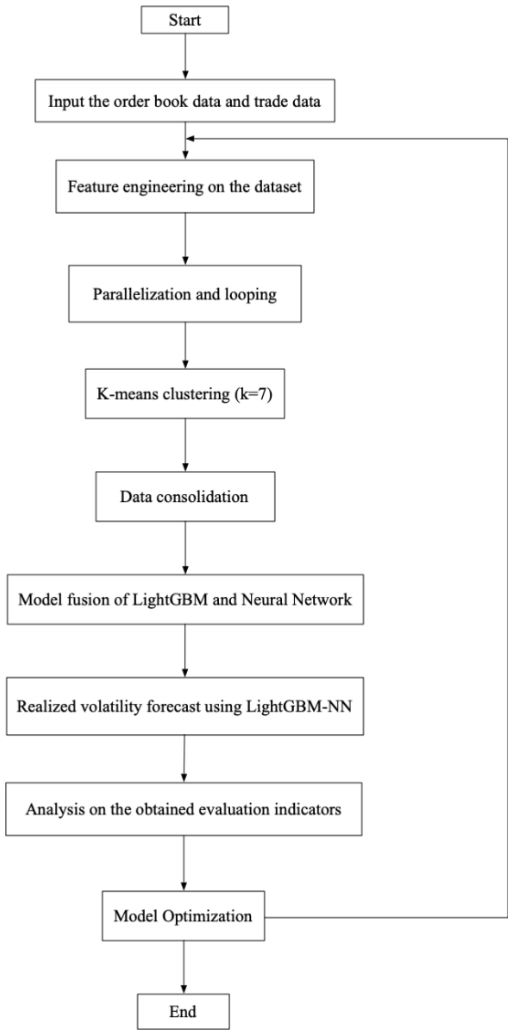
Start
Input the order book data and trade data
Feature engineering on the dataset
Parallelization and looping
K-means clustering (k=7)
Data consolidation
Model fusion of LightGBM and Neural Network
Realized volatility forecast using LightGBM-NN
Analysis on the obtained evaluation indicators
Model Optimization
End

Figure 3 The process of LightGBM-NN

Also, the values of the evaluation metrics are obtained during the training process. Based on these values, optimization of the model parameters, removal of less relevant features, refinement of the steps, etc. can be carried out to improve the quality of the model.

2.2.2.Parameter settings

The key parameters of LightGBM-NN used in the experiment are summarized in Table 1. Through continuous operation and optimization, these parameters are set to relatively reasonable values after comprehensive consideration. Max_bin is set to 100 to speed up while avoiding overfitting. Num-leaves and max_depth limit the breadth and depth of a tree. And the former value needs to be less than or equal to twice the latter value to ensure that no overfitting occurs. Since the dataset applied in the experiment is very large, min_data_in_leaf is set to 500 to avoid underfitting. Learning_rate is set to 0.1 so that the speed of moving the parameters to the optimal value is relatively fast while ensuring convergence.

3. DATA ANALYSIS AND PREPROCESSING

3.1.Sample overall exploration

The dataset used in this paper is real data provided by Optiver, a leading global electronic market maker. It contains stock market data, including order book snapshots and executed trades, which is relevant to the practical execution of trades in the financial markets. The one-second resolution of the dataset provides a fine-grained look at the microstructure of the modern financial market. The target values of the train set are roughly 150,000, with which the model is designed forecasting volatility over 10-minute periods. The model is evaluated against real market data collected in the three-month evaluation period after training.

The order book data shows the most competitive buy and sell orders entered into the market. Its features include stock ID, time ID, bucket seconds, bid price, ask price, bid size, and ask size.

While the trade data shows the trades that were executed. Its features include stock ID, time ID, bucket seconds, price, size, and order count. Because there are fewer actual trades than passive buy or sell intention updates in the market, the trade data is more sparse than the order book data. Besides, the prices have been normalized and their averages have been weighted by the number of shares traded in each transaction.

Table 2 shows some of the important given data features in the order book data and the trade data. The specific interpretations of the data are also shown in Table 2.

Table 1 LightGBM-NN parameter settings

Parameter	Description	Value
max_depth	The maximum depth of a tree	5
max_bin	The maximum number of features to be stored in a bin	100
min_data_in_leaf	The minimum number of records a leaf may have	500
num_leaves	The maximum number of leaves on a tree	10
feature_fraction	The proportion of features used in each iteration	0.8
bagging_fraction	The proportion of data used in each iteration	0.8
learning_rate	The rate of learning	0.1
output_dim	The vector dimension	24
units	The spatial dimension	128, 64, 32
activation	The activation function	'linear'

