COMP 551 - MiniProject 1: Getting Started with Machine Learning, Fall 2023

Johan Trippitelli - 260917958, Minh Anh Trinh - 260853143, Chris Chong - 260976714

October 5, 2023

Abstract: The following mini project was conducted as part of the COMP 551 class at McGill university. It focuses on building a foundation in machine learning through the implementation and analysis of two fundamental models: Linear Regression and Logistic Regression with Gradient Descent. The project has three tasks: data acquisition and preprocessing, model implementation, and conducting experiments.

In the first task, two datasets are employed: the Boston Housing dataset and the Wine dataset. Data is loaded into Python using the Pandas library, cleaned, and utilized for the subsequent tasks.

The second task is implementing the two machine learning models. The models are built from scratch using the knowledge obtained from COMP 551 lectures.

The third task is to perform experiments on the given data sets in order to fully train the Machine Learning Algorithm to predict a good model for the datasets. For this a 80/20 train/test split was employed. Performance metrics such as Mean Squared Error, accuracy, precision, recall, and F1-score were employed to evaluate the effects of varying hyperparameters such as training data size and minibatch size.

Introduction: The primary objective is to train and test Linear regression and Logistic regression with gradient descent to analyze a boston housing dataset and a wine dataset, while varying the hyperparameters and observing the effects on performance. These hyperparameters include size of training data, minibatch sizes, and learning rates. The optimal parameters for linear regression were a learning rate of 1e-8, an epsilon of 1e-6, and a max iteration of 1000. In contrast, for logistic regression, both a learning rate of 0.5, epsilon of 1e-6, and max iteration of 10,000, as well as a learning rate of 0.5, epsilon of 0.001, and max iteration of 1,000 achieved comparable peak performance metrics. Among other findings, it is shown that increasing size of training data improved performance overall, more so on the train set rather than the test sets.

In order to get a proper sense of how different algorithms performed on the data, Stochastic Gradient Descent using the mini batch method was performed on both the housing and wine data sets. From these implementations, it was discovered that with larger mini batches convergence speed improved immensely. This improvement in speed was met with a decline in prediction accuracy (demonstrated by higher MSE values). The MSE trend was more apparent in linear regression than in logistic regression. When comparing the mini batch algorithm to the analytical solution, however, the performance was superior for both speed and accuracy when using the analytical algorithm. More in depth discussions on these findings can be found in the Discussion and Conclusion section of this report.

Datasets: In order to get the best results when using given data sets, a necessary step is to always clean the data. A data set is said to be unclean if it includes rows or columns of missing data values. Such missing data could skew the results of the Machine Learning Algorithm leading to improper model estimation. The Pandas library is very useful for cleaning data sets.

The implemented code takes the data set (df) and parses to see if there are any missing values. If it encounters any missing values then the row is immediately deleted. Once all the rows with missing values are removed, the cleaned data structure is stored in df-cleaned. In the instance of the Boston Housing data, it was observed that no missing values were present. Therefore the original data set was used.

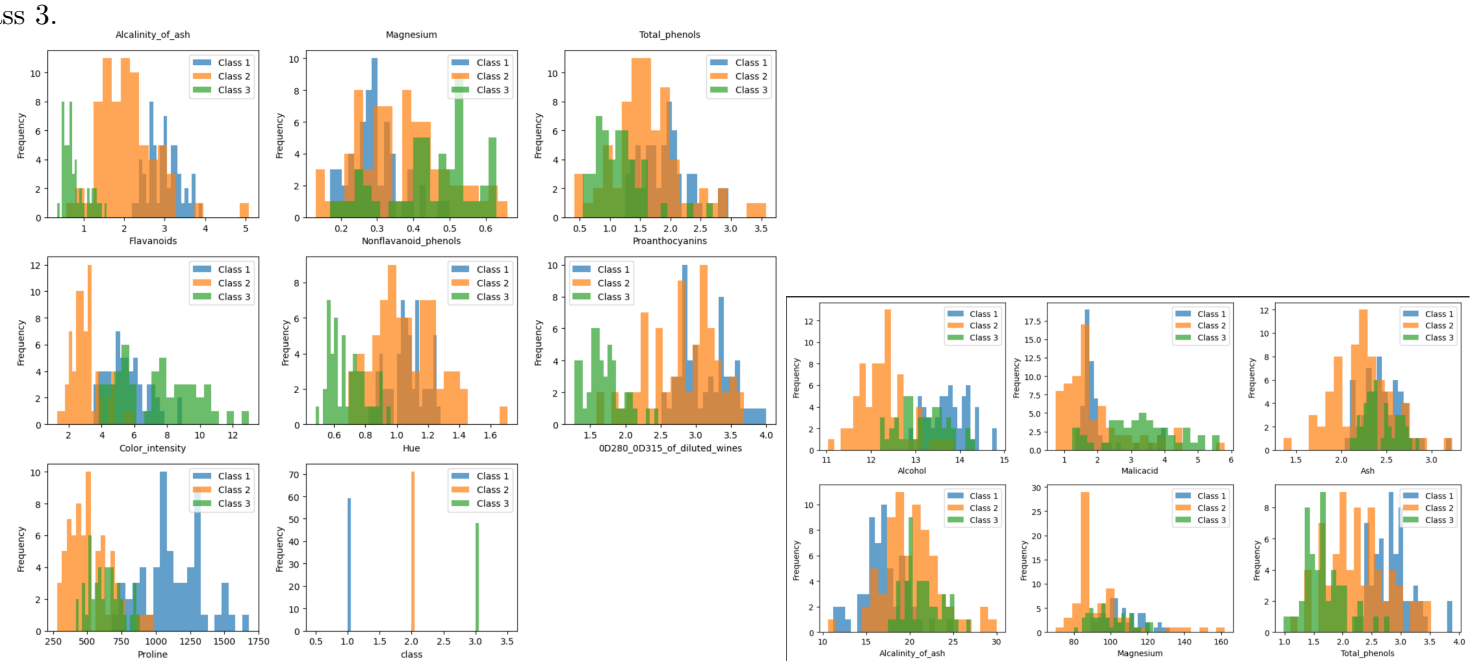
The boston housing dataset contains 506 cases with 14 attributes per case from the U.S Census Service regarding housing in Boston, Massachusetts.[2] These attributes include statistics such as the crime rate per capita by town, the pupil to teacher ratio by town, and the number of rooms per dwelling on average. [2]

