Project Title: Corporate Bankruptcy Prediction
Team Member Name: Neha Abajirao Patil
SUNet Id: nehap7
Project Category: General Machine Learning

Abstract

According to Wikipedia, Bankruptcy Prediction is the art of predicting bankruptcy and various measures of financial distress of public firms. Bankruptcy Prediction has been an interesting problem in the financial world for a long time. The significance of predicting financial distress is to develop a reliable predictive model that will provide a means to measure and predict the financial condition of a corporate entity. In this domain, there is a vast majority of research published in various scientific journals noting their observation of accuracy of models for predicting the bankruptcy of corporations using forecasting techniques. Goal of this project is to evaluate the performances of statistical techniques like Logistic Regression, Naïve Bayes, SVM, neural network and Gradient Boosting that we learned in class for accuracy, performance and scalability.

1. Introduction

Economic crisis of 2007 highlighted the need for a market sustainability. We need predictive tools to better predict such events to prevent such catastrophic occurrence from happening again. The ability to predict bankruptcy gives lenders better understanding of their profit margins in their financial transaction worldwide. Therefore, due to consequence of the high social and economic costs attached to corporate bankruptcies it has attracted lots of significance and attention of researches for better understanding and developing performant models which predicts the financial distress with high accuracy. The purpose of the bankruptcy prediction is to assess the financial condition of a company and its future perspectives within the context of long-term operation on the market [1]. As with any other machine learning problems, prediction depends heavily on availability of data to train it accurately. For forecasting we rely on data with economic indicators curated by domain experts and apply machine learning methodologies learned in the course and evaluate their accuracy and performance when applied to real world problems of corporate accounting exclusively focused towards predicting corporate bankruptcy.

2. Related Work

Bankruptcy prediction has been a subject of analysis since 1932, starting with Fitzpatrick using a feature set of date, size and industry [3]. His work laid the foundation of modern-day statistical approach in financial forecasting domain. In 1967, William Beaver applied statistical techniques like t-tests to evaluate the performance of prediction [4] followed by Edward Altman who effectively applied multiple discriminant analysis and use of Altman Z-score which is still used in today's model [5] which was further developed by others. In congruity, a great interest was paid to generalized linear models that can be used both in decision making as well as for forecasting. One of the most successful models was support vector machine by Shin [6]. Lately, Neural network models and other sophisticated models have been tested on bankruptcy prediction with evolved features like age, bad press, payment incidents from creditors as well. The latest research involves comparison of various approaches and modeling techniques to ascertain whether any one technique is superior to its counterpart. A novel rule-based system was introduced by Zhang, Wang, and Ji [7] to obtain compressible models in terms of first-order logic with an easy to understand knowledge representation. Recently, it has been shown that the

ensemble classifier and automatic feature extraction can be successfully applied to the bankruptcy prediction [8] and it significantly beats other methods.

3. Dataset and Features

3.1 Data Gathering

To evaluate the performance and accuracy of various techniques we are relying on dataset [2] which is hosted on UCI Machine Learning Repository. The dataset is about bankruptcy prediction of Polish companies in manufacturing sector. The motivation in choosing this repository was because since 2004 Poland saw many manufacturing sector going bankrupt. The data was collected from Emerging Markets Information Service (EMIS), which is a database containing information on emerging markets around the world. The bankrupt companies were analyzed in the period 2000-2012, while the still operating companies were evaluated from 2007 to 2013. The research sample consists of bankrupt and still operating companies (imbalanced sample). Finally, data has the 64 financial indicators to be analyzed as features.

3.2 Data Preparation

The data that is available is in classified in 5 cases depending on forecasting period. First year forecasting period data had economic indicator for bankruptcy status after 5 years for 7,027 instances (financial statements) where 271(3.8%) represents bankrupted companies, 6,756(96%) firms that did not bankrupt in the forecasting period. Second year indicated bankruptcy status after 4 years for 10173 instances where 400 (3.9%) represents bankrupted companies, 9,773 (96%) firms that did not bankrupt in the forecasting period. Similarly, for third year, 10,503 instances, 495(4.7%) represents bankrupted companies, 10,008(95%) firms that did not bankrupt in the forecasting period. For fourth year, 9,792 instances, 515(5%) represents bankrupted companies, 9,277(95%) firms that did not bankrupt in the forecasting period. Finally, for fifth year of the forecasting period and corresponding class label that indicates bankruptcy status after 1 year. The data contains 5,910 instances, 410(6%) represents bankrupted companies, 5500(94%) firms that did not bankrupt in the forecasting period. I am considering evaluating on third year of forecasting data to maximize the number of training examples, percentage of positive and negative instances.

