Assignment_1

☑ Reviewed ☐

Ashish Bargoti

2022114

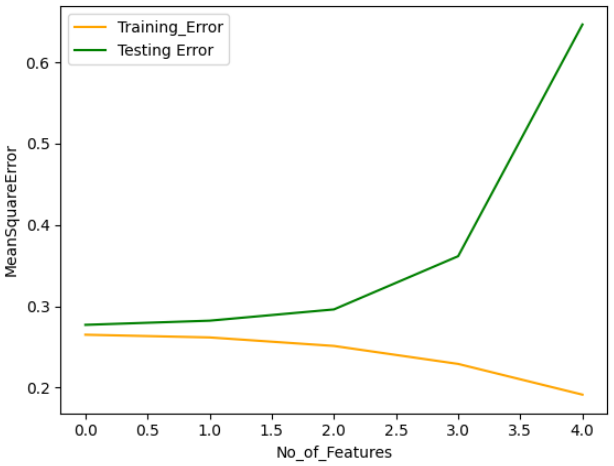
ML Assignment 1

1. **(10 points) Section A (Theoretical)**
   (a) (2 marks) You are developing a machine-learning model for a prediction task. As
   you increase the complexity of your model, for example, by adding more features
   or by including higher-order polynomial terms in a regression model, what is most
   likely to occur? Explain in terms of bias and variance with suitable graphs as
   applicable.

Ans:

As we increase the complexity of the model by adding more features we are decreasing bias but increasing the variance.The model will move towards overfitting that is it is going to perform better on the training model but it will give poor performance over new unseen data.

Bias decreased because the model can fit more complex relationships in the data, capturing finer details and reducing underfitting however varianve increases because the model becomes more sensitive to small variations in the training data, potentially leading to overfitting.



- As We can see from the graph that as we increase the complexity of graph by increasing no of features the meanSquareError for training data decreases i.e Bias decreases but meanSquareError for Testing Data Increases.This shows that increasing complexity leads to overfitting..

(b) (3 marks) You're working at a tech company that has developed an advanced email

filtering system to ensure users' inboxes are free from spam while safeguarding legit-
imate messages. After the model has been trained, you are tasked with evaluating

its performance on a validation dataset containing a mix of spam and legitimate
emails. The results show that the model successfully identified 200 spam emails.

However, 50 spam emails managed to slip through, being incorrectly classified as le-
gitimate. Meanwhile, the system correctly recognised most of the legitimate emails,

with 730 reaching the users' inboxes as intended. Unfortunately, the filter mistak-
enly flagged 20 legitimate emails as spam, wrongly diverting them to the spam

folder. You are asked to assess the model by calculating an average of its overall
classification performance across the different categories of emails.

Ans:

Positive:Spam Emails identified

let true positive be that spam emails are identified correctly as spam hence false negative becomes spam emails incorrectly identified as legitimate:
Then

TP=200;

FN=50

True Negative(TN):Legitimate emails correctly classified as legitimate.

TN:730

False Positive(FP):Legitimate emails incorrectly classified as spam

FP=20

Precision=TP/(TP+FP)=200/(200+20)=0.90

Recall=(TP/TP+FN)=200/(200+50)=200/250=0.8

Accuracy=
(TP+TN)/(TP+TN+FP+FN)=200+730/(200+50+730+20)=930/1000=0.93

F1 Score=2*(precision*recalll/(precision*recall))=2*(0.90∗0.8/(0.90+0.8))=0.85

(c) (3 marks) Consider the following data where y(units) is related to x(units) over a
period of time: Find the equation of the regression line and, using the regression

x y
3 15

6 30
10 55
15 85
18 100

Table 1: Table of x and y values equation obtained, predict the value of y when x=12.

Ans let $\hat{y}^{(i)}$ be the predicted value and $x^{(i)}$ be the input data point then

let $\hat{y}^{(i)} = a x^{(i)} + b$

This is a univariate linear regression

→ To find the optimal parameter of univariate linear regression we can minimize the MSE loss function

By minimizing loss function with respect to a, b, we get

$$a = \frac{\frac{\sum x_i y_i}{n} - \frac{\sum x_i}{n} \times \frac{\sum y_i}{n}}{\frac{\sum x_i^2}{n} - \left(\frac{\sum x_i}{n}\right)^2}$$

$$a = \frac{\overline{xy} - \bar{x}\bar{y}}{\overline{x^2} - (\bar{x})^2}$$

and $b = \bar{y} - a\bar{x}$

x_i	y_i	x_i^2	$x_i y_i$
3	15	9	45
6	30	36	180
10	55	100	550
15	85	225	1275
18	100	324	1800
sum 52	285	694	3850
mean 10.4	57	138.8	770
	↓ $\bar{x}$	↓ $\bar{y}$	$\overline{x^2}$	$\overline{xy}$

