COMP 551 Mini Project 1 Report

Group 21: Taran Dwivedula, Sarah Bellemare, Max-Henri Chanut

October 7, 2023

Abstract

In this project, we investigated two fundamental machine learning models—Linear Regression and Logistic Regression—applied to two distinct benchmark datasets. The primary aim was to gain practical experience in implementing and analyzing these models while acquiring insights into their capabilities and limitations. Our findings encompass a deeper understanding of the practical nuances of Linear and Logistic Regression models, specifically the effects of hyper-parameters, different gradient descent techniques, the importance of preprocessing, and the importance of having a rich dataset.

1 Introduction

In this project, our primary objective was to implement and investigate the performance of Linear and Logistic Regression, with Stochastic Gradient Descent. We applied these models to two distinct benchmark datasets, the Boston Housing and Wine datasets respectively.

We observed that the Boston Housing dataset provided a very good base for Linear Regression to perform. We believe this was due to the rich features and statistics that have been included in this dataset, providing many different features for Linear Regression to analyze, and specifically, many opportunities to find trends in the features and plot along them, which is what Linear Regression specializes in. Prior research also supports our claim, "Our data base is superior to others because it contains a large number of neighborhood variables (necessary to isolate the independent influence of air pollution) and more reliable air pollution data" (Harrison, D., & Rubinfeld, D. L., 1978).

Concerning the Wine dataset, prior studies have explored the classification challenges posed by this dataset, with research highlighting its relevance in the context of pattern recognition and classification (Forina, M., et al., 1990). However, we did observe large improvements when testing different techniques of Stochastic Gradient Descent by tuning various hyper-parameters.

2 Datasets

For both datasets, we first addressed incomplete or incorrectly populated data in the dataset by dropping these samples. We had determined that this would not cause an issue with the size of the dataset, as we found there to be relatively few samples of bad data. This ensured that we were able to train on data of the highest quality that was available to us. Next, we applied mean-wise normalization to our features in both datasets, in anticipation of numerical issues when training and predicting with our models. This allowed for ease of computation and better performance, while retaining much of the information from the original data. The trade-off was in our favour.

Dataset 1: Boston Housing Dataset

The first dataset, the Boston Housing dataset, comprised 506 data samples and originally contained 14 real attributes. However, due to ethical concerns raised later in this paragraph, we removed one feature, "B," and retained 13 attributes. We initially examined the class distribution of the target variable, the Median value of owner-occupied homes (MEDV). Understanding the distribution of housing prices is crucial, as it can provide insights into the affordability and accessibility of housing in different neighborhoods. While conducting the exploratory analysis, we also paid attention to potential ethical concerns. In Dataset 1, we acknowledged and addressed ethical issues related to the removal of the "B" feature. The decision to remove this feature was driven by the recognition that certain variables can raise ethical questions, such as issues of bias or discrimination. Ethical considerations are especially pertinent in datasets related to housing, as they can impact fair housing practices and policies. Beyond the numbers, we sought to understand the broader context of the datasets. For instance, we considered how factors like air quality, crime rates, and accessibility to public services might impact housing prices. This context can uncover potential ethical concerns related to environmental justice, accessibility, and affordability.

Dataset 2: Wine Dataset

The second dataset, the Wine dataset, encompassed 178 data samples distributed across three different classes, and consisted of 13 attributes. As this dataset primary objective is for testing, we did not recognize any potential ethical concerns in this dataset and did not remove any data, except for malformed data or data with errors.

In summary, our exploratory analysis not only focused on understanding the datasets' basic statistics and class distributions but also emphasized the importance of ethical considerations.

