Review of Bankruptcy Prediction Using Machine Learning

By: Javiy Wang **May 20, 2021**

Table of Contents

Introduction

From financial institutions to iconic department stores, the coronavirus has seemingly spared no one in its devastation of the world's economy. The fall of consumer demand, stay-at-home orders, a reduction in consumer spending, and a mandate for certain businesses to stay closed continue to take their toll on the financial stability of said businesses. Bankruptcy prediction is an important technique of forecasting a company's future as well as projecting a company's financial distress. The reason for predicting bankruptcy is basic in evaluating the monetary state of an organization and its prospects in its operations. Corporate bankruptcy is an exceptionally critical phenomenon in financial institutions. The monetary security of an organization plays a critical role in determining its future.

Additionally, bankruptcy prediction is fundamental for financial backers just as providers or retailers to the business. Credit loan specialists and financial backers need to assess the monetary risk of an organization before settling on an investment or credit-giving decision to avoid a significant loss by banks and other credit lenders. An organization's providers or retailers consistently manage credit exchanges for the organization, and they likewise need to completely comprehend the organization's monetary status and settle on choices on the credit exchange. To accurately anticipate an organization's monetary distress is of incredible worry to the different partners of an organization.

The purpose of this paper aims to see which variables are most important in predicting whether a company will fall into bankruptcy. Machine learning methods that will be used for this paper are Logistic Regression, Random Forest, Xgboost, and Gradient Boosting Machines. To evaluate the performance of each of my models, I will primarily be using 4 criteria to assess my models which are precision, recall, f1 score, and area under the curve (AUC). Precision quantifies the number of positive class predictions that actually belong to the positive class. Recall quantifies the number of positive class predictions made out of all positive examples in the dataset.[2] The differences between accuracy, recall, and precision can be seen in Figure 1. One might notice I neglected one of the most common metrics in evaluating a model, which would be accuracy score. I neglected to use the accuracy metric for two reasons. First, my response variable is imbalanced with over 6000 instances of non-bankruptcy, and only 220 instances of bankruptcy. Secondly, in the case of bankruptcy prediction we are more interested in the sensitivity rather than specificity. Most financial models are like this, where predicting the true positive rate is of much greater importance. Such as a credit card company predicting if an individual is being accused of fraud. Therefore, rather than correctly predicting all the cases where no fraud is involved, financial institutions are much more interested in finding those few transactions of fraud that could greatly benefit both the client and company itself. By comparing the precision, recall, AUC, and f1 scores of each of the models to one another, I will be able to

obtain the best model in predicting company bankruptcy.

$$\textbf{Precision} = \frac{\text{True Positive}}{\text{Actual Results}} \quad \text{or} \quad \frac{\text{True Positive}}{\text{True Positive + False Positive}}$$

$$\textbf{Recall} = \frac{\text{True Positive}}{\text{Predicted Results}} \quad \text{or} \quad \frac{\text{True Positive}}{\text{True Positive + False Negative}}$$

$$\textbf{Accuracy} = \frac{\text{True Positive + True Negative}}{\text{Total}}$$

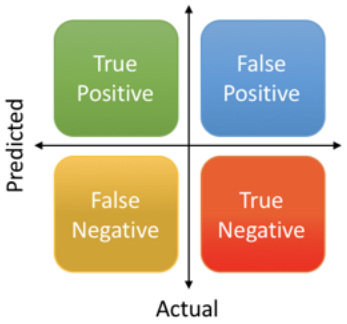

Figure 1: Precision Recall And Accuracy

Data Description

The data was collected from the Taiwan Economic Journal from the years of 1999 to 2009. Company bankruptcy was defined based on the business regulations of the Taiwan Stock Exchange. For the purpose of my project the second version of the dataset can be found on Kaggle (www.kaggle.com/fedesoriano/company-bankruptcy-prediction). Version 2 of the data has updated column names and data descriptions to make the data easier to understand (Y = Output feature, X = Input features). The data contains 6819 instances, 1 response variable, and 95 explanatory variables, there are no NA values, and all the columns from the data set are numeric values. From Figure 2 you can see the output feature and many of the possible input features. Evidently there are many more variables I will just not mention until they are relevant.

```
RangeIndex: 6819 entries, 0 to 6818
Data columns (total 96 columns):
 #   Column                                                  Non-Null Count  Dtype
---  ------                                                  --------------  -----
 0   Bankrupt?                                               6819 non-null   int64
 1    ROA(C) before interest and depreciation before interest 6819 non-null   float64
 2    ROA(A) before interest and % after tax                  6819 non-null   float64
 3    ROA(B) before interest and depreciation after tax       6819 non-null   float64
 4    Operating Gross Margin                                  6819 non-null   float64
 5    Realized Sales Gross Margin                             6819 non-null   float64
 6    Operating Profit Rate                                   6819 non-null   float64
 7    Pre-tax net Interest Rate                               6819 non-null   float64
 8    After-tax net Interest Rate                             6819 non-null   float64
 9    Non-industry income and expenditure/revenue             6819 non-null   float64
 10   Continuous interest rate (after tax)                    6819 non-null   float64
 11   Operating Expense Rate                                  6819 non-null   float64
 12   Research and development expense rate                   6819 non-null   float64
 13   Cash flow rate                                          6819 non-null   float64
```

