Group Project Report

Stock analysis based on prediction model

Subject: Data Mining

Course Teacher: Dr. Zongwei LUO

Group 14

Gu Yiting 1930026037

Wang Qiaoyi 1930026118

Lin Xiaotao 1930026081

Content

1. Abstract

In this project, we focus on the stock data of CMBC (China Minsheng Banking Corp. Ltd.) from the Shanghai Stock Exchange with the stock code “600016”. CMBC is the first national commercial bank established by private capital in mainland China. In recent years, it has repeatedly appeared in the list of China's top 500 companies, so, we choose it as the subject of our analysis. We imply 6 models, there are respectively Knn, AIRMA, Deep Forest, Random Forest, XGBoost and LightGB, on this project to analyze this stock data and analyze the performance of the model, which can help us to know which model is more suitable for this stock data and where can be improved.

2. Data collection

In terms of data collection, we used a python library called baostock to get our data. Baostock itself is an open-source securities data service platform. Considering the advantages of Python pandas package in financial quantitative analysis, most of the data formats returned by baostock are pandas dataframe type, which is very convenient for data analysis and visualization with pandas, numpy and Matplotlib. Baostock offers an interface to get all kinds of historical information of stocks in Shanghai stock exchange. It not only provides basic level data like opening price, closing price, volume and amount of exchange, but also high level data like dynamic price-earnings ratio, price-to-book ratio, price-to-sales ratio and price cash flow ratio.

In our project, in order to simplify the process of analyzing, we only used basic level data. We used the data of China Minsheng Bank (Code: 600016) with a date range from 2020/7/22 to 2021/7/21.

3. Data modeling

3.1 Arima algorithm

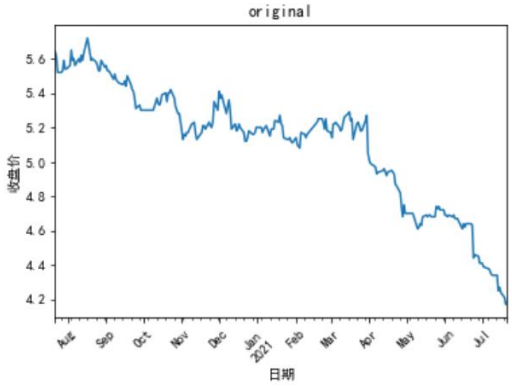
3.1.1 Introduction:

The full name is called Autoregressive Integrated Moving Average Model. The basic idea of the ARIMA model is to treat the data sequence formed by the prediction object as a random sequence, and use a certain mathematical model to approximate this sequence. Once the model is identified, it can predict future values from the past and present values of the time series.

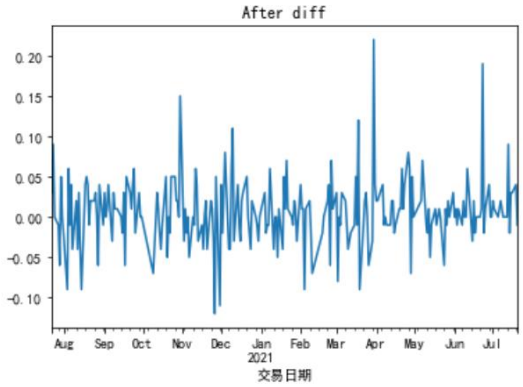
3.1.2 Implement:

ARIMA (p, d, q) is called differential autoregressive moving average model, AR is autoregressive, p is auto-regressive term; MA is moving average, q is the number of moving average terms, d is what is done when the time series becomes stationary Number of differences. Thus, if we want to implement this model, it is necessary for us to figure out the best fitted p, d, and q.

And as for d, it represents the time series data that needs to be differentiated in several steps to be stable, also called the integrated term. We used ADF test to see the stationarity of the original time series. When d=0, which means the original data, it is not stable after we draw the graph to observe it.

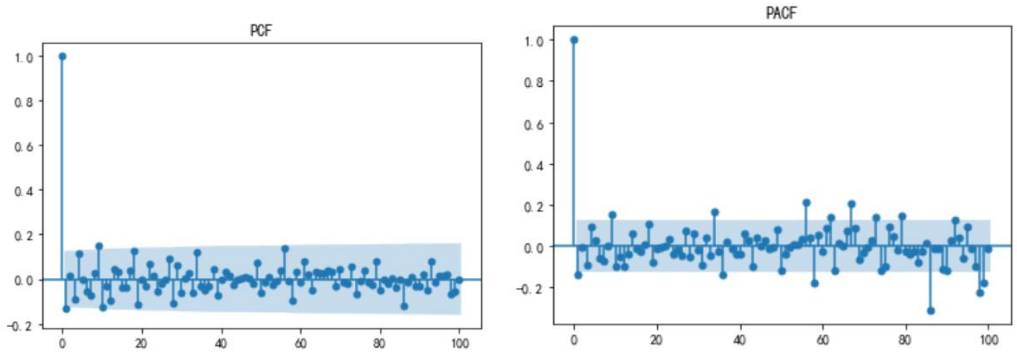


And it can be seen that basically the time series is already close to a stationary series in the first difference. Thus, we set the d into 1.



And we also need to pay attention to the linear correlation between time series observations and their past observations. Here we use the PCF and PACF to verify that.

And the corresponding PCF graph and PACF graph after we make the first difference is:



And we can get the p and q by observing the Tailing and censoring, which is roughly p=3 and q=3.

As for p and q, p represents the number of lags of the time series data itself used in the prediction model, also known as the AR/Auto-Regressive term, and q represents the number

of lags of the prediction error used in the prediction model, also known as the MA/Moving Average term. And there are totally two methods to find the suitable. From the above function, we can know the rough range of the p and q, but if we want to get the accurate value, we need to use Information Criterion Function Method, which is AIC rules and BIC rules.