3 Experiments and Results

3.1 Benchmark Experiments

We first evaluated our models using an 80-20 train/test split on both datasets, and used default hyper-parameters (1000 epochs, 0.01 learning rate). This provided a good starting benchmark for our future experimentation. Our analytical linear regression's mean squared error came out to about 22.778, while that of our SGD model came out to about 34.694. It makes sense that the analytical model outperformed the SGD model, as the analytical model directly solves for the best weights, therefore providing the minimal error. Our logistic regression peformed at an accuracy of about 91.111%. Additional metrics for our logistic regression model (precision, recall, F1) are displayed in Table 1. Next, we implemented 5-fold cross validation on both of our models, to slight improvement. These results are displayed in Table 2.

3.2 Tuning of Hyper-Parameters

Pertaining to our models using stochastic gradient descent, we attempted to find optimal values for our proportion of testing data, minibatch sizes, and the learning rate. Figure 1 shows a plot of the training and testing metrics over the proportion of data left as the test set. For our linear regression model, this seems to be optimized at around 0.3, while it is around 0.6 for logistc. We found it interesting how these optimal values are different for both models. Next, we looked at optimizing minibatch sizes. Figure 2 shows the convergence in epochs of different minibatch sizes

for our linear regression model. A point of interest to us was the fact that the plot for 8 minibatches was worse than all other sizes at 1500 epochs, but quickly surpassed all other experiments by 3000. Contrary to the linear regression model, accuracy converged quickly for all minibatch sizes in our logistic regression model, as shown in figure 3. Again, the smallest minibatch size outperformed the rest. We found it interesting how accuracy in our logistic regression model was able to converge so quickly in comparison to our linear regression model. Figures 4 and 5 show our results when attempting to optimize the learning rate for both models. In both cases, performance became better as we increased the learning rate within our range. However, we observed gradients growing to numerical overflow and loss increasing when increasing the learning rate beyond this point.

3.3 Optimizing Metrics

We implemented a grid-search algorithm to optimize metrics for both models, to determine which combination of parameters gave the best result. For the Boston dataset, we determined that the mean squared error was a reliable measure of loss which we decided to optimize. For the Wine dataset, we decided to optimize the accuracy, since after looking at the dataset, we determined there wasn't a need to prioritize one class over another (unlike when classifying cancer, where the cost of false positives is outweighed by the benefit of capturing all positives). For our linear regression model, we attempted to find the best combination of minibatch size and learning rate, which resulted in a minumum mean squared error of 27.261 using a batch size of 256 and a learning rate of 0.1. For our logistic regression model, we attempted to find the best combination of epochs and learning rate, which resulted in a maximum accuracy of 97.222% using 2000 epochs and a learning rate of 0.001.

3.4 Enriching the Feature Set with Gaussian Basis Functions

We attempted to use gaussian basis functions to further enrich the feature set of Dataset 1. We did this by randomly generating 5 of these functions, and creating a new feature for each one, for a total of 5 new features added to the dataset. However, contrary to what we expected, the error got slightly worse, from 22.778 to 23.191. This is a very slight shift, and perhaps due to bad luck with random numbers. However, we suspect that this is due to the fact that the model has more points to minimize the residuals of.

3.5 Analytical vs. SGD Linear Regression

Throughout our experiments, we have also noted how the analytical linear regression model performs compared to the stochastic gradient descent model. We have observed that the analytical linear regression model has always outperformed the stochastic gradient descent model, for both computation time and loss minimization. We think that, due to the relatively small size of samples and features, the analytical model was able to get away with having a faster computation time. But in real life cases with feature counts in the millions, and dataset size growing even larger, performing this one step matrix multiplication is not practical. We understand that analytical linear regression always provides the optimal solution for weights, but the computational workload quickly becomes intense. Using stochastic gradient descent, while not always providing the optimal solution, can quickly get us to a good estimate of the optimal solution, with the computational workload increasing much more slowly as input size increases.