Figure 2: Data Set Attributes

Data Cleaning And Transformations

Unlike most real world data this data set does not contain any NA values, however, there are still changes that need to be made to the data set in order to achieve higher scores for precision and recall. First there is the problem with outliers and normalizing the data set. The goal of normalization is to scale the values of numeric columns in the dataset to a range of [0,1], while also no distortion in the differences in the ranges of values. The formula used to scale this data is Xnormalized = (X - XMinimum)/(Xmaximum - Xminimum), or better known as the MinMaxScaler function in Python.[15]

For machine learning purposes, we should only normalize the data if the ranges between predictors are vastly different. In the case of this data set we have features that range from 0 to 1 and we have other features that range in the millions. In this instance the variable that ranges in the millions is about a million times larger than the range from 0 to 1. When we do further analysis, and we begin to model the data using machine learning, the attributes in the range of the millions will intrinsically influence the result more due to its larger value. Even though machine learning methods will place more value into variables with larger ranges, it does not necessarily mean that it is more important as a predictor. Thus, I will be normalizing the data before running the data through my machine learning models to bring all the values into the same range.[15]

Before we normalize the data we have to examine the outliers in the data set. As you can see from the boxplots before and after outlier removal from Figure 3 we have better looking box plots. The original formula I used to eliminate outliers was if observations are (Q1-1.5*IQR) or *(Q3+1.5*IQR). However, using this standard method I eliminated over 2000 out of the 6819 observations. Typically outliers will fall into one of 3 categories: **Outliers**: Data points that are far away from the mean or median. e.g., a heart rate of 140 beats/min for a subject during resting condition, **Anomaly**: A data point that is not legitimate due to error in data collection or device issue. e.g. a heart rate of 10 beats/min, **Outliers-anomaly**: a data point that is far away from the mean or median, but you cannot decide the source of it. It would be fine to remove outliers if it were the anomaly case, however, we have no information about the data collection and data entry of this data set so eliminating outliers will not do here.[7]

