EC4308: Machine Learning & Economic Forecasting

AY20/21 Semester 2

Group Project

BANKRUPTCY FORECASTING

National University of Singapore

Heng Jian Shun A0166871U

Hoon Yong Zhi, Joseph A0167097R

Leong Yu Gene A0167742X

Tay Wei Wen Jonathan Brendan A0168702A

1. Introduction

Company bankruptcy has been a popular forecasting object for researchers to predict on. The likelihood of a firm becoming insolvent would be important for retail, institutional investors and venture capitalists who are considering companies to inject capital and investment into. In the context of developing countries, forming accurate judgements about firm bankruptcy is especially crucial for the public sector, where government expenditure may be tight. As such, bankruptcy predictions are paramount in the pursuit of economic growth; an ability to accurately preemptively spot a company that is on the verge of bankruptcy would create lower investor risks and create growth opportunities.

2. Literature Review

Much of the current literature makes use of several types of predictors in order to make predictions about bankruptcies. Tinoco & Wilson (2013) combined accounting ratios, market-based data and macroeconomic data to explain and predict corporate credit risk using different specifications of logit regressions (LR), and found that constructing predictions using all three aspects of data yielded the best performance. In other works, predictors like company-specific financial ratios (FRs) and corporate governance indicators (CGIs) have been added to provide a more holistic picture of company health and solvency, which theoretically and intuitively could improve bankruptcy predictions.

Liang, et al. (2016), from which our dataset comes from, use a combination of seven different categories of FRs and five different categories of CGIs to form predictive classification models. The methods considered in this paper were popular supervised machine learning techniques such as support vector machines (SVM), *k*-nearest neighbour (KNN), naive Bayes (NB) classifier, classification and regression tree (CART), and multilayer perceptron (MLP). One of the main findings was that SVM performed the best, and that the most important FR features for prediction were categories of solvency and profitability.

Fedorova, et al. (2013) applied combinations of machine learning methods, using multivariate discriminant analysis (MDA), LR, CART and artificial neural networks on Russian manufacturing company data. Here, combinations of financial indicators were chosen by the different learning algorithms to be used as features in the final neural network model.

3. Data

Our dataset was sourced from the Taiwan Economic Journal from 1999-2009 and collected by Liang et al. (2016). The label is a binary variable, which equals 1 if the company became bankrupt, and 0 otherwise. The definition of bankruptcy is defined by the business regulations under the Taiwanese Stock Exchange. Given the limitations of the dataset, our analysis will be constrained to making use of FRs and other accounting measures as our features of interest.

The key strength of this dataset is that it is very comprehensive, as it contains 6819 unique observations with 95 features of various accounting ratios and financial indicators. A full list of these features can be found in the Appendix. However, one drawback of the data is that it is heavily skewed towards companies that do not go bankrupt. 220 companies go bankrupt while 6599 companies remain solvent. As such, we will consider resolving this through performing some weighted, balanced and upsampling methods on top of the traditional machine learning models to account for the imbalance and improve our predictions for this scenario.

In terms of data cleaning, the dataset provides very clean observations with no missing values. However, upon further investigation, we removed the binary variable *Net Income Flag,* since all observations equaled 1, therefore providing no value in our predictive models.

4. Methodology

For all our models, we used a 1:1 split for our training and test sets. Seeing as the data is heavily imbalanced in favour of the majority class (non-bankrupt), we run a simple loop in order to determine a suitable seed to ensure that our test set has at least 100 of the minority class (bankrupt) observations. The range of models that we used are: Logit Regression, LASSO Logit Regression, Boosted Trees, Random Forests and Artificial Neural Network. Finally, we produced forecast combinations based on the results of the individual models using majority vote and average probability.

To compare between the models as well as different specifications between the models, we use several methods, namely AUC, confusion matrix output, as well as precision-recall.
We follow the suggestion of Branco et. al. (2015) in utilizing the F-Measure (F_β) score. The F_β scores would be more suitable in our context as compared to accuracy since we have an uneven class distribution, notably a larger number of actual negatives.

$$F_\beta = (1+\beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{(\beta^2 \cdot \text{precision}) + \text{recall}}.$$

First, we consider the F_1 ($\beta = 1$) score (the harmonic mean of precision and recall), which is proportional to the product of precision and recall and inversely proportional to the sum of precision and recall:

$$F_1 = \frac{2}{\mathrm{recall}^{-1} + \mathrm{precision}^{-1}} = 2 \cdot \frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}} = \frac{\mathrm{tp}}{\mathrm{tp} + \frac{1}{2}(\mathrm{fp} + \mathrm{fn})}.$$

Next, we choose $\beta = 2$, which considers recall twice as important as precision: we aim to utilize this measure towards our goal of minimizing false negatives.

