Case Study 4: Bankruptcy Prediction

Matt Farrow

June 17, 2022

1 Introduction

In this case study, my goal is to predict whether a company may declare bankruptcy using [data](https://archive.ics.uci.edu/ml/datasets/Polish+companies+bankruptcy+data) collecting from Polish companies between 2000-2012 for companies that went bankrupt, and 2007 to 2013 for those that did not.

The objective of this case study is to use Random Forest and XGBoost to accurately predict bankruptcy so that the company has an opportunity to potentiall divest their investments and save, or at least now lose as much, money.

2 Methods

## 2.1 Data Examination

The initial data set contained in five files, each of which spans a year of information.

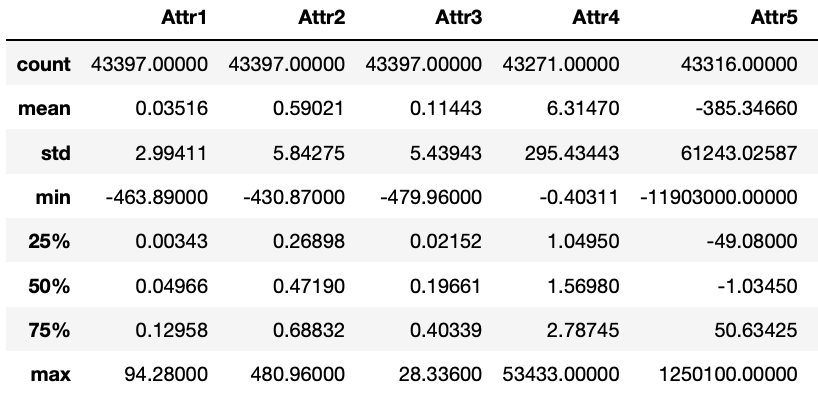


Figure 1: Sample of Original Data

I next looked at the response variable, class, to see what the distribution of bankruptcies to non-bankruptcies was.

Chart

Description automatically generated

Figure 2: Distribution of Response Variable

Unfortunately, on intial observation the data appears to be significantly imbalanced. That said, the imbalanced nature of the data confirms that our company is making far more investments with companies who do not go bankrupt than with those who do. However, this observation only applies to the number of investments, not the size of them. I decided that I was to address the scaling I would do so in the model building pipeline.

After examining the response variable, I next looked at how many missing values were in the data and noted attributes varied between no missing values and almost 19,000. I made the decision to drop the attributes that contained more than 1,000 missing values; the rest would be imputed during model building.

Finally, before proceeding to model building, I checked attribute correlation using a number of different thresholds for isolating which attributes were most closely correlated. Starting at a level of 0.8 and progressing to 0.9, 0.95, and 0.98, I made the decision to drop only the attributes that had a 98% or great correlation (Figure 3: **Final Correlation Heatmap After Removing Highly-Correlated Variables**).

Chart

Description automatically generated

Figure 3: Final Correlation Heatmap After Removing Highly-Correlated Variables

## 2.2 Model Preparation & Execution

Two approaches were taken to model building. In both, the original data was first split into test and training sets with missing values imputed using the attribute’s mean value. In addition, the data was scaled ahead of model building in order to correct for the class imbalance that appears in the data. The RobustScaler was chosen because of its ability to handle outliers.

For both the Random Forest as well as the XGBoost models, a list of hyperparameters was defined along with cross-validation to be used in the grid search process of each model.

3 Results

## 3.1 Random Forest Results

The Random Forest model did perform quite admirably (Table 1: **Random Forest Performance Metrics**) with an accuracy score of 0.9455. In addition to the performance metrics, the ROC curve (Figure 4: **ROC Curve for Random Forest Model**) and confusion matrix (Figure 5: **Random Forest Confusion Matrix**) give additional clarity to the model’s performance.

|  |  |
| --- | --- |
| Metric | Value |
| Accuracy | 0.9455 |
| Recall | 0.1531 |
| Precision | 0.3497 |

Table 1: Random Forest Performance Metrics

Chart, line chart

Description automatically generated

Figure 4: ROC Curve for Random Forest Model

In the confusion matrix, we can see that the model did misclassify over 300 companies as being ones that are likely to file for bankruptcy when in face they did not. That error rate would need additional evaluation to determine whether that is an acceptable risk to reward ratio for putting this model into production.

Chart, treemap chart

Description automatically generated

Figure 5: Random Forest Confusion Matrix

## 3.1 XGBoost Results

The XGBoost model was configured to run 1,000 rounds with an early stopping rounds value of five. As a result, the final trained model stopped at 632 trees (Figure 6: **Results of XGBoost Log Loss Error for Train & Test Data**). The best estimator occurred at 193 trees.

A picture containing chart

Description automatically generated

Figure 6: Results of XGBoost Log Loss Error for Train & Test Data

The model’s overall accuracy was 0.94 which puts it on par wit the Random Forest model.

|  |  |
| --- | --- |
| Metric | Value |
| Accuracy | 0.94 |
| Recall | 0.11 |
| Precision | 0.51 |

Chart, line chart

Description automatically generated

Figure 7: XGBoost ROC Curve

Chart, treemap chart

Description automatically generated

Figure 8: XGBoost Confusion Matrix

4 Conclusion

While both models provide strong accuracy results, the XGBoost model did provide the greatest results. This is due in no small part to the way that the algorithm learns from its mistakes to minimize loss. One challenge for XGBoost in this scenario is its more “black box” nature than the Random Forest model. Depending on the amounts of money invested in each of these companies, it is possible that management would feel more comfortable with a model that offered slightly more interpretability, especially if this is a new idea for them. Either way, both models offer strong performance as well as pros and cons for consideration.

# Appendix

## Code

Code begins on the following page.