1. Abstract

This project investigated the performance and training dynamics of linear and logistic regression using two benchmark datasets: the Energy Efficiency and Qualitative Bankruptcy datasets. Models were implemented in Python. Gradient descent was used to optimize linear and logistic regression models. Furthermore, the least square (LS) solution was implemented for the linear regression model. The results revealed that proper data preprocessing was important for improving model performance. For instance, normalization improved performance and prevented gradient instabilities from different data scales. Also, there was importance in model regularization, and in choosing an appropriate training ratio, as small training ratios caused underfitting and reduced test set performance. In our project, the addition of subtle L2 regularization (α=0.01) improved performance and brought the model weights to reasonable ranges. For gradient descent optimization, the learning rate was also important as, over equal iterations, models with small learning rates (e.g., 0.0001) converged slowly, while high learning rates (e.g., 1) prevented full convergence. We did not observe huge differences in test set performance and convergence speed between different batch sizes, possibly from the simple and linear datasets. For linear regression models, the analytic solution differed from minibatch gradient descent for several model weights. LS seemed to slightly overfit the data, while minibatch gradient descent found a more general solution. Lastly, momentum improved convergence speed for both models compared with vanilla gradient descent. For logistic models, most tests resulted in extremely accurate predictions, likely from the data provided having high correlation between the parameters and the results.

2. Introduction

This project aimed to provide an implementation of original linear and logistic regression models, using common Python libraries and two benchmark datasets. The effects of different data preprocessing, train-test split ratios, regularization, and several gradient descent hyperparameters were also investigated. The Energy Efficiency Data Set provides a simulated energy analysis of various building shapes. It has also been previously used by Žegklitz et al. to investigate the performance of several symbolic regression models like GPTIPS (2017), and by Yazici et al. to implement and assess three machine learning models, including Random Forests, on a Raspberry Pi as a step toward improving applications of Internet of Things (2018). The Qualitative Bankruptcy dataset provides data on six different factors affecting bankruptcy of companies which was collected from Kim et al. (2003). We found that preprocessing was important for improving model performance and avoiding gradient destabilization due to different feature scales. Regularization was also important as it prevented overfitting and extremely large weights. Having a balanced train-test ratio was also of note as relatively training ratios decreased model performance on unseen test data. We also observed that tuning the learning rate was important in models that utilized gradient descent, as minuscule learning rates cause delayed convergence and large learning rates result in the model's fluctuation. Lastly, substituting vanilla gradient descent with momentum accelerated gradient descent convergence.

3. Datasets

The Energy Efficiency Data Set consisted of **768 samples, 8 features, and two outputs** (Table 1). The collectors implemented energy analysis using 12 different building shapes simulated in Ecotect. Since the two outputs were highly collinear (spearman's rho=0.97), we assumed that a retrained model would perform well on an output (e.g., Y2) if it performs suitably on another (e.g., Y1). Therefore, we reported our results with only one outcome (Y1). On the other hand, the Qualitative Bankruptcy dataset consisted of **250 samples, 6 attributes, and one output** (Table 2). The dataset displays a rating of either P (positive), A (average), or N (negative) for each qualitative parameter in bankruptcy in every instance, and these parameters are industrial risk, management risk, financial flexibility, credibility, competitiveness, and operating risk. Then, each instance has a corresponding outcome class related to these ratings, which is either B (bankruptcy) or NB (non-bankruptcy). One hot encoding with a first-column drop (drop='first') was used to encode categorical data. Z-score normalization was also used in several sections for the numerical data.

