Bankruptcy Predictor

Sebastian Morel (sebmor-8)

July 15, 2024

Introduction

Bankruptcy can be synonymous with the end of any company and have a negative impact for the company's investors, and their employees. Getting to know the symptoms and indicators of a company being close to bankruptcy would not only benefit the company itself, but also investors to know if they should invest in that company.
The aim of this project is to build an accurate model that will predict a company's bankruptcy from the dataset. The report will also measure the capability of different prediction models, and use different methods such as oversampling and dimensionality reduction in order to improve the prediction model.

1 Method

In this section the report will cover the dataset used for the model, the method used to retrieve the data and an analysis of the dataset itself. The programming was performed with Spyder inside a virtual environment using the Anaconda navigator.

1.1 The dataset

The dataset used in this report was found on Kaggle.com[2], titled "Company Bankruptcy Prediction". The data was collected from the Taiwan Economic Journal for the years 1999 to 2009. The dataset contains 6819 rows which represents Taiwanese companies existing between 1999 to 2009, columns which represents the different financial key figures and ratios such as "Cost of Interest-bearing Debt", "Net Income Growth" and so on, in total there are 95 difference financial variables. In the first column we find if the companies have gone bankrupt during the period, the companies that have gone bankrupt are indicated with a "1" and the ones that survived are indicated with a "0". In order to retrieve the data from Kaggle.com, I downloaded a csv file to my local storage and used pandas to convert it to a dataframe, so it could be implemented into the model.

1.2 Exploring the dataset

After getting the data into Spyder, a check was performed to find any missing values or duplicate items in the dataset, that could cause the training set to be misled.

```
#Checking for missing values
if self._df.isnull().sum().sum() == 0:
    print('There are no missing values.')
else:
    print('There are missing values:')
    print(self._df.isnull().sum())

#Checking for duplicates
if self._df.duplicated().sum() == 0:
    print('There are no duplicated values.')
else:
    print('There are duplicated values:')
    print(self._df.duplicated())
```

Figure 1: Codes used to check for missing and duplicate values.

```
Analyzing the data...
There are no missing values.
There are no duplicated values.
```

Figure 2: Console output from the code.

From 2 we can see that neither duplicate nor missing values were found in the dataset.
Next I analyze what the model wants to predict which is if a company is bankrupt or not, by doing a bargraph showing the frequency of non-bankrupt and bankrupt companies in the dataset.

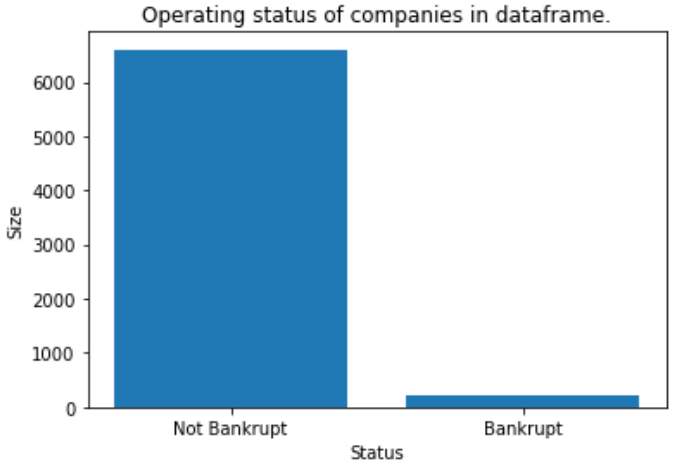

Figure 3: Barchart representing the amount of bankrupt vs non-bankrupt companies in the dataset.

