Company Bankruptcy Prediction

Muya Liu

1 Introduction

Financial distress and bankruptcy pose significant risks to investors, creditors, and stakeholders. This project aims to develop a predictive model for company bankruptcy based on the data from Taiwan Economic Journal for the years 1999 to 2009.

2 Data Preparation

2.1 missing data & data format

We first need to check the missing values and data format. We find that there are no missing values in both the training set and the test set. The data is int or float, which means all are numeric.

2.2 data exploration

(1) label

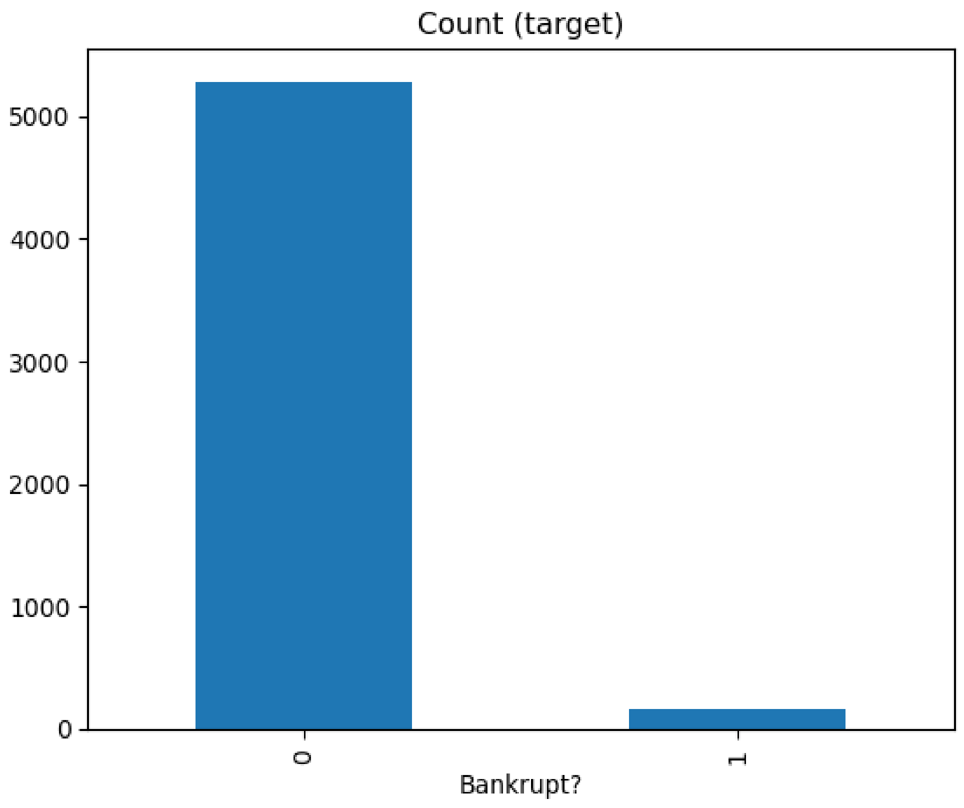

Figure 1: Amount of Class 0 & 1

The label of this project is 'Bankrupt?'. We can see that the amount of data with values of 0 and 1 is imbalanced, so we may need to resample when building our models.

