Stock Volatility Prediction with Hybrid Model

Yiyang Zhang
University of Michigan, MI, United States of America
zhyiyang@umich.edu

Abstract

The existing prediction system for realized volatility is limited and cannot effectively describe the stocks' highly complex and nonlinear characters. In this study, we built a hybrid model by combining Feedforward Neural Network (FFNN) with Light Gradient Boosting Machine (LightGBM). Then we extract three important categories of features based on high frequency stock trading and quotation data, feed them into the hybrid model for predicting volatility, and test it on the real-market data in the next three months. We also compared our hybrid model with other models in the experiment process. Compared with traditional machine learning models like Naïve Bayes and SVM, or the single Lightgbm model, our hybrid model has the lowest RMSPE result of 0.192. And in the following three-month real-market data test, our hybrid model's RMSPE result remained in range $[0.199, 0.219]$. This test result further demonstrates the accuracy and robustness of our model's out-of-sample performance.

Keywords: Realized volatility, High Frequency Data, LightGBM, FFNN, RMSPE, real-market test

1 Introduction

With the development of the social economy, people's concept of investment and financial management has gradually strengthened. The stock market and its corresponding derivative markets are important parts of the financial market and attract many investors' attention. As stocks' volatility is one of the most important factors for stock trading, corresponding derivative pricing, and risk management, researchers have great interest in this field.

Furthermore, with the boom in machine learning and deep learning, they have started to play a big role in financial trading. Since the late 1980s, scientists have started to use deep learning techniques in financial market prediction (White, 1988). Since then, after more than 30 years of development, more technologies in machine learning and deep learning have been gradually applied in the financial field, and the scope of application has gradually expanded, and now it has penetrated into every corner of the financial field.

In our study, we researched the data set provided by Optiver on the Kaggle data science competition, which is a Limit Order Book data set that anonymized stock names. Optiver is one of the biggest market maker companies globally, and the competition aims to use machine learning and deep learning techniques to help predict stock volatility, which is essential for their further market-making strategies and option pricing models.

Our paper first does feature engineering based on stock trading and quotation data. Next, we proposed a hybrid model based on FFNN and LightGBM. Finally, we put our model into three months' length of the real-world test. Related work is described in section 2, the data we used is described in section 3, and we introduce our methodology and experiment in sections 4 and 5. The real-world test result and conclusion are stated in section 6.

2 Related Work

Volatility forecasting is always one of the hot topics in academic and industry research. In the first years, people first introduced time series methods. (Engle, 1982) introduced the autoregressive conditional heteroskedasticity model (ARCH) to model the residual series. To analysis the volatility process of the asset return, (Bollerslev, 1986) proposed the generalized autoregressive conditional heteroscedasticity (GARCH) model. Moreover, later on, the heteroscedasticity characteristics of financial assets and the variation characteristics of long memory were captured by the integral auto-regressive conditional heteroscedasticity model (FIGARCH)

(Baillie et al., 1996). However, these methods are more suitable for the volatility prediction of the low-frequency data.

With the need to analysis the high-frequency data and the development of the machine learning techniques, the application of Neural Network and LightGBM algorithm start to show in the financial market. In 1988, (White, 1988) first use the Neural Network on the prediction of the stock returns. Since then, deep learning technology has been rapidly and widely used in the financial field. In recent years, (Thavaneswaran et al., 2019) first introduced the Feedforward Neural Network (FFNN) predictive model for conditional variance and demonstrate the superiority of the FFNN for option pricing over other volatility models like GARCH and Heston-Nandi GARCH. Furthermore in (Zhao and Khushi, 2020), they proposed a Wavelet Denoised-ResNet with LightGBM model to predict the rate of change of foreign exchange price after five time intervals. And for the authors of (Sun et al., 2020), they adopted a novel Gradient Boosting Decision Tree (GBDT) algorithm, LightGBM, to forecast the price trend (falling, or not falling) of cryptocurrency market. In year 2021, Lei et al. used the temporal convolutional networks (TCN) model and combined with web search index data to predict the stock volatility and got a robust result that performs over the traditional times series model and LSTM model. In (Yang et al., 2021), authours make use of XGBoost model and LightGBM model to realize the prediction of stock price.

