CIS 520 Fall 2019, Project Proposal

Identifying Predictive Features of Business Failure

Team Members:

- Chuning Zhu; zchuning@seas.upenn.edu
- Takanao Ishimura; takanao@seas.upenn.edu
- Daniel Tao; dtao@seas.upenn.edu

---

1 Abstract

Predicting the success of a business is a problem of importance for investors, business owners, employees, and customers. In this project, we employ KNN, Random Forest, Logistic Regression, SVM, various Boosting Methods, and a Multilayer Perceptron to predict the future status of a business given key financial metrics. We identify Gradient Tree Boosting as the best general algorithm for this task. Additionally, we explore methods of handling imbalanced data sets.

2 Motivation

Anyone who has taken an accounting class knows how difficult understanding and extracting actionable insights from the glut of available financial data can be. Though companies release financial data on a regular basis, whether yearly or quarterly, it is still a challenge to sift through that financial data in order to act on it. In the past, failure to recognize the warning signs of bankruptcy or general failure as a business has put the livelihoods of countless people at risk, as when a business goes under, those employed by the business and the investors in that business are put at great risk. The denseness of financial information most companies put out can even be used to obscure behavior, as seen in the failure of the fraudulent energy company Enron, which destroyed billions of dollars of value and put thousands of employees out of their jobs. With the advent of modern VC and the stupendous amounts of money exchanging hands in hotspots such as Silicon Valley, more and more people have come to rely on the existence of companies such as Lyft and Uber despite their dubious future prospects. In exploring this dataset consisting of the financial metrics of companies and whether they failed or not in the following years, we hope to better understand what financial metrics are the most important in predicting the future prospects of a business and also to train a model which predicts whether a business will survive in the long run. The results of such work could greatly benefit investors, consumers, and employees of companies which could be at risk for bankruptcy.

3 Related Work

Past work in training models to predict whether a company will go bankrupt have applied numerous techniques from subfields of Machine Learning ranging from NLP to Deep Learning to improve the model's predictive power with varying degrees of success. Researchers have applied models which have traditionally seen use in classification problems ranging from SVM to neural nets with dropout [6]. Others have applied random forests and logistic regressions to the problem [1]. Some have even applied NLP to the official statements of a company and combined that with more traditional financial metrics data models to increase their prediction accuracy [5]! Despite the amount of work being done in this field, there is still much debate over which types of models are best suited to the problem, or whether such models are improvements over existing statistical methods of prediction or models with expert knowledge at all.

4 Data Set

The dataset we investigate contains bankruptcy records of various Polish companies. Using the different financial metrics offered in the dataset, we approach this as a classification problem, predicting whether companies will succeed or fail. The full dataset consists of 64 financial metrics over 10K different companies, of which only 3.9% end up filing for bankruptcy. These metrics are given once per year over a 5 year period. For all our analyses (except for PCA) we focus on the partial dataset containing only the 5th year's data, which are presumably the most predictive as companies go bankrupt in only one year, as opposed to 2nd year's data where companies don't go bankrupt until four years later. A detailed description of the features is given below in Figure 1.

