Machine Learning Engineer Nanodegree

Capstone Project - Bankruptcy Prediction of Polish Companies

Morgan Kidd
July 29, 2018.

I. Introduction

The problem of bankruptcy prediction has been widely researched both before and after the popularization of machine learning. Beaver (1966) and Altman (1968) first presented the analysis of certain financial statistics correlated with bankruptcy that became the basis for many subsequent studies. This lead to a growth in Discriminant Analysis for bankruptcy forecast, notably Ohlson (1978), who's method is still taught in finance and business schools today.

Odom & Sharda (1990) present the first neural network attempt at bankruptcy prediction using the variables introduced in Altman (1968). Wilson & Sharda (1994) expanded on the work of this paper, and lead to a large proliferation of machine learning-oriented papers towards prediction of company failure. Decision trees also began to be experimented with shortly after and compared to the neural network approaches. Examples include bank failures (Fernandez & Olmeda 1995, Martinelli et. al 1999, McKee & Greenstein 2000), and in credit risk evaluation (Piramuthu 1999). Support Vector Machines, first introduced by Cortes & Vapnik (1995), were also quickly applied to the bankruptcy prediction task in Fan & Palaniswami (2000), and more notably by Min & Lee (2005) and Shin, Lee & Kim (2005).

In more recent years, the breadth of methods present in papers has increased, including many ensemble method papers such as and Kim & Kang (2010), although some skepticism on the efficacy of multiple classification networks has also been expressed by Tsai & Wu (2008). Recent papers tend to focus on successful replication of previous methods, and applying existing methods to new datasets, such as Iturriaga & Sanz (2015) who focused on visualization and commercial banks, or meta-studies such as Abellan & Mantas (2014).

In this paper, I with exploratory analysis of the data set used by Zieba, Tomczak & Tomczak (2016) in section 2.1. I then explore possible data augmentation and filtering in section 3.1, then begin with model implementation, starting with support vector machines in section 3.2 based on Min & Lee (2005). Section 3.3 looks at neural networks based on the work of Shin, Lee & Kim (2005). Finally, section 3.4 will explore newer gradient boosting methods such as LightGBM (Ke et al. 2017), and section 3.5 looks at some model and measurement improvement along with the inclusion of an ensemble method. Test results are discussed at the end along with possible future research.

1.1 Problem statement

Each datapoint includes a binary target variable, X65, corresponding to the firm's bankruptcy state by the end of the sampling period. The problem of interest is whether bankruptcy can be accurately predicted ahead of time. In other words, can the target variable be accurately classified based on the feature set.

1.2 Metrics

For this task, I will be focusing on three evaluation metrics. From an investor's perspective, bankruptcy prediction is most valuable for avoiding portfolio losses, which would be accomplished by excluding the as many bankrupt firms as possible. This can be measured through recall, defined as follows:

$$\text{Recall} = \frac{\text{TP} \cap \text{PP}}{\text{TP}}$$

Where TP is the set of "true positives", firms that go bankrupt in the sample period, and PP is the set of "predicted positives", or firms that are predicted to go bankrupt. Recall is the inverse of the false negative rate, and a lower recall indicates that the prediction more accurately identifies all bankrupt firms.

Additionally, portfolio losses could be avoided by maximizing the investment space with many non-bankrupt firms. Precision is the inverse of the false positive rate. A higher precision metric indicates that the algorithm does not falsely identify many companies as going bankrupt. Precision is defined as follows:

$$\text{Precision} = \frac{\text{TP} \cap \text{PP}}{\text{PP}}$$

Where, once again, TP is the set of "true positives", firms that go bankrupt in the sample period, and PP is the set of "predicted positives", or firms that are predicted to go bankrupt.

Finally, recall may always be sacrificed for higher precision by altering the class weights or probability threshold to predict one class over the other. Unless otherwise noted, the default acceptance threshold is 50%. To create a more objective measure of performance, I will use the area under the receiver operating characteristic curve (ROC AUC), which essentially measures the probability that a classifier will assign a higher score to a randomly chosen positive example than a negative example. (Fawcett, 2006). ROC AUC is a simple way of representing the tradeoff between precision and recall without making any (potentially biased) prioritization of precision over recall.

Although precision and recall are included in the diagnostic metrics for the qualitative evaluation of models, ROC AUC is the primary metric used for model assessment and gradient descent/model learning in this project.

II. Analysis

2.1 Data Exploration

The dataset is a collection of financial data on Polish companies sampled from EMIS, a data provider on emerging markets, for the period 2007-2013. The data was originally collected by Zieba, Tomczak & Tomczak (2016) and donated to the UCI Machine Learning Repository.

Table 1

Feature Set

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 + receivable – 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) * 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 / total assets
X17	Total assets / total liabilities	X49	EBITDA / 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 (over 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

The dataset has a total sample size of 43405 and includes 64 features derived from the financial statements of companies. The full feature description may be seen in Table 1. The classification label is a binary variable representing whether the company went bankrupt before the end of the sample period. There were 41322 missing values in the dataset, representing approximately 1.5% of the total sample. Missing values were interpolated as the average of that feature based on all present samples.

