COMP 551 - Miniproject #1 Report

Getting Started with Machine Learning

Maxim Boucher - 260902501
Allison Mazurek - 260895254
Youssef Samaan - 261032708

February 9th, 2023

1 Abstract

The goal of this project was to investigate the performance of two machine learning models using the Energy Efficiency [1] and Qualitative Bankruptcy [2] data sets provided by the University of California's Center for Machine Learning and Intelligent Systems database. The former is modeled using linear regression and the latter using logistic regression, with both additionally having batched and stochastic gradient descent implementations. Furthermore, K-Nearest-Neighbour (KNN) technique was implemented on Dataset 2 in order to explore a completely different ML model and compare its performance to the other two more similar models. The conclusion is that Analytical Linear Regression is the most effective model on Dataset 1 and that KNN was the most effective for Dataset 2, though increasing batch sizes and carefully tuning hyper-parameters can yield similar accuracies for the fully-batched and minibatch models. As well, Linear Regression is more sensitive to changes in learning rate than Logistic Regression, but carefully splitting the train and test data is more important for Logistic Regression on Dataset 2.

2 Introduction

This project implements linear and logistic regression on two distinct datasets, respectively, regarding energy consumption [1] and financial stability [2]. A comparison of analytical linear and logistic regression techniques with those combined with mini-batch and stochastic gradient descent algorithms is discussed in detail in Section 4. The results show that Analytical Linear Regression is the most effective model on Dataset 1 and that KNN was the most robust classification model for Dataset 2. Despite this, increasing batch sizes and carefully tuning hyper-parameters can yield similar accuracies for the fully-batched and minibatch models. Furthermore, Linear Regression is more sensitive to changes in learning rate than Logistic Regression, but train and test data must be thoughtfully split for Logistic Regression on Dataset 2.

Dataset 1, the Energy Efficiency Dataset [1], is used to perform energy consumption analysis on residential buildings with 12 different shapes. Eight input variables were used to develop models describing the heating and cooling load requirements of buildings. Statistical machine learning tools previously conducted on this dataset have been found to accurately estimate heating and cooling loads with 's' having 0.5 and 1.5 points of deviation from the ground truth, respectively [3].

Dataset 2, the Qualitative Bankruptcy Dataset [2], is used to assess bankruptcy through an analysis of financial and non-financial features which allows for a predictive model to be developed. This project aims to qualitatively describe an institution's financial status as bankrupt (B) vs non-bankrupt (NB) using logistic regression. A previous work of classifying this dataset, conducted by Myoung-Jong Kim and Ingoo Han [4], indicates that the use of generic algorithms indicates a better performance compared to inductive learning methods and neural network techniques.

3 Acquiring, Pre-processing, and Analyzing the Data

3.1 Dataset 1

Dataset 1, the Energy Efficiency Dataset [1], describes buildings using 8 features listed in Figure 2 and is used in this project to develop models describing a building's heating and cooling load requirements. Probability density distributions of the features and labels, with corresponding statistics in the form of histograms, are displayed in Figure 1. Dataset 1 was found to have no features or labels missing. As seen in Figure 1, the values corresponding to features and labels are within range of each other, and no outlier values are indicated through the probability density distributions. Plots of normalized input variables to output variables, shown in the attached Google Colab, also do not indicate outlier data points. As such, no instances were disregarded, leaving a total of 768 instances to be used and split into relevant training and testing data sets as described in Section 4.

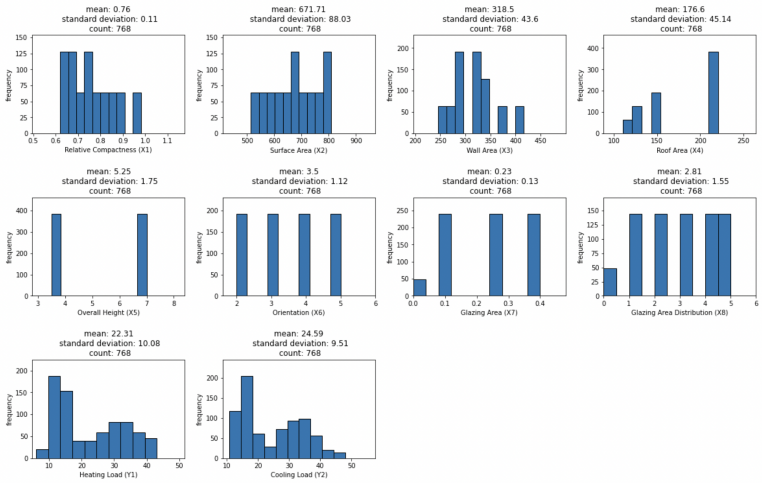

Figure 1: Probability Density of Features and Labels of Dataset 1

Variable Names	Short Form	Possible Values
Relative Compactness	X1	12
Surface Area	X2	12
Wall Area	X3	7
Roof Area	X4	4
Overall Height	X5	2
Orientation	X6	4
Glazing Area	X7	4
Glazing Area Distribution	X8	6
Heating Load	Y1	587
Cooling Load	Y2	636