The full name of the AIC criterion is the Akaike Information Criterion (Akaike Information Criterion), the calculation formula is as follows:

AIC = 2 * (number of model parameters) – 2 * ln (maximum likelihood value of the model)

The full name of the is criterion is Bayesian information criterion is the improved version of AIC:

BIC = ln(n) * (number of model parameters)-2*ln (maximum likelihood value of the model)

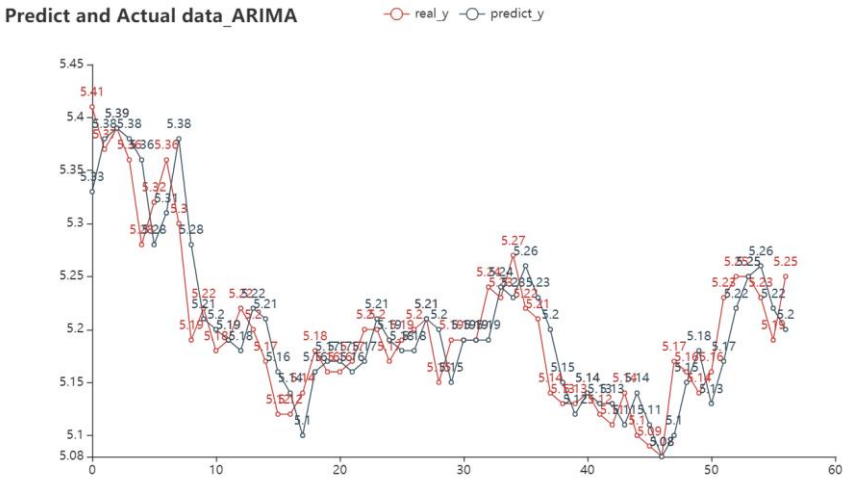
We find the best combination of p and q for our model in a similar way to grid search. Here we use AIC for experimentation.

```
#training
from statsmodels.tsa.arima_model import ARIMA
import statsmodels.api as sm


train_results = sm.tsa.arma_order_select_ic(D_ts, ic=['aic', 'bic'], trend='nc', max_ar=4, max_ma=4)

print('AIC', train_results.aic_min_order)
print('BIC', train_results.bic_min_order)
```

And we finally get the best pair is AIC(3,3)

```
AIC (3, 3)
BIC (1, 0)
```

After we confirm the parameters, we can now build the model by the package using python. And we also mark the comparison of the prediction and the actual value.



Although this model can make a relatively accurate result, it seems not perform that well because we are making predictions for the stock. It is possible that because the stock is irregular, the time series method may not be accurate in prediction

3.2 KNN

3.2.1 Introduction

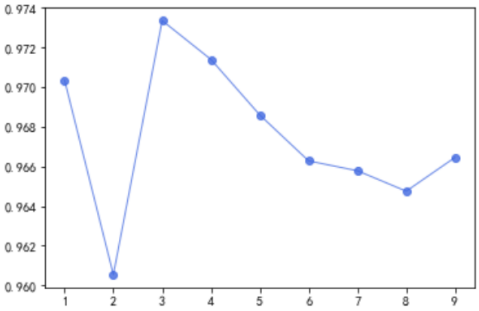
This is a method with very simple and intuitive idea. So, we implement it by hand. If most of the K most similar (the nearest neighbors in the feature space) samples of a sample in the feature space belong to a certain category, the sample also belongs to this category. In the classification decision, this method only determines the category of the sample to be classified based on the category of the nearest one or several samples

3.2.2 Implement

As for this model, there is two things we need to figure out, the second one is the method of how we judge the distance between each point. In this case, we select the Euclidean Distance to manage the distance between each point. The formula of the Euclidean Distance is shown as followed.

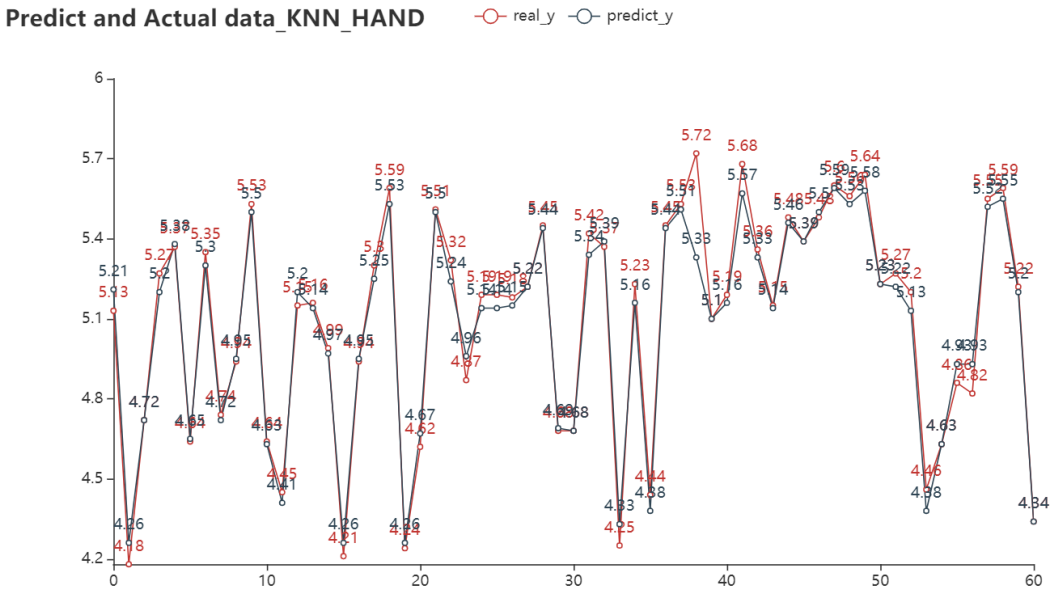
$$d(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

And the second one is the k, which means the number of neighborhoods. In this case, we use to select the best k according to the R square performance. If the model's R square has a bigger number, it has better performance. We draw the average R square according to the difference k. And we can see that when k=3, it has a better performance.