ID	Description	ID	Description
X1	net profit / total assets	X33	operating expenses / short-term liabilities
X2	total liabilities / total assets	X34	operating expenses / total liabilities
X3	working capital / total assets	X35	profit on sales / total assets
X4	current assets / short-term liabilities	X36	total sales / total assets
X5	[(cash + short-term securities + receivables - short-term liabilities) / (operating expenses - depreciation)] * 365,	X37	(current assets - inventories) / long-term liabilities
X6	retained earnings / total assets	X38	constant capital / total assets
X7	EBIT / total assets	X39	profit on sales / sales
X8	book value of equity / total liabilities	X40	(current assets - inventory - receivables) / short-term liabilities
X9	sales / total assets	X41	total liabilities / ((profit on operating activities + depreciation) * (12/365))
X10	equity / total assets	X42	profit on operating activities / sales
X11	(gross profit + extraordinary items + financial expenses) / total assets	X43	rotation receivables + inventory turnover in days
X12	gross profit / short-term liabilities	X44	(receivables * 365) / sales
X13	(gross profit + depreciation) / sales	X45	net profit / inventory
X14	(gross profit + interest) / total assets	X46	(current assets - inventory) / short-term liabilities
X15	(total liabilities * 365) / (gross profit + depreciation)	X47	(inventory * 365) / cost of products sold
X16	(gross profit + depreciation) / total liabilities	X48	EBITDA (profit on operating activities - depreciation) / total assets
X17	total assets / total liabilities	X49	EBITDA (profit on operating activities - depreciation) / sales
X18	gross profit / total assets	X50	current assets / total liabilities
X19	gross profit / sales	X51	short-term liabilities / total assets
X20	(inventory * 365) / sales	X52	(short-term liabilities * 365) / cost of products sold)
X21	sales (n) / sales (n-1)	X53	equity / fixed assets
X22	profit on operating activities / total assets	X54	constant capital / fixed assets
X23	net profit / sales	X55	working capital
X24	gross profit (in 3 years) / total assets	X56	(sales - cost of products sold) / sales
X25	(equity - share capital) / total assets	X57	(current assets - inventory - short-term liabilities) / (sales - gross profit - depreciation)
X26	(net profit + depreciation) / total liabilities	X58	total costs /total sales
X27	profit on operating activities / financial expenses	X59	long-term liabilities / equity
X28	working capital / fixed assets	X60	sales / inventory
X29	logarithm of total assets	X61	sales / receivables
X30	(total liabilities - cash) / sales	X62	(short-term liabilities *365) / sales
X31	(gross profit + interest) / sales	X63	sales / short-term liabilities
X32	(current liabilities * 365) / cost of products sold	X64	sales / fixed assets

Figure 1: Feature Description

4.1 Principal Component Analysis

As a preliminary exploration, we run PCA on the full dataset and plot the points in the principal component space, where projections of the first and second largest components are represented on the x-axis and y-axis respectively (Figure 2). The first and second components capture 20.5% and 14.4% of the total variance in the dataset respectively. We color the points by y-classification, where blue points represent companies that do not go bankrupt whereas green points represent companies that do. Furthermore, we overlay the factor loadings of the first and second principal components and label features that have a loading magnitude over a specified threshold (0.3). We can observe that at lower dimensionality, the two classes are not linearly separable, meaning that it is necessary for further analysis to be conducted on all features. However, given more time, we would use the factor loadings to identify the most informative features in the dataset. Considering that more of the variance between the two classes is captured by the first principal component, these identified features are more likely to be ones that have greater magnitude in the first component, such as X24, X30, and X31.

5 Problem Formulation

Our main goal is to predict whether a business will go bankrupt from a set of features as shown in Figure 1, including but not limited to sales, gross profit, liabilities, assets, and working capital. Additionally, we hope to gain a better understanding of various classification methods by attempting to explain the differences in performance on this imbalanced dataset.

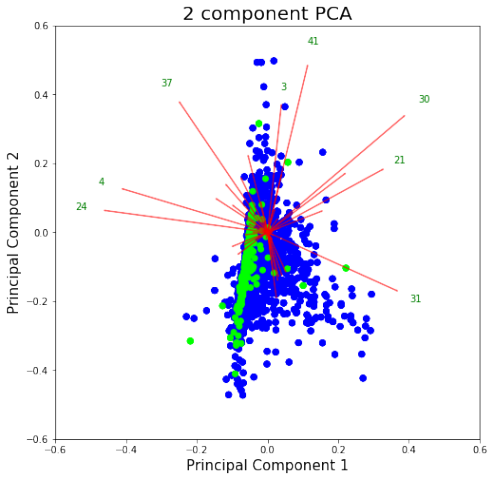

Figure 2: Principal Component Analysis

6 Methods

We first impute the missing data using regression imputation with 2 epochs. Then we split our dataset 70/30, using the 70 percent as our training set and the 30 percent as our test set. For our baseline, we train a logistic regression model without class weights on the full training set. Since the dataset is quite imbalanced (positive:negative = 1:15), we expect to observe high accuracy and low precision/recall (i.e. error disparity) in our baseline model. We proceed to balance the dataset via two methods: undersampling the majority class and adjusting class weights. We choose one of these two methods to process our dataset with by training our baseline model on data processed by each method, and comparing the performance of the resulting models on our testing set. Then, we run multiple classification models on our processed data set and compare their performance. We constrain the scope of our models to contain only discriminative models, including KNN, random forest, logistic regression, SVM, boosting methods, and multi-layer perceptron (neural net). While working with these models, we explore bias-variance and precision-recall tradeoffs by tuning model hyperparameters. Finally, we do qualitative analysis on the features that are most predictive of bankruptcy by interpreting the weights of the logistic regression classifier and the loadings of the first two principle components.