Table 1. *Statistical summary of the Energy Efficiency Dataset*

Numerical data	
Feature Name	**Median (IQR)**
Relative Compactness (X1)	0.75 (0.15)
Surface Area (X2)	673.75 (134.75)

Categorical data		
Feature Name	**Subgroups**	**Percentage %**
Orientation (X6)	North (2)	25
	East (3)	25
	South (4)	25

Wall Area (X3)	318.5 (49)		West (5)	25
Roof Area (X4)	183.75 (79.63)		Unknown (0)	6.25
			Uniform (1)	18.75
Overall Height (X5)	5.25 (3.5)	Glazing Area	North (2)	18.75
Glazing Area (X7)	0.25 (0.3)	Distribution (X8)	East (3)	18.75
			South (4)	18.75
Heating Load (Y1)	18.95 (18.67)		West (5)	18.75
Cooling Load (Y2)	22.08 (17.51)			

Numerical values are displayed using median and interquartile range (IQR). Categorical data is displayed in ratios.

4. Experiments

4.1 Overview

The implemented gradient descent models with the following parameters: learn rate=0.005, epsilon=1e-4, tolerance =5 (i.e., if the gradient change was small for 5 consecutive iterations, halt training). Gradient descent weights were initialized using random values from the normal distribution. Furthermore, we fixed the train-test split random state for each experiment block to reduce selection bias. Mean squared error (MSE) and R^2 score were used to assess linear regression performance. AUC, F1score, and logistic loss were used to evaluate logistic models. A train-test ratio of 80/20% was used for sections that did not specifically test the effects of different training ratios. We stratified train-test sets in the classification task to amend unbalanced ratio of output classes (NB: 57.2%, B: 42.8%). Missing instances were removed from the data. For batching, we randomly sorted the data and selected batches without replacement to allow the models to learn from all training instances.

4.2 Effects of preprocessing and regularization

The performance of the LS model on unprocessed inputs was acceptable (e.g., test MSE=8.443, R2=0.909), some LS weights were huge (e.g., 1.78 * 1010), which pointed to the possibility of overfitting, as mentioned in lecture. Furthermore, as the data scales were vastly different without normalization, the gradient descent became unstable. Therefore, in the next step, we z-score normalized the data. Although normalization removed large weights from the gradient descent model (e.g., largest non-bias weight: 7.8), large weights persisted in the LS solution, possibly due to the model overfitting the training data. To address this issue, we added L2 regularization (α=0.01), which resolved the large weight issues (Table 3) and provided decent performance (Table 4). To investigate which features were impactful in the linear regression model, we implemented the model with L1 regularization (β=0.1) over 10000 iterations. As observed in Figure 1.A, X3, X4, X5, and X7 contribute significantly to the model output. Overall, this is in line with the weights from L2 normalized model (Table 3). Finally, we replaced categorical features with corresponding one-hot encoding representations. This resulted in further performance improvement (Table 5). Thus, we used this framework (i.e., normalized input + L2 regularization (α =0.01) + one hot encoding of categorical features) as a testing benchmark for further linear regression sections. Similarly, the logistic regression model's performance improved with L2 regularization (F1: 0.949 AUC: 0.977 vs F1: 0.982 AUC: 0.995). The logistic model with lasso regularization revealed that encoded versions of Financial Flex., Credibility, and Competitiveness significantly impacted model predictions (Figure 1B) in line with weights from L2 regularized model (Table 6).

Figure 1. Weights from linear (**A**) and logistic (**B**) regression models with lasso regularization.

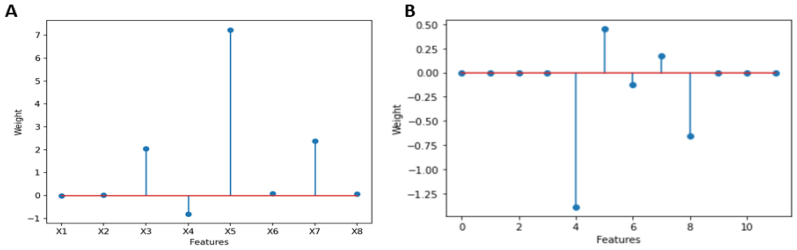

Table 2. *Statistical summary of the Qualitative Bankruptcy Dataset*

Attribute	Rating	Percentage in dataset %	Percent among bankrupt%
	P	32.0	24.3
Industrial Risk	A	32.4	26.2
	N	35.6	49.5
	P	24.8	10.3
Management Risk	A	27.6	21.5