Then we need to consider how to make it more accurate because we are making prediction for the distance and this is not a constant, thus we cannot a treat the value as a label. Here we use the average of the five to measure the distance to make it more accurate.

After we confirm the k and the method how we measure the distance, we can make a prediction and the result is shown as followed.



This method can perform better than the previous one. Because the number of samples used for training is relatively small and vulnerable to sample imbalance, we need to continue to optimize the method with other better models.

3.3 BGDT

Gradient boosting decision tree is a useful model developed on the basis of boosting tree. When dealing with regression problems, boosting tree can be regarded as a special case of gradient boosting tree. Since in each step of building the tree, the boosting tree is to fit the

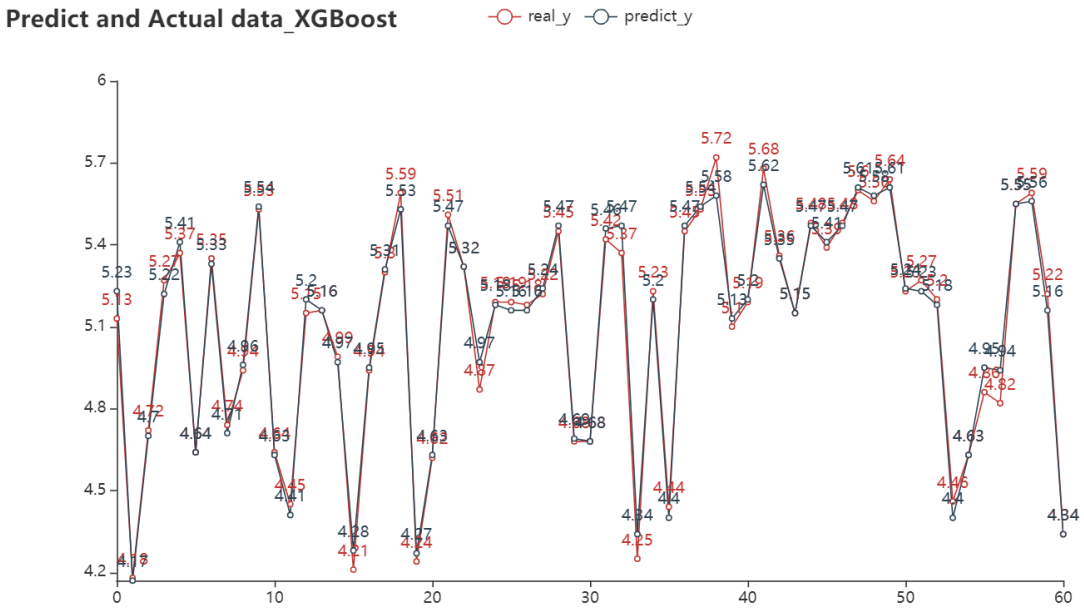
residual of the model in the training set obtained in the previous step.

The main idea of gradient boosting decision tree is doing the gradient descent in the function space where the target function is located, that is, consider the model as a parameter and get the gradient of the target function with respect to the model obtained in the previous procedure step by step. Therefore, the parameters are updated in the direction of minimizing the target function.

3.3.1 XGBoost

XGBoost is a special kind of GBDT. The biggest difference between XGBoost and GBDT is that XGBoost calculates the leaf node weight of the tree to be fitted in the next step by making a second-order Taylor expansion of the objective function. So as to say, the model calculates the loss of each split node according to the loss function, and select appropriate attributes for splitting according to the splitting loss.

After we implement the model, the comparison between predictions and the results are shown as followed:



3.3.1.1 Characteristic of XGBoost

- In addition to the reduction coefficient similar to GBDT, XGBoost penalizes the number

and weight of leaf nodes of each tree to avoid overfitting.

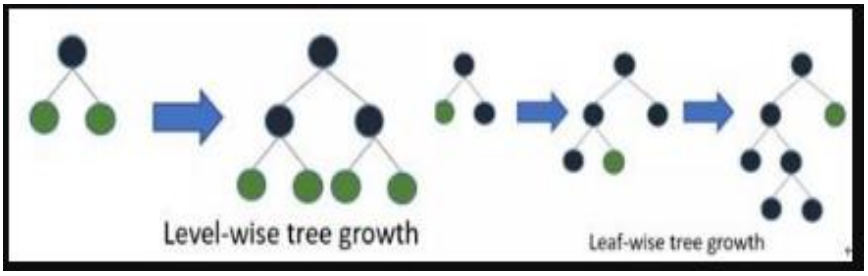
- Similar to random forest, XGBoost randomly selects some attributes for each tree as splitting attributes in the process of building the tree.
- XGBoost adds support for sparse data.
- XGBoost supports parallelization in implementation.
- When implementing XGBoost, you need to import all data into memory and do a presort (exact algorithm), which is faster when selecting split nodes.

