

# Ensembling two deep learning algorithms to efficiently solve the problem of predicting volatility in applied finance

Petr Pylov<sup>1\*</sup>, Anna Dyagileva<sup>1</sup>, and Andrey Protodyakonov<sup>1</sup>

<sup>1</sup>T.F. Gorbachev Kuzbass State Technical University, 650000, Kemerovo, 28 Vesennya st., Russian Federation

**Abstract.** Volatility is one of the most commonly used terms in the trading platform. In financial markets, volatility reflects the magnitude of price fluctuations. High volatility is associated with periods of market turbulence and sharp price fluctuations, while low volatility characterizes more relaxed pricing. When trading options, it is especially important for trading firms to accurately predict volatility values, since the price of options is directly related to the profit of a trading firm. A proactive artificial intelligence model that allows predicting volatility for future periods of time will be presented in this article.

## 1 Introduction

All financial companies are interested in obtaining an accurate forecast of volatility for the nearest future time interval. Thanks to such information (a justified forecast of short-term volatility), any company gets the opportunity to make a unique offer to the market (based on knowledge of the predicted share price, the company can lower the cost of buying shares) and differentiate its company from competitors [1].

One of the global trading companies is Optiver, whose specialists have created an open data set containing thousands of lines of detailed financial information, which will be enough to form the generalization ability of an artificial intelligence model that predicts financial market volatility [2].

To solve any applied forecasting problem, it is necessary to choose the appropriate machine learning algorithm correctly. In this regard, the task of forecasting volatility is no exception to the rule. The subdomain of problems related to forecasting trends, finance and time series is successfully solved by deep learning algorithms (deep learning) [3]. The problem under consideration ideally fits the criteria for using deep learning algorithms, since financial markets are inextricably linked with the forecast of numbers and time series.

Python was chosen as the initial programming language. Based on it, it is very easy to operate with the main methods and basic models of a wide range of deep learning algorithms. This factor allows you to significantly optimize the listing of program code and increase the convenience of writing it [4].

---

\* Corresponding author: [pylovpa@kuzstu.ru](mailto:pylovpa@kuzstu.ru)

The problem formulated in this article can be effectively solved on the basis of artificial intelligence algorithms in two ways:

1. To find all relationships in terms of data, determine the functions of their relationship, evaluate the correlation of unrelated data. To transfer all the features of the subject area to the artificial intelligence model, transforming, if necessary, the architecture of the algorithm itself.
2. To leave the search for relationships in the data to the artificial intelligence algorithm

Let's start in order of points: to determine all the relationships in the data, it is necessary not only to have an excellent knowledge of the entire subject area, but also to understand how individual parameters are related (or not related) to each other. However, such a situation is impossible due to the fact that even specialists - subject matter researchers - are not able to create and, a fortiori, transfer a fully connected mathematical model into a programmatic form of an artificial intelligence model [5].

In this case, the search for relationships in the data should be left to the artificial intelligence algorithm.

## **2 Results and discussion**

The problem can be efficiently solved using a Fuzzy Neural Network – FNN. This type of artificial neural networks has an important specific feature - FNN can predict the result based on internal, non-obvious analytics of the source data parameters [3]. This feature has an invaluable advantage: if analysts do not fully understand the principle of how the source data is arranged and interrelated, then they can be analyzed using an algorithm. The algorithm determines data relationships and constructs a model based on them [4]. It is not possible to study the model itself and how the relationships in it are arranged [3], that's why the use of FNN is prohibited in tasks related to ensuring the life, health and safety of people [6]. But, since the subject area of the problem considered in the article is not related to ensuring the safety of life and health of people, the use of the FNN algorithm is permissible.

Taking into account the special orientation of the fuzzy neural network to independent data analytics, we note that FNN is, first of all, an artificial intelligence model that, in the learning process, also “makes” assumptions about the best way to form each stage of the decision algorithm architecture [3]. The process of forming a single stage in most cases is random, since a large analytical deepening into the data by the algorithm will require huge computing power and serious time resources [4].