Figure 2: Input and output variables for Dataset 1

Data correlation between the 8 features was used to showcase similar patterns. A correlation matrix indicates that features X1 and X2, X4 and X5 are highly correlated. Despite this being the case, all features were kept in the dataset for our tests since early trials showed results with nigh-imperceptible differences.

3.2 Dataset 2

Dataset 2, the Qualitative Bankruptcy Dataset [2], is used to assess bankruptcy through an analysis of 6 features, including Industrial Risk, Management Risk, Financial Flexibility, Credibility, Competitiveness, and Operating Risk. Features are described using one of three attributes: positive (P), negative (N), and average (A), which allow for a predictive model to be developed and, in turn, describe an institution's financial status as bankrupt (B) vs non-bankrupt (NB). A per-class representation of each feature is displayed in Figure 3. Dataset 2 was found to have no features or labels missing, resulting in a total of 250 data points. Features *financial flexibility*, *credibility*, and *competitiveness* exhibit a strong correlation between the data points being attributed as

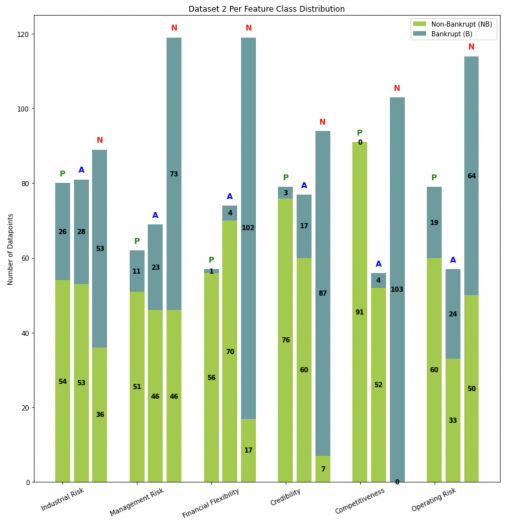

Figure 3: Dataset 2 Features and Label Distribution

P/N and labeled as NB/B, respectively. To represent
these features and labels numerically, one-hot encoding was used as follows:
$x^{(n)} \in \{\text{N, A, P}\} \implies x^{(n)} \in \{[1,0,0],[0,1,0],[0,0,1]\}$
$y^{(n)} \in \{\text{NB, B}\} \implies y^{(n)} \in \{[1,0],[0,1]\}$.

4 Project Results

4.1 Performance and Weights [Subtasks 1-2]

We used an 80/20 train/test split on both datasets before feeding them the model, with both randomized and non-randomized splits. First, we'll study Linear Regression on Dataset 1 and then full batch Logistic Regression on Dataset 2.

Using the analytical Linear Regression approach, we obtained an accuracy of 38.31% with the randomized split, while the non-randomized split yielded an accuracy of 24.84%. It was expected that the randomized split would perform better since similar data entries are clumped together in the provided dataset. Thus, simply cutting off the first 80% misses out on crucial data entries.

The weights obtained with this approach are shown below. To interpret this data, we note that features with high magnitudes will greatly affect the output (either positively or negatively), while smaller magnitudes barely affect it. Thus we find that in this case, Wall, Roof, and Surface area have the most influence by far, while Glazing Area Dist., and Orientation have little to no influence.

Features	w1	w2
Rel. Compactness	-5.80895e+01	-6.62717e+01
Surface Area	1.10413e+11	4.07429e+11
Wall Area	-1.10413e+11	-4.07429e+11
Roof Area	-2.20826e+11	-8.14858e+11
Overall Height	4.02939e+00	4.22342e+00
Orientation	5.08208e-02	1.60203e-01
Glazing Area	1.96617e+01	1.41809e+01
Glazing Area Dist.	2.01899e-01	7.06822e-02
Bias	7.53664e+01	9.18647e+01

Figure 4: LinReg Randomized Data

Features	w1	w2
Relative Compactness	-6.37946e+01	-6.92463e+01
Surface Area	-3.23523e+11	-2.07170e+11
Wall Area	3.23523e+11	2.07170e+11
Roof Area	6.47047e+11	4.14340e+11
Overall Height	4.23779e+00	4.41469e+00
Orientation	-3.21305e-02	1.15029e-01
Glazing Area	2.20744e+01	1.60956e+01
Glazing Area Distribution	2.94948e-01	8.80177e-02
Bias	8.01314e+01	9.19457e+01

Figure 5: LinReg Non-Randomized Data

Now, onto fully-batched Logistic Regression. Running this model with a learning rate of 1 and 100k iterations with a randomized split yields a result with 100% accuracy. Running the same experiment with the non-randomized (naive) split returned an accuracy of 97.96%. Just as before, this is expected since the given data is organized, with the first half being 'NB' entries and the last half being 'B' entries. Without randomization, we have a disproportionate amount of 'B' and 'NB' data entries in our training set, so the accuracy suffers, even if only a little bit in this case.