Table 2a

Descriptive Statistics – X1 to X32

ID	Mean	Std. Dev.	Min	Max	25%	50%	75%	Mean – 0	Mean – 1	Sig.
X1	0.035	2.994	-463.89	94.28	0.003	0.050	0.130	0.053	-0.320	**0.000**
X2	0.590	5.843	-430.87	480.90	0.269	0.472	0.688	0.544	1.505	**0.000**
X3	0.114	5.439	-479.96	28.336	0.022	0.197	0.403	0.157	-0.735	**0.000**
X4	6.314	295.4	-0.403	53433	1.050	1.570	2.787	6.424	4.146	0.731
X5	-385.4	61243	-1.2e+07	1.3e+06	-49.08	-1.035	50.63	-367.0	-746.2	0.782
X6	-0.056	7.201	-508.41	543.25	0.00	0.00	0.089	0.000	-1.165	**0.000**
X7	0.093	5.713	-517.48	649.23	0.006	0.060	0.151	0.114	-0.311	**0.001**
X8	12.64	505.9	-141.41	53432	0.430	1.070	2.616	12.95	6.600	0.576
X9	2.652	62.93	-3.496	9742.3	1.019	1.200	2.063	2.689	1.930	0.591
X10	0.627	14.67	-479.91	1099.5	0.295	0.506	0.709	0.677	-0.365	**0.002**
X11	0.131	5.307	-463.89	681.54	0.015	0.075	0.167	0.150	-0.253	**0.001**
X12	1.132	67.59	-6331.8	8259.4	0.015	0.172	0.587	1.377	-3.707	**0.001**
X13	0.810	86.94	-1460.6	13315	0.023	0.068	0.135	0.837	0.265	0.770
X14	0.093	5.713	-517.48	649.23	0.005	0.060	0.151	0.114	-0.311	**0.001**
X15	1991	96432	-9632400	1.0e+07	222.5	846.3	2226.9	1873.2	4334.5	0.255
X16	1.411	68.52	-6331.8	8259.4	0.073	0.246	0.665	1.624	-2.799	**0.004**
X17	13.80	507.32	-0.413	53433	1.452	2.116	3.701	14.11	7.669	0.571
X18	0.097	5.738	-517.48	649.23	0.006	0.060	0.151	0.119	-0.311	**0.001**
X19	0.156	48.69	-1578.7	9230.5	0.004	0.036	0.091	0.170	-0.109	0.799
X20	243.0	37545	-29.34	7.8e+06	15.41	35.15	63.72	251.7	71.83	0.831
X21	3.885	228.67	-1325	29907	0.908	1.045	1.204	3.988	1.355	0.665
X22	0.114	5.156	-431.59	681.54	0.000	0.062	0.150	0.131	-0.231	**0.002**
X23	0.139	48.33	-1578.7	9230.5	0.002	0.030	0.078	0.151	-0.112	0.809
X24	0.270	7.988	-463.89	831.66	0.021	0.155	0.356	0.298	-0.268	**0.002**
X25	0.393	12.89	-500.93	1353.3	0.021	0.155	0.356	0.446	-0.663	**0.000**
X26	1.264	66.22	-6331.8	8262.3	0.067	0.222	0.599	1.446	-2.732	**0.005**
X27	1107.9	35012	-259010	4.2e+06	0.045	1.084	5.139	1145.6	31.91	0.246
X28	6.003	153.46	-3829.9	21701	0.038	0.465	1.497	6.146	3.116	03387
X29	4.005	0.827	-0.886	9.698	3.495	4.014	4.520	4.015	3.800	**0.000**
X30	7.371	814.49	-6351.7	152860	0.0827	0.218	0.408	7.328	8.215	0.961
X31	0.177	48.75	-1495.6	9244.3	0.007	0.043	0.102	0.187	-0.033	0.841
X32	1162.6	95593	-9295.6	1.7e+07	0.002	0.030	0.078	913.05	6079.4	**0.016**

Feature descriptive statistics can be seen in Table 2a (for features 1-32), and Table 2b (for features 33-64). Generally, many of the features have significant outliers in both the minimum and maximum values which may create some problems when variable scaling. Table 2a and Table 2b also include the mean feature value grouped by bankruptcy outcome, with significance tests included. The test statistic for each was computed as follows:

$$\left(\frac{\sum x_1}{n_1} - \frac{\sum x_0}{n_0}\right) / \sigma \sim t_{n_1+n_0}$$

Table 2b

Descriptive Statistics – X33 to X64

ID	Mean	Std. Dev.	Min	Max	25%	50%	75%	Mean – 0	Mean – 1	Sig.
X33	8.636	118.99	-19.20	21944	2.820	4.626	7.803	8.565	10.03	0.583
X34	5.411	120.98	-1696	21944	0.006	0.061	0.150	5.400	5.637	0.930
X35	0.112	4.783	-431.59	626.92	0.006	0.061	0.150	0.131	-0.271	**0.000**
X36	2.911	62.98	-0.001	9742.3	1.101	1.643	2.421	2.916	2.808	0.939
X37	105.1	3058.4	-525.52	398920	1.142	3.096	11.41	107.0	63.99	0.646
X38	0.724	14.75	-479.91	1099.5	0.420	0.612	0.772	0.774	-0.266	**0.002**
X39	-0.289	39.26	-7522	2156.5	0.004	0.037	0.092	-0.114	-3.757	**0.000**
X40	2.147	56.03	-101.2	8007.1	0.053	0.177	0.652	2.143	2.225	0.948
X41	7.718	1398.8	-1234.4	288770	0.027	0.086	0.206	8.118	-0.037	0.795
X42	-0.143	15.99	-1395.8	2156.8	0.000	0.038	0.092	-0.149	-0.022	0.725
X43	1074.1	147218	-115870	3.0e+07	66.61	99.40	140.7	1117.2	220.6	0.786
X44	831.1	831.1	-115870	2.3e+07	34.88	54.77	80.52	865.6	148.8	0.772
X45	14.83	2428.2	-256230	366030	0.019	0.283	0.956	15.34	4.330	0.846
X46	5.429	295.4	-101.26	53433	0.607	1.027	1.911	5.554	2.948	0.694
X47	357.8	33146	-96.11	6.1e+06	16.22	38.13	70.34	371.9	82.59	0.697
X48	0.029	5.097	-542.6	623.9	-0.038	0.018	0.107	0.047	-0.331	0.001
X49	-0.483	45.15	-9001	178.9	-0.027	0.011	0.062	-0.487	-0.396	0.928
X50	5.835	307.4	-0.045	53433	0.775	1.222	2.208	5.967	3.242	0.693
X51	0.484	5.438	-0.187	480.9	0.190	0.341	0.535	0.441	1.324	**0.000**
X52	6.478	639.9	-25.47	88433	0.127	0.214	0.350	6.739	1.369	0.708
X53	23.77	1213.8	-3828.9	180440	0.687	1.205	2.224	22.48	49.82	0.324
X54	24.65	1220.9	-3828.9	180440	0.956	1.377	2.370	23.35	50.99	0.322
X55	7672.2	70053	-1.8e+06	6.1e+06	27.55	108.8	499.3	8020.5	790.53	**0.000**
X56	-26.22	5327.9	-1.1e+06	293.2	0.009	0.053	0.129	-27.36	-3.710	0.843
X57	-0.011	13.67	-1667.3	552.64	0.015	0.120	0.285	0.049	-1.178	**0.000**
X58	30.02	5334.5	-198.7	1.1e+06	0.875	0.951	0.993	31.49	1.094	0.800
X59	1.333	122.1	-328.0	23853	0.000	0.006	0.236	1.370	0.606	0.780
X60	448.1	32346	-12.44	4.8e+06	5.546	9.792	20.18	463.3	135.5	0.665
X61	17.03	553.0	-12.66	108000	4.510	6.636	10.39	17.12	15.32	0.885
X62	1502.3	139267	-2.3e+06	2.5e+07	42.14	71.33	117.2	1505.9	1432.14	0.981
X63	9.343	124.2	-1.543	23454	3.098	5.088	8.600	9.367	8.872	0.859
X64	72.79	2369.3	-10677	294770	2.177	4.283	9.776	71.03	108.3	0.491