3.3.1.2 Drawback of XGBoost

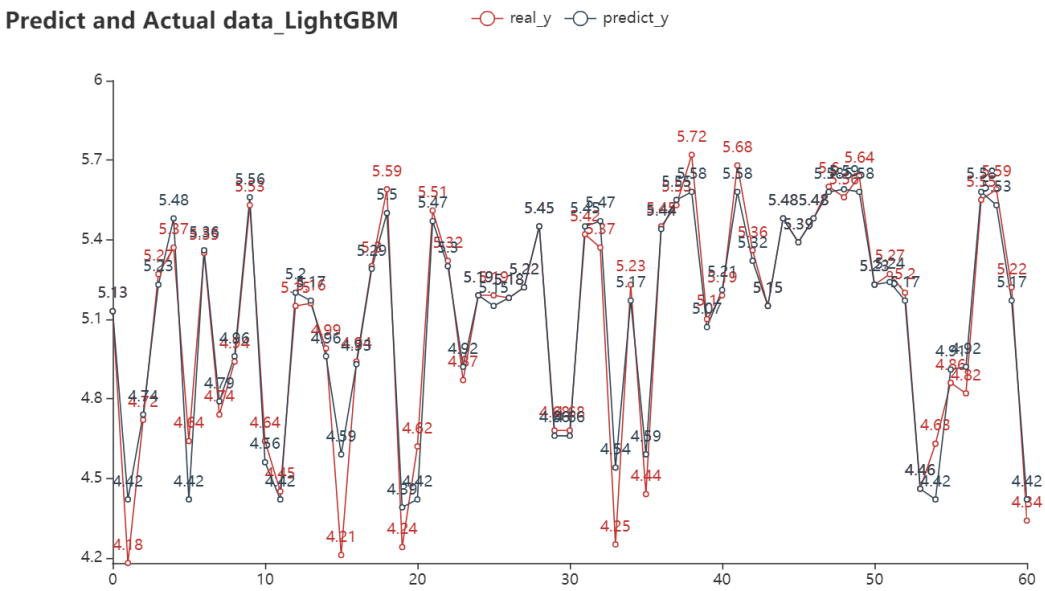
- The level-wise tree building method treats all leaf nodes of the current layer equally. Some leaf nodes have very small splitting income and have no impact on the results, but they still have to be split, which increases the calculation cost.
- The space consumption of the presorting method is relatively large. It is necessary to save not only the eigenvalues, but also the sorting index of the features. At the same time, it is also time-consuming. The splitting gain must be calculated when traversing each splitting point.

3.3.2 LightGBM

LightGBM (light gradient boosting machine) is a framework for implementing gbdt algorithm. It is considered as an improved model of XGBoost. It supports efficient parallel training, and has the advantages of faster training speed, lower memory consumption, better accuracy, distributed support, and better performance when data size is huge.



After we implement the model, the comparison between predictions and the results are shown as followed:



3.3.2.1 Some advantages of LightGBM

- Accelerate the speed:
  - LightGBM model implements a histogram algorithm, which view a histogram instead of view a table when reading the data.
  - Use leaf-wise method to create tree, which decreases the calculation cost.
- Save the memory
  - The histogram algorithm not only increase the speed, but also save the memory. It uses bin value to replace the specific value which was used in XGBoost.

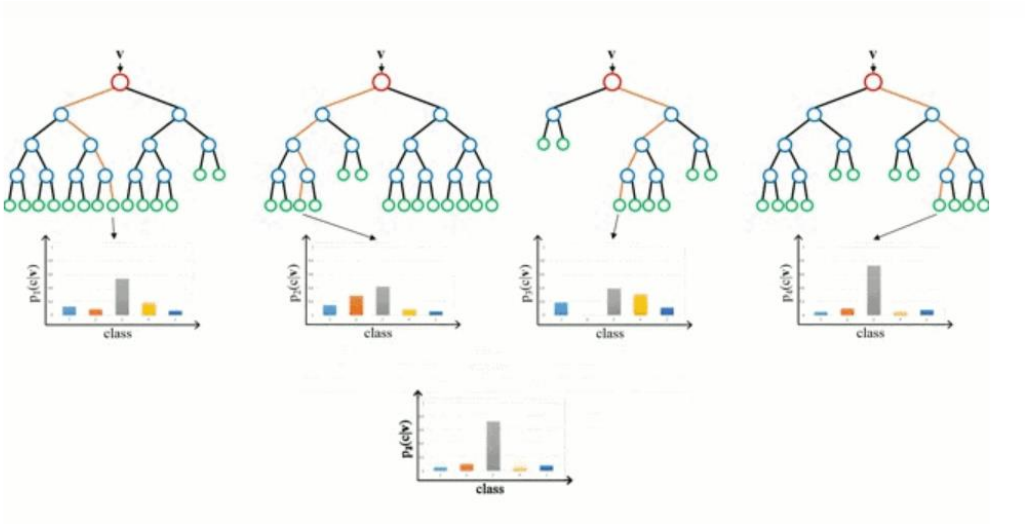
3.3.2.2 Drawbacks of LightGBM

- It may create a really deep decision tree which leads to overfitting. Therefore, LightGBM model adds a maximum depth limit on leaf-wise to ensure high efficiency and prevent overfitting.
- Since LightGBM is an algorithm based on bias, it is really sensitive to noise points.
- When doing optimization, the model relies on the optimal segmentation variable but ignores the idea that the optimal solution is the set of all characteristics.

3.4 Random Forest

Random forest is a classifier that uses multiple decision trees to train and predict samples. There are many decision trees in the forest, and there is no correlation between each decision tree in the random forest.

After obtaining the forest, when an input enters the model, each tree in the forest would make a classification and consider the category which was classified by the trees the most time as the category of the sample.



