Evaluating Decision Tree Ensemble Methods in Predicting Corporate Bankruptcy

Jay Lee
Department of Computer Science
Princeton University
`junghwan@princeton.edu`

Abstract

Corporate bankruptcy brings many problems: lenders fail to receive their promised pay, employees are laid off, and in the extremist cases, can affect the macro-economy as a whole. This paper explores the predictive capabilities of using fundamental financial data to forecast whether firm will file for bankruptcy in the future. Past studies have used various different methods for this problems, most commonly discriminant analysis, logistic regression, support vector machines, and artificial neural networks. We focus on two key tree-based ensemble methods: random forest and gradient boosting using the XGBoost library. We find that both of these methods far outperform traditional methods in both AUC and Precision-Recall. We also note that while using principle component analysis for dimensionality reduction improves our training time substantially, the performance of our models also decrease.

1 Introduction

When a corporation is unable to repay the debt they received from creditors, they file for bankruptcy. Bankruptcy brings with it many economic and social problems: lenders fail to receive their promised pay, and employees are laid off, to name a few. Therefore, being able to predict bankruptcy ahead of time would be a valuable tool for investors, regulators, and firms alike.

Many reasons can contribute to bankruptcy: obvious factors such as poor management, corruption, and unfavorable market conditions can all impact the firm's performance, and more subtle factors such as financing (how much debt does the company have versus equity) can also increase risk of financial distress [6].

One of the most common methods in which analysts assess the performance of firms is through *financial data*. While this is a loose term, the most common types of financial data are drawn from company income statements (information on cost and sales), balance sheets (information on company's assets and liabilities), and cash flow statements (details on cash flow). This data is often represented as financial ratios, conveniently allowing analysts express detailed information about the firm. We are interested in the predictive capabilities of these fundamental financial ratios in predicting corporate bankruptcy.

2 Related Work

The first published study on this topic to attract attention was published by Edward Altman in 1968 [1] in which he used discriminant analysis to classify companies as solvent (healthy) or insolvent (danger of bankruptcy) by using five key financial ratios. While the original study was simple, it inspired many future works. In 1980, Ohlson proposed a logistic regression models to determine the probability of a firm entering bankruptcy in the near future [5], Min and Lee demonstrated that a Support Vector Machine (SVM) can achieve accuracy of over 90% with the necessary feature selection and hyperparameter tuning methods, and in the results of a 1997 study by Pompe and Feelders, Artificial Neural Networks (ANN) have also been shown as particular good solvers for this problem.

More recently, backpropogated ANNs and different ensemble methods have shown to be most successful in this field [2]. Wang et al. and Barboza et al. both demonstrated that ensemble methods such as bagging and boosting significantly outperform traditional methods when applied carefully [2].

In essence, various different methods have been applied to various different data sets in this problem space. Many studies examined companies in the U.S. and Canada, while many others focused on companies in the European Union or Asia. Some studies work with very few attributes per company (five, as in Altman's original case), while others used about twenty five (for example, by Barboza et al.).

Our goal in this paper is to assess the performance of two decision tree based ensemble classifiers in particular: random forest and gradient boosting. As with Altman's original paper and many others in this field, we will make sure our data set is balanced between firms that go bankrupt and those that do not, since the significant imbalance in reality can introduce bias into our models.

3 Approach

3.1 Data set

We use the Polish companies bankruptcy data set generously provided by the UC Irvine Machine Learning Repository [7]. The data we use contains 15702 rows and 65 columns, where each row represents a company that was examined in some time between 2000 and 2013. Each row contains 64 features, each representing a specific financial ratio (see Appendix A), and an output label of either 1 (the firm filed for bankruptcy within the next two years) or 0 (the firm did not file for bankruptcy in the next two years).

Each feature represents a commonly used financial ratio that describes either the company's financial structure, income, or cash flow in that given year. For instance, column 2 is the ratio of liabilities to assets (i.e. how much debt does the company have compared to its equity and cash), and column 1, net profit to total assets, measures the profit the company made that year relative to their total asset value. Every one of the 64 features is therefore a real-numbered value.