7 Experiments and Results

7.1 Baseline Model

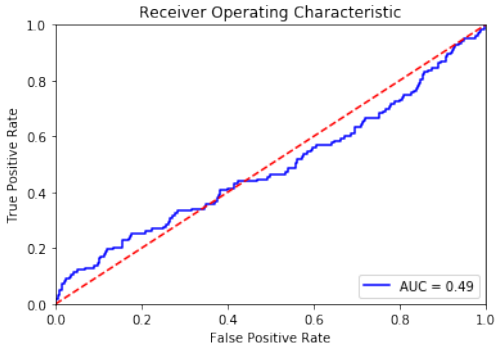

Figure 3: Baseline Logistic Regression ROC

Accuracy: 0.93, Precision: 0.29, Recall: 0.03, F1: 0.06

We use vanilla logistic regression with L2 regularization as our baseline model. From the ROC curve we can see that the baseline model performs slightly worse than random. It suffers from error disparity, as manifested in the extreme low recall, which means most true positive samples are classified as negative.

7.2 Balancing the Dataset

As expected, our baseline model has extremely low precision and recall. In order to mitigate the inbalance in the dataset that appears to be causing this problem, we attempt two different approaches: undersampling the majority class weight or adjusting class weights.

7.2.1 Undersampling

In order to shift the positive to negative ratio of our dataset from 1:15 to 1:1, we randomly undersample negative data points to include in our training and testing sets. However, because the number of positive samples (~ 400) is relatively small, we find that the characteristics of resulting models are extremely sensitive to which negative samples are randomly selected. This could have been a more viable solution to this problem if the dataset were much larger and the number of positive samples were an order of magnitude greater.

7.2.2 Adjusting Class Weights

Given that the ratio of positive to negative samples in our dataset is 1:15, we weight the positive class 15 times more than the negative class. This approach is much more stable than the undersampling method given above, and hence we will use this class weighting rule, whenever applicable, prior to running our discriminatory models below.

7.3 Discriminative Models

7.3.1 Overall Results

Model	Accuracy	Precision	Recall	F1
K-Nearest Neighbors (K=5)	1.00	1.00	1.00	1.00
Random-Forest (150 Estimators)	0.95	0.58	0.92	0.71
Logistic Regression	0.68	0.16	0.84	0.27
SVM	0.80	0.23	0.78	0.35
AdaBoost (100 Estimators)	0.97	0.91	0.60	0.72
Gradient Tree Boosting (200 Estimators)	0.99	1.00	0.91	0.95
Extreme Gradient Tree Boosting (100 Estimators)	0.98	0.99	0.74	0.85
Multilayer Perceptron	0.78	0.22	0.82	0.34

Figure 4: Evaluation on Training Set

Model	Accuracy	Precision	Recall	F1	AUC
K-Nearest Neighbors (K=5)	0.93	0.59	0.14	0.22	0.77
Random-Forest (150 Estimators)	0.95	0.45	0.63	0.53	0.91
Logistic Regression	0.72	0.16	0.73	0.26	0.79
SVM	0.80	0.19	0.59	0.29	0.78
AdaBoost (100 Estimators)	0.96	0.87	0.54	0.66	0.92
Gradient Tree Boosting (200 Estimators)	0.96	0.81	0.54	0.65	0.94
Extreme Gradient Tree Boosting (100 Estimators)	0.96	0.89	0.51	0.65	0.95
Multilayer Perceptron	0.77	0.18	0.67	0.29	0.77

Figure 5: Evaluation on Testing Set

Note that due to the imbalanced nature of our dataset, the testing accuracy is high across the board. Thus in the analysis that follows we are going to focus predominantly on precision and recall. We are also interested in combined metrics such as F1 score and Area Under Curve (AUC) of the ROC curve.

