# CSE343
# Machine Learning
Monsoon 2024

Report for Assignment 1

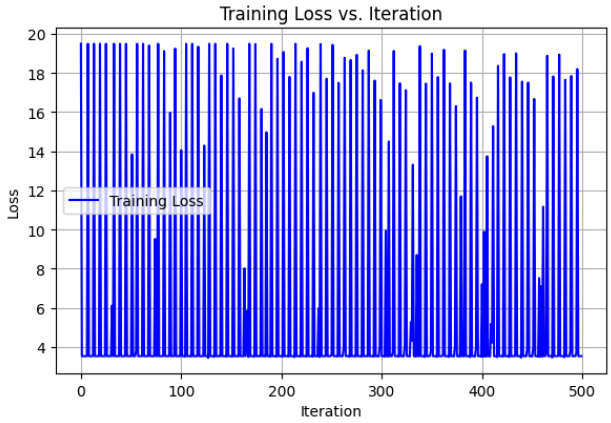
Shamik Sinha - 2022468

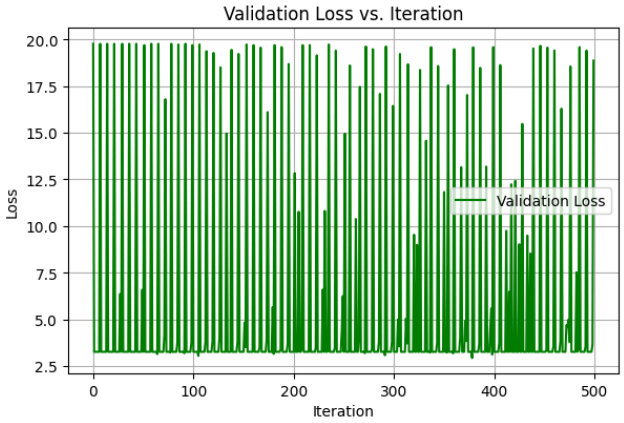
# Section B

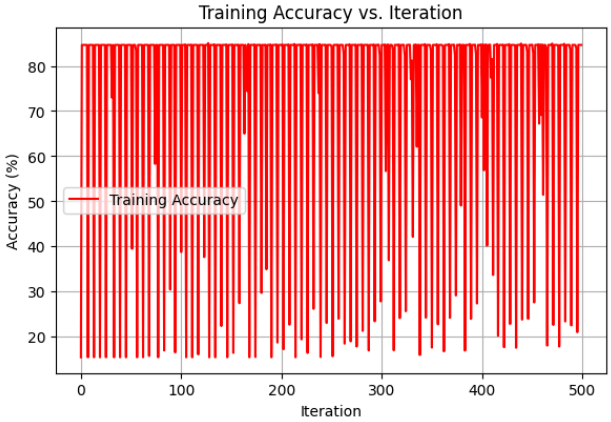
part a.

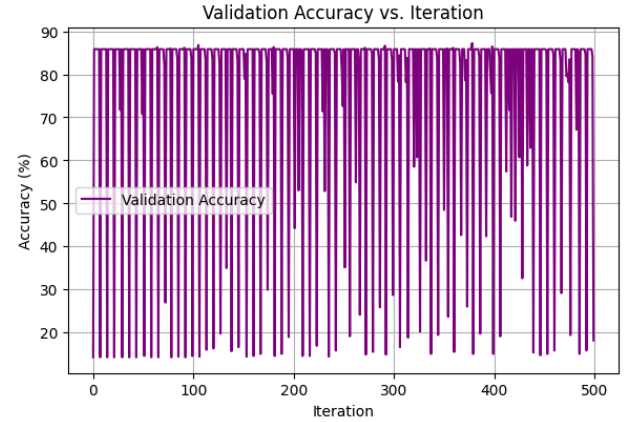
Learning rate = 1
Epochs / Iterations = 500

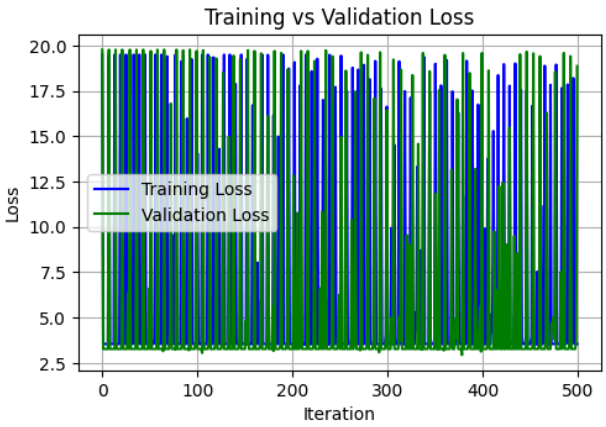
| Training Data Metrics | Validation Data Metrics |
|---|---|
| 1. Confusion Matrix : [[2472 39][ 412 43]]<br>2. True Negative : 2472<br>3. False Positive : 39<br>4. False Negative : 412<br>5. True Positive : 43<br>6. Recall : 0.09451<br>7. Precision : 0.52439<br>8. F1 Score : 0.16015<br>9. Training Accuracy: 0.84794 | 1. Confusion Matrix : [[546 0][ 90 0]]<br>2. True Negative : 546<br>3. False Positive : 0<br>4. False Negative : 90<br>5. True Positive : 0<br>6. Recall : 0.00000<br>7. Precision : 0.00000<br>8. F1 Score : 0.00000<br>9. Validation Accuracy: 0.85849 |

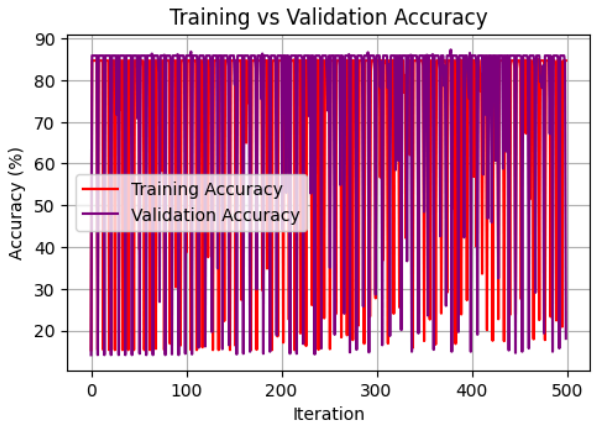


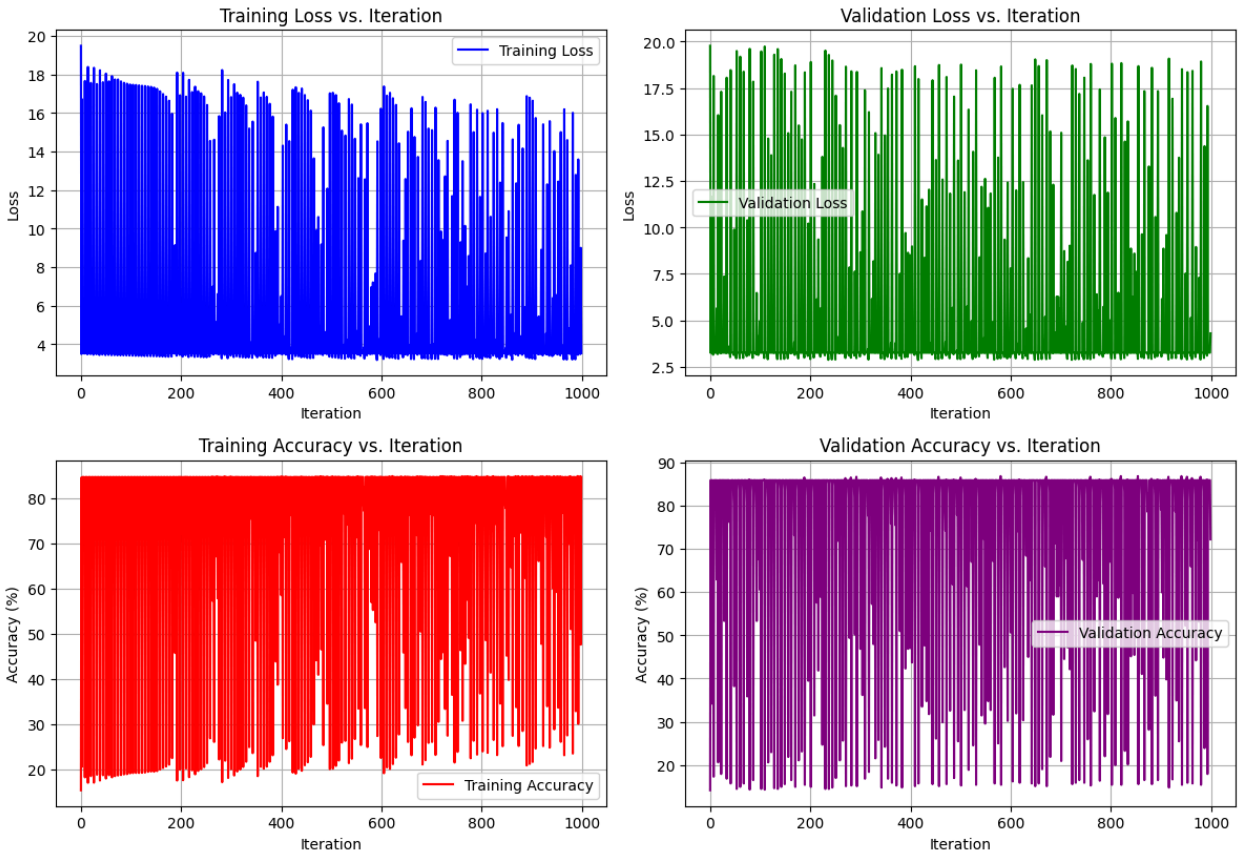



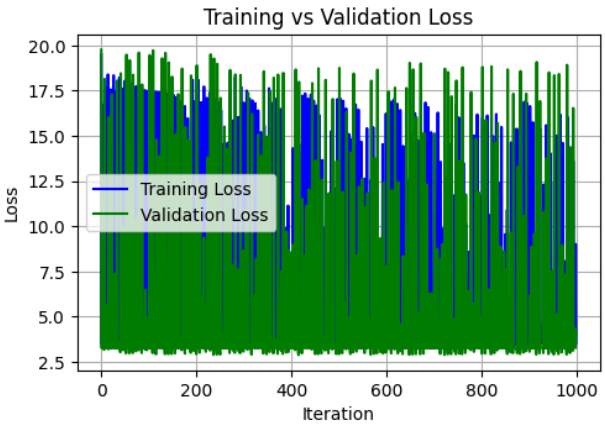

Training vs Validation Loss
Training Loss
Validation Loss
Loss
Iteration
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0
100
200
300
400
500