3.2 Data Pre-processing

3.2.1 Imputing

The data contained missing data across different rows and columns. In particular, we decided to drop column 37 (current assets - inventories divided by long-term liabilities), since over 45% of our total data set had a missing value, and the remaining rows had considerably large variance between them. In every other case, we simply cared for missing data by imputing its value with the mean across all rows.

3.2.2 Oversampling

Of the 15702 rows, only 925 companies actually filed for bankruptcy (about 5.9%). We decided to balance the classes because not only do many of the studies in past literature also only consider balanced training sets, but also to make sure the bias does not negatively impact our models (for instance, a SVM simply predicting '0' for every company, which would be meaningless despite the 94% accuracy).

Before balancing the classes, we first randomly divided our data into a 75% training set and a 25% test set, which was withheld and used to evaluate our performance. Then, we used the Synthetic Minority-Class Oversampling Technique (SMOTE) to oversample our minority class in the training set. SMOTE is a technique in which we artificially create new samples in the training data such that one class label (not bankrupt) does not overwhelmingly dominate the other (bankrupt) by count.

SMOTE picks two samples (A and B) in the minority class at random, and then takes the difference of their feature vectors (B - A). This difference vector is then multiplied by a real value chosen uniformly at random in $[0, 1]$, then added to one of the original point (A) to produce a new, synthetic sample. A visual representation in two dimensions can be seen in Figure 1. We used SMOTE to balance the classes to 1:1, so in the final training set, we ended up with 11075 non-bankrupt samples from the original data, 701 non-synthetic bankrupt samples, and 10374 synthetic bankrupt samples. We used and followed the demonstration found here [3].

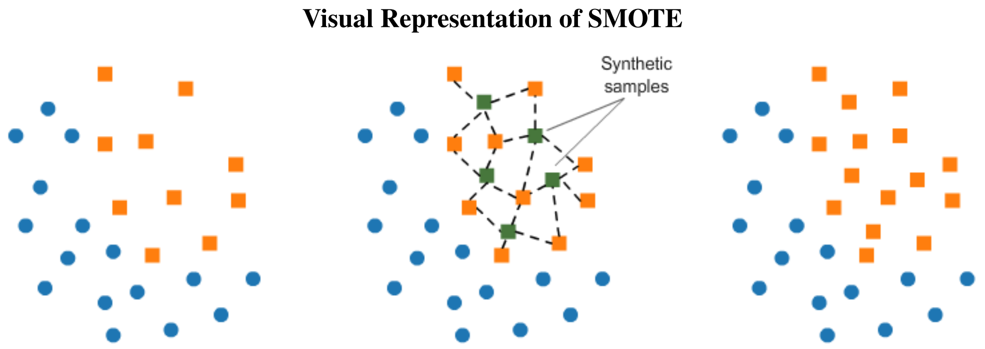


Figure 1: Visual demonstration of SMOTE, image from https://www.kaggle.com/rafjaa/resampling-strategies-for-imbalanced-datasets. Artificial samples are generated in the minority class (orange) until they equal the number of majority class (blue) samples.

3.3 Data Exploration

3.3.1 Correlation

Before analyzing the methods, we performed a preliminary data exploration of the provided sample. First and foremost, many financial data are usually correlated with one another: generally speaking, when a firm is ”healthy”, its sales increase, share price increase, cash flows increase, and so on. Thus, suspecting a high correlation amongst the features, our first step in analysis was observing the correlation heat map.

Observing the plot in Figure 2, contrary to our preconception, we see that while *overall* many variables are uncorrelated (represented by white squares), there exist a few variables that are heavily correlated with one another.

For instance, the dark blue coordinate at (1, 11) indicates that feature 1 and 11 are heavily positively correlated. Looking at Appendix 1, this makes sense, since feature 1 is net profit/total assets while feature 11 is gross profit + extraordinary profit + financial expenses/total assets. Since both feature 1 and 11 depict the ratio of *some* measure of profit to total assets, it makes sense that they are strongly positively correlated.

3.3.2 Principal Component Analysis

Since a few of our data points are strongly correlated with one another, in hopes of reducing bias and improving our training time, we applied Principal Component Analysis (PCA) on our data set (Figure 3).