Where $\sum x_i / n_i$ is the grouped mean for the bankruptcy group and non-bankruptcy group, and the significance level is determined from a Student T distribution with $n_1 + n_0$ degrees of freedom. Variance σ^2 is either the total dataset variance or the pooled variance, based on the result of the equal variance statistical test:

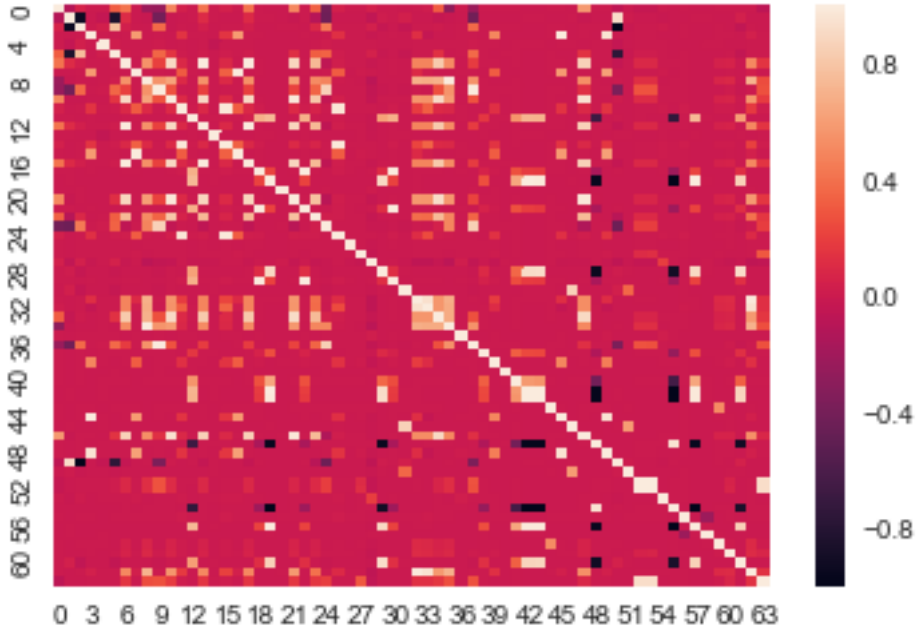
$$\frac{\sigma_0^2}{\sigma_1^2} \sim F_{n_0-1, n_1-1}$$

Where σ_0^2 and σ_1^2 are the grouped variances for the non-bankrupt and bankrupt groups respectively. The resulting test statistic is F-distributed with $n_0 - 1, n_1 - 1$ degrees of freedom. The null hypothesis is that the group has equal variance.

Overall, 23 of the 64 features had a statistically significant difference in means between the bankrupt and non-bankrupt classes.

By examining the correlation matrix in Figure 1, we can also see some relationships between variables. The majority of correlations are quite low, between +0.2 and -0.2, though a handful exhibit very strong positive or negative correlations. This is not unexpected, given how the features were calculated. For instance, Feature X33 is the ratio of operating expenses to short-term liabilities, while X34 is the ratio of operating expenses to total liabilities. Given that short-term liabilities are a component of total liabilities, and indeed may be proportional, it is not surprising that the ratios of X33 and X34 are highly correlated.

Figure 1: Correlation matrix of features.



To establish a benchmark, a simple random forest model was fitted with GridCV Search to tune the

hyperparameters. Overall a max depth of 24, minimum leaf sample size of 42, and a class weight of 50 times for bankrupt companies was determined to give optimal performance. This same parameterization is reused in subsequent random forest-based models by default unless superior parameters are determined. Overall, this model gave a validation ROC AUC of 0.80, with a recall of 0.74 and precision 0f 0.21.

Additionally, we can use the results of this model to examine feature importance. Notably feature 27, 24, 46, and 13 stand out as highly important.

Figure 2: Feature importance based on simple random forest.