3.6 Applying More Hyper-Parameters

One of the things which we noted above was our gradients getting large, possibly due to overstepping and missing the minimum. To combat this, we attempted to implement momentum in our stochastic gradient descent for both models. Figure 6 shows the performance of our model as a function of different momentum parameters used during stochastic gradient descent. The results show that this did help both models greatly, achieving very good results compared to many of our other experiments. For linear regression, the MSE was steadily minimizing until a momentum value of around 0.9. After this, loss increased, possibly due to the gradients again overshooting more than smoothing out. For logistic regression, accuracy steadily improved, reaching a high of about 97%. Another issue we thought about was our models overfitting to the data, especially for the Boston dataset in which we had relatively few samples compared to the many distinct features we had. This is why we wanted to experiment with L_p regularization weight penalty. Table 3 shows the performance of our models over different values of p used in regularization. While our logistic accuracy did not change for any experiment, possibly due to the small amount of samples and small amount of classes, our linear regression model saw slight improvement. Error was extremely large for $p = 0.5$, however as we increased p, we saw improvement, however it seemed to converge at a value slightly less than our original MSE.

4 Discussion and Conclusion

4.1 Points of Note

There were several points which we have observed from our experiments which were remarkable to us and which we wanted to explore the reasoning and the potential cause behind. Firstly, we noted that the optimal train-test split for both models were far apart from each other, but we concluded that this was actually due to the data more than the model. We concluded that a proportion that is too large would hurt the model by allowing it less data to learn from, while a proportion that is too small could result in overfitting to the given training data. We determined that since the Boston dataset was far larger then the Wine dataset, this optimal point would be shifted to the left for the Boston dataset, as the datasets differ in size in orders large enough such that the actual number of samples in the training and testing groups would matter. We note that analyzing this problem would be interesting, but experimentation to determine the best value is usually more practical. Another point we observed pertains to the convergence of minibatch sizes for our SGD linear regression model. We noted that the smallest minibatch size in our testbed, 8, was outperformed by all other sizes, however quickly surpassed them between 1500 and 3000 epochs. We concluded that this was due to an effect in learned information. We determined that smaller minibatches could allow for more information to be learned, as there is more possible combinations of the data that the model learns from. However, this takes a longer time to converge. So, while the rest of the minibatch sizes were reaching their convergence, the 8 minibatch size was in the process of sharp optimization, although requiring much more epochs to do so. We concluded that, while a smaller minibatch size could help in finding a lower error, in real life cases there would be a computational workload trade-off that would need to be determined, and larger mini batch sizes tend to help speed up convergence.

4.2 Conclusions

In general, we found our grid search algorithm to be a highly effective way to fine tune our hyper-parameters and improve the performance of our models, as it ended up producing the best per-

forming metrics out of all of our experiments. The 5-fold cross validation technique also marginally improved the performance of our models. The Gaussian Basis technique did not work as we had hoped, but we believe that in more general cases, it can help the performance of models. Our methods of fine tuning the hyper-parameters turned out to be integral in improving the performance of our models. This was one of our key takeaways; that there are many good ways to tune hyper-parameters, and selecting a combination of hyper-parameters and fine tuning them is integral to improving model performance. Something that we'd like to explore in the future is experimenting with even more hyper-parameters, seeing how many we can add, and seeing if more hyper-parameters can correlate to better model performance.

5 Statement of Contributions

- Max-Henri: Implementing analytical linear regression and mean squared error, implementing minibatch sgd, implementing momentum for stochastic gradient descent, initial testing for momentum experiments, report abstract, report section 1, report section 2.
- Sarah: Implementing logit, softmax, accuracy, precision, recall, and f1 for logistic regression, implementing experiment 3 (different test sizes), implementing experiment 5 (different learning rates), implementing experiment 6 (grid search), implementing experiment 7 (gaussian basis functions), report section 3.3.
- Taran: Implementing feature normalization for both models, implementing gradient calculation and gradient descent for both models, implementing minibatch sgd, implementing experiment 1 (train test split), implementing experiment 2 (5-fold cross validation), implementing experiment 4 (growing minibatch sizes), implementing L_p regularization for stochastic gradient descent, report abstract, report section 1, report section 2, report section 3, report section 4, all plots and tables.

