Data Mining I: Bankruptcy Detection

Group 9: Project Report

presented by
Amela Medolli
Matriculation Number 1729253
Mert Özlütiras
Matriculation Number 1727644
Katharina Prasse
Matriculation Number 1721782
Florian Rüffer
Matriculation Number 1500176

submitted to the
Data and Web Science Group
Prof. Dr. Heiko Paulheim
University of Mannheim

December 2020

1 Application area and goals

Bankruptcy is the phenomenon that a company or firm is incapable of paying its debts. In case of a bankruptcy, all the stakeholders of the company suffer from it. Bondholders stop receiving payments, stockholders stop receiving their dividend and face huge losses in stock prices, many employees lose their jobs. The larger the company is that is going bankrupt, the more sever its effects are. Banks, credit rating institutions, even governments and many 3rd parties can be affected by a corporation's bankruptcy. Investment decisions, mergers acquisitions and worker union strategies highly depend on a companies future financial status, to just name a few examples.
Due to the high importance of estimating the future financial status of a company and its operation prospects, an early detection system is really valuable, allowing to take corrective actions in the best case. Hence, we did this study that digs deeper into Polish companies' financial status with the aim of creating an early detection system. In our study, we had a classification task to predict the target variable showing whether a company would go bankrupt(1) or not(0).
Our primary goal was to develop a preventive and predictive system to forecast the financial condition of bankruptcy by comparing different machine learning models and choosing the best model performance. Furthermore, we wanted our model to be able to predict the bankruptcy from 1-5 years before the company actually goes bankrupt. In order to achieve our goal, we applied different machine learning algorithms together with many different preprocessing methods in order to find the best model in identifying companies that are going to go bankrupt.

Keywords: Bankruptcy, preventive system, classification, F2 score

2 Data Understanding

The source of the analyzed data sets is the UCI Irvine Machine Learning Repository[2] and is based on data from the Emerging Markets Information Service [EMIS]. The data sets describe the financial situation of Polish companies. A total of five data sets were used, each of them containing the same 64 financial indicators, thus 64 attributes. Different data sets contain bankruptcy predictions after five, four, three, two and one years respectively. Data set 1 contains financial data from year 1 and the bankruptcy status after five years, data set 2 contains financial data from year 2 and the bankruptcy status after four years and the other sets follow the same logic. In each data set, a different number of companies is included. Even though it is highly likely that the same companies are included in multiple data sets, it is not

possible to identify them as such as the companies are not named and each data set contains a different number of records. Thus, a time series could not be created and the observations are treated as separate companies. All rows are labeled with a binary class value, giving information if a company actually went bankrupt (1) or not (0) after the respective prediction time. As one of our main goals was to create and train a model that is generalizable, we chose to train on the third year data set as it presumably gives the best predictions when tested for other years too. This data set's financial indicators are expected to perform well for short-term and long-term prediction as well.
All attributes are numeric and real-valued, and have the data type float. They are shortly explained and provide information about the financial and operational situation of each company. They consist of financial performance indicators and calculations thereof. Among many others; total assets, total sales, the relative amount of sales in period n compared to period n-1, working capital and liabilities are used. Exact description of each attribute can be found at UC Irvine Machine Learning Repository[2]. As expected, in terms of class distribution the data is highly unbalanced, having bankrupt companies as minority class. The exact distribution of classes in the 3rd-year data set is shown in Figure 1.

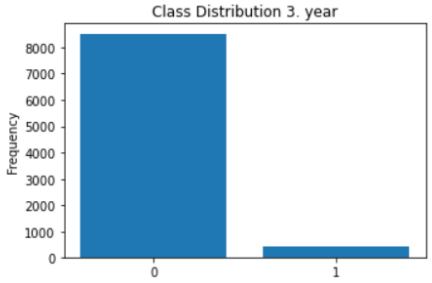

Figure 1: Class distribution of 3rd year data set.

The quality of the data is high, with only few missing values [1st year (1,3%), 2nd year (1,9%), 3rd year (1,5%), 4th year (1,4%), 5th year (1,2%)]. Given the fact that many of the attributes have similar descriptions, we expect to find some correlations in the data or even redundant attributes.

