

Enterprise Credit Rating

陈琪颖、陈以庭、陈云轩、仇实、王俊衢

1. Problem Description

1.1 Background Information

The enterprise credit rating refers to activities done by credit rating agencies to evaluate the credit rating of collected enterprise credit information based on certain financial indicators, such as P/E ratio, Sharpe ratio, liquidity risk and so on.

In the US the enterprise credit rating system is quite mature. By contrast, the system in China is still not well-structured.

1.2 Our Project

In this project, we chose to do enterprise credit rating of Chinese companies. Using available data and method learned in machine learning, we tried to establish a model to do the rating.

1.3 The Meaning of Our Project

When the rating can be given out in a model, we can know the credit level of a company much more easily, and compare companies with really different financial information much more easily.

If the model is efficient enough, the information asymmetry between investors and enterprises can be reduced, and the market might be more efficient.

2. Data Collection

After finding some information, we found that the financial information might have strong effect on the credit risk and also easy to collect and we turned to the data of companies on East Money.

Some of the companies have the rating rate and some don't. For those with the rating level, we viewed them as the training set data, and for others we view them as testing set data.

3. Feature Engineering

3.1 Variable Selection

Since the data we got have too many variables and some variables are highly related to each other. We read some articles about companies' credit rating, and used the variables most article chose.

3.2 Missing Data

In the data we got, there were many missing data. According to the financial information we learned in other classes, we knew that financial indicators might different in different industries.

We did the interpolation according to the category of industry. And we believed that the missing data is minority. In order not to affect the raw data, we used the mean of the indicator of the given industry to interpose.

3.3 Data Imbalance

Since our goal is to train a model that can do the rating, the proportion of each rating level is really

important, so we tried to make the data balance due to the rating level.

But when we were training the model, we had to split our training set into a new training set and validating set, and that caused the effect of overfitting.

In later training, we didn't do the resampling and we kept the code in our program commented out.

4. Model formulation & Evaluation

4.1 Multi-Categorical Logistic Regression

The linear model was the first model we consider. We constructed a multi-categorical logistic regression using all the preliminary processed data, hoping to perform a tentative simple model fitting. But the model results were not ideal. After multiple attempts, we found that the accuracy of multi classification logistic regression is difficult to exceed 0.4. Therefore, we hoped for more complex models, namely neural networks and decision trees.

4.2 Neural Network

Regarding the code implementation of neural networks, we had undergone three optimizations due to the changes in dataset preprocessing methods.

In the first attempt, we performed the simplest processing on the dataset, replacing all missing values with 0 and identifying some of the most effective indicators from the literature for fitting. We chose relu and softmax as the Activation function, and two hidden layers and one output layer to implement the model. After adjusting the nodes of the two hidden layers to 170 and 45 respectively and setting 25 epochs, we obtained relatively optimized results. However, the final accuracy of this model was not ideal, about 0.7. We believed that this result was due to the insufficient effectiveness of preprocessing.

Therefore, we preprocessed the dataset of the model through resampling and mean imputation. We adjusted the composition of the neural network accordingly for the adjusted dataset. The adjusted model added a hidden layer, and set the Activation function of the hidden layer to sigmoid to improve the accuracy. After trying, we found that in this dataset, the model requires more epoch to advance the accuracy climb. Therefore, we set epoch to 80. Under this epoch, the model could achieve good test set accuracy and avoid overfitting. After multiple attempts, we found the accuracy of the test set was about 0.89, which is a relatively satisfactory result.