Due to ethical concerns, one attribute 'B' has been removed, so the boston housing dataset now has 13 attributes. This attribute describes the proportion of black people per town, and is an ethical concern, as any

model trained on the full data may be seen as encoded with racism, and attribute certain house prices to levels of 'B'. [3]

By plotting the density distribution of the 'MEDV' column at figure 1a, it is shown that the majority of the houses have a median value of $20,000. This displays the fact that this data comes from a study done in 1978 by Harrison, D. and Rubinfeld, D.L [1]. This data is reflective of the views and data available at the time, as shown by the 'B' column and the low median values of houses when compared to today's standards.

The wine dataset contains 178 instances with 13 attributes per case. This dataset is the result of chemical analysis of wines derived from three different Italian cultivators. It depicts the quantities of 13 distinct components found in each of the three classes of wine. As shown in the graphs shown below, each distinct class of wine has unique features which will help the logistic regression model distinguish between the wines. For example, in alcohol content, class 2 tends to have the least amount, class 1 has the most amount, and class 3 falls in between. In the class graph, it is shown that in the dataset, the count of class 2 wine is the highest, followed by class 1 and finally class 3.



For ethical concerns, the intentions and motivations of the authors of the datasets should be considered. In the case of the boston dataset, racial prejudices and assumptions held by the authors should be scrutinized when utilizing the dataset. In the case of the wine dataset, it should be investigated whether the authors were commissioned or financially incentivized by the cultivators to examine their wines and what the motivation behind choosing those three specific cultivators was when organizing the dataset.

Results: So results are reproducible, the number used in the np.random.seed() command has been noted in the captions for the images themselves when relevant. In experiment 1, to establish a baseline of performance for the models, the following results were found shown in figure 1b. Clearly the mean square error for the training set is lower than the mean square error for the test set, which is to be expected as the test set is data the model hasn't yet seen before. For logistic regression, performance on training set is higher across metrics than on the testing set. The high accuracy performance of 93% on the training set can be attributed to the max_iters of the model being set to 1e5.

In experiment 2, for both the linear regression model and the logistic regression model, each of the five folds in the five-fold cross-validation were used as the validation set, then evaluated on the training set and the testing set, the results of which are shown at figure 1d to figure 1i

In result of linear regression model at figure 1d, besides one outlier (fold 1), the training set generally had a lower MSE than the testing set, which is especially evident when viewing fold 4, which had an MSE of 77.1 for the testing set and an MSE of 14.2 for the training set

In the results for the logistic model, accuracy shown at figures 1e - 1i was found to be higher on the training set when compared to the testing set in every fold, but there were some instances where precision, recall, and f1-score would be higher on the testing set rather than the training set. Precision, recall, and f1 score were calculated by

taking the weighted average of each metric by class size. The model was run with max_iters=1e5. There were some outliers such as the recall on the training set from fold 5 at figure 1i being 0.4, or the precision being 1.0 in fold 5. In general, performance was stronger on the training set when compared to the testing set. This trend held on most sets except for fold 4 at figure 1h, where accuracy was 97% on the testing model and 93% on the training model. The results show that these performance metrics are incredibly variable and depend on which folds were used to train and which folds were used to validate.

In experiment 3, each of the models have graphs indicating their respective performance metrics as a function of training size. In the result of the linear model at figure 2a, it is shown that as training data size increases, the MSE for both the test set and the training set has a downward trend.

For the logistic regression model at figures 2b to 2e, the metrics accuracy, precision, recall, and f1-score also show a similar trend. Performance metrics increase from the initial value at 20% of the training data, to a higher score at 80% of the training data, with some spikes at 0.3, 0.5, and 0.7 of the training data.

The results from Fig. 4a demonstrates an inversely proportional relationship between Batch Size and convergence time. Specifically when the size of the mini Batches taken from the training data was increased the time of convergence was faster. There were some exceptions for the logistic regression model (batch size = 2**1) but generally the relationship holds. The results from Fig. 4b portrays a somewhat proportional relationship between log2(batch Size) and MSE for the linear regression model. The logistic regression however portrays randomized results where MSE does not seem to be related to batch size whatsoever. Fig. 4c shows the best batch size for each variable. Fastest convergence is 256 for both models. For linear the MSE is lowest for batch size 4 and for logistic regression the lowest MSE is for batch size 1.

In Experiment 5, we tested our Linear Regression and Logistic Regression models with different learning rates. For Linear Regression, we examined four different learning rates: 1e-12, 1e-10, 1e-8, and 1e-6, while keeping the epsilon and max iteration values fixed at 1e-5 and 1000, respectively. Among these learning rates, the best performance was achieved with a learning rate of 1e-8, resulting in the lowest Mean Squared Error (MSE) of 122.87 (see Figure 5a).

Regarding Logistic Regression, we tested three different learning rates: 0.00001, 0.001, and 0.05, with fixed epsilon and max iterations set at 1e-5 and 1e5, respectively. Among these options, a learning rate of 0.001 delivered the most favorable results, yielding accuracy, precision, recall, and F1 score values exceeding 94% in each category (see Figure 5b).

In Experiment 6, both the linear and logistic regression models were assessed across a wide array of parameter combinations, encompassing learning rates, epsilon values, and maximum iteration settings. Specifically, each parameter was tested with three different values, resulting in a total of 27 distinct combinations.

For linear regression, the optimal outcome was achieved when employing a learning rate of 1e-8, an epsilon value of 1e-6, and a maximum iteration setting of 1000. This configuration yielded the lowest Mean Squared Error (MSE) of 121.9264 (as depicted in 6a).