6 References

1. Harrison, D., & Rubinfeld, D. L. (1978). Hedonic prices and the demand for clean air. Journal of Environmental Economics and Management, 5(1), 81-102.
2. Forina, M., et al. (1990). PARVUS: An Extendable Package for Data Exploration, Classification, and Correlation. Institute of Pharmaceutical and Food Analysis and Technologies, Via Brigata Salerno, 16147 Genoa, Italy.

A Appendix

	Class 1	Class 2	Class 3
Precision	0.875	0.938	0.923
Recall	0.933	0.833	1.000
F1	0.903	0.882	0.960

Table 1: Metrics of logistic regression on an 80-20 train/test split

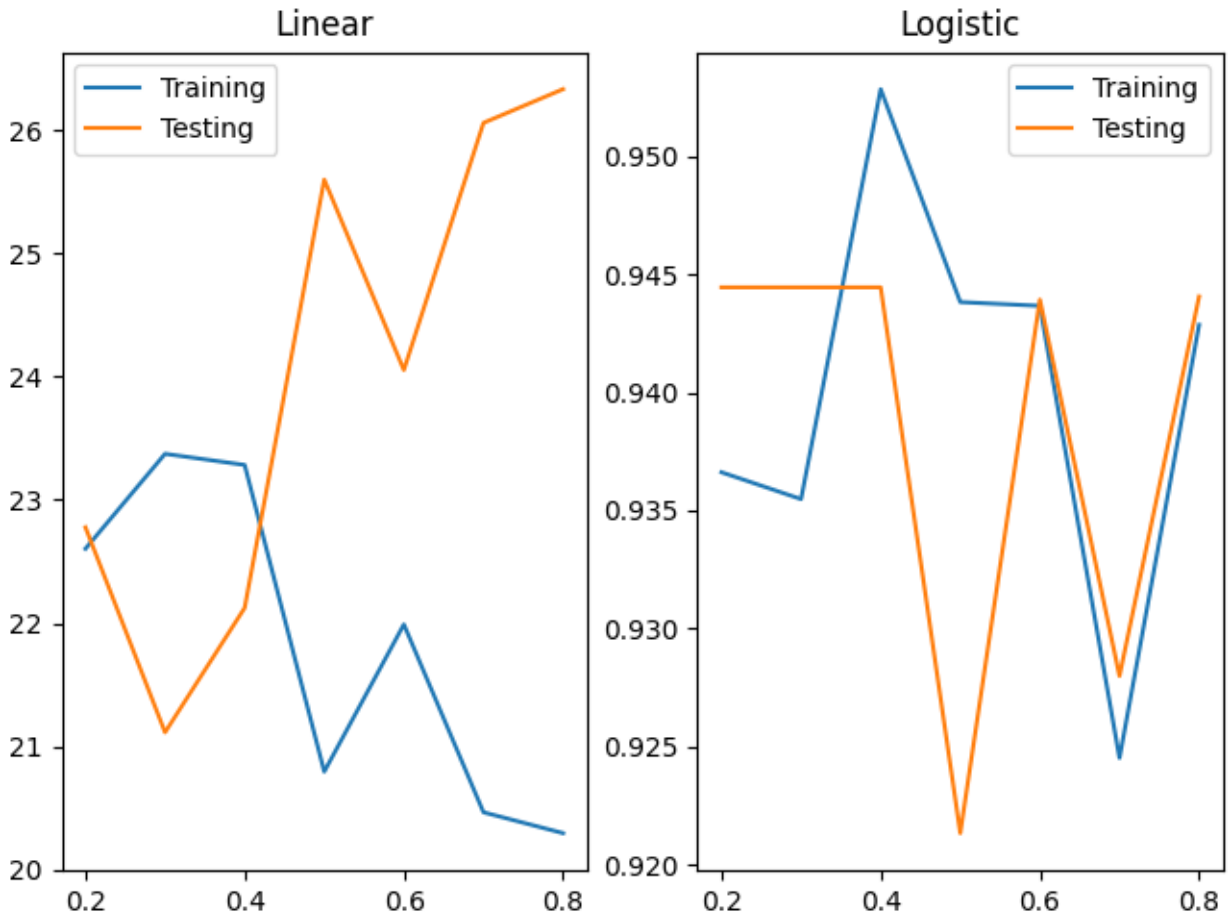

Figure 1: Performance of different test-split proportions for linear and logistic regression

