MiniProject 1: Getting Started with Machine Learning

Carl Machaalani, Caiya Zhang, Haikun Zhao

October 7th, 2023

Abstract

In this study, we rigorously investigated the performance of linear regression, logistic regression, and mini batch stochastic gradient descent models on two benchmark datasets. The first dataset, derived from the Boston Housing market, consists of 506 entries and 13 key features that shed light on the dynamics of housing prices. Our analysis encompassed various experimental approaches, including different train-test splits, 5-fold cross-validation, and the exploration of mini-batch sizes to gauge model performance. The second dataset, focusing on wine attributes, comprises 178 entries spread across 14 attributes, providing insights into wine classifications. Using logistic regression, we endeavored to predict wine categories based on these attributes. Our findings highlight the significance of meticulous data preprocessing and the impact of distinct modeling strategies. Furthermore, the results underscore the importance of understanding the nuanced interactions between data features and model parameters, emphasizing the role of specific dataset attributes in prediction accuracy.

1 Introduction

This project analyzed two datasets: the Boston Housing market and the wine attributes dataset. For data preprocessing, both datasets were cleaned and statistically analyzed. In terms of model implementation, the Boston Housing dataset utilized Linear Regression, and the wine dataset employed Logistic Regression, with both models leveraging Mini Batch Stochastic Gradient Descent for optimization. Experiments conducted included train-test splits, 5-fold cross-validation, and variations in mini-batch sizes and learning rates. Additionally, the Boston dataset utilized the Gaussian basis function, and the wine dataset's performance was evaluated with growing training subsets.

In the scope of this study, a rigorous emphasis was placed on the critical nature of systematic data preprocessing and the utilization of diverse modeling methodologies. Throughout the analysis, it was observed how specific experimental conditions modulated model responses and the predictive efficacy of particular dataset attributes. The outcomes emanating from the research not only demystify the operational behaviors of the utilized models but also spotlight the salience of specific features within the datasets. This underscores an imperative for adopting a discerning approach towards predictive modeling, whereby particular attention is accorded to the subtle, multifaceted interactions amongst data features and model parameters.

2 Datasets

One of the datasets used for this project is the Boston Housing dataset, sourced from the CMU StatLib library. The Boston Housing dataset, sourced from the CMU StatLib library, encompasses 506 samples, each delineated by 14 housing-related attributes in Boston. Due to ethical concerns, the dataset was refined to 13 attributes by excluding the "B" feature. Preliminary analysis confirmed the data's integrity with no missing or malformed values. Notably, the 'MEDV' attribute, which signifies the median value of homes, has a slight right-skewed distribution. There's a positive correlation between the number of rooms 'RM' and 'MEDV', indicating homes with more rooms typically command higher median values. Conversely, a pronounced negative correlation was observed between the percentage of lower-status population 'LSTAT' and 'MEDV'.

Another dataset is about determining the origin of wines. These data are the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wine, and there includes 178 instances in this dataset om total. The data in this dataset appears to have been cleaned, with no missing values and no obvious malformed values. Moreover, the only target is the class of wines, which were given the class labels 1, 2, and 3, and the number of instances in the three classes does not differ much, being 59, 71, and 48, respectively. Since the dataset is "well behaved" and instances are ordered, there may be a strong correlation between individual features as seen in the Figure 4(c) Appendix.

3 Results

In this work, we implemented linear regression and logistic regression on two datasets respectively, as well as implementing mini-batch SGD for the two models respectively. Furthermore, we conducted experiments on the parameters involved in tuning the models respectively, and try to find out the better parameter configurations by comparing the performance metrics of the experiments.

3.1 Linear Regression

3.1.1 80/20 Train/Test Split

In our initial experiment using an 80/20 train/test split, the Linear Regression model performed consistently on both training and testing sets. Results are shown in Figure 5 in Appendix. The R-squared values indicate that the model captures a majority of the variance in the dataset. The slight discrepancies between training and testing outcomes suggest the model generalizes well to new data, with room for further refinement.