3.4.1 Process to conduct a random forest

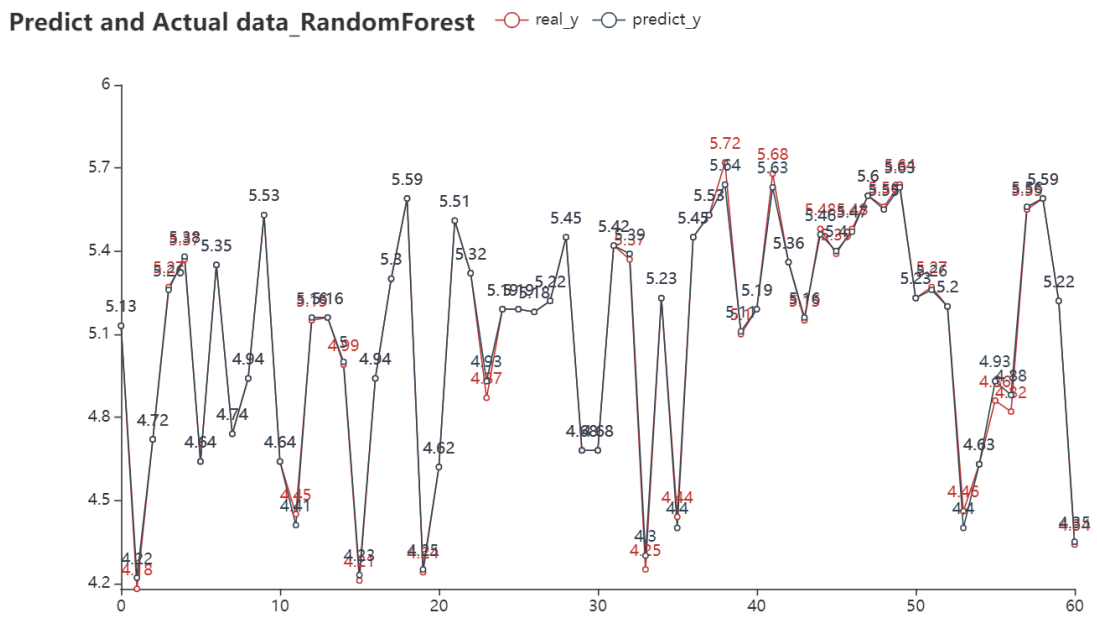
1. If the size of the training set is N, randomly select N training samples (with putting back) for each tree as the training set.

2. If each sample has M features, specify a constant m<<M. Randomly select m subsets from m features, and select the best one from these subsets each time the tree splits.

3. When creating trees, we use step 2 to split each node of the trees. Until it cannot be split anymore.

4. Repeat step 1 to step 3, create trees to form a forest.

The comparison between predictions and the results are shown as followed:



3.4.2 Advantages of random forest

- It is unexcelled in accuracy among current algorithms.
- It runs efficiently on large databases.
- It can handle thousands of input variables without variable deletion.
- It gives estimates of what variables are important in the classification.
- It generates an internal unbiased estimate of the generalization error as the forest building progresses.
- It has an effective method for estimating missing data and maintains accuracy when a large proportion of the data are missing.

3.5 Deep Forest:

3.5.1 Introduction:

Deep forest, also calls gcForest, is a model that stacks multi-layer of random forest in a cascading way to obtain better feature representation and learning performance. It only needs very little training data to achieve good performance, and basically does not need to adjust the settings of hyperparameters. For whole structure of deep forest, there are two main

parts of the deep forest. The following picture is the overall structure of deep forest.

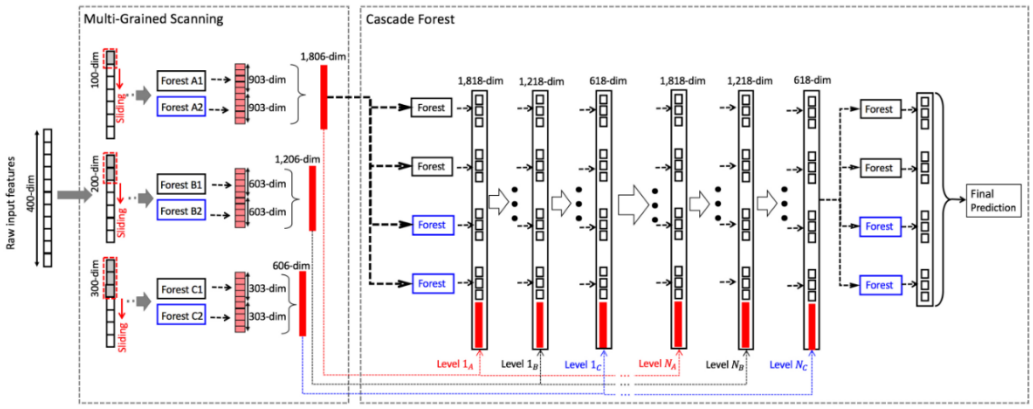


Figure 1. the overall structure of deep forest

With this structure, there are two steps to build this model.

1. Use Multi-Grained Scanning to preprocess the input features.
2. The obtained feature vectors are input into the cascade forest for training

First of all, it uses a Cascade Forest Structure. It looks like neural network, also has many layers. But the difference of it is that each level is a set of decision forests and the output of pre-level is the input of next level. Also, in order to increase diversity each level has to include different types of forests because diversity is essential to the overall structure. In this structure, each layer of it is assumed to consist of four tree models, two are random forests and the others are completely random forests. And we assume that there are three predictable categories. So, each random forest will output a three-dimensional class vector and then write it together to represent the original input of next level. For these two kinds of trees, we have following settings.

a. each completely random forest consists of 500 decision trees and number of characters in each tree is obtained by randomly selecting classification features at each node. And during this process, the tree grows completely, so each leaf of the tree contains only one type of sample.

b. Similarly, each random forest consists of 500 decision trees. Each tree is selected by randomly selecting $\sqrt{k}$ candidate features (k- the number of features). Then select

the largest feature of the Gini coefficient for division.