Figure 3 shows an illustration of the difference in the amount of bankrupt companies compared to non-bankrupt ones. Numerically the amount of bankrupt companies amount to 220, while the amount of non-bankrupt amount to 6599, this is a very significant difference. This raises some concerns as the model will be very capable at recognizing a non-bankrupt company but will perform very poorly when it comes to recognizing a bankrupt one, to the point of just assuming most of the test data are non-bankrupt ones, in other words it will create a strong bias in our model. In order to prevent this two different methods will be used in order to remove the class imbalance. First we will use stratified sampling a method used in Liang et al.[7] where the amount of non-bankrupt companies is cut down through stratified sampling in order to match the amount of bankrupt companies, which will result in 220 bankrupt companies and 220 non-bankrupt companies. Another method is oversampling, this method duplicates the minority class which in this case is the bankrupt companies until it equates to the number of non-bankrupt companies (majority class). For this report we will use a technique for oversampling called Synthetic Minority Oversampling Technique (SMOTE), SMOTE utilizes k-nearest neighbors to create synthetic data, instead of just duplicating the data until both classes matches[1]. However unlike for undersampling we need to apply SMOTE onto our training set, instead of applying it on the whole dataset, to avoid misleading results [3].
Another issue with the dataset is the amount of variables the dataset has, in total there are 95 different variables. Too many dimensions in a dataset will lead to the so called "The Curse of Dimensionality", which means that too many features are added while the amount of training samples stays the same. This will lead to the model finding a "Perfect solution" which is likely due to overfitting [6]. For our model we will use principle component analysis (PCA) to reduce the dimensionality in our dataset. PCA compresses the dataset onto a lower-dimensional subspace with less feature, all while trying to keep the most relevant information from the original dataset [5].
Continuing the analysis of the dataset, a heatmap over the correlation between all the 95 features would be a good idea in theory but is really hard to read in practice and would be very clunky. Therefore a list was created which sorts out the financial variables with a correlation higher than 0.4 and lower than (-0.4).

```
#Check for correlation above 0.4 and below (-0.4).
corr = self._df.corr()
highcorr = ~(corr.mask(np.eye(len(corr), dtype=bool)).abs() > 0.4).any()
raw = corr.loc[highcorr, highcorr]
print(raw.columns.tolist())
```

Figure 4: List of variables with a correlation of more than 0.4 and less than (-0.4).

The code in figure4 first computes pairwise correlation between each variable. The "highcorr" variable is a pandas series which contains all the variables in the dataset where if the variable has a correlation of above 0.4 or below (-0.4) it sets is as True, otherwise False. We then sort out the ones that are set as true and print them out into a list.

```
['Bankrupt?', ' Operating Expense Rate', ' Research and development expense rate', ' Interest-
bearing debt interest rate', ' Tax rate (A)', ' Revenue Per Share (Yuan ¥)', ' Realized Sales
Gross Profit Growth Rate', ' Continuous Net Profit Growth Rate', ' Total Asset Growth Rate', '
Total Asset Return Growth Rate Ratio', ' Current Ratio', ' Quick Ratio', ' Interest Expense
Ratio', ' Total debt/Total net worth', ' Long-term fund suitability ratio (A)', ' Accounts
Receivable Turnover', ' Inventory Turnover Rate (times)', ' Fixed Assets Turnover Frequency', '
Revenue per person', ' Operating profit per person', ' Allocation rate per person', ' Quick
Assets/Current Liability', ' Cash/Current Liability', ' Inventory/Working Capital', '
Inventory/Current Liability', ' Long-term Liability to Current Assets', ' Cash Turnover Rate',
' Fixed Assets to Assets', ' Total assets to GNP price', ' No-credit Interval', ' Degree of
Financial Leverage (DFL)', ' Interest Coverage Ratio (Interest expense to EBIT)', ' Net Income
Flag']
```

Figure 5: List of variables with a correlation of more than 0.4 and less than (-0.4).

Figure 5 is the console output of the code shown above. We can see that out of 95 variables only 33 have a correlation of more than 0.4 and below (-0.4).

1.3 Models

In this report four different models will be compared against eachother, the models that will be compared are naive Bayes (NB), support vector machine (SVM), logistic regression (LR) and classification and regression tree (CART). NB, SVM and CART have been chosen due to their use in a related paper by Liang et al.[7]. LR has been chosen due to it's prominent use for binomial classification of response variables, which is very relevant to what we want to do. LR is based on the concept of probability, the classifier a probability, for example 0.2% chance a company is bankrupt, which will classify it as non-bankrupt[4].
Furthermore, a standard scaler will be used to avoid one feature dominating the rest of the features in the dataset. Lastly to ensure that the results are reproducible and to remove any random elements when comparing the different models a seed for randomness is used and set to 100.

2 Results

In this section, the results of the methods discussed in the previous section are applied to the dataset and presented.

2.1 Class imbalance

Type I Error refers to the company not being bankrupt but the model classified it as bankrupt. Type II Error means that the company is bankrupt but the model classified it as non-bankrupt.

Table 1: Overall performance of the prediction models.