7.3.2 K-Nearest Neighbors

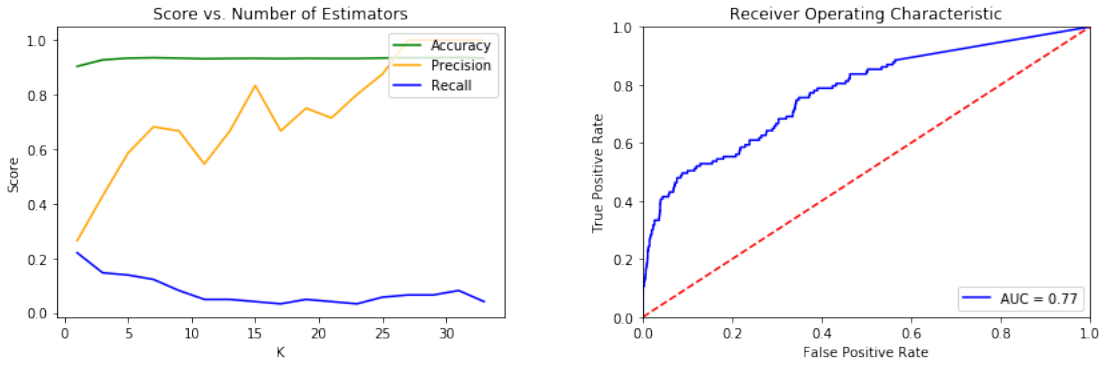

Figure 6: Left: KNN Learning Curve | Right : KNN ROC

We observe that KNN suffers from the imbalance of positive to negative samples. Figure 6 shows that as k increases, the recall plummets to near zero, as the model tends toward a majority vote system. Since negative

samples in this dataset far outnumber positive samples, the higher values of k would lead to a model that exclusively predicts 'non-bankrupcy'. However, limiting k to smaller values leads to severe overfitting. Given more time, we could alter the distance metric used in this model to assign more weight to positive samples in order to circumvent this problem. However, at this stage, we do not believe KNN is good predictive model for this particular dataset.

7.3.3 Random Forest

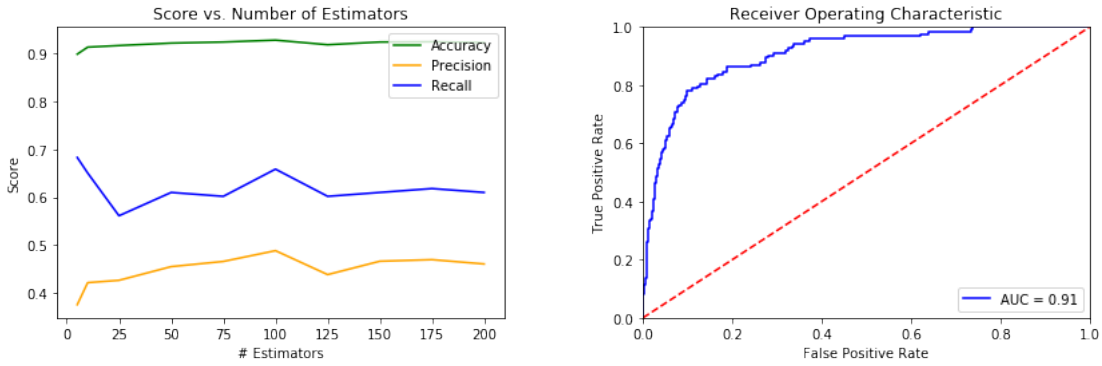

Figure 7: Left: Random Forest Learning Curve | Right : Random Forest ROC

Using class weights defined above, we observe that the Random Forest classification performs very well compared to the baseline model and KNN. Figure 7 shows that above 25 estimators in the ensemble, the accuracy, precision, and recall change only marginally. We conclude that Random Forest performs best on this dataset with 150 estimators. The evaluation metrics of the model with 150 estimators are given in Figure 5.

7.3.4 Logistic Regression

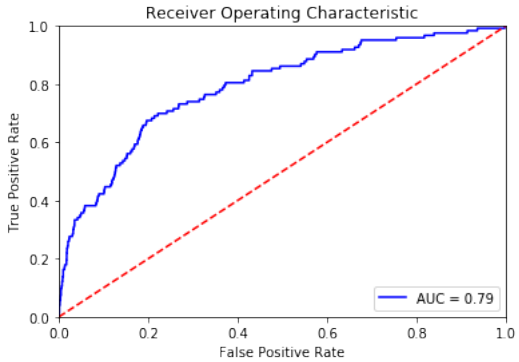

Figure 8: Logistic Regression ROC

Using class weights defined above, we observe that Logistic Regression achieves the worst accuracy among all models, as shown in Figure 4. However, we see a statistically significant improvement from the baseline logistic model that did not use the class weighting scheme, which stresses the importance of our data balancing stategy.