Our work mainly focused on the following points.

- We performed three types of feature engineering based on fundamental data, similar stock data, and similar time data. Then select the features based on feature importance.
- We put forward a hybrid volatility prediction model based on FFNN and LightGBM.
- We used root mean square percentage error (RMSPE) as the evaluation criterion and did compare experiments on our mixed model and other machine learning models.

3 Data

Our dataset is provided by Kaggle. This dataset contains stock market data relevant to the practical execution of trades in the financial markets. In particular, it includes order book data and executed trades data. With one second resolution, it provides a uniquely fine-grained look at the microstructure of modern financial markets.

The order book data is a parquet file partitioned by stock id. Provides order book data on the most competitive buy and sell orders entered into the market. The top two levels of the book are shared. The first level of the book will be more competitive in price terms. It will then receive execution priority over the second level. The features contained in the order book data is shown in the Table 1.

stock id	ID code for the stock.
time id	ID code for the time bucket.
seconds in bucket	Number of seconds from the start of the bucket, starting from 0.
bid price[1/2]	Normalized prices of the most/second most competitive buy level.
ask price[1/2]	Normalized prices of the most/second most competitive sell level.
bid size[1/2]	The number of shares on the most/second most competitive buy level.
ask size[1/2]	The number of shares on the most/second most competitive sell level.

Table 1: Order Book Data Variables

Trade data is also a parquet file partitioned by stock id. It contains data on trades that actually executed. Usually, in the market, there are more passive buy/sell intention updates (book updates) than actual trades, therefore this data is more sparse than the order book data. The features contained in the trade data is shown in the Table 2.

stock id	ID code for the stock.
time id	ID code for the time bucket.
seconds in bucket	Number of seconds from the start of the bucket, starting from 0.
price	The average price of executed transactions happening in one second. Prices have been normalized and the average has been weighted by the number of shares traded in each transaction.
size	The sum number of shares traded.
order count	The number of unique trade orders taking place.

Table 2: Trade Data Variables

We can plot the raw data to get an overview and check the relationship between the actual trade price and the bid and ask prices. From Figure 1, we can see that the actual trade price has a similar overall trend with the band formed by the bid and ask prices. The fluctuation of the actual trade price is also smaller than that of the bid and ask prices. This consistent with the fact that the data in trade data is more sparse. Furthermore, most of the time, the actual trade price lies in the band formed by the most competitive pair of bid and ask prices.

4 Methodology

In this section, we will introduce the methods we used in the feature engineering, model building and related experiment in detail.

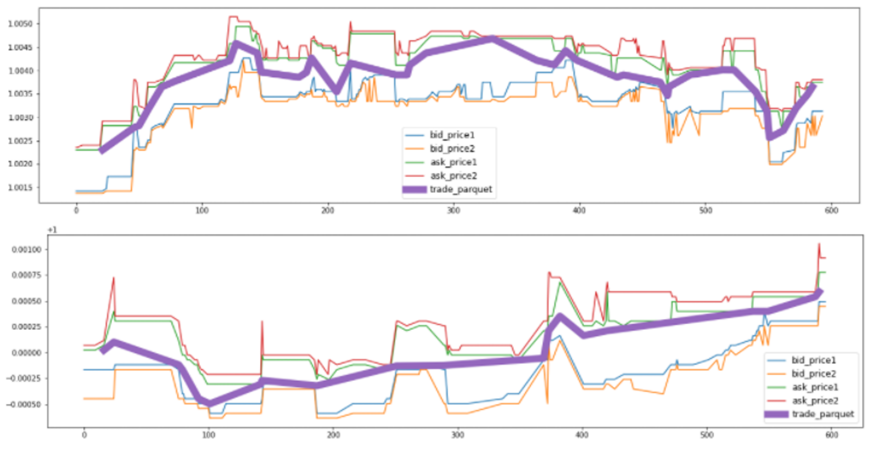

Figure 1: The Overview of Raw Data

4.1 Feature Engineering

As the quality of the features will have a huge influence on the final result of the model. We did three categories of feature engineering based on the different nature of our data. They are the basic features, time correlated features and the stock correlated features.