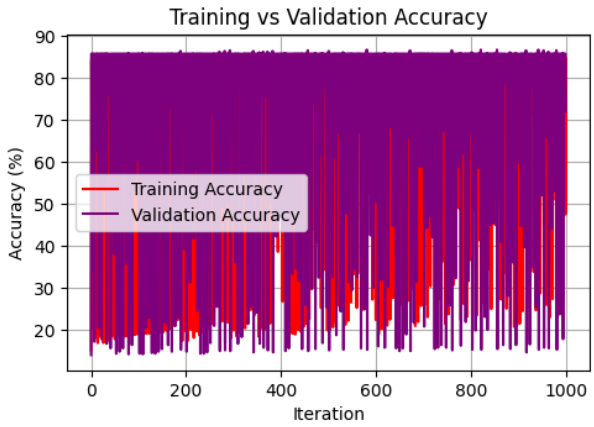

Training vs Validation Accuracy
Training Accuracy
Validation Accuracy
Accuracy (%)
Iteration
90
80
70
60
50
40
30
20
0
100
200
300
400
500

Learning rate = 0.01
Epochs / Iterations = 1000

| Training Data Metrics | Validation Data Metrics |
|---|---|
| 1. Confusion Matrix : [[2511 0][ 455 0]]<br>2. True Negative : 2511<br>3. False Positive : 0<br>4. False Negative : 455<br>5. True Positive : 0<br>6. Recall : 0.00000<br>7. Precision : 0.00000<br>8. F1 Score : 0.00000<br>9. Training Accuracy: 0.84659 | 1. Confusion Matrix : [[546 0][ 90 0]]<br>2. True Negative : 546<br>3. False Positive : 0<br>4. False Negative : 90<br>5. True Positive : 0<br>6. Recall : 0.00000<br>7. Precision : 0.00000<br>8. F1 Score : 0.00000<br>9. Validation Accuracy: 0.85849 |

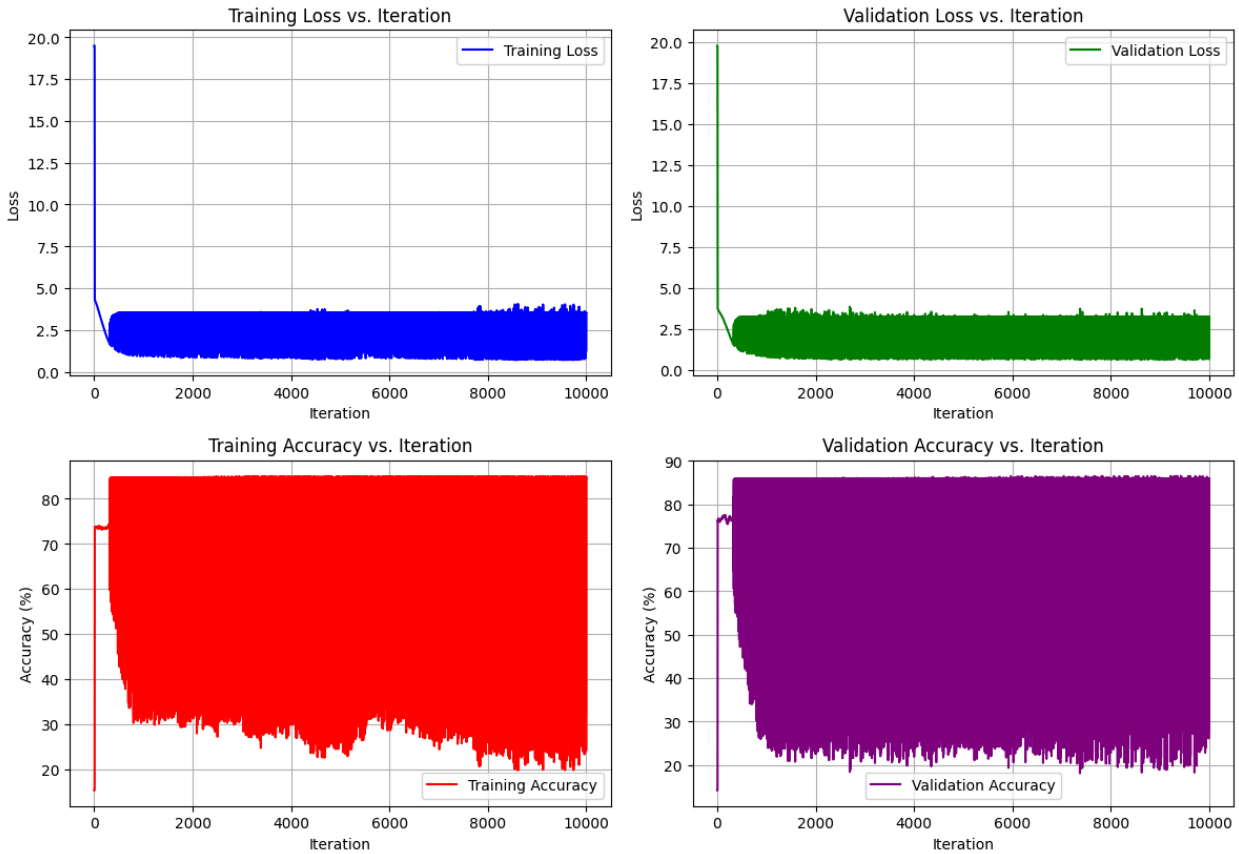


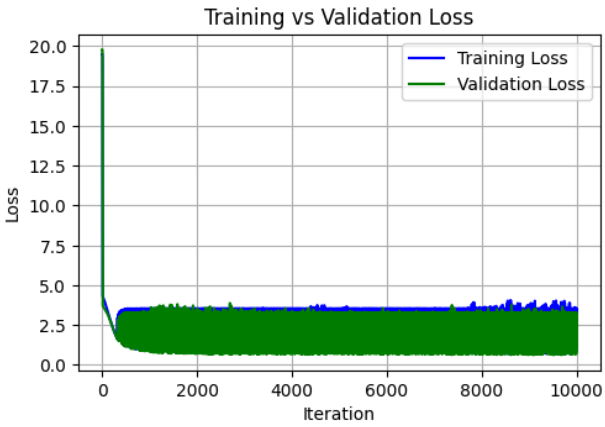

Training vs Validation Loss
Training Loss
Validation Loss
Loss
Iteration
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0
200
400
600
800
1000

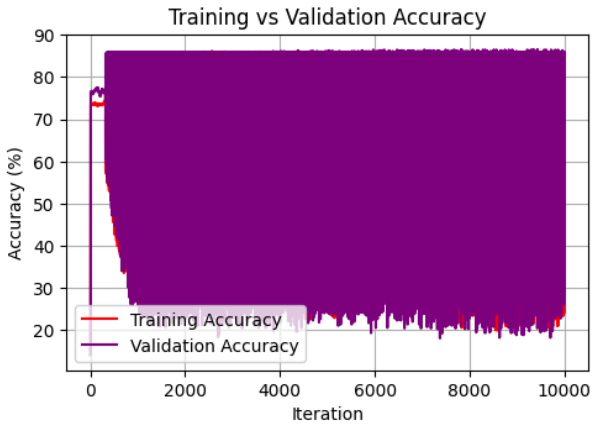

Training vs Validation Accuracy
Training Accuracy
Validation Accuracy
Accuracy (%)
Iteration
90
80
70
60
50
40
30
20
0
200
400
600
800
1000

Learning rate = 0.001
Epochs / Iterations = 10000

| Training Data Metrics | Validation Data Metrics |
|---|---|
| 1. Confusion Matrix : [[2511 0] [ 455 0]]<br>2. True Negative : 2511<br>3. False Positive : 0<br>4. False Negative : 455<br>5. True Positive : 0<br>6. Recall : 0.00000<br>7. Precision : 0.00000<br>8. F1 Score : 0.00000<br>9. Training Accuracy: 0.84659 | 1. Confusion Matrix : [[475 71] [ 59 31]]<br>2. True Negative : 475<br>3. False Positive : 71<br>4. False Negative : 59<br>5. True Positive : 31<br>6. Recall : 0.34444<br>7. Precision : 0.30392<br>8. F1 Score : 0.32292<br>9. Validation Accuracy: 0.79560 |