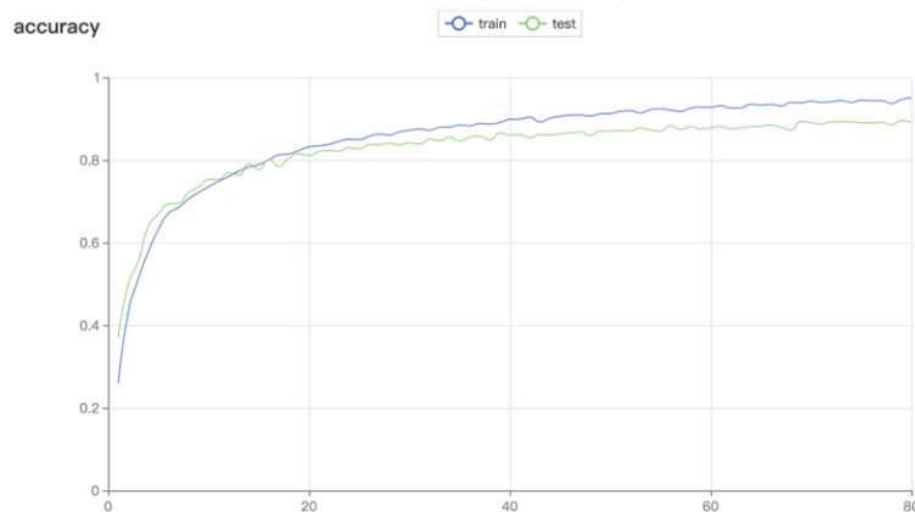
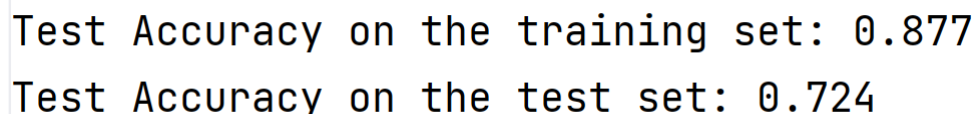


Figure 1: Accuracy Curve of Neural Network

However, we have gradually found that resampling can easily cause significant errors in situations where rating information is imbalanced. Therefore, we attempted to use only imputed datasets. In this case, we reran the previous algorithm and found a sudden decrease in accuracy, indicating that the previous set of algorithms is still unreliable.

Therefore, we adjusted the algorithm to increase its complexity in order to achieve a more satisfactory accuracy rate. In this model, we set five hidden layers, the nodes are 2000, 1000, 600, 200, 100 in turn. At the same time, the Activation function is set to tanh and sigmoid, and the epoch is set to 100 periods. Moreover, considering the fluctuation of ratings, we believed that appropriate adjustments should also be made in determining accuracy. Therefore, we adjusted the accuracy setting method and believe that the fluctuation between the predicted rating and the actual rating is less than one (we have set a total of 19 ratings based on rating classification), which is acceptable and can be considered as an accurate estimation. Therefore, after resetting the hidden layer and accuracy measurement method, we obtained a new model. The learning effect and accuracy of the model are shown in the figure below. We can see that after undergoing changes in dataset preprocessing methods, model level modifications, and evaluation index modifications, we ultimately achieved good prediction accuracy.



Test Accuracy on the training set: 0.877
Test Accuracy on the test set: 0.724

Figure 2: Learning Effect and Accuracy of Neural Network

4.3 Decision Tree

Since our project is about a classification problem, decision tree is also a good choice to do the prediction and classification. Besides, due to our classification is related to rank, some tolerance is needed to do the prediction, so we define the `acc()` function to calculate the predicting accuracy.

```
def acc(tol,label,pred):  
    correct=list()  
    for i in range(len(label)):  
        t=(abs(label[i]-pred[i])<=tol)  
        correct.append(t)  
    c=np.array(correct).astype('int')  
    return np.mean(correct)
```

With pre-processed data, we divided data into train set and test set and started the model training of decision tree. By changing the parameter in `tree.DecisionTreeClassifier()` function, we attempted to find out the local best solution.

```

depth = 10 min_sample = 3
Test set accuracy: 0.699438202247191
Train set accuracy: 0.7869198312236287
depth = 10 min_sample = 5
Test set accuracy: 0.702247191011236
Train set accuracy: 0.7721518987341772
depth = 10 min_sample = 7
Test set accuracy: 0.7162921348314607
Train set accuracy: 0.7658227848101266
depth = 15 min_sample = 3
Test set accuracy: 0.651685393258427
Train set accuracy: 0.8220815752461322
depth = 15 min_sample = 5
Test set accuracy: 0.6601123595505618
Train set accuracy: 0.7883263009845288
depth = 15 min_sample = 7
Test set accuracy: 0.6741573033707865
Train set accuracy: 0.7721518987341772
depth = 20 min_sample = 3
Test set accuracy: 0.6432584269662921
Train set accuracy: 0.8248945147679325
depth = 20 min_sample = 5
Test set accuracy: 0.6573033707865169
Train set accuracy: 0.7911392405063291
depth = 20 min_sample = 7
Test set accuracy: 0.6741573033707865
Train set accuracy: 0.7749648382559775

```

Figure 3: Accuracy of Decision Tree

As is shown in the output, we can conclude that train accuracy increases when tree depth increases, but test accuracy decreases when tree depth increases, which is due to overfitting. Also, train accuracy increases when min sample decreases, but test accuracy decreases when min sample decreases. To balance overfitting and accuracy, we chose tree depth=15, min sample = 5.