Table 2 Description of provided data features

Feature	Description
Stock ID	ID code of the stock
Time ID	ID code of the time bucket
Bucket seconds	Number of seconds from the start of the bucket
Bid price 1	Normalized prices of the most competitive buy level
Bid price 2	Normalized prices of the second most competitive buy level
Ask price 1	Normalized prices of the most competitive sell level
Ask price 2	Normalized prices of the second most competitive sell level
Price	The average price of executed transactions happening in one second
Bid size 1	The number of shares on the most competitive buy level
Bid size 2	The number of shares on the second most competitive buy level
Ask size 1	The number of shares on the most competitive sell level
Ask size 2	The number of shares on the second most competitive sell level
Size	The sum number of shares traded
Order Count	The number of unique trades taking place

3.2.Feature Engineering

Data features can directly affect the predictive performance of the model. Since the experimental results receive many interdependent attributes, there is a need to create good features that can describe the internal structure of the data well. The final target is the realized volatility, so a series of operations are performed on the provided data to make the correlation between the created features and the target higher. Some of the necessary computational processing is explained below.

Each sort of price spread represents a unique meaning.

$$Price\ Spread_n = \frac{AskPrice_n - BidPrice_n}{\frac{AskPrice_n + BidPrice_n}{2}} \quad (1)$$

$$Bid\ Spread = BidPrice_1 - BidPrice_2 \quad (2)$$

$$Ask\ Spread = AskPrice_1 - AskPrice_2 \quad (3)$$

$$Bid\ Ask\ Spread = |BidSpread - AskSpread| \quad (4)$$

Total volume and volume imbalance are indicators of size.

$$Total\ volume = AskSize_1 + AskSize_2 + BidSize_1 + BidSize_2 \quad (5)$$

$$Volume\ imbalance = (AskSize_1 + AskSize_2) - (BidSize_1 + BidSize_2) \quad (6)$$

Weighted averaged price (WAP) is the weighted price that combines the bid price and the asking price.

$$\mathrm{WAP_n} = \frac{\mathrm{BidPrice_n} \times \mathrm{AskSize_n} + \mathrm{AskPrice_n} \times \mathrm{BidSize_n}}{\mathrm{BidSize_n} + \mathrm{AskSize_n}} \quad (7)$$

WAP balance measures the weighted averaged price gap between the most and the second most competitive level.

$$WAP\ Balance = WAP_1 - WAP_2 \quad (8)$$

Log returns are the logarithmic rate of return as distinguished from the percentage rate of return.

$$r_{t_1,t_2} = log\left(\frac{S_{t_2}}{S_{t_1}}\right) \quad (9)$$

Equation (9) realized volatility is calculated by taking the weighted averaged price as the price, thus calculating the log-returns of each time bucket and finally calculating the log returns' variance.

$$\sigma = \sqrt{\sum_t r_{t-1,t}^2} \quad (10)$$

4. RESULTS AND DISCUSSION

4.1.Applied evaluation criteria

To fully verify the performance of the proposed model, two widely used forecast accuracy evaluation criteria are chosen to compare with LightGBM and LightGBM-NN in the experiment. The smaller value means a better forecasting effect. These criteria are the Root Mean Square Error (RMSE) and the Root Mean Square Percentage Error (RMSPE). Each criterion is defined as the following formula:

$$\mathrm{RMSE} = \sqrt{\frac{1}{\mathrm{N}}\sum_{\mathrm{i}=1}^{\mathrm{N}} |\mathrm{y_i} - \mathrm{y_i}'|^2} \quad (11)$$

$$\mathrm{RMSPE} = \sqrt{\frac{1}{\mathrm{N}}\sum_{\mathrm{i}=1}^{\mathrm{N}} \left|\frac{\mathrm{y_i} - \mathrm{y_i}'}{\mathrm{y_i}}\right|^2} \quad (12)$$

Precision, recall, and F1-score are used as measures to evaluate the output quality of the model. Higher precision is related to a lower false-positive rate and higher recall is related to a lower false-negative rate. Higher scores for both show that the model is returning more accurate results, as well as returning a majority of more positive results. To take the precision and recall of the model into account, the F1-score is used to evaluate the analytical effectiveness and comprehensive performance of the model. The formula for each indicator is as follows:

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive} \quad (13)$$

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative} \quad (14)$$

$$F_1\ score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \quad (15)$$

4.2.Results analysis

To avoid overfitting and to ensure the generalization performance of the model, cross-validation is used for model optimization and result evaluation. A 5-fold multiple division is used to utilize the full data sets, and finally, an averaging method is used to represent the model performs reasonably. Meanwhile, the selected algorithms require a sufficient amount of data and training times to guarantee the model quality, so the models are continuously optimized through parallelization and looping. To minimize randomness, the models of each fold are trained until validation scores do not improve for 50 rounds.