(2) distribution of features

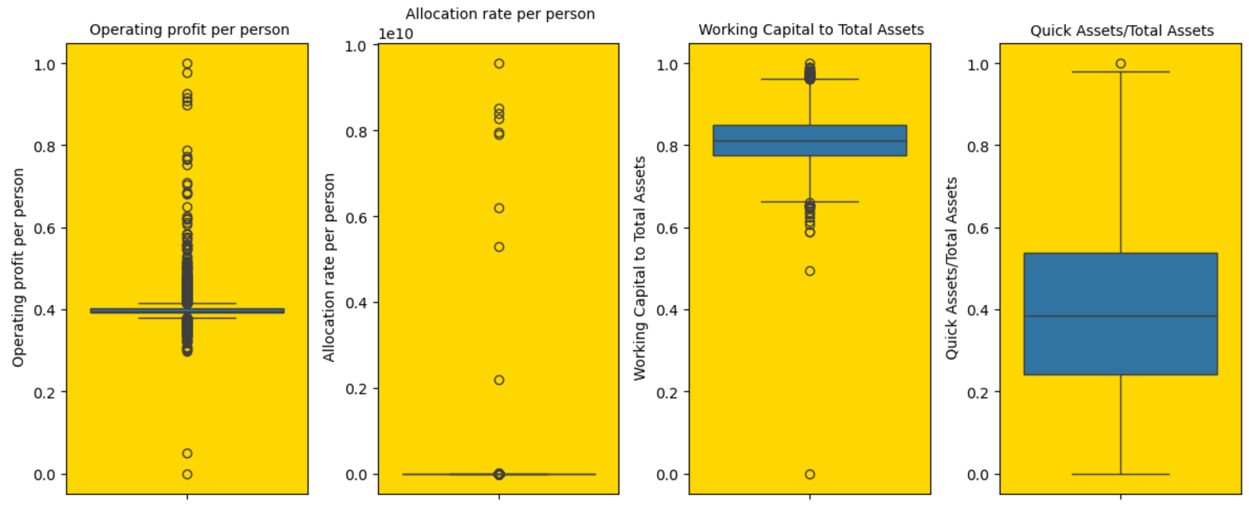

Figure 2: Boxplots of several features

The above figure shows the boxplot of some features. It can be found that many features have obvious outliers. You may need to consider the impact of this factor when modeling.

2.3 data oversampling

Since the data volume is only about 5k, and the ratio of the data with label 0 and 1 in this data set is too large (31:1), if we use under sampling, the data volume will be too small. Therefore, the most ideal state here is to oversample the data.

Code: integrating SMOTE into the model
```
pipeline = Pipeline([
    ('smote', SMOTE(sampling_strategy='auto', random_state=42)),
    ('clf', XGBClassifier(n_estimators=100, random_state=42))
])
```

We use the Pipeline in imblearn.pipeline to integrate SMOTE into the model we want to use, and we can find that the results of the model have a significant improvement(from 0.35 to 0.47).

Result: f1 score before & after oversampling
```
Cross-Validated F1 Score:
[0.42553191 0.43137255 0.3255814  0.34482759 0.24      ]
0.35346268909379175

Cross-Validated F1 Score (oversampling):
[0.5625     0.475      0.44117647 0.39534884 0.50704225]
0.4762135122637329
```

2.4 Outlier

We try to use IQR to remove outliers. The operation is as follows:

- Find the 25% and 75% quantiles of the discrete feature, and calculate IQR (IQR = Q3 - Q1)
- Calculate lower bound and upper bound:

lower bound = Q1 - 1.5 × IQR, upper bound = Q3 + 1.5 × IQR

- Regard the data beyond the Lower bound and Upper bound as outliers, and use the lower and upper bounds to replace the outliers

However, the results after processing outliers are not as good as before, so we choose not to process outliers in the final model.

Result: f1 score before & after removing outliers

```
Cross-Validated F1 Score (outlier):
[0.34146341 0.4        0.35714286 0.13953488 0.44897959]
0.3374241494669337

Cross-Validated F1 Score (oversampling , outlier):
[0.4137931  0.34920635 0.43478261 0.20833333 0.38709677]
0.3586424337754318
```

3 Feature Engineering

In this section, we try to perform selection, generation, and other operations on features, expecting to make the model perform better.

3.1 feature importance

We use XGBoost Classifier to calculate feature importance.