We found that 35 principal components explained over 99% of the variance in our inputs. In our methods, we will run everything two times: once without PCA applied, and a second time with PCA, for $n = 35$.

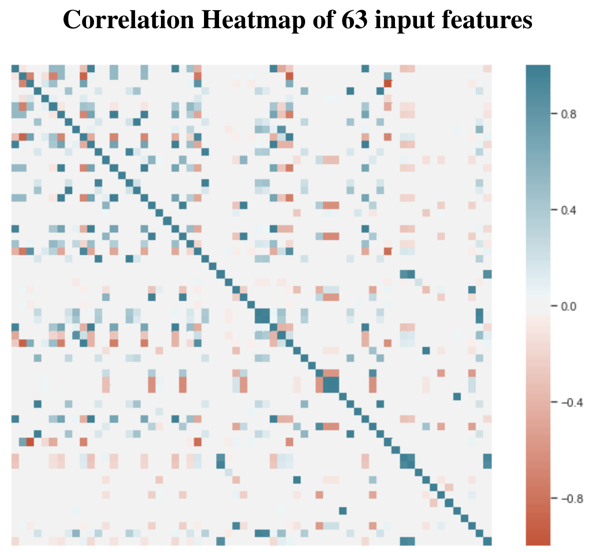


Figure 2: Correlation heat map of the 63 input features. Dark blue (1) represents a strong, positive correlation, while a dark red represents a strong, negative correlation. White indicates no correlation between the variables. For instance, the dark red square at the coordinate (6, 2) (or, conversely at (2,6)) indicates that variable 6 and 2 are strongly negatively correlated.

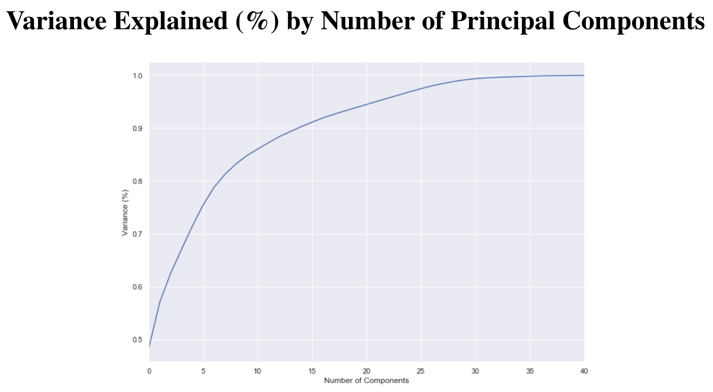


Figure 3: PCA applied onto feature vectors. Horizontal Axis measures the number of components, and Vertical measures the percentage of total input variance explained.

4 Methods

As mentioned before, the task we are performing is given the 63 financial ratios for a firm, measured at a specific year between 2000 and 2013, classify whether that firm will file for bankruptcy within two years or not (0 is healthy, 1 is bankrupt).

To measure the performance of our ensemble methods, we will use three baseline models: logistic regression, SVM, and K-Nearest Neighbors Classifier (KNN). For SVM, we used a RBF kernel, and we chose $K = 147$ for KNN.

Our two classifiers of interest are random forest and gradient boosting. Both algorithms are ensemble learners, which in the simplest terms mean that the overall classifier is comprised of multiple smaller classifiers, and the overall decision is chosen as some kind of aggregate of the individual decisions. Random forest uses decision trees as its weak learners, and we did the same for gradient boosting in our work. As we have previously observed, while decision trees boast of great training accuracy, they are prone to overfitting as the depth of the tree increases. By aggregating multiple decision trees (created independently of one another) and by splitting *at random* the subset of features, Random Forest boasts of higher accuracy than conventional decision trees.

4.1 Gradient Boosting

Boosting is an ensemble method similar to but different from Random Forest in many ways. Most notably, while trees are generated at random forest, boosting generates each tree iteratively. Consider the objective function below:

$$\text{obj} = \sum_{i=1}^{n} l(y_i, \hat{y}^{(t)}) + \sum_{i=1}^{t} \Omega(f_i)$$

where n is the number of samples, l is a loss function, y_i is the observed data, $\hat{y}^{(t)}$ is the t-th predicted outcome, and where the last term in the equation is a regularization term. In tree-based gradient boosting, the regularization term is modified at each step. Namely, the regularization in step $t + 1$ is specifically engineered so that it covers for the worst sample predictions in step t. For gradient boosting, we used the XGBoost library.