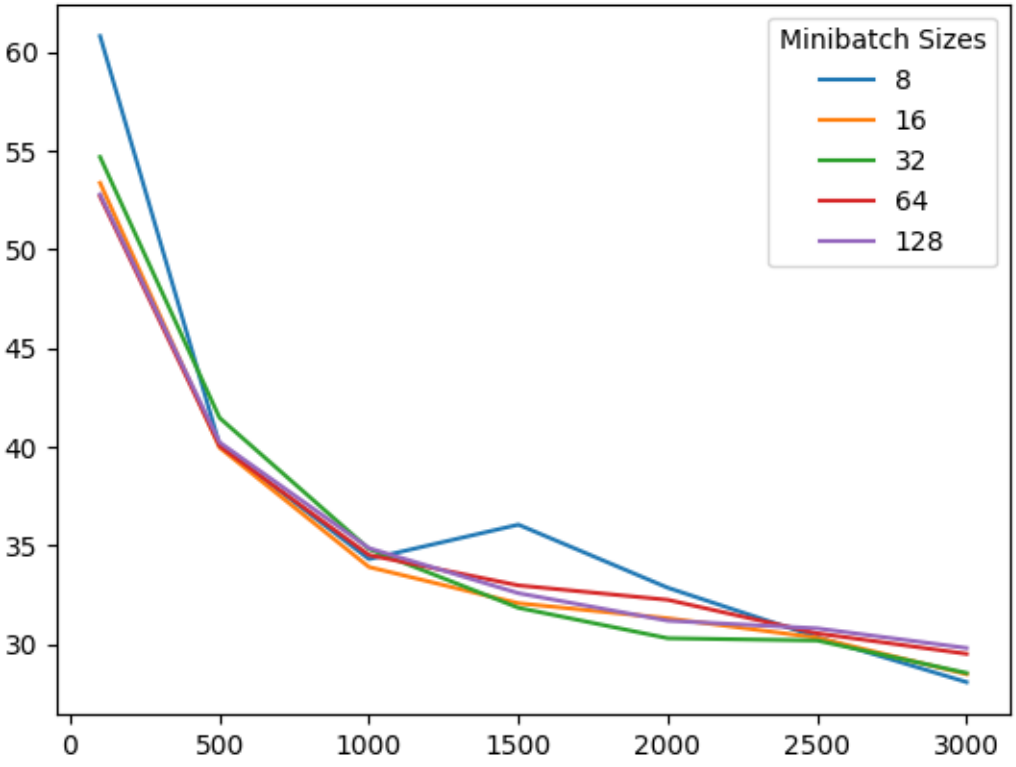

Figure 2: Convergence of MSE in epochs of different minibatch sizes for SGD linear regression