3.3 Training and Test Data

We choose and split the third-year forecasting data randomly into 80% for training and hold out 20% data for testing. Each test data instance represents 64 economic indicators and bankruptcy status after three years. In training data, we have total 8,402 instances(firms), out of which 384(4.5%) represents bankrupted companies, 8,018 (95.5%) firms that did not bankrupt in the forecasting period. In test data, we have 2,101 instances(firms), 111(5.2%) represents bankrupted companies, 1,990(94.7%) firms that did not bankrupt in the forecasting period.

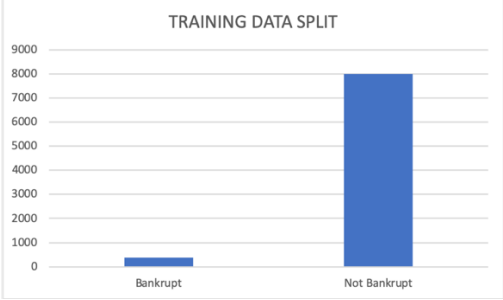
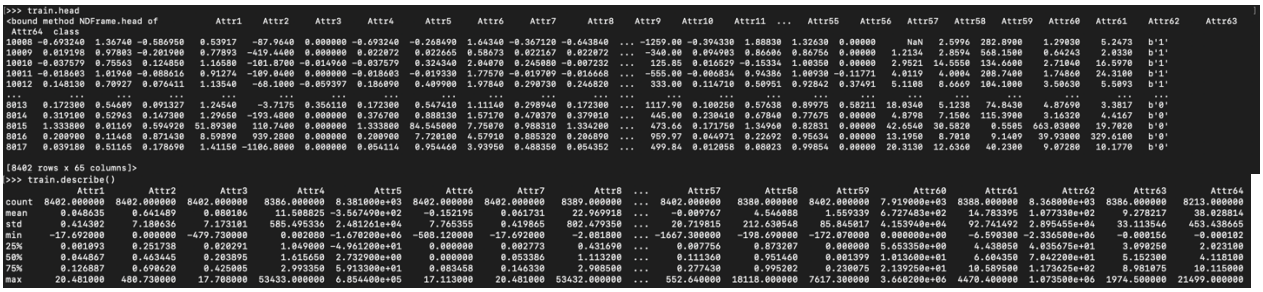