4.1.1 Basic Features

Basic features are built directly through basic mathematical operations based on the variables contained in the order book and transaction data. And they represents the most valuable information and can be the base of other features. They mainly focus on the following perspectives: Bid/Ask Spread, Log-return of the stock, Weighted average price, and the real volatility. They are defined as follows.

- Bid/Ask Spread:
  BidAskSpread = BestOffer / BestBid - 1

- Weighted Average Price:
  $\text{WAP} = \frac{\text{BidPrice}_1 * \text{AskSize}_1 + \text{AskPrice}_1 * \text{BidSize}_1}{\text{BidSize}_1 + \text{AskSize}_1}$

- Log-return:
  $r_{t_1,t_2} = \log\left(\frac{S_{t_2}}{S_{t_1}}\right)$, where t_1 and t_2 denotes the time point we use to calculate the log-return r, and S_t denotes the stock price at time t.

- Real Volatility: $\sigma = \sqrt{\sum_t r^2_{t-1,t}}$

Besides the features that we built based on the above perspectives. Our basic features also included order counts, size, amount, and volume-related features.

4.1.2 Time Correlated Features

We chose to build time-correlated features because stocks volatility should be similar at a similar time because of the momentum. The features (similar stock price, similar volatility, similar trade volume, etc) of the most adjacent time points (1, 3, 5, 10, 60) are arithmetically averaged to obtain the features of similar stocks.

4.1.3 Stock Correlated Features

To build stock correlated features, we first use the K-means algorithm to simulate several stock plates. We clustered 113 stocks into seven categories, representing seven stock plates. This is because of the plate linkage effect of the stocks, which means that for the stocks in the same plate, when some stocks' prices start to rise or drop first, they will pull up or drag down the prices of other stocks. The cluster result is shown in Figure 2. Next, the characteristics of the correlated stocks are arithmetically averaged to obtain the stock correlated features.

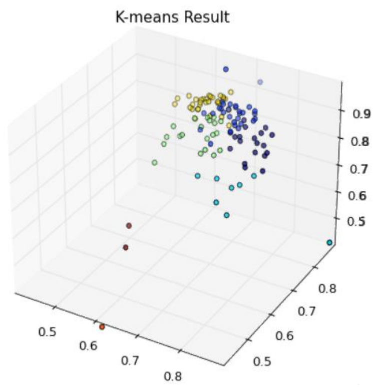

Figure 2: Stock Correlation K-means Result

Furthermore, after we got all the features, we can check the features' importance and choose the features based on that. Here in Figure 3, we show the top 20 important features. The final feature number we choose to use in our hybrid model is 51.

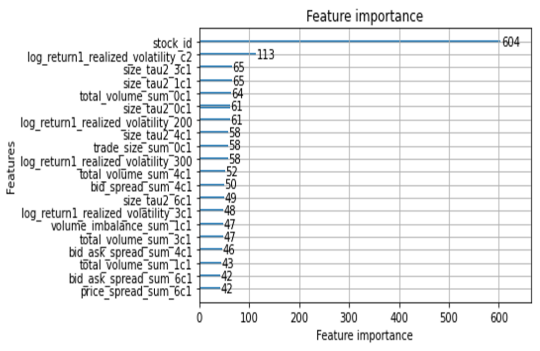

Figure 3: Feature Importance Selection

4.2 Hybrid Model

4.2.1 Feedforward Neural Network

Feedforward neural network is simple in structure and widely used, and can approximate any continuous function and square-integrable function with arbitrary precision. Furthermore, it can accurately implement any finite set of training samples. From a system point of view, a feedforward network is a static nonlinear mapping. Complex nonlinear processing capabilities can be obtained through composite mapping of simple nonlinear processing units. From a computational point of view. Lack of rich kinetic behavior. Most feedforward networks are learning networks, and their classification and pattern recognition capabilities are generally stronger than feedback networks.

