Bankruptcy Prediction
on
Real World Dataset
using
Machine Learning Algorithms

In partial fulfillment for course CIS 520 by
Vishal Karmalkar
Shalin Shah
Akshay Rajhans

Table of Contents

1. **Project Objective** 3
2. **Data Description** 3
3. **Detailed Description of Dataset** 4
4. **Data Standardization** 4
5. **Algorithms used** 4
   (A) Decision Tree 4
   (B) Linear regression 6
   (C) Boosting 7
   (D) KMEANS 8
   (E) Logistic Regression 8
6. **Conclusions** 9
7. **Future Work** 9
8. **References** 9

1. Project Objective

Prediction of bankruptcy is a phenomenon of increasing interest to firms who stand to loose money because on unpaid debts. Since computers can store huge dataset pertaining to bankruptcy making accurate predictions from them before hand is becoming important. In this project we will use various classification algorithms on bankruptcy dataset to predict bankruptcies with satisfying accuracies long before the actual event.

2. Data Description

For the purview of this project we have used bankruptcy dataset provided to us, in five different data sets. The features of the datasets are as follows

```
Datasets        Bankruptcies   total number observations
bank_1.data         458               20000
bank_2.data         442               20000
bank_3.data         467               20000
bank_4.data         431               20000
bank_5.data         446               20000
```

In order to reduce complexity and scale of computations we have decided to use complete bank_1.data and only the bankruptcies from other datasets. Hence now our distribution is

Total Non Bankruptcies = 20000-458
= 19542

Total Bankruptcies = 458+442+467+431+446
= 2244

Total Observation = 19542+2244
= 21786

Also for such a kind of problem only the accuracy of prediction is not important but also the payoff. For example it is much more profitable to predict a non bankruptcy as a bankruptcy than vice versa as the company stands to loose much more in the latter scenario.

Hence we introduce the concept of a payoff matrix/confusion matrix which gives an indication of the penalty for false positives and true negatives.

		Prediction	
		YES	**NO**
Real	**YES**	0	100
	NO	1	0

Where YES => Bankruptcy
NO => Non Bankruptcy

3. Detailed Description of Dataset

As stated above we have 21786 observations. Now each observation has 148 features associated with it. The first feature is a binary feature which represents Bankruptcy or not. This is considered as the output feature Y. The next 146 features are descriptive features and are considered as X. The detailed description of the features can be found here

http://www.seas.upenn.edu/~cis520/Data/bankruptcy/bank.names

4. Data Standardization

For the purview of this project it was very important to find standardize categorical data into numeric one as otherwise it would be difficult to upload in Matlab. For categorical feature X having possible values as say A, B, C, D we broke up the feature into 3 distinct ones like X_A, X_B and X_C. Now if the original feature had value of A then only X_A had a value of 1 and rest all are 0. Thus we converted categorical features into numeric one.

Missing values was another area of concern where we added an Indicator feature which has value 1 when the feature that it corresponds to has value missing. If the feature has value which is present then indicator function has value of 0. We add indicator variable here because we are assuming that data is not missing at random. Otherwise we would have replaced the missing value with the mean of the feature.

Thus now our total dataset had 21786 observations with 402 features. Now if we consider only 1^{st} level interactions the total number of features would have been blown to 80601. Calculating such a huge dataset would have been beyond the time and space complexities of the facilities available. Hence we did not consider full fledged first level interactions beforehand.

5. Algorithms used

(A) Decision Tree

Since we could not expand all the 1^{st} level interactions of the given dataset we first used feature selection algorithm stepwise regression to reduce the number of features down to 42. Once we had 42 features in hand the problem of interactions terms was solved. So now we computed all the 1^{st} level interactions and the total number of features formed were 861.

Now we run Decision tree algorithm on this dataset where 50% of values were used to train the tree and rest 50% were used to find how well the tree has learnt. The training and testing errors are as follows. The first level split was on feature X21

Training Error

Bankruptcies classified as Non Bankruptcies (-1)	=	17
Non Bankruptcies classifies as Bankruptcies (1)	=	16

The Training dataset contained

Bankruptcies	=	1118
Non Bankruptcies	=	9775

Test Error

Bankruptcies classified as Non Bankruptcies (-1)	=	115
Non Bankruptcies classifies as Bankruptcies (1)	=	143
Bankruptcies	=	1126
Non Bankruptcies	=	9767

Test Set Accuracy

Bankruptcies classified as Non Bankruptcies (-1)	=	(1126-115)/1126
	=	**89.78%**
Non Bankruptcies classifies as Bankruptcies (1)	=	(9767-143)/9767
	=	**98.53%**

Now we repeated the procedure once again of feature selection on 1st level interactions dataset consisting of 402 features. This time we got a total of 60 features for which we got a total of 1803 features when we considered 1st level interactions. We ran the decision tree algorithm once again and got the following results. Again the first level split was on feature X21.