	NB	SVM	LR	CART
Average Accuracy	42.44%	96.77%	96.33%	95.23%
Type I error	1562	1	27	67
Type II error	8	87	73	63

In Table1 we can see that SVM, LR and CART all have a pretty high accuracy, although these numbers are misleading due to the amount of bankrupt companies in the dataset. Out of 88 bankrupt companies in the test set SVM only managed to classify one as bankrupt. The type I error is out of 2640, and type II error is out of 88.

Table 2: Overall performance of the prediction models when the data is undersampled.

	NB	SVM	LR	CART
Average Accuracy	59.09%	81.06%	79.55%	83.33%
Type I error	7	16	15	12
Type II error	19	9	12	10

In Table 2 a method used by Liang et al.[7] is used, where stratified sampling is used to get the same amount of bankrupt as non-bankrupt companies in the original dataset. Although the accuracy is worse in this case we can see that the amount of Type II errors are considerably less than in Table 1, which is an improvement. Here the type I and II error is out of 66.
Lastly we will use SMOTE to oversample our training set set in order to balance out the classes.

```
#Oversample
self._X_train, self._y_train = SMOTE().fit_resample(self._X_train,self._y_train)
```

Figure 6: Code snippet to apply SMOTE on the training set.

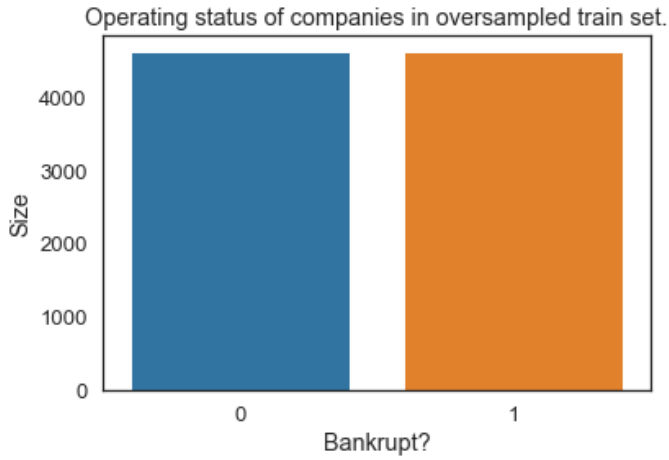

Figure 7: Barchart showing the amount of bankrupt vs non-bankrupt companies after oversampling the training set.

Table 3: Overall performance of the prediction models when the data is oversampled.

	NB	SVM	LR	CART
Average Accuracy	36.98%	90.10%	89.66%	92.85%
Type I error	1712	243	257	144
Type II error	7	27	25	51

Table 3 shows the performance of the prediction models when oversampling is applied to the training set. We can see a clear improvement in all the accuracies except for naive Bayes. Moving forward we will continue using oversampling, as it produces the best results.

2.2 Dimensionality Reduction

As previously mentioned a high dimensionality in the dataset can be an issue, PCA was therefore applied to the oversampled train set and also the test set, to reduce their dimensionality from 95 variables to 20.

```
#Dimensionality reduction
pca = PCA(n_components=20)
self._X_train = pca.fit_transform(self._X_train)
self._X_test = pca.transform(self._X_test)
self._X_train.shape, self._X_test.shape
```

Figure 8: Code snippet for applying PCA on our input variables for dimensionality reduction.

Table 4: Overall performance of the prediction models with dimensionality reduction.

	NB	SVM	LR	CART
Average Accuracy	93.29%	87.10%	87.83%	92.28%
Type I error	125	333	314	167
Type II error	58	19	18	38

If we compare table 4 with table 3 we can see a slight drop in the average accuracy in three out of four of our models. NB has surprisingly managed to improve to the point of being the model with the highest accuracy, compared to before where it only had an accuracy of 36.98%.