Figure 1 Training Data Split



```
>>> train.head
<bound method NDFrame.head of          Attr1    Attr2    Attr3    Attr4    Attr5    Attr6    Attr7    Attr8    Attr9   Attr10   Attr11  ...   Attr55   Attr56  Attr57  Attr58  Attr59  Attr60  Attr61  Attr62     Attr63
 Attr64  class
10008 -0.693240  1.36740 -0.586950  0.53917   -87.9640  0.000000 -0.693240 -0.268490  1.64340 -0.367120 -0.643840  ... -1259.00 -0.394330  1.88830  1.32630  0.00000     NaN  2.5996  282.8900  1.29030   5.2473   b'1'
10009  0.019198  0.97803 -0.201900  0.77893  -419.4400  0.000000  0.022072  0.022665  0.58673  0.822167  0.022072  ...  -340.00  0.094903  0.86606  0.86756  0.00000  1.2134   2.8594  568.1500  0.64243   2.0330   b'1'
10010 -0.037579  0.75563  0.124850  1.16580  -101.8700 -0.014760 -0.037579  0.324340  2.04870  0.245080 -0.007232  ...  -125.85  0.016529 -0.15334  1.00350  0.00000  2.9521  14.5550  134.6600  2.71840  16.5970   b'1'
10011 -0.018603  1.01960 -0.088616  0.91274  -109.0400  0.000000 -0.018603 -0.019330  1.77570 -0.019709 -0.016668  ...  -555.00 -0.006834  0.94386  1.00930 -0.11771  4.0119   4.0004  208.7400  1.74860  24.3100   b'1'
10012  0.148130  0.70927  0.076411  1.13540   -68.1000 -0.059397  0.186090  0.409900  1.97840  0.290730  0.246820  ...   333.00  0.114710  0.50951  0.92842  0.37491  5.1108   8.6669  104.1000  3.50630   5.5093   b'1'
...         ...      ...      ...      ...        ...       ...       ...       ...      ...       ...       ...  ...      ...       ...      ...      ...      ...      ...       ...      ...      ...      ...
8013  0.172300  0.54489  0.091327  1.24540   -3.7175  0.356110  0.172300  0.547410  1.11140  0.298940  0.172300  ...  1117.90  0.100250  0.57638  0.89975  0.58211  18.0340  5.1238   74.8430  4.87690   3.3817   b'0'
8014  0.319100  0.52963  0.147300  1.29650  -193.4800  0.000000  0.376700  0.888130  1.57170  0.470370  0.379010  ...   445.00  0.230410  0.67840  0.77675  0.00000  4.8798   7.1506  115.3900  3.16320   4.4167   b'0'
8015  1.333800  0.01169  0.594920  51.89300   110.7400  0.000000  1.333800  84.645000  7.75070  0.988310  1.334200  ...   473.66  0.171750  1.34960  0.82831  0.00000  42.6540  30.5820   0.5505  663.03000  19.7020   b'0'
8016  0.280900  0.11468  0.871430   8.59890   939.2800  0.000000  0.280900   7.720100  4.57910  0.885320  0.286890  ...   959.97  0.044971  0.22692  0.95634  0.00000  13.1950   8.7010   9.1409  39.93000  329.6100  b'0'
8017  0.039180  0.51165  0.178690  1.41150 -1106.8000  0.000000  0.054114  0.954460  3.93950  0.488350  0.054352  ...   499.84  0.012058  0.00023  0.99854  0.00000  20.3130  12.6360  40.2300   9.07280   10.1770   b'0'

[8402 rows x 65 columns]>
>>> train.describe()
              Attr1         Attr2         Attr3         Attr4         Attr5         Attr6         Attr7         Attr8  ...         Attr57        Attr58        Attr59        Attr60        Attr61        Attr62        Attr63        Attr64
count  8402.000000  8402.000000  8402.000000  8386.000000  8.381000e+03  8402.000000  8402.000000  8389.000000  ...    8402.000000   8380.000000   8402.000000  7.919000e+03  8388.000000  8.368000e+03  8386.000000  8213.000000
mean      0.048635     0.641489     0.080106    11.508825 -3.567490e+02    -0.152195     0.061731    22.969918  ...      -0.009767      4.546088      1.559339  6.727483e+02    14.783395  1.077330e+02     9.278217    38.028814
std       0.414302     7.180636     7.173101   585.495336  2.481261e+04     7.765355     0.419865   802.479350  ...      20.719815    212.630548     85.845017  4.153940e+04    92.741492  2.895455e+04    33.113546   453.438665
min     -17.692000     0.000000  -479.730000     0.002080 -1.670200e+06  -508.120000   -17.692000    -2.081800  ...   -1667.300000   -198.690000   -172.070000  0.000000e+00    -6.590300 -2.336500e+06    -0.000156    -0.000102
25%       0.001093     0.251738     0.020291     1.049000 -4.961200e+01     0.000000     0.002773     0.431690  ...       0.007756      0.873287      0.000000  5.653350e+00     4.438050  4.035675e+01     3.099250     2.823100
50%       0.044867     0.463445     0.203895     1.615650  2.732900e+00     0.000000     0.053386     1.113200  ...       0.111360      0.951460      0.001399  1.013600e+01     6.604350  7.042200e+01     5.152300     4.118100
75%       0.126887     0.698620     0.425085     2.993350  5.913300e+01     0.083458     0.146330     2.908500  ...       0.277430      0.995202      0.230075  2.139250e+01    10.589500  1.173625e+02     8.981075    10.115000
max      20.481000   480.730000    17.708000  53433.000000  6.854400e+05    17.113000    20.481000  53432.000000  ...     552.640000  18118.000000  7617.300000  3.660200e+06  4470.400000  1.073500e+06  1974.500000  21499.000000
```

Figure 2 Training Data Details

3.4 Missing Data

We saw that our dataset had lots of missing values across features. Missing data can introduce a substantial amount of bias in our learning during training. It can also impact the accuracy and efficiency of our models. Dropping such data is likewise harmful for the same reason as above. We considered using Mean, Median, Nearest Neighbors and Multivariate Imputation. We choose to use Mean Imputation for our problem as it reduces the correlations involving the feature variable been imputated. We achieved mean imputation using Imputer class from scikit-learn's library.

4. Methods

4.1 Logistic Regression

Logistic regression is a linear model of classification. We use L2 regularization to avoid overfitting the data and introducing high variance in our model. In this model, the probabilities describing the possible outcomes are modelled using a logistic function also called as sigmoid function.

$$p(y \mid x; \theta) = (h_\theta(x))^y \left(1 - h_\theta(x)\right)^{1-y}$$

The cost function mentioned on the scikit-learn documentation with regularization is as below

$$\min_{w,c} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \log(\exp(-y_i(X_i^T w + c)) + 1).$$

4.2 SVM

A support vector machine is a supervised classification model that constructs a hyper-plane or set of hyper-planes in a high dimensional space, which can be used for classification tasks. Intuitively, a good separation is achieved by the hyper-plane that has the largest distance to the nearest training data points of any class (called functional margin), since in general the larger the margin the lower the generalization error of the classifier.