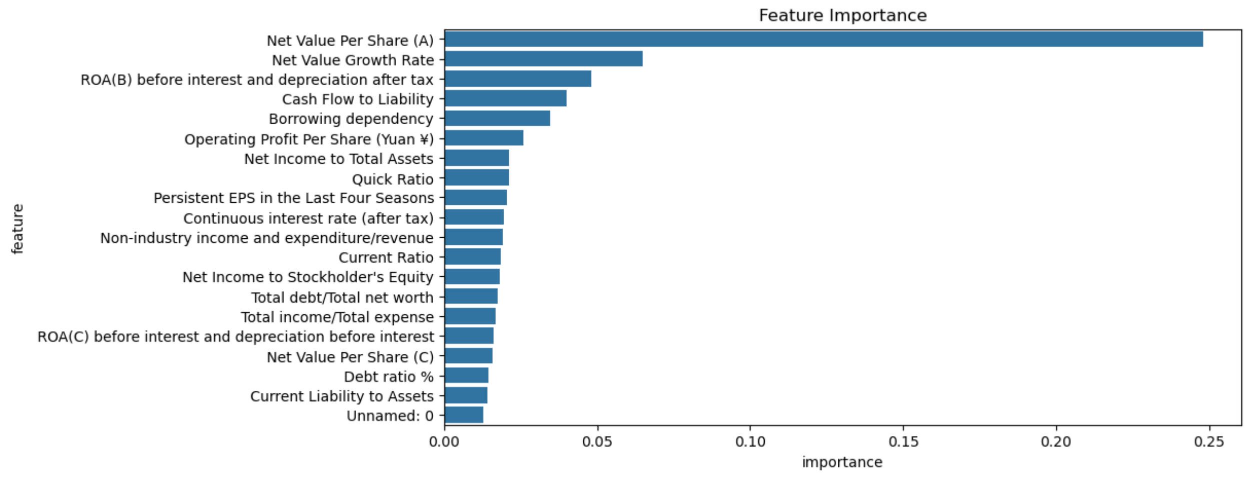

Figure 3 : Feature Importances

We select the features according to the calculated feature importance, and take the filtered features back to the model for verification. We found that setting the threshold to 0.005 will have a better result.

Result: f1 score for feature importance > 0.01,0.005

```
Cross-Validated F1 Score (oversampling, feature 1):
[0.49230769 0.53846154 0.38235294 0.4        0.51351351]
0.4653271370918429

Cross-Validated F1 Score (oversampling, feature 2):
[0.45333333 0.47058824 0.34146341 0.43956044 0.43181818]
0.4273527209280437
```

3.2 Drop Duplicated Features

We use DropConstantFeatures, DropCorrelatedFeatures and DropDuplicateFeatures in feature_engine.selection to delete features with duplicate information. We can see the effect of the model has been improved to a certain extent.

Result: f1 score before & after dropping duplicated features

```
Cross-Validated F1 Score (oversampling, feature 1):
[0.49230769 0.53846154 0.38235294 0.4        0.51351351]
0.4653271370918429

Cross-Validated F1 Score (oversampling, feature 1 dropped):
[0.5625     0.475      0.44117647 0.39534884 0.50704225]
0.4762135122637329
```

3.3 Feature Generation

We perform feature generation through the following steps:

- We first group the remaining features by name:

  Group1: XXX per XXX, such as 'Net Value Per Share (B)', 'Cash Flow Per Share'

  Group2: XXX to XXX, such as 'Cash Flow to Liability', 'Quick Ratio'

  Group3: growth rate, such as 'Operating Profit Growth Rate', 'Total Asset Growth Rate'

- Generate new features by adding, subtracting, multiplying and division operations inside the groups, and put the dataset into the model to calculate feature importance.

- Select features according to their importance.

Here we need to try the results of the features model under different thresholds (0.01, 0.005, 0.001, 0.0005) and check whether the results of adding these features to the model are improved. If the model's results are significantly improved, we retain the results of feature generation and selection and apply them to the test set.

Result: f1 score of Group1 with feature importance > 0.005,0.001

```
Cross-Validated F1 Score (oversampling, feature 1):
[0.55384615 0.51219512 0.42105263 0.44705882 0.51282051]
0.4893946487452491

Cross-Validated F1 Score (oversampling, feature 2):
[0.57142857 0.52054795 0.3880597  0.39506173 0.45      ]
0.46501958930432996
```

Result: f1 score of Group2 with feature importance > 0.01,0.005,0.001

```
Cross-Validated F1 Score (oversampling, feature 1):
[0.50574713 0.46       0.27083333 0.44897959 0.37037037]
0.41118608439544396

Cross-Validated F1 Score (oversampling, feature 2):
[0.49315068  0.53333333  0.475          0.43956044  0.5           ]
0.4882088915650559

Cross-Validated F1 Score (oversampling, feature 3):
```

```
[0.52307692 0.5        0.4057971  0.41975309 0.52941176]
0.4756077751303668
```

Result: f1 score of Group3 with feature importance > 0.01,0.005,0.001, 0.0005

```
Cross-Validated F1 Score (oversampling, feature 1):
[0.54761905 0.41975309 0.3908046  0.39583333 0.38202247]
0.42720650739667915

Cross-Validated F1 Score (oversampling, feature 2):
[0.47222222 0.525      0.35       0.4        0.56      ]
0.46144444444444443

Cross-Validated F1 Score (oversampling, feature 3):
[0.5915493  0.54761905 0.44736842 0.41975309 0.53164557]
0.5075870840972666

Cross-Validated F1 Score (oversampling, feature 4):
[0.50847458 0.525      0.43037975 0.41463415 0.5       ]
0.4756976938896186
```

According to the results, we found that Group1 has a significant improvement in model performance when threshold = 0.005, and Group3 has a significant improvement in model performance when threshold = 0.001, while Group2 has a poor feature generation performance.