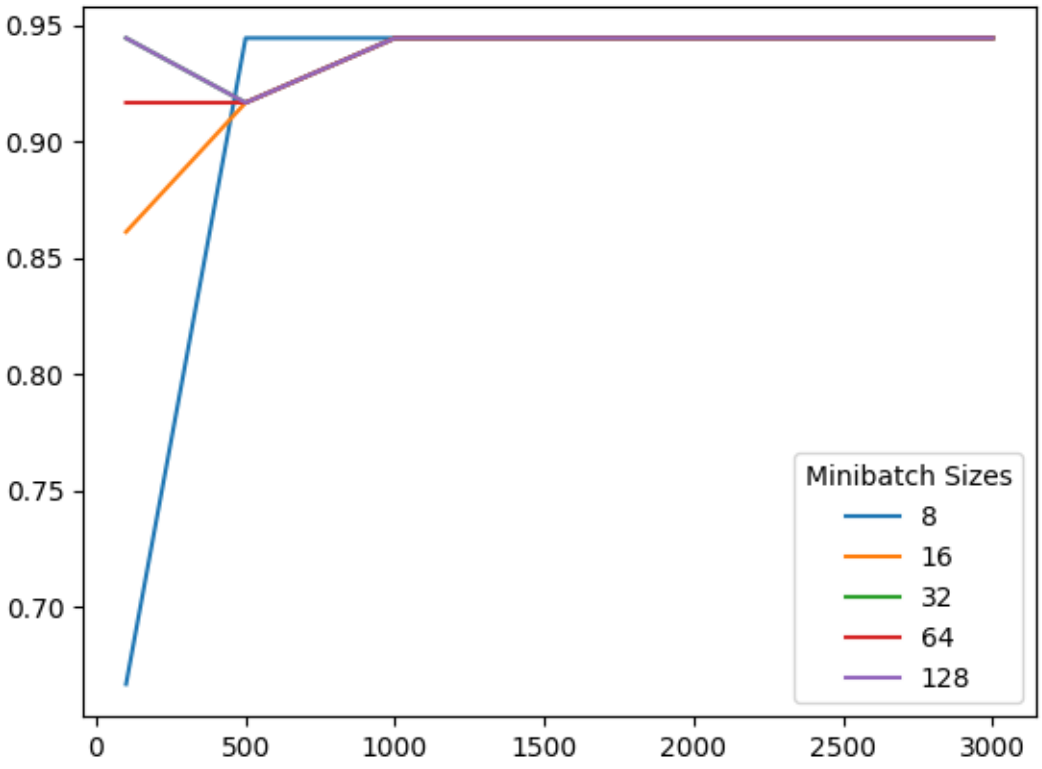

Figure 3: Convergence of accuracy in epochs of different minibatch sizes for logistic regression