The advantages of support vector machines are that it is effective in high dimensional spaces, effective in cases where number of dimensions is greater than the number of samples. It also uses a subset of training points in the decision function (called support vectors), so it is also memory efficient due to kernels. Our cost function is as below

$$= \sum_{i=1}^{n} \alpha_i y^{(i)} \langle x^{(i)}, x \rangle + b.$$

4.3 Neural Network

We use Neural Network, which is part of supervised classification model using a multi-layer perceptron (MLP) algorithm that trains using backpropagation. We used L2 regularization term which helps in avoiding overfitting by penalizing weights with large magnitudes and stochastic

Gradient descent as a way to achieve optimum value. Adam Gradient descent though performs better, did not perform well for us. Our model had 5 neurons in 2 hidden layers. We simply used logistic function with Square Error loss function.

4.4 Naïve Bayes
Naive Bayes methods are a set of supervised learning algorithms based on applying Bayes' theorem with the "naive" assumption of conditional independence between every pair of features given the value of the class variable. Bayes' theorem states the following relationship, given class variable and dependent feature vector

$$P(y \mid x_1, \cdots, x_n) = \frac{P(y)P(x_1, \dots x_n \mid y)}{P(x_1, \cdots, x_n)}$$

We used Naïve Bayes classifier based on Bernoulli Model, we experimented with different model, but they were not able to perform the prediction with high accuracy, For instance Gaussian based model was resulting in negative score no matter how we fine tune it.

4.5 Extreme Gradient Boosting
According to Wikipedia, Extreme Gradient Boosting is based on the principle of gradient boosting framework. Gradient boosting produces a prediction model in the form of an ensemble of weak prediction models typically decision tree. It builds the model in a stage-wise fashion like other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function. XGBoost uses a more regularized model formalization to control overfitting for better performance. We used a depth of 3 with squared error objective function and gblinear booster (linear) with L2 regularization to avoid overfitting.

4.6 Light Gradient Boosting Machine
Light Gradient Boosting Machine is gradient boosting framework that uses tree-based learning algorithm and optimizes using histogram-based algorithms for performance efficiency. The difference is LGBM grows vertically as oppose to horizontally. In other words, it grows leaf vise while other grow level wise. Similar as above we use L2 regularization as it fits better due to binary problem nature.

5. Results

All the methods were L2 regularized and had a learning rate of 0.001. We found that any other learning rate was causing delay in converging. Most of the parameters we used is explained in section 4 Models. In favors of brevity of this report, we have not shared the experimentation results where different model parameters gave worst performance in terms of accuracy.

5.1 Cross Validation
We perform K Fold cross validation for evaluation the performance of our models where K=5. Our Observations are graphically displayed in Figure 3.a and Figure 3.b.

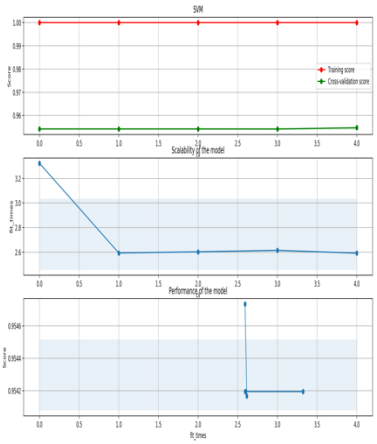
5.2 Metrics
Primary metrics used to evaluate the performance of models are Accuracy Score, Log Loss, Fit Times score and confusion matrix over both test and training data set. Accuracy Score and Log Loss are defined as below

$$\texttt{accuracy}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} 1(\hat{y}_i = y_i) \qquad L_{\log}(y, p) = -\log \Pr(y|p) = -(y \log(p) + (1-y) \log(1-p))$$

For instance, Confusion matrix for SVM was [[1990 0] [111 0]]

5.3 Data Scaling

We choose not to standardize the features because for our observation of data we deduced that our data was not standard normally distributed. Forcing to remove mean and scaling to unit variance was resulting in performance drop.