Given the context of our problem, the cost of predicting a false positive and false negative are asymmetric. We believe that false negatives would be costlier as large financial repercussions would be incurred for all stakeholders involved if a firm that actually goes bankrupt is misclassified under a non-bankruptcy prediction, as opposed to vice versa. Thus, recall would be a more important measure of effectiveness and we would primarily be using F_2 to compare models.

5. Results

5.1. Logit Regression

For our benchmark model, we used a logit regression to form our predictions. This base model is used as a benchmark in many binary predictive models. We report the confusion matrix as well as AUC, precision, recall, F1 and F2 values as shown below:

		Actual	
		0	**1**
Predicted	**0**	3261	74
	1	40	34

AUC	0.8627
Precision	0.4595
Recall	0.3148
F1 Score	0.3736
F2 Score	0.3360

5.2. LASSO Logit Regression

Next, we perform a LASSO logit model to examine the effects of shrinkage on the predictions. We used the *rlasso* package which provides the theoretical plug-in of the shrinkage parameter, leaving it at default settings. This model is also useful since it provides feature selection, since coefficients of variables that are not that useful for prediction would be reduced to zero. Our results are shown below:

		Actual	
		0	**1**
Predicted	**0**	3298	103
	1	3	5

AUC	0.9078
Precision	0.6250
Recall	0.04630
F1 Score	0.08621
F2 Score	0.05681

5.3. Boosted Trees

We performed 3 sets of different boosted trees. We first use the *gbm* package to develop a baseline boosted tree model. Next we implement the *xgb* package which performs the XGBoost algorithm, utilising 10-fold cross validation to select the optimal depth of the trees and applying stratified sampling. Finally, we also try a class-weighted XGBoost model with the same specifications as before to deal with our imbalanced dataset. This scales the errors produced by the model during training on our positive class (Bankrupt). By making the algorithm over-correct them, this would theoretically result in better performance when predicting. To begin, we ran the baseline boosted tree model and obtained these results:

		Actual	
		0	**1**
	0	3288	83

Predicted	**1**	13	25

AUC	0.9158
Precision	0.6579
Recall	0.2315
F1 Score	0.3425
F2 Score	0.2660

Next, we applied stratified sampling and 10-fold cross validation to select the best tree depth. We tried different values of maximum tree depth, and cross validation revealed the best iteration for each choice.

Max Depth	5	6	7	8	9
Best Iteration	21	43	46	61	43

Then, we ran the model for all 5 maximum depths with their respective best number of iterations and computed predictions. We then extracted the AUC values for each.

Max Depth	5	6	7	**8**	9
AUC Values	0.9129893	0.9148294	0.9124928	**0.9244253**	0.9135924

We found that the highest AUC value was 0.9244. Hence, the final choice of tuned parameters for our *xgboost* stratified CV model uses max depth of 8 and 61 iterations. This model gave us the following score metrics:

		Actual	
		0	**1**
Predicted	**0**	3282	80
	1	19	28

AUC	0.9244

Precision	0.5957
Recall	0.2593
F1 Score	0.3613
F2 Score	0.2923

Finally, we tried out a weighted stratified CV model using the exact same method as the previous model, using 5 specifications of max depth.

Max Depth	5	6	7	8	9
Best Iteration	37	38	92	41	41

Again, we ran the model for all 5 max depths with their respective best number of iterations and computed predictions. Extracting the AUC for each specification yielded the following results:

Max Depth	5	6	**7**	8	9
AUC Values	0.8942885	0.9067987	**0.9133231**	0.9029447	0.9084705

Here, the highest AUC value found was 0.9133. The F2 scores were also uniform throughout the models. Hence, the final choice of tuned parameters for our weighted stratified CV model uses a max depth of 7 and 92 iterations. The model gave us an F2 score of 0.4237288 and a confusion matrix as shown below.

		Actual	
		0	**1**
Predicted	**0**	3247	63
	1	54	45

AUC	0.9133
Precision	0.4546
Recall	0.4167
F1 Score	0.4348
F2 Score	0.4237

5.4. Random Forest

We ran several models using the Random Forest method. We first used *rftune* to search for the optimal value of mtry - it returns mtry = 9, which is the same as the default rule. We therefore use mtry = 9 to fit a benchmark random forest model as well as the other random forests models.

5.4.1. Benchmark Random Forest

We then ran a benchmark model. The baseline random forest model resulted in the confusion matrix and other metrics as shown:

		Actual	
		0	**1**
Predicted	**0**	3292	87
	1	9	21

AUC	0.9075
Precision	0.7000
Recall	0.1944
F1 Score	0.3043
F2 Score	0.2273

5.4.2. Balanced Random Forest

Next, we fitted balanced random forest models using three different class weights, notably 1:1, 4:1 and weights based upon inverse probability. They key behind the balanced random forest idea lies in downsampling the majority class (with replacement) as a means of dealing with our imbalanced data. If we use this naively, by only downsampling once at the start and conducting the standard random forest approach, our results may be negatively influenced by the loss of information contained in the observations excluded via downsampling. By way of contrast, we report the inferior results obtained via naive downsampling.