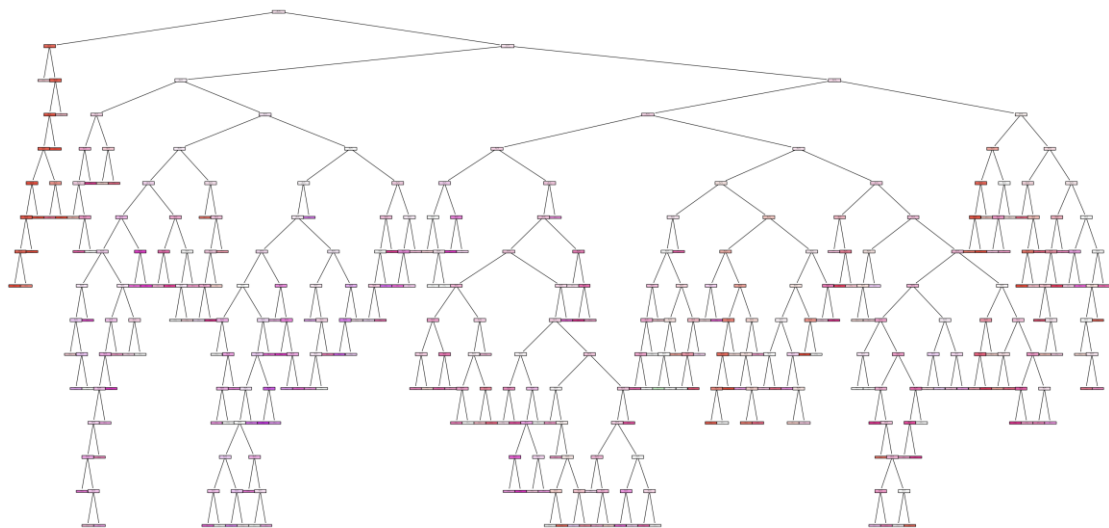


Figure 4: Visualization of Decision Tree

The decision tree picture shown above is very complex, which is due to our various classification, though we don't have so many features. However, it also gives us relatively high accuracy.

5. Exploration of Ensemble Learning

Based on the exploration above, we further tried a variety of ensemble learning algorithm, and thus

obtained our final model.

In this part, we still use the adjusted accuracy(tolerance=1) as the major evaluation index, but we also use confusion matrix (mainly for test set) to check the detailed classification of the models.

According to whether the types of individual learners are the same, the ensemble learning algorithms we tried can be roughly divided into heterogeneous integration and homogeneous integration.

5.1 Heterogeneous Integration

Based on the previous exploration, we tried heterogeneous integration first. In terms of heterogeneous integration, we tried two kinds of algorithms: Voting and Stacking, in which Voting is divided into Hard Voting and Soft Voting.

5.1.1 Voting

Hard Voting is a kind of voting algorithms which uses predicted class labels for majority rule voting, while Soft Voting predicts the class label based on the argmax of the sums of the predicted probabilities, which is recommended for an ensemble of well-calibrated classifiers.

The component learners we used are DecisionTreeClassifier, MLPClassifier and LogisticRegression.