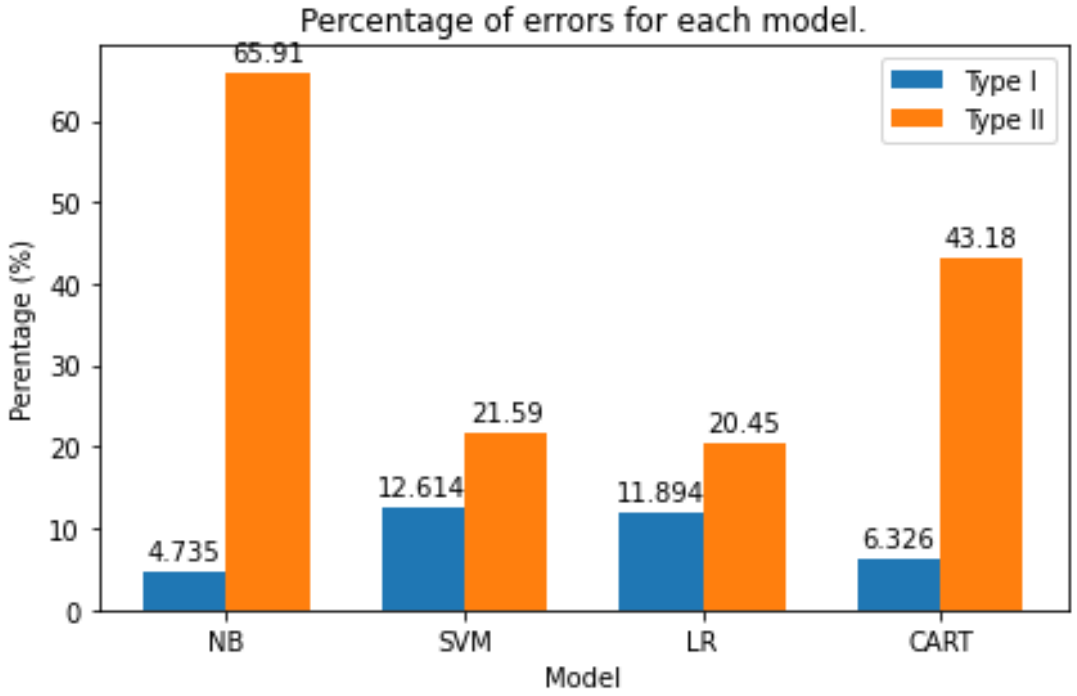

Figure 9: Bargraph representing the percentage of errors for each model.

In Figure 9 the percentage was calculating by taking the number of errors for each type divided by the total amount of bankrupt or non-bankrupt companies. From the graph we can get a clearer picture over the performance of each model. Although NB and CART had a comparatively high accuracy, they did not perform very good at classifying non-bankrupt companies. NB misclassified 65.91% and CART misclassified 43.18%, which is a clear bias. The prefered model would therefore be LR who had considerably less bias and performed better than SVM.

3 Discussion

I would deem the final results of this project to be adequate enough to give companies and investors a broad idea if a companies performance is approximately like one that has gone bankrupt the year after. Although more research would be needed to further improve the model's performed and maybe divide up each individual model tested in this report to optimize it according to their needs, for example further adjust the dimensions to improve NB's accuracy, or by adjusting the test size.
The data used in this project has no identifiable information on any individual nor company, additionally the access of a companies financial figures is made public upon the company going public. Finally I have no personal gain in creating a bias in the results of this report and therefore I deem this work to have been ethically executed.

References

[1] "5 SMOTE Techniques for Oversampling your Imbalance Data.". https://towardsdatascience.com/5-smote-techniques-for-oversampling-your-imbalance-data-b8155bdbe2b5. Accessed: 2021-03-17.

[2] "Bankruptcy data from the Taiwan Economic Journal for the years 1999–2009.". https://www.kaggle.com/fedesoriano/company-bankruptcy-prediction. Accessed: 2021-03-17.

[3] "Handling Imbalanced Data: SMOTE vs. Random Undersampling.". https://www.irjet.net/archives/V4/i8/IRJET-V4I857.pdf. Accessed: 2021-03-17.

[4] "Introduction to Logistic Regression". https://towardsdatascience.com/introduction-to-logistic-regression-66248243c148. Accessed: 2021-03-17.

[5] "Principal Component Analysis for Dimensionality Reduction.". https://towardsdatascience.com/principal-component-analysis-for-dimensionality-reduction-115a3d157bad. Accessed: 2021-03-17.

[6] "Why is Dimensionality Reduction so Important?". https://medium.com/@cxu24/why-dimensionality-reduction-is-important-dd60b5611543. Accessed: 2021-03-17.

[7] C. Tsai G. Shih D. Liang, C. Lu. Financial ratios and corporate governance indicators in bankruptcy prediction: A comprehensive study, 2016.

Appendices

A UML diagram