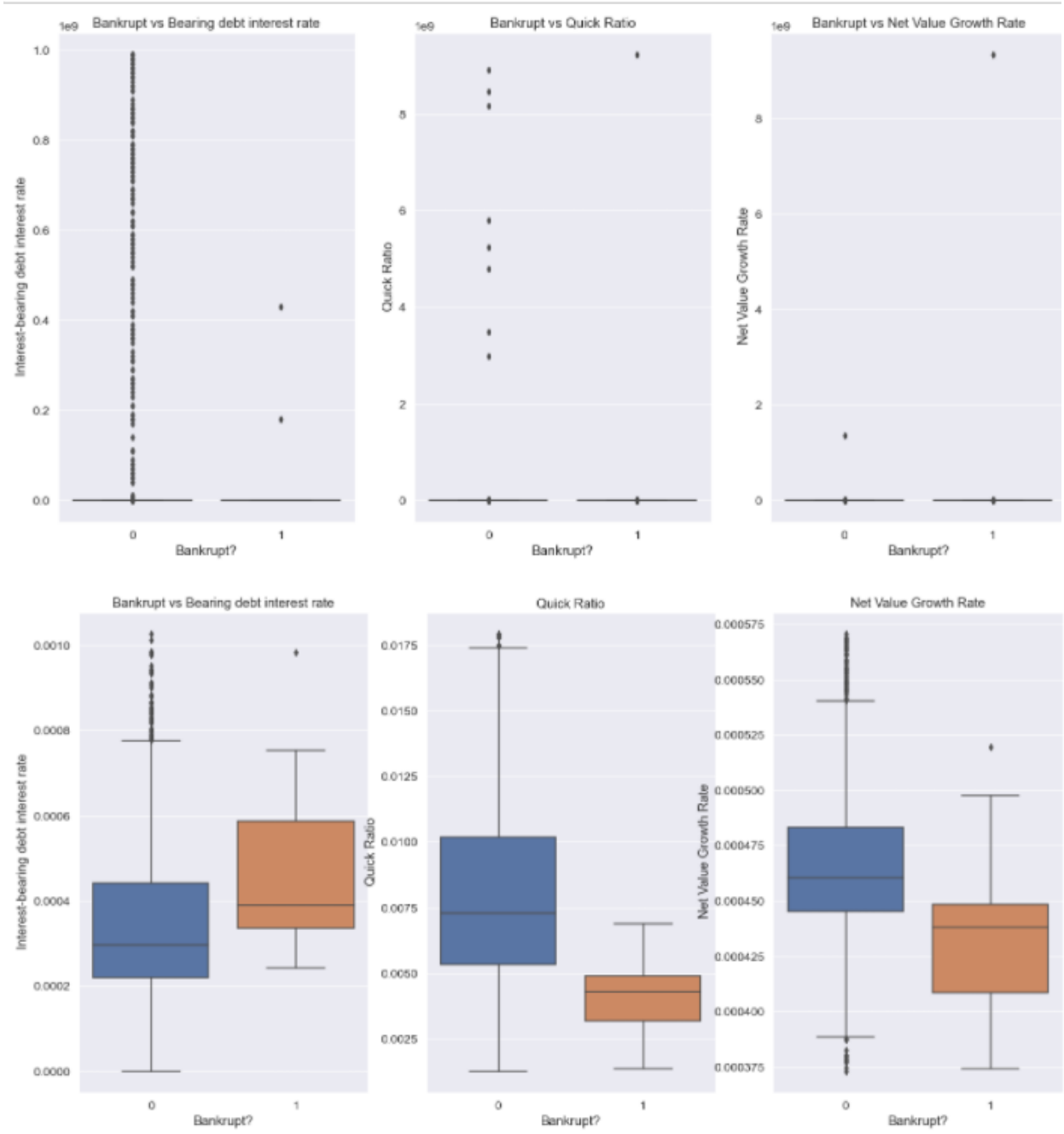

Figure 3: Before And After Boxplots After Outlier Removal

Instead I will use a new method to deal with outliers by transforming the data with a log(x+1) transformation. My new criteria to transform a column is if observations in the 100th percentile are at least 100 times greater than 99th percentile values, or there are 68 or less records for 100th percentile. For any variables satisfying my outlier condition I will transform from x to log(x+1).[1] Before applying my transformations, it splits the data into 24 non fractional only columns and 72 fractional columns for a total of 95 features. A 'fractional only column' is defined by a column whose values are strictly in between 0 to 1, thus making it a column consisting of only fractions. Only data from the non fractional columns are considered for a log

transformation. As you can see in Figure 4, we have transformed 19 out the 24 non fractional columns in boxplot format. After the log transformation, 5 of the columns are corrected in regards to outliers, while the other 14 columns remain practically unchanged. However, I found this method to be preferable since it handles the outlier problem without removing 2000 observations.

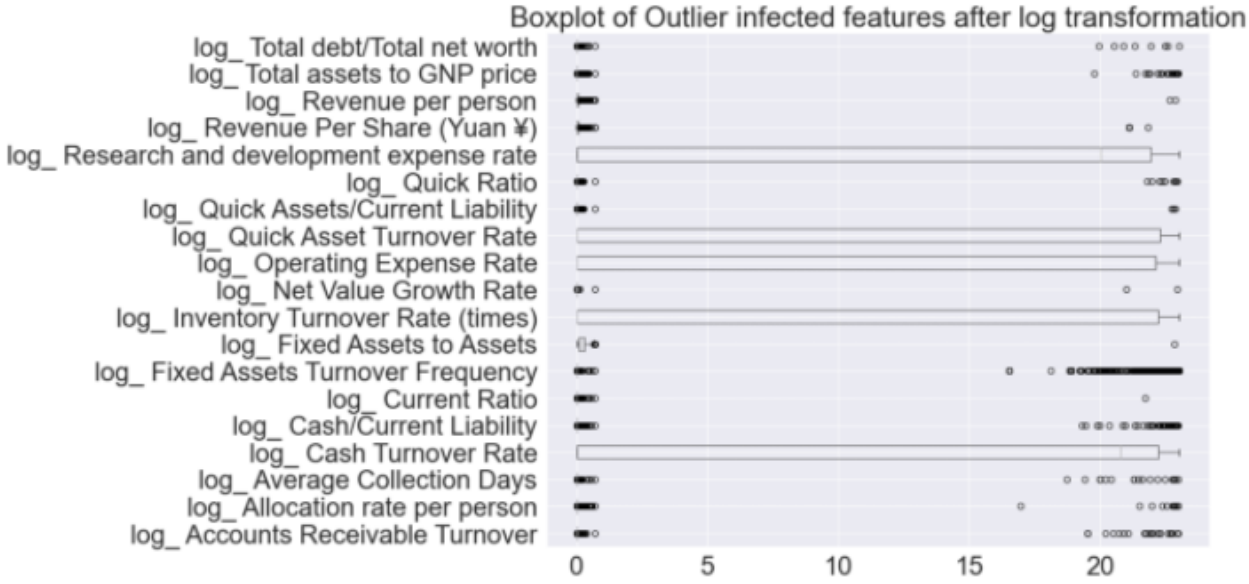

Figure 4: Boxplot of Outlier Features After Log Transformation

Variable Selection

Why would one go through variable selection when the machine learning methods can handle all 95 attributes and make an accurate model with that information? If unnecessary variables are placed into the machine learning models then you will get unnecessary noise for the output of your models. Feature/variable selection is especially important in a situation where the number of features at your disposal is very large, much like the data set I am currently using.[5] Furthermore, the main reasons to use feature selection boils down to: it enables the machine learning algorithm to train faster, it reduces overfitting, it can improve metrics such as accuracy and AUC if the right subset of features are chosen, lastly, it will reduce dimensionality and complexity in the model, making it easier to interpret.