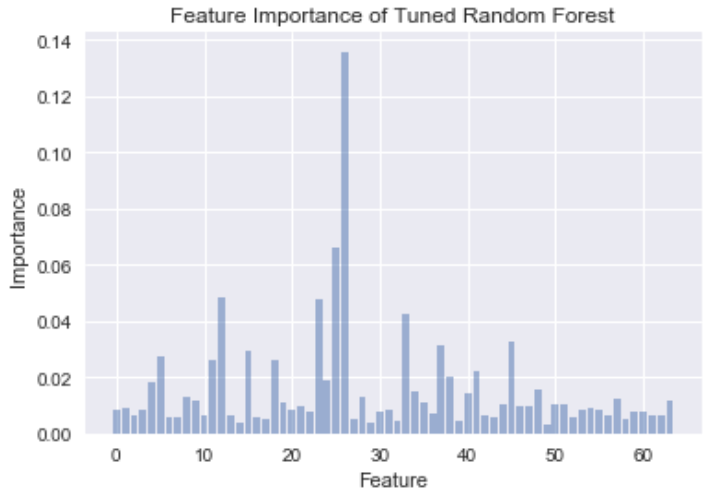


Table 4

Ranking of Feature Importance

Rank	ID	Importance	Rank	ID	Importance
1	27	0.136	11	9	0.022
2	24	0.066	12	5	0.020
3	46	0.049	13	17	0.019
4	13	0.048	14	16	0.018
5	26	0.043	15	55	0.016
6	6	0.033	16	38	0.015
7	34	0.031	17	29	0.014
8	12	0.030	18	21	0.013
9	25	0.028	19	56	0.013
10	10	0.026	20	35	0.013

III. Methodology

3.1 Data Processing

PCA

With such a large feature set principle component analysis (PCA) could be applied to reduce the feature size.

Applying PCA to the dataset, much of the variance fails to be explained adequately. The first 20 components explain only 54% of variance, and even 40 components improve the explained variance to only 63%. As show in Table 4, there is a rapid diminishment of explained variance through subsequent principle components. Considering this result, I do not use any dimensionality reduction in this paper.

Table 5

Explained Variance of PCA

Component	Explained Variance	Component	Explained Variance
1	9.79%	11	1.22%
2	8.81%	12	1.19%
3	4.83%	13	1.13%
4	3.28%	14	1.04%
5	3.09%	15	1.00%
6	2.78%	16	1.00%
7	2.49%	17	1.00%
8	2.24%	18	1.00%
9	1.78%	19	0.99%
10	1.51%	20	0.96%

Outlier Removal

Based on the data exploration conducted in Table 2, I consider the exclusion of outliers to potentially improve performance. The outlier removal criteria are specified by determining the Xth and 100-Xth percentiles for each feature, and then removing any rows containing data points with magnitudes more than 50 percent smaller/greater than the lower/upper bound specified.

Table 6: Random forest model benchmark performance

Outliers	Sample	Positives	Train			Validation		
			ROC AUC	Recall	Precision	ROC AUC	Recall	Precision
Default	43405	4.82%	0.91	0.97	0.25	0.79	0.74	0.21
5/95%	31242	4.36%	0.93	0.97	0.28	0.81	0.75	0.22
2.5/97.5%	36120	4.50%	0.93	0.97	0.28	0.82	0.77	0.23
1/99%	39895	4.65%	0.92	0.97	0.26	0.84	0.82	0.21
0.5/99.5%	41398	4.67%	0.92	0.97	0.28	0.82	0.76	0.24

Compared to the benchmark model, exclusion of outliers categorically increases performance across all measures. In addition, fewer removed data points generally results comparable or better performance, while also maintaining a proportion of bankrupt companies closest to the full sample.

Variable Scaling & Missing Values

As noted in the data analysis section, roughly 1.5% of the dataset contains missing values. Missing values were interpolated based on the given feature mean from the entire dataset. Processing order was experimented with by conducting the mean value interpolation before or after outlier removal, but this does not noticeably affect the results of the descriptive analysis performed in Table 2, model performance in Table 5, or the PCA results shown in Table 4.

As a final step before fitting any models, the features were scaled to a magnitude between 0 and 1 using the SKLearn preprocessing package. The resulting features were all scaled to have zero mean and unit variance, in line with standard pre-processing practices. (citation needed)

Implementation

3.2 Support Vector Machine

SVM implementation is borrowed from Shin, Lee & Kim (2005) and Min & Lee (2005) who both use a kernel-based approach with radial basis functions (RBF). For sake of thoroughness, I also examine using both a linear SVM and a polynomial kernel SVM model to compare the results. All three model types were hyperparameter tuned with Gridsearch CV. The RBF kernel model tuned both the C and gamma (bandwidth) parameters, while the polynomial kernel model tuned C, gamma, and the degree of polynomial as well.

The parameter tuning for SVMs take a significant amount of time, and the results were frequently non-convergent even with 100,000 iterations. I do not expect the parameter results to be greatly

affected by additional iterations, however, as tuning with both 100,000 and 20,000 iterations yielded the same parameter recommendation.

Table 7: SVM Train and Validation performance.

	Train			Validation		
SVM Model	ROC AUC	Recall	Precision	ROC AUC	Recall	Precision
Linear	0.73	0.62	0.16	0.75	0.66	0.17
Polynomial Kernel	0.80	0.82	0.16	0.81	0.84	0.17
RBF Kernel	0.93	0.95	0.37	0.86	0.80	0.32

Table 8: RBF Kernel SVM hyperparameter tuning.

		Gamma							
		2^-1	2^-3	2^-5	2^-7	2^-9	2^-11	2^-13	2^-15
C	2^-1	0.555	0.664	0.759	0.795	0.754	0.719	0.700	0.686
	2^1	0.549	0.651	0.751	0.796	0.786	0.748	0.715	0.700
	2^3	0.549	0.645	0.750	0.802	0.815	0.765	0.740	0.715
	2^5	0.548	0.643	0.733	0.806	0.823	0.797	0.758	0.736
	2^7	0.548	0.645	0.727	0.805	0.824	0.823	0.776	0.756
	2^9	0.550	0.650	0.726	0.798	0.829	0.832	0.809	0.765
	2^11	0.550	0.655	0.724	0.787	0.832	**0.841**	0.829	0.785
	2^13	0.550	0.660	0.721	0.785	0.832	0.839	0.839	0.798

3.3 Neural Network

The neural network architecture is mainly inspired by Kim & Kang (2010), who use a single hidden layer model with nodes roughly doubling the feature count. They also find that more complicated boosted models do not noticeably improve performance. To replicate this, I initially fit a simple neural network with a single hidden layer with 128 nodes, relu activation and 100 iterations. To compensate for the class imbalance in the data, a class weight of 25 is applied to the bankrupt class to roughly obtain balance. The model achieves a validation ROC AUC of 0.83, a recall of 71%, and a precision of 47%.