	N	47.6	68.2
	P	22.8	0.9
Financial Flexibility	A	29.6	3.7
	N	47.6	95.3
	P	31.6	2.8
Credibility	A	30.8	15.9
	N	37.6	81.3
	P	36.4	0.0
Competitiveness	A	22.4	0.37
	N	41.2	96.3
	P	31.6	17.8
Operating Risk	A	22.8	22.4
	N	45.6	59.8

Categorical data is displayed in ratios corresponding to each rating's occurrence in every attribute.

Table 3. Feature weights from the L2 regularized LS and gradient descent linear regression models

Feature	LS model	Gradient descent model
Relative Compactness (X1)	-6.745	-2.651
Surface Area (X2)	-3.545	-1.294
Wall Area (X3)	0.703	1.559
Roof Area (X4)	-3.796	-2.026
Overall Height (X5)	7.636	7.178
Orientation (X6)	0.020	0.013
Glazing Area (X7)	2.642	2.655
Glazing Area Distribution (X8)	0.344	0.487

Table 4. Performance of regularized linear regression models

Training set		
Regression Model	**R^2**	**MSE**
Least square solution	0.917	8.551
Gradient descent	0.912	9.109
Test set		
Regression Model	**R^2**	**MSE**
LS solution	0.910	8.441
Gradient descent	0.906	8.762

Table 5. Performance of regularized linear regression models with one hot encoding on inputs

Train		
Regression models	**R^2**	**MSE**
Least square	0.927	7.606
Gradient descent	0.922	8.081
Test		
Regression models	**R^2**	**MSE**
Least square	0.912	8.295
Gradient descent	0.910	8.382

Table 6. Feature weights from the L2 regularized logistic regression model with one hot encoding

	Feature Importance
Attribute	**Weights**
Industrial Risk	-0.314 0.3619
Management Risk	-0.185 0.342
Financial Flexibility	-1.495 0.572
Credibility	-1.270 1.276
Competitiveness	-2.215 1.101
Operating Risk	0.036 0.351

4.3 Testing different training ratios

Next, with linear regression, we aimed to see the effect of different training ratios on model performances. Although the MSE increased on the training set as the training ratio increased (Figure 2 A), possibly due to MSE being calculated over more instances, the test MSE displayed a downward trend overall (Figure 2 B). The observed small differences between performances could be from the effects of randomness in components, like weight initialization. With logistic regression, the AUC remained relatively constant across most training ratios (Figure 2C), with most values for both the test and training dataset being consistently above 0.965, likely due to the high accuracy of the model in general.

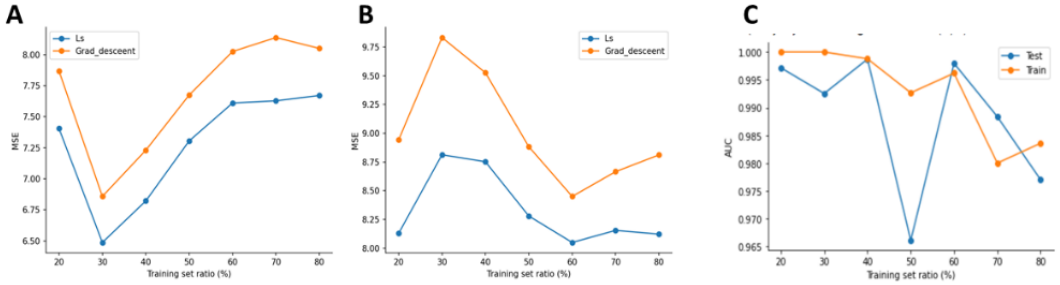

Figure 2. Performance of linear regression models with different training ratios on the training (**A**) and test (**B**) sets. (**C**) Train and test performance of logistic regression models with different training ratios.