The full table of weights is omitted to save space. The results show that financial flexibility, credibility, and especially competitiveness have the highest weight (and thus the highest influence on the output) out of all the features by a fair amount; most features have magnitudes between 0 and +/- 1, whereas these have magnitudes at around +/- 3, 4 and 9 respectively. This was expected since (as mentioned in Section 3.2) Dataset 2 analysis revealed that these features are all highly correlated to the Bankruptcy state (especially competitiveness, where one can easily notice that all bankrupt data entries had 'N' competitiveness).

4.2 Varying Subsets of Training Data [Subtask 3]

As described in the P1 handout, we tested the performance of our models (Linear Regression on Dataset 1 and Fully-Batched Logistic Regression on Dataset 2) using growing subsets of train and test data. To

show further evidence and solidify the notion that the non-randomly-split data is not ideal, we split the data naively. The performance of each model according to the training data sizes is shown in the plots below. For Linear Regression, we see that the train data starts out better than the test data, but as the training data size increases, the test accuracy surpasses it. This is because the model adapts to more generalized data as the training set size increases. Note also that it doesn't reach as high a value as it did with randomized data in Subtask 1.

The Logistic Regression plot clearly illustrates the importance of randomizing train/test data splits. Recall that Dataset 2's data is loosely organized: All entries with 'NB' appear in the first half of the file, and all those with 'B' appear in the second half. As we can see, the accuracy of the training data is perfect because we're training and testing on data with the same proportion of 'B's and 'NB's. The test accuracy is more interesting: The accuracy is quite poor and only gets worse as we approach a 50% data split but then sharply improves. This is because from 20% to 50%, the model trains with data containing only 'NB's and tests on data containing 'B's, so it lacks enough information to make correct guesses. Once we pass 50%, the model gets the information it requires to make better guesses, and the accuracy increases accordingly.

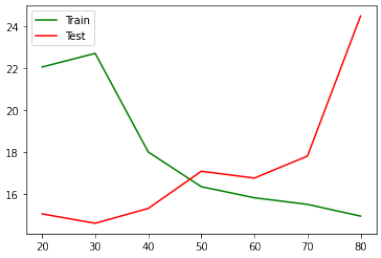

Figure 6: Accuracy Graph for Linear Regression on Dataset 1

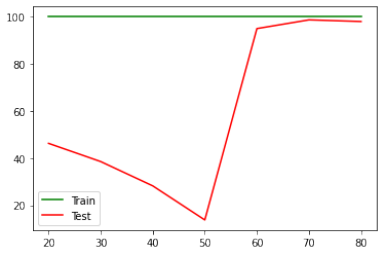

Figure 7: Accuracy Graph for fully-batched Logistic Regression on Dataset 2

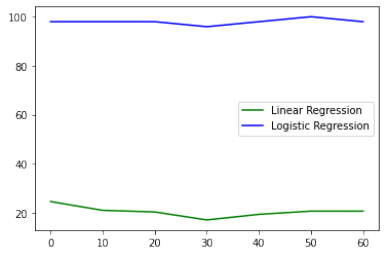

Figure 8: Accuracy Graph for Linear and Logistic Regression as a function of batch size

4.3 Varying Minibatch Sizes [Subtask 4]

In this subtask, we ran the gradient-descent minibatch version of Linear Regression and Logistic Regression with varying minibatch sizes. The results are shown in the plot below. First, we notice that Logistic Regression's accuracy, despite dipping slightly, stabilizes at 97.96%.

Now onto Linear Regression. As one would expect, the overall accuracy of the model increases as the batch size increases and the influence of outlier data points decreases, which means that they will have less of a negative impact in each iteration. Additionally, the number of iterations from our code shows us that the convergence speed increases as the batch size increases. As well, minibatch runs faster when compared to fully-batched because the weights are updated after analyzing a lesser amount of data points each time.