To verify the results of Kim & Kang (2010), I also fit a deeper layered network. This network has 4 hidden layers of 128 nodes with relu activation, and 10% dropout layers woven in between. Fitted with 100 iterations, the model achieves a validation ROC AUC of 0.96, 81% recall and 49% precision. This reflects the expected result that a deeper network leads to only a marginal improvement in performance.

Finally, I also fit a 24-network bagged model with the single layer model from above. 24 separate models are fitted on training subsets sampled with replacement from the primary training subset. Each model follows the same 128 node single hidden layer architecture used above. Classification is based on voting results from the models.

Table 9: Neural network model train and validation results.

	Train			Validation		
NN Model	ROC AUC	Recall	Precision	ROC AUC	Recall	Precision
Single Layer	0.99	0.99	0.62	0.93	0.76	0.47
Deep Layer	0.99	0.99	0.61	0.96	0.81	0.49
Single Layer Bagging	0.99	0.99	0.70	0.95	0.75	0.54

The most significant challenge with a neural network implementation is the extremely wide range of possible model configurations that could be used. The purpose of fitting both a single layer and deep layered model is to show that the model results are relatively insensitive to changes in model architecture. Nonetheless, better performing architectures may exist.

3.4 Gradient Boosting

Gradient boosting can be uses composite sequential models to fit more accurate models quickly (Friedman 2001). As an analog to model bagging, which parallelizes the fitting process by fitting many models simultaneously, boosting fits many sequential models to derive a best fitting model. First consider some model f_0 with loss function based on target y, $L_y(\theta, f(\mathbf{x}))$, with parameters θ and input $\mathbf{x}$. We would then derive the model as follows

$$f_0(\mathbf{x}) = \text{argmin} L(\theta, f(\mathbf{x}))$$

Predictions and errors can then be calculated

$$\bar{y}_1 = f_0(x)$$
$$e_1 = y - \bar{y}_1$$

Now, a subsequent model f_1 is fitted with the errors of the first model as the target, and calculate resulting predictions

$$f_1(\mathbf{x}) = \text{argmin}_{e1} L(\theta, f(\mathbf{x}))$$
$$\bar{e}_1 = f_1(x)$$
$$\bar{y}_2 = \bar{y}_1 + \bar{e}_1$$

And finally, new residuals can then be computed from the resulting predictions

$$e_2 = y - \bar{y}_2$$

This process repeats until stopping. Generally, a stopping rule based on cross validation scores is used, for instance stopping when an improvement in cross validation performance has not been observed in at least 10 iterations.

There are several python packages that implement the gradient boosting framework, but I choose to use three: the base implementation in SKLearn, XGB, and a recent framework, LightGBM, introduced by Ke et al. (2017).

The SKLearn and XGB implementations are parameterized in the same way as the base random forest model discussed earlier (24 max depth, 42 min leaf sample). LightGBM uses the default parameters. XGB and LightGBM, in particular, are highly efficient models that are quite robust to overfitting, so there was no difficulty with implementation. I also opted to use the SKLearn wrapper for XGB to maintain familiar syntax.

Table 10: Gradient boosted model results compared to base random forest model.

	Train			Validation		
Model	ROC AUC	Recall	Precision	ROC AUC	Recall	Precision
Base Model	0.91	0.97	0.25	0.79	0.74	0.21
Outliers Removed	0.92	0.97	0.28	0.82	0.76	0.24
SKLearn Grad Boost	1.00	1.0	1.0	0.77	0.54	0.94
XGB	0.73	0.47	1.0	0.71	0.41	0.99
LightGBM	0.99	0.95	1.0	0.97	0.59	0.97

3.5 Refinement

LightGBM Tuning

The overall best performer based on ROC AUC is the LightGBM model, but it generally prioritizes accuracy in precision over accuracy in recall. By tuning some of the parameters, it may be possible to improve the recall accuracy or balance the two scores.

The LightGBM model does not function well with GridSearchCV, so I manually tune the parameters instead. In the new model, the 'max_bin' parameter is quadrupledfrom 255 to 1023, 'num_leaves' is increased from 31 to 47, the learning_rate is reduced from 0.05 to 0.01, these

parameters boost accuracy. Next, a subsampling ratio of 0.75 is used to prevent overfitting. Finally, a balancing class weight of 15 is added to the bankrupt class. Overall validation ROC AUC remains almost constant at 0.97, but validation recall is boosted to 0.74 at the cost of precision, which drops to 0.65.

Ensemble Method

With several models performing better on the recall or precision metrics, it may be useful to combine these models in to an ensemble. Specifically, combining the Bagged neural network model, the XGB random forest model, and the two LightGBM models. The models each make a prediction and the net prediction is based on the weighted average of the models. The XGB random forest outputs a discrete prediction, which is then underweighted to 0.50 to avoid dominating the probability predictions of the other 3 models.

Overall, the model has a training ROC AUC of 99%, and validation ROC AUC of 97%. The validation recall and precision are 0.61 and 0.91 respectively. The model outperforms the balanced LightGBM model in terms of balanced recall and precision, but fails to outperform the first LightGBM in terms of ROC AUC. Part of this underperformance can be explained by examining the prediction biases of the component models. On average, the ensemble model receives the largest probability contribution from the balanced LightGBM and bagged neural network models, both of which fail to outperform the base LightGBM model in terms of ROC AUC, despite reasonable performance in 50%-threshold recall and precision.