On the other hand, the logistic regression model demonstrated its best performance with a learning rate of 0.5, an epsilon value of 1e-6, and a maximum iteration setting of 10,000, as illustrated in 6b. This combination consistently achieved accuracy, precision, recall, and F1 score metrics of at least 97%. Another combination also yielded similar results with a learning rate of 0.5, an epsilon value of 0.001, and a maximum iteration of 1,000.

For a pictorial reference observe 7a and 7b. Using Gaussian Basis allows our model to fit non-linear relationships while still using linear regression. In this case, five gaussian bases were used in an attempt to find a good fit to the boston housing data. The amount of gaussian basis functions used was limited to five in order to limit the possibility of overfitting. From the resulting predictions found the Gaussian model had an MSE of 58.309 on the training data and an MSE of 71.975 for the testing data. Although the model predicted the training data more accurately than the testing data this difference isn't statistically significant and therefore indicating overfitting is kept relatively low when using five gaussian bases. The model was able to make good predictions, however these predictions were not more accurate than the standard analytical solution to linear regression. This would lead to the conclusion that the data can be fitted more accurately with a simple linear regression.

When testing the analytical model of linear regression against the mini batch model, the results in Fig.8a occurred. The testing was performed once for the analytical solution and twice for the mini Batch solution. For the mini Batch, sizes of 8 and 256 were used to comprehend the advantages of small and large batches against

the analytical solution. The analytical solution found the functions global minima leading to a low MSE value of 27.13. The size = 8 mini batch had a larger MSE 159.19 and the size = 256 mini batch had less accurate predictions leading to a MSE 283.29. These results are to be expected. The larger the batches, the less accurate each gradient step we take therefore our final solution is less accurate. For convergence time, the Analytical solution took 0.0007 seconds to fit a model while the mini batches took 0.03 and 0.01 seconds for size 8 and 256 respectively. When the size of the mini batch increased the convergence time was faster as expected, but surprisingly the analytical solution was faster by a factor of 100.

Discussion and Conclusion: Experiment 1 establishes what performance can be expected from each model. Experiment 2 shows that performance results across both models and vary greatly depending on which fold is used for validation. Results for experiment 1 and 2 show that performance on training data is usually higher than performance on testing data, and experiment 3 shows that using a larger subset of training data increases performance across both models on both training and testing data. In both experiment 5 and 6, we observed notable variations in results and accuracy based on parameter choices. These findings underscore the significance of parameter tuning in optimizing the performance of linear and logistic regression models for our specific dataset. For future investigations, a wider range of minibatch sizes, or different nonlinear basis functions like sigmoid bases or polynomial bases can be tested.

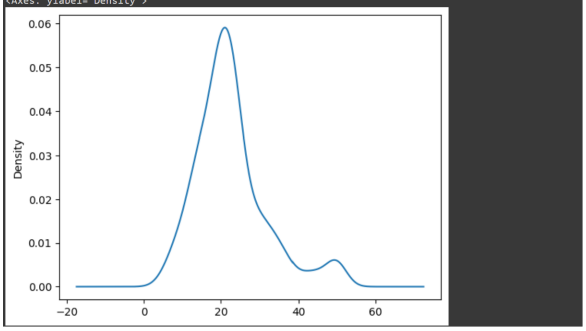
For experiment 8, although both mini batch solutions were outperformed in accuracy and efficiency this method is still useful in practical cases. The mini batch method is commonly used when datasets are too large to store in memory or when constant updating of weights is necessary due to new data points. At times, the analytical solution is not available. In these cases, we see from our testing that mini-Batch SGD is a good approximation for fitting a model to our data sets.

Statement of Contributions: Chris did analytical linear regression, descriptions and ethical analyses of the datasets, and experiments 1, 2, and 3. Minh Anh implemented logistic regression and did experiments 5 and 6. Johan Trippitelli was responsible for the implementation of the Mini-Batch for Linear and Logistic models, the Gaussian bases modeling, and expirements 4, 7, 8.

References and Appendix:

Figure 1:

```
df['MEDV'].plot(kind='kde', xlabel='price') # plots distribution of values from the MEDV column

It is True that the data is clean
<Axes: ylabel='Density'>
```



(a) density function of 'MEDV' for boston dataset

```
Performance on Test Set:
Linear Regression - Mean Square Error:  40.56145020875674

Performance on Training Set
Linear Regression - Mean Square Error:  18.535826568587463
```

(b) Experiment 1: MSE for Linear Regression

```
Performance on Testing Set:
Accuracy: 0.8571428571428571
Precision: 0.9120879120879121
Recall: 0.8571428571428571
F1-score: 0.8633271490414346

Performance on Training Set:
Accuracy: 0.9300699300699301
Precision: 0.9306053025565222
Recall: 0.9300699300699302
F1-score: 0.9295001295001295
```

(c) Experiment 1: Performance for Logistic Regression

```
Testing Set: validated using Fold 1 :  12.648895273886541
Training Set: validated using Fold 1 :  25.179937520500303

Testing Set: validated using Fold 2 :  26.201555743989637
Training Set: validated using Fold 2 :  22.697768769047922

Testing Set: validated using Fold 3 :  35.12882654924537
Training Set: validated using Fold 3 :  21.561131941500754

Testing Set: validated using Fold 4 :  77.09985345257094
Training Set: validated using Fold 4 :  14.152057056423034

Testing Set: validated using Fold 5 :  29.833469495109576
Training Set: validated using Fold 5 :  22.78359745817192
```

(d) Experiment 2: MSE for all 5 folds of Linear Regression

```
On Testing Set
Validated using Fold 1 :
Accuracy: 0.8857142857142857
Precision: 1.0
Recall: 0.8857142857142857
F1-score: 0.9393939393939393

On Training Set
Validated using Fold 1 :
Accuracy: 0.965034965034965
Precision: 0.9658649223866616
Recall: 0.9650349650349651
F1-score: 0.9650247038361491
```