4.2 Hyperparameter Tuning

For random forest and gradient boosting, we rigorously tuned for a few specific hyperparameters, namely the the number of trees, the minimum number of samples to split on, the max depth, and for gradient boosting, the learning rate.

For a compromise between efficiency and accuracy, we first started by using a randomized grid search with 3-fold cross validation to estimate initial values for these features. After initial values were found, we used an exhaustive grid search in a range nearby from the randomized results to find a more precise value for the hyperparameters. This section of hyperparamter tuning composed the majority of our training time. The process was performed following the steps in this guide here [4].

5 Results

All models were trained on CPU, using 4 cores, with an Intel Core i5-7500 CPU at 3.40GHz, with 8 GB of available RAM. Without PCA, the training time for random forest totaled 115 minutes, and gradient boosting took 58 minutes. With PCA, the numbers were 91.6 and 37.6 respectively, which were improvements of about 20% and 35% in training time.

Before discussing the performance results, one important caveat to have in mind for this project is that recall (out of all the true 1s, how many did we correctly predict) than the precision (out of all the 1s we predicted, how many of them were actually 1s). This makes intuitive sense in the context of our problem. Erroneously classifying a healthy firm as in risk of bankruptcy of course comes with its consequences. However, classifying an insolvent firm as healthy can be more problematic. Incorrectly classifying an insolvent firm can cause lenders to give money to firms with high levels of default, and investors and credit scoring agencies will overprice the target firm's assets. These phenomena were all a few of the many causes for the 2007-2008 economic crisis in the U.S..

Figure 4 below illustrates the performance of our models, without PCA, in terms of ROC. We can clearly see that the two ensemble methods significantly outperform the three baseline methods. And between the two ensemble methods, gradient boosting outperformed random forest by a handsome margin.

Table 1 below contains a more detailed view of the model performances. We see that the two ensemble methods both outperformed the baseline methods in every measure, and gradient boosting in particular performed the best in every measure. The AUROC values for the random forest and gradient boosting were 93% and 96% respectively, demonstrating great predictive capabilities. Most notably, while the recall values of the baseline models were noticeably low (11%, 14%, 11%), the recall values for the ensemble values were significantly higher (50% and 75%). As discussed before, this result has great implications, since one of the most important tasks in this problem is making sure to not misclassify an insolvent firm.

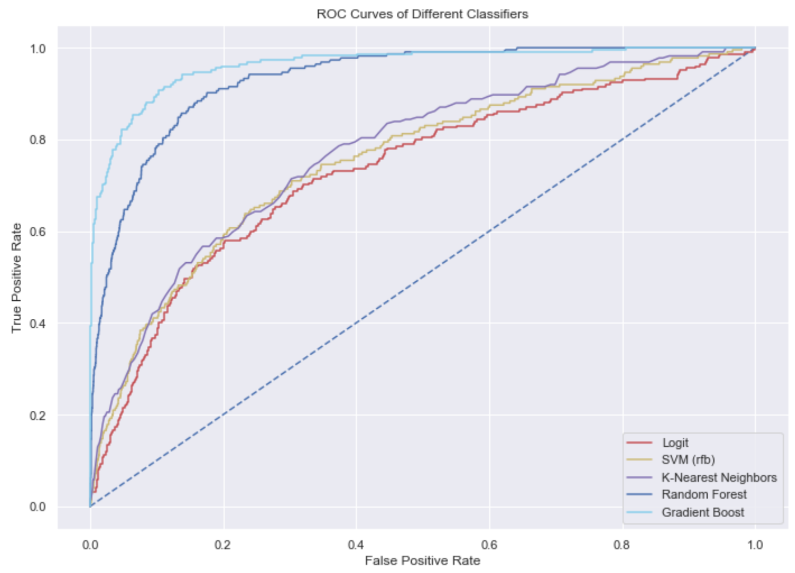
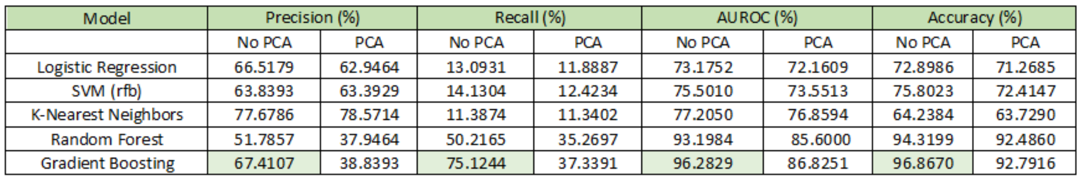