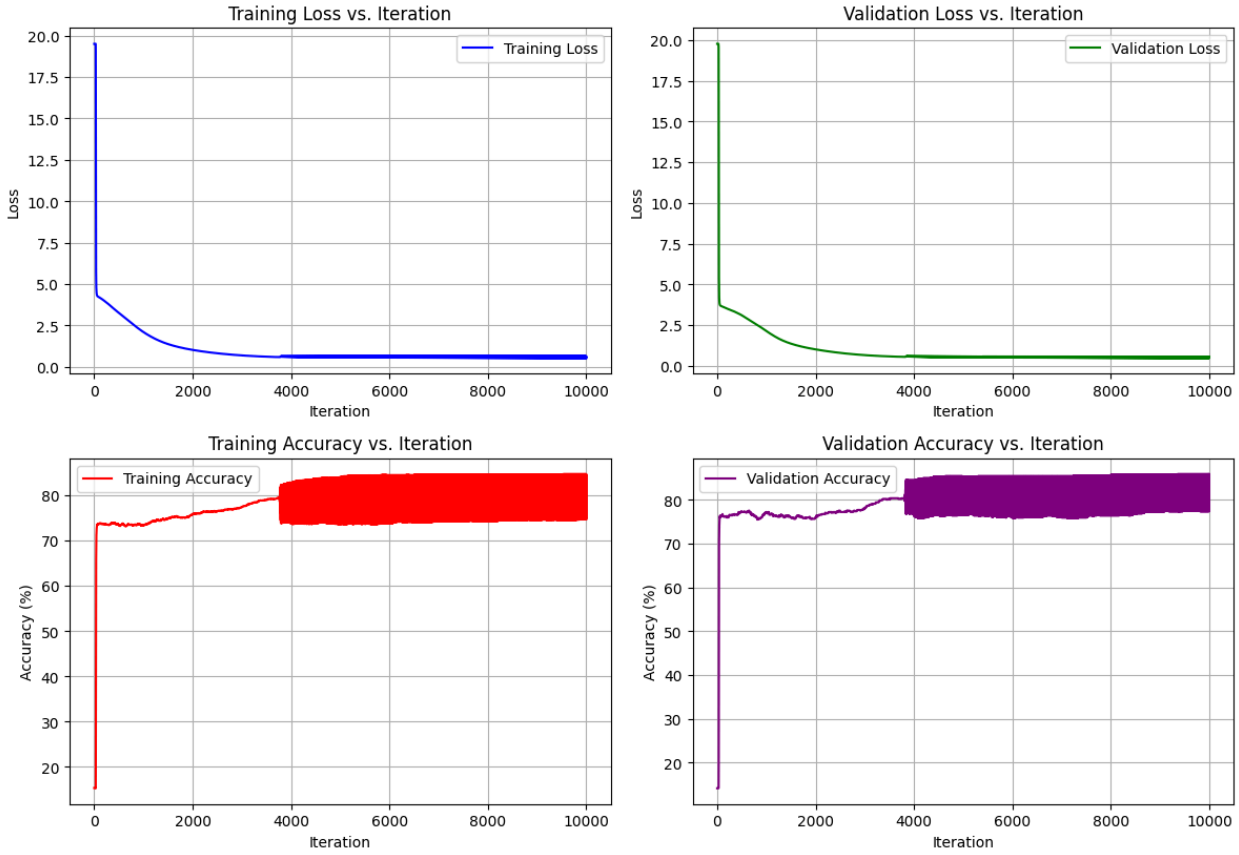


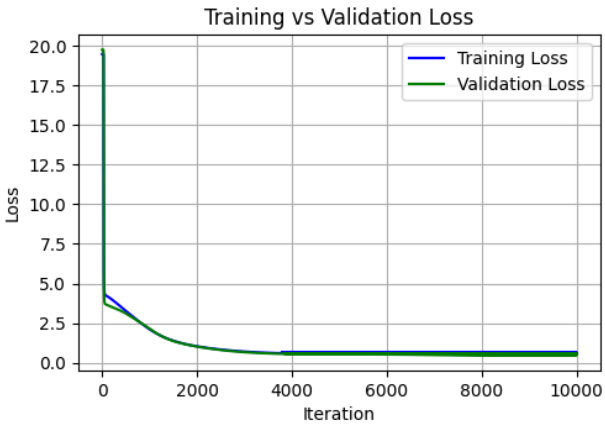

Training vs Validation Loss
Training Loss
Validation Loss
Loss
Iteration
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
0
2000
4000
6000
8000
10000

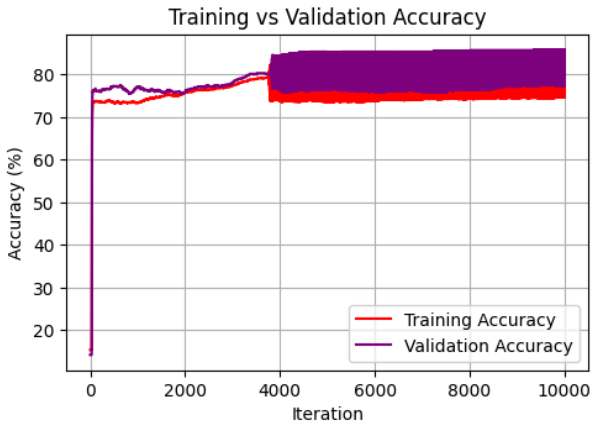

Training vs Validation Accuracy
Training Accuracy
Validation Accuracy
Accuracy (%)
Iteration
20
30
40
50
60
70
80
90
0
2000
4000
6000
8000
10000

Learning rate = 0.00025
Epochs / Iterations = 60000

| Training Data Metrics | Validation Data Metrics |
|---|---|
| 1. Confusion Matrix : [[2511 0][ 455 0]]<br>2. True Negative : 2511<br>3. False Positive : 0<br>4. False Negative : 455<br>5. True Positive : 0<br>6. Recall : 0.00000<br>7. Precision : 0.00000<br>8. F1 Score : 0.00000<br>9. Training Accuracy: 0.84659 | 1. Confusion Matrix : [[475 71] [ 59 31]]<br>2. True Negative : 475<br>3. False Positive : 71<br>4. False Negative : 59<br>5. True Positive : 31<br>6. Recall : 0.34444<br>7. Precision : 0.30392<br>8. F1 Score : 0.32292<br>9. Validation Accuracy: 0.79560 |

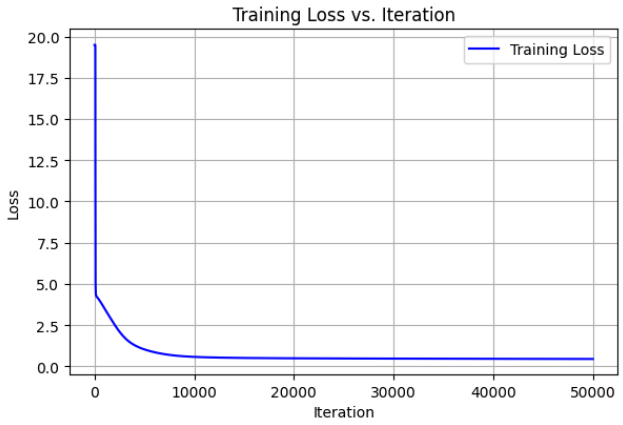


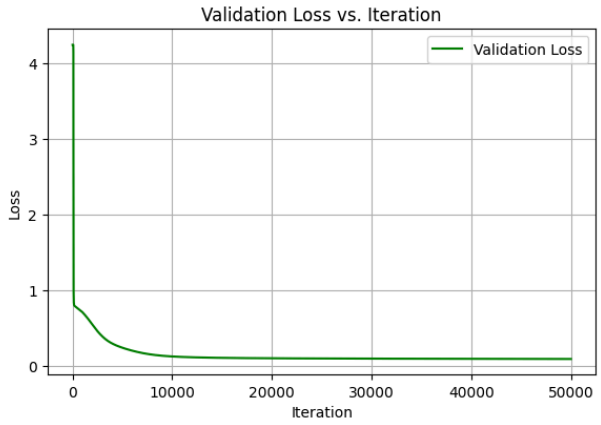

Training vs Validation Loss
Training Loss
Validation Loss
Loss
Iteration
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0
2000
4000
6000
8000
10000

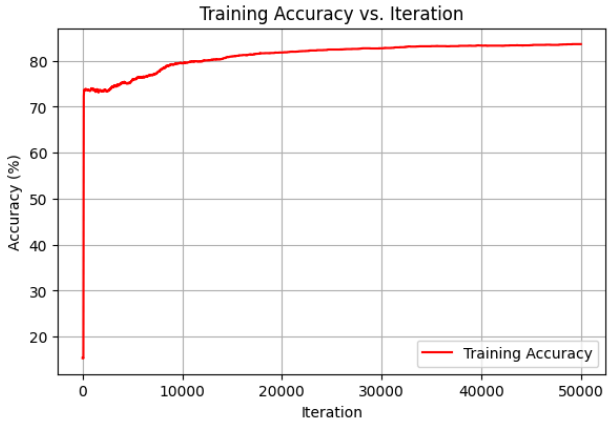

Training vs Validation Accuracy
Training Accuracy
Validation Accuracy
Accuracy (%)
Iteration
80
70
60
50
40
30
20
0
2000
4000
6000
8000
10000

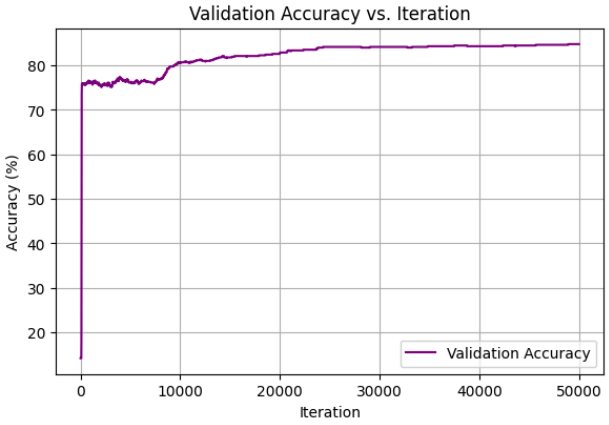
Learning rate = 0.0001
Epochs / Iterations = 60000

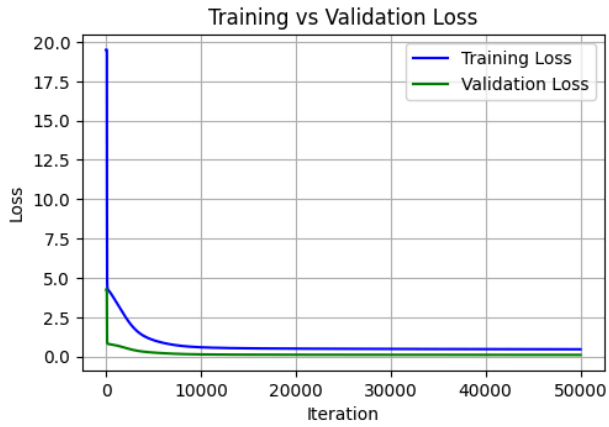
| Training Data Metrics | Validation Data Metrics |
|---|---|
| 10. Confusion Matrix :[[2444, 67] [ 42, 34]]<br>11. True Negative : 2444<br>12. False Positive : 67<br>13. False Negative : 421<br>14. True Positive : 34<br>15. Recall : 0.07473<br>16. Precision : 0.33663<br>17. F1 Score : 0.12230<br>18. Training Accuracy: 0.83546 | 1. Confusion Matrix : [[535, 11] [ 87, 3]]<br>2. True Negative : 535<br>3. False Positive : 11<br>4. False Negative : 87<br>5. True Positive : 3<br>6. Recall : 0.03333<br>7. Precision : 0.21429<br>8. F1 Score : 0.05769<br>9. Validation Accuracy: 0.84591 |

