Capstone Project

COMPANY BANKRUPTCY PREDICTION

By – RAHUL SINGH WALDIA

Year of study	Observations	Dimensions	Years to forecast	Bankruptcy after forecasting period
1	7027	64 financial ratios	5	271
2	10173	- ditto -	4	400
3	10503	- ditto -	3	495
4	9792	- ditto -	2	515
5	5910	- ditto -	1	410
Total	**43405**			**2091**

The dataset used for this exercise is the bankruptcy status of Polish companies over a 5-year study period here: . It contains 43405 observations spread over 5 subsets (one per year), with 64 financial ratios for each observation. Some of the companies went bankrupt during the study period (indicated by "1"), while the others survived (indicated by a "0"). A summary for each year in the study period is given .

After due imputing and pre-processing, the data is split into training and test datasets in the 70:30 ratio. Classifiers are built on the training data, and their performance is measured using the confusion matrix for the training and test datasets, as follows:

where TP = number of true positives,
TN = number of true negatives,
FP = number of false positives and
FN = number of false negatives.

These can be used to compute the following metrics:

Number of observations, n = TN+FP+FN+TP
Number of errors = FP+FN
Misclassification (error rate) = FP+FN / n
Sensitivity (true positive) = TP / FN+TP
False positive = FP / TN+FP
Specificity (true negative) = TN / TN+FP
Precision = TP / FP+TP
Prevalence = FN+TP / n
Accuracy = TN+TP / n

Actual	Predicted	
	0	1
0	TN	FP
1	FN	TP

Different machine learning algorithms were built for this exercise . Logistic Regression,Perceptron as a classifier Deep Neural Network Classifiers (with different size and depth),Fischer Linear Discriminant Analysis.K Nearest Neighbor Classifier (with different values of k),Naive Bayes Classifier,Decision Tree (with different bucket size thresholds),Bagged Decision Trees,Random Forest (with different tree sizes),Gradient Boosting,Support Vector Machines (with different kernels). Some algorithms were built with different initializations, and the performance of each was documented in order to find the best initialization. For example, for the K-Nearest Neighbor, different values of k, ranging from 1 to 19 were tried. The performance of each is shown below color coded from red to green, with red indicating 'poor' and green indicating 'good':We see that as k increases, the error rate (as well as Accuracy) decreases and then starts to increase again. Sensitivity increases with k, while Specificity decreases as k increases.

k	no_errors	error rate	sensitivity	false_positive	specificity	precision	prevalence	accuracy
1	1021	7.840577	18.94	3.962525	95.84	18.61	0.91	92.16
2	988	7.58716	19.06	4.031639	95.78	17.19	0.84	92.41
3	701	5.383198	34.84	4.277377	95.65	12.15	0.59	94.62
4	706	5.421594	33.64	4.300415	95.63	11.67	0.57	94.58
5	648	4.976194	43.86	4.484718	95.48	7.89	0.38	95.02
6	642	4.930118	46.67	4.438642	95.52	8.83	0.43	95.07
7	621	4.768853	58.23	4.515435	95.46	7.26	0.35	95.23
8	624	4.791891	56.41	4.530794	95.44	6.94	0.34	95.21
9	622	4.776532	61.11	4.615266	95.37	5.21	0.25	95.22
10	624	4.791891	61.36	4.661342	95.32	4.26	0.21	95.21
11	625	4.79957	60.98	4.676701	95.31	3.94	0.19	95.20
12	624	4.791891	63.16	4.68438	95.30	3.79	0.18	95.21
13	624	4.791891	65.63	4.707418	95.28	3.31	0.16	95.21
14	625	4.79957	66.67	4.730456	95.26	2.84	0.14	95.20
15	625	4.79957	66.67	4.730456	95.26	2.84	0.14	95.20
16	626	4.807249	68.18	4.753494	95.24	2.37	0.12	95.19
17	626	4.807249	72.22	4.768853	95.22	2.05	0.10	95.19
18	626	4.807249	72.22	4.768853	95.22	2.05	0.10	95.19
19	628	4.822608	68.75	4.784211	95.21	1.74	0.08	95.18

Similarly, we tried different number of trees for Random Forest, ranging from 50–500 in jumps of 50:The model performs best with 200 trees.

trees	no_errors	error rate	sensitivity	false_positive	specificity	precision	prevalence	accuracy
50	456	3.50	83.21	3.16	96.78	35.17	1.71	96.50
100	450	3.46	84.07	3.13	96.81	35.80	1.74	96.54
150	445	3.42	84.87	3.10	96.83	36.28	1.77	96.58
200	437	3.36	85.05	3.03	96.90	37.70	1.84	96.64
250	454	3.49	83.33	3.14	96.79	35.49	1.73	96.51
300	452	3.47	83.21	3.12	96.82	35.96	1.75	96.53
350	446	3.42	84.06	3.09	96.85	36.59	1.78	96.58
400	441	3.39	85.09	3.07	96.86	36.91	1.80	96.61
450	449	3.45	84.64	3.13	96.80	35.65	1.74	96.55
500	448	3.44	84.44	3.12	96.82	35.96	1.75	96.56

We made a comparison chart of all the models that we built. The models are seen to perform differently based on the hyper-parameters used, as well as the metric chosen for measuring performance. An overview of the metrics on Training data is given below.