<div><div>● evaluation(vhard,X_train,X_test,y_train,y_test)</div><div>✓ 0.4s</div></div> <div>train accuracy score: 0.8509142053445851 test accuracy score: 0.6320224719101124</div>	<div><div>● evaluation(vsoft,X_train,X_test,y_train,y_test)</div><div>✓ 0.4s</div></div> <div>train accuracy score: 0.9191279887482419 test accuracy score: 0.6882022471910112</div>
<div>test confusion matrix:</div> <div>[[0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 2 0 1] [0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0] [0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 3 0 0] [0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0] [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0] [0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0] [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0] [1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0] [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3 0 0] [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0] [2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 0] [1 0 0 0 0 0 0 0 0 0 0 0 0 4 0 1 0 1] [0 0 0 1 0 0 0 0 0 0 0 1 0 11 6 23 1 1] [5 0 0 1 0 0 0 0 0 0 0 1 9 23 34 0 2] [3 0 0 1 0 0 2 0 0 0 0 1 9 15 56 4 6] [0 0 0 0 0 0 0 0 0 0 0 1 2 12 19 7 9] [3 0 0 0 0 0 0 0 0 0 0 0 3 6 20 1 26]]</div>	<div>test confusion matrix:</div> <div>[[0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 2 0 1] [0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0] [0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 2 0 0] [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1] [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0] [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0] [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0] [1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0] [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 2 0] [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0] [1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 1] [0 0 0 0 0 0 0 0 0 0 0 0 0 3 1 2 0 1] [0 0 0 0 0 0 0 0 0 0 0 0 0 6 10 22 5 1] [0 0 0 0 0 0 0 0 0 0 0 0 0 6 26 35 2 6] [0 0 0 0 0 0 0 1 0 0 0 0 0 5 15 57 12 7] [0 0 0 0 0 0 0 0 0 0 0 0 0 1 9 16 12 12] [0 0 0 0 0 0 0 0 0 0 0 0 0 3 4 17 7 28]]</div>

Figure 5: Results of Voting Model Evaluation (Hard vs Soft)

We found that the soft voting algorithm is much better than the hard voting algorithm. By comparing the confusion matrix, we also found that the main advantage of soft voting over hard voting is that it has a strong ability to classify high-rated data.

Deficiency of Voting:

One of the drawbacks of voting is that it equalizes the contributions of individual component learners. As a result, there was no significant improvement in the performance of voting compared to component learners. This may be because although we follow the heterogeneity requirements of the voting algorithm for component learners when processing, high-quality voting also requires that component learners have relatively similar effects. Also, both voting algorithm have the problem of

running slowly.

5.1.2 Stacking

Stacking is stack of estimators with a final classifier. The component learners we used are DecisionTreeClassifier, MLPClassifier and LogisticRegression, the final learner we used is the default LogisticRegression model.

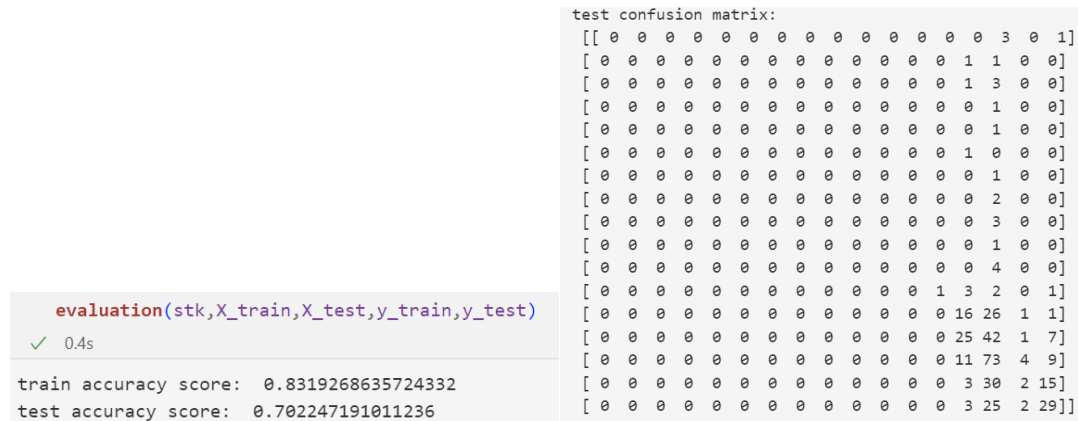


Figure 6: Results of Stacking Model Evaluation

We found that both the accuracy score and the performance of confusion matrix is better than voting, this means that based on the results of the same component learner, further fitting of the results is better than just voting.

Deficiency of Stacking:

Although this algorithm has better performance, the running speed of the algorithm is much slower than other algorithms. At the same time, it is common for low grades to be misclassified into high grades.

5.2 Heterogeneous Integration

In the exploration of heterogeneous integration, we find that the running speed of heterogeneous integration is slow (related to the complexity of the component learner we selected), and the tuning is not easy, so we further consider using homogeneous integration.

In homogeneous integration, based on the above exploration results and the consideration of running speed, we choose DecisionTreeClassifier as base learner. We tried Bagging, Random Forest, Extra Trees, AdaBoost and GBDT.

5.2.1 Bagging

Bagging classifier is an ensemble meta-estimator that fits base classifiers each on random subsets of the original dataset and then aggregate their individual predictions to form a final prediction.