Figure 4: ROC curve of the 5 methods examined. Gradient boost had the highest ROC, followed by Random Forest. All of the baseline methods scored considerably lower.



Model	Precision (%)		Recall (%)		AUROC (%)		Accuracy (%)	
	No PCA	PCA	No PCA	PCA	No PCA	PCA	No PCA	PCA
Logistic Regression	66.5179	62.9464	13.0931	11.8887	73.1752	72.1609	72.8986	71.2685
SVM (rfb)	63.8393	63.3929	14.1304	12.4234	75.5010	73.5513	75.8023	72.4147
K-Nearest Neighbors	77.6786	78.5714	11.3874	11.3402	77.2050	76.8594	64.2384	63.7290
Random Forest	51.7857	37.9464	50.2165	35.2697	93.1984	85.6000	94.3199	92.4860
Gradient Boosting	67.4107	38.8393	75.1244	37.3391	96.2829	86.8251	96.8670	92.7916

Table 1: Performance results of the five models measured by precision, recall, Area under the ROC curve, as well as accuracy on the test data. Results for model performance both with and without PCA are included.

We also notice that applying PCA actually decreases our performance in almost every aspect. the AUROC values for random forest and gradient boosting decrease by about 8% and 10% respectively, and the recall also drop by a substantial amount.

6 Discussion and Conclusion

6.1 Evaluation

As aforementioned, ensemble decision tree classifiers outperformed the others by a handsome margin. Not only did they have greater accuracy and AUC, but they also had much higher recalls (gradient boosting's recall was almost seven times that of KNN). However, we must note that a recall of 75% is still definitely non-ideal. While 96% AUC is certainly a high value, misclassifying every fourth insolvent firm as healthy is not a very desirable ratio.

Regarding PCA, we have noted that transforming the feature space with PCA significantly improved the training time, but as a cost, our ensemble models both decreased significantly in accuracy (the baseline models were worsened, but comparatively not by much). This makes sense, since PCA decreases both the number of features (from 63 to 35) and the collinearity between variables, which both boost the performance of random forest classifiers.

6.2 Future Work

The obvious first step in improving performance would be employing a different tactic for feature selection and extraction. One easy place to start would be to use extra trees to pick out the most important features, and explicitly use them instead of transforming our feature space. Another more involved process would be to first reduce the set down by using a univariate selection method (maybe on basis of mutual information), then hand-pick the most relevant ones by using financial research and expertise.

Another improvement topic to note is that since the model data was between 2000 and 2013, one of the worst economic downturns of recent years. This could suggest that normally, it may be the case that even fewer firms go bankrupt, in which case getting a high value for recall would be even more difficult. Thus, if the information was provided to us, we would stratify the companies by year, and ideally also include macroeconomic variables as features to consider (unemployment rate, inflation rate, Gross Domestic Product growth, etc).

And lastly, an interesting extension to this problem would be to reorient the problem from one of a classification task to preventing a time-to-event. Models such as the Cox Hazard Model can help predict the first occurrence of an unfavorable event using machine learning, and albeit more difficult, if an accurate model could be developed, it would be more practical than our current problem, since it would predict a specific time for bankruptcy instead of a simply binary classification.

References

[1] Edward I. Altman. Financial ratios, discriminant analysis and the prediction of corporate bankruptcy. *The Journal of Finance*, 23(4):589–609, 1968.

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

[3] Nick Becker. The right way to oversample. `https://beckernick.github.io/oversampling-modeling/`. Accessed: 2019-05-10.