$$a = \frac{770 - 10.4 \times 57}{138.8 - (10.4)^2}$$

$$= \frac{770 - 592.8}{138.8 - 108.16}$$

$$= \frac{177.2}{30.6}$$

$$a = 5.79 \approx 5.8$$

$$b = \bar{y} - a\bar{x} = 57 - (5.79) \times (10.4)$$

$$= 57 - 60.21$$

$$= (-3.2)$$

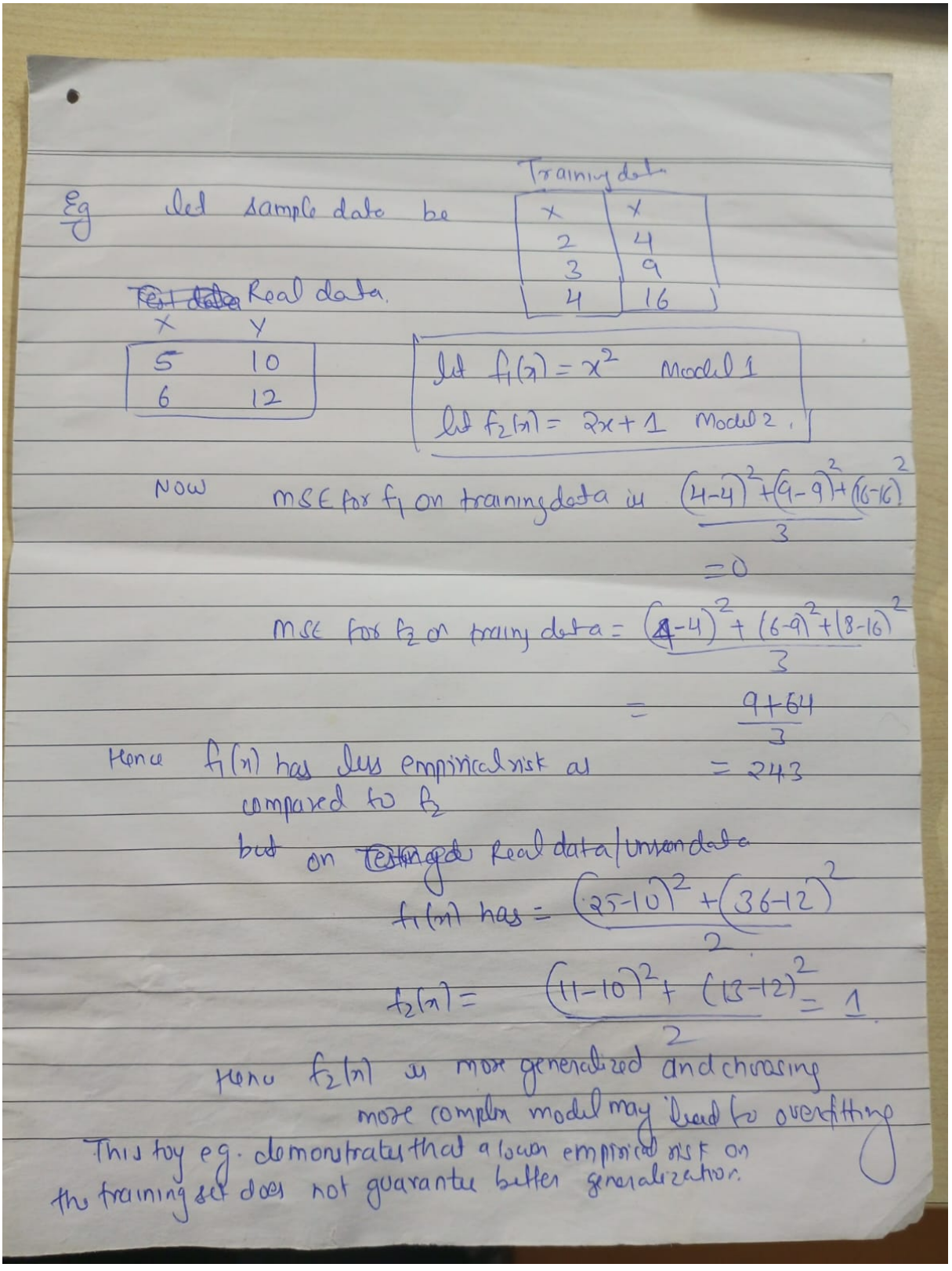
So, $\hat{y}^{(i)} = 5.8 x^{(i)} - 3.2$ eqn of Regression line

when $x = 12$

$\hat{y}(12) = 5.8 \times 12 - 3.2 = 66.4$ Ans

(d) (2 marks) Given a training dataset with features X and labels Y , let $\hat{f}(X)$ be the prediction of a model f and $L(\hat{f}(X), Y)$ be the loss function. Suppose you have two models, f1 and f2, and the empirical risk for f1 is lower than that for f2. Provide a toy example where model f1 has a lower empirical risk on the training set but may not necessarily generalize better than model f2.

Ans:



on real data f2(x) performs far better than f1(x) because being less simpler f2(x) is generalising better.

Section B (Scratch Implementation)
Implement Logistic Regression in the given dataset. You need to implement Gradient
Descent from scratch, meaning you cannot use any libraries for training the model (You
may use libraries like NumPy for other purposes, but not for training the model). Split

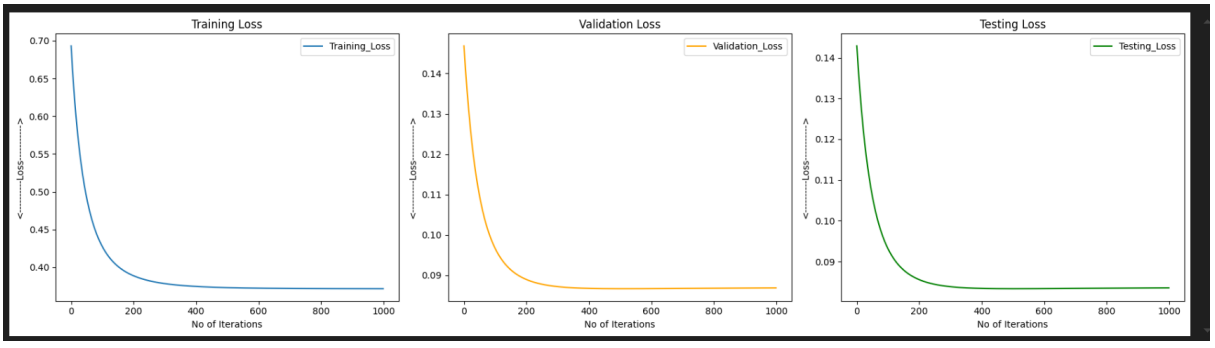
the dataset into 70:15:15 (train: test: validation). The loss function to be used is Cross-
entropy loss.

