ABDO ELIE-DIMITRI (260948346), DANGEARD ANTOINE (260962884), CHAI YUTING (260888064)

COMP 551 MACHINE LEARNING- MINI-PROJECT 1

Abstract: This study investigates the performance of two very common machine learning models — linear regression and logistic regression – on two benchmark datasets. To optimize our models, we used the closed form for linear regression, as well as gradient descent on both the logistic and linear regression models. We implemented mini-batch stochastic gradient descent with optional momentum, regularization, non-linear base function and learning rate arguments.

Keywords: Linear Regression, Logistic Regression, Mini-Batch Stochastic Gradient Descent, Regularization, Gradient Descent with Momentum

I- INTRODUCTION

The effectiveness of the linear regression and logistic regression models is examined on two benchmark datasets in this study. Linear regression is applied to the energy efficiency dataset, consisting of eight features for various building attributes. Its outputs reflect the building's heating load and cooling load. Logistic regression is applied to the qualitative bankruptcy dataset, which contains six discrete features, to determine a company's possible bankruptcy. We examine four variations of these two algorithms – an analytical model and a mini-batch gradient descent model for linear regression, a fully-batched model and a mini-batch gradient descent model for logistic regression. Also, we experiment with various machine learning techniques to optimize our models, such as adding momentum to gradient descent, using non-linear bases, regularization, and varied learning rates. The models are tested with different parameters and their performance is measured using the MSE (mean squared error) for the linear models and the F1 score for the logistic models. It should be noted that similar works exists testing these models with other methods, which we have not included, such as Nesterov Momentum and Adaptive Gradient (AdaGrad) [(1)], as well as Multiple Gradient Descent Algorithm (MGDA) [(2)].

II- DATASETS

The energy efficiency dataset is used for regression, which is a dataset whose features represent eight different building characteristics (like relative compactness, surface area, wall area, overall height, etc.) and whose outputs represent the heating load (Y1) and cooling load (Y2) of the building. The qualitative bankruptcy dataset maps six discrete features (like industrial risk, management risk, financial flexibility) to ‘B’ (bankruptcy) or ‘NB’ (non-bankruptcy).

We noticed when visualizing the features and outputs of the energy efficiency data that the features fall on different scales of magnitude. The energy efficiency dataset is also unbalanced, containing relatively few samples with heating/cooling load around 20-25. We can also see that the qualitative bankruptcy dataset is imbalanced: i.e., the features and outputs contain uneven amounts of each class. We processed the data sets by standardizing the continuous features and outputs to a mean of 0 and standard deviation of 1 and converted the discrete values to numerical values representative of their meaning (i.e., mapping negative to 0, average to 1 and positive to 2, etc.).

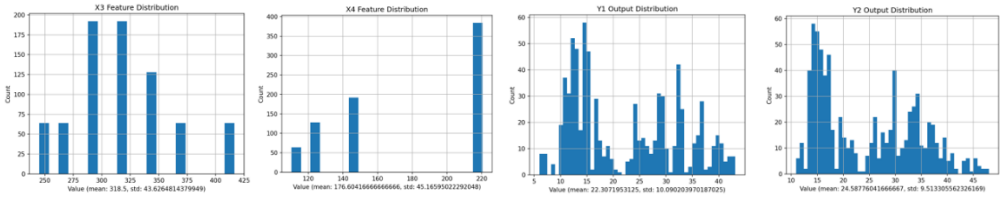

Figure 1.1. Distribution of some of the features and targets of the energy efficiency data set (X3, X4, Y1, Y2)

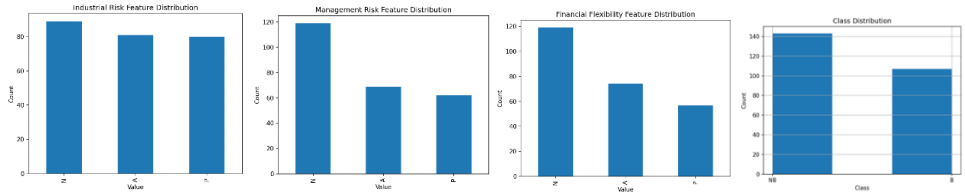

Figure 1.2. Distribution of some of the features and targets of the qualitative bankruptcy data set (Industrial Risk, Management Risk, Financial Flexibility, Output)

(see the full Figure 1.1 and Figure 1.2 in the APPENDIX)

Some ethical concerns with these datasets may arise depending on what the models trained on them will be used for in a real-world application. Imbalances or bias contained in the dataset could be learned by a model, and if that model is used to determine the financial viability of a company, or the equipment required to temperature control a building, then it could incur heavy financial losses if the model is incorrect in its prediction.