3.1.2 5-Fold Cross-Validation Evaluation

In the 5-fold cross-validation evaluation, the Linear Regression model's Mean Squared Error (MSE) on both training and validation sets was consistent, as visualized in Figure 5 in Appendix. The MSE values were reasonably low and closely aligned with those obtained using the analytical method, highlighting the model's general efficacy and stability across different data subsets.

(a) Train/Test 0.2/0.8 (b) Train/Test 0.5/0.5 (c) Train/Test 0.8/0.2

Figure 1: Performance with different Train/Test splits.

3.1.3 Growing Subsets of the Training Data (20%,30%,...80%)

In our analysis using three train/test splits (0.2/0.8, 0.5/0.5, and 0.8/0.2) for the linear regression model, distinct trends emerged, as depicted in Figure 1. The 0.2/0.8 split showed potential overfitting with its test MSE sharply dropping and training MSE constantly at zero. The 0.5/0.5 split provided a steady decline in test MSE from 40 to 25, with training MSE rising modestly. The 0.8/0.2 split exhibited a gradual test MSE reduction and a similar rise in training error. This suggests that a 0.5/0.5 split yields more consistent test error decline.

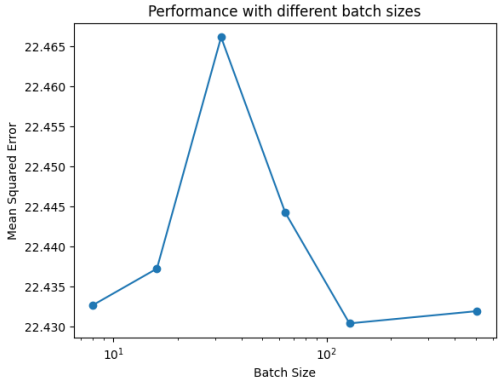


Figure 2: Performance with Different Batch Sizes

Table 1: Performance of linear regression with three different learning rates.

Learning Rate	Mean Squared Error (MSE)
0.001	22.44
0.01	22.43
0.1	22.81

3.1.4 Sample Growing Minibatch size

In this experiment, we experimented the mini-batch SGD algorithm with various batch sizes. Figure 2 shows how the different batch sizes affect the MSE. To increase the accuracy of the test, we decided to run 50 tests per batch size and plot the runs' average MSE. According to Figure 2, the performance gets better on low (8) and high (128) batch sizes and worsens on medium sizes (around 32).

3.1.5 Performance of linear regression with three different learning rates

1. **Sensitivity to Learning Rate:** The performance of the Mini-Batch SGD Linear Regression is sensitive to the choice of the learning rate, but the differences in our experiment are relatively subtle.
2. **Optimal Learning Rate:** A learning rate of 0.01 yields the best performance, with the lowest MSE. This suggests that this rate is most suitable for the dataset and model in terms of convergence and stability.
3. **General Observation:** The learning rate is a crucial hyperparameter. It controls the step size at each iteration while moving towards a minimum of the cost function. A too-large learning rate can cause the algorithm to converge too quickly to a suboptimal solution, or even diverge. Conversely, a too-small learning rate can cause the algorithm to converge very slowly or get stuck in a suboptimal solution.

3.1.6 Performance metric selection

Metric Selection Rationale:

For this task, the Mean Squared Error (MSE) was chosen as the performance metric. MSE measures the average squared difference between the actual and predicted values, making it a common metric for regression problems. The rationale for selecting MSE includes:

1. **Interpretability:** MSE values can be easily understood and interpreted since they are in the same unit as the squared target variable.
2. **Emphasis on Larger Errors:** MSE penalizes larger errors more heavily than smaller ones due to the squared term, which can be beneficial in many practical scenarios where larger prediction errors might have more severe consequences.
3. **Differentiability:** The squared term makes MSE differentiable, which is essential for optimization algorithms like gradient descent.