These are the results obtained for the naive downsampling case.

		Actual	
		0	**1**
Predicted	**0**	2768	25
	1	533	83

AUC	0.898
Precision	0.1347403
Recall	0.7685185
F1 Score	0.2292818
F2 Score	0.3959924

Following the work of Chen et al. (2004), we therefore downsample at each stage of the random forest as follows: we randomly select a bootstrap sample (by definition, with replacement) of the minority class observations and then randomly select an equal number of majority class observations with replacement from our data. Following this, we create a tree at each stage contingent on the value of *mtry* as in our base random forest. After doing this iteratively, we aggregate over the trees generated with each bootstrap replication in order to obtain our final prediction.

We implement balanced random forest primarily through the base *randomForest* package utilizing the *sampsize* and *strata* flags to control the number of observations sampled from the minority and majority classes at each stage of the random forest. First, we run a balanced forest with 1:1 weight.

		Actual	
		0	**1**
Predicted	**0**	2952	27
	1	349	81

AUC	0.9067
Precision	0.1884

Recall	0.75
F1 Score	0.3011
F2 Score	0.4698

Second, we run on a 4:1 weight.

		Actual	
		0	**1**
Predicted	**0**	3154	45
	1	147	63

AUC	0.9091
Precision	0.3000
Recall	0.5833
F1 Score	0.3962
F2 Score	0.4907

Secondarily, we utilize the *s.RANGER* package, which is built upon the *ranger* package - here, we use the *sample.fraction* flag which, in a similar vein, determines the fraction of observations of each class to be used at each stage. In addition to the 1:1 ratio, we also utilize a ratio based upon the inverse probability of each class (we define this as *classweights)*. In essence, the majority class weight corresponds to the percentage of minority observations we have, and vice versa. This is intuitively consistent with our goal of representing both classes somewhat equally in our predictive process. Firstly, we run random forests using the s..RANGER function using a sample ratio of 1:1.

		Actual	
		0	**1**
Predicted	**0**	3269	76
	1	32	32

AUC	0.9109
Precision	0.5
Recall	0.2962963
F1 Score	0.372093
F2 Score	0.3225806

Next, we do the same based on the inverse probability of each class.

		Actual	
		0	**1**
Predicted	**0**	2787	24
	1	514	84

AUC	0.8962
Precision	0.1404
Recall	0.7778
F1 Score	0.2379603
F2 Score	0.407767

Lastly, we try the inverse probability weighting built into the *s.RANGER* package to assign a probability to each observation, using the flag *ipw.case.weights*. This probability will determine how likely the particular observation is drawn within each bootstrap sample. Again, this probability is based upon whether the observation is in the minority or majority class, where observations which belong to the minority class are weighted heavily and vice versa, inversely proportional to how often the class appears in our dataset. The results are shown below.

s.RANGER with IPW caseweights

		Actual	
		0	**1**
Predicted	**0**	3230	63
	1	71	45

AUC	0.9114
Precision	0.387931
Recall	0.4166667
F1 Score	0.4017857
F2 Score	0.4105839

5.4.3. Weighted Random Forests

Again following in the steps of Chen et al. (2004), we implement a weighted random forest. The key behind this idea is assigning a larger weight, or equivalently, a higher misclassification cost, to the minority class. We follow the standard random forest algorithm with two changes: First, when building a tree at each stage, our class weights are used in weighting the Gini criterion in order to determine splits (and therefore the resulting tree). Second, when we reach the terminal nodes of each tree, the class weights are used to modify the 'majority vote' rule. The new 'majority vote' rule takes the weighted vote for each class as the weight of the class multiplied by the number of cases for the class at the terminal node. Finally, we aggregate over the (weighted vote) of each tree in order to obtain our final prediction.

We implement this again by utilizing the *s.RANGER* package, which is built upon *ranger*. We use inverse probability weighting of each class in order to determine our class weights, using the flag *ipw.class.weights*.

s.RANGER with IPW classweights

		Actual	
		0	**1**
Predicted	**0**	3292	87
	1	9	21

AUC	0.9073
Precision	0.7
Recall	0.1944444
F1 Score	0.3043478
F2 Score	0.2272727

5.4.4. Upsampling

We attempt naive upsampling for the sake of comparison - in contrast to naive downsampling, this is in essence randomly sampling the minority class with replacement until we obtain the same number of observations we have in our majority class. Following this, the standard random forest algorithm is performed. These are the results obtained:

		Actual	
		0	**1**
Predicted	**0**	3268	75
	1	33	33

AUC	0.9111
Precision	0.5
Recall	0.3055556
F1 Score	0.3793103
F2 Score	0.3313253

5.5. Artificial Neural Network (ANN)

We try out several iterations of ANN models on our dataset. Here, we use the *nnet* package, which restricts our model to just 1 hidden layer. We test out a set of 10 different weight decay parameters on 1, 5, 10, 15 and 20 derived features (neurons) to find the optimal combinations for the best predictions. Using these 50 combinations, we evaluated using AUC and F2 score to find the combination of decay factor and number of neurons that produces the highest metric accordingly. We report the results in the combination table below:

Metric: AUC

Decay Factor

	0	**0.001**	**0.003**	**0.01**	**0.03**	**0.1**	**0.3**	**1**	**3**	**10**
1 Neuron	0.5	0.83387	0.86094	0.87028	0.89380	0.89810	0.89601	0.83675	0.81179	0.79975
5 Neurons	0.5	0.82258	0.84569	0.82421	0.87435	0.89752	0.89570	0.84866	0.82112	0.80423
10 Neurons	0.5	0.82280	0.82746	0.82615	0.87237	0.89762	0.89563	0.85089	0.82513	0.80665
15 Neurons	0.5	0.82672	0.82364	0.84234	0.86657	0.89765	0.89569	0.85158	0.82766	0.80843
20 Neurons	0.5	0.81652	0.82164	0.85029	0.86483	0.89766	0.89569	0.85188	0.82937	0.80971

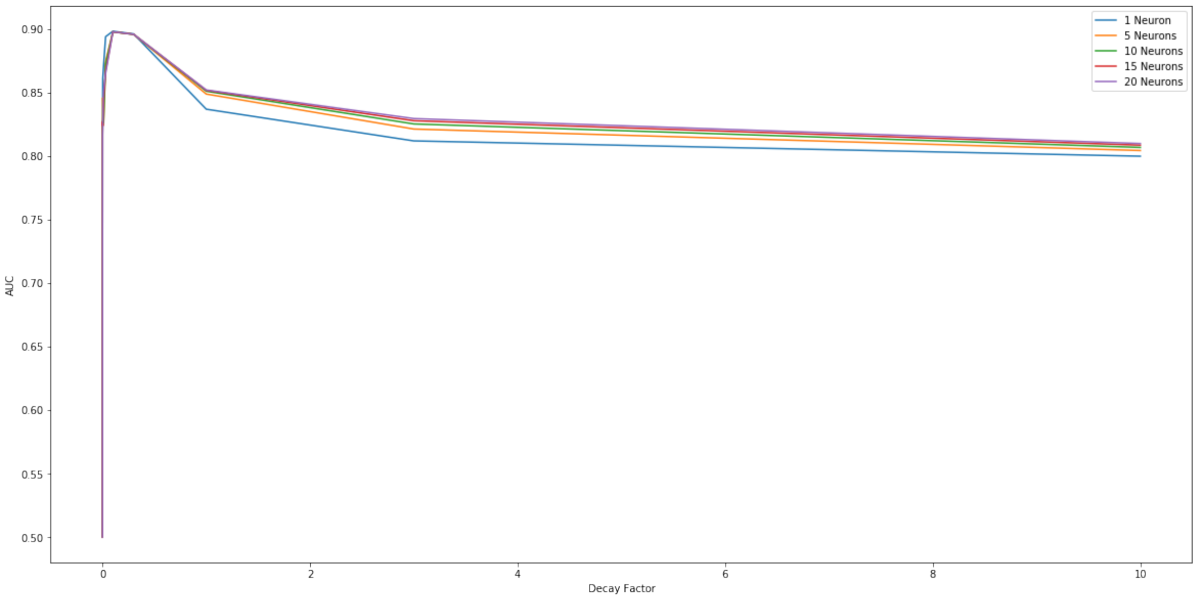

Figure 1: Decay Factor vs AUC (Decay factors ranging from 0 to 10)

From figure 1 above, we notice that the highest AUC values are clustered around decay factor values ranging from 0.03 to 0.3. As such, we decided to run another combination of the same set of number of neurons with 8 new decay factor values spanning within this range. The results with the new choices of decay factors are found below in figure 2.

Metric: AUC

Decay Factor

	0.03	**0.06**	**0.1**	**0.13**	**0.16**	**0.2**	**0.26**	**0.3**
1 Neuron	0.893806	0.895242	0.898106	0.898196	**0.898305**	0.897859	0.896754	0.896016
5 Neurons	0.874353	0.893764	0.89752	0.897587	0.897618	0.897122	0.896431	0.895702
10 Neurons	0.872376	0.893225	0.897624	0.897629	0.897537	0.897107	0.896356	0.895635
15 Neurons	0.866572	0.893251	0.897657	0.897632	0.897542	0.897136	0.896359	0.895694
20 Neurons	0.86483	0.893122	0.89766	0.897657	0.897542	0.897133	0.896364	0.895699