3 Preprocessing

Firstly, all data sets have been split into train and test sets (85-15). All missing values have been replaced by the respective train set's mean value of the given at-

tribute. Not all ML algorithms require complete examples, however, the goal was to use the same data sets for all algorithms to ensure a maximum comparability of results. The data sets have been checked for outliers using histrograms, box plots and DB Scan. None of the tests indicated that there are a substantial number of outliers in the data set and thus no action was taken. This decision should be kept in mind when discussing the results. Validity checks looked at the several attribute's minimum and maximum values and set them in context using domain knowledge. No invalid values were found. Furthermore, inconsistencies between several attributes were checked in comparing the magnitudes of several attributes using domain knowledge. Again, no inconsistencies were found. Lastly, duplicated examples were not excluded, due to the low and thus insignificant number of them in the data set.

For all data sets, both the train and the test set have been normalized using z-scores based on the respective train set mean and standard deviation values for each attribute. No additional features have been generated, as the existing features covered all relevant information.

Secondly, we adjusted the train set individually for each machine learning algorithm to optimize its performance as described in the following part. We aim to improve the models performance by balancing, due to the highly skewed class distribution. More precisely, the methods RandomOversampling, RandomUndersampling and SMOTE sampling are employed. Whereas in over- and undersampling existing examples are duplicated or removed respectively, in Smote sampling new examples are generated using an approach similar to KNN [1].

Furthermore, since the attribute descriptions imply strong correlation between several attributes, a correlation matrix is used to visualize the data sets Pearson correlation coefficients. If correlations between the attributes exist, as expected, then a feature selection makes sense. For feature selection, two different methods are compared: removing all features below an importance threshold and removing all features that are correlated to the N most important features. For each feature selection method we use either different thresholds or different numbers of most important features (N) respectively when optimizing it for each ML algorithm. In the second method, correlation was always assumed when two attributes Person correlation coefficient was above corr = 0.5. Finally, for each algorithm, the effect of binning was discussed and hyper-parameter tuning was completed where applicable.

4 Data Mining

4.1 ML approaches

In total five machine learning algorithms were used. We classified with two lazy classification algorithms, K Nearest Neighbours and Naïve Bayes. We chose these two to have both a very simple model and a probabilistic model in our analysis. Furthermore, we classified using three eager algorithms: two tree-based algorithms were chosen, Random Forest and XGBoost, and one neural network. The selection thus includes two symbolic approaches, the lazy algorithms, and three non-symbolic approaches, the eager algorithms. No external data sets were used, as the attributes from the given data set were providing a sufficient level of detail to create a meaningful classification.
The eager algorithms were expected to outperform the lazy ones. Among the three eager algorithms, the tree-based ones are expected to outperform the neural net, as the classification problem appears to be a deterministic rather than probabilistic one [3]. The Naïve Bayes model will serve as a baseline for the comparison among models.

4.2 Evaluation

A test set was split off at the beginning and the different ML algorithms are compared using a 85-15 train-test split. Given the fact that the data set includes almost nine thousand samples, this choice makes sense and is computationally cheaper than a 10 fold cross validation. The train-test split was implemented using stratification, which was especially relevant given the highly imbalanced classes.
Our goal is the maximize the correct prediction of bankruptcy. Given the fact that it is far worse to misclassify a soon-to-be-bankrupt company as financially healthy than the other way around, the evaluation is focused on the recall score, as it gives the fraction of actual bankrupt firms over all firms classified as bankrupt. In order to not allow the model to classify all examples as bankrupt (to get the highest recall), we evaluated the results using the F2 score, which is an alteration of the F1 score with a stronger weight for recall and a weaker weight on precision. Additionally, a confusion matrix is used to provide insight in the overall classification quality of the model. We did not use a cost matrix, as it is not possible to give specific weights to False-Positives (FP) and False-Negatives (FN) across companies and across different business applications of our predictive model.
For each algorithm individually, the best setting for balancing, feature selection, binning, and hyper-parameter tuning were implemented. For balancing and feature selection, the test were systematic and identical to allow for a high comparability