Table 3. Evaluation results

	LightGBM	LightGBM-NN
RMSE	0.0021	0.0012
RMSPE	0.2575	0.2115
Accuracy	0.978	0.986
Precision	0.971	0.982
Recall	0.967	0.970
F1 score	0.969	0.977

Table 3 shows the evaluation results of LightGBM and LightGBM-NN. It can be seen that the RMSE and RMSPE of the LightGBM-NN model are both lower than those of the LightGBM model, which indicates that compared to LightGBM, the prediction of LightGBM-NN produces fewer errors. Meanwhile, LightGBM-NN has higher accuracy and precision of 0.986 and 0.982 respectively. The precisions of LightGBM and LightGBM are 0.971 and 0.982, which are almost the same. The F1 score can reflect the overall ability of the model to some extent. And the F1 score of LightGBM-NN is higher than that of LightGBM. Fig 4 shows the predicted value and the actual value of a stock's realized volatility.

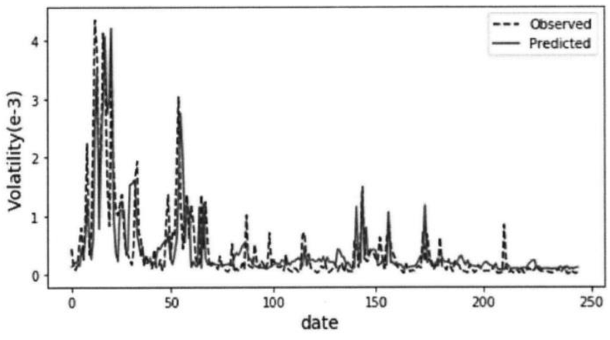

Figure 4 Realized volatility forecasting

4.3.Discussion

Both LightGBM and neural network algorithms have their unique strengths. The ensemble learning model allows them to pool their strengths and fill their shortcomings. Therefore, the LightGBM-NN model has better quality than a single algorithm.

First of all, LightGBM is an algorithm that is very suitable for processing massive data, and it has good predictive power while ensuring high operational efficiency. The neural network, by its construction, is well suited to handle sequential problems. Both LightGBM and neural networks can implement complex nonlinear mappings and can solve the problem with complex internal mechanisms. The relationship between the features and the target cannot be determined in this experiment. The application of these two models facilitates the construction of correlations among them, thus improving the model. After the above analysis, it can be seen that the LightGBM-NN model is well suited for predicting the realized volatility of the stock market.

However, LightGBM-NN still has shortcomings. It requires more memory to support, which places a higher demand on the running devices. In addition, it consumes more computational resources. Fortunately, the computational complexity of the LightGBM and the neural network is not particularly high, so the running time does not increase much.

5. CONCLUSION

This paper proposed a realized volatility forecasting model based on LightGBM and neural networks. The model achieves the realized volatility forecasting over 10-minute periods by using approximately 150000 rows of ultra-high frequency stock market data at a one-second resolution. The superiority of the LightGBM-NN model is verified by comparing it with the single LightGBM model. The LightGBM-NN model produces less error and has higher accuracy, precision, and F1 score.

LightGBM itself is well suited to be promoted in the financial field as an efficient and massive data processing algorithm with good learning ability. The use of some algorithms that are more suitable for specific problems is beneficial to further improve the capability of the ensemble learning model. In this paper, the neural networks algorithm, which is suitable for handling sequential problems, is selected to assist LightGBM in predicting the realized volatility. The ensemble model has advanced the application of LightGBM in the field of financial measurement, which brings new ideas on how to handle the massive data efficiently and fast in the stock market.

In future work, it is worth considering how to shrink the more computational resources consumed by LightGBM when it works together with other algorithms. This will be beneficial to further reduce the model running time and improve the efficiency of processing problems when dealing with massive data.

REFERENCES

[1] Merton, R. C. (1980). On estimating the expected return on the market: An exploratory investigation. Journal of financial economics, 8(4), 323-361.

[2] Andersen, T. G., & Bollerslev, T. (1998). Answering the skeptics: Yes, standard volatility models do provide accurate forecasts. International economic review, 885-905.

[3] Andersen, T. G., Bollerslev, T., Diebold, F. X., & Labys, P. (2003). Modeling and forecasting realized volatility. Econometrica, 71(2), 579-625.

[4] French, K. R., Schwert, G. W., & Stambaugh, R. F. (1987). Expected stock returns and volatility. Journal of financial Economics, 19(1), 3-29

[5] White, H. (2000). A reality check for data snooping. Econometrica, 68(5), 1097-1126.

[6] Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., ... & Liu, T. Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. Advances in neural information processing systems, 30, 3146-3154

[7] Al Daoud, E. (2019). Comparison between XGBoost, LightGBM and CatBoost using a home credit dataset. International Journal of Computer and Information Engineering, 13(1), 6

[8] Hassan, M. R., Nath, B., & Kirley, M. (2007). A fusion model of HMM, ANN and GA for stock market forecasting. Expert systems with Applications, 33(1), 171-180.

[9] Ballings, M., Van den Poel, D., Hespeels, N., & Gryp, R. (2015). Evaluating multiple classifiers for stock price direction prediction. Expert systems with Applications, 42(20), 7046-7056.

[10] Barak, S., Arjmand, A., & Ortobelli, S. (2017). Fusion of multiple diverse predictors in stock market. Information Fusion, 36, 90-102.