Mini Project 1 Report

Giane Mayumi Galhard (51261276747), Daria Goptsii (261275056), Yixuan Qin (261010963)

COMP 551, Fall, McGill University

Abstract

This project implements two supervised machine learning models: Linear Regression and Logistic Regression, along with their mini-batch stochastic gradient descent optimization. The goal is to observe the performance of these models in predicting the motor_UPDRS score for Parkinson's disease and classifying breast cancer diagnoses. Experiments highlight the critical role of data pre-processing and train-test set splitting. They also provide a general view of how mini-batch hyperparameters affect convergence speed. Results show that Linear Regression struggles to achieve low error on the Parkinson's Telemonitoring dataset, while Logistic Regression achieves strong and consistent classification accuracy on the Breast Cancer Wisconsin dataset. To gain deeper insight into the datasets, we experimented with feature engineering techniques for the Parkinson's Telemonitoring data and also evaluated the effect of gradient descent with momentum on model performance.

1 Introduction

In medical applications, Machine Learning is capable of analyzing versatile data and effectively identifying patterns in data, which can lead to effectual predictions of diseases and health conditions [1]. When the model is trained in a range of features that are associated with a known outcome, it is considered a supervised technique. Therefore, the model can make predictions for new unseen data [2].

The goal of this project is to implement two different supervised machine learning models: Linear Regression for a Parkinson's telemonitoring dataset [3] to predict `motor_UPDRS` values, and Logistic Regression to classify breast cancer diagnoses [4]. In addition, various experiments were conducted to understand the behavior of the models, including changing hyperparameters, implementing non-linear improvements, and momentum to find the best possible performance for the models.

2 Datasets and Data Cleaning

The *Parkinson's Telemonitoring dataset* contains 5,875 biomedical voice measurements collecting from 42 patients, each with 19 features. This project targets `motor_UPDRS`.

During the data cleaning stage, patient ID and test time were removed, outliers were handled with the IQR rule, and `total_UPDRS` **was excluded** to avoid target leakage. Although removing strongly correlated features is essential in Linear Regression, removing them with F-score-based k-best feature selection did not improve test performance, and therefore they were kept.

The *Breast Cancer Wisconsin (Diagnostic) dataset* has 569 samples with 30 cell nucleus features. During pre-processing, IDs were dropped and the target was mapped to numeric values (malignant=1, benign=0).

For both datasets, non-numeric values were standardized, incomplete rows removed, numeric features z-score scaled, and categorical/binary features one-hot encoded.

We computed descriptive statistics and plotted histograms for both datasets. In the Parkinson's dataset, some features (e.g., `Jitter`, `Shimmer`) are highly skewed with long-tailed outliers, while others (e.g., `HNR`, `RPDE`, `DFA`) are closer to symmetric. In the Breast Cancer dataset, features vary greatly in scale. Hence, scaling before model training is crucial for model performance.

Possible ethical concerns may include privacy, bias, and inappropriate use. Even without their IDs, patients may still be re-identified with their medical information, and predictions must not be misinterpreted as medical advice.

3 Results

3.1 Performance of linear regression and fully batched logistic

Linear regression on the *Parkinson's Telemonitoring dataset* showed limited accuracy, though outlier removal and scaling slightly reduced RMSE (7.50 → 7.37) (Figure 1). Logistic regression on the *Breast Cancer Wisconsin dataset* performed strongly, with good generalization between train and test sets. Further feature engineering (for example use of non-linear bases) is needed to improve linear regression results.

```
=== Unified Performance Table (80/20 split) ===
                                                             MSE    RMSE     MAE      R2
Model               Split Variant
Linear Regression   Train Baseline                       56.0966  7.4898  6.3080  0.1581
                    Test  Baseline                       56.3915  7.5094  6.3654  0.1165
                    Train Outliers removed, scaled+OHE   56.0566  7.4871  6.3549  0.1711
                    Test  Outliers removed, scaled+OHE   54.3581  7.3728  6.2708  0.1609
Logistic Regression Train Fully batched                   0.0088  0.0938  0.0088  0.9624
                    Test  Fully batched                   0.0263  0.1622  0.0263  0.8869
```

Figure 1: Linear and logistic regression model performance (80/20 train–test split)

3.2 Weights of features

Feature Weights (Sorted by Absolute Value)

Feature	Weight	AbsWeight
Shimmer:APQ3	-81.4934686040145	81.4934686040145
Shimmer:DDA	79.91273121459649	79.91273121459649
Jitter:RAP	-47.98137092715928	47.98137092715928
Jitter:DDP	47.820912636490455	47.820912636490455
sex_0	8.171021938222847	8.171021938222847
sex_1	5.686102333736189	5.686102333736189
Jitter(Abs)	-4.264630061715335	4.264630061715335
Shimmer:APQ5	-2.9696990603188147	2.9696990603188147
Jitter(%)	2.666373025575904	2.666373025575904
Shimmer:APQ11	2.3548362212088336	2.3548362212088336
HNR	-2.089489554482639	2.089489554482639
DFA	-2.0107779722243104	2.0107779722243104
Jitter:PPQ5	1.2253967445367624	1.2253967445367624
NHR	-1.0962200054102775	1.0962200054102775
Shimmer	0.9001340056103355	0.9001340056103355
age	0.8882009623585658	0.8882009623585658
Shimmer(dB)	0.7908585614490573	0.7908585614490573
PPE	0.7000239951190901	0.7000239951190901
RPDE	0.5461663131775905	0.5461663131775905

Figure 2: Weights of linear regression

Top 15 Features by |Weight|

Feature	AbsWeight
num__texture3	2.3937200825499474
num__radius2	2.366237940220028
num__symmetry3	1.8265455780960815
num__compactness2	-1.7651485196250227
num__area2	1.7064267752417146
num__concavity3	1.5649079723507833
num__concave_points1	1.5520144004134202
num__area3	1.406551078960837
num__perimeter2	1.3251627275139797
num__radius3	1.303075270394426
num__compactness1	-1.212430130372182
num__concavity1	1.1993563628781623
num__concave_points3	1.0903952476555263
num__fractal_dimension2	-1.0231779832623498
num__concave_points2	1.0078814807113672

Figure 3: Top-15 weights of logistic regression

The largest weights of linear regression were assigned to `Shimmer:APQ3`, `Shimmer:DDA`, `Jitter:RAP`, and `Jitter:DDP`, with opposite signs despite measuring similar voice irregularities. This highlights the limitations of linear models and suggests that non-linear transformations may improve performance. (Figure 2)

For breast cancer classification, `texture3`, `symmetry3`, and error-based features (`radius2`, `area2`) were strong positive predictors, while compactness (`compactness1`, `compactness2`) and `fractal_dimension2` showed strong negative weights. (Figure 3) These features with high absolute weights could tilt the decision boundary towards them, meaning the **decision boundary is more sensitive to changes in these features**.

3.3 Growing Subset of Training Data

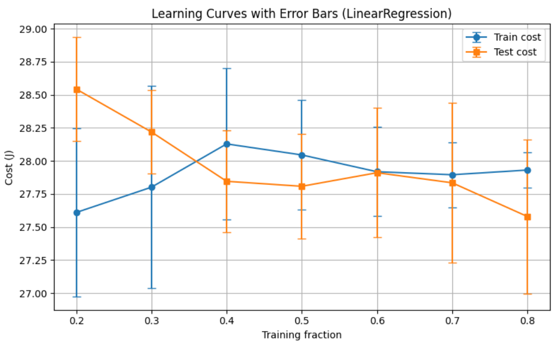

Figure 4: Cost as training data fraction increases (Linear Regression)

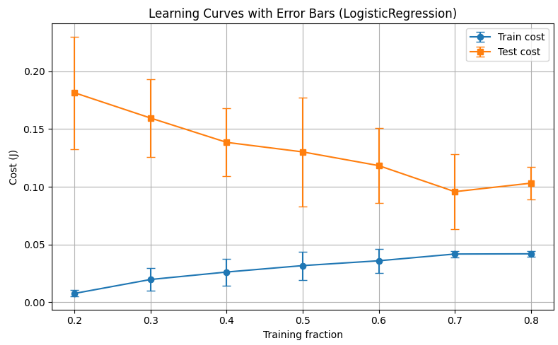

Figure 5: Cost as training data fraction increases (Logistic Regression)

Overall, a larger training dataset reduces test cost, while training cost increases as fitting a larger and more diverse dataset is more challenging (Figure 4). For Logistic Regression, the test cost starts to rise at a training fraction of 0.7, which may indicate that adding more training data may hinder test performance (overfitting) (Figure 5).

3.4 Growing mini-batch sizes

Parameters used:

Linear Regression: learning_rate = 0.01, max_iters=10
Logistic Regression: learning_rate = 0.001, max_iters=10
learning rates are selected such that clear curves can be observed.

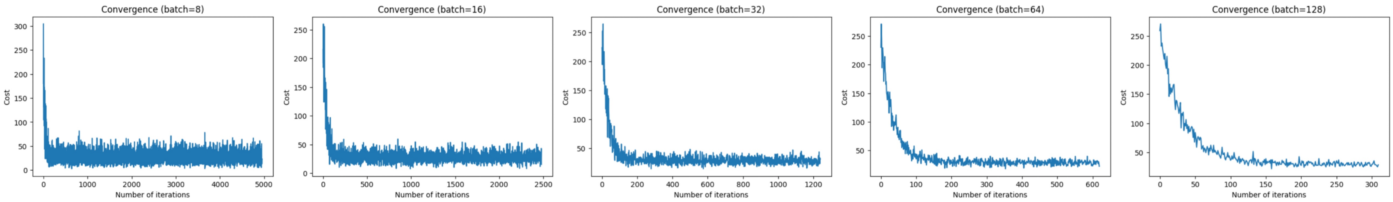

Figure 6: Learning curve as batch size increases(Linear Regression)

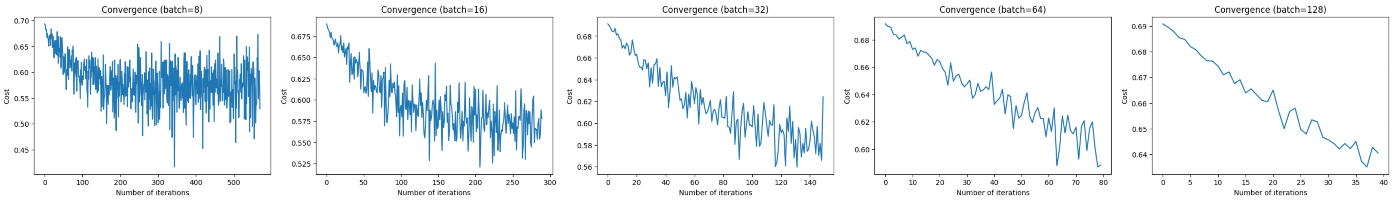

Figure 7: Learning curve as batch size increases(Logistic Regression)

For large mini-batch sizes, fewer yet more stable gradient updates are performed, producing smoother and clearer learning curves with smaller fluctuations.

Performance Summary (Linear Regression)

	final_cost	convergence_iter
8	23.116746611795204	4949.0
16	25.856516061470717	2479.0
32	19.9644341772889	1239.0
64	24.831661530765462	619.0
128	29.947412057762705	309.0
full batch	197.60543968963518	9.0

Figure 8: Performance summary (Linear Regression)

Performance Summary (Logistic Regression)

	final_cost	convergence_iter
8	0.6227151128029218	44.0
16	0.5398478744122281	139.0
32	0.6442764582481612	25.0
64	0.6213361928073186	36.0
128	0.6360317298596039	36.0
full batch	0.6756876082047041	9.0

Figure 9: Performance summary (Logistic Regression)

Larger batch sizes converge in fewer iterations, but the gradient estimates are less precise, while too small a batch size performs poorly due to noisy updates. The batch size that works best in general is **32 for Linear Regression and 16 for Logistic Regression**, both better than the fully-batched baseline.

3.5 Growing learning rate

Different learning rate values (0.001, 0.005, 0.01, 0.05, and 0.1) were tested. The training fraction was 0.5, and the following parameters were used:
Linear Regression: max_iters=5000
Logistic Regression: max_iters=10000, threshold = 0.5

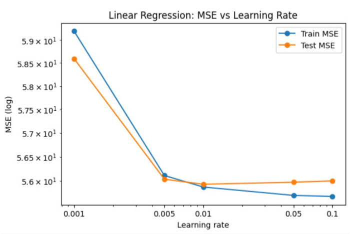

Figure 10: MSE vs Learning Rate (Linear regression)

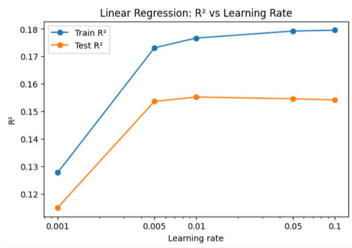

Figure 11: R^2 vs Learning Rate (Linear regression)

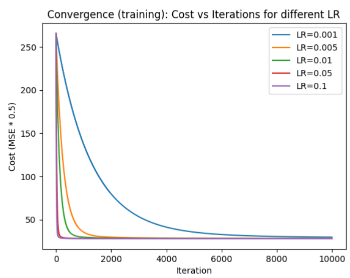

Figure 12: Convergence for different Learning Rate (Linear regression)

For linear regression, all learning rates above 0.005 displayed subtle changes and converged to similar MSEs and R^2 values. In addition, the higher the learning rate, the faster the convergence. 0.01 is a good trade-off between speed to converge and stability for MSE and R^2.

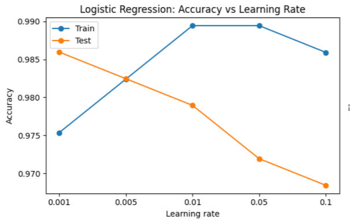

Figure 13: Accuracy vs Learning Rate (Linear regression)

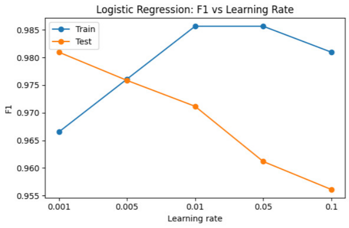

Figure 14: F1-Score vs Learning Rate (Linear regression)

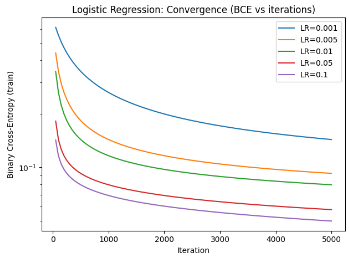

Figure 15: Convergence for different Learning Rate (Logistic regression)

On the other hand, for logistic regression, the binary cross-entropy per iterations shows that the bigger the learning rate, the faster the training loss is minimized for the number of iterations. However, the test metrics show that, as the learning rate grows, the accuracy and F1-Score drop. Therefore, generalization works better under small steps and 0.001 is the best trade-off among tested values for in this solution.

3.6 Analytical Linear Regression vs Mini-batch Stochastic Gradient Descent Linear Regression

In this experiment, the training fraction was also 0.5. The batch size for the mini-batch stochastic gradient descent-based Linear Regression was 32, and the learning_rate was 0.005. The mini-batch stochastic gradient descent-based linear regression converges with the analytical linear regression training optimum, but it has stochastic noise (Figure 16). It is the recommended approach when you need a good enough result that uses less memory per step.

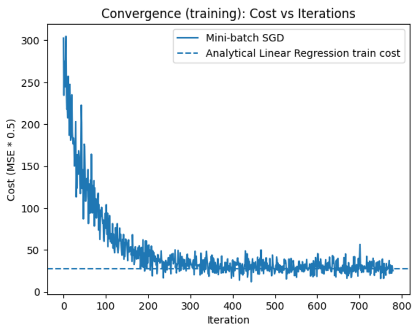

Figure 16: Convergence of Mini-batch SGD and Analytical Linear Regression

3.7 Non-linear improvement

We expanded the numeric features using non-linear basis functions (Gaussian and Sigmoid) with different numbers of basis functions (D = 10, 20, 35). The Gaussian basis with D=20 performed best, comparing to smaller Gaussian expansions and all sigmoid variants (Figure 17). However, further increasing D led to overfitting and hindered generalization performance

3.8 Gradient Descent with momentum

For gradient descent, it is possible that, during the training process, it gets stuck in local minima. Therefore, momentum is a strategy implemented to accelerate convergence by adding a "velocity" term [5]. The objective is to get closer to an optimal solution with the same number of iterations. Linear regression with Gradient Descent with and without momentum is compared. The parameters used were learning_rate = 0.01, max_iters=5000 and momentum=0.9.

Basis	MSE	RMSE	MAE	R2
gauss_D35	19.7708	4.4464	3.4068	0.6948
gauss_D20	28.3924	5.3285	4.1673	0.5617
gauss_D10	36.3445	6.0286	4.8722	0.439
sigm_D35	21.9963	4.69	3.468	0.6605
sigm_D20	21.9248	4.6824	3.6496	0.6616
sigm_D10	38.2973	6.1885	5.0672	0.4088

Figure 17: Test Metrics with Non-Linear Basis

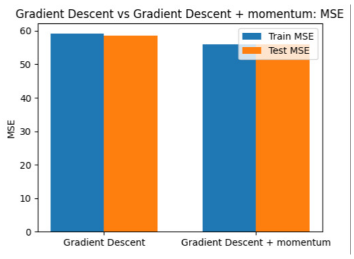

Figure 18: MSE comparison with and without momentum

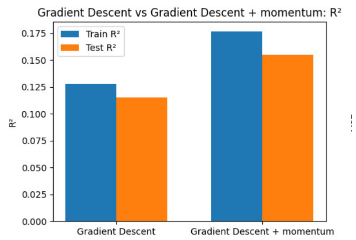

Figure 19: R^2 comparison with and without momentum

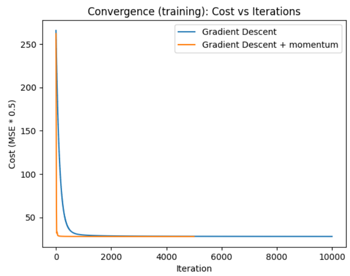

Figure 20: Convergence with and without momentum

Gradient descent with momentum can minimize the cost in a smaller number of iterations compared to the implementation without momentum (Figure 20). Therefore, it can converge faster. It also provides slightly better MSE (Figure 18) and R^2 (Figure 19), as, for the same number of iterations, the momentum implementation is closer to the optimum.

4 Discussion and Conclusion

This project demonstrates that data pre-processing is an important step for achieving reliable performance in both Linear and Logistic Regression, and consequently, it should be done carefully. While removing highly correlated features can improve linear regression in some cases, we found that for the Parkinson's Motor Scores dataset, excluding these features may actually reduce prediction R^2. Applying non-linear basis expansions can better capture complex relationships and improve performance. Overall, the dataset was not a good fit for Linear Regression.

The training set size is also an important factor for a model that can generalize and predict unseen data. A training set that is too small can limit the model's precision, while an overly large training set without proper regularization may risk overfitting in this context.

In mini-batch stochastic gradient descent, larger learning rates and batch sizes often lead to faster convergence; however, excessively large values can cause divergence or poor performance. This approach is particularly efficient for large datasets, as it has less computational cost per step.

Improvements for model performance include introducing non-linear regression on the Parkinson's Telemonitoring dataset, by transforming the feature matrix with non-linear basis functions (e.g., Gaussian). Future work could also explore adding regularization to gradient descent, tuning hyperparameters systematically, and trying alternative evaluation metrics for a fuller picture of model performance.

5 Statement of Contributions

Daria Goptsii was responsible for data preprocessing, implementing experiments 1 and 2, and improvements to linear regression using non-linear bases. Giane Mayumi was responsible for experiments 5, 6, and implementing momentum on Gradient Descent for an extra experiment. Yixuan was responsible for running experiments 3 and 4. Overall, all team members helped each other during implementation, discussed the results together, and contributed to the report.

References

[1] S. C. Mana, G. Kalaiarasi, Y. R, L. S. Helen, and R. Senthamil Selvi, "Application of machine learning in healthcare: An analysis," in *2022 3rd International Conference on Electronics and Sustainable Communication Systems (ICESC)*, 2022, pp. 1611–1615.

[2] J. A. M. Sidey-Gibbons and C. J. Sidey-Gibbons, "Machine learning in medicine: a practical introduction," *BMC Med. Res. Methodol.*, vol. 19, no. 1, p. 64, Mar. 2019.

[3] M. L. Athanasios Tsanas, "Parkinsons telemonitoring," 2009.

[4] William Wolberg, Olvi Mangasarian, Nick Street, W. Street, "Breast cancer wisconsin (diagnostic)," 1993.

[5] S. Masood, M. N. Doja, and P. Chandra, "Analysis of weight initialization methods for gradient descent with momentum," in *2015 International Conference on Soft Computing Techniques and Implementations (ICSCTI)*, 2015, pp. 131–136.