4.4 Learning rate and its effect on gradient descent

Learning rates of *[0.0001, 0.001, 0.01, 0.1, 1]* were used to run the linear and logistic regression models. As seen in Figure 3, the learning rate 0.0001 did not provide convergence speeds that allow for sufficient convergence within the equal iterations. In contrast, high-valued learning rates (e.g., 1) converged fast but reduced the test set performance.

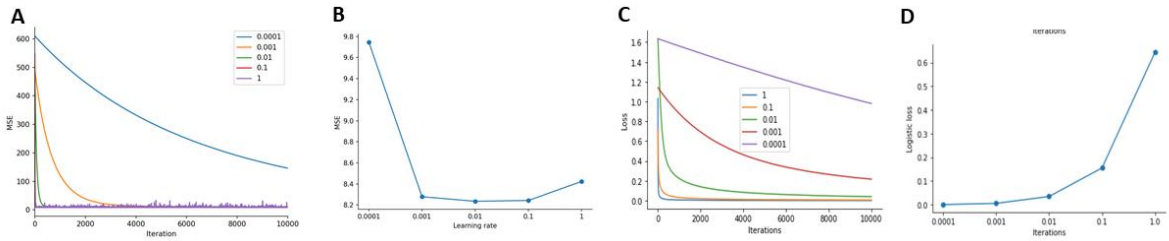

Figure 3. loss dynamics (**A**) and final test performance (**B**) of linear regression with gradient descent. (**C**) and (**D**) similar to (**A**) and (**B**) but for the logistic regression.

4.5 Effect of batch size on model training and performance

Implementation of the gradient descent linear regression with normalization and regularization revealed little difference regarding the convergence speed, measured by MSE in each iteration (Figure 4 A). Although differences were observed between mini-batch and full batch models, the final test MSE differences were relatively small, with the full batch model displaying the best performance (Figure 4 B). Similar results were found for the logistic regression, where the AUC remained consistently above 0.95 (Figures 4 C and 4 D). The models likely settle into slightly different final states in the weight space, causing the observed performance differences.

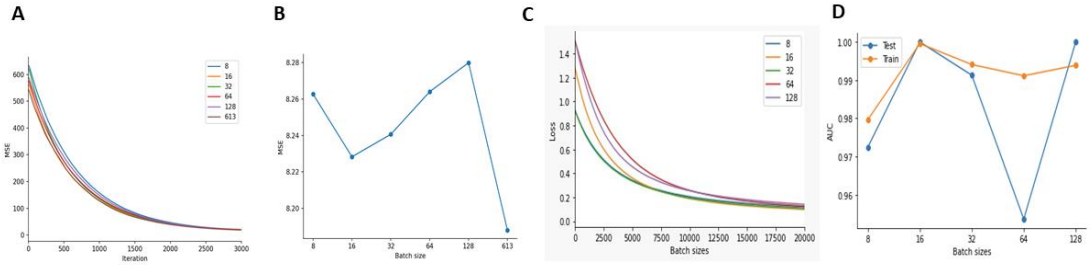

Figure 4. Gradient descent linear regression loss dynamics (**A**) and final test performance (**B**). Gradient descent logistic regression loss dynamics (**C**) and final test performance (**D**).

4.6 Analytic vs mini-batch stochastic gradient descent linear regression models

Next, we opted to investigate the differences between the model using LS closed-form solution and mini-batch stochastic gradient descent model with a batch size equal to 64. Interestingly, although the LS model performed better on the training set, the mini-batch model performed slightly better on the test set (Table 7). It is possible that the LS model weakly overfits the training data, while the minibatch model converges to a different local minimum. This is further supported by the models displaying different weights for several features, an example of which is displayed in Table 8.