To explore the effect of regression penalties on the results, we train three different classifiers with l2, l1, and elastic net regularization. Each model is trained with 5-fold cross-validation. However, no significant difference is observed. We conclude that overfitting is well-controlled with l2 regularization.

7.3.5 SVM

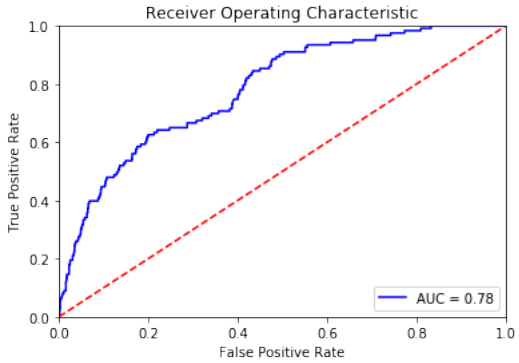

Figure 9: SVM ROC

SVM with an rbf kernel achieves a relatively high recall score but a suffers low precision. Since SVM is reliant on the relative magnitudes of the features, we normalize the dataset when running SVM. In fact, running SVM on the un-normalized dataset yields a model that is comparable in performance to our poor baseline model.

7.3.6 Boosting

7.3.6.1 AdaBoost

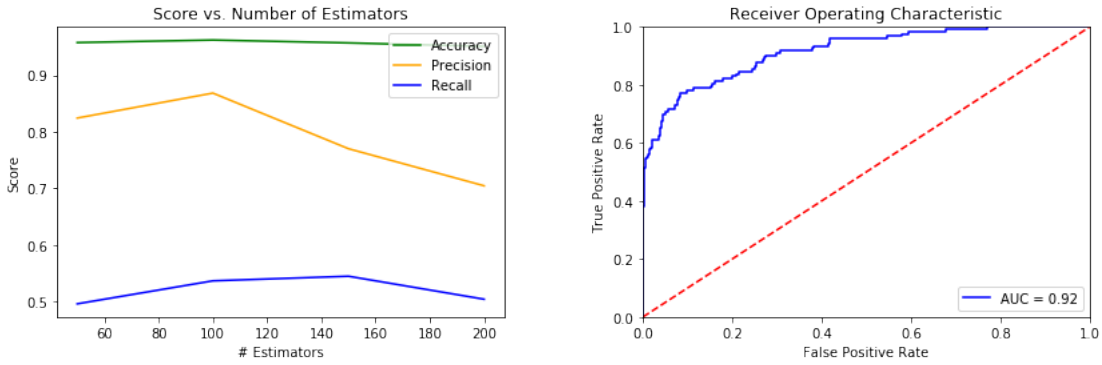

Figure 10: Left: AdaBoost Learning Curve | Right : AdaBoost ROC

As shown in Figure 5 and 10, Adaboost achieves an overall excellent result on the test set. Note that the algorithm naturally balances the dataset by assigning weights for each sample and adjusting the weights of misclassified samples each iteration. Since the minority-class samples are frequently misclassified, their weights will increase multiplicatively.

Besides the default base learner (decision stump), we also experiment with a variety of base learners types for the Adaboost algorithm, including logistic regression and perceptrons. However, all of them display severe overfitting, so we decide to stick with using decision trees as the base classifiers to control the complexity of our model. We then examine the precison-recall tradeoff by adjusting the max depth of the base classifier. As can be seen in Figure 11 below, the models with higher precision tend to have lower accuracy, and vice versa. The optimal model is obtained when the max depth of the decision tree is 3, after which the model begins to overfit as the recall approaches 0.