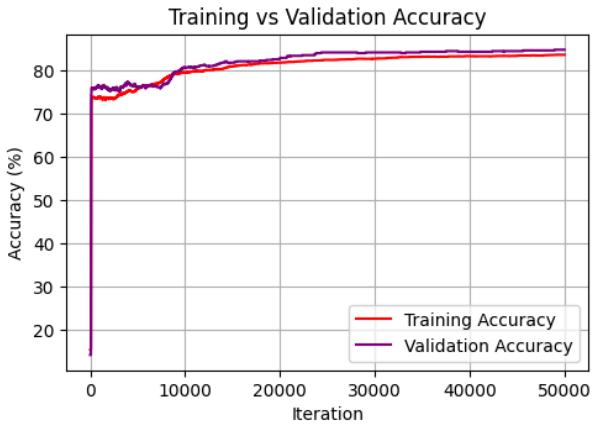





## Conclusion :

Initially the learning rates 1, 0.1, 0.01 were too high and the model was not able to converge properly, however further decrease in learning rate along with an increase in the number of iterations, the model was able to converge smoothly and achieve a good accuracy.

After a certain point, if the validation loss starts rising or validation accuracy stagnates, it may suggest overfitting, where the model is fitting too closely to the training data without improving on new data. Increased iterations thus benefit early learning but must be monitored to avoid overfitting. In the last case for example we can see the validation accuracy starts to stagnate after 20000 to 30000 iterations. Hence we could have stopped training there to prevent overfitting.

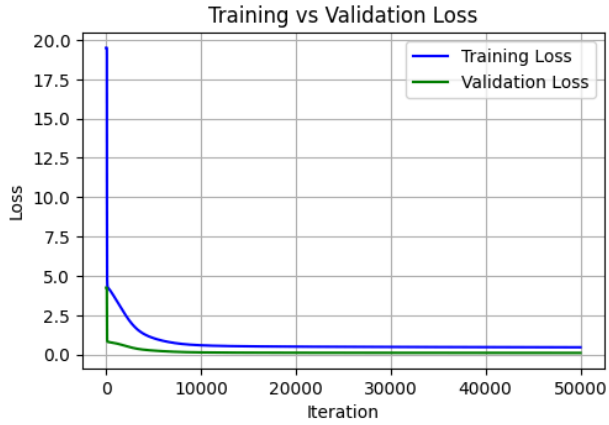
Note on Precision and Recall:

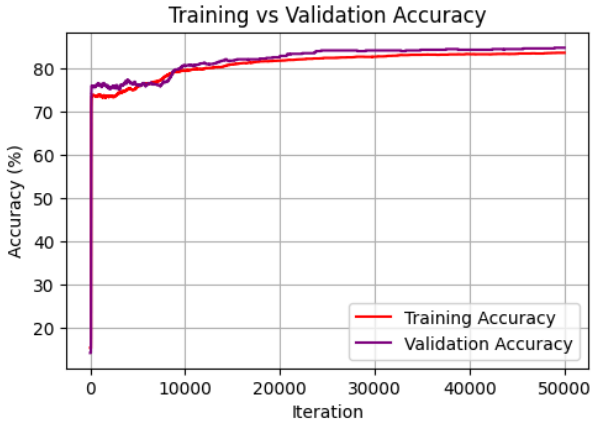
Although the model achieved high accuracy during training and validation after tuning the learning rate and iteration combinations, it was also observed that precision and recall metrics were significantly lower across all observations. This suggests that while the model is performing well in terms of overall accuracy, it may be failing to correctly classify some of the minority or critical classes, leading to an imbalance in its ability to predict positive outcomes. This highlights the need for further investigation, possibly due to a skewed dataset. I will explore this anomaly further to better understand the cause of low precision and recall despite high accuracy.

part b.

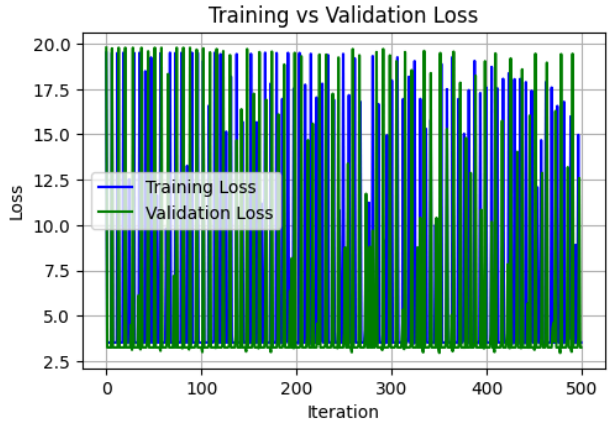
Feature scaling is a technique used to standardize the range of independent variables or features of data. Many machine learning algorithms perform better or converge faster when features are on a similar scale. We try to demonstrate two methods below. One without scaling and another where we utilize min-max scaling. We also plot the loss vs iteration for both training and validation sets in both the cases along with the accuracy to better understand the concept.

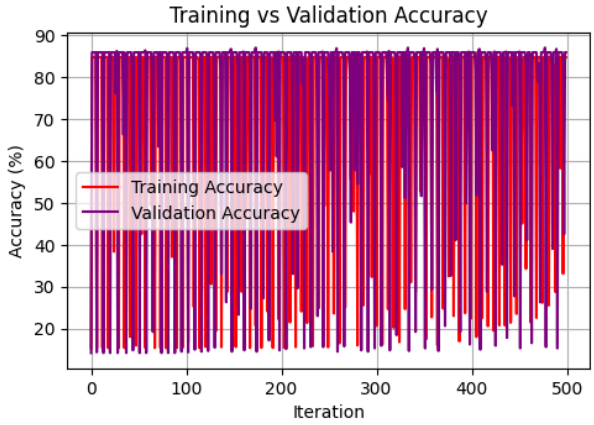
No Scaling :
Learning rate = 0.0001
Number of epochs/iterations = 50000



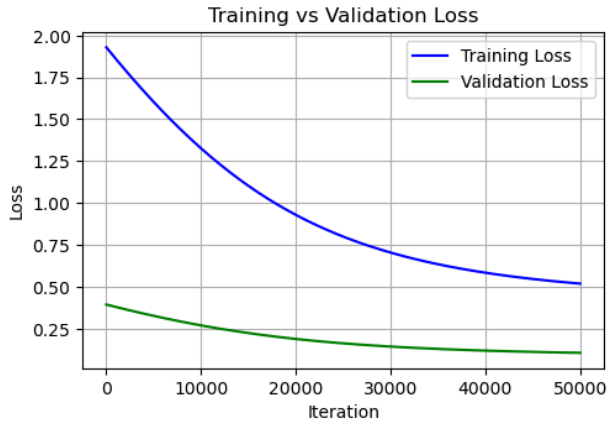



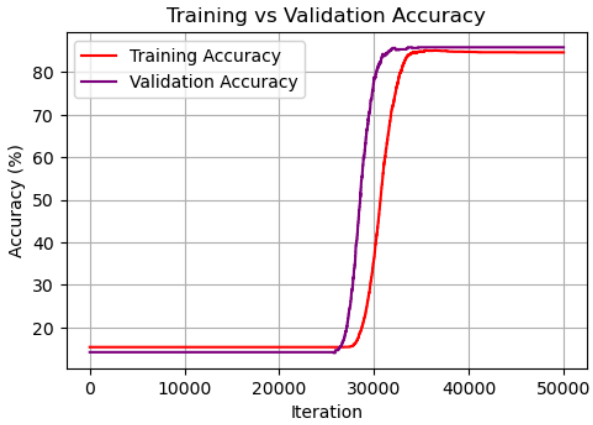

No Scaling :
Learning rate = 0.05
Number of epochs/iterations = 500



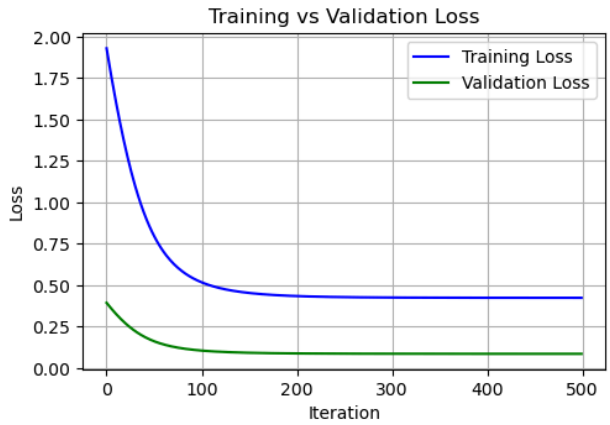



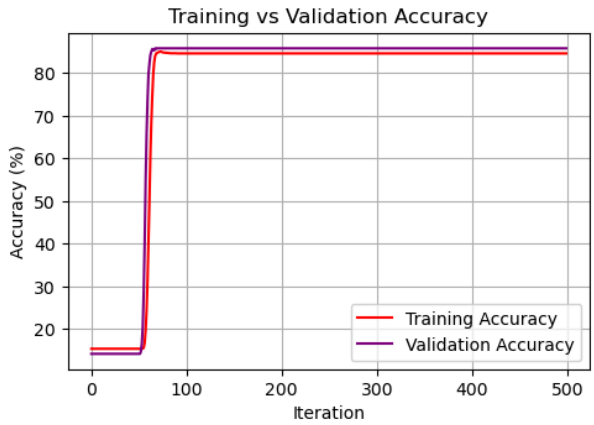
Min-Max Scaling :
Learning rate = 0.0001
Number of epochs/iterations = 50000