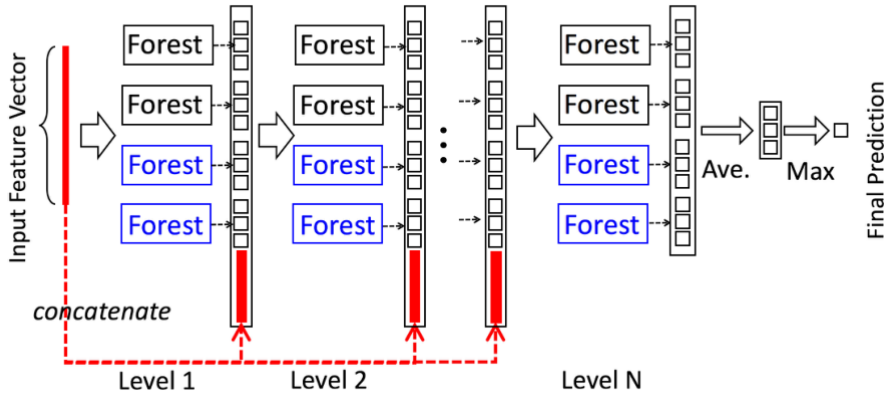


Figure 2. Cascade forest Structure

For example, each forest will calculate the percentage of the relevant samples falling into different categories on leaf nodes in each random tree, and then the results of all random trees in the forest are averaged to produce an estimate of the category distribution of the sample.

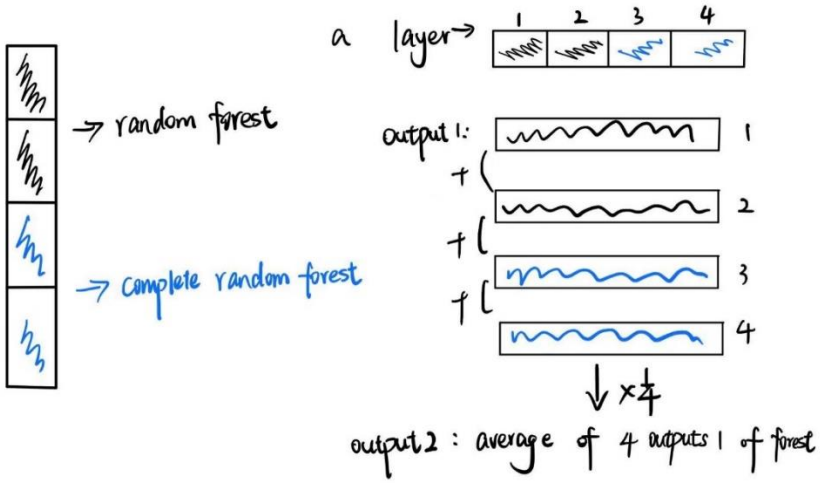


Figure 3. Example for cascade forest structure

Besides, to reduce the risk of overfitting, class vector produced by each forest is generated by k-fold cross validation. So, each sample will be used k-1 times as training data, resulting in K-1 class vectors, which will then be averaged to generate the final class vector as the enhanced feature for the next cascade. After a new layer is generated, the performance of the model is estimated on a test set, and if there is no significant improvement compared to the last time, the training process is stopped.

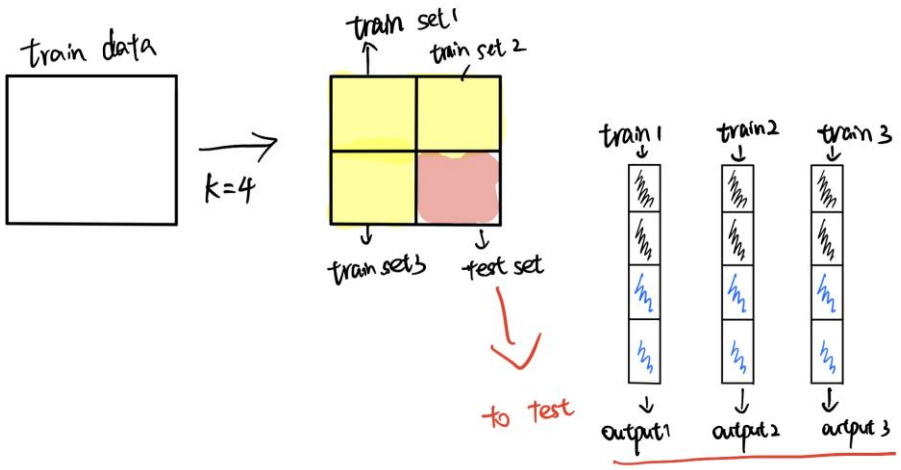


Figure 4. k-fold procedure

And before we use the Cascade Forest Structure, the data need to be processed by multi-Grained scanning. It is actually a sliding window to scan the data, which is usually used to deal with data such as images that are associated in spatial position. In the following picture, it uses three sizes of sliding windows as an example and there are respectively 100-dim, 200-dim and 300-dim. And we assure that the input data is 400-dim sequence data. For sequence data, a 100-dimensional feature vector will be generated by sliding a feature window. Since (400-100)/1 + 1) = 301, so, a total of 301 feature vectors will be generated. Then, these 301 feature vectors will be inputted in to a completely-random tree forest and random forest, since we assure there are 3 classifications, these two forests will get 301 3-dim vectors respectively and the feature vectors obtained from the two forests will be spliced together to get 1806-dim feature vectors. Since our data is Time series data, so, we don't use it in our project.