As an alternative where false positive are of less concern, I also fit a voting ensemble model which predicts a positive if any of the component models votes positively. This alternate model achieves a training and validation ROC AUC of 99% and 89% respectively, along with perfect training recall and 0.62 training precision. Validation performance is similar, with 0.51 precision and 0.84 recall. This model has better recall than the gradient boosted models, and comparable recall to the neural network and SVM models but with better precision

Acceptance Threshold

As one final experiment, I consider a revised acceptance threshold for the best performing probability-based models, the bagged neural network, LightGBM models, and probability ensemble model. Up to this point, a 50% acceptance threshold has been used when describing the recall and precision for each model. For future testing and predictions where the result is not known, it may be useful to use a different threshold.

The sample data has a bankruptcy rate of 4.82% (4.67% with outliers removed), so I suggest setting the probability threshold such that 5% of the validation sample is predicted positive, rounded to 3 digits, as well as a more conservative option at a 10% threshold. Empirically, the bagged neural network overestimates the number of bankrupt companies, so changing the probability threshold should reduce false positives. The LightGBM model, by comparison, strongly underestimates the frequency of bankruptcy, so increasing the threshold may reduce false negatives.

The advantage of using a percentile determined probability rather than using the percentile itself is to handle variance in the bankruptcy proportion present in a sample. For instance, suppose that economic times are particularly hard such that an unusually high proportion, e.g. 10%, of companies in a sample go bankrupt. If a percentile cutoff were used, such as 5%, then mathematically even a perfect model would only score at most 50% recall. By using a percentile-derived cutoff instead, the hope is to derive a confidence level representing the certainty of the model with respect to the prediction. The value of this approach can be demonstrated by simply duplicating each bankrupt sample: a percentage cutoff would, again, achieve at most only a 50% recall, where as the percentile derived acceptance threshold would simply predict the same class for each pair of duplicates, which would leave the recall unchanged.

Table 11: Validation prediction counts of ensemble and underlying models.

Model	# Predicted	False Positive	False Negative
Bagged NN	553	255	102
LightGBM	243	9	166
LightGBM Balanced	455	161	106
Probability Ensemble	276	14	148
Voting Ensemble	666	335	69
Actual	400		

The 5th percentile probability threshold for the 4 models are 61.3%, 12.7%, 54.8%, and 34.6%, for the bagged neural network, LightGBM 1, LightGBM Balanced, and the ensemble method. The 10th percentiles are 6.5%, 8.3%, 47.9%, and 25.2% respectively.

The results (Table 12) for both show a decrease in false negatives except for the bagged neural network, with the probability ensemble and LightGBM models proving to be especially accurate.

Table 12: Prediction sensitivity with adjusted acceptance threshold.

	5% Cutoff			10% Cutoff		
Model	# Pred	False Pos	False Neg	# Pred	False Pos	False Neg
Bagged NN	415	144	129	828	501	73
LightGBM	415	115	100	828	486	58
LightGBM Balanced	414	132	118	830	489	59
Probability Ensemble	414	117	103	830	481	51
Actual	400					

IV. Results

4.1 Model Evaluation and Validation

Table 13: Test results compared to validation.

	Validation			Test		
Model	ROC AUC	Recall	Precision	ROC AUC	Recall	Precision
Base Model	0.79	0.74	0.21	0.49	0.15	0.04
SVM Linear	0.75	0.66	0.17	0.74	0.64	0.16
SVM Poly Kernel	0.81	0.84	0.17	0.81	0.83	0.16
SVM RBF Kernel	0.86	0.80	0.32	0.86	0.81	0.30
Single Layer NN	0.92	0.71	0.47	0.92	0.74	0.42
Deep Layer NN	0.96	0.84	0.41	0.95	0.78	0.46
Bagged NN	0.95	0.75	0.54	0.95	0.72	0.51
SKLearn Grad Boost	0.77	0.54	0.94	0.76	0.53	0.91
XGB	0.71	0.41	0.99	0.69	0.39	1.0
LightGBM	0.97	0.59	0.97	0.97	0.53	0.95
LightGBM Balanced	0.97	0.74	0.65	0.97	0.73	0.63
Probability Ensemble	0.97	0.63	0.91	0.97	0.90	0.34
Voting Ensemble	0.88	0.78	0.57	0.90	0.83	0.47

The clear best models are the two LightGBM models alongside the probability ensemble model, with the bagged neural network and deep layered neural network also performing well. To verify the results, I bootstrap the confidence intervals by random sampling the prediction vector for each model and recalculating ROC AUC 1000 times.

As expected, all 5 of the best performing models very significantly outperform the base model (Table 14). In addition, the two LightGBM models and the ensemble model also outperform the neural network models at a <1% significance level.

Table 14: 95% Confidence Interval for select models

Model	Lower Bound	Upper Bound
Base	0.474	0.511
Deep NN	0.942	0.962
Bagged NN	0.936	0.964
LightGBM	0.967	0.980
LightGBM Balanced	0.964	0.976
Probability Ensemble	0.967	0.980

4.2 Justification

To add some credibility to the model, we can add the probability thresholds derived in section 3.5 to see how usefully the models perform with respect to minimizing false negative or false positive rates.

Table 15: Test sample demonstration of lowered acceptance thresholds. P is the acceptance threshold for each model.

	5% Cutoff				10% Cutoff			
Model	P	# Pred	False Pos	False Neg	P	# Pred	False Pos	False Neg
LightGBM	0.127	381	112	99	0.083	492	192	78
LightGBM Balanced	0.548	394	133	117	0.479	453	180	100
Probability Ensemble	0.346	1302	946	22	0.252	1628	1270	20
Actual		378						

In the case of the test set, the probability ensemble significantly over-predicts at both the 5% and 10% probability thresholds, although this may be useful if there is a large cost to mis-classifying bankrupt companies. This over-prediction is not surprising given that the ensemble method already achieves a 90% recall rate with the 50% threshold in the test set, so there is not much room for improvement. Both LightGBM models perform more reasonably, with the base LightGBM model performing better with respect to both false negatives and false positives.