A feedforward neural network has an input layer, one or more hidden layers in the middle, and an output layer. The input and output transformation relationship in the feedforward neural network is:

$$s_i^{(q)} = \sum_{j=0}^{n_{q-1}} w_{ij}^{(q)} x_j^{(q-1)}, \left(x_0^{(q-1)} = \theta_i^{(q)}, w_{i0}^{(q-1)} = -1\right)$$

$$x_i^{(q)} = f\left(s_i^{(q)}\right) = \begin{cases} 1, s_i^{(q)} \geq 0 \\ -1, s_i^{(q)} < 0 \end{cases}$$

$$i = 1, 2, \ldots, n_q; j = 1, 2, \ldots, n_{q-1}; q = 1, 2, \ldots, Q$$

Here, $x_i^{(q)}$ is the output of the layer q, w_{ij} is the connection weight for x_j to x_i, q is the number of the layers, and n_q is the dimension for layer q.

The structure of our FFNN is shown in the following Figure 4.

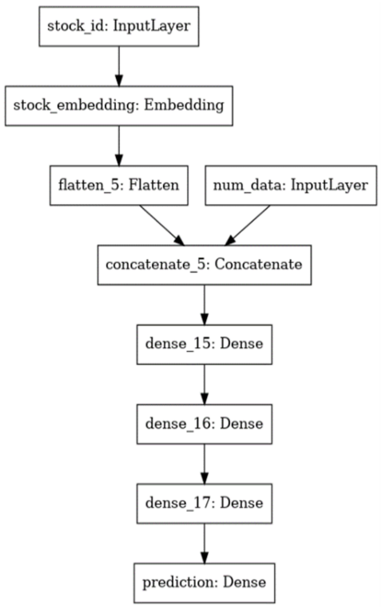

Figure 4: The Structure of FFNN

4.2.2 LightGBM

LightGBM is a framework to implement the GBDT algorithm, combines the GOSS and EFB algorithms, and is used in data sets with large sample data and high dimensions.

Its optimization features include Leaf-Wise-based decision tree growth strategy, optimal segmentation of category eigenvalues, feature parallelism, and data parallelism. It reduces the communication overhead and time complexity between data and ensures the rationality of data.

The main advantages of the LightGBM model include its relatively fast training speed, low memory consumption, high accuracy, distributed support, and the ability to process large amounts of data quickly.

Furthermore, in addition to the above features of the LightGBM algorithm, we also adopted a 5-fold cross-validation method in the training process to improve the model's predictive ability for out-of-sample data.

4.3 Ensemble Step

We combine LightGBM with FFNN as a hybrid model by taking the average of the prediction results from the LightGBM model and the FFNN model as the final hybrid model prediction result. This method is one of the most common ways to construct a hybrid prediction model, and it can take advantage of both of the original models and make results more stabilized. The model structure is shown in Figure 5.

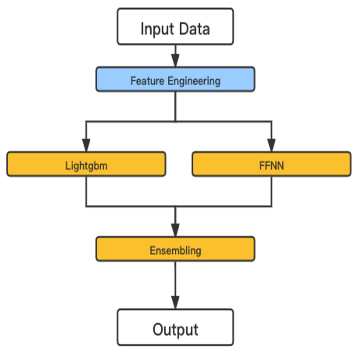

Figure 5: The Structure of Hybrid Model

5 Experiment and Results

5.1 Experiment Settings

The evaluation criterion we used is root mean square percentage error, which can be defined as following:

$$\text{RMSPE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\left(y_i - \hat{y}_i\right)/y_i\right)^2}$$

where y_i is the real value, $\hat{y}_i$ is the estimated value and n is the sample number. From the definition of RMSPE, we know that the lower the RMSPE is, the better performance the model will gain. And it is a good evaluation in our analysis since it is less sensitive to the outliers and can provide us a relatively stable evaluation result for the long term performance.

And for the prediction target, it is the realized volatility computed over the 10 minute window following the feature data under the same stock id or time id. And there is no overlap between feature and target data.

Tables 3 and 4 show our experimental settings for the FFNN and the LightGBM. We built and trained our FFNN model using TensorFlow 2.5.

epochs	1000
stock embedding size	24
learning rate	0.006
optimizer	Swish
hidden units	(192, 96, 48)

Table 3: The parameter of FFNN

learning rate	0.05
boosting type	gbdt
subsample	0.72
Lambda L1	0.5
Lambda L2	1.0
feature fraction	0.5
max bin	100

Table 4: The parameter of LightGBM

Here for the Swish activation function used as optimizer in our FFNN model, it is defined as

$$\text{Swish}(x, \beta) = x \times \text{sigmoid}(\beta \times x)$$

And for the parameters in the LightGBM model, we got this set of parameters from the best result of the 5-fold cross validation.