III- RESULTS

The experiments have been run with "experiments.py", which includes all the functions used. The implementation of these functions is specified in table 1 of the appendix.

1. Performance of linear regression and fully batched logistic regression with 80/20 train/test split

In order to evaluate the performance of the models, the functions "mse" and "f1_score" are used in "experiments.py". The following models are trained and tested:

- Linear Regression (analytical) training error (MSE): **8.17432877145204**
- Linear Regression (analytical) test error (MSE): **9.79669958240376**
- Logistic Regression (full batched) training performance (F1 score): **0.9328063241106719**
- Logistic Regression (full batched) test performance (F1 score): **0.8771929824561403**

2. Weights of each features in models

$$\begin{bmatrix} w_1 & w_2 \end{bmatrix} = \begin{bmatrix} -7.986953 & -8.089112 \\ -3.561091 & -2.947977 \\ 0.266653 & -0.531627 \\ -5.492378 & -5.292383 \\ 6.861513 & 7.410312 \\ -0.024370 & 0.063008 \\ 2.759717 & 2.082236 \\ 0.319003 & 0.086381 \\ 22.286956 & 24.547127 \end{bmatrix} \qquad w = \begin{bmatrix} -0.15094709 \\ 0.08877941 \\ 0.58808468 \\ 0.52001674 \\ 0.76169582 \\ -0.02715301 \\ -0.4602182 \end{bmatrix}$$

Figure 2.1. The weights for linear regression model (analytical, 80% train size) for the energy efficiency dataset (left), and the weights for the logistic regression model (full-batched, 80% train size) on the qualitative bankruptcy dataset (right)

From these weights, we can see how the different features affect the model output. For example, for the energy efficiency dataset, we can see that X8 (glazing area distribution) has a large positive correlation to heating and cooling load for buildings. For our logistic regression model, we can see that features X1 (Industrial Risk) and X5 (Competitiveness) have very small values, meaning our model has determined that these features have a small impact on the bankruptcy decision for a given company.

3.Sampling growing subsets of the training data (20%,30% ->80%)

This experiment is run with the function "runTrainSizeExperiment". As shown in figure 3.1, the MSE on the test set increases as the size of the training size grows past 50%, while the MSE on the training set continues to decrease. This is clearly a result of overfitting, as the model essentially learns the data points in the training set instead of their distribution, and performs poorly when given unseen data from the test set. On the opposite end, for a training size that is too small, the model also performs poorly as the model does not fit the data enough, this is underfitting.

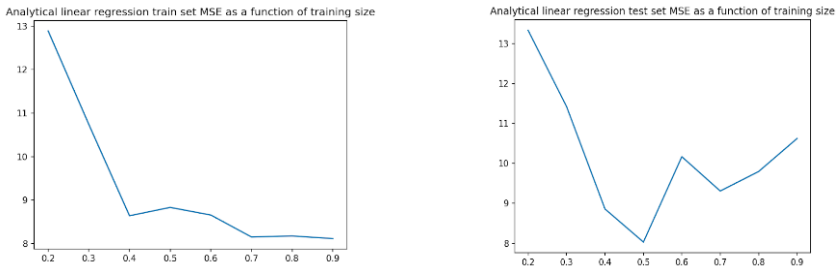

Figure 3.1. Graphs of MSE of Analytical Linear Regression Train (a) / Test (b) as a function of training size.

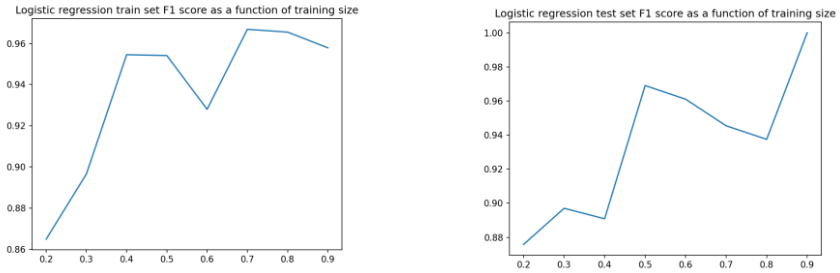

Figure 3.2. Graphs of F1 score of Logistic Regression with Gradient Descent Train (a) /Test (b) as a function of training size.