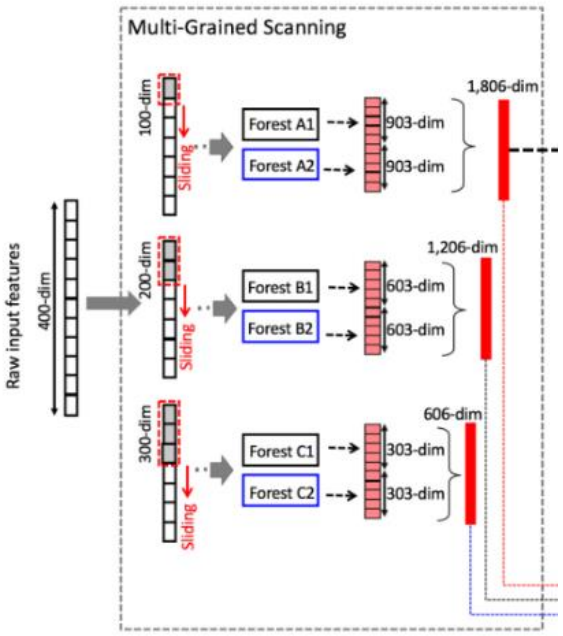
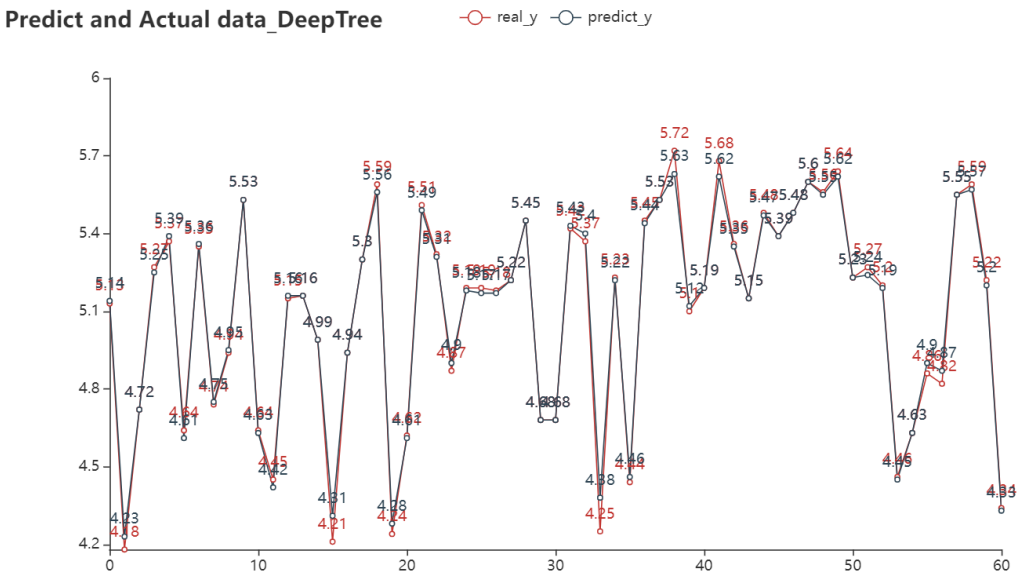


Figure 5.Multi-Grained Scanning

3.5.2 Model application:

We call the package to realize this model. Firstly, we need to do data preprocessing that use "MinMaxScaler" function to have data normalization. Then read the file and set the input is the other data except closing price and transaction date. The output we want to know is closing price. After that, we split the data into training set and test set. In our project, we use Deep forest to do regression. Since original data is already good enough, we just use Cascade forest Structure in the code. So, we recall the "CascadeForestRegressor" function to get the result. Then calculate the score index to analyze this model.

The comparison between predictions and the results are shown as followed:



4. Modal evaluation

As for the fitting effect of each model. We evaluate the prediction results through the following indicators, which can be roughly divided into model accuracy and model error stability. They are Mean Squared Error, Mean Squared Log Error, Media Absolute Error, Mean Absolute Error, R2 Score, Explained Variance Score. And the description of these index is shown as follow.

Mean Squared Error (MSE)

The statistical parameter is the mean value of the sum of squares of the errors of the predicted data and the corresponding points of the original data. And the calculation formula is:

$$MSE = \frac{SSE}{N} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Mean Squared Log Error (MSLE)

Mean squared logarithmic error (MSLE) can be interpreted as a measure of the ratio between the true and predicted values. Mean squared logarithmic error is, as the name suggests, a variation of the Mean Squared Error. And the calculation formula is:

$$MSLE = \frac{1}{n}(\log(y_i + 1) - \log(\hat{y}_i + 1))^2$$

Median Absolute Error (MAE)

The average absolute error refers to the average value of the distance between the model predicted value f(x) and the sample true value y. The formula is as follows

Mean Absolute Error

$$MAE = \frac{1}{m}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

R2 Score (R2)

The fixed coefficient reflects the proportion of all the variation of the dependent variable that can be explained by the independent variable through the regression relationship. Calculation formula:

$$R^2 = 1 - \frac{SSE}{SST} = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}, \bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$$

Explained Variance Score (EVS)

It explains the variance score of the regression model. Its value range is [0,1]. The closer it is to 1, the more the independent variable can explain the variance of the dependent variable. The smaller the value, the worse the effect.

$$EVS = 1 - \frac{var(y - y_i)}{var(y)}$$

Each index for every model is shown in the following figure:

	MSE	MDAE	MSLE	MAE	EVS	R2
AIRMA	0.001329	0.022368	0.000034	0.028235	0.759625	0.759298
Knn	0.004649	0.026667	0.000119	0.042459	0.975015	0.973360
XGBoost	0.002055	0.020649	0.000057	0.032262	0.988278	0.988226
LightGB	0.009595	0.030117	0.000310	0.061013	0.945209	0.945024
RandForest	0.000962	0.011750	0.000030	0.019403	0.994502	0.994489
DeepForest	0.000628	0.004900	0.000019	0.013670	0.996408	0.996401