of the results. The hyper-parameters were tuned using grid-search to avoid bias to influence the result. For the neural network, the tests were more evolutionary due to the model complexity.
The feature importance analysis clearly shows that some attributes are better predictors than others. Employing the feature selection methods as described above, the most important features can be assessed. Using the Random Forest algorithm, profit on operating activities / financial expenses [Attribute 27] is the best predictor. Furthermore, operating expenses/total liabilities [Attribute 34] and (current assets - inventory)/short-term liabilities [Attribute 46] also provide significant information where a company will soon be bankrupt, as shown in Figure 2. Comparing the results to the one's from the Decision Tree algorithm, it is shown that they are similar. Two out of the top three important features are identical.

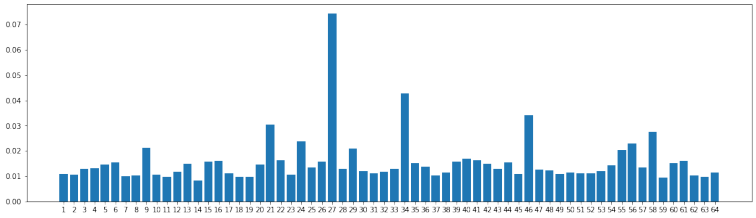

Figure 2: Features importances according to Random Forest algorithm.

In order to achieve some deeper understanding of the predictive power of the most important attributes and the decisions of rule based algorithms, we had a look at the first three splits of an optimized decision tree classifier without any feature selection. It is interesting to see that attribute 27 - profit on operating activities / financial expenses – which is the most important feature considering random forest, is used as first splitting criterion here, which emphasizes the importance of that feature. As we did not choose any balancing here, we can see that the splitting value is quite high, which is a result of the biasing towards the majority class. The splits on the second level are also in line with the random forests feature selection, as they are the second and third most important features respectively, as shown in Figure 3.

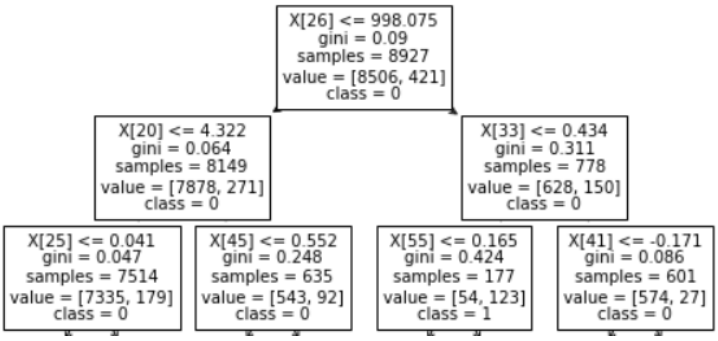

Figure 3: Decision tree first three splits

The problem of the bias towards the majority class can solved by either letting the tree grow larger (which is hard to show here) or using smote balancing. However, as we used fully grown random forest and decision tree for feature selection, these differences in rules and importances became minimal, which is the reason we stayed with feature selection based on the non-balanced data.

5 Results

In order to create the best classifier, each algorithm was first optimized and then compared to the baseline (Naïve Bayes). In general, a balanced training set was expected to perform better than a unbalanced one. Further, feature selection was expected to improve the results, whereas the random forest method was expected to yield better results due to its lower risk of overfitting compared to decision tree. If for two sets of selected features, the model's F2 scores were identical, the model with the lower total number of features was chosen in line with Occam's razor. Binning was expected to reduce the performance of the classifier, as it reduces the variation in the training data.

5.1 Naïve Bayes (Baseline)

The Naïve Bayes performed surprisingly badly for this classification problem. It is inexplicable for us, why the algorithm mainly predicted the minority class, as shown in Figure 4.The optimal settings for Naïve Bayes were Smote oversampling, feature selection using the decision tree algorithm and a threshold of 0.01 (27 features) and no binning. The F2 score improved from 0.19 to 0.21.