There are 4 different ways I did variable selection, which were: variance threshold, recursive feature selection, logistic stepwise, and through economic factors. Based on these 4 methods I wanted to see how they would compare to one another when selecting from the 95 features, what they had in common, what was different, why are there differences?

From the sklearn library I imported the VarianceThreshold function.[12] I used this as a foundation, however, I made alterations to the function to account for unscaled data. If the features have different medians, quartiles, and distributions, then we can not compare these features using a variance threshold. One solution to this problem is by dividing all the features by

their mean.[13] The idea behind using a variance threshold was to only consider variables with high variability in the columns[7]. If variables had low variance, it would be difficult for the machine learning algorithm to distinguish between bankrupt and non bankrupt companies with values so close to one another. There are many ways to decide what threshold to use. For instance, one could pick a value that seems ok for you and your dataset through intuition then eliminate the variables below your chosen threshold. Another way to do this is by creating a function, which when given a threshold, tells you how many variables would be removed, if you used that threshold. Then you could plot out the number of variables removed/kept with different thresholds.[7] I used a mix of both methods, and with this I can eliminate 74 columns off the bat, which can be seen in Figure 5. In order to check my work I also utilized the caret package in R and the function nearZeroVar() to see whether or not this function would eliminate the same columns as 'VarianceThreshold ,' and sure enough the results were the same.[11]

NearVarZero Results
Net.Income.Flag
Operating.Profit.Rate
Pre.tax.net.Interest.Rate
Operating.Profit.Growth.Rate
After.tax.net.Interest.Rate
Interest.Expense.Ratio
Working.Capital.Turnover.Rate

	Variance
Net Income Flag	0.000000
Operating Profit Growth Rate	0.000161
Operating Profit Rate	0.000170
Cash Flow to Sales	0.000193
Working capitcal Turnover Rate	0.000227
Working Capital/Equity	0.000252
Pre-tax net Interest Rate	0.000261
Continuous interest rate (after tax)	0.000263
After-tax net Interest Rate	0.000283
Net Income to Stockholder's Equity	0.000299
Interest Expense Ratio	0.000317
No-credit Interval	0.000388

Total assets to GNP price	408.274712
Accounts Receivable Turnover	473.277969
Average Collection Days	680.550015
Allocation rate per person	684.500651
Liability-Assets Flag	851.375000
Quick Ratio	853.129754
Total debt/Total net worth	1453.890641
Revenue Per Share (Yuan ¥)	1514.333244
Quick Assets/Current Liability	2281.318433
Revenue per person	3450.486394
Net Value Growth Rate	5312.004777
Fixed Assets to Assets	6817.997213
Current Ratio	6817.999489

Figure 5: Highest Variances Amongst The Predictor Variables

In Figure 6 we see the scores from Random Forest recursive feature elimination by using the function RFE().[13] I decided to use the Random Forest algorithm and run a for loop to fit a model with 5, 10, 15, 20, and 25 features, while reporting back their corresponding score. When looking at the code one can see that as we added more variables the scores showed miniscule improvement. By the time the model had 5 features the test score was already at 0.965. When adding an additional 10 more variables the score only increased by 0.004. However, the model with 15 variables was still taken into account for the final model since it still received the highest score amongst the 5 models.

```
n_features = [5, 10, 15, 20, 25]
for i in n_features:
    # Building the model based feature selection
    select = RFE(RandomForestClassifier(n_estimators=100, random_state=42), n_features_to_select=i)

    select.fit(X_train, Y_train)

    mask = select.get_support()

    X_train_rfe = select.transform(X_train)
    X_test_rfe = select.transform(X_test)

    score = RandomForestClassifier().fit(X_train_rfe, Y_train).score(X_test_rfe, Y_test)

    print("Test score: {:.3f}".format(score), " number of features: {}".format(i))
```

```
Test score: 0.965  number of features: 5
Test score: 0.965  number of features: 10
Test score: 0.969  number of features: 15
Test score: 0.966  number of features: 20
Test score: 0.967  number of features: 25
```

Figure 6: Recursive Feature Elimination And Test Scores

Logistic stepwise was utilized next, by using the caret and MASS packages and the function step().[6] This method can be used backwards, forwards, or with a combination of the two. In all three cases logistic stepwise returns the same 18 variables by comparing the AIC between all possible models and returns the model with the lowest AIC. As a reminder of estimating the amount of information lost by a model, AIC deals with the trade-off between the goodness of fit of the model and the simplicity of the model. In other words, AIC deals with both the risk of overfitting and the risk of underfitting. Lastly, I looked at the economic side of things. For instance, if I were running a company, what would be the most important metrics one would keep track of in order to evaluate the financial performance of the company? The most important indicators for a company's financial performance were liquidity, solvency, profitability, the bottom line, and operating efficiency. In Figure 7, we have a table of the variables selected from each of the feature selection methods.