On Training data	Logistic regression	Perceptron	DNN (300, 200)	DNN (300, 300, 200)	DNN (300, 400, 500, 400, 300)	DNN (300,1000, 200)	KNN
Time to fit the model (sec)	198.00	35.00	240.00	334.00	1071.00	830.00	Training not relevant for the K Nearest Neighbour algorithm
Number of errors	1452	2101	1283	1316	1335	1437	
Misclassification (error rate)	4.78	6.92	4.22	4.33	4.39	4.73	
Sensitivity (true positive)	1.03	3.91	17.02	12.15	10.84	1.99	
False positive	0.03	2.42	0.26	0.12	0.12	0.03	
Specificity (true negative)	99.97	97.69	99.76	99.88	99.88	99.97	
Precision	1.03	3.91	17.02	12.15	10.84	1.99	
Prevalence	4.80	4.80	4.80	4.80	4.80	4.80	
Accuracy	95.22	93.08	95.78	95.67	95.61	95.27	

On Training data	Fischer LDA	Naïve Bayes	Decision Tree	Bagged DT	Random Forest	Gradient Boosting	SVM (linear)	SVM (radial)
Number of errors	1466	28376		33	7.00	40	1549	57
Misclassification (error rate)	4.83	93.39	3.56	0.11	0.02	0.13	5.10	0.19
Sensitivity (true positive)	40.00	4.74	99.98	100.00	100.00	100.00	16.18	100.00
False positive	4.74	0.16		0.11	0.02	0.13	4.72	0.19
Specificity (true negative)	95.26	92.57	26.15	99.89	99.98	99.86	95.26	99.80
Precision	1.24	96.71	96.41	97.74	99.52	97.25	1.51	96.09
Prevalence	0.06	4.64		4.69	4.77	4.66	0.07	4.61
Accuracy	95.17	6.61	96.44	99.89	99.98	99.87	94.90	99.81

An overview of the metrics on Test data is as follows:

On Test data	Logistic regression	Perceptron	DNN (300, 200)	DNN (300, 300, 200)	DNN (300, 400, 500, 400, 300)	DNN (300,1000, 200)	KNN
Time to score (sec)	0.39	0.43	0.65	0.91	2.31	1.87	
Number of errors	645	926	592	601	604	634	*1810*
Misclassification (error rate)	4.95	7.11	4.55	4.62	4.64	4.87	7.84
Sensitivity (true positive)	0.16	1.58	14.35	9.46	8.68	0.79	18.94
False positive	0.10	2.44	0.40	0.22	0.20	0.04	3.96
Specificity (true negative)	99.91	97.68	99.62	99.79	99.81	99.96	95.84
Precision	0.16	1.58	14.35	9.46	8.68	0.79	18.61
Prevalence	4.87	4.87	4.87	4.87	4.87	4.87	0.91
Accuracy	95.05	92.89	95.45	95.38	95.36	95.13	92.16
Time to Train and Test (sec)							1810.37

On Test data	Fischer LDA	Naïve Bayes	Decision Tree	Bagged DT	Random Forest	Gradient Boosting	SVM (linear)	SVM (radial)
Number of errors	1149	12172		365	437	355	680	673
Misclassification (error rate)	8.82	93.47	3.55	2.80	3.36	2.73	5.22	5.17
Sensitivity (true positive)	4.75	4.82	99.98	87.26	85.05	87.20	4.00	10.20
False positive	4.66	0.15		2.45	3.03	2.36	4.85	4.83
Specificity (true negative)	95.13	92.52	28.39	97.48	96.90	97.57	95.13	95.15
Precision	4.26	97.00	96.46	49.68	37.70	51.58	0.32	0.79
Prevalence	0.21	4.72		2.42	1.84	2.51	0.02	0.04
Accuracy	91.18	6.53	96.50	97.20	96.64	97.27	94.78	94.83
Time to Train and Test (sec)	1.46	1000.36	29.28	62.83	806.57	139.96	1162.08	1195.71

Some findings

The dataset has a large number of ‘0’s (companies not going bankrupt), and very few ‘1’s (companies going bankrupt). As a result, most models give high Accuracy, but perform poor when predicting a ‘1’. Since the task is to predict bankruptcy, we believe that we must focus on Sensitivity (True Positive) as the relevant measure for comparing the models.

We find that the ensemble models such as Gradient Boosting and Bagged Decision Trees perform best on Training and Test datasets, outperforming even the neural network algorithms. They are also computationally fast, completing the training and scoring in a couple of minutes. The model that is seen to perform the poorest is the Naïve Bayes model, which has resulted in a high number of errors and a poor Sensitivity score.