(e) Experiment 2: Fold 1 for Logistic Regression

```
On Testing Set
Validated using Fold 2 :
Accuracy: 0.7714285714285715
Precision: 0.9208791208791208
Recall: 0.7714285714285715
F1-score: 0.7984761904761903

On Training Set
Validated using Fold 2 :
Accuracy: 0.916083916083916
Precision: 0.9224518331493488
Recall: 0.916083916083916
F1-score: 0.9153428856399153
```

(f) Experiment 2: Fold 2 for Logistic Regression

```
On Testing Set
Validated using Fold 3 :
Accuracy: 0.8857142857142857
Precision: 1.0
Recall: 0.8857142857142857
F1-score: 0.9393939393939393

On Training Set
Validated using Fold 3 :
Accuracy: 0.958041958041958
Precision: 0.9640359640359641
Recall: 0.9580419580419581
F1-score: 0.9590162041708257
```

(g) Experiment 2: Fold 3 for Logistic Regression

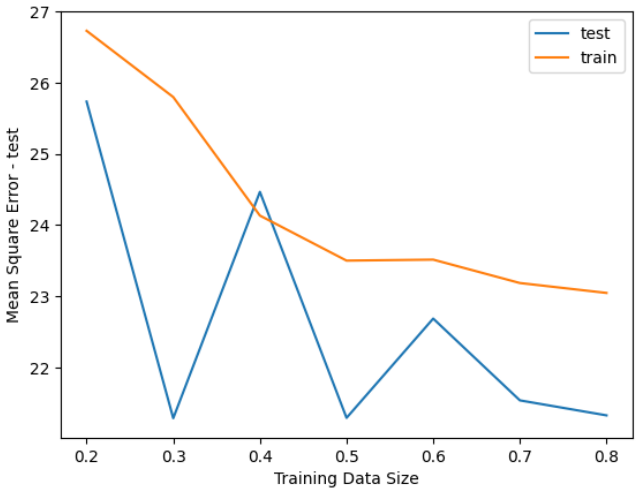
```
On Testing Set
Validated using Fold 4 :
Accuracy: 0.9714285714285714
Precision: 0.9725274725274725
Recall: 0.9714285714285714
F1-score: 0.9709568037741412

On Training Set
Validated using Fold 4 :
Accuracy: 0.9300699300699301
Precision: 0.9302981339740528
Recall: 0.93006993006993
F1-score: 0.9296160242394804
```

(h) Experiment 2: Fold 4 for Logistic Regression

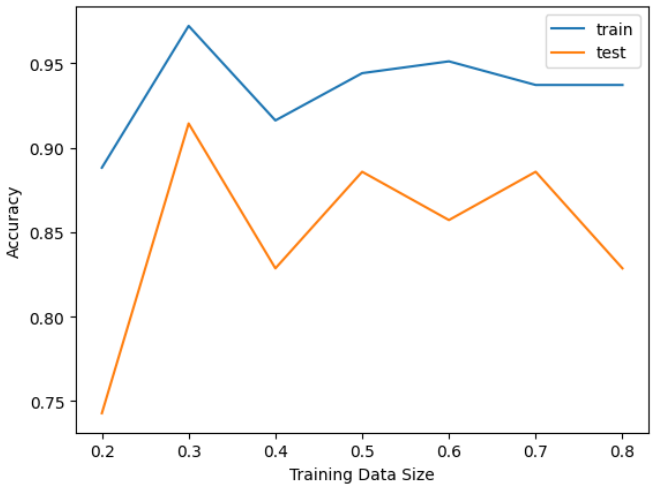
```
On Testing Set
Validated using Fold 5 :
Accuracy: 0.6285714285714286
Precision: 1.0
Recall: 0.6285714285714286
F1-score: 0.7719298245614035

On Training Set
Validated using Fold 5 :
Accuracy: 0.8642857142857143
Precision: 0.8929365079365079
Recall: 0.8642857142857142
F1-score: 0.8426881204831514
```

(i) Experiment 2: Fold 5 for Logistic Regression

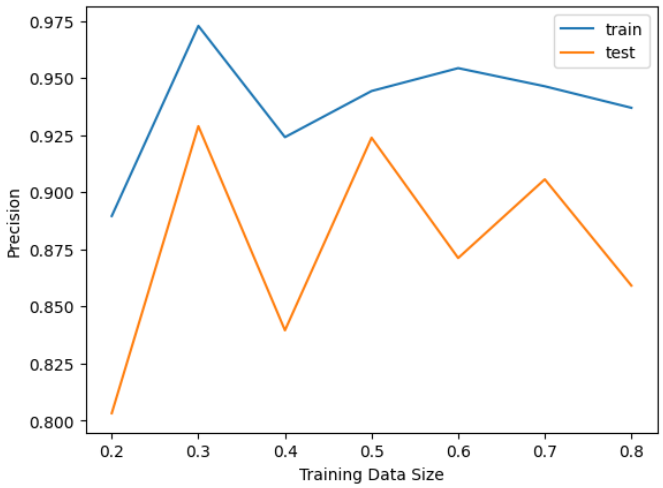
Figure 2:



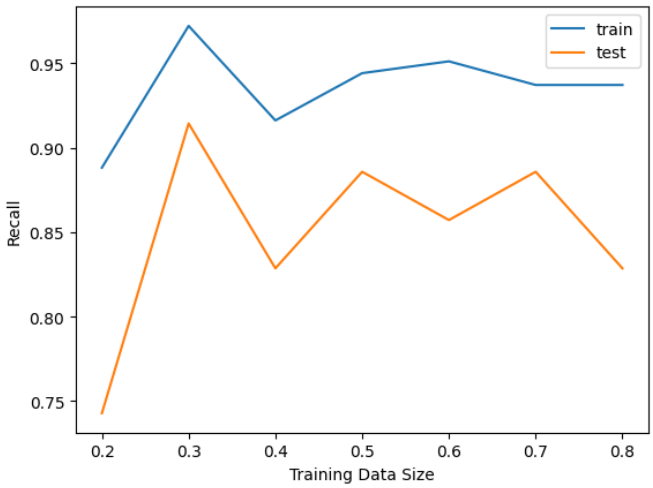
(a) Experiment 3: Graph for accuracy in train and test for Linear Regression (seed:65) (max_iters=1e5)