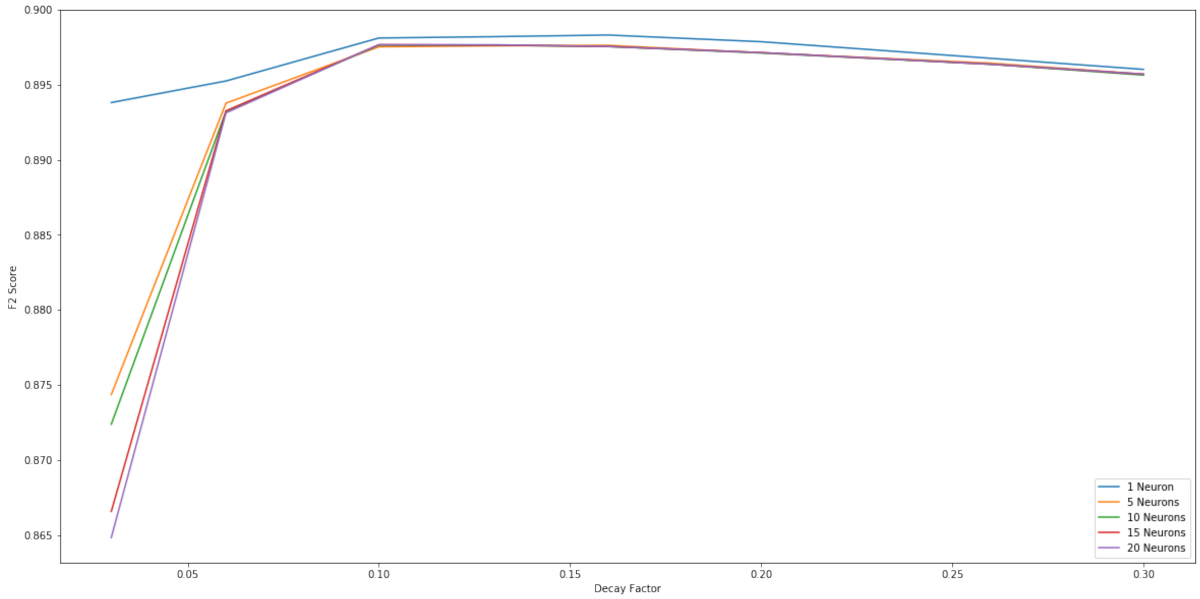

Figure 2: Decay Factor vs AUC (Decay factors ranging from 0.03 to 0.3)

The best combination of parameters selected using AUC as a metric would be 1 neuron with a decay factor of 0.16.

Next, we used F2 score values to test different combinations of parameters - we are presented with a clear highest F2 value at 10 neurons and a decay factor of 0.001.

Metric: f2

Decay Factor

	0.001	**0.003**	**0.01**	**0.03**	**0.1**	**0.3**
1 Neuron	0.2008032	0.2263374	0.2178423	0.2205882	0.130719	0.03424658
5 Neurons	0.2462121	0.2480159	0.2490040	0.2277433	0.130719	0.07918552
10 Neurons	**0.3321033**	0.2394636	0.2366864	0.2965235	0.130719	0.09029345
15 Neurons	0.2156863	0.3055556	0.2606178	0.2371134	0.130719	0.09029345
20 Neurons	0.2895753	0.3177570	0.3053435	0.2286902	0.130719	0.0902934

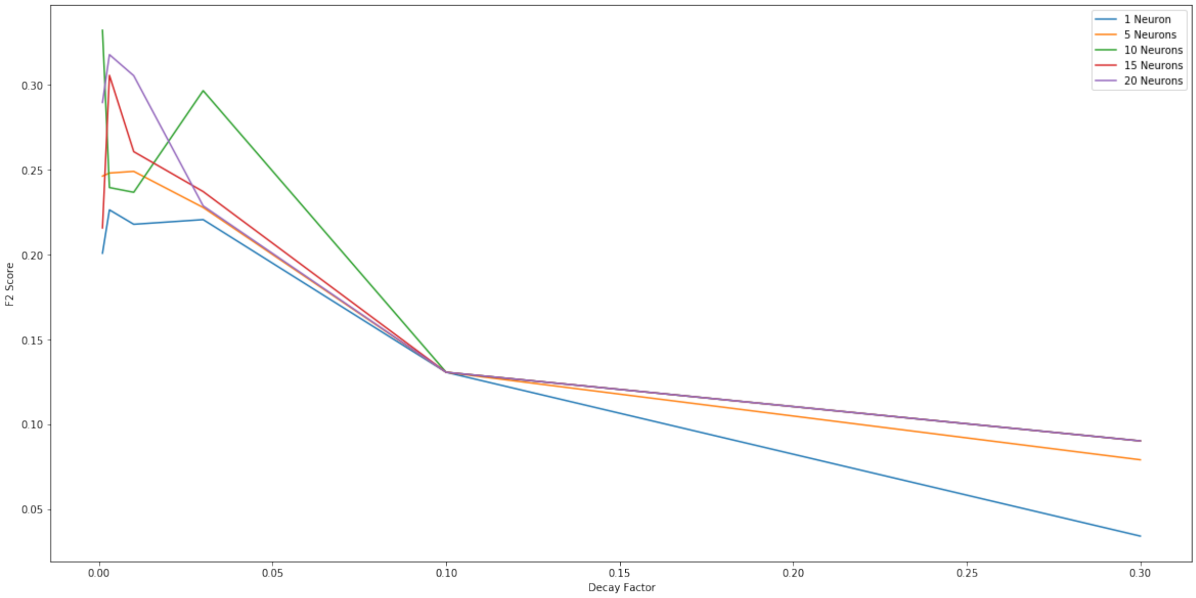

Figure 3: Decay Factor vs F2 Score (Decay factors ranging from 0.001 to 0.3)