This experiment clearly demonstrates the importance of sizing the training size accordingly in order to not overfit or underfit the data. According to graphs a) and b) in figure 3.1, a 50/50 split seems to be optimal for the linear regression model. A similar effect is observed in figure 3.2 for the logistic regression model, where overfitting and underfitting results in a decrease of the F1 score for the test set. Again, a 50/50 split of the training and test sets seems to be optimal in this case.

4. Growing minibatch sizes, 8, 16, 32, 64, 128 and full.

The following experiment is run with the function “runMiniBatchExperiment”.

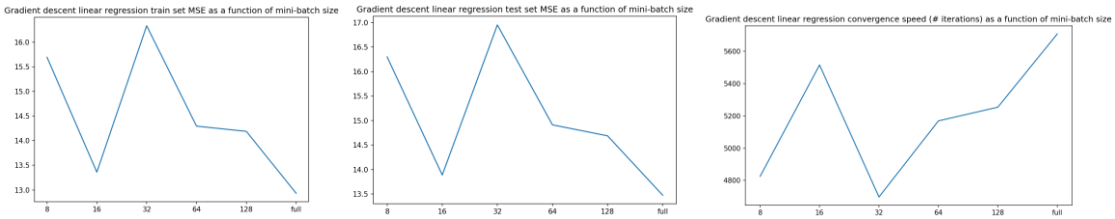

Figure 4.1. Graphs of MSE of Mini Batch Linear Regression Train (a)/Test (b)/Number of iterations (c) as a function of batch size.

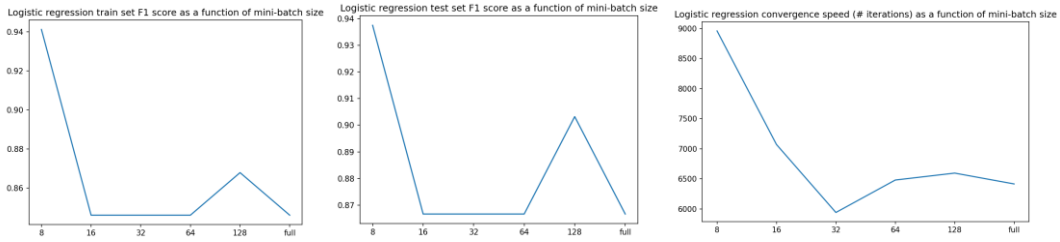

Figure 4.2. Graphs of F1 score of Mini Batch Logistic Regression Train (a) / Test (b) / Number of iterations (c) as a function of batch size.

After careful analysis of the norms of the gradient vectors for the linear and the logistic regressions, we have determined that the epsilon for gradient descent should be set at 1e0 and 1e-1 respectively in order to obtain results for number of iterations (otherwise GD maxes out at 10000). The MSE/F1 scores are compared with the fully-batched ones as well as the convergence speed (in number of iterations).

In figure 4.1, graph a) and graph b) show that a batch size of 16 results in a small MSE, comparable to fully-batched MSE, as well as a lower number of iterations than fully-batched, shown in graph c). Therefore, we conclude that a batch size of 16 is optimal for the mini batch linear regression as it gives a good trade-off between computational resources and performance.

In figure 4.2, graph a) and graph b) show that a batch size of 8 results in a higher F1 score than fully-batched despite a larger number of iterations, shown in graph c). Therefore, we conclude that a batch size of 8 is optimal for the mini batch logistic regression, although more expensive.

5. Performance of both linear and logistic regression with 4 different learning rates.

The following experiment is run with the function “runLearningRateExperiment”.