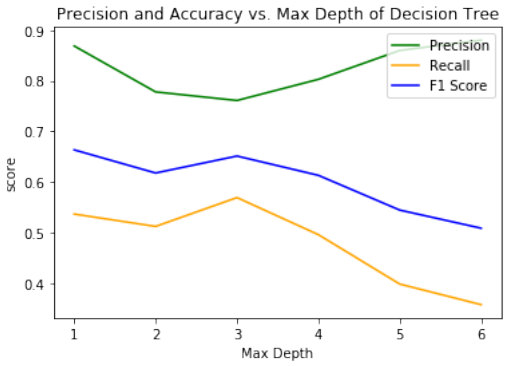

Figure 11: AdaBoost Scores vs. Max Depth

7.3.6.2 Gradient Tree Boosting

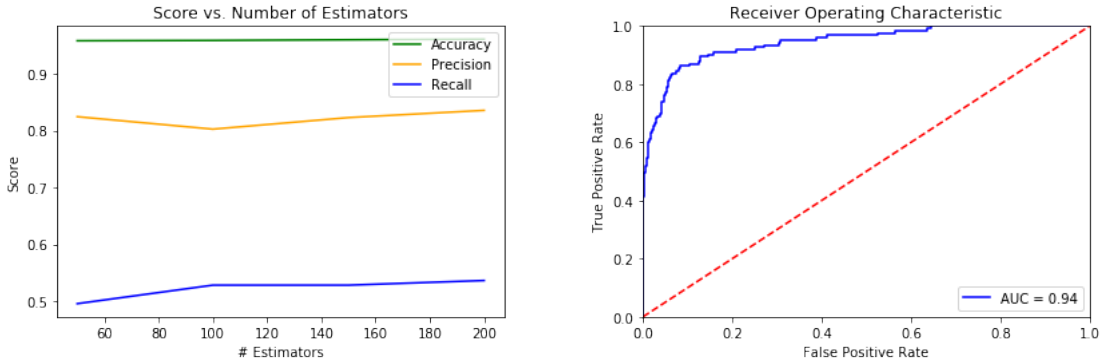

Figure 12: Left: Gradient Tree Boosting Learning Curve | Right : Gradient Tree Boosting ROC

Unlike Adaboost which trains a new classifier to fit the data with individually adjusted weights each round, Gradient Tree Boosting[3] grows a new tree on the residual of the previous classifier. Formally, a gradient boosting classifier attempts to optimize the following objective function in the tth round:

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t-1)} + f_t\left(x_i\right)\right) + \Omega\left(f_t\right) + \text{constant}$$

If we use MSE for l and take the second order Taylor expansion to approximate the objective, the objective can be simplified to

$$\tilde{\mathcal{L}}^{(t)} = \sum_{i=1}^{n} \left[g_i f_t\left(\mathbf{x}_i\right) + \frac{1}{2} h_i f_t^2\left(\mathbf{x}_i\right)\right] + \Omega\left(f_t\right)$$

where

$$g_i = \partial_{\hat{y}^{(t-1)}} l\left(y_i, \hat{y}^{(t-1)}\right)$$

$$h_i = \partial^2_{\hat{y}^{(t-1)}} l\left(y_i, \hat{y}^{(t-1)}\right)$$

As shown in Figure 5 and 12, Gradient Boosting is on par with Adaboost in terms of testing recall and F1 score, but it slightly outperforms Adaboost in the AUC metric.

7.3.6.3 Extreme Tree Boosting

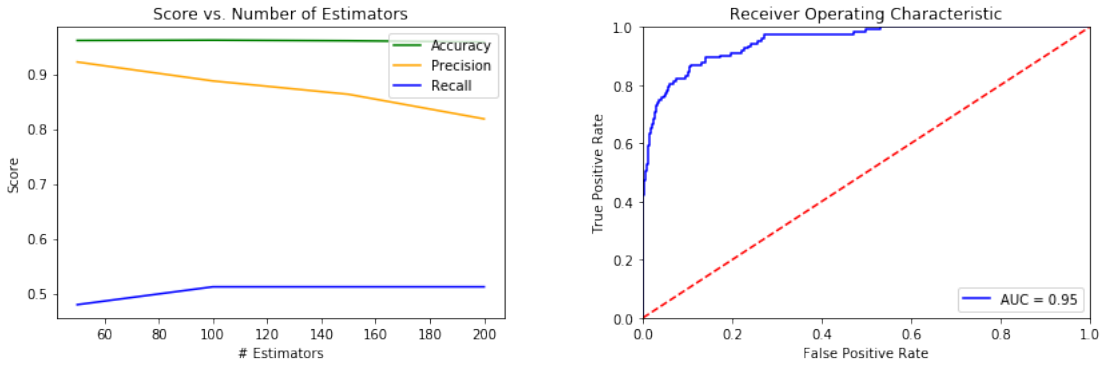

Figure 13: Left: Extreme Tree Boosting Learning Curve | Right : Extreme Tree Boosting ROC

The Extreme Gradient Boosting method (XGBoost), proposed by Tianqi Chen et al.[2][4], is an optimization of the gradient boosting algorithm which introduces more regularization to the objective function. From Figure 4 we can see that the training accuracy, precision, and recall of XGBoost are lower than those of Gradient Boosting, which means XGBoost is less prone to outfitting than Gradient Boosting. In the test set, XGBoost outperforms Gradient Boosting on every metric except recall.