4 Model Construction

4.1 Model selection

At first, we choose the commonly used classification models, and combine them with data oversampling SMOTE. Here we use StratifiedKfold to divide the training set into 5 folds and use cross validation to evaluate the results of the model.

Code: integrating SMOTE into the model

```
pipeline = Pipeline([
    ('smote', SMOTE(sampling_strategy='auto', random_state=42)),
    ('clf', model)
])
```

We train the model directly without adjusting the parameters and the chart below shows the result of each model:

Table 1: f1 score for different baseline models

model	result for cross validation					mean
LogisticRegression	0.1818	0.2743	0.1698	0.1798	0.2043	0.2020
RidgeClassifier	0.2545	0.2470	0.2432	0.2667	0.2233	0.2470
SGDClassifier	0.0623	0.0629	0.0238	0.0500	0.0620	0.0522
RandomForestClassifier	0.5672	0.5581	0.4762	0.3294	0.4935	0.4849
GradientBoostingClassifier	0.4752	0.4786	0.3750	0.3500	0.4220	0.4202
AdaBoostClassifier	0.3580	0.3279	0.3067	0.2892	0.3291	0.3222
ExtraTreesClassifier	0.4828	0.5063	0.4286	0.3200	0.3684	0.4212
BaggingClassifier	0.4675	0.4691	0.4301	0.3704	0.4762	0.4427

XGBClassifier	0.5231	0.5455	0.4167	0.4286	0.5263	0.4880
LGBMClassifier	0.5672	0.6000	0.4444	0.4419	0.5250	0.5157
CatBoostClassifier	0.4944	0.5049	0.3958	0.4314	0.4681	0.4589
MLPClassifier	0.0000	0.0721	0.0816	0.1081	0.1060	0.0736
SVC	0.0000	0.0000	0.0000	0.1275	0.0000	0.0255
KNeighborsClassifier	0.2500	0.1951	0.1500	0.0488	0.1463	0.1580

We can see that random forest, xgboost, and lightgbm perform outstandingly, so we choose these three models for the following parameter adjustment and model fusion.

4.2 Parameter adjustment

We use grid search to adjust parameters and use cross validation to test the results.

Code: Hyper parameters for tunning

```
param_grid_rf = {
    'smote__sampling_strategy': [0.5, 1.0, 'auto'],
    'rf__n_estimators': [50, 100, 200],
    'rf__max_depth': [None, 10, 20],
    'rf__min_samples_leaf': [1, 2, 4]}
param_grid_xgb = {
    'smote__sampling_strategy': [0.5, 1.0, 'auto'],
    'xgb__n_estimators': [50, 100, 200],
    'xgb__max_depth': [3, 5, 10],
    'xgb__learning_rate': [0.1, 0.3, 0.5]}

param_grid_lgbm = {
    'smote__sampling_strategy': [0.5, 1.0, 'auto'],
    'lgbm__n_estimators': [100, 200, 300],
    'lgbm__max_depth': [3, 5, 7],
    'lgbm__learning_rate': [0.01, 0.05, 0.1]}
```

The final results of the parameter tunning are as follows:

Table 2: Best Parameter & f1 score after parameter tunning

model	parameter	score
random forest	rf__max_depth': 20, 'rf__min_samples_leaf': 2, 'rf__n_estimators': 50, 'smote__sampling_strategy': 0.5	0.4895
xgboost	smote__sampling_strategy': 0.5, 'xgb__learning_rate': 0.3, 'xgb__max_depth': 5, 'xgb__n_estimators': 100	0.5049
lightgbm	lgbm__learning_rate': 0.05, 'lgbm__max_depth': 7, 'lgbm__n_estimators': 300, 'smote__sampling_strategy': 1.0	0.5108

4.3 Model fusion

Based on three models with tuned parameters, we tried to use soft voting, hard voting and stacking (meta model: catboost) to conduct model fusion, and the results were as follows:

Table 3: f1 score after model fusion

model	result for cross validation					mean
soft voting	0.5806	0.557	0.5	0.4198	0.5333	0.51814
hard voting	0.5714	0.575	0.5135	0.4146	0.5333	0.52158
stacking	0.28	0.4211	0.3846	0.4058	0.4483	0.38795

The effect of voting is much higher than stacking, and the result of hard voting is higher than that of soft voting. Therefore, we finally chose to use hard voting for model fusion to predict the bankruptcy probability on the test set, and the mind map below shows the structure of the model we finally built.