Variance Threshold	Recursive Feature Selection	Logistic Stepwise	Economic Features
Revenue Per Share (Yuan ¥)	Non-industry income and expenditure/revenue	Persistent EPS in the Last Four Seasons	Current Ratio
Net Value Growth Rate	Interest-bearing debt interest rate	Debt ratio %	Quick Ratio
Current Ratio	Net Value Growth Rate	Net Worth Turnover Rate times	Debt Ratio %
Quick Ratio	Quick Ratio	Cash Total Assets	Operating Gross Margin
Total debt/Total net worth	Persistent EPS in the Last Four Seasons	Net Value Per Share B	Operating Profit Rate
Long-term fund suitability ratio (A)	Per Share Net profit before tax (Yuan ¥)	Cash Turnover Rate	Operating Expense Rate
Accounts Receivable Turnover	Net profit before tax/Paid-in capital	Net Value Per Share C	Net worth/Assets
Average Collection Days	Accounts Receivable Turnover	Cash Flow to Liability	Net Income to Total Assets
Fixed Assets Turnover Frequency	Net Income to Stockholder's Equity	Fixed Assets Turnover Frequency	Gross Profit to Sales
Revenue per person	Cash/Total Assets	Operating profit Paid in capital	Net Value Growth Rate
Allocation rate per person	Borrowing dependency	Accounts Receivable Turnover	
Quick Assets/Current Liability	Interest Expense Ratio	Fixed Assets to Assets	
Cash/Current Liability	Debt ratio %	Total debt Total net worth	
Inventory/Current Liability	Degree of Financial Leverage (DFL)	Cash flow rate	
Long-term Liability to Current Assets	Interest Coverage Ratio (Interest expense to EBIT)	Equity to Liability	

Figure 7: Variables Selected From Each of The Feature Selection Methods

Final feature selection is the culmination of all these methods. My criteria for the final features that would go into my machine learning models is that a variable must show up on at least 2 of the 4 columns. For example 'Quick Ratio' was found in 3 of the 4 columns so it was considered into my final model. After I had a list of these variables I made sure there was at least one variable in my final feature selection that incorporated the following economic features: liquidity, solvency, profitability, the bottom line, and operating efficiency. The final list of 8 variables with their description can be seen in figure 8.

Interest-bearing debt interest rate	interest rate from the debt accrued by the company
Quick Ratio	measures the dollar amount of liquid assets available against the dollar amount of current liabilities of a company
Net Value Growth Rate	percentage change of a company's value within a specific time period
Accounts Receivable Turnover	is an efficiency ratio that measures how efficiently a company is collecting revenue and how efficiently it is using its assets
Total debt/Total net worth	Total net worth or debt of the company
Debt ratio %	measures the amount of leverage used by a company in terms of total debt to total assets
Operating Profit Rate	Operating Profit*100/Net Sales
Persistent EPS in the Last Four Seasons	EPS measures each common share's profit allocation in relation to the company's total profit

Figure 8: Final list of 8 variables with their descriptions

Model Specification

The machine learning models I will be comparing with one another will be Logistic Regression, Random Forest, Gradient Boosting, and Xgboost. Logistic regression typically never does the best in comparison to the other methods, but it is the one most commonly used for classification projects and it sets a foundation for our performance metrics. "The random forest algorithm is an expansion of a decision tree, in that, you first construct some-axis real-world decision trees with training data, then fit your new data within one of the trees as a random forest."[5] This method averages the data and connects it to the nearest tree within the data scale. The main advantage of random forest over the decision tree is that it does not force data points into categories.

Both Xgboost and gradient boost machines follow the principle of gradient boosting. Gradient boosting consists of 3 components: a loss function, weak learner, and an additive model.[1] The loss function estimates how well a model is at making predictions with the given data. In the case of bankruptcy prediction we need a loss function that helps us understand how accurate our model is when it comes to classifying companies into bankrupt or not bankrupt. Weak learners are used as building blocks in order to make a more complicated mode.[1] Finally, our additive model component sequentially adds one weak learner at a time, then the model gets evaluated, and with each iteration we should be reducing the value of our loss function. The main difference in Xgboost and gradient boost machines are in the modeling details. Specifically, Xgboost used a more regularized model formalization to control over-fitting, which typically allows it to perform better.[2]

Sampling Techniques

With the data set we have an imbalance class problem. Over 96% of the observations are under 'Financially Stable' while a little over 3% of the observations are under 'Financially Unstable,' which can be seen in Figure 9.