Dataset: Heart Disease

(a) (3 marks) Implement Logistic Regression using Batch Gradient Descent. Plot train-
ing loss vs. iteration, validation loss vs. iteration, training accuracy vs. iteration,

and validation accuracy vs. iteration. Comment on the convergence of the model.
Compare and analyze the plots.

Ans:

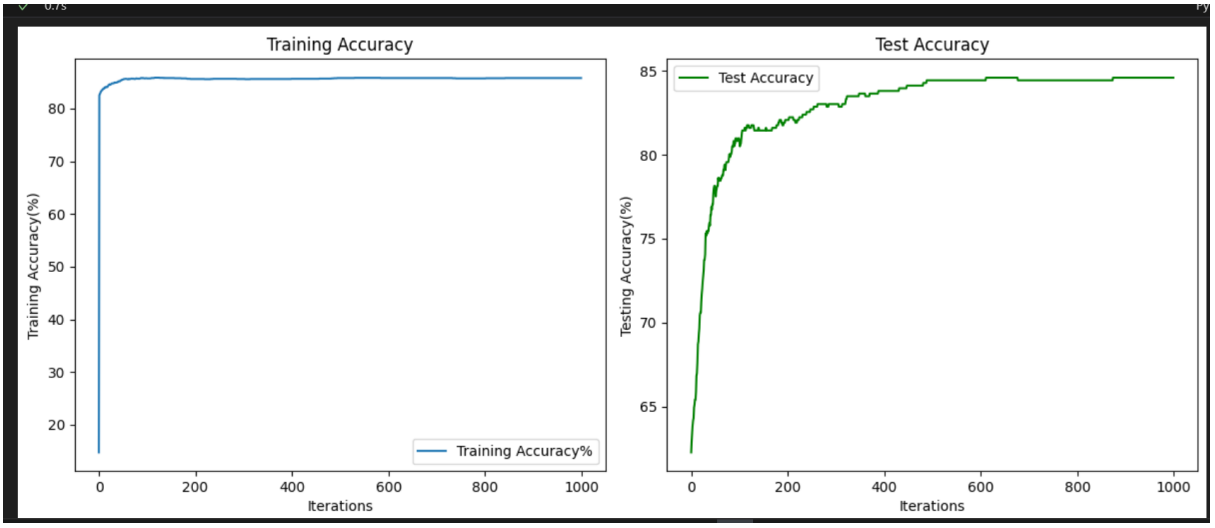


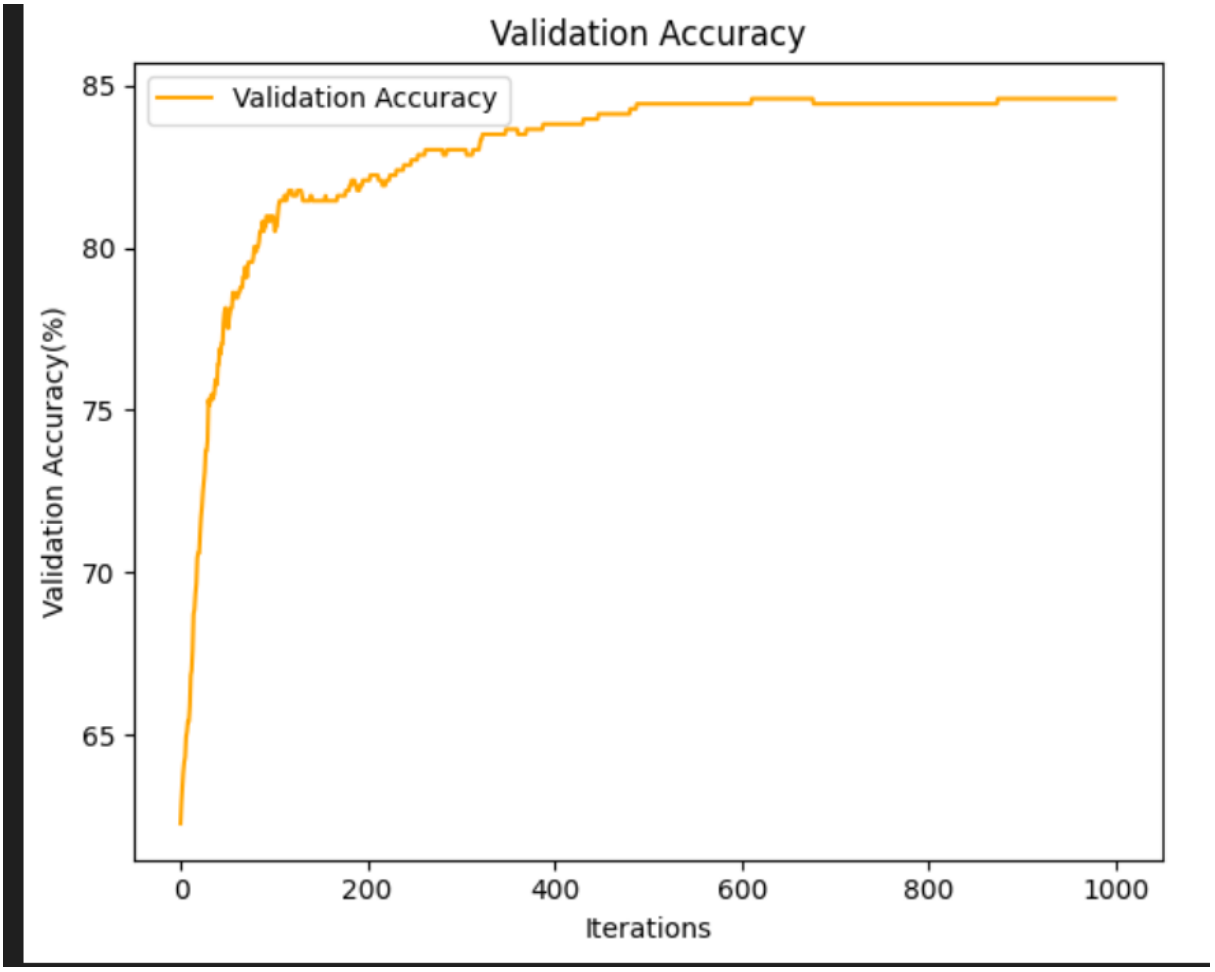
Loss Plots Analysis

The loss plots (training, validation, and testing) show a clear decreasing trend, indicating that the model is learning and improving with each iteration. The

training loss decreases rapidly initially, then slows down gradually, suggesting that the model is approaching convergence.

- The model appears to be converging, as the training loss is no longer decreasing significantly after around 800 iterations.
- There's a slight gap between the training and validation loss, especially towards the later iterations. This might indicate a tendency towards overfitting, where the model is learning the training data too well but might not generalize well to unseen data.
- The testing loss initially decreases but then plateaus, suggesting that the model's performance on unseen data has stabilized.
-





- Both training and testing accuracies show a steep rise in the early iterations. This indicates that the model is quickly learning the underlying patterns in the data.
- The training accuracy plateaus around 85%, suggesting that the model has converged.