Min-Max Scaling :
Learning rate = 0.05
Number of epochs/iterations = 500




**Observation:** Impact of feature scaling on convergence of our model

**Why Both Cases (Scaling and No Scaling) Converge at the Same Point with a Much Lower Learning Rate (0.0001):**

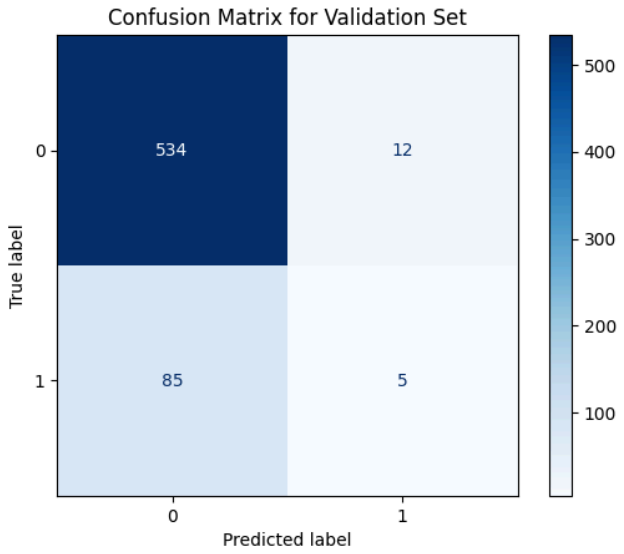
When a lower learning rate of 0.0001 is used, both the scaled and unscaled models converge to approximately the same accuracy after a large number of iterations. This is because the small learning rate results in gradual weight updates during each iteration, ensuring that no feature dominates the learning process, even when features are not scaled. As a result, the optimization process becomes stable and both models follow a smooth trajectory toward a local minimum, allowing them to reach similar convergence

points. The lower learning rate helps avoid large jumps in gradient descent, leading to steady improvement in both cases, regardless of feature scaling.

**Why Min-Max Scaling Converges Faster with a Higher Learning Rate (0.05) When It Didn't in the No Scaling Case:**

With a higher learning rate of 0.05, Min-Max scaling allows for faster and more stable convergence compared to the unscaled case. Min-Max scaling standardizes the feature values, ensuring that all features are within the same range, which prevents any large feature values from causing disproportionately large updates during gradient descent. This enables the model to make efficient, stable progress toward convergence. In contrast, without scaling, the larger magnitude of certain features leads to erratic updates with the high learning rate, causing the model to diverge or oscillate without reaching a stable solution. Therefore, scaling ensures that the higher learning rate accelerates convergence while maintaining stability.

part c.

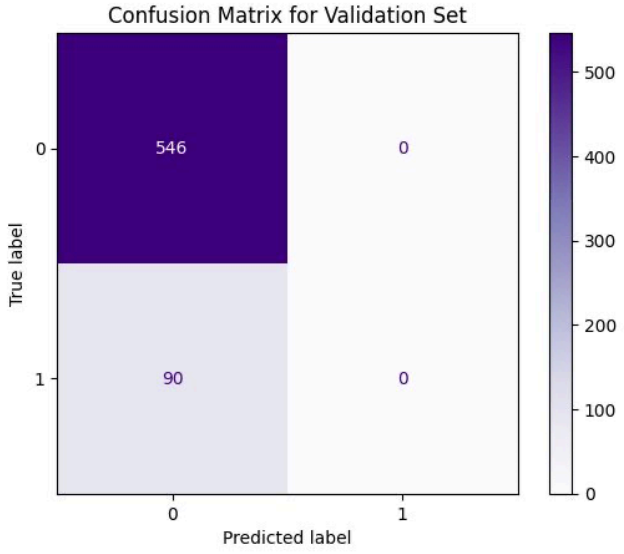



Confusion matrix for the validation dataset: [TN, FP], [FN, TP]] = **[[534 12] [ 85 5]]**
Model without scaling,
**Validation Accuracy = 83.58057990559676 %**

Classification threshold : 0.5
LR = 0.0001
Epochs = 50000

**Precision: 0.2941**
**Recall: 0.0556**
**F1 Score: 0.0935**
**ROC-AUC Score: 0.6248**

**Observation**: The model achieved a high accuracy of 83.58%, but significantly lower precision (29.41%) and recall (5.56%), resulting in a low F1 score (9.35%). This discrepancy suggests that the model is biased towards the majority class, as it struggles to correctly identify minority class instances, likely due to class imbalance. The moderate ROC-AUC score (0.6248) indicates limited ability to distinguish between classes, further confirming the impact of imbalance on the model's performance.




Confusion matrix for the validation dataset: [TN, FP], [FN, TP]] = **[[546 0] [ 90 0]]**
Model with min max scaling,
**Validation Accuracy = 84.65947403910991 %**

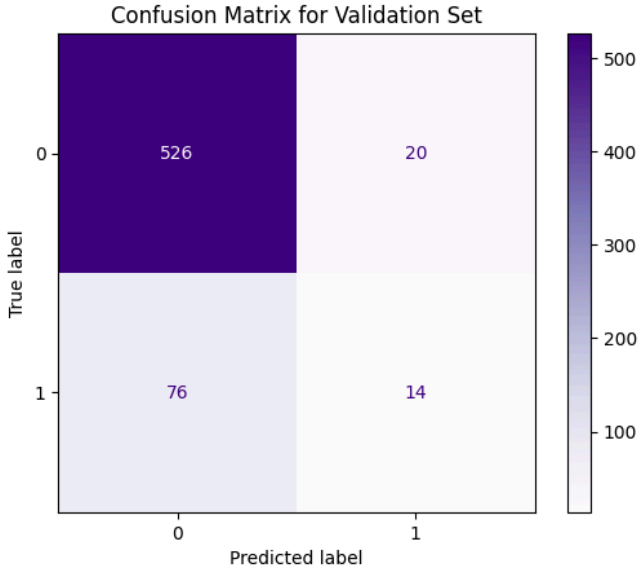
Classification threshold : 0.5
LR = 0.05
Epochs = 500

**Precision: 0.0000**
**Recall: 0.0000**
**F1 Score: 0.0000**
**ROC-AUC Score: 0.7059**

**Observation**: Despite achieving a validation accuracy of 84.66%, the model shows zero precision, recall, and F1 score. This is because the model predicts the negative class for all instances, as seen in the confusion matrix where no positive predictions were made (TP = 0). This suggests the model is heavily biased toward the majority (negative) class. The ROC-AUC score of 0.7059 indicates the model has some

ability to distinguish between the classes, but this is not reflected in the classification performance, likely due to class imbalance and an unsuitable threshold.

A possible way to improve the model's performance in detecting positive instances is by adjusting the classification threshold. The current threshold of 0.5 causes the model to classify all instances as negative, as indicated by the confusion matrix. By lowering the threshold, the model may begin predicting more positive cases, thereby improving recall and F1 score.




Confusion matrix for the validation dataset: [TN, FP], [FN, TP]] = **[[526 20] [ 76 14]]**
Model with min max scaling,
**Validation Accuracy = 66.9925826028321 %**

Classification threshold : 0.16
LR = 0.05
Epochs = 500

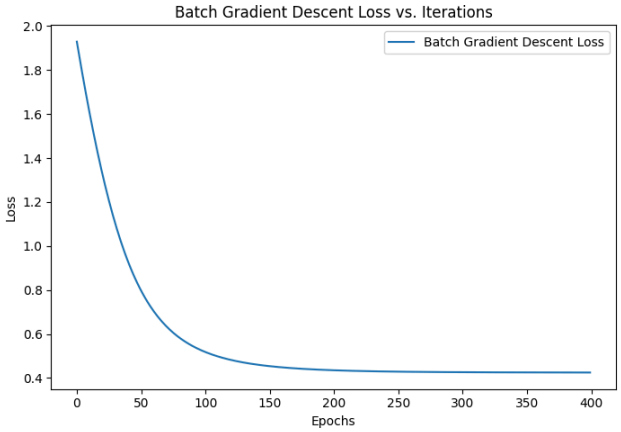
**Precision: 0.4118**
**Recall: 0.1556**
**F1 Score: 0.2258**
**ROC-AUC Score: 0.6934**

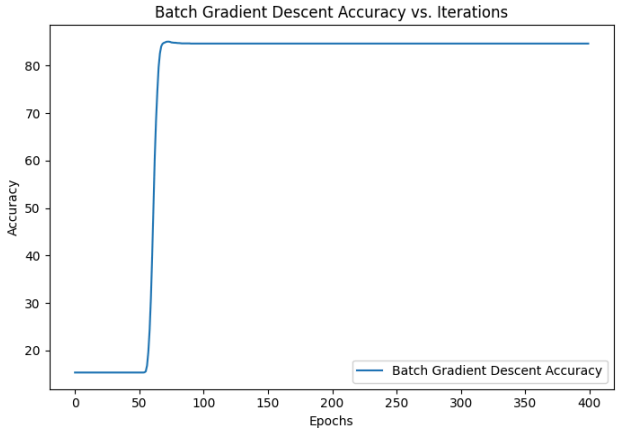
**Observation:** At a lowered threshold of 0.16, the model improved its ability to detect positive cases, reflected in the precision (41.18%) and recall (15.56%), though at the cost of validation accuracy (66.99%). The confusion matrix shows that while more true positives were identified, a significant number of false negatives (76) and false positives (20) remain. The F1 score (22.58%) is a balance between precision and recall, and the ROC-AUC score (0.6934) suggests moderate class discrimination.

The decrease in accuracy to 66.99% is attributed to an increase in false positives (20) and false negatives (76), which affects the total count of correct predictions. Lowering the threshold to 0.16 improved recall (15.56%) and precision (41.18%), allowing the model to detect more true positives, though at the expense of more false positives. *In the context of diagnosing heart disease, identifying more true positives (potentially missed cases) is crucial, even if it means accepting some false positives. This approach might be more beneficial in ensuring that potential cases are not overlooked.*

part d.