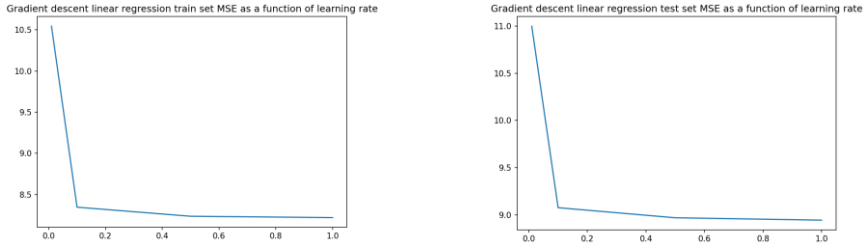

Figure 5.1. Graphs of MSE of Linear Regression Train (a) / Test (b) as a function of learning rate.

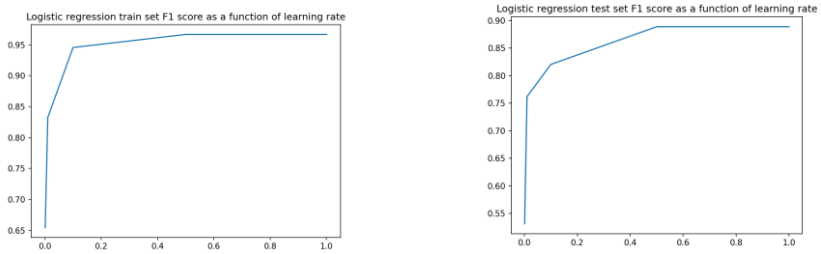

Figure 5.2. Graphs of F1 score of Logistic Regression Train (a) / Test (b) as a function of learning rate.

As shown in figure 5.1 graph a) and graph b), the optimal learning rate that results in the smallest MSE for linear regression is 1.0.

As shown in figure 5.2 graph a) and graph b), the learning rates that result in larger F1 scores for logistic regression are those above or equal to 0.5. Similarly, to linear regression, learning rates closer to 1.0 seem to perform the best on these data sets.

6. Comparison of analytical linear regression solution and mini-batch stochastic linear descent

As we have determined in experiment 4, the optimal batch size for mini-batch linear descent is 16. The following is a comparison between analytical linear regression and mini-batch linear regression with a batch size of 16.

	Training Set MSE	Test Set MSE
Analytical Linear Regression	8.1	9.7
Mini-Batch Linear Regression	13.2	14.0

Table 6.1. Comparison of MSE for both training and test for analytical and mini-batch regression

The analytical solution results in a lower MSE for both the training set and the test set. This is expected as a closed form solution is always more precise than an iterative solution.

7. Addition of momentum in gradient descent

We wanted to experiment with adding momentum to the gradient descent because we observed that running linear regression multiple times with gradient descent gave varying MSEs (in a range of 3-4 units) when randomly shuffling the dataset. Our theory was that this was due to the oscillations around the point of convergence. The following experiment is run with the function "runMomentumExperiment", which trains and tests with a momentum of 0.9 multiple times and prints out the MSE for each run.

Run Number	MSE on Training set	MSE on Test set
1	10.499982159761215	10.74700398218756
2	9.735265289110776	10.881025635302576
3	10.433229673407745	11.8875518945858

This experiment has proved our theory to be correct as it is evident from the table above that the values obtained are more uniform even when shuffling the dataset randomly and fall within a smaller range (approximately 1 unit).

8. Addition of regularization in gradient descent

Through analysis of experiment *2. Weights of each of features in models*, we noticed that some of the weights for linear regression are very high. This observation motivated us to experiment with L2 regularization on our linear regression model as we know that this technique makes the values of a model's weights smaller. The following experiment is run with the function "runRegularizationExperiment", and the regenerated new weights w_1, w_2 are demonstrated below:

$$[w_1 \quad w_2] = \begin{bmatrix} -2.775439 & -2.979462 \\ -1.412363 & -1.480159 \\ 1.597868 & 1.101855 \\ -2.148952 & -1.975509 \\ 7.431984 & 7.337662 \\ 0.014940 & 0.164889 \\ 2.629197 & 1.978992 \\ 0.308606 & 0.034184 \\ 21.950446 & 24.162334 \end{bmatrix}$$

L2 Regularization Strength	MSE on train set	MSE on test set
0.1	8.2175173572105	8.9398176949261
1.0	8.234234530159764	8.95998743656291
10	8.668474454743063	9.381323949775089

The conclusion of this experiment is that L2 regularization for this dataset is effective, as our MSEs are much lower than the MSEs recorded on analytical linear regression. We can see that it also influenced the weights, as they are smaller than those produced by analytical linear regression.