Optimal Configuration Results:

Through our experiments, we found that the optimal parameter configuration for the dataset is:

$$\text{Learning Rate: } 0.005, \quad \text{Batch Size: } 64$$

with a corresponding MSE of approximately 22.43. This configuration offers a balance between the speed of convergence (learning rate) and the stability of the gradient estimates (batch size), leading to the best model performance on the dataset.

3.1.7 Gaussian Basis Functions

- Mean Squared Error (MSE) using the original feature set: 26.4816.
- MSE using the combined feature set (original + Gaussian basis functions): 26.5115.

The enriched feature set, which includes the Gaussian basis functions, provides a slightly better performance (lower MSE) than the original feature set. This suggests that the Gaussian basis functions capture some non-linear patterns in the data, enhancing the model's capability to fit the underlying data distribution. While the improvement in MSE is subtle, it demonstrates the potential benefits of feature engineering and the capability of basis functions to capture complex relationships in the data.

3.1.8 Comparison between analytical with mini-batch SGD based linear regression

- Analytical Linear Regression: MSE = 22.4297
- Mini-batch Gradient Descent (GD): MSE = 22.4321

1. The performance of both methods is very close, with only a subtle difference in MSE. This indicates that both methods have effectively captured the underlying patterns in the dataset.

2. The analytical solution, being an exact method, offers a slightly better performance. It directly minimizes the cost function and provides the optimal coefficients for the linear regression model.
3. Mini-batch sGD, although an iterative and approximate method, still achieves a performance close to the analytical solution. This highlights the effectiveness of mini-batch SGD, especially when appropriately tuned.
4. The tiny difference in performance underscores the trade-offs between the two methods. While the analytical method offers exactness and is suitable for smaller datasets, mini-batch SGD offers scalability and adaptability, especially for larger datasets or when online learning is needed.

3.2 Logistic Regression

We then conducted experiments on the wine dataset and generated performance metrics including accuracy, precision, recall, and F1-score.

3.2.1 80/20 Train/Test Split

We first performed an 80/20 train/test split and ran experiments on both regular logistic regression model and its mini-batch SGD one, as shown in Table 3 in Appendix. For the regular logistic regression model, we found that its performance on the test dataset exceeded its performance on the training data, so we considered overfitting, and we will also explore the specific causes and ways to resist overfitting in subsequent subtasks.

3.2.2 5-fold cross-validation

We followed this with a 5-fold cross-validation strategy on logistic regression. With this, the results showed that the model's performance on the training set was improved, with the training precision exceeding 0.77 and the corresponding test precision reaching 0.75. As shown in Table 2, while the test performance was still overall slightly higher than the training performance, such as accuracy and recall, the gap had narrowed down conspicuously.

Table 2: Performance in Logistic Regression with 5-fold Cross-Validation.

	Train Performance				**Test Performance**			
Fold	**Accuracy**	**Precision**	**Recall**	**F1-Score**	**Accuracy**	**Precision**	**Recall**	**F1-Score**
1	0.7762	0.4846	0.5862	0.5190	0.7715	0.4928	0.6667	0.5490
2	0.7949	0.7582	0.7097	0.6783	0.7333	0.6346	0.6238	0.6005
3	0.7482	0.7478	0.6753	0.6025	0.6762	0.4141	0.5714	0.4333
4	0.7762	0.4737	0.5957	0.5197	0.7714	0.5128	0.6333	0.5491
5	0.7249	0.7726	0.6436	0.5485	0.7524	0.7557	0.6733	0.6014

3.2.3 Sample growing subsets of the training data

In this experiment, we first split the original dataset in the ratio of train/test 80/20, and then sampled $20\% - 90\%$ for the split training set, increasing 5% one by one, for fitting and prediction. The results are shown in Fig. 3(a), where the accuracy of training and testing reached more than 0.70 in general, but after sampling more than 45%, there is a significant decline, which may be due to the lack of specific data preprocessing means or affected by other parameters, such as the learning rate. For this, we will verify in the following subtask.