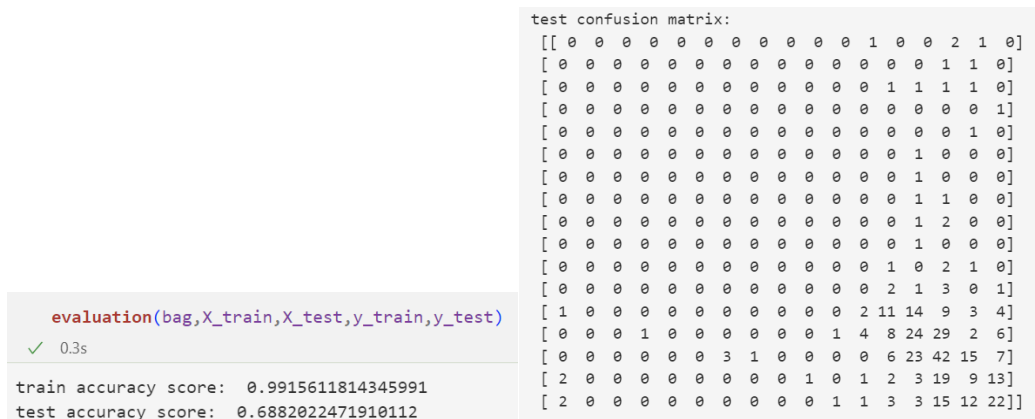


Figure 7: Results of Bagging Model Evaluation

In terms of homogeneous integration, we first tried bagging algorithm. We found that bagging algorithm can effectively reduce variance(improved data concentration).

Deficiency of Bagging:

It can be seen that the classification performance of bagging is not as good as that of stacking, and it has certain deviations.

5.2.2 Random Forest

Random Forest classifier is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.

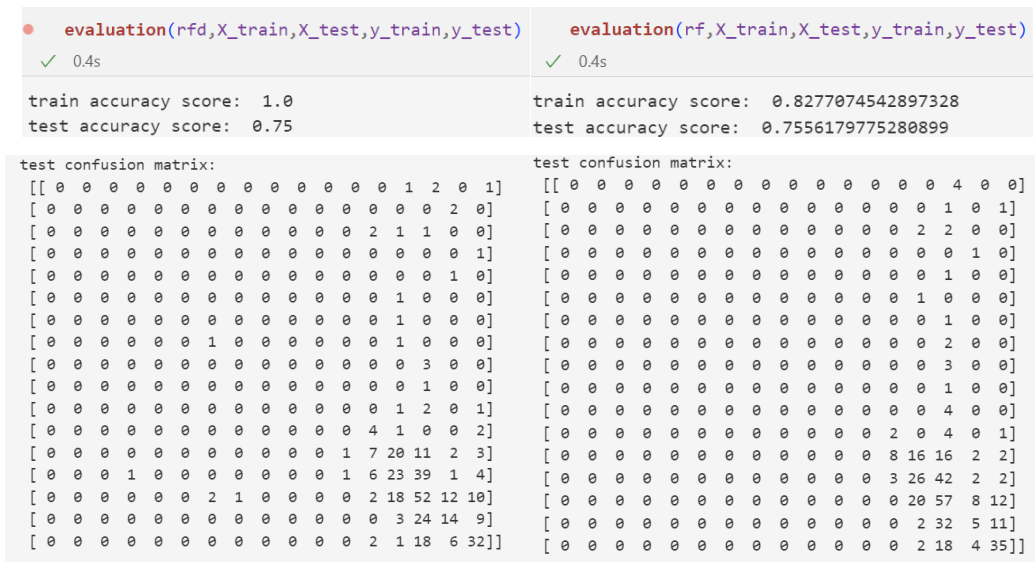


Figure 8: Results of Random Forest Model Evaluation (Default vs Tuned)

Random forests are known to prevent over-fitting, but when we first tried with the initial parameters, we found that there was over-fitting (the appearance of this unusual over-fitting is also related to the evaluation index we chose). After tuning the parameters for max_depth and min_samples_leaf, we got better results.

Deficiency of Random Forest:

The feature dimension we used is small, so the random forest does not show a very prominent advantage, and the random forest operates like a black box, so it is not easy to adjust parameters. Data imbalance also affects model performance.

5.2.3 Extra Trees

Extra Trees classifier implements a meta estimator that fits a number of randomized decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.