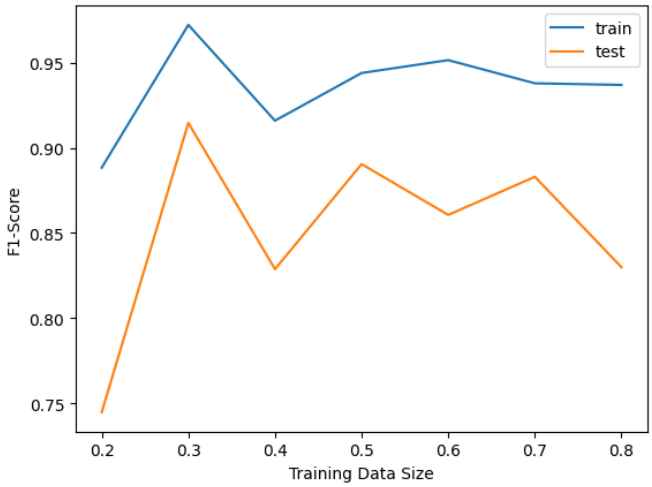
(b) Experiment 3: Graph for accuracy in train and test for Linear Regression (seed:65) (max_iters=1e5)



(c) Experiment 3: Graph for precision in train and test for Linear Regression (seed:65) (max_iters=1e5)



(d) Experiment 3: Graph for recall in train and test for Linear Regression (seed:65) (max_iters=1e5)



(e) Experiment 3: Graph for f1-score in train and test for Linear Regression (seed:65) (max_iters=1e5)

Figure 3:

(a) Experiment 4: Graph of batch Size vs Convergence time for linear and logistic SGD

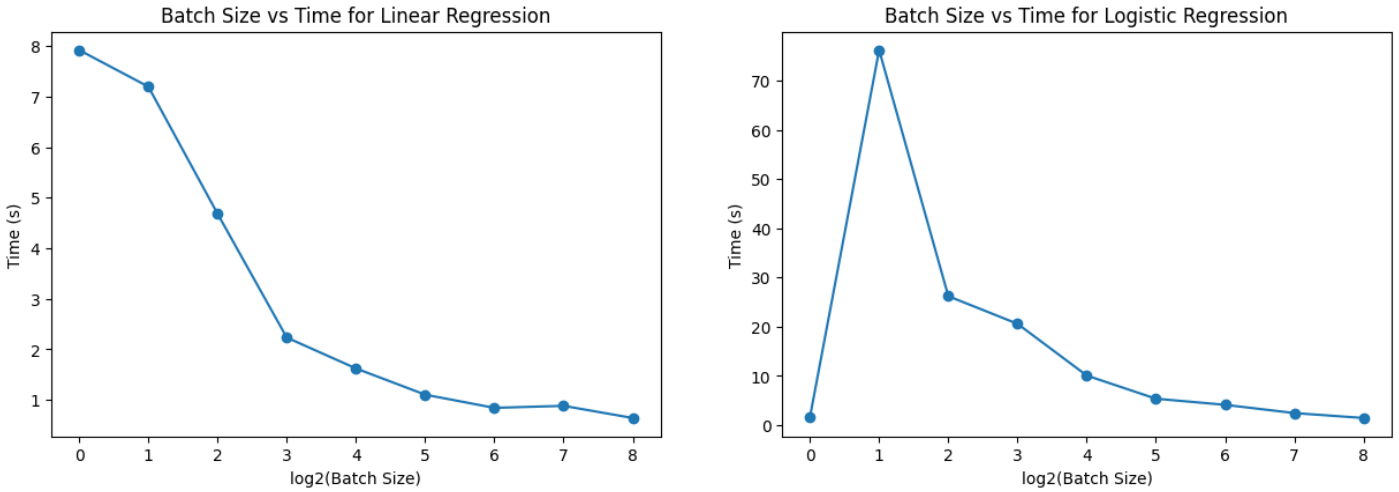
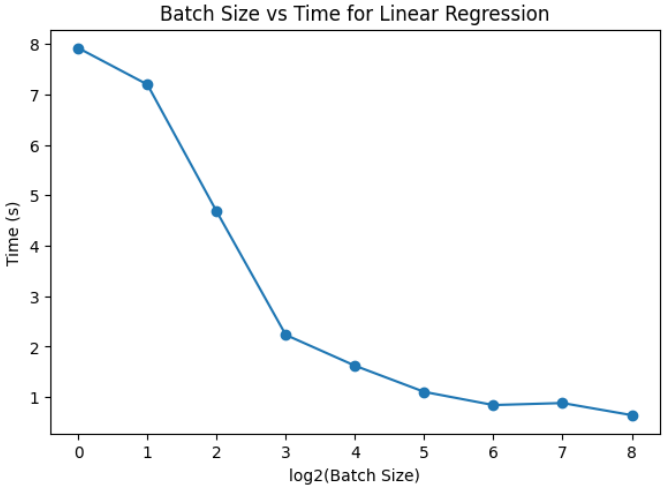
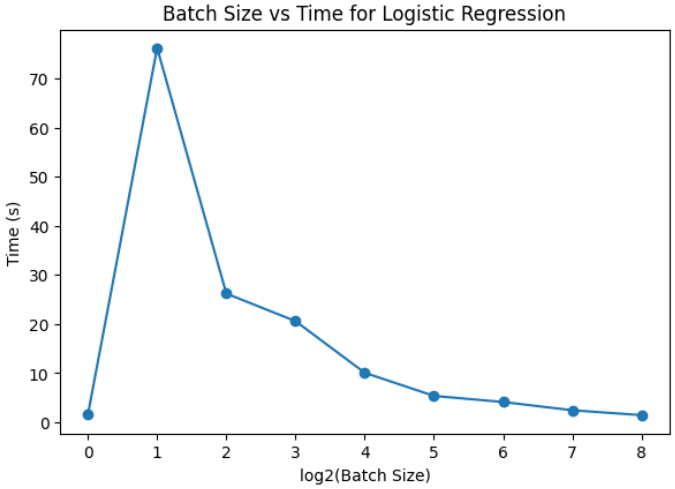
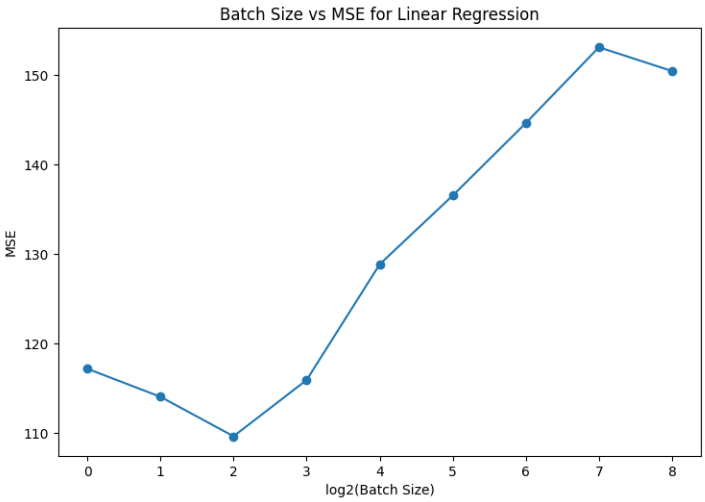