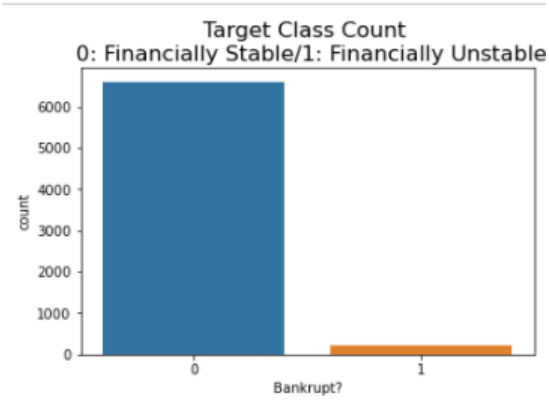

Figure 9: Bar Graph of Companies That Are Financially Stable And Not Financially Stable

The solution to this problem stems from resampling the data, and there are many ways to resample it. I decided to use 3 sampling techniques, run each of the new datasets into their own machine learning models to compare which of the 3 were best for my situation. The 3 techniques are random undersampling, random oversampling, and oversampling using SMOTE.[4] Random undersampling involves randomly selecting examples from the majority class and deleting them from the training dataset. In this case it would delete instances of companies that are 'Financially Stable.'' Random oversampling increases the size of the training data minority class through repetition of the original examples. It does not cause any increase in the variety of training examples. Figure 10 shows how the original data set will be transformed into the final data set after using oversampling and undersampling respectively.

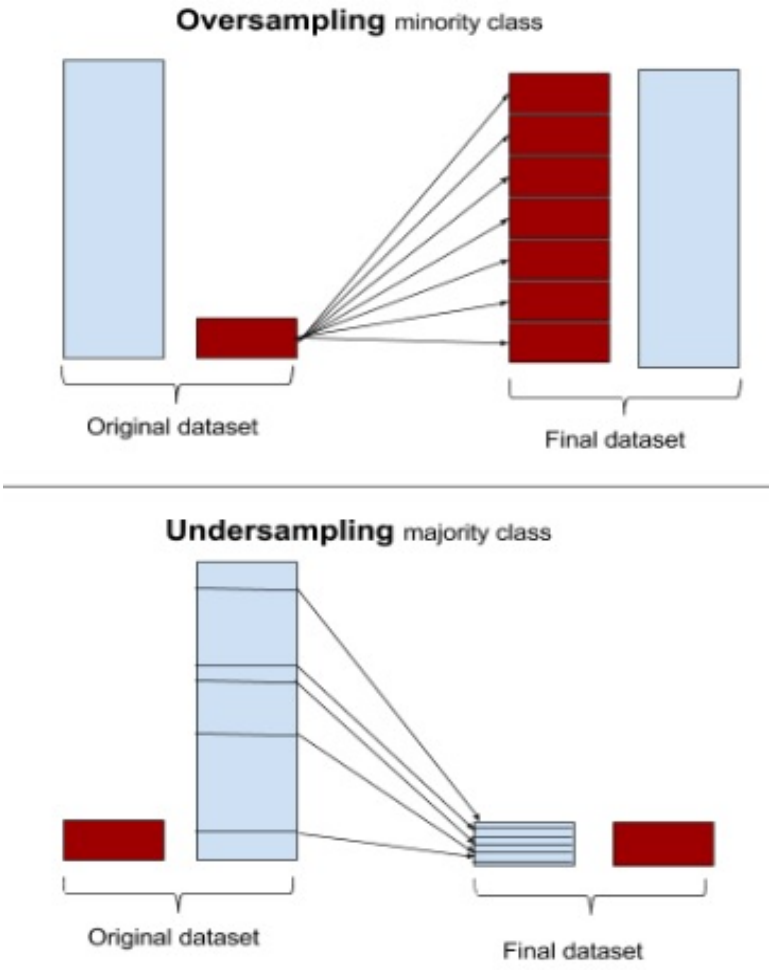

Figure 10: Changes To Data After Oversampling Or Undersampling

However, the most commonly used technique is sampling using SMOTE. This technique not only increases the size of the minority class, it also does not rely on simply replicating the data to increase its size, thus, increasing the variety of the data set. “SMOTE first selects a minority class instance at random and finds its k nearest minority class neighbors. The synthetic instance is then created by choosing one of the k nearest neighbors b at random and connecting a and b to form a line segment in the feature space. The synthetic instances are generated as a convex combination of the two chosen instances a and b.” (Page 47, Imbalanced Learning: Foundations, Algorithms, and Applications , 2013.) When done correctly SMOTE is the preferable method compared to random oversampling. Moreover, an instance where it would not be proper to use SMOTE would be on an input variable with categorical values such as “table” and “chair.” In this instance there is no ‘legal’ value in between the 2 values. However, in my data set SMOTE would be fine to use since all of the variables are numeric.