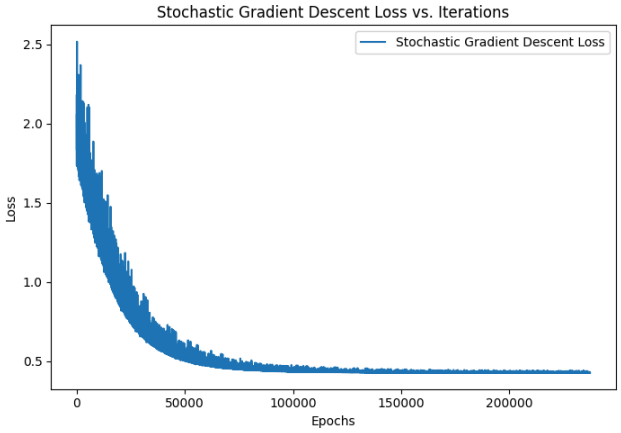
Batch Gradient Descent

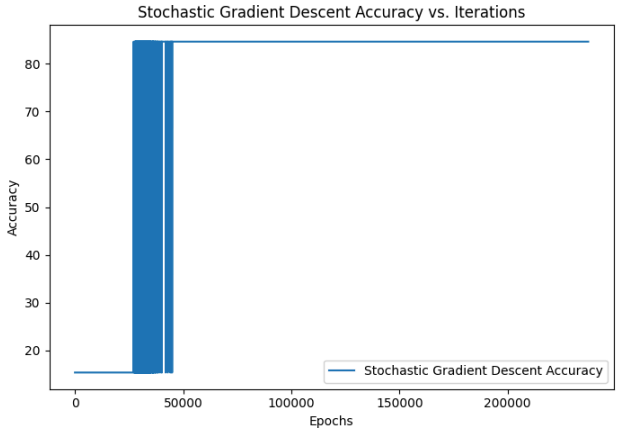


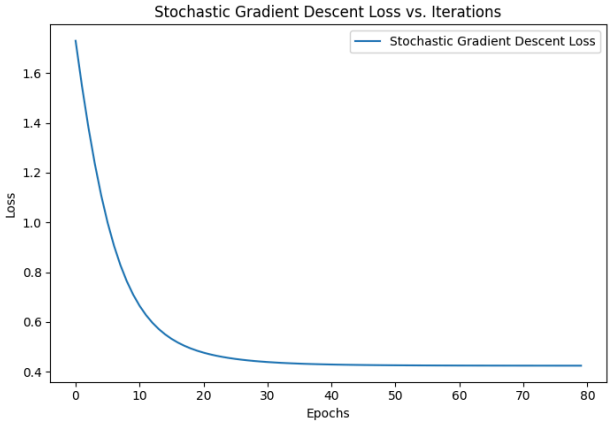

Stochastic Gradient Descent

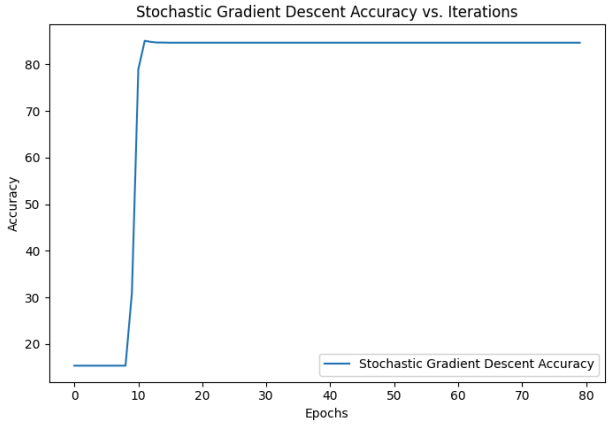
1. Taking observations after every random selection of data points.



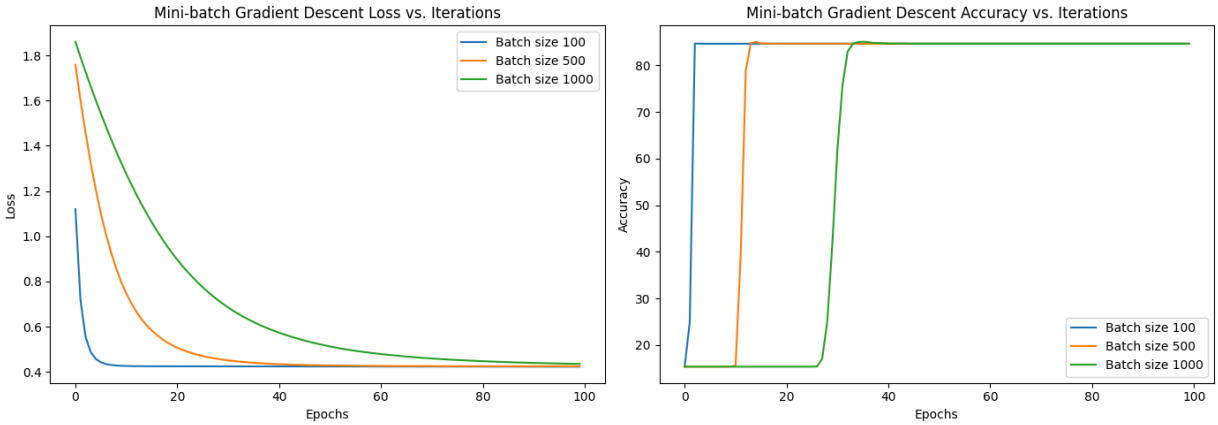




2. Taking observations once every epoch.



Mini Batch Gradient Descent




**Observation:** The trade-offs in terms of convergence speed and stability between these methods.

**Batch Gradient Descent:**

1. Loss vs. Iterations: The batch gradient descent exhibits a smooth, gradual decrease in loss over a larger number of iterations (around 400). The convergence is slow but stable, with no significant fluctuations.
2. Accuracy vs. Iterations: The accuracy rises sharply after a certain number of iterations (~100) and then stabilizes at a high value (84.58%) for the remaining iterations.

**Trade-offs:**

1. Speed: Slow convergence because batch gradient descent updates weights only after processing the entire dataset, which takes more time per iteration.
2. Stability: Very stable due to the full dataset being used for each update, leading to fewer fluctuations in both loss and accuracy.

**Stochastic Gradient Descent (SGD):**

1. Loss vs. Iterations: The loss decreases much faster than batch gradient descent but with more noise and fluctuations. Convergence is achieved after a much smaller number of iterations (~80), but the path to convergence is less smooth.
2. Accuracy vs. Iterations: Similar to loss, accuracy improves rapidly, though there are more fluctuations before it stabilizes at a similar high accuracy (84.58%).

**Trade-offs:**

1. Speed: Faster convergence than batch gradient descent since weights are updated after each individual sample. This makes SGD computationally faster, requiring fewer iterations to reach convergence.
2. Stability: Less stable due to the noisy nature of updates. The accuracy fluctuates more before stabilizing.

**Mini-Batch Gradient Descent:**

1. Loss vs. Iterations: For batch sizes of 100, 500, and 1000, the loss decreases rapidly with batch size 100 converging the fastest, followed by 500 and 1000. The convergence path is smoother compared to SGD but not as smooth as batch gradient descent.
2. Accuracy vs. Iterations: The accuracy also rises faster with smaller batch sizes (100), stabilizing at a high accuracy value (84.48%). Larger batch sizes take longer to converge.

**Trade-offs:**

1. Speed: Mini-batch gradient descent offers a good trade-off between speed and stability. Smaller batch sizes (100) lead to faster convergence, while larger batch sizes behave more like batch gradient descent and take more time to converge.
2. Stability: Smaller batches introduce some fluctuations (though less than SGD), while larger batches are more stable but slower.

**Overall Comparison:**

- Convergence Speed: Stochastic gradient descent converges the fastest but is prone to instability. Mini-batch gradient descent with smaller batch sizes provides a middle ground, offering fast convergence with moderate stability. Batch gradient descent is the slowest but provides the most stable updates.
- Stability: Batch gradient descent is the most stable as it uses the entire dataset for each update. Mini-batch gradient descent provides stability while still being faster than batch gradient descent. Stochastic gradient descent sacrifices stability for speed, leading to noisier updates.

part e.

## K-Fold Cross-Validation Results (k=5)

The performance of the model was evaluated using 5-fold cross-validation, and the results, including the mean and standard deviation for each metric, are as follows:

- Mean Loss: 16.0936 (± 6.3928)
- Mean Accuracy: 0.3011 (± 0.2776)
- Mean Precision: 0.2404 (± 0.1749)
- Mean Recall: 0.8322 (± 0.3270)
- Mean F1 Score: 0.2669 (± 0.0134)

**Across all the folds, I got the following readings:**

- Fold 1: Loss = 19.1656, Precision = 0.1677, Recall = 1.0000, F1 Score = 0.2872
- Fold 2: Loss = 18.9209, Precision = 0.1411, Recall = 0.9828, F1 Score = 0.2468
- Fold 3: Loss = 19.4918, Precision = 0.1535, Recall = 1.0000, F1 Score = 0.2661
- Fold 4: Loss = 3.3166, Precision = 0.5897, Recall = 0.1783, F1 Score = 0.2738
- Fold 5 : Accuracy = 0.1499, Precision = 0.1499, Recall = 1.0000, F1 Score = 0.2608

**Performance Discussion:**

**Inconsistent Model Performance:** The model shows a high variance in performance across the folds, as evidenced by the standard deviations in accuracy and precision. Accuracy varies significantly, with one fold (Fold 4) achieving 85.60%, while others are much lower (~15-18%). This suggests that the model struggles to generalize well across different data subsets and may be overfitting to specific folds.

**High Recall, Low Precision Despite a Lower Threshold:** To account for the seriousness of false negatives in this problem, a lower classification threshold of 0.16 was chosen. The aim was to capture as many true positives as possible, which led to a high mean recall of 83.22%. However, despite this adjustment, the model's precision remains low (24.04%), indicating that a large proportion of the predicted positives are incorrect (false positives). This imbalance could be due to class imbalance in the data or the inability of the model to distinguish between classes effectively.