It is not necessary to complicate the FNN with internal mathematical calculations so that the process of forming a stage is not random, but has a reasonable analytical character. A rational solution would be to introduce an auxiliary LGBM (Light Gradient Boosting Machine) algorithm into the FNN algorithm. This approach will increase the capabilities of FNN, while not requiring large time and computational resources [2]. The LGBM auxiliary algorithm allows each new stage of the architecture to be selected from those represented by the FNN algorithm based on the minimum loss function value and the best model accuracy value, rather than on the principle of assumption.

Figure 1 shows a fragment of the application of the algorithm in programmatic form in the Python programming language.

```

def model_preprocessing_of_lgbm(train, test, params, boost=10000):
    features = [col for col in train.columns if col not in {"time_id", "target", "row_id"}]
    y = train['target']
    oof_predictions = np.zeros(train.shape[0])
    test_predictions = np.zeros(test.shape[0])
    kfold = KFold(n_splits = 5, random_state = 21, shuffle = True)
    for fold, (trn_ind, val_ind) in enumerate(kfold.split(train)):
        print(f'Training fold {fold + 1}')
        x_train, x_val = train.iloc[trn_ind], train.iloc[val_ind]
        y_train, y_val = y.iloc[trn_ind], y.iloc[val_ind]
        train_weights = 1 / np.square(y_train)
        val_weights = 1 / np.square(y_val)
        train_dataset = lgb.Dataset(x_train[features], y_train, weight = train_weights)
        val_dataset = lgb.Dataset(x_val[features], y_val, weight = val_weights)
        model = lgb.train(params = params,
                          num_boost_round=boost,
                          train_set = train_dataset,
                          valid_sets = [train_dataset, val_dataset],
                          verbose_eval = 250,
                          early_stopping_rounds=50,
                          feval = feval_rmspe)
        oof_predictions[val_ind] = model.predict(x_val[features])
        test_predictions += model.predict(test[features]) / 5
    rmspe_score = rmspe(y, oof_predictions)
    lgb.plot_importance(model, max_num_features=20)
    return test_predictions

```

**Fig. 1.** LGBM auxiliary algorithm module.

After the algorithmization of the auxiliary module for the fuzzy neural network, it is necessary to proceed to the application of the FNN algorithm (Figure 2).

```

from keras.backend import sigmoid
def swish(x, beta = 1):
    return (x * sigmoid(beta * x))

from keras.utils.generic_utils import get_custom_objects
from keras.layers import Activation
get_custom_objects().update({'swish': Activation(swish)})

hidden_units = (512,64,32)
stock_embedding_size = 24
cat_data = train_nn['stock_id']

def base_model_FNN():
    stock_id_input = keras.Input(shape=(1,), name='stock_id')
    num_input = keras.Input(shape=(244,), name='num_data')
    stock_embedded = keras.layers.Embedding(max(cat_data)+1, stock_embedding_size,
                                             input_length=1, name='stock_embedding')(stock_id_input)
    stock_flattened = keras.layers.Flatten()(stock_embedded)
    out = keras.layers.Concatenate()([model_preprocessing_of_lgbm, num_input])
    for n_hidden in hidden_units:
        out = keras.layers.Dense(n_hidden, activation='swish')(out)
    out = keras.layers.Dense(1, activation='linear', name='prediction')(out)
    model = keras.Model(
        inputs = [stock_id_input, num_input],
        outputs = out, )
    return model

```

**Fig. 2.** Fuzzy neural network module

After the implementation of the algorithm, it is necessary to determine what metrics will be used to evaluate its accuracy [4]. The key factors are the validity of the metric and its reliability. In the case of the FNN algorithm, these two factors are satisfied by the "inverse" validation loss metric [3]. The difference between the unity and this metric allows the most complete assessment of the accuracy of the artificial intelligence model.

The results of the model accuracy will be evaluated by the metric of validation losses (Figure 3), and cross-validation will act as the best epoch of model training (at which the highest generalization ability is achieved) [3].