Training Error

Bankruptcies classified as Non Bankruptcies (-1)	=	15
Non Bankruptcies classifies as Bankruptcies (1)	=	9

Test Error

Bankruptcies classified as Non Bankruptcies (-1)	=	96
Non Bankruptcies classifies as Bankruptcies (1)	=	109

Calculating Test Set Accuracy as before since the number of observations remain the same

Test Set Accuracy

Bankruptcies classified as Non Bankruptcies (-1)	=	(1126-96)/1126
	=	**91.47%**
Non Bankruptcies classifies as Bankruptcies (1)	=	(9767-109)/9767
	=	**98.88%**

Hence we can see that as we run feature selection on original feature set and then explode to consider all interactions the accuracy of predictions goes on increasing. This can be explained by fact that when we do such a process we are in fact calculating first level interactions between important features selected by feature selection. At the first stage itself if we could consider all the 80601 interactions then we would get the best predictions but this is not possible due to computational complexities.

(B) Linear regression

On the original dataset we carried out Linear Regression and the results were as follows

Training Error

Bankruptcies classified as Non Bankruptcies (-1)	=	269
Non Bankruptcies classifies as Bankruptcies (1)	=	491

The Training dataset contained

Bankruptcies	=	1118
Non Bankruptcies	=	9775

Test Error

Bankruptcies classified as Non Bankruptcies (-1)	=	531
Non Bankruptcies classifies as Bankruptcies (1)	=	293
Bankruptcies	=	1126
Non Bankruptcies	=	9767