We then constructed our predictions using the best combination of parameters that we had observed earlier for both the AUC and F2 score metrics. These are the results obtained:

<u>Using AUC metric</u>

Highest AUC: 1 neuron, 0.16 decay factor

		Actual	
		0	**1**
Predicted	**0**	3293	98
	1	8	10

AUC	0.8983
Precision	0.5555
Recall	0.0926
F1 Score	0.1587
F2 Score	0.1111

<u>Using F2-score metric</u>

Highest F2: 10 neurons, 0.001 decay factor

		Actual	
		0	**1**
Predicted	**0**	3227	72
	1	74	36

AUC	0.8228
Precision	0.3272
Recall	0.3333
F1 Score	0.3303
F2 Score	0.3321

6. Combined Forecast

We perform 2 combined forecasts, using majority vote and average probability over all our previously specified models. We first consider the majority vote combination, where we look at all observations across all models. For example, if most of the models predict 1 (bankrupt), then the combination model will follow the majority count.

		Actual	
		0	**1**
Predicted	**0**	3285	78
	1	16	30

Precision	0.6522
Recall	0.2778
F1 Score	0.3896
F2 Score	0.3138

Next, we form predictions using a simple average of all forecasted probabilities obtained from all of our predicted models.

		Actual	
		0	**1**
Predicted	**0**	3282	73
	1	19	35

AUC	0.9113
Precision	0.6481
Recall	0.3241
F1 Score	0.4321
F2 Score	0.3601

7. Discussion of Results

We compile the results of the different models and the evaluation metrics into the tables below.

Method	AUC	F1	F2
Logit Regression	0.8627	0.3736	0.336
LASSO Logit Regression	0.9078	0.08621	0.05681
Baseline Boosted Tree	0.9158	0.3425	0.266
Boosted Tree (10-Fold CV, Stratified)	**0.9244**	0.3613	0.2923
Weighted Boosted Tree (10-Fold CV, Stratified)	0.9133	**0.4348**	0.4237
Random Forest	0.9075	0.3043	0.2273
Balanced Random Forest (at 1:1 sampsize)	0.9067	0.3011	0.4698
Balanced Random Forest (at 4:1 sampsize)	0.9091	0.3962	**0.4907**
Balanced Random Forest (sample fraction 1:1)	0.9109	0.372093	0.3225806
Balanced Random Forest (sample fraction = classweights)	0.8962	0.2379603	0.407767
Balanced Random Forest (IPW caseweights)	0.9114	0.4017857	0.4105839
Random Forest with Naive Downsampling	0.898	0.2292818	0.3959924
Weighted Random Forest (IPW classweights)	0.9073	0.3043478	0.2272727
Random Forest with Naive Upsampling	0.9111	0.3793103	0.3313253
ANN, 1 neuron, $\lambda = 0.16$ (Using AUC)	0.8983	0.1587	0.1111
ANN, 10 neurons, $\lambda = 0.001$ (Using F2)	0.8228	0.3303	0.3303
Forecast combination (Majority Vote Rule)	-	0.3896	0.3138
Forecast combination (Simple Average Probability)	0.9113	0.4321	0.3601

Firstly, if we use AUC as a comparison metric, we see that all other methods beat our benchmark logit regression model, which is encouraging. However, the same is not true when looking at F1 and F2 values, with some of our models underperforming the benchmark logit model.

We find that the boosted tree with iterations selected by 10-fold CV and stratified sampling perform the best under AUC. However, if we look at F1 and F2, the best models chosen are the weighted boosted tree and the balanced random forest (4:1) respectively. This shows that our efforts at attempting to tackle the class imbalances has yielded some merits at least since all these methods were crafted specifically to address the issue through the assigning of weights to the binary values of our dependent variable.

While balanced random forest at a 1:1 sample size and 4:1 sample size perform similarly with respect to AUC, it is interesting to note that the 4:1 sample size outperforms 1:1 in terms of F1 and F2. Considering how imbalanced our original dataset is, this may mean that we are overcorrecting in the 1:1 case.

As expected, balanced random forest largely outperforms random forest with naive downsampling. Intuitively, we can understand naive upsampling outperforming naive downsampling by a significant margin due to the loss of information in the excluded observations when we do naive downsampling.