- There's a noticeable gap between the training and testing accuracies. This suggests that the model might be overfitting to the training data, leading to suboptimal performance on unseen data.

(b) (2 marks) Investigate and compare the performance of the model with different
feature scaling methods: Min-max scaling and No scaling. Plot the loss vs. iteration

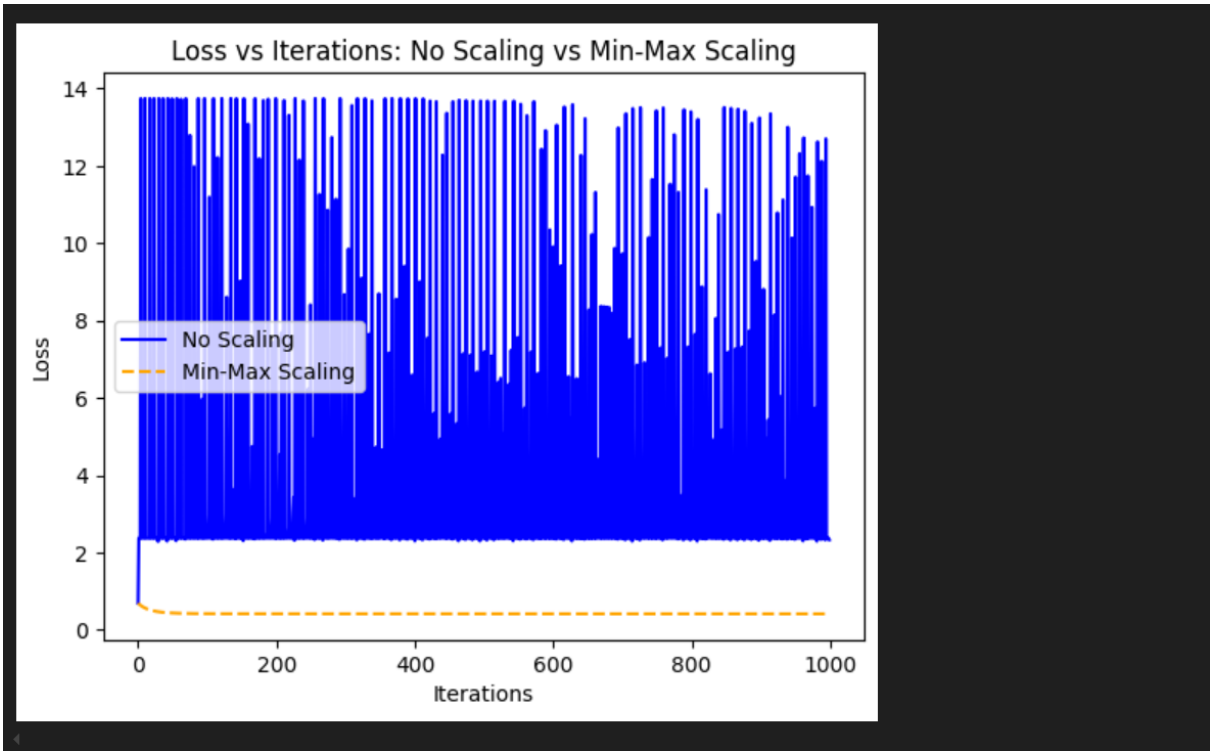
for each method and discuss the impact of feature scaling on model convergence.

For No Scaled Data:

Features: 15 samples: 2966
Cost after 0 iterations: 0.6931471805599454
Cost after 100 iterations: 2.374783564520836
Cost after 200 iterations: 2.374783564520836
Cost after 300 iterations: 2.374783564520836
Cost after 400 iterations: 2.301688666977855
Cost after 500 iterations: 2.369349282534281
Cost after 600 iterations: 9.911232586636071
Cost after 700 iterations: 9.730371658301546
Cost after 800 iterations: 2.374783564520836
Cost after 900 iterations: 2.374783564520836

For Min_Max Scaled Data:

Features: 15 samples: 2966
Cost after 0 iterations: 0.6931471805599454
Cost after 100 iterations: 0.4223928313459275
Cost after 200 iterations: 0.4197176105327445
Cost after 300 iterations: 0.4194855228225117
Cost after 400 iterations: 0.41931928665200974
Cost after 500 iterations: 0.41915781576555283
Cost after 600 iterations: 0.4189992496048626
Cost after 700 iterations: 0.41884346984446336
Cost after 800 iterations: 0.4186904116592798
Cost after 900 iterations: 0.41854001310828537



For No Scaled Data:

In this case, the data was used in its raw form without any feature scaling.