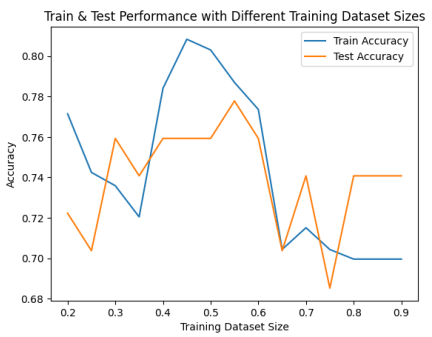
3.2.4 Sample growing mini-batch size

In this round of experiments, we focus on the batch sizes involved in each training in the mini-batch SGD algorithm, which are trained and tested according to the mini-batch sizes of 8, 16, 32, 64, and 128, respectively. The results are shown in Fig. 3(b), where increasing the mini-batch size helps in the stability of convergence within a certain range, e.g., from 8 to 32, but the performance of the model decreases as the mini-batch size gets larger and larger. In particular, the model performs best when the mini-batch size is 32, achieving a test accuracy of more than 0.77 and a test F1-score of 0.65 (along with test precision = 0.76 and test recall = 0.65).

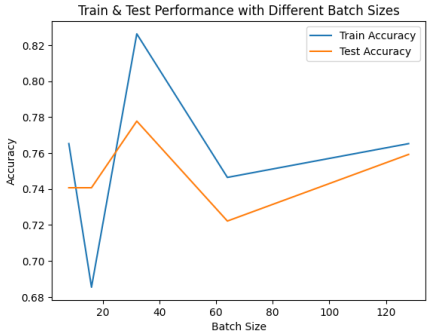
3.2.5 Sample different learning rate

In addition to the size of the sample size involved in training, the setting of the learning rate size is also crucial. The learning rate controls the learning progress of the network model and determines whether the network can succeed or how long it will take to successfully find the global minimum and thus obtain the global optimal

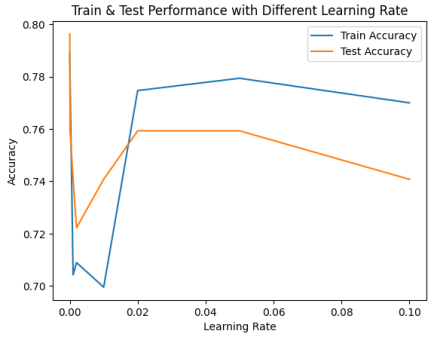
solution. In this experiment, we chose 8 common learning rate values from 0.1 to 0.00001, and found that the larger the learning rate, the more difficult it may be for the model to learn effective weights, and the worse the test performance becomes. However, smaller learning rates, while showing better results, may simultaneously lead to slower convergence and thus loss of performance.



(a) Performance with Different Training Dataset Sizes



(b) Performance with Different Batch Sizes



(c) Performance with Different Learning Rate

Figure 3: Experiments on different super-parameters

3.2.6 Potential optimal parameter choice

Considering that the mini-batch SGD performed better than the regular logistic regression's model in the very first fitting and testing, we chose the mini-batch SGD at the end and chose the batch size (=32) and the learning rate (=0.001) that had performed better in the first few rounds of subtasks. The model ended up with a training accuracy of 0.7793 and a testing accuracy of 0.7778, which is very close, and other metrics were improved overall.

4 Discussion and Conclusion

The main objective of this study is to explore the widely used linear regression and logistic regression models, along with their Mini-Batch SGD methods. In the exploration process of machine learning, it is necessary to have an insight on and do cleaning of the data, and then we build and select proper models and develop effective strategies to explore the optimal parameter configurations. For both linear and logistic regression, adequate training data and smaller learning rates seem promising, but the quality of the data and time consumption should also be considered. For both Mini-Batch SGD methods, the choice of batch size also reveals a trade-off between convergence speed and model accuracy. Also, the use of Gaussian basis functions on the Horse dataset demonstrates the potential of feature engineering.

For future learning and exploration, exploring automated methods for hyper-parameter optimization, such as grid or stochastic search, can be a promising direction. Considering more high-quality external data to improve the predictive power of both models is also a common tactic. And, as the complexity of the models increases, the study of their interpretability will become more and more important, and we call for more researchers to analyze and decipher the kernel of the black box.