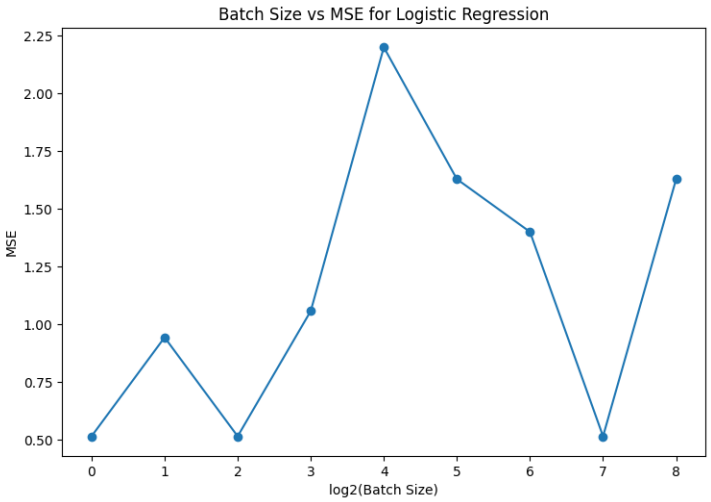
Figure 4:





(a) Experiment 4: Graph of batch Size vs Convergence time for linear and logistic SGD