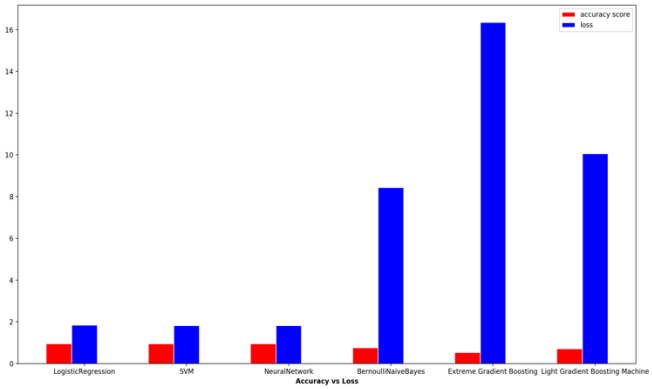


Figure 3.a: Learning Curve Metrics like Scalability, Efficiency and Correctness for SVM
Figure 3.b: Comparison of Model Performance under evaluation.

Classifier	Test Accuracy	Test Log Loss	Train Accuracy	Train Time
LogisticRegression	0.946692	1.841192	0.952392	1.01
SVM	0.947168	1.824752	1.000000	2.7
NeuralNetwork	0.947168	1.824752	0.954297	0.8
NaiveBayes	0.755831	8.433488	0.765413	0.006
ExtremeGradientBoosting	0.526416	16.357391	1.000000	1.8
LightGradientBoostingMachine	0.708710	10.061028	1.000000	0.7

Table 1 Comparison of Performance of models under evaluation

Based on our observation SVM performed better than any other model in terms of maximum accuracy and minimum loss. SVM to be performing better for this scale of data and features was expected as we learned in class since it uses vectors aka subset of data to operate. Same can be said for performance for Logistic Regression and Neural Network. It's understandable that Naïve Bayes did not perform so well because of model assumptions. The unexpected observation was for Gradient Boosting, which needs further inspection to elevate the performance. Future work should entail starting with replicating recent successes in academia research with synthetic feature generation to such data problem.

6. Conclusion

The project evaluated the approaches for the problem of predicting the bankruptcy basing on the financial factors. We took under consideration the financial condition of Polish companies from 2007 to 2012 for (still operating). To solve the stated classification problem, we applied various models. The results gained by the SVM, Logistic and Neural Network were significantly better than the results gained by Boosting. Furthermore, we should explore other methods like synthetic feature generation for better performance and try to propose a novel solution.

References

[1] Constand, R.L., Yazdipour, R., 2011. Firm failure prediction models: a critique and a review of recent developments, in: Advances in Entrepreneurial Finance. Springer, pp. 185–204.
[2] https://archive.ics.uci.edu/ml/datasets/Polish+companies+bankruptcy+data
[3] FitzPatrick, P. Joseph. (1932). *A comparison of the ratios of successful industrial enterprises with those of failed companies.* [Washington.
[4] Beaver 1966. Financial ratios predictors of failure. *Journal of Accounting Research, 4 (Supplement), p. 71-111.*
[5] https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1540-6261.1968.tb00843.x
[6] Shin, K.S., Lee, T.S., Kim, H.j., 2005. An application of support vector machines in bankruptcy prediction model. Expert Systems with Applications 28, 127–135.
[7] https://dl.acm.org/citation.cfm?id=2936632
[8] https://ieeexplore.ieee.org/abstract/document/859421
[9] Fan, W., Stolfo, S.J., Zhang, J., Chan, P.K., 1999. AdaCost: misclassification cost-sensitive boosting, in: ICML, pp. 97–105.
[10] Nanni, L., Lumini, A., 2009. An experimental comparison of ensemble of classifiers for bankruptcy prediction and credit scoring. Expert Systems with Applications 36, 3028–3033.
[11] http://www.scirp.org/reference/ReferencesPapers.aspx?ReferenceID=2152779
[12] Scikit-learn: Machine Learning in Python, Pedregosa *et al.*, JMLR 12, pp. 2825-2830, 2011.
[13] J. D. Hunter, "Matplotlib: A 2D Graphics Environment", Computing in Science & Engineering, vol. 9, no. 3, pp. 90-95, 2007.
[14] https://en.wikipedia.org/wiki/Gradient_boosting
[15] https://pandas.pydata.org
[16] Chen, T., He, T., 2015b. xgboost: eXtreme Gradient Boosting. R package version 0.3-0. Technical Report .
[17] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, Tie-Yan Liu. "LightGBM: A Highly Efficient Gradient Boosting Decision Tree". Advances in Neural Information Processing Systems 30 (NIPS 2017), pp. 3149-3157.
[18] https://en.wikipedia.org/wiki/Bankruptcy_prediction
[19] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, CJ Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E.A. Quintero, Charles R Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. (2019) SciPy 1.0–Fundamental Algorithms for Scientific Computing in Python. *preprint*

Code Link: https://github.com/codeatri/Bankruptcy_Prediction