7.3.7 Multilayer Perceptron

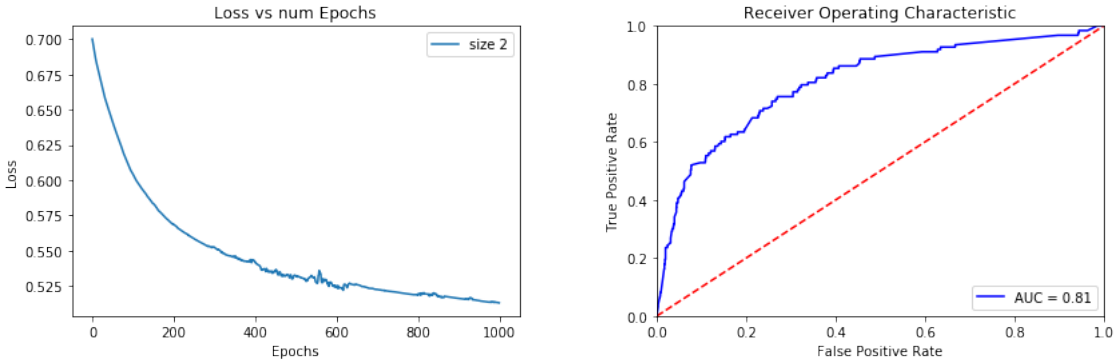

Figure 14: Left: Multilayer Perceptron Learning Curve | Right : Multilayer Perceptron ROC

The architecture of our neural network consists of one input layer, two hidden layers, and one output layer. The two hidden layers are made of 16 and 8 nodes respectively, and the output is a two-node binary classifier. We use cross-entropy loss as our loss function and ADAM as our optimizer. Our multi-layer perceptron performs relatively well but fails to beat the evaluation scores yielded by our boosting algorithms, as shown in Figure 5.

8 Conclusion and Discussion

8.1 Predictive Features

To elucidate the predictive power of a specific feature, we examine the magnitude of its corresponding weight in our Logistic Regression. A histogram separating the data into 36 evenly spaced bins is shown below (not

including one feature X21, which has a magnitude of 291.7 and was removed from the distribution to allow the histogram to display the distribution in much greater detail). Most features in our dataset have a corresponding coefficient of less than 30, with most being less than 10. Features X21 and X1 have high coefficients over 30; we will discuss them in greater detail. Although features X48 and X22 also have coefficient magnitudes over 30, we will not discuss them as they are highly colinear to X1. If we had more time, we would do feature selection to reduce multicolinearity. Additionally, we choose to examine some features which contribute to Principal Component 1 specifically because bankrupt companies tend to cluster together along a specific range of the axis delineated by PC 1 as shown in Figure 2. We will examine features X4 and X31 as they are among the features which contribute the most to PC 1 and had relatively high coefficients in our Logistic Regression model's weights.

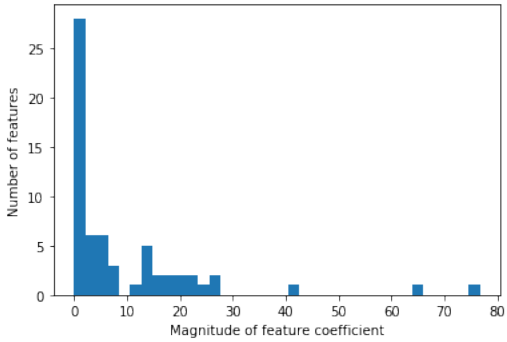

Figure 15: Features by Coefficient Magnitude

8.1.1 Feature X21: sales(n)/sales(n-1)

This feature coefficient has the highest magnitude out of all features by far with a value of -291.7, meaning that a business with a high value for this feature is unlikely to go bankrupt in a year. This feature is large when the annual sales of a business show great improvement from year n-1 to year n, which suggests that businesses showing improvement in their sales are unlikely to go bankrupt. Conversely, a business which sees its sales drop from year n-1 to year n is more likely to go bankrupt. While sales(n) and sales(n-1) are unlikely to be predictive separately since different businesses can operate successfully at different scales, the sales(n-1) can be used to contextualize sales(n), providing a highly predictive business performance ratio which captures whether the success of a business is trending upward or downward with regards to time.