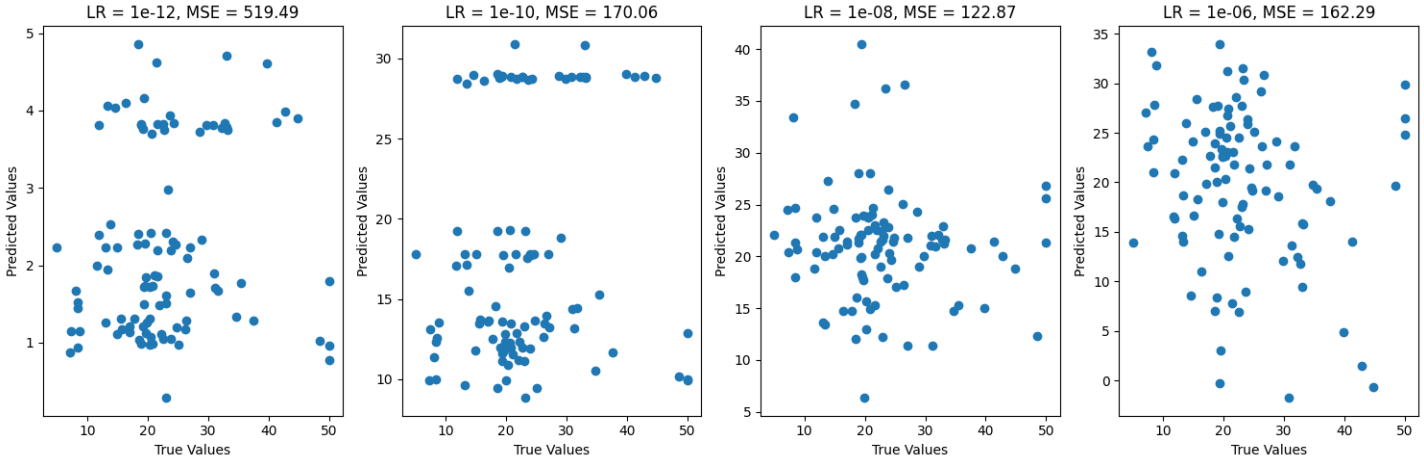


(b) Experiment 4: Graph of batch Size vs MSE for linear and logistic SGD

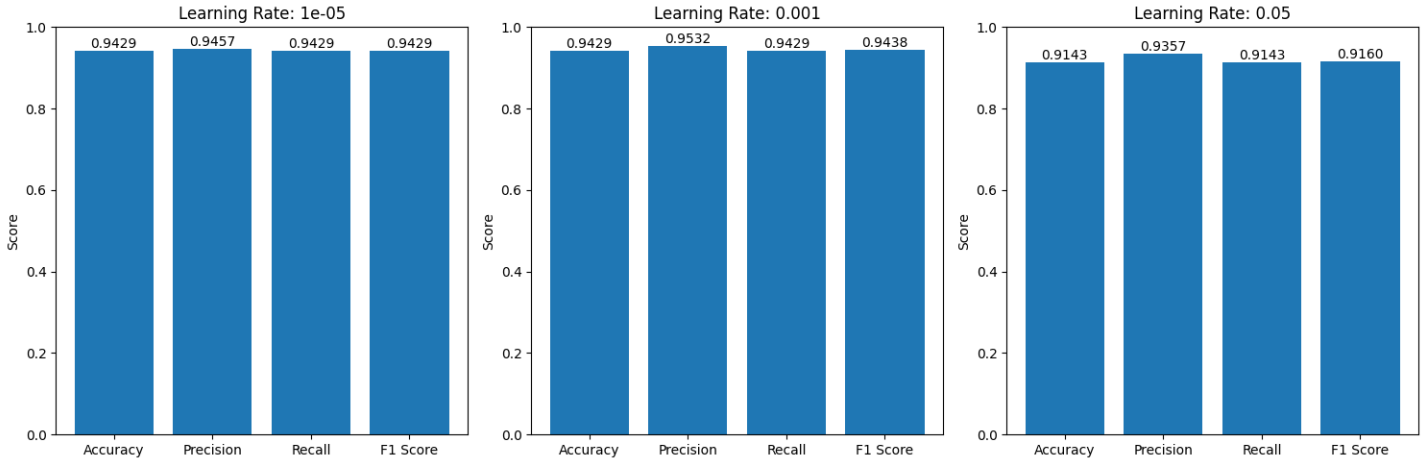
```
Testing MiniBatch on Linear Regression:
Lowest MSE: 109.64374939977478
Corresponding Batch Size: 4


Fastest convergence time: 0.6429619789123535
Corresponding Batch Size 256


Testing MiniBatch on Logistic Regression
LowestMSE 0.5142857142857142
Corresponding Batch Size 1


Fastest convergence time 1.4427342414855957
Corresponding Batch size 256
```

(c) Fastest times and lowest MSE for each Mini Batch model

Figure 5:

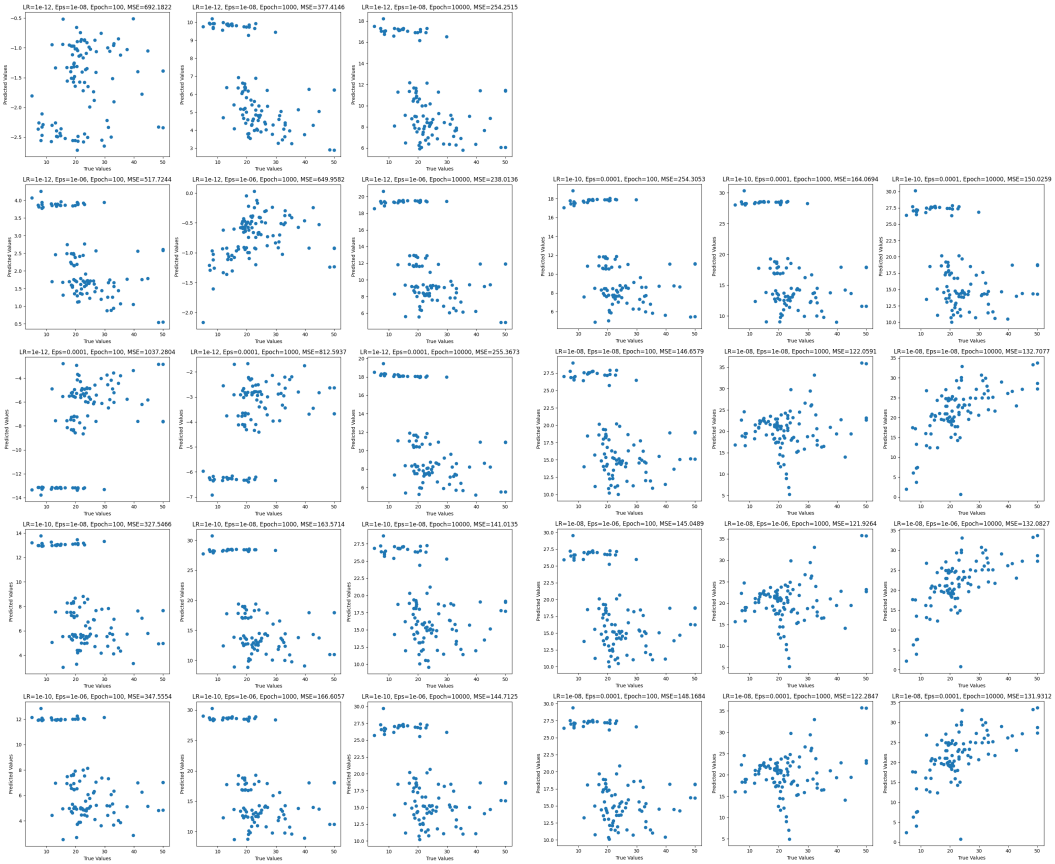


(a) Experiment 5: Graph for true values and predicted values across different learning rates for Linear Regression