The superiority of smote was expected as it increases not only the number of

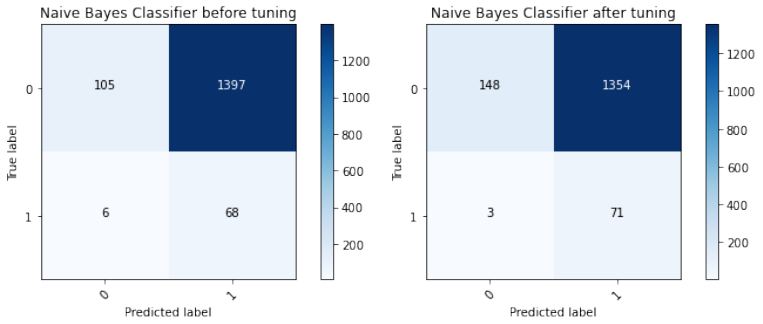

Figure 4: Gaussian Naive Bayes without (left) and with (right) optimization.

observations, but also increases their variation. Against our expectations regarding feature selection, decision tree's important features always outperformed the random forest's. It appears that the probabilistic Naïve Bayes models prefers non-probabilistic inputs. Binning reduced the F2 scores of the classifier as expected. Naïve Bayes did require neither normalization nor handling of missing values due to its attribute independence assumption.

5.2 K Nearest Neighbors

The KNN algorithm tended to predict the majority class. We tried different hyperparameters and preprocessing steps and it turns out that the best model for KNN was trained on underbalanced data, with feature selection using the random forest algorithm with a threshold of 0.03 and number of neighbours(K)=10. Additionally, KNN needed normalization and improved with replacing of the missing values. With these settings and hyperparameters, the F2 score increased from 0 to 0.37. However, binning reduced the performance. The reason behind underbalacing yielding better results may be that the observations for both class labels were similar. Thus a lower number of observations in total made it easier for the model to classify. Furthermore, in line with our expectations, the removal of unimportant features also improved the algorithm, because KNN gives the same weight to each of the attributes.

5.3 Neural Networks

Generally the Neural Network did not perform too bad, but still had some disadvantages, comparing it to XGBoost. For hyperparameter optimization, we considered a evolutionary approach using the highly customizable kears implementation to check the learning rate, the learning rate algorithm, and also the number of epochs.

Additionally we considered all three sampling methods, as well as simply weighting the different classes. One main problem was that the Neural Network seemed to overfit in most of the cases and predicted outcomes which were biased towards the majority class, even if sampling was applied. For that reason, we also played around with differently sized Neural Networks and different numbers of epochs as well as different dropouts. Finally, best results where achieved by using Adam optimizer with a learning rate of 0.0001 and with a weighting of the different classes 1:25. Additionally, training was stopped after 50 epochs in order to prevent the bias towards the majority class.

5.4 Random Forest

Random Forest achieved comparably good results on the dataset. With a systematic grid search we tried to discover the best hyperparameters, which are gini as splitting criterion and 70 as number of estimators. Surprisingly random forest works best with the removal of attributes highly correlated to the 15 most important features considering the decision tree importances. Additionally smote balancing improved the quality of the algorithm quite a bit as it prevents the random forest from biasing towards the majority class.

5.5 XGBoost

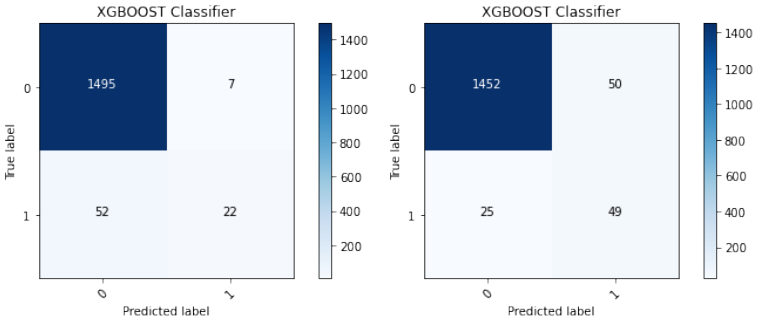

Figure 5: XGBoost without (left) and with (right) optimization.