Model Diagnostics

		Precision Score	Recall Score	F1 Score	AUC
Strategy 1 Using My 8 Final Features along with the first outlier removal method aka (1.5*IQR)	Logistic - 0	0.98	1	0.99	
	Logistic - 1	0	0	0	0.931
	RF - 0	0.98	1	0.99	
	RF - 1	1	0.05	0.09	0.907
	GB - 0	0.99	0.92	0.95	
	GB - 1	0.08	0.38	0.14	0.606
	XGB - 0	0.99	0.96	0.97	
	XGB - 1	0.05	0.14	0.07	0.913
Strategy 2 Using My 8 Final Features along with the second outlier removal method aka x->log(x)	New Outlier Tech. Logistic - 0	0.97	1	0.98	
	New Outlier Tech. Logistic - 1	0	0	0	0.892
	New Outlier Tech. RF - 0	0.97	0.99	0.98	
	New Outlier Tech. RF - 1	0.37	0.15	0.22	0.887
	New Outlier Tech. GB - 0	0.97	1	0.98	
	New Outlier Tech. GB - 1	5	0.02	0.03	0.408
	New Outlier Tech. XGB - 0	0.98	0.99	0.98	
	New Outlier Tech. XGB - 1	0.48	0.3	0.37	0.904
Strategy 3 Using My 8 Final Features along with the second outlier removal method aka x->log(x) and under sampling	Under Sample Logistic - 0	0.87	0.88	0.87	
	Under Sample Logistic - 1	0.88	0.86	0.87	0.935
	Under Sample RF - 0	0.88	0.91	0.9	
	Under Sample RF - 1	0.91	0.88	0.89	0.959
	Under Sample GB - 0	0.87	0.92	0.9	
	Under Sample GB - 1	0.92	0.86	0.89	0.939
	Under Sample XGB - 0	0.87	0.92	0.9	
	Under Sample XGB - 1	0.92	0.86	0.89	0.945
Strategy 4 Using My 8 Final Features along with the second outlier removal method aka x->log(x) and over sampling	Up Sample Logistic - 0	0.83	0.84	0.84	
	Up Sample Logistic - 1	0.85	0.84	0.84	0.925
	Up Sample RF - 0	1	0.99	0.99	
	Up Sample RF - 1	0.99	1	0.99	1
	Up Sample GB - 0	0.92	0.9	0.91	
	Up Sample GB - 1	0.9	0.93	0.92	0.968
	Up Sample XGB - 0	1	0.98	0.99	
	Up Sample XGB - 1	0.98	1	0.99	1
Strategy 5 Using My 8 Final Features along with the second outlier removal method aka x->log(x) and SMOTE	SMOTE Logistic - 0	0.85	0.82	0.83	
	SMOTE Logistic - 1	0.82	0.85	0.83	0.919
	SMOTE RF - 0	0.97	0.98	0.96	
	SMOTE RF - 1	0.94	0.98	0.96	0.994
	SMOTE GB - 0	0.91	0.86	0.88	
	SMOTE GB - 1	0.86	0.91	0.89	0.951
	SMOTE XGB - 0	0.99	0.94	0.96	
	SMOTE XGB - 1	0.94	0.99	0.96	0.991

Figure 11: All Strategies With Performance Metrics

Above in Figure 11 is my table with all the different models along with each strategy that was used. I would like to note that all 5 strategies use normalization. As you can see the first two strategies did not utilize the sampling techniques I discussed previously, therefore, the precision, recall, and F1 Scores are all below 0.5. Once we get to strategies 3 and beyond all of our scores exceed 0.85. If we strictly used AUC as the metric to evaluate our performance then over sampling with random selection would be the best choice. However, the main problem with over sampling with random selection is that it just duplicates the minority class observations with no variation. That would inherently make all the models to be overfit and perform better since it already had the same observations to classify with.

Now the question comes down to whether our final table will be from the undersampling strategy verus oversampling using SMOTE. The initial problem I had with SMOTE was that logically it seemed unrealistic to create over 6000 companies that do not exist and label them as 'Financially Unstable.' However, even regarding this issue I have decided to use the SMOTE

strategy for my final model. The main disadvantage of using undersampling is that it can discard potentially useful information. SMOTE is optimal to solve our imbalance class problem if one is looking to build a predictive model, however, it should not be used when building a representative model.[4] For the purposes of this project I am indeed looking to build a predictive model. Therefore, SMOTE along with the log transformation will be the only portion of the chart that will be taken into account when looking for the optimal machine learning model.

Conclusion

Ultimately, there are 2 models from strategy 5 that performed extremely well. They were Random Forest and Xgboost. When taking into consideration recall, precision, f1, and AUC I decided that the Xgboost model will be the one that represents my final predictive model. From Figure 12, we can directly compare the metrics between Random Forest and Xgboost. Although Random Forest had an AUC of 0.003 greater while also having a higher recall score when predicting the 0's. The Xgboost model had a recall score of 0.01 greater when predicting the 1's, which is more important for the purposes of predicting bankruptcy.