The reason for the poor improvement in precision, even with a lower threshold, likely stems from the fact that while the model becomes more sensitive and captures more true positives, it is not robust enough to filter out false positives. Lowering the threshold increases recall but does not enhance the model’s ability to discriminate between true positives and false positives, thus failing to significantly improve precision.

**Stability of the F1 Score:** The F1 score, which balances precision and recall, is consistently low across the folds, with a mean of 26.69% and a low standard deviation (± 0.0134). This stability suggests that while the precision-recall trade-off is relatively consistent, the overall performance remains poor.

Model Instability: The high variance in metrics like accuracy and precision indicates that the model's performance is unstable and likely dependent on the data distribution in each fold.

part f.

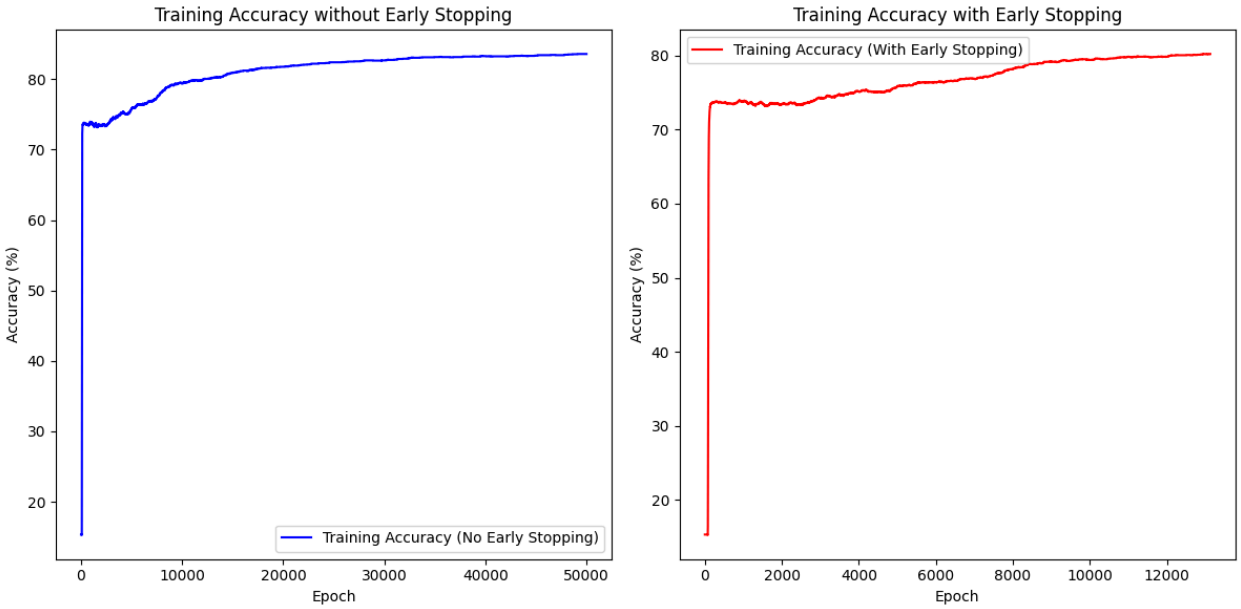
## Early Stopping Results without L1 and L2 regularization:

With early stopping configured using a patience of 10 epochs and a tolerance of 1e-3, training progressed as follows: Initially the model was trained on train set data and then tested on validation set data. The model's cost, during training, decreased significantly from 19.49 at epoch 0 to 0.45 by epoch 45,000, while accuracy improved from 15.34% to 83.34%. Early stopping was triggered at epoch 13,130, when the cost plateaued at 0.53, indicating minimal improvement beyond this point.

**Before stopping early, the training accuracy peaked at 84.75%**.

**After early stopping, the training accuracy slightly decreased to 81.13%.**

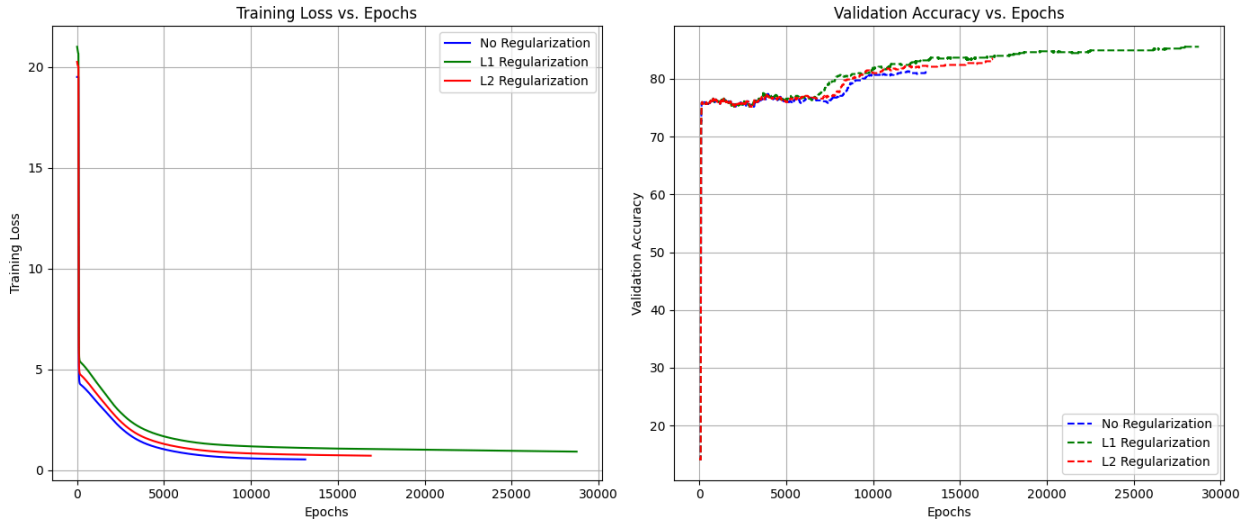
This reduction in training accuracy after early stopping suggests that halting training prevented overfitting by stopping further training once the model's performance on the set began to stabilize. The early stopping mechanism effectively balanced the model's ability to fit the training data while maintaining its generalization capability, ensuring that it did not become overly tuned to the training data at the expense of its performance on unseen data.



L1 and L2 regularization with different learning rates:

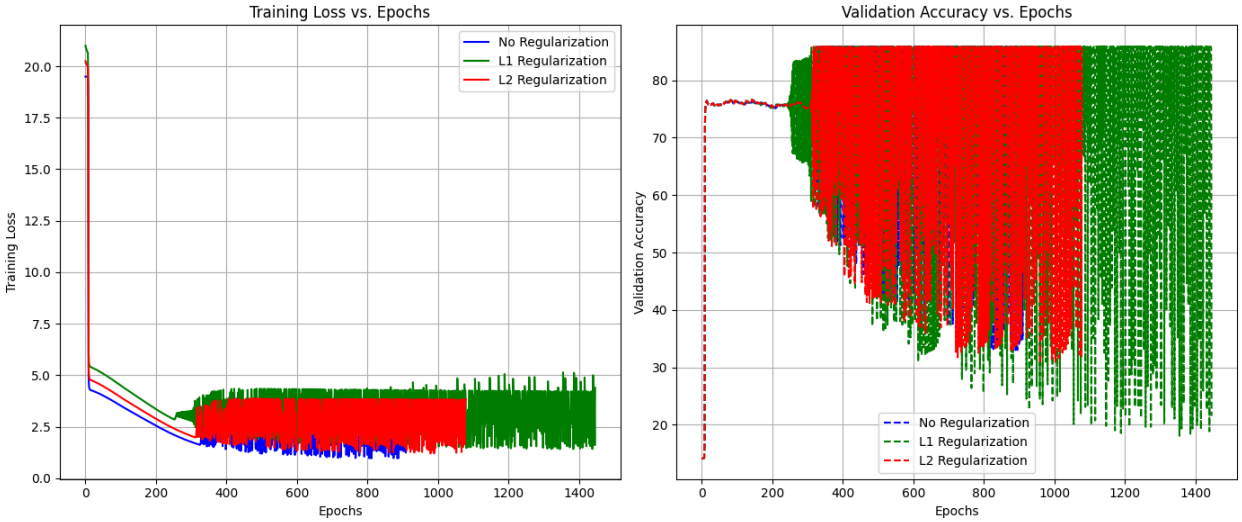
Early stopping cases:

**Learning rate = 0.0001**



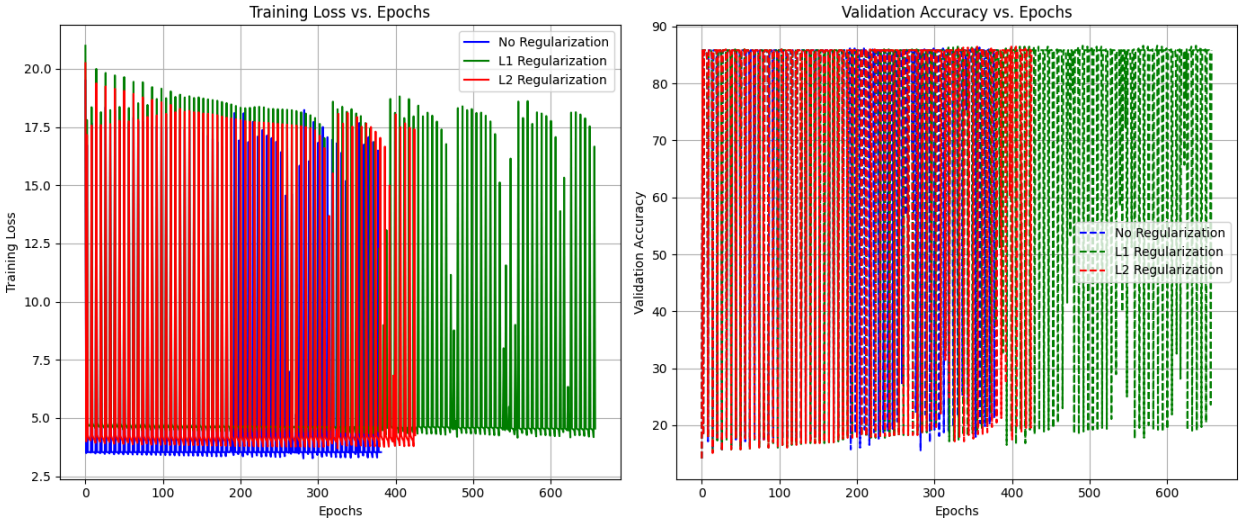

In this experiment, the model was trained with early stopping across three scenarios: without regularization, with L1 regularization, and with L2 regularization. The results indicate that applying regularization improves both training stability and validation performance. Without regularization, early stopping occurred at epoch 13,130 with a validation accuracy of 81.13%. Introducing L1 regularization extended training to epoch 28,748, achieving a higher validation accuracy of 85.53%. L2 regularization triggered early stopping at epoch 16,900, yielding a validation accuracy of 83.49%. This demonstrates that L1 regularization provided the best generalization, likely due to its feature selection property, while L2 regularization also improved performance but to a lesser extent.