[4] Will Koerhsion. Hyperparameter tuning the random forest in python using scikit-learn. `https://towardsdatascience.com/hyperparameter-tuning-the-random-forest-in-python-using-scikit-learn-28d2aa77dd74` Accessed: 2019-05-10.

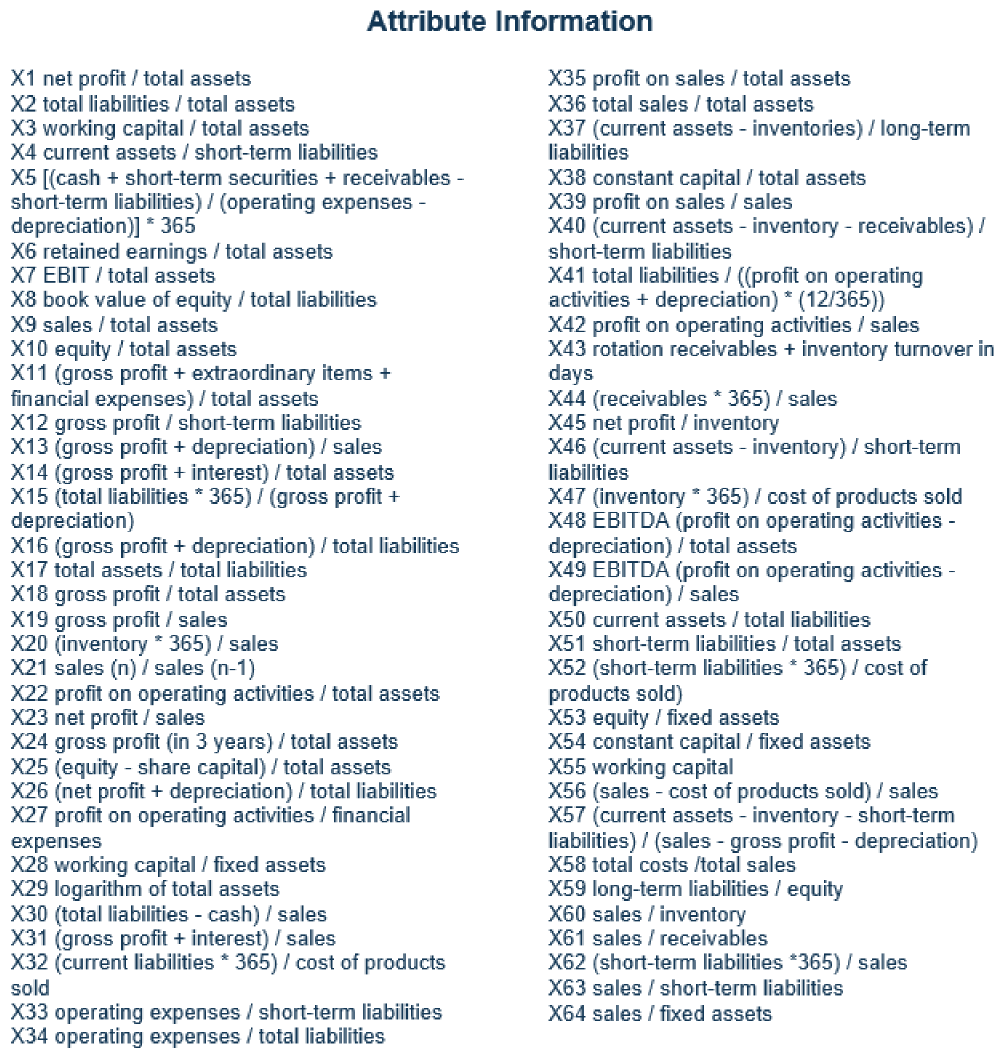
[5] James A. Ohlson. Financial ratios and the probabilistic prediction of bankruptcy. *Journal of Accounting Research*, 18(1):109–131, 1980.

[6] David Schoenherr. Lecture notes: Capital strucutre and the cost of debt, 2017.

[7] Sebastian Tomczak. UCI machine learning repository, 2016.

7 Appendix

A Features in the Data set



Attribute Information

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

Appendix A: The 64 features that were included in the original data set provided by the UCL Repository for Machine Learning.