- The initial cost at iteration 0 is 0.6931, which is typical for logistic regression as this represents the log-loss when predictions are 50% for each class.
- There is a significant fluctuation in the cost values, with high jumps at certain points (e.g., iteration 600 with a cost of 9.9112).
- The model fails to show a consistent downward trend in the cost function, which indicates poor convergence. The cost oscillates and even increases in some cases (e.g., iteration 600).
- Hence, the model did not perform well when the raw data was used without scaling. The lack of scaling caused instability in the gradient descent process, as features with different scales affected the updates to the weights unevenly.

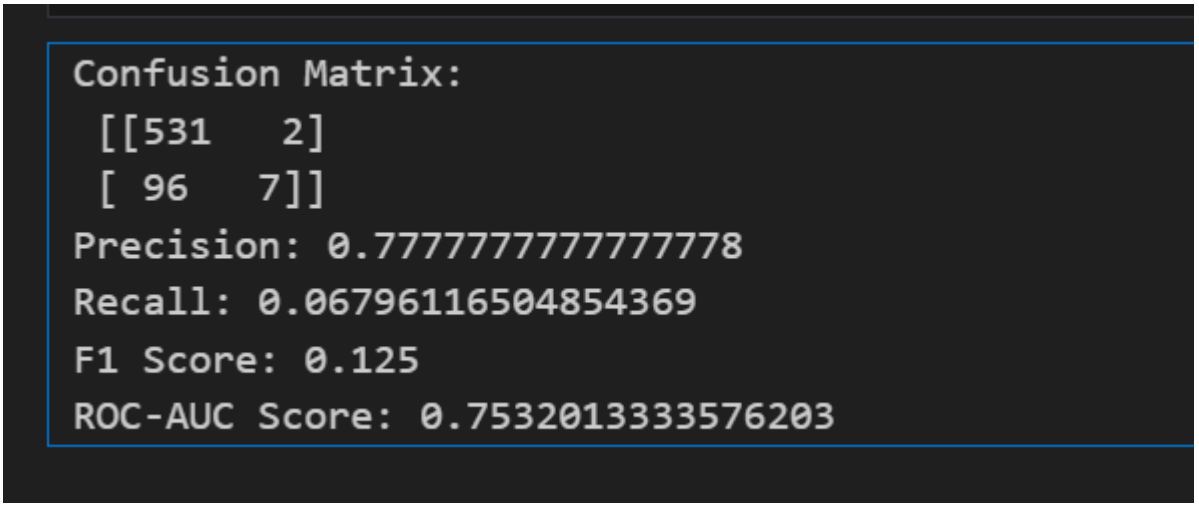
For Min_MAX Scaled DAta:

In this case, the features were scaled using min-max normalization, where each feature was scaled to a range of [0,1].

- The initial cost is 0.6931, the same as the unscaled data, representing an initial random prediction with 50% probability for each class.
- The cost decreases steadily over iterations, showing a smooth and continuous decline from 0.6931 to 0.4185. This indicates that the model is converging well.
- By iteration 900, the cost reaches 0.4185, which is significantly lower than the final cost with unscaled data.
- Hence Scaling the data using min-max normalization dramatically improved the model's performance. The gradient descent algorithm was able to converge smoothly, and the cost function consistently decreased, indicating effective learning.

Therefore, We can conclude that Min-max scaling helped to normalize the data, ensuring that all features contributed proportionally to the weight updates. This resulted in a smooth and consistent decrease in the cost function over time.

(c) (2 marks) Calculate and present the confusion matrix for the validation set. Report precision, recall, F1 score, and ROC-AUC score for the model based on the
validation set. Comment on how these metrics provide insight into the model's
performance.



Confusion Matrix

	Predicted Positive	Predicted Negative
Actual Positive	7	96
Actual Negative	2	531

Interpretation:

- **True Positives (TP)**: 7 - The model correctly predicted 7 positive cases(Heart Diseases).
- **True Negatives (TN)**: 531 - The model correctly predicted 531 negative cases(No Heart Diseases).
- **False Positives (FP)**: 2 - The model incorrectly predicted 2 cases as positive when they were negative.
- **False Negatives (FN)**: 96 - The model incorrectly predicted 96 cases as negative when they were actually positive.

This confusion matrix reveals that the model is highly skewed towards predicting negatives, as it correctly classified most negative samples but struggled to correctly classify positive ones.

This may be due to data on which model was trained since we realized that there were 85 percent zeroes and ony 15 percent ones.

Precision: 0.7778

- **Formula**: Precision = TP / (TP + FP) = 7 / (7 + 2) = 0.7778

- **Interpretation**: Precision measures the percentage of positive predictions that were correct. A precision score of 0.7778 indicates that when the model predicted a positive case, it was correct about 77.8% of the time.

Recall: 0.06796

- **Formula**: Recall = TP / (TP + FN) = 7 / (7 + 96) = 0.06796
- **Interpretation**: Recall measures how well the model identifies actual positive cases. The recall of 0.06796 means that the model only identified about 6.8% of the total actual positive cases, which is very low.