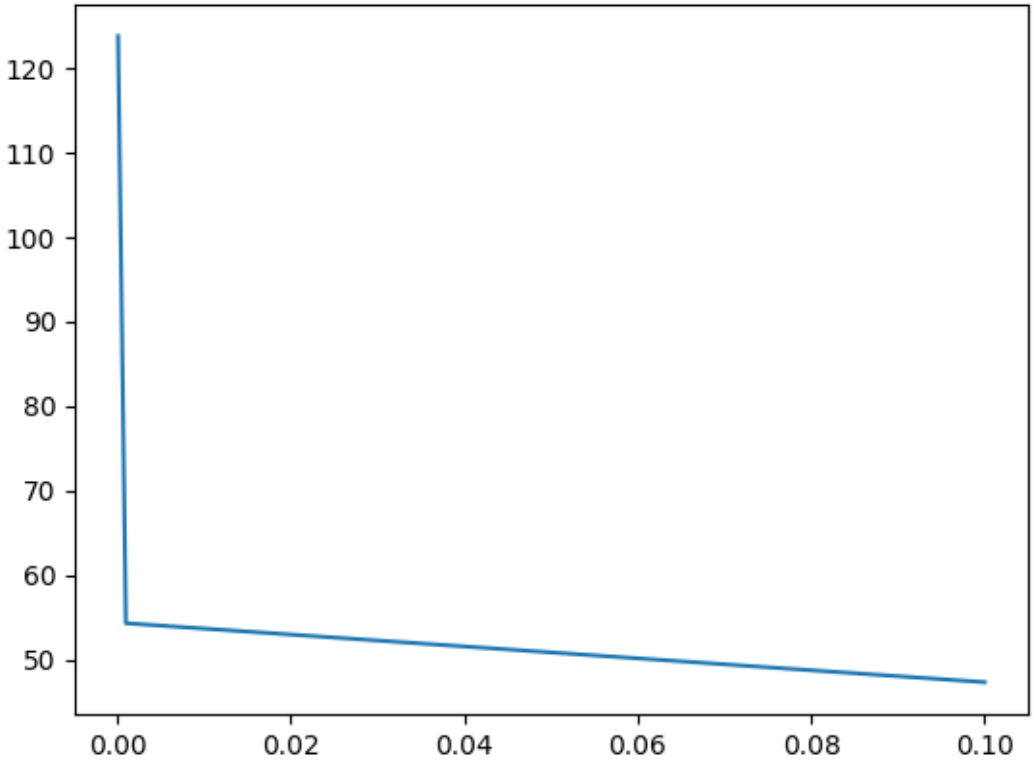

Figure 4: MSE as a function of learning rate for SGD linear regression

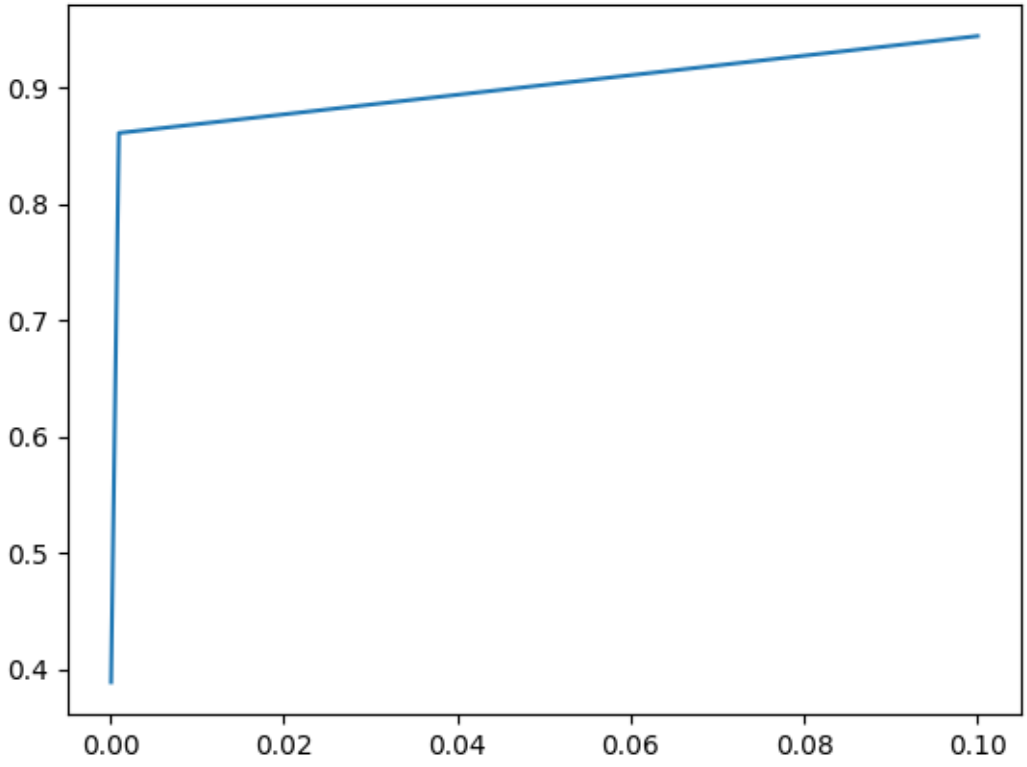

Figure 5: Accuracy as a function of learning rate for logistic regression

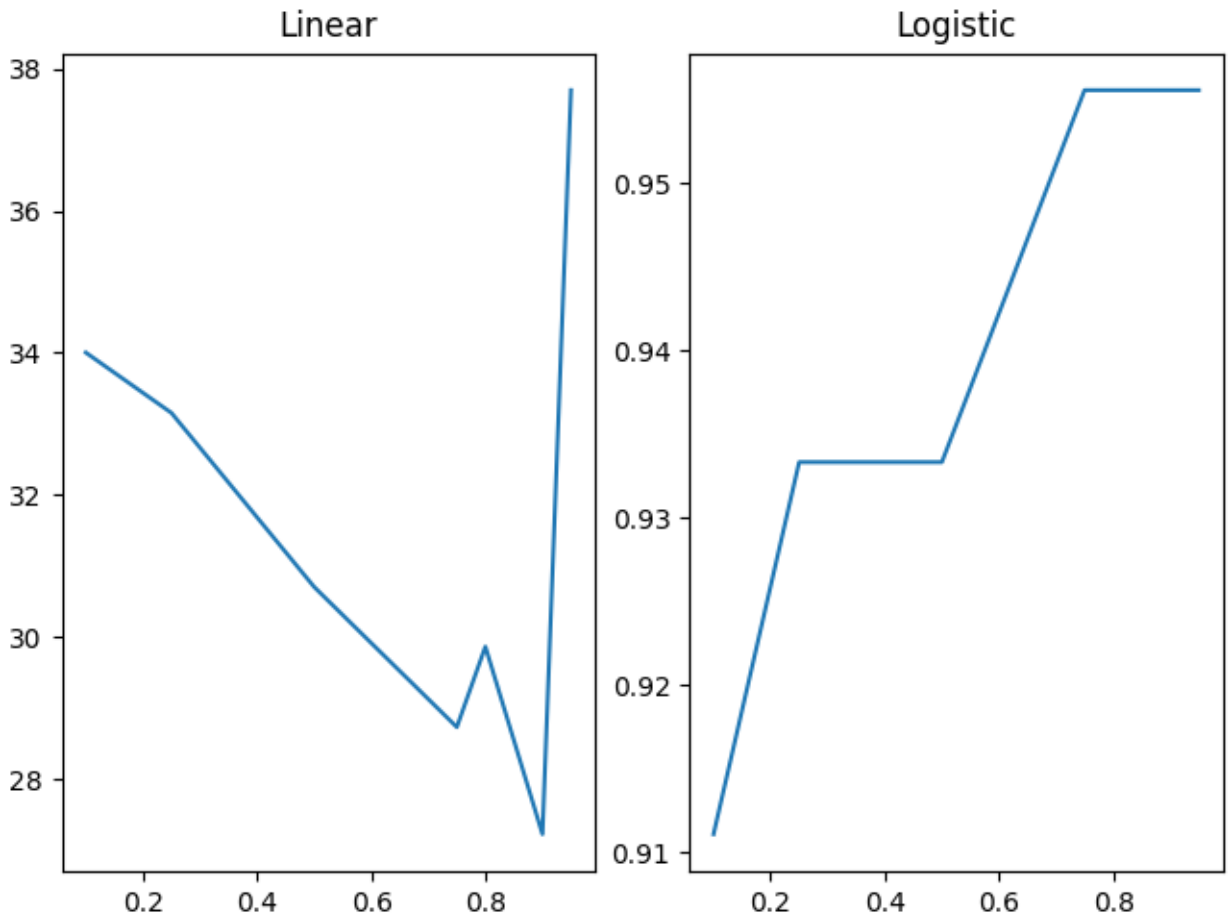

Figure 6: MSE and Accuarcy as a function of the momentum parameter for both linear and logistic regression

	Analytical Linear (MSE)	SGD Linear (MSE)	Logistic (Accuracy)
Training	24.520	39.659	0.938
Testing	23.005	35.444	0.924

Table 2: Metrics after performing 5-Fold cross validation

p	Linear (MSE)	Logistic (Accuracy)
0.5	4.01e27	0.911
1.0	43.459	0.911
1.5	33.932	0.911
2.0	33.237	0.911
2.5	32.531	0.911

Table 3: Metrics after applying L_p regularization with fixed $\lambda = 0.1$