5.2 Results

The experimental results of competing models and our model are shown in Table 5.

Models	RMSPE
Naïve Bayes	0.362
SVM	0.269
lightgbm	0.219
FFNN+Lightgbm	**0.192**

Table 5: Performance of Different Model

From the definition of RMSPE, we know that the lower the RMSPE is, the better performance the model will gain. From the compared experiments, our hybrid model owns the lowest RMSPE score of 0.192. In contrast, the traditional models like Naïve Bayes, SVM, LightGBM's RMSPE are 0.362, 0.269, 0.219 respectively. This result proved the advantage of our hybrid model.

In the out-of-sample forecasting tests in real market data for three months provided by the Kaggle platform, the RMSPE score of our hybrid model fluctuated in the range 0.199 to 0.219 and has a final score of 0.218 at the end of the three months testing period. This result proves that our hybrid model still has good performance in out-of-sample testing, and this performance has high stability.

6 Conclusion

In order to improve the existing prediction system for realized volatility, we first did three categories of feature engineering and then built a hybrid model by combining Feedforward Neural Network (FFNN) with Light Gradient Boosting Machine (LightGBM).

In the model comparison experiment result, our hybrid model owns the lowest RMSPE score of 0.192 among all compared models. Furthermore, in the three-month real market data test from September 20th, 2021 to January 10th, 2022, our hybrid model's RMSPE result remained in the range $[0.199, 0.219]$ with a final RMSPE result of 0.218. These results show our hybrid model's advantage in both accuracy and stability.

In future research, we think there are several potential ways to improve the result of our hybrid prediction model. They can be mainly classified into two categories:

1. For the feature engineering: we can try to add more features, either based on the intuition about the data or using other feature selection method. And delete the over-fitting features after that.

2. Because we are limited by running time and memory when running a single code, we need to choose model components that are more efficient to train and run with fewer memory requirements. Some other options for the hybrid model's components might increase the model performance if we remove the above constraints. For example, we can experiment on the CatBoost and the Tabnet on the LightGBM part and experiment Fastai NN, Conv1D, RNN attention, and transformer on the FFNN part. A hybrid model composed of different combinations of these models may bring about just the right prediction effect.

Acknowledgements

Thanks to the Kaggle platform, we could advance our research in the long period and produce this paper documenting the work. And due to Dr. Chenhui Xiang's encouragement, the work could finish efficiently.

References

Richard T. Baillie, Tim Bollerslev, and Hans Ole Mikkelsen. 1996. Fractionally integrated generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics*, 74.

Tim Bollerslev. 1986. Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics*, 31.

Robert F Engle. 1982. Autoregressive conditional heteroscedasticity with estimates of the variance of united kingdom inflation.

Bolin Lei, Boyu Zhang, and Yuping Song. 2021. Volatility forecasting for high-frequency financial data based on web search index and deep learning model. *Mathematics*, 9:1–17.

Xiaolei Sun, Mingxi Liu, and Zeqian Sima. 2020. A novel cryptocurrency price trend forecasting model based on lightgbm. *Finance Research Letters*, 32:101084.

A. Thavaneswaran, Ruppa K. Thulasiram, Julieta Frank, Zimo Zhu, and Manmohit Singh. 2019. Fuzzy option pricing using a novel data-driven feed forward neural network volatility model. In *2019 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 1–6.

Halbert White. 1988. Economic prediction using neural networks: The case of ibm daily stock returns. *Neural Networks on IEEE*, pages 451–458.

Yue Yang, Yang Wu, Peikun Wang, and Xu Jiali. 2021. Stock price prediction based on xgboost and lightgbm. volume 275. EDP Sciences.

Yiqi Zhao and Matloob Khushi. 2020. Wavelet denoised-resnet cnn and lightgbm method to predict forex rate of change. *ICDW 2020*.