Epoch 24/1000	168/168 [=====] - 3s 15ms/step - loss: 0.2160 - val_loss: 0.2213
Epoch 25/1000	168/168 [=====] - 3s 17ms/step - loss: 0.2175 - val_loss: 0.2268
Epoch 26/1000	168/168 [=====] - 3s 15ms/step - loss: 0.2172 - val_loss: 0.2233
Epoch 27/1000	168/168 [=====] - 3s 15ms/step - loss: 0.2181 - val_loss: 0.2310
Epoch 28/1000	168/168 [=====] - 3s 18ms/step - loss: 0.2173 - val_loss: 0.2201
Epoch 29/1000	168/168 [=====] - 3s 18ms/step - loss: 0.2155 - val_loss: 0.2221
Epoch 30/1000	168/168 [=====] - 3s 16ms/step - loss: 0.2146 - val_loss: 0.2227
Epoch 31/1000	168/168 [=====] - 3s 16ms/step - loss: 0.2127 - val_loss: 0.2233
Epoch 32/1000	168/168 [=====] - 3s 15ms/step - loss: 0.2151 - val_loss: 0.2193
Epoch 33/1000	168/168 [=====] - 2s 14ms/step - loss: 0.2128 - val_loss: 0.2395
Epoch 34/1000	168/168 [=====] - 2s 14ms/step - loss: 0.2213 - val_loss: 0.2242
Epoch 35/1000	168/168 [=====] - 3s 15ms/step - loss: 0.2242 - val_loss: 0.2180
Epoch 36/1000	168/168 [=====] - 3s 16ms/step - loss: 0.2119 - val_loss: 0.2575
Epoch 37/1000	168/168 [=====] - 3s 17ms/step - loss: 0.2216 - val_loss: 0.2241
Epoch 38/1000	168/168 [=====] - 3s 15ms/step - loss: 0.2121 - val_loss: 0.2301
Epoch 39/1000	168/168 [=====] - 3s 15ms/step - loss: 0.2131 - val_loss: 0.2410
Epoch 40/1000	168/168 [=====] - 3s 16ms/step - loss: 0.2158 - val_loss: 0.2295
Fold 5 NN: 0.21466	

**Fig. 3.** Fuzzy neural network module

The green block in Figure 3, outlines the values of the validation loss metric at each iteration of the training epoch of the algorithm. Epoch iteration is an attempt by the model to improve the internal structure of the algorithm on the training data during the training process [4].

The blue block in Figure 3 shows the state of the optimal algorithm configuration, characterized by the final value of the validation loss metric [3]. Thus, the most valid value of the algorithm model accuracy, which can be estimated by the validation loss metric, is

the value equal to the difference between unity and 0.21466. The numerical value of this difference is 0.78534, which is a good indicator of the accuracy of trend forecasting algorithms [6].

The use of FNN algorithm ensembling with additional auxiliary LGBM module made it possible to obtain a really high accuracy of the model (Figure 3). But without a comparative comparison of the two types of algorithms (FNN with and without LGBM), it would be unfounded to argue that the use of LGBM is so necessary. Figure 4 shows the results of the accuracy of the FNN model, which were obtained without using the auxiliary LGBM algorithm.



**Fig. 4.** Accuracy results of the “pure” FNN model

### 3 Conclusion

From the received accuracy reports of machine learning algorithms (Figures 3-4), we can conclude that the difference in accuracy is significant and amounts to almost 0.2 (0.78534 vs. 0.59088). This gives grounds to argue that the ensemble of several types of algorithms

really allows getting a gain in the overall performance of the applied artificial intelligence model for predicting volatility.

However, it should be noted that not in all cases this approach will allow to obtain a similar performance gain in the machine learning model, since in order to acquire the best approximation ability, it is often enough to use the basic design of the model, and additional add-ons can only worsen the initial accuracy indicators [2, 4].

## References

1. K. Bugg, *Library Leadership and Management* **29**, 4 (2015)
2. R. Wiblin, *Positively shaping the development of artificial intelligence* (2017)
3. I.V. Ponkin, A.I. Redkina, *Artificial intelligence from the point of view of law* (2018)
4. K. Tuyls, S. Omidshafiei, *Journal of Artificial Intelligence Research* **71**, 41-88 (2021)
5. G.I. Kolesnikova, *Artificial Intelligence: Problems and Prospects* (2018)
6. M. Tegmark, *Benefits & Risks of artificial intelligence* Future of life institute