4.4 Varying Learning Rates [Subtask 5]

In this experiment, we varied the learning rates of the fully-batched models for both Linear and Logistic Regression. For Linear Regression, we varied the learning rate from 0.000001 to 0.000002 to 0.000003 and obtained accuracies of 18.63%, 22.22%, and 24.51%, respectively. We also tested against the Sklearn implementation, which has an accuracy of 26.14%. The difference in this result can be explained by the difference in parameters, namely the learning rate and epsilon values.

For Logistic Regression, we varied the learning rate from 0.01 to 0.1 to 1.0 and obtained an accuracy of 97.96% on all three attempts. We can thus conclude that learning rates are much more sensitive to change in Linear Regression than they are in Logistic Regression since very small changes greatly affect the accuracy of Linear Regression, while large changes may not significantly change Logistic Regression

accuracy. However, it is worth noting that early tests have shown that too low a learning rate will still result in a model which does not converge in either model.

4.5 Comparison of Analytical Linear Regression and Mini-Batch Stochastic Gradient Descent [Subtask 6]

The final required experiment asks us to compare analytical Linear Regression on Dataset 1 with Mini-Batch SGD Linear Regression on the same Dataset. In section 4.1, we already reported that analytical Linear Regression returns an accuracy of 38.31%, and in Section 4.3, we've seen various SGD minibatch results. Comparing them, we see that analytical linear regression is more accurate by a fairly significant margin. As discussed in class, convex functions are used because we can easily get their global minimum, and analytical finds such a minimum without too much difficulty by using the least squares solution. We plotted the cost function, and We know the function we are dealing with is convex because its cost function is clearly convex, as shown in our code. Meanwhile, gradient descent takes several small steps towards a minimum and halts after a certain amount of iterations or if the change in cost is smaller than the epsilon value. Therefore, with the right epsilon values and with enough room to iterate, it is entirely possible (and expected) for gradient descent methods to reach a similar result.

4.6 K-Nearest-Neighbours Model

Using our KNN model implementation with 80/20 non-randomly-split training data and a K value of 5 (in order to break ties in the case where a neighbor had the same amount of neighbors in 'B' and 'NB', see code for details), we got a result of 100%. This shows that this technique was simple and even more effective than Logistic Regression with this data since it did not require a random split to obtain perfect accuracy.

5 Discussion and Conclusion

The analytical version of Linear Regression gave us the best accuracy overall for our Linear Regression models at 38.31% and 100% for the Logistic regression with Randomized data splits. Linear Regression was also much more sensitive to learning rates than Logistic Regression. In addition to that, we found that as the batch size increases, the overall accuracy of both Logistic and Linear Regression increases, but the convergence speed decreases. Pre-processing and analyzing data was important to check, as it gives us insight into why we get certain results. Both Logistic Regression and the KNN models performed well. However, the KNN algorithm is more robust in this case since it outperformed the Logistic Regression when given non-randomized data. Adding features like Momentum or Adagrad for Linear Regression would be a worthy venture in order to speed up the convergence speed for the batching methods and bring their accuracy closer to that of the Analytical version. KNN and Logistic Regression have proved to be reliable classification models for dataset 2, so it would be worth looking into further regression models like Gaussian Process Regression or Neural Networks to determine how they compare to Linear Regression.

6 Statement of Contributions

- **Allison Mazurek:** Implemented Task 1 code. Wrote the first 3 report sections. Formatted, revised the report and analyzed results. Conducted research into relevant works connected to this project.
- **Youssef Samaan:** Implemented Task 2 code and KNN. Debugged Task 2&3 code. Ran tests and explained results for the report, refined the code for submission, and helped with the final revisions.
- **Maxim Boucher:** Implemented Task 3 experiments and helped refine them. Helped debug Task 2&3 code. Wrote the 'Results' and 'Discussion' sections of the report and helped revise the report.

References

[1] A. Xifara and A. Tsanas, "UCI Machine Learning Repository: Energy Efficiency Data Set," https://archive.ics.uci.edu/ml/datasets/Energy+efficiency, [Online; accessed 4-Feb-2023].

[2] A. Martin, J. Uthayakumar, and M. Nadarajan, "UCI Machine Learning Repository: Qualitative Bankruptcy Data Set ," https://archive.ics.uci.edu/ml/datasets/Qualitative_Bankruptcy, [Online; accessed 4-Feb-2023].

[3] A. Tsanas and A. Xifara, "Accurate quantitative estimation of energy performance of residential buildings using statistical machine learning tools," *Energy and Buildings*, vol. 49, pp. 560–567, 2012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S037877881200151X

[4] M.-J. Kim and I. Han, "The discovery of experts' decision rules from qualitative bankruptcy data using genetic algorithms," *Expert Systems with Applications*, vol. 25, no. 4, pp. 637–646, 2003. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417403001027