	Precision Score	Recall Score	F1 Score	AUC
SMOTE RF - 0	0.97	0.98	0.96	
SMOTE RF - 1	0.94	0.98	0.96	**0.994**
SMOTE XGB - 0	0.99	0.94	0.96	
SMOTE XGB - 1	0.94	0.99	0.96	**0.991**

Figure 12: Random Forest And Xgboost Performance Metrics

Even though this project achieved great accuracy with a small subset of predictors there are limitations regarding this project. The primary issue stems from not knowing which industry the companies are gathered from. For instance, I do not know whether all the companies are all software, finance, healthcare, retail, or a mix of all of them. I predict that if my models were trained with the 'software industry companies' then applying models from other industries will result in a reduction in performance. Additionally, my project strictly predicts bankruptcy, it does not take into account other company's financial states. For example, a company does not always need to result in bankruptcy, they could fall under debt settlement, debt consolidation, restructure, etc[6]. In order to avoid future errors when applying models such as mine, it is mandatory for researchers to understand the purposes of models like mine as well as their limitations.

References

1. Breheny, Patrick. “Transformations and Outliers.” *Introduction Transformations Outliers Summary*, 17AD, myweb.uiowa.edu/pbreheny/4120/s14/notes/4-17.pdf.

2. Brownlee, Jason. “A Gentle Introduction to the Gradient Boosting Algorithm for Machine Learning.” *Machine Learning Mastery*, 14 Aug. 2020, “How to Calculate Precision, Recall, and F-Measure for Imbalanced Classification.” *Machine Learning Mastery*, 1 Aug. 2020, machinelearningmastery.com/precision-recall-and-f-measure-for-imbalanced-classification/.

3. machinelearningmastery.com/gentle-introduction-gradient-boosting-algorithm-machine-learning

4. Fedesoriano. “Company Bankruptcy Prediction.” *Kaggle*, 13 Feb. 2021, www.kaggle.com/fedesoriano/company-bankruptcy-prediction.

5. He, Haibo, and Yunqian Ma. *Imbalanced Learning: Foundations, Algorithms, and Applications*. John Wiley & Sons, 2013. (Pages 47-50)

6. Kaushik, Saurav. “Feature Selection Methods: Machine Learning.” *Analytics Vidhya*, 19 Oct. 2020, www.analyticsvidhya.com/blog/2016/12/introduction-to-feature-selection-methods-with-an-example-or-how-to-select-the-right-variables/#:~:text=It%20enables%20the%20machine%20learning,It%20reduces%20overfitting.

7. Kassambara, and Barazini. “Stepwise Logistic Regression Essentials in R.” *STHDA*, 11 Mar. 2018, www.sthda.com/english/articles/36-classification-methods-essentials/150-stepwise-logistic-regression-essentials-in-r/.

8. Kriegel, Hans Peter. “Outlier Detection Techniques.” *The 2010 SIAM International Conference on Data Mining*, 2010, archive.siam.org/meetings/sdm10/tutorial3.pdf.

9. Liang, D., Lu, C.-C., Tsai, C.-F., and Shih, G.-A. (2016) Financial Ratios and Corporate Governance Indicators in Bankruptcy Prediction: A Comprehensive Study. European Journal of Operational Research, vol. 252, no. 2, pp. 561-572.

10. Mavavilj. “How to Decide What Threshold to Use for Removing Low-Variance Features?” *Data Science Stack Exchange*, 1 Feb. 1967, datascience.stackexchange.com/questions/31453/how-to-decide-what-threshold-to-use-for-removing-low-variance-features.

11. Maverick, J.B. “What Is the Best Measure of a Company's Financial Health?” *Investopedia*, Investopedia, 12 Apr. 2021, www.investopedia.com/articles/investing/061916/what-best-measure-companys-financial-health.asp.

12. “NearZeroVar: Identification of near Zero Variance Predictors.” *RDocumentation*, www.rdocumentation.org/packages/caret/versions/6.0-86/topics/nearZeroVar.

13. Scikit-learn: Machine Learning in Python, Pedregosa *et al.*, JMLR 12, pp. 2825-2830, 2011.

14. T., Bex. “How to Use Variance Thresholding For Robust Feature Selection.” *Medium*, Towards Data Science, 11 Apr. 2021, towardsdatascience.com/how-to-use-variance-thresholding-for-robust-feature-selection-a4503f2b5c3f.

15. Team, Towards AI. “How, When, and Why Should You Normalize / Standardize / Rescale Your Data?” *Towards AI - The Best of Tech, Science, and Engineering*, 29 May 2020, towardsai.net/p/data-science/how-when-and-why-should-you-normalize-standardize-rescale-your-data-3f083def38ff#:~:text=Normalization%3A,when%20features%20have%20different%20ranges.