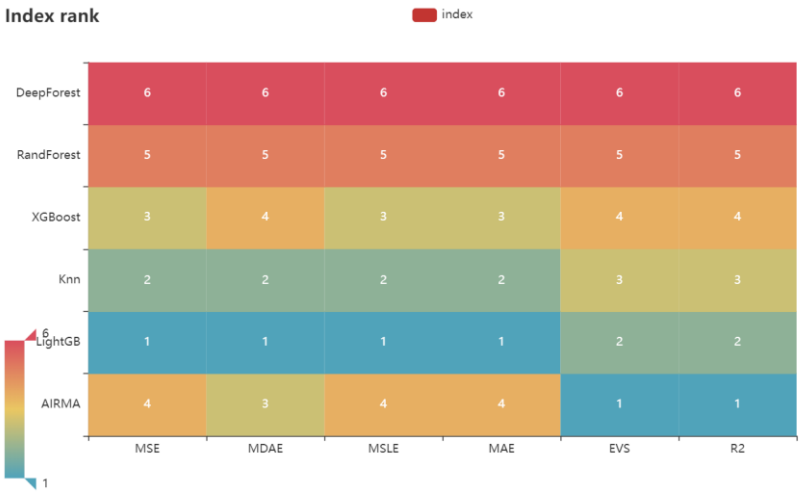
Since the table is not that clear enough, we can rank and score according to the performance of the model in each index. However, for some indicators, the smaller the value, the better the model, such as MSE, MDAE, MSLE, MAE. For other indicators, the closer the value is to 1, the better the model, such as EVS and R2. Thus, before we rank and score, we need to normalize these values, which means that the smaller the number, the better the model. And for EVS and R2, we took the reciprocal of these numbers. And we get the following table:

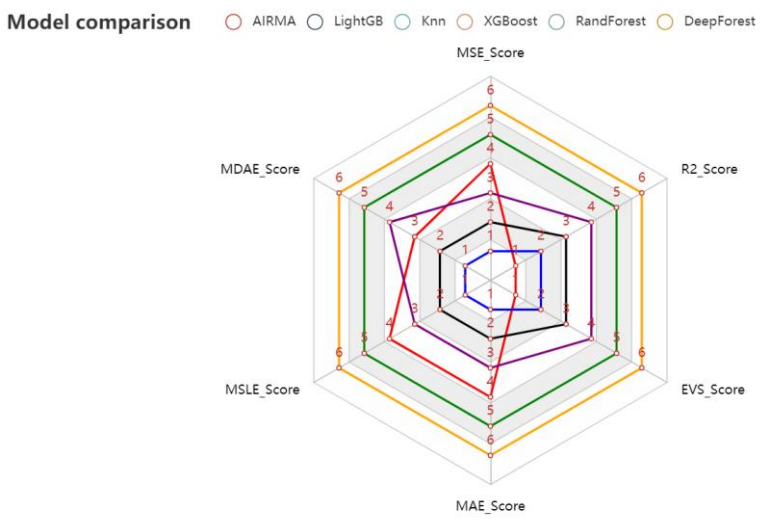
	MSE	MDAE	MSLE	MAE	EVS	R2
AIRMA	0.001329	0.022368	0.000034	0.028235	1.316439	1.317006
Knn	0.004649	0.026667	0.000119	0.042459	1.025625	1.027369
XGBoost	0.002055	0.020649	0.000057	0.032262	1.011861	1.011914
LightGB	0.009595	0.030117	0.000310	0.061013	1.057967	1.058174
RandForest	0.000962	0.011750	0.000030	0.019403	1.005528	1.005541
DeepForest	0.000628	0.004900	0.000019	0.013670	1.003605	1.003612

Then, as for each indicator, we rank and score according to the performance of each model.

	MSE	MDAE	MSLE	MAE	EVS	R2
AIRMA	4.0	3.0	4.0	4.0	1.0	1.0
LightGB	1.0	1.0	1.0	1.0	2.0	2.0
Knn	2.0	2.0	2.0	2.0	3.0	3.0
XGBoost	3.0	4.0	3.0	3.0	4.0	4.0
RandForest	5.0	5.0	5.0	5.0	5.0	5.0
DeepForest	6.0	6.0	6.0	6.0	6.0	6.0

And according to the table, we draw a heatmap and radar chart to make the result can be present more clearly.





From the map we can get that. the deep forest and the random forest perform the best, and KNN, LightGB don't perform comparing to these models.

Reference

[1] Zhou, Z. H., & Feng, J. (2017). Deep forest: towards an alternative to deep neural networks.
[2] Deep understanding of XGBoost. (2019, December 23). Retrieved December 11, 2021, from https://zhuanlan.zhihu.com/p/83901304
[3] Deep understanding of LightGBM. (2020, January. 6). Retrieved December 11, 2021, from https://zhuanlan.zhihu.com/p/99069186
[4] Random Forest algorithm and its implementation. (2018, September 4). Retrieved December 11, 2021, from https://blog.csdn.net/yangyin007/article/details/82385967
[5] Baostock: a free and open-source Python securities data interface package. (2018, Jan 14). Retrieved December 11, 2021, from https://blog.csdn.net/dongdong2980/article/details/79059789
[6] Junhan, S., Yimai W., & Wenbo Li. (2021, Augest 14). Deep Forest-Financial Forecast. Retrieved December 11, 2021, from https://aistudio.baidu.com/aistudio/projectdetail/2282775?channelType=0&channel=0
[6] Introduction to ARIMA model. (2018, April 8). Retrieved December 11, 2021, from https://blog.csdn.net/HHXUN/article/details/79858672
[7] Xiaoying L. (2015). Research on stock price trend prediction based on KNN method. Retrieved December 11, 2021, from https://kns.cnki.net/KCMS/detail/detail.aspx?dbname=CMFD201601&filename=1016017737.nh