F1 Score: 0.125

- **Formula**: F1 = 2 * (Precision * Recall) / (Precision + Recall) = 2 * (0.7778 ∗ 0.06796) / (0.7778 + 0.06796) = 0.125
- **Interpretation**: Given the large imbalance between precision and recall, the F1 score is low (0.125), indicating that the model's overall ability to balance between correctly identifying positives and minimizing false positives is poor.

ROC-AUC Score: 0.7532

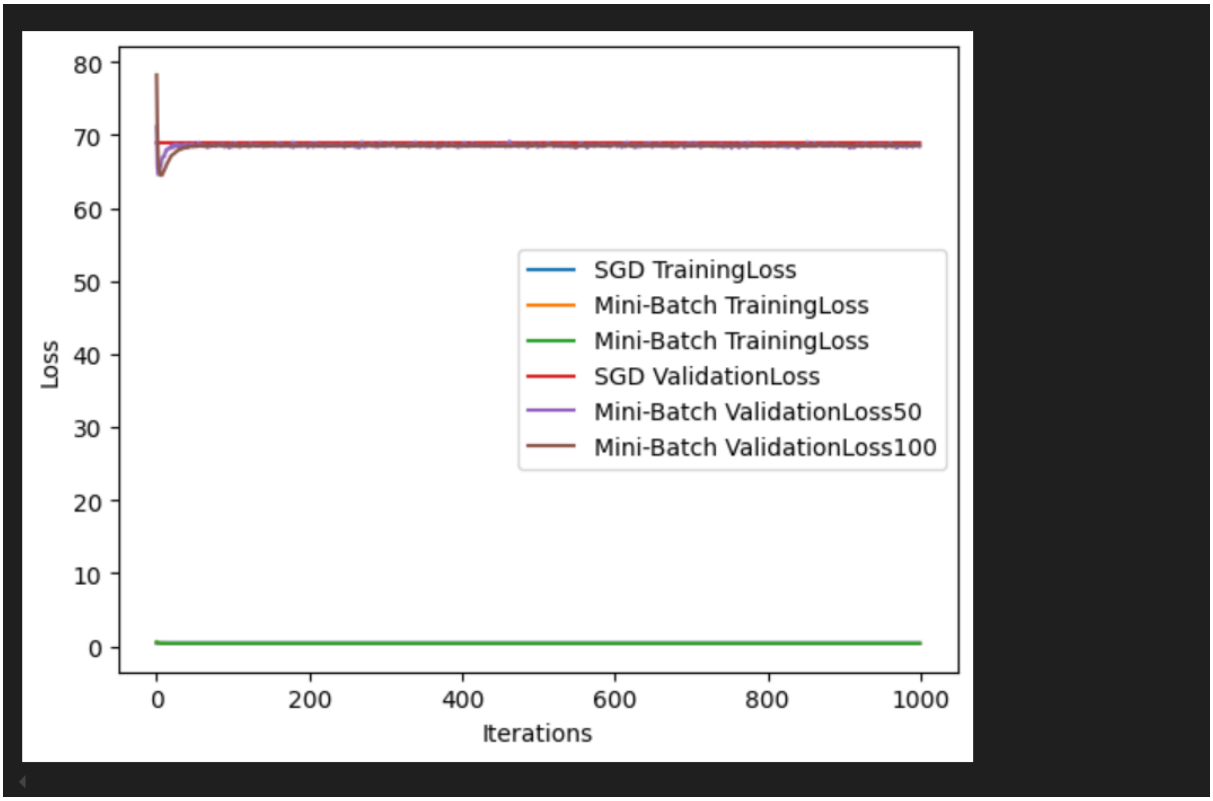
- **Interpretation**: A score of 0.7532 indicates that the model has a moderately good ability to distinguish between the two classes.

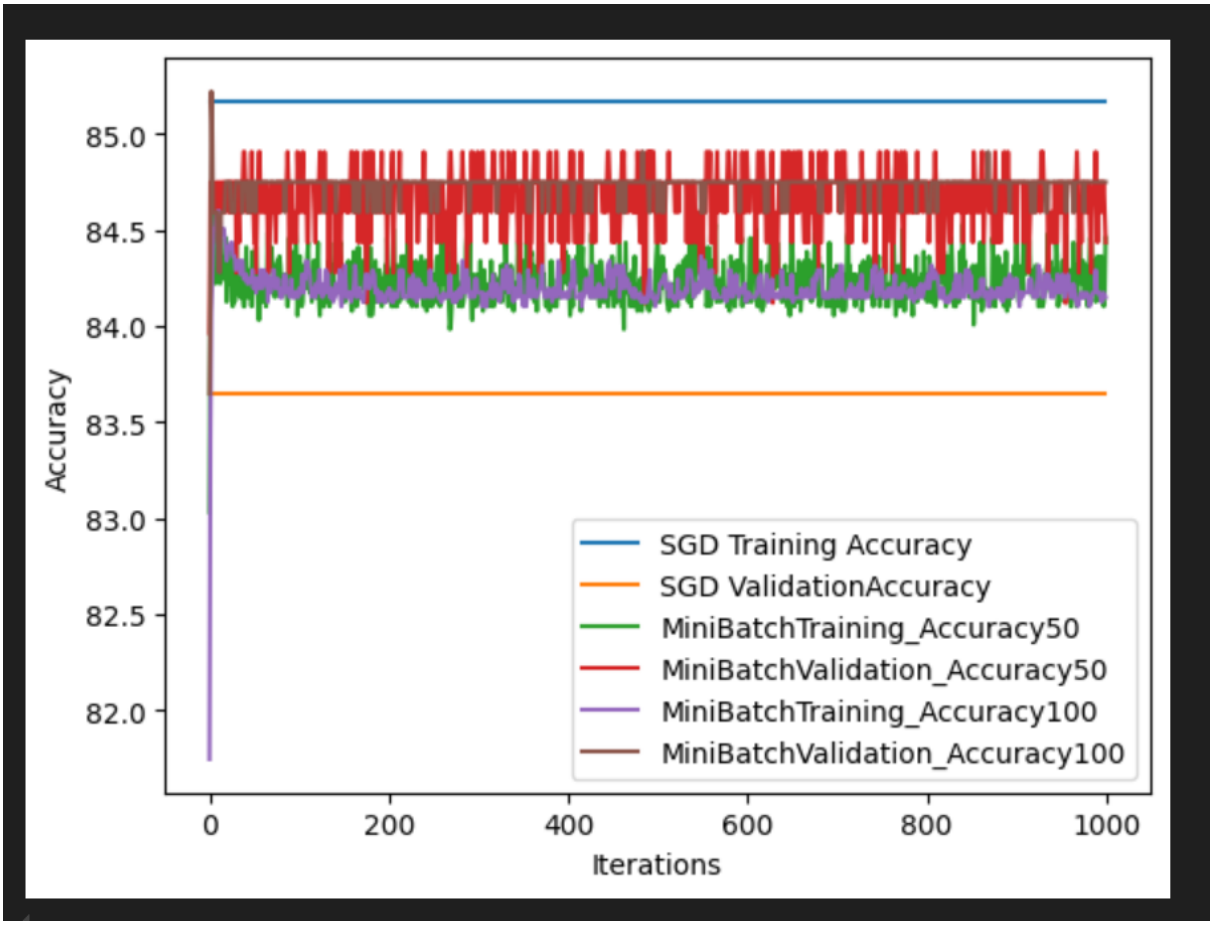
This suggests that the model is overfitting to the majority negative class and failing to generalize well to the minority positive class.