Beyond these tests, it is difficult to conduct many robustness checks without introducing bias from resampling the data. Nonetheless, the test results for many of the models are generally quite close to the validation results, which gives credibility towards the robustness of these results.

Conclusion

The above test results can be re-presented as the raw number of correct predictions, false positives, and false negatives (Table 16).

Considering these results, the optimal model to use for real world applications depends on the objective. For instance, most investors will benefit from avoiding bankrupt companies, even if they miss the opportunity to invest in some non-bankrupt companies as well through the admission of false positives. The conservative probably ensemble model performs best in this regard, with a high number of correct predictions (90% recall) and a relatively small number of false positives, less than 2.1% of the sample

Table 16: Test set prediction counts of ensemble and underlying models.

Model	True Pred.	False Positive	False Negative	Total Pred.
Bagged NN	269	249	109	518
SVM RBF Kernel	306	717	72	1023
XGB	147	0	231	147
LightGBM	202	10	176	212
LightGBM Balanced	276	102	161	437
Probability Ensemble	189	7	189	541
Voting Ensemble	289	252	89	196
			Actual	378

On the other hand, for a short investor – an individual who's investing position benefits from a company losing value or going bankrupt – or a government regulator with limited resources, accurate prediction of bankruptcies with minimum false positives will be the most valuable. For this, the LightGBM models clearly present the best performance with nearly perfect precision in both validation and testing. Further analysis of the ROC curve for the LightGBM model is visible in Figure 3.

Additionally, we can examine the feature importance of the LightGBM model to get some intuition on which features were the most relevant (Figure 4). The results generally resemble the feature importance shown in Figure 2, but with a somewhat reordered listing. Interestingly, Feature X5 (Column 4) surfaces as the most important feature, despite being relatively unimportant in the base model analysis, Features X58 (Column 57) is entirely absent from the top 20 feature importance ranking. Nonetheless, many of the important features in the base model, such as X46, X34, and X27 remain present in the model at high importance levels.

Figure 3: LightGBM ROC Curve from Test Set.

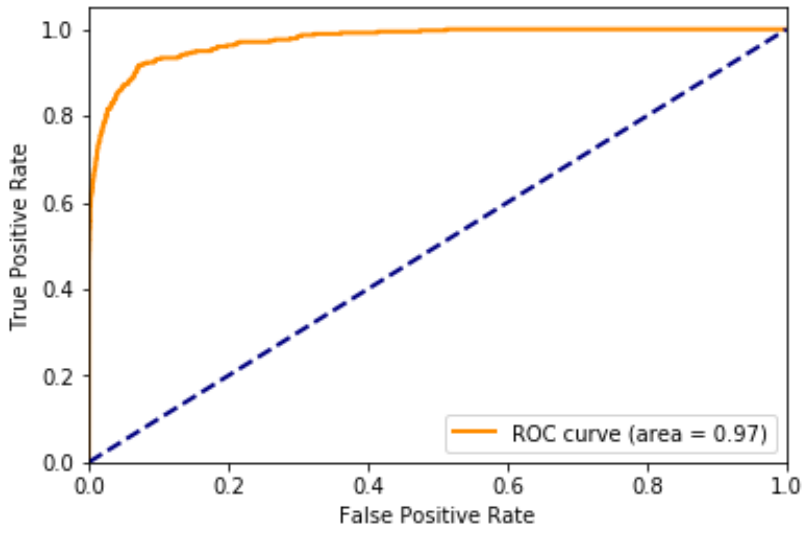
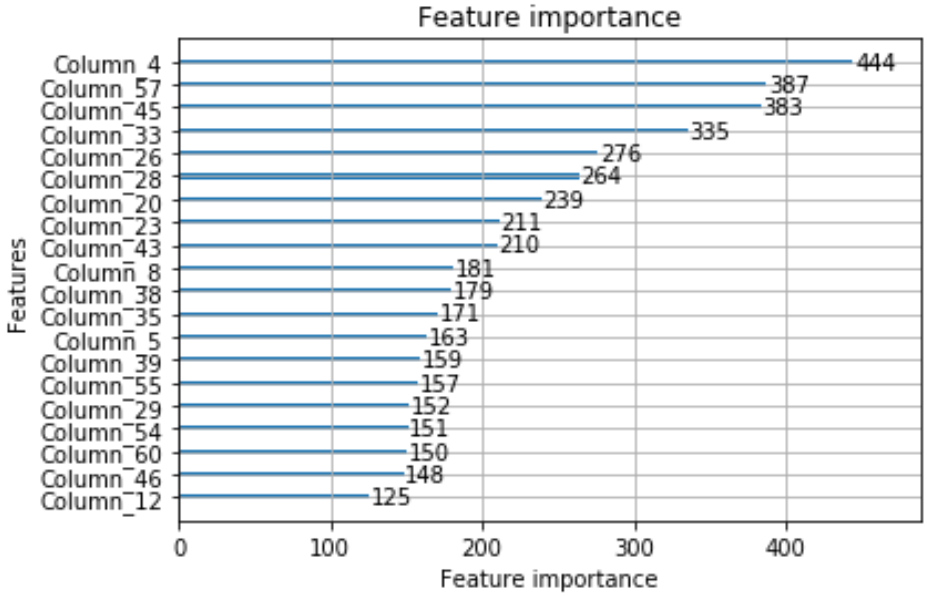


Figure 4: Feature importance of LightGBM model.



5.1 Reflection

By far the most difficult part of this project was collecting the variety of models used in this paper and ensuring they are properly parameterized using outside research materials. Additionally, transforming all the model outputs to be standardized and easy to analyze was problematic.