8.1.2 Feature X1: net profit/total assets

This feature has a corresponding weight of -76.7, implying that when a business earns a high net profit relative to its total assets, it is unlikely to go bankrupt. While one might intuitively expect that the ratio between net profit and annual operating expenses would be more predictive of business success, this finding is intuitive considering the difference between fixed and variable costs. Annual operating expenses, especially for businesses at scale, are likely composed of mostly variable costs which scale with the gross income that a business makes. Total assets are a strong proxy for fixed costs, as fixed costs often stem from upkeep costs of business assets. Once a business is in operation, it is unlikely that the total variable costs would predict its success as it is unlikely that a business would produce a product which costs more to make than sell. However, a business with high fixed costs relative to its profit is inherently expensive to operate, especially at a small scale. In an economic downturn where consumer spending decreases, a business might find itself not moving enough volume to pay fixed costs that it would otherwise be able to afford had it sold more of its product or service. Hence, businesses with a high net profit to total asset ratio are less sensitive to economic downturns and less likely to go bankrupt. Features X48 (net operating profits - depreciation / total assets) and X22 (net operating profits/ total assets) also have magnitudes over 30, likely due to the fact that net operating profits make up most of net profits for most businesses. Hence, features X48, X22, and X1 are strongly correlated.

8.1.3 Feature X4: current assets/short-term liabilities

This feature has a corresponding weight of -13.6, meaning that when this ratio is high a business is less likely to go bankrupt. Current assets consist of assets that are highly liquid in that they are expected to be converted to cash within one year. Short-term liabilities, on the other hand, are liabilities that are due within one year. Consider what it means for this ratio to be high: a business has many assets that it expects to be able to convert into cash or other liquid assets and hence can be used to reinvest into new assets, and relatively few liabilities to other organizations. Conversely, when this ratio is low, a business is strapped for sources of capital and has taken out many liabilities to other organizations which expire within a year. This is an undeniably dire operating situation. When a business finds itself with very little cash on hand and many liabilities to be paid, the business may find itself declaring bankruptcy to fulfill and absolve itself of debts that it can no longer handle.

8.1.4 Feature X31: (gross profit + interest)/sales

This feature has a corresponding weight of 11.8, meaning that when it is high a business is more likely to go bankrupt. Initially this is confusing, as when gross profit is high compared to sales, a business has good margins, which conventional business theory covets as a predictor of success. However, since gross profit is sales minus operating costs, the ratio of gross profit to sales is capped at 1. For this ratio to be higher than 1, a business has to receive a disproportionate amount of its incoming cash from interest instead of its actual sales. This is indicative of a situation where a business owes most of its success not from the strength of its business model but instead through financial dealings. Such businesses are unlikely to be successful in the long run once these financial deals expire.

8.2 Classification Model Recommendation

To conclude, the classification model we recommend for this dataset depends on what the user prioritizes as important. If the user decides the correct prediction for companies that are truly going bankrupt is most important, then recall becomes the trumping evaluation metric and Logistic Regression may be the most applicable model for the user. If the user seeks good precision in the model, then Extreme Gradient Boosting is most likely the best choice. Otherwise, if the user is seeking an overall robust model, then we recommend the Gradient Tree Boosting Algorithm, which scores highly across all evaluation metrics.

References

[1] Flavio Barboza, Herbert Kimura, and Edward Altman. Machine learning models and bankruptcy prediction. *Expert Systems with Applications*, 83:405 – 417, 2017.

[2] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794, New York, NY, USA, 2016. ACM.

[3] Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5):1189–1232, 2001.

[4] Dimitris Leventis. Xgboost mathematics explained, Jul 2019.

[5] Feng Mai, Shaonan Tian, Chihoon Lee, and Ling Ma. Deep learning models for bankruptcy prediction using textual disclosures. *European Journal of Operational Research*, 274(2):743 – 758, 2019.

[6] Nanxi Wang. Bankruptcy prediction using machine learning. 2017.