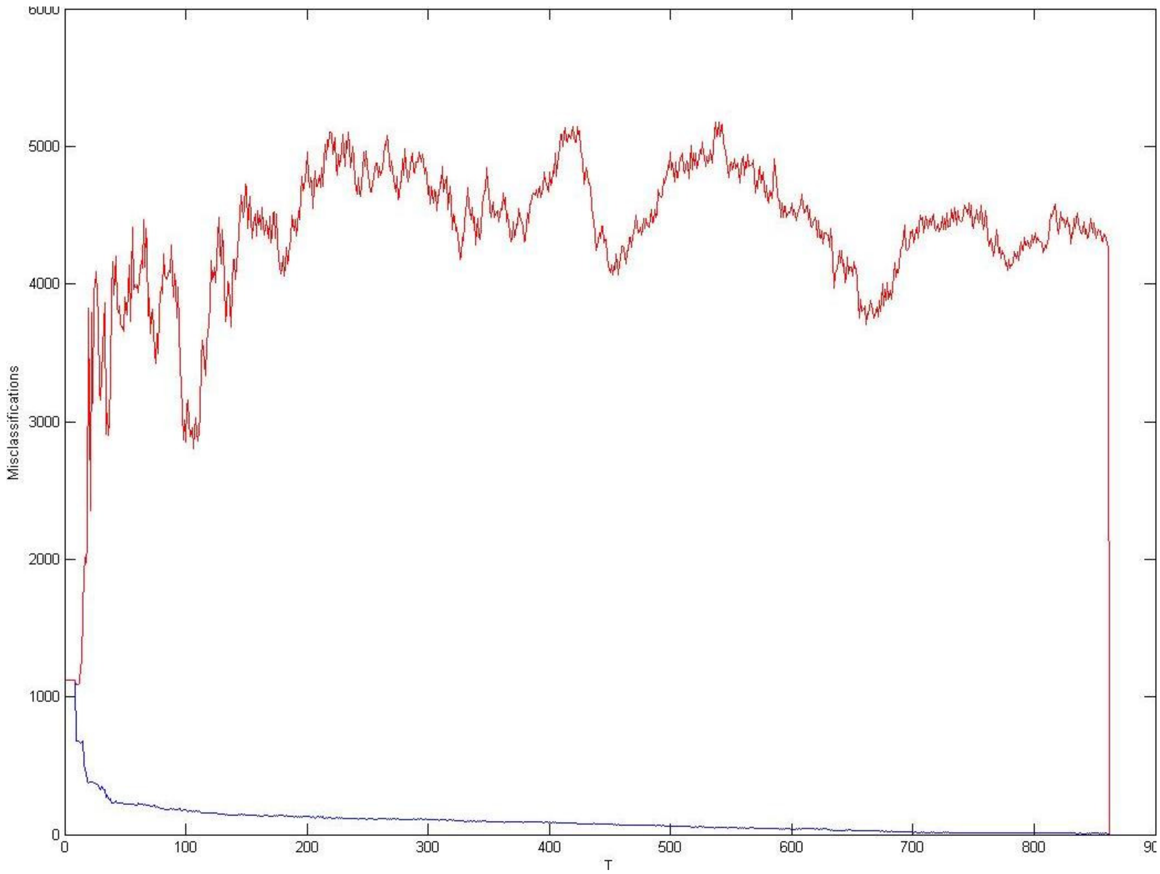
Test Set Accuracy

Bankruptcies classified as Non Bankruptcies (-1)	=	(1126-531)/1126
	=	**52.84%**
Non Bankruptcies classifies as Bankruptcies (1)	=	(9767-293)/9767
	=	**97.00%**

As we Linear Regression performs poorly. Hence we can conclude that the dataset is not linearly separable.

(C) Boosting

In boosting we have tried to use every feature as a weak learner and predict the set of w based upon decision stumps. However this approach does not work and boosting performs poorly on dataset as shown below.



Here we can clearly see as training error decreases there is no appreciable decrease in test error. Test error here is highest as compared to any of the above methods and does not show any particular trends also. This might be because of the fact that we have used a single feature as a weak leaner and run the algorithm for number of features. Hence every feature should ideally give an error < 0.5 (better than random) but it seems that this is not what is happening. What we have used here is a very naïve form of boosting where every feature is considered as weak learner. Some kind of greedy approach or feature selection approach might have worked better. Hence Naive Boosting is not a suitable method to use for this dataset.

(D) KMEANS

Within KMEANS method we tried classifying the data using 2 clusters bankruptcy or Non Bankruptcy. However the results for KMEANS algorithm were very poor.

(E) Logistic Regression

On the same dataset we carried out Logistic Regression and the results were as follows

Training Error

Bankruptcies classified as Non Bankruptcies (-1)	=	147
Non Bankruptcies classifies as Bankruptcies (1)	=	183

The Training dataset contained

Bankruptcies	=	1118
Non Bankruptcies	=	9775

Test Error

Bankruptcies classified as Non Bankruptcies (-1)	=	162
Non Bankruptcies classifies as Bankruptcies (1)	=	205
Bankruptcies	=	1126
Non Bankruptcies	=	9767

Test Set Accuracy

Bankruptcies classified as Non Bankruptcies (-1)	=	(1126-162)/1126
	=	**85.61%**
Non Bankruptcies classifies as Bankruptcies (1)	=	(9767-205)/9767
	=	**95.80%**

Logistic Regression performs exceedingly well on the given dataset. However the Time complexity of this algorithm is very high where it required close to 2 days getting the above set of results.

6. Conclusions

Of all the algorithms we applied on bankruptcy dataset we observed that only Decision Tree and Logistic Regression gave us satisfactory results. An interesting thing to observe is in both cases of Decision Tree the 1st level split was on feature X21. If we had an idea about what the feature meant it would have served to improve our understanding of the problem. Decision tree out performing every other algorithm here shows us the importance of interactions for this dataset. Referring to the analysis of the same dataset by Dean P Foster and Robert A Stine we see interaction terms like Number of credit cards, prior cards past 60 days, late charge prior month appear frequently in all models. Hence we assign a score to every observation based on predictions by decision tree and decide on proper course of action.

7. Future Work

- Although standard decision tree function provided by Matlab has a facility to provide a cost function we observed that providing one does not change the predictions in any way. Hence there is need to explore a way in which Decision Tree code can be improved to include classification based on cost function
- The predictions derived from random forests for same dataset would give us a fair idea of variance and important features/interactions.
- We could not explode the original dataset into 1st level interactions because of computational complexity. If a way around is found then this would definitely give better results than what we have achieved

8. References

Dean P. Foster and Robert A. Stine "Variable Selection in Data Mining: Building a Predictive Model for Bankruptcy"

http://www.cis.upenn.edu/group/datamining/ReadingGroup/papers/foster-stine.pdf