**Learning rate = 0.001**




The erratic behavior in the graphs, with significant fluctuations and lack of convergence, suggests that the learning rate of 0.001 might be too high, causing the model to overshoot during optimization. This is evident from the sharp jumps in validation accuracy, particularly for models with L1 and L2 regularization. The regularization strengths may also be too strong, hindering proper convergence.

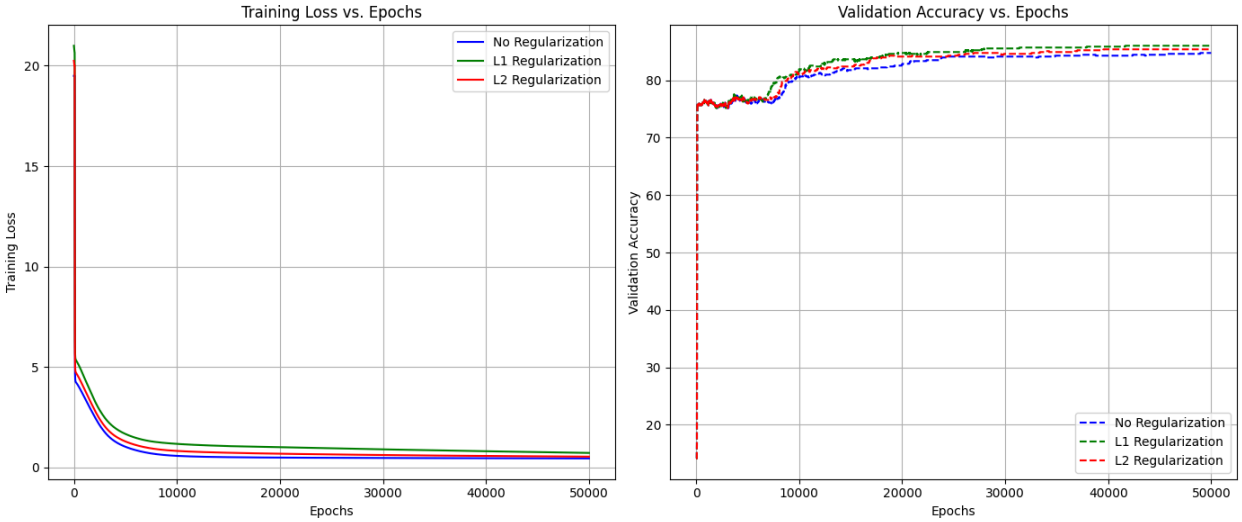
**Learning rate = 0.01**




With the higher learning rate of 0.01, the training loss and validation accuracy graphs show extreme instability, with high-frequency oscillations and no sign of convergence. This suggests that the learning rate is too large, causing the model to constantly overshoot the optimal point during training.

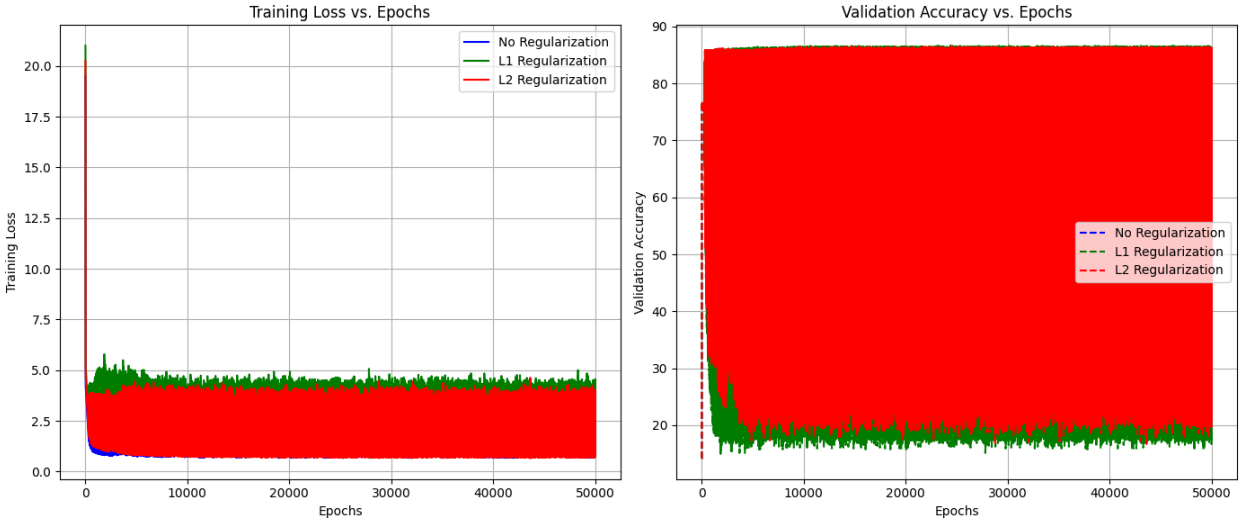
No Early Stopping Results with L1 and L2 regularization:

**Learning rate = 0.0001**




With no early stopping algorithm in place, both regularization techniques failed to reduce overfitting or enhance generalization, as evidenced by the similar training loss and validation accuracy curves. Early stopping therefore proved to be a more effective approach to prevent overfitting in this case.

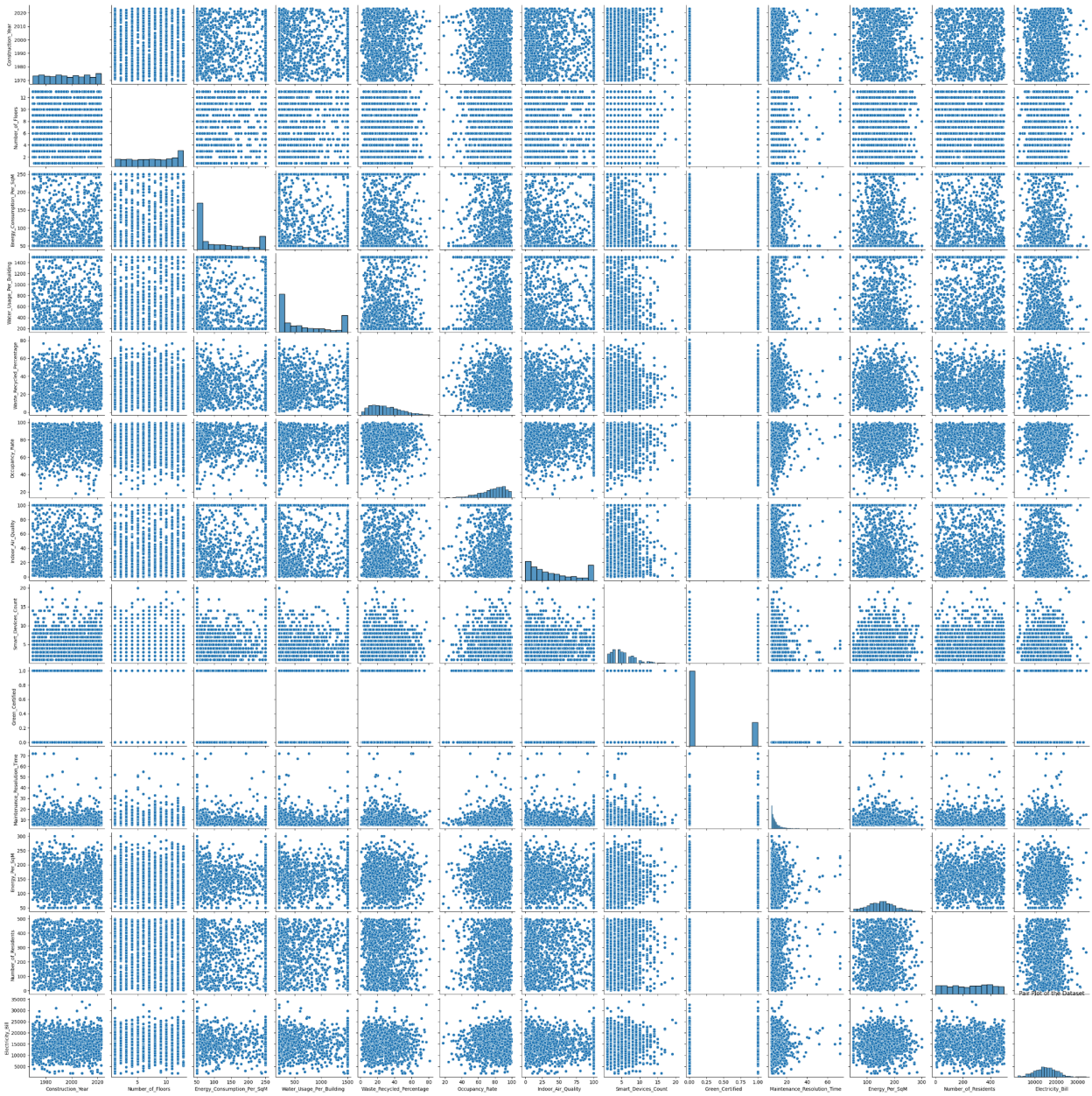
**Learning rate = 0.001**