9. Data with non-linear bases

We were also curious about adding non-linear bases to our models, in order to understand how they would perform if we assumed the data was distributed in a non-linear pattern. This experiment is run with the function "runNonLinearBasesExperiment". As we can see from figure 9.1, we have small MSEs, comparable to linear regression, when assuming the data follows a polynomial distribution with k=2, 3 and 4, whereas the MSEs are high when assuming the data follows a gaussian or sigmoid distribution. We performed the same experiment on the bankruptcy dataset, but found that none of the non-linear bases had any positive impact on performance, so we have omitted the results.

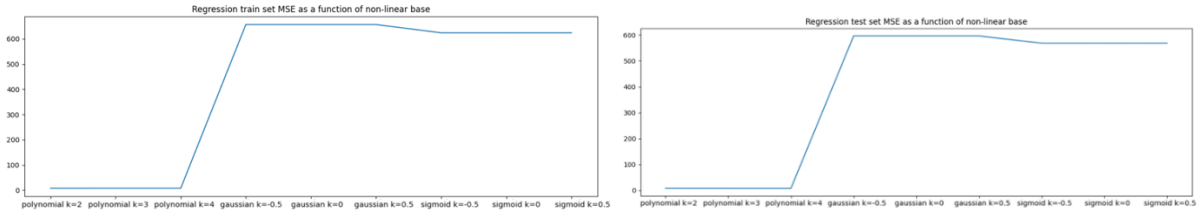

Figure 9.1. Graphs of MSE of Linear Regression Train (a) / Test (b) as a function of data distributions.

IV- DISCUSSION AND CONCLUSIONS

The two ML models, linear regression and logistic regression, were the subject of several tests, which are fully summarised in this study. We came to the conclusion that a 50/50 split between the training set and test set results in the best performance for both logistic and linear regression models. We discovered that, when performance and convergence time were taken into account, batch sizes of 16 for linear regression and 8 for logistic regression were overall optimal. The outcomes, however, demonstrate that the full-batch linear regression still generates a lesser error than the ideal mini-batch one. Four different learning rates were used in our experimentation which concluded that smaller learning rates perform worse than learning rates between 0.6 and 1.0. Additionally, momentum resulted in more uniform errors when running the experiments multiple times with shuffled data. We have also explored the possibility that the data followed non-linear bases. This experiment proved to be successful for regression assuming polynomial bases, but not for logistic regression as the F1 scores were all lower.

One future investigation would be to use adaptive momentum estimation (Adam) to estimate the model. We could compare the model generated by the Adam approach with the gradient descent with momentum approach to see if Adam performs better.

V- STATEMENT OF CONTRIBUTIONS

Antoine Dangeard: Implemented classes for gradient descent regression, analytical linear regression, and regularized regression with non linear bases, implemented data standardization for energy efficiency dataset, implemented functions for experiments, helped with report.

Elie Dimitri-Abdo: Implemented classes for analytical linear regression and gradient descent, standardized energy efficiency dataset, implemented mini-batching for cleaned data, implemented functions for experiments, write up of the report.

Yuting Chai: Cleaned up the dataset, write up the report.

REFERENCES

(1) Saad Hikmat Haji, & Adnan Mohsin Abdulazeez. (2021). COMPARISON OF OPTIMIZATION TECHNIQUES BASED ON GRADIENT DESCENT ALGORITHM: A REVIEW. *PalArch's Journal of Archaeology of Egypt / Egyptology*, *18*(4), 2715-2743. Retrieved from https://archives.palarch.nl/index.php/jae/article/view/6705

(2) Jean-Antoine Désidéri. Multiple-Gradient Descent Algorithm (MGDA). [Research Report] RR-6953, INRIA. 2009, pp.17. ⟨inria-00389811v3⟩

APPENDIX

Functions	Class 1	Class 2
• Analytical linear regression for dataset 1	***LinearRegression*** **Parameters**: default	N/A
• Logistic regression with gradient descent for dataset 2	***RegressionWithBasesAndRegularization*** **Parameters**: non_linear_base_fn = logistic	***GradientDescent*** **Parameters**: batch_size =None (full=None)
• Mini-batch stochastic gradient for linear regression in dataset 1	***RegressionWithBasesAndRegularization*** **Parameters**: default	***GradientDescent*** **Parameters**: batch_size
• Mini-batch stochastic gradient for logistic regression in dataset 2	***RegressionWithBasesAndRegularization*** **Parameters**: non_linear_base_fn =logistic	***GradientDescent*** **Parameters**: batch_size