The ROC-AUC (Receiver Operating Characteristic - Area Under the Curve) score measures the model's ability to discriminate between positive and negative cases across various classification thresholds.

d) Implement and compare the following optimisation algorithms: Stochastic Gradient Descent and Mini-Batch Gradient Descent (with varying batch sizes,
at least 2). Plot and compare the loss vs. iteration and accuracy vs. iteration for
each method. Discuss the trade-offs in terms of convergence speed and

stability
between these methods.





- For SGD The loss decreases slightly at the beginning but stagnates after 200 iterations at a value of ~0.4136.
- Validation loss remains significantly high, suggesting poor generalization and stability for SGD.

Mini-Batch Gradient Descent (Batch Size 50):

- The loss shows a faster and more consistent decline, stabilizing at ~0.3716 after 100 iterations, indicating better convergence.

Mini-Batch Gradient Descent (Batch Size 100):

- The loss follows a similar pattern, stabilizing at ~0.3716, but it starts from a higher initial value than batch size 50.

- For SGD the accuracy on the training set remains constant at 83.65% across all iterations, indicating slow or stalled learning.
- Both mini-batch gradient descent methods achieve better accuracy (~84.75%) on validation data compared to SGD, showing more consistent improvement and stability.

- SGD show slow convergence speed.The loss stagnates after only 100 iterations with minimal improvement thereafter.
- For Mini Batch Gradient Descent, the convergence rate is aster than SGD. Both mini-batch configurations converge to a stable loss value after approximately 100 iterations, reflecting the advantage of smoother gradient updates with less variance compared to SGD.

TradeOFFs:

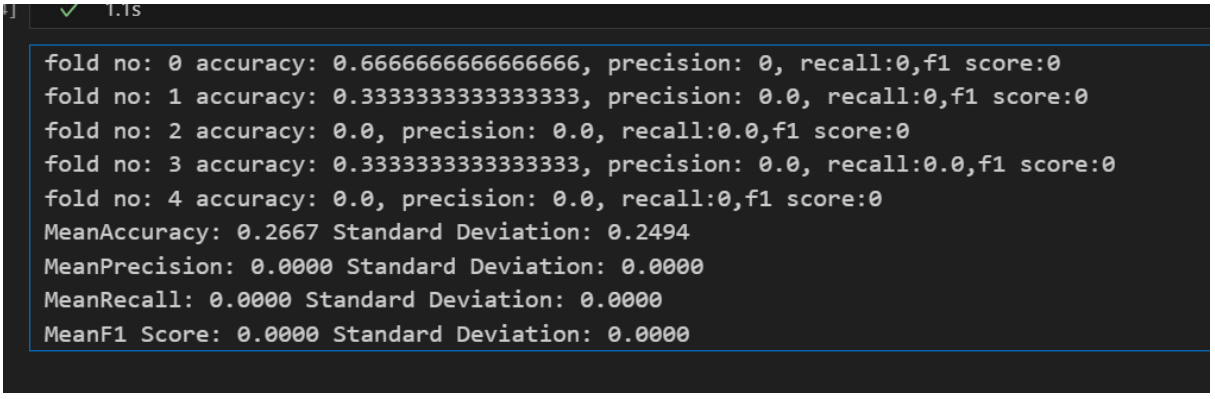
- if we want quicker convergence in early stages, SGD can be faster per iteration because it updates the model parameters after each sample (instead of after processing a batch of samples)
- However, because SGD updates after each sample, it introduces a lot of variance into the gradient updates. This means it can oscillate around the minimum and never settle exactly, leading to slower convergence in the long run.
- Mini-batch gradient descent strikes a balance between stability and efficiency. By updating parameters based on a batch of samples rather than one sample at a time, MBGD reduces the variance in updates.
- This makes it more stable compared to SGD, leading to smoother convergence and better generalization on validation data.

(e) (2 marks) Implement k-fold cross-validation (with k=5) to assess the robustness

of your model. Report the average and standard deviation for accuracy, precision,
recall, and F1 score across the folds. Discuss the stability and variance of the model's performance across different folds.