The relatively poor performance, in particular with respect to the F2 measure, of weighted random forest is somewhat surprising. It is possible that, though they are intuitive, using inverse probability weights to weight the classes does not assign enough weight to the minority class. In other words, we are not penalizing the misclassification into false negatives heavily enough. One possible extension here would be to use the out-of-bag estimate from random forest to select weights.

Surprisingly, both ANN models perform close to the worst among all the models for all 3 metrics, despite testing several combinations of neurons and decay factors to obtain the best AUC and F2 values. We suspect that this is because our dataset is overly complex with many parameters, which could have led to overfitting; weighted neural networks with more than one hidden layer may lead to better performances.

An interesting observation to note here is that the eventual forecast combinations did not produce improved results, in terms of either the AUC, F1 or F2 scores, over some of the other models that were specifically designed to tackle the class imbalance problem. This is probably because a few of the models included in the construction of our combined forecasts were basic ones that were only used as benchmarks to compare against our forecasts from the more sophisticated models used. We note that the simple average probability forecast combination method does significantly better than the forecast combination conducted on the majority vote rule; this is logical since averaging over all forecasts means that we retain a linear combination of both good and poor forecasts rather than simply evaluating binary 0 and 1 values through the majority vote; we lose information in the latter.

8. Limitations

One limitation of our project was the fact that our dataset was extremely unbalanced. The ratio of negative to positive observations of our prediction variable was around 30:1, and this made predicting very challenging. In our project, we tried out a few methods to account for this. An extension of our methods could be to use other sophisticated techniques such as Synthetic Minority Oversampling Technique (SMOTE), which uses a nearest-neighbour algorithm in order to generate new, synthetic data points to be used for training the model.

Another limitation is that we did not try out hybrid learning techniques to achieve better predictive performance. For example, one way we could have done this was to first fit a LASSO model to generate our predictions, calculate the residuals of the model and then train a random forest on the residuals using the same predictors. The hybrid prediction would be the summation of the predictions formed by LASSO and random forest respectively. Additionally, as mentioned above, we only performed a naive combination forecast using simple average. One way to improve on this would be to perform a weighted average when combining the predictions of the different forecast models; this essentially assigns a larger weight to the forecasts generated by models that considered the class imbalance problem and lower or even zero weights to the weaker performing forecasts.

Lastly, our research is only constrained to using the dataset collected on Taiwanese companies. Taiwan is a relatively developed country with institutions that support the role of small & medium enterprises (SMEs), especially in the electronics sector, within their economy (Matthews, 1997). In this aspect, it is difficult to generalise the results to other settings simply because the business environment and regulatory rules vary from country to country. Given that the idea of predicting bankruptcies may be more important for developing countries in their pursuit of economic growth, any insights garnered should be extrapolated with caution. Additionally, our dataset relies only on financial metrics and ratios to develop our predictive models. However, there certainly are political and geographic factors that may be affecting bankruptcy probabilities that are not being modeled in our report.

APPENDIX

List of Features

X1 - ROA(C) before interest and depreciation before interest: Return On Total Assets(C)

X2 - ROA(A) before interest and % after tax: Return On Total Assets(A)

X3 - ROA(B) before interest and depreciation after tax: Return On Total Assets(B)

X4 - Operating Gross Margin: Gross Profit/Net Sales

X5 - Realized Sales Gross Margin: Realized Gross Profit/Net Sales

X6 - Operating Profit Rate: Operating Income/Net Sales

X7 - Pre-tax net Interest Rate: Pre-Tax Income/Net Sales

X8 - After-tax net Interest Rate: Net Income/Net Sales

X9 - Non-industry income and expenditure/revenue: Net Non-operating Income Ratio

X10 - Continuous interest rate (after tax): Net Income-Exclude Disposal Gain or Loss/Net Sales

X11 - Operating Expense Rate: Operating Expenses/Net Sales

X12 - Research and development expense rate: (Research and Development Expenses)/Net Sales

X13 - Cash flow rate: Cash Flow from Operating/Current Liabilities

X14 - Interest-bearing debt interest rate: Interest-bearing Debt/Equity

X15 - Tax rate (A): Effective Tax Rate

X16 - Net Value Per Share (B): Book Value Per Share(B)

X17 - Net Value Per Share (A): Book Value Per Share(A)

X18 - Net Value Per Share (C): Book Value Per Share(C)

X19 - Persistent EPS in the Last Four Seasons: EPS-Net Income

X20 - Cash Flow Per Share

X21 - Revenue Per Share (Yuan ¥): Sales Per Share

X22 - Operating Profit Per Share (Yuan ¥): Operating Income Per Share

X23 - Per Share Net profit before tax (Yuan ¥): Pretax Income Per Share

X24 - Realized Sales Gross Profit Growth Rate

X25 - Operating Profit Growth Rate: Operating Income Growth

X26 - After-tax Net Profit Growth Rate: Net Income Growth

X27 - Regular Net Profit Growth Rate: Continuing Operating Income after Tax Growth