Table 7. Performance of LS and mini-batch linear regression models

Train		
Model	**MSE**	**R^2**
LS	7.606	0.928
Mini batch gradient descent	8.041	0.923
Test		
Model	**MSE**	**R^2**
LS	8.295	0.911
Mini batch gradient descent	8.235	0.912

Table 8. Feature weights from LS and mini-batch gradient descent linear regression models

Feature	LS weight	Gradient descent weight
X1	-6.733	-2.489
X2	-3.546	-1.338
X3	0.695	1.712
X4	-3.792	-2.148
X5	7.660	7.209

4.7 Effect of momentum of the convergence speed

Lastly, we implemented momentum for both linear and logistic regressions and compared its convergence speed with regularized gradient descent. Experimentally, we multiplied linear regression initialized weights by 1000 to highlight the differences. As expected, the momentum algorithm showed faster convergence compared to classic gradient descent (Figures 5A, 5B).

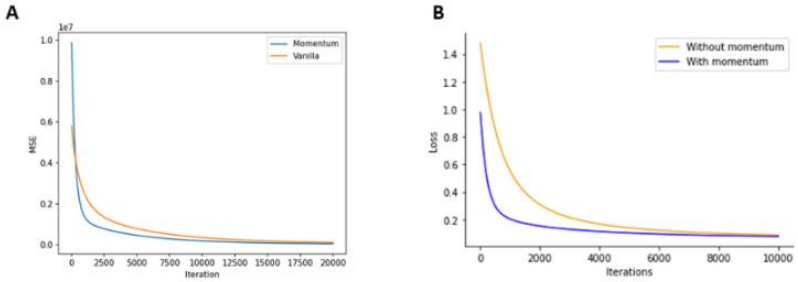

Figure 5. Loss dynamics of vanilla gradient descent and momentum algorithms in the linear (**A**) and logistic (**B**) regression models. Linear regression initial weights were multiplied by 1000 for experimental purposes.

5. Discussion and Conclusion

This project implemented and evaluated logistic and linear regression models using gradient descent and closed-form solutions, when applicable. We observed that normalization and proper regularization can significantly benefit the model by decreasing overfitting and improving generalizability. We also observed that one-hot-encoding categorical data results in a decent performance in the implemented logistic regression model and improved performance for the linear regression model. Tuning model hyperparameters and train/test ratios can also affect model performances. For instance, very large or small learning rates prevent the model from settling into a proper minimum and decrease performance on unobserved data. Although changing batch sizes did not significantly affect test performances, it did display slightly faster convergence in the logistic regression model. Small training data also causes suboptimal performance on the test set. A comparison of LS and gradient descent linear regression models revealed that the solution obtained by the mini-batch gradient descent model could differ from the LS solution. This is not necessarily unfavourable, as the mini-batch model provided slightly better performance on the test set. Finally, the addition of momentum to both regression models resulted in faster convergence in comparison to the vanilla gradient descent algorithm. Future projects could investigate the differences between the convergence speed of various gradient descent algorithms (e.g., RMSProp, Adam, etc.).

Statement of Contributions

Yu Cheng: Logistic regression implementation and experiments. Redaction of report.
Taylor Fergusson: Logistic regression implementation and experiments. Redaction of report.
Mahdi Mahdavi: Linear regression implementation and experiments. Redaction of report.

References

1. Žegklitz, J & Pošík, J. (2017). *Symbolic Regression Algorithms with Built-in Linear Regression*.
2. Yazici, M., Basurra, S., & Gaber, M. (2018). Edge Machine Learning: Enabling Smart Internet of Things Applications. *Big Data and Cognitive Computing*, *2*(3), 26. https://doi.org/10.3390/bdcc2030026
3. Kim, M.-J., & Han, I. (2003). The discovery of experts' decision rules from qualitative bankruptcy data using genetic algorithms. *Expert Systems with Applications*, *25*(4), 637–646. https://doi.org/10.1016/s0957-4174(03)00102-7