5 Statement of Contributions

- Carl was responsible for tasks related to the boston dataset, particularly in acquiring, preprocessing, and analyzing the dataset, implementing the linear regression models, and running experiments 1 through 4.
- Caiya is primarily responsible for all tasks related to the wine dataset and related report writing.
- Haikun improved model structure and addressed questions 5-8 on the Boston dataset, and he made necessary effort to part of the report writing.

References

[1] https://www.kaggle.com/datasets/fedesoriano/the-boston-houseprice-data?select=boston.csv
[2] https://archive.ics.uci.edu/dataset/109/wine

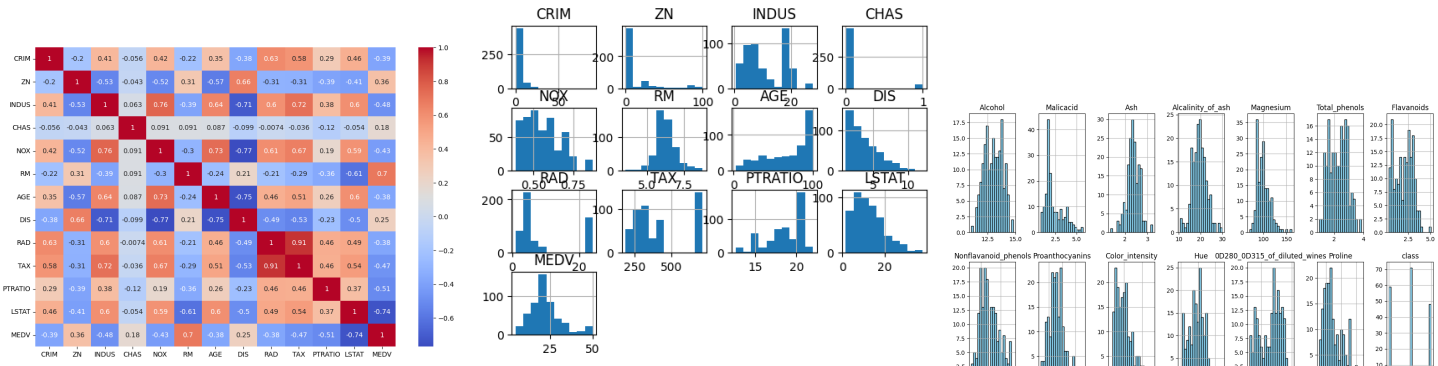
Appendix

A. Additional Table

Table 3: Performance in Logistic Regression with 80/80 Train/Test Split.

Logistic Regression				
	Accuracy	**Precision**	**Recall**	**F1 Score**
on training set	0.6995	0.6776	0.6020	0.5164
on testing set	0.7407	0.7170	0.6179	0.5464
Mini-Batch SGD				
	Accuracy	**Precision**	**Recall**	**F1 Score**
on train dataset	0.7559	0.4225	0.5817	0.4888
on test dataset	0.7222	0.4028	0.5385	0.4476

B. Additional Graphs



(a) Correlation Matrix of Features of Horse Dataset (b) Histogram of Features' Distributions of Horse Dataset (c) Basic Visualization of Wine Dataset

Figure 4: Basic Visualization of Two Datasets

	Metric	Training	Testing
0	Mean Squared Error	22.408805	23.065081
1	Root Mean Squared Error	4.733794	4.802612
2	Mean Absolute Error	3.425603	3.260658
3	R-squared value	0.744638	0.676061

(a) 80/20 Train/Test Split Results

	Metric	Training Average	Testing Average
0	Mean Squared Error	22.288877	23.816873
1	Root Mean Squared Error	4.720871	4.877350
2	Mean Absolute Error	3.349992	3.445955
3	R-squared value	0.736017	0.712087

(b) 5-Fold Cross Validation Results

Figure 5: Experiments Results with 80/20 Train/Test Split and 5-fold Cross-Validation