XGBoost was the best classifier in the given selection of ML algorithms, as shown in Figure. This finding is in line with our expectations. The model performed best with oversampling and feature selection removing features which are highly correlated with top 15 features according to random forest. Since our data sets had a lot more instances in the non-bankrupt class, we expected this improvement. The

reason behind feature selection resulting in better test scores might be associated with a prevention of overfitting. As our dataset has 64 features, it might be causing a boosted model such as XGBoost to overfit. Therefore, limiting it to the important features (in removing correlated ones) helps. This results in higher f2-scores and a better confusion matrix in general. Binning did not improve the performance, as expected. Hyperparameter tuning is also sought, in our business case XGBoost worked best with a learning rate of 0.4.

5.6 Model Comparison

Predicting bankruptcies correctly is the most important aim of our project. Table 1 visualises each of our tested models' confusion matrices in a compact form. We can see that many models were good at predicting most of the actual 74 instances of the bankruptcy class correctly. However, even though recall is the main aim, it is not the only aim, hence we should also pay attention to precision. Tree based models, namely Random Forest and XGBoost, were the ones with the least false positives. They achieved to predict most of the bankrupt class labels, without having the tendency to predict bankrupt whatever the real value.
XGBoost outperforms Random Forest with higher true positives and lower false positives. If we compare XGBoost with a more advanced model, Neural Network, which had slightly more true positives (49 v 55); we see that false positive values increase almost 6 fold (50 v 295). Even though more true positives can be accepted in the expense of some more false positives; we believe 6 fold is a very high increase.

	Actual = 0		Actual = 1	
Model	Predicted = 0	Predicted = 1	Predicted = 0	Predicted = 1
Naive Bayes	120	1382	3	71
KNN	1232	270	28	46
Random Forest	1438	64	34	40
Neural Network	1207	295	19	55
XGBoost	1452	50	25	49

Table 1: Models' Confusion Matrices for Comparison

Figure 6 allows to compare the models from another perspective; when examining their precision, recall, f1 and f2 scores, XGBoost has by far the highest f2, f1 and precision score. XGBoost achieves these scores without a large drop in recall. It has a 8% lower score than neural network, the model with highest recall statistic. We should exclude Naive Bayes from the comparison since it predicts the majority

of all instances as bankrupt. Therefore, both looking at the confusion matrices and performance scores allow the same conclusion; our model for bankruptcy prediction is XGBoost.

Algorithm	Precision	Recall	F1-score	F2-score	% Delta F2
Baseline (NB)	0.05	0.95	0.09	0.193	-
KNN	0.15	0.62	0.24	0.376	94,8%
Neural Net	0.16	0.74	0.26	0.427	121,2%
Random Forest	0.38	0.54	0.45	0.49	153,9%
XGBoost	0.49	0.66	0.57	0.62	220,1%

Figure 6: Model Comparison Scores

5.7 Application to Other Years

Since it was not only our goal to predict bankruptcy three years before its occurrence, but also for one to five year prior, we also assessed the generalizability of our model. We used the third year data set to train our data, used the optimized preprocessing and hyperparameters for XGBoost, our best performing model. Testing showed that the model performed well for all years. As shown in table 2, for the first and the fifth year, the F2 scores were higher than for the third year (~ 0.70). This finding allows the conclusion, that the most important features were already indicating a bankruptcy five years before its occurrence. In the fourth and third year before the possible bankruptcy, it appears to be harder for the model to assess the future financial status of a company. This might be due to the ambiguity of some attributes. It is no surprise that the performance increases, the closer the observation are to the actual event (bankrupt/not bankrupt) as the attributes will show strong evidence of the looming bankruptcy.

Year	Precision	Recall	F1	**F2**
First	0.71	0.71	0.71	**0.71**
Second	0.48	0.58	0.53	**0.55**
Fourth	0.52	0.62	0.57	**0.6**
Fifth	0.71	0.74	0.72	**0.73**

Table 2: Model Trained on 3rd Year Data, Predicts Other Years

References

[1] L. O.Hall W. P. Kegelmeyer N. V. Chawla, K. W. Bowyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 2002.

[2] UC Irvine. Machine Learning Repository. Polish companies bankruptcy data.

[3] Andre Ye. When and why tree-based models (often) outperform neural networks.