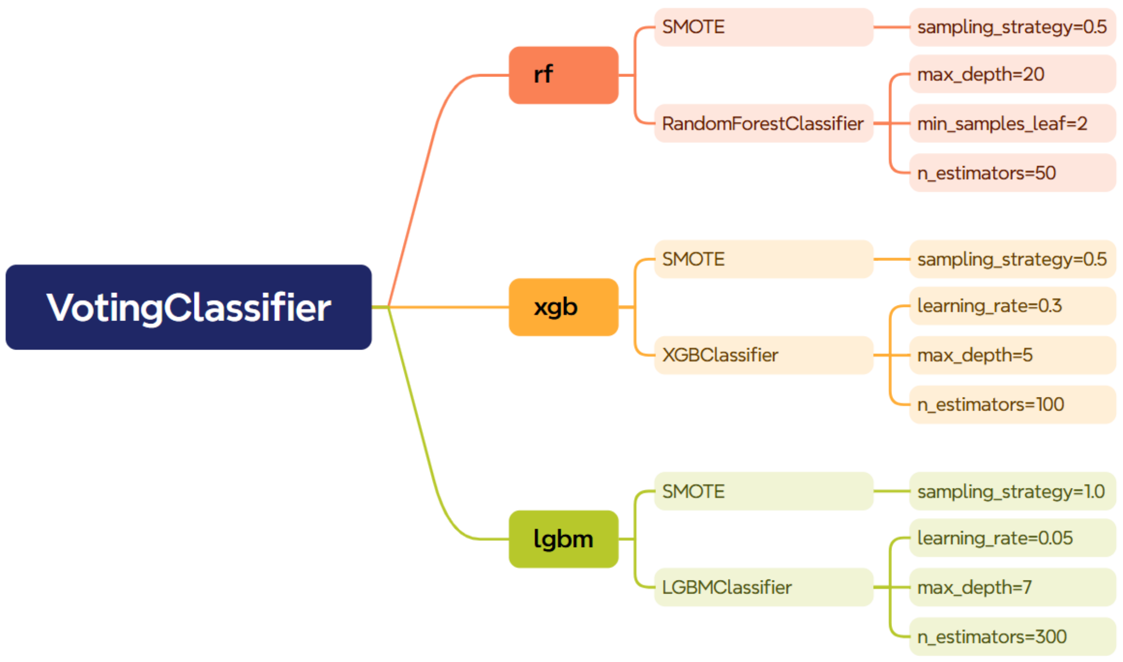

Figure 4: Structure of the final model

5 Conclusion

In this project, we oversampled the imbalanced data, performed feature generation and selection. Then we built random forest, xgboost and lightgbm models and fused them using hard voting. The final f1 score for cross validation on the training set is 0.5215.

References

1 Websites

https://www.kaggle.com/datasets/fedesoriano/company-bankruptcy-prediction/code

https://www.kaggle.com/code/jacopoferretti/company-bankruptcy-classif-w-feature-selection

https://www.kaggle.com/code/jacopoferretti/examples-of-feature-selection-in-classification

https://www.kaggle.com/code/mennaalaaali/company-bankruptcy-dbscan-rn-smote

https://www.kaggle.com/code/semihgnak/bankruptcy-perdiction-pca-smote-99-f1-score

https://www.kaggle.com/code/sayamkumar/company-bankruptcy-prediction#Feature-Engineering

https://www.kaggle.com/code/max398434434/bankruptcy-logreg-model-acc-93-f1-macro-0-68#Data-preparation

2 AI tool:

(1) Github Copilot: mainly used for quick completion of similar or repeated code

(2) Chatgpt: mainly used to provide baseline and also used to search related web pages

Questions for chatgpt:

How to combine oversampling with the model I want when using cross validation?

Please list the commonly used classification and write the baseline of the model?

List several common methods of model fusion for me?

Appendix

Appendix 1 Boxplot of all the data.docx

Appendix 2 parameter tunning results.xlsx

Appendix 3 prediction for test.csv