<code>evaluation(etc,X_train,X_test,y_train,y_test)</code>	<code>evaluation(etc,X_train,X_test,y_train,y_test)</code>
✓ 0.4s	✓ 0.4s
train accuracy score: 1.0 test accuracy score: 0.7106741573033708	train accuracy score: 1.0 test accuracy score: 0.7275280898876404
test confusion matrix: [[0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 2 0 1] [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0] [0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 2 0 0] [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0] [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0] [0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0] [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0] [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0] [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 1] [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1] [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 1 1] [0 0 0 0 0 0 0 0 0 0 0 0 0 0 3 2 1 0 1] [0 0 0 0 0 0 0 0 0 0 0 0 2 7 15 16 3 1] [1 0 0 1 0 0 0 0 0 0 0 1 8 25 31 4 4] [0 0 0 0 0 0 2 1 0 0 0 0 2 13 56 12 11] [0 0 0 1 0 0 0 0 0 0 0 0 0 6 24 9 10] [1 0 0 0 0 0 0 0 0 0 0 2 3 15 6 32]]	test confusion matrix: [[0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 1 0 1] [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0] [0 0 0 0 0 0 0 0 0 0 0 0 0 1 2 1 0 0] [0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0] [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0] [0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0] [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0] [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 0] [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 1] [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0] [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 1 1] [0 0 0 0 0 0 0 0 0 0 0 0 0 2 2 2 1 0] [0 0 0 0 0 0 0 0 0 0 0 0 2 6 18 14 1 3] [1 0 0 1 0 0 0 0 0 0 0 0 7 28 32 3 3] [0 0 0 0 0 0 2 1 0 0 0 0 1 17 56 9 11] [1 0 0 1 0 0 0 0 0 0 0 0 0 4 25 10 9] [1 0 0 0 0 0 0 0 0 0 0 1 3 16 6 32]]

Figure 9: Results of Extra Trees Model Evaluation (Default vs Tuned)

Although parameter adjustment improves the accuracy, it does not alleviate the situation of overfitting.

Deficiency of Extra Trees:

In the case of high accuracy of the base learner, Extra Trees can achieve better results, but the data set we used obviously does not meet this feature. In this case, the extra randomness of Extra Trees makes the deviation greater, and the effect is not as good as Random Forest.

5.2.5 AdaBoost

AdaBoost classifier is a meta-estimator that begins by fitting a classifier on the original dataset and then fits additional copies of the classifier on the same dataset but where the weights of incorrectly classified instances are adjusted such that subsequent classifiers focus more on difficult cases.

<code>evaluation(adaboost,X_train,X_test,y_train,y_test)</code>	<code>evaluation(adaboost,X_train,X_test,y_train,y_test)</code>
✓ 0.4s	✓ 0.4s
train accuracy score: 0.6779184247538678 test accuracy score: 0.651685393258427	train accuracy score: 0.9486638537271449 test accuracy score: 0.7471910112359551

classifiers, GBDT is difficult to perform parallel computation. At the same time, GBDT has a very slow running speed.

6. Final Model

Based on the above exploration, we finally choose random forest as our model and carry out further parameter adjustment. Considering the large number of categories and the small number of test set samples, we combined the categories into seven categories: Class C (including those rated as CCC CCC), B, BB, BBB, A, AA and AAA.

In the aforementioned exploration, although some models have achieved good accuracy, it is common for low-grade bonds to be misclassified into high-grade bonds. This has much to do with data imbalance. However, one of the goals of our research was to identify "safe" bonds, meaning that we wanted as few junk bonds as possible to be misclassified as high-quality bonds.

Based on this consideration, we construct a new evaluation index according to the classification ability of the model for low-grade bonds, and find the optimal classweight through cyclic traversal in the fitting process.

```
def cnoa(label, pred):
    correct = list()
    for i in range(len(label)):
        if list(label)[i] == 1:
            if list(pred)[i] >= 5: #A AA AAA
                t = -1
            elif list(pred)[i] in [2, 3, 4]: #B BB BBB
                t = -0.5
            else:
                t = 0
            correct.append(t)
    c = np.array(correct).astype('int')
    m = np.mean(c)
    return m
```