Additionally, a key challenge was deciding on accurate performance metrics. The use of ROC AUC was decided early, but I went back and forth on the best way to demonstrate model accuracy through precision and recall. Finally deciding to use the additional 10 and 5% cutoffs for the results section was a breakthrough.

Finally, despite outperforming comparable papers, the best performing model, LightGBM, was published in 2017. This gives me a strong appreciation for how crucial model research is for achieving improved results in empirical problems. I would not have been able to out perform the data source if not for the LightGBM model, which was published almost a year after the data source used in this project.

5.2 Improvement

Some of the models fitted in this project have a very high overall performance, and even outperform the data source, Zieba, Tomczak & Tomczak (2016), at a statistically significant level. However, it is still not possible to achieve a high recall rate without the admission of around 30-40% precision, so there is some room for improvement. Additionally, it is possible that the Polish data happened to be more conducive to bankruptcy predicting for structural economic reasons. The methods I have explored may not generalize to other geographies or specific subindustries, so future research may be required. Nonetheless, I believe the models detailed here present strong potential for more accurate bankruptcy detection.

On the implementation side, a significant amount of time was spent on parameter tuning through GridSearchCV, even though many of the parameter ranges were informed by outside research. This process could be improved by implementing smarter parameter tuning, for instance by using early stopping. Additionally, often the parameter tuning results would be strictly decreasing away from the optimal parameter choice, so cutting off parameter search directions could save time.

Works Cited:

Abellán, J., & Mantas, C. J. (2014). Improving experimental studies about ensembles of classifiers for bankruptcy prediction and credit scoring. *Expert Systems with Applications*, *41*(8), 3825-3830.

Altman, E. I. (1968). Financial ratios, discriminant analysis and the prediction of corporate bankruptcy. *The journal of finance*, *23*(4), 589-609.

Altman, E. I., Haldeman, R. G., & Narayanan, P. (1977). ZETATM analysis A new model to identify bankruptcy risk of corporations. *Journal of banking & finance*, *1*(1), 29-54.

Beaver, W. H. (1966). Financial ratios as predictors of failure. *Journal of accounting research*, 71-111.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine learning*, *20*(3), 273-297.

Fan, A., & Palaniswami, M. (2000, May). A new approach to corporate loan default prediction from financial statements. In *Proceedings of the computational finance/forecasting financial markets conference, London (CD), UK*.

Fawcett, T. (2006). An introduction to ROC analysis. *Pattern recognition letters*, *27*(8), 861-874.

Fernández, E., & Olmeda, I. (1995, June). Bankruptcy prediction with artificial neural networks. In *International Workshop on Artificial Neural Networks* (pp. 1142-1146). Springer, Berlin, Heidelberg.

Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of statistics*, 1189-1232.

Iturriaga, F. J. L., & Sanz, I. P. (2015). Bankruptcy visualization and prediction using neural networks: A study of US commercial banks. *Expert Systems with applications*, *42*(6), 2857-2869.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., ... & Liu, T. Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems* (pp. 3146-3154).

Kim, M. J., & Kang, D. K. (2010). Ensemble with neural networks for bankruptcy prediction. *Expert systems with applications*, *37*(4), 3373-3379.

Martinelli, E., de Carvalho, A., Rezende, S., & Matias, A. (1999, January). Rules extractions from banks' bankrupt data using connectionist and symbolic learning algorithms. In *Proc. Computational Finance Conf.*

McKee, T. E., & Greenstein, M. (2000). Predicting bankruptcy using recursive partitioning and a realistically proportioned data set. *Journal of forecasting*, *19*(3), 219-230.

Min, J. H., & Lee, Y. C. (2005). Bankruptcy prediction using support vector machine with optimal

choice of kernel function parameters. *Expert systems with applications*, *28*(4), 603-614.

Min, S. H., Lee, J., & Han, I. (2006). Hybrid genetic algorithms and support vector machines for bankruptcy prediction. *Expert systems with applications*, *31*(3), 652-660.

Odom, M. D., & Sharda, R. (1990, June). A neural network model for bankruptcy prediction. In *Neural Networks, 1990., 1990 IJCNN International Joint Conference on* (pp. 163-168). IEEE.

Ohlson, J. A. (1980). Financial ratios and the probabilistic prediction of bankruptcy. *Journal of accounting research*, 109-131.

Piramuthu, S. (1999). Feature selection for financial credit-risk evaluation decisions. *INFORMS Journal on Computing*, *11*(3), 258-266.

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*(1), 127-135.

Tsai, C. F., & Wu, J. W. (2008). Using neural network ensembles for bankruptcy prediction and credit scoring. *Expert systems with applications*, *34*(4), 2639-2649.

Wilson, R. L., & Sharda, R. (1994). Bankruptcy prediction using neural networks. *Decision support systems*, *11*(5), 545-557.

Piramuthu, S. (1999). Feature selection for financial credit-risk evaluation decisions. *INFORMS Journal on Computing*, *11*(3), 258-266.

Zięba, M., Tomczak, S. K., & Tomczak, J. M. (2016). Ensemble boosted trees with synthetic features generation in application to bankruptcy prediction. *Expert Systems with Applications*, *58*, 93-101.

Appendix A – Polynomial Kernel SVM Parameter Sensitivity

Validation Score (ROC AUC):

		gamma			
	C = 1	2^-1	2^-3	2^-5	2^-7
degree	1	0.524	0.784	0.770	0.761
	5	0.493	0.501	0.512	0.498
	10	0.478	0.477	0.482	0.474
	C = 5				
degree	1	0.471	0.525	0.787	0.766
	5	0.491	0.498	0.510	0.516
	10	0.478	0.482	0.482	0.475
	C = 10				
degree	1	0.496	0.476	0.537	**0.790**
	5	0.490	0.495	0.513	0.571
	10	0.478	0.484	0.484	0.476
	C = 15				
degree	1	0.519	0.547	0.489	0.788
	5	0.490	0.500	0.504	0.615
	10	0.478	0.483	0.478	0.489