Table 1. Implementation of the models in "models.py"

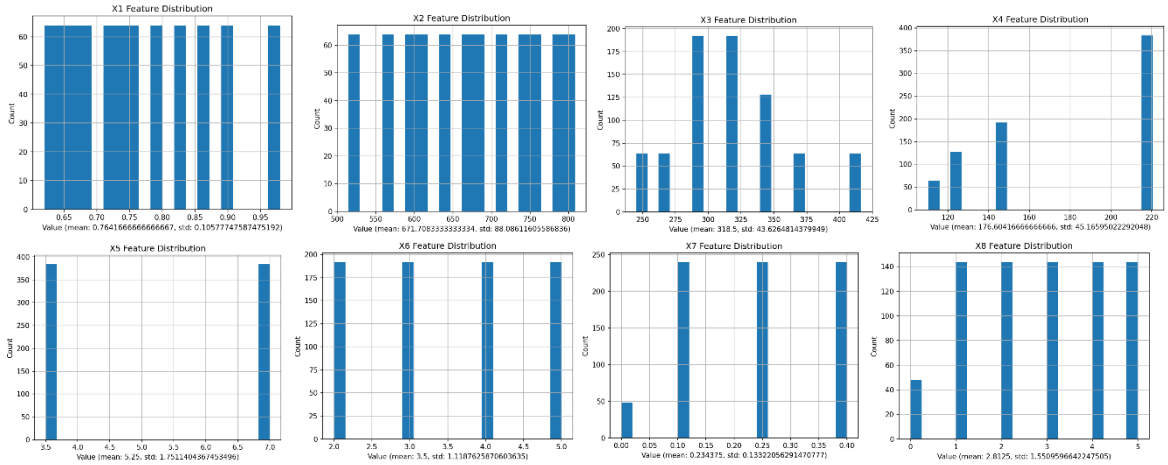

Figure 1.1. Distribution of the features of the energy efficiency data set

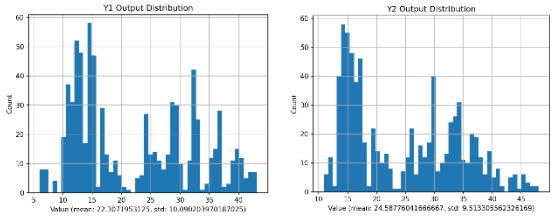

Figure 1.2. Distribution of the targets of the energy efficiency data set (Y1, Y2)

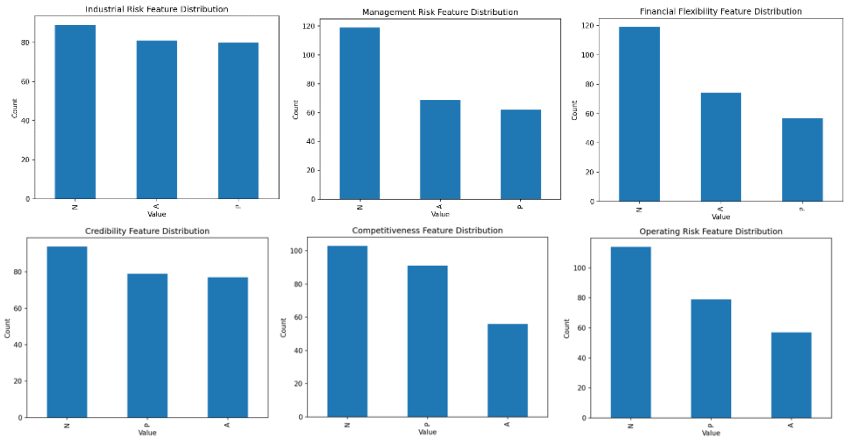

Figure 1.3. Distribution of the features of the qualitative bankruptcy data set (X1-X6)

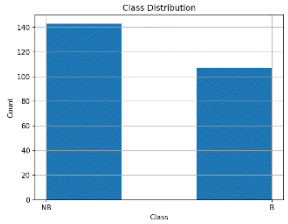

Figure 1.4. Distribution of the targets of the qualitative bankruptcy data set (Y)