Figure 12: New Evaluation Index

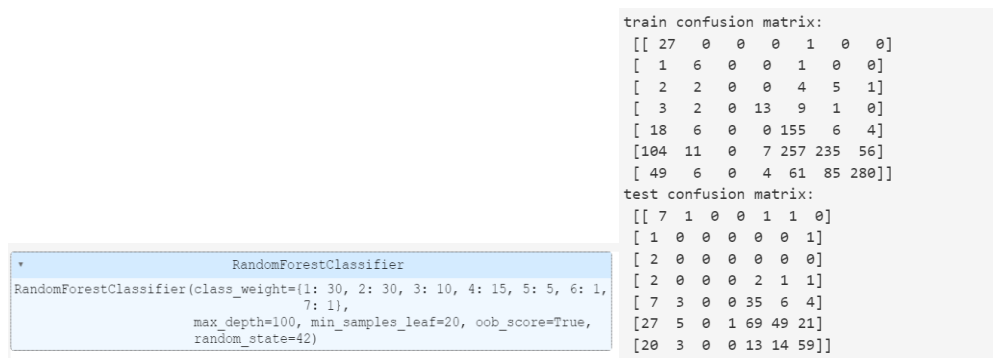


Figure 13: The Final Model and Results of Model Evaluation

Compared with other results obtained by traversal, the final model has higher prediction accuracy for middle and low grades, and alleviates overfitting to some extent. However, due to the limitation of the amount of data and the number of independent variables, the overall accuracy rate is reduced in this process.

7. Implications and Insights

To check the correctness of model, we chose some companies with our rating before to prove whether our prediction is reasonable.

In order to see the differences, we chose some of the worst ones and some of the best ones. Since different industries might be different in each indicator, the companies we chose were all from the manufacturing industry—the one we had sufficient data. And we were going to use As and Cs to do the discussion in the following part. Cs means low rating in our model and As means high rating in our model.

Operational capacity and Solvency indicate whether a company is safe or not. For Operational capacity part we check the PE, EPS and ROE, and for Solvency we check equity multiplier, current liability divided by liability and also current ratio. And a safe company would operate well and have the ability to pay their liability.

7.1 Operational Capacity

7.1.1 PE

PE equals to Price divide by Earnings. This indicator can give us the information of whether the companies operate well recently. The As in our prediction as a relatively stable PE ratio, at about 20-40, but the Cs in contrast, have ridiculous PE ratio, like less than -50 or more than 1000.

7.1.2 EPS and ROE

For EPS and ROE, it can tell as how much money we can get if we do the investment, assume that everyone is rational, what they want to get from the investment is higher return, so for similar risk, they would choose the one with higher EPS and ROE.

The As in our prediction have much higher EPS and ROE than the Cs. Somehow the Cs might have a negative one.

7.2 Solvency

7.2.1 Equity Multiplier

When it comes to equity multiplier, what we can come out with is the leverage. Higher leverage would magnify return, but also bring out much larger losses. To keep our assets safe, we won't choose a company with such high leverage. But in our prediction, the equity multiplier have no obvious different. we think that the reason behind might be that banks and investors are risk sensitive, if they think a company have high risk, they won't lend their money to it.

7.2.2 Current Liabilities/Liabilities

Then let's look at current liabilities/liabilities. Current liabilities are those need to be paid in a year or shorter time period. The higher the rate is, means that the company need more cash in a short time, and that is a bit risky for companies, if they don't have enough cash, they might need to sell their asset at discount. That might also bring out other risks. For the As they have relatively low rate, but for the Cs is almost four fifth.

7.2.3 Current Ratio

And current ratio equals to current asset divided by current liability. It means the ability of paying current liability without selling fixed asset. When the rate is too low companies can't afford to pay. But when the rate is too high, the asset is not fully used, they would have a high opportunity cost. The current ratio for As is about 2, but for Cs is very low.

Though the model is so complex that we cannot do the explanation only depend on the model, but due to our finance knowledge, the prediction is reasonable, and that might give us some advice when selecting a safe company.

Due to our goal of finding out the safe investments for the investors, we focus on the high rating ones in our prediction. Since when we are training our model the thing that matters is the predicted As should be right. In later prediction, we will keep the high rating ones.

To sum up, in this project, we trained a model that can predict credit rating level of companies according to some of the financial indicators of the company. The companies with high credit rating level are relatively safe from others. The prediction can give investor an overview of a company's credit risk, and the result can be easily understood by investors with poor financial knowledge.

In the future, our model might be upgraded to give investors more information, maybe the VaR or default possibility.