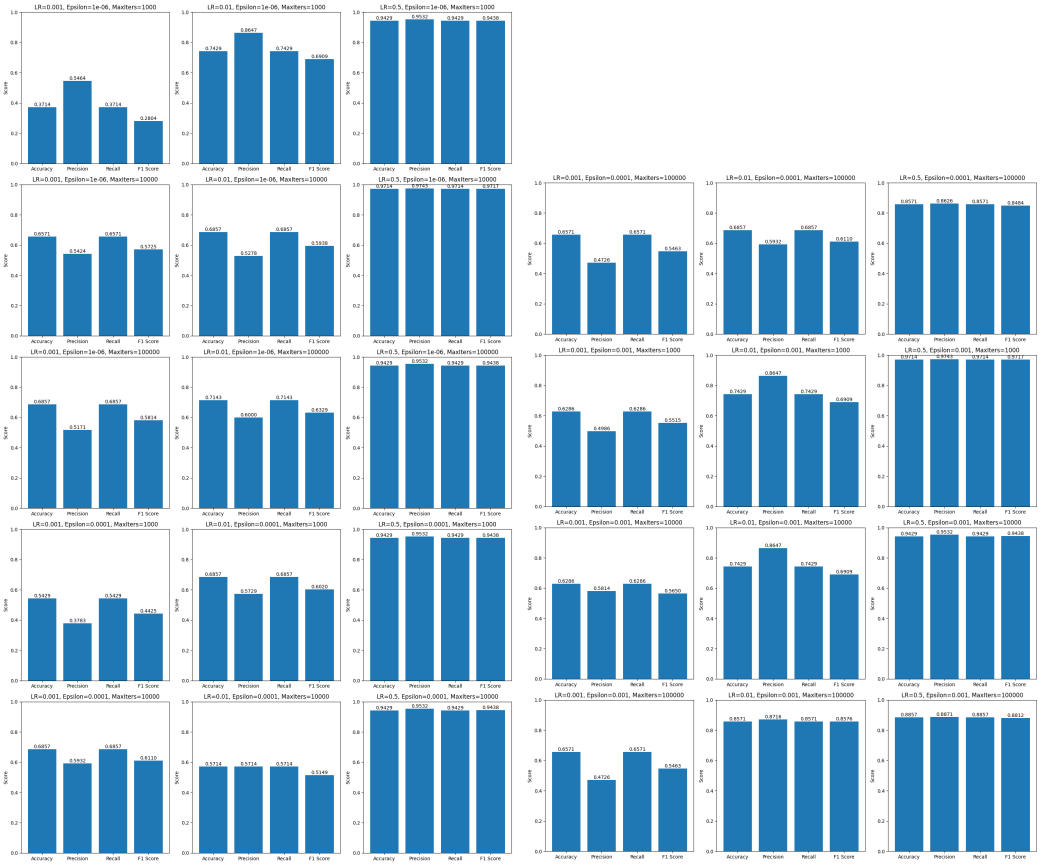


(b) Experiment 5: Graph for evaluation metrics across different learning rates for Logistic Regression

Figure 6:

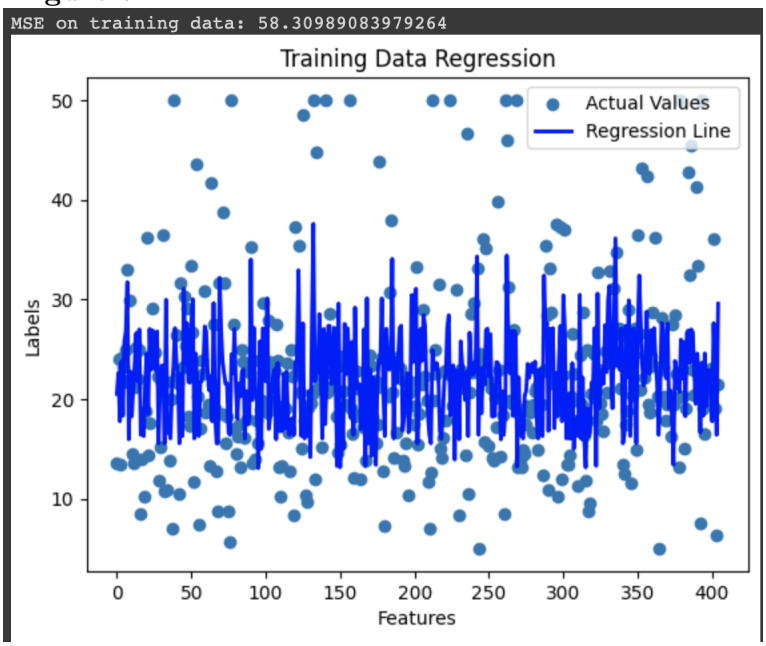


(a) Experiment 6: Graph for true values and predicted values across different parameters for Linear Regression

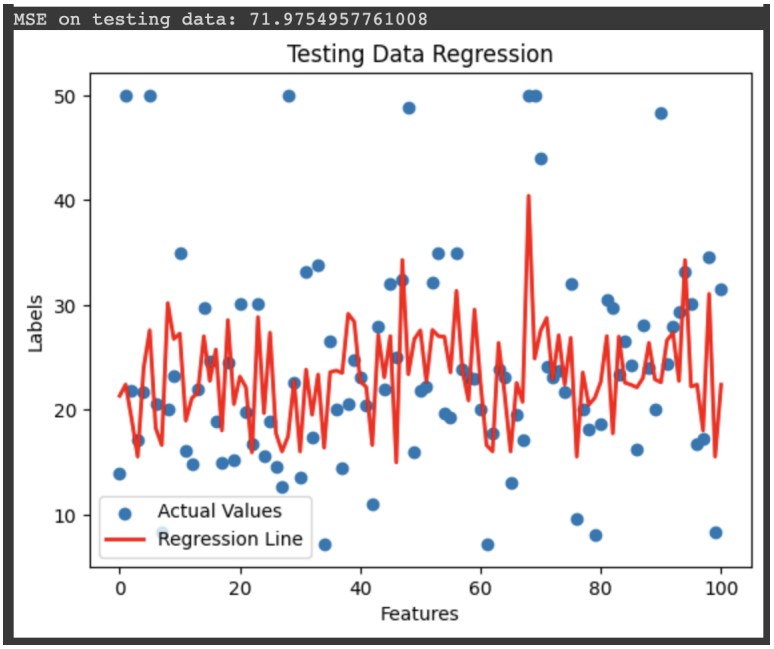


(b) Experiment 6: Graph for evaluation metrics across different parameters for Logistic Regression

Figure 7:



(a) Experiment 7: Graph for Non-Linear regression using Gaussian Bases: Training



(b) Experiment 7: Graph for Non-Linear regression using Gaussian Bases: Testing

Figure 8:

```
Analytical Solution:

Convergence Time:   0.0006954669952392578
MSE:  27.132995535962323

 Mini Batch Size = 8 Solution:

Convergence Time:   0.033548593521118164
MSE:  159.1893784122269

 Mini Batch Size = 256 Solution:

Convergence Time:   0.010907411575317383
MSE:  283.28557008714887
```

(a) Experiment 8: Comparing Analytical model fitting to mini Batch SGD

References

[1] David Harrison and Daniel L Rubinfeld. Hedonic housing prices and the demand for clean air. *Journal of Environmental Economics and Management*, 2004.

[2] University of Toronto. The boston housing dataset, 1996.

[3] Fairlearn Userguide. Revisiting the boston housing dataset - fairlearn 0.10.0.dev0 documentation, 2018.