X28 - Continuous Net Profit Growth Rate: Net Income-Excluding Disposal Gain or Loss Growth

X29 - Total Asset Growth Rate: Total Asset Growth

X30 - Net Value Growth Rate: Total Equity Growth

X31 - Total Asset Return Growth Rate Ratio: Return on Total Asset Growth

X32 - Cash Reinvestment %: Cash Reinvestment Ratio

X33 - Current Ratio

X34 - Quick Ratio: Acid Test

X35 - Interest Expense Ratio: Interest Expenses/Total Revenue

X36 - Total debt/Total net worth: Total Liability/Equity Ratio

X37 - Debt ratio %: Liability/Total Assets

X38 - Net worth/Assets: Equity/Total Assets

X39 - Long-term fund suitability ratio (A): (Long-term Liability+Equity)/Fixed Assets

X40 - Borrowing dependency: Cost of Interest-bearing Debt

X41 - Contingent liabilities/Net worth: Contingent Liability/Equity

X42 - Operating profit/Paid-in capital: Operating Income/Capital

X43 - Net profit before tax/Paid-in capital: Pretax Income/Capital

X44 - Inventory and accounts receivable/Net value: (Inventory+Accounts Receivables)/Equity

X45 - Total Asset Turnover

X46 - Accounts Receivable Turnover

X47 - Average Collection Days: Days Receivable Outstanding

X48 - Inventory Turnover Rate (times)

X49 - Fixed Assets Turnover Frequency

X50 - Net Worth Turnover Rate (times): Equity Turnover

X51 - Revenue per person: Sales Per Employee

X52 - Operating profit per person: Operation Income Per Employee

X53 - Allocation rate per person: Fixed Assets Per Employee

X54 - Working Capital to Total Assets

X55 - Quick Assets/Total Assets

X56 - Current Assets/Total Assets

X57 - Cash/Total Assets

X58 - Quick Assets/Current Liability

X59 - Cash/Current Liability

X60 - Current Liability to Assets

X61 - Operating Funds to Liability

X62 - Inventory/Working Capital

X63 - Inventory/Current Liability

X64 - Current Liabilities/Liability

X65 - Working Capital/Equity

X66 - Current Liabilities/Equity

X67 - Long-term Liability to Current Assets

X68 - Retained Earnings to Total Assets

X69 - Total income/Total expense

X70 - Total expense/Assets

X71 - Current Asset Turnover Rate: Current Assets to Sales

X72 - Quick Asset Turnover Rate: Quick Assets to Sales

X73 - Working Capital Turnover Rate: Working Capital to Sales

X74 - Cash Turnover Rate: Cash to Sales

X75 - Cash Flow to Sales

X76 - Fixed Assets to Assets

X77 - Current Liability to Liability

X78 - Current Liability to Equity

X79 - Equity to Long-term Liability

X80 - Cash Flow to Total Assets

X81 - Cash Flow to Liability

X82 - CFO to Assets

X83 - Cash Flow to Equity

X84 - Current Liability to Current Assets

X85 - Liability-Assets Flag: 1 if Total Liability exceeds Total Assets, 0 otherwise

X86 - Net Income to Total Assets

X87 - Total assets to GNP price

X88 - No-credit Interval

X89 - Gross Profit to Sales

X90 - Net Income to Stockholder's Equity

X91 - Liability to Equity

X92 - Degree of Financial Leverage (DFL)

X93 - Interest Coverage Ratio (Interest expense to EBIT)

X94 - Net Income Flag: 1 if Net Income is Negative for the last two years, 0 otherwise

X95 - Equity to Liability

References:

Branco, P., Torgo, L. & Ribeiro, R. (2015). A Survey of Predictive Modelling under Imbalanced Distributions. Retrieved from: https://arxiv.org/pdf/1505.01658.pdf

Chen, C., Liaw, A., & Breiman, L. (2004). Using Random Forest to Learn Imbalanced Data. Retrieved from: https://statistics.berkeley.edu/tech-reports/666

Fedorova, E., Gilenko, E., & Dovzhenko, S. (2013). Bankruptcy prediction for Russian companies: Application of combined classifiers. *Expert Systems with Applications*, 40(18), 7285-7293. https://doi.org/10.1016/j.eswa.2013.07.032

Liang, D., Lu, C., Tsai, C., & Shih, G. (2016). Financial ratios and corporate governance indicators in bankruptcy prediction: A comprehensive study. *European Journal of Operational research*, 252(2), 561-572. https://doi.org/10.1016/j.ejor.2016.01.012

Matthews, A. (1997). A Silicon Valley of the East: Creating Taiwan's Semiconductor Industry. *California Management Review,* 39(4)

Tinoco, M. & Wilson, N. (2013). Financial distress and bankruptcy prediction among listed companies using accounting, market and macroeconomic variables. *International Review of Financial Analysis*, 30, 394-419. https://doi.org/10.1016/j.irfa.2013.02.013