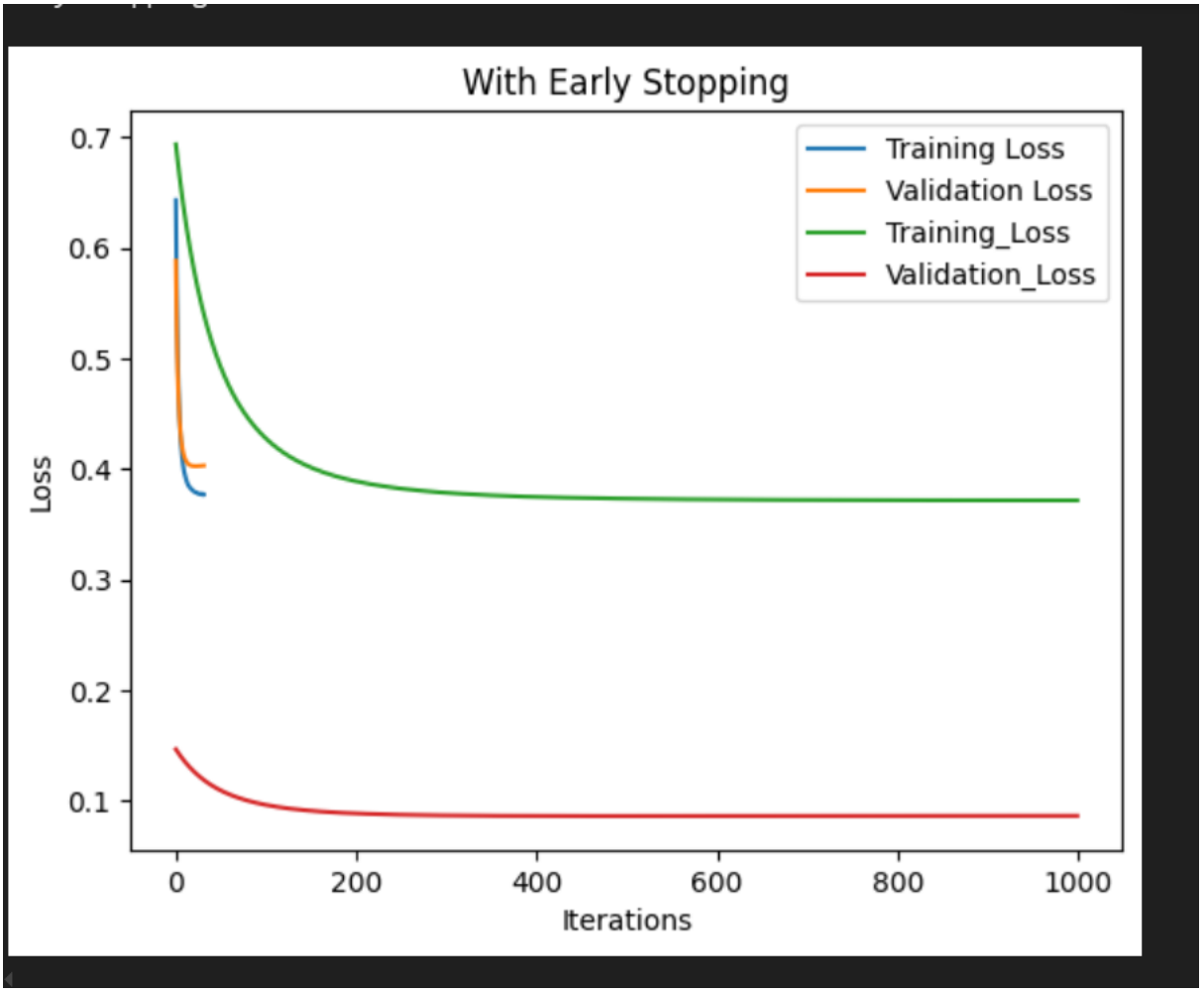
Observations Across the Folds:

Fold No.	Accuracy	Precision	Recall	F1 Score
0	0.6667	0.0	0.0	0.0
1	0.3333	0.0	0.0	0.0
2	0.0000	0.0	0.0	0.0
3	0.3333	0.0	0.0	0.0
4	0.0000	0.0	0.0	0.0



- The average accuracy across the 5 folds is 0.2667, with a high standard deviation of 0.2494. This high variance indicates that the model's performance is inconsistent across different subsets of the data.
- All metrics related to positive class predictions (precision, recall, and F1 score) are consistently zero across all folds. This indicates that the model fails to correctly identify any positive samples in any of the folds.
- The standard deviation of precision, recall, and F1 score is zero across all folds, reflecting that the model's predictions are uniformly poor when it comes to identifying positive cases.
- The reason is again due to sample data provided to us which contains mmajority of data as zero around 85% perc hence the model which is trained on this data is biased towards predicting zero that is no heart disease

(f) (3 marks) Implement early stopping in your best Gradient Descent method to avoid overfitting. Define and use appropriate stopping criteria. Experiment with different learning rates and regularization techniques (L1 and L2). Plot and compare the performance with and without early stopping. Analyze the effect of early stopping on overfitting and generalization.



- The training stopped at iteration 31 with a training loss of approximately 0.40 and a validation loss of 0.59.
- The model avoided overfitting by halting training when the validation loss started to degrade.
- This method showed a balance between training and validation performance, maintaining generalization on unseen data.
- The training continued for 1000 iterations, with the training loss decreasing to below 0.10.
- However, the validation loss stagnated around 0.58 after the first few iterations, indicating that the model was overfitting to the training data.
- The absence of early stopping allowed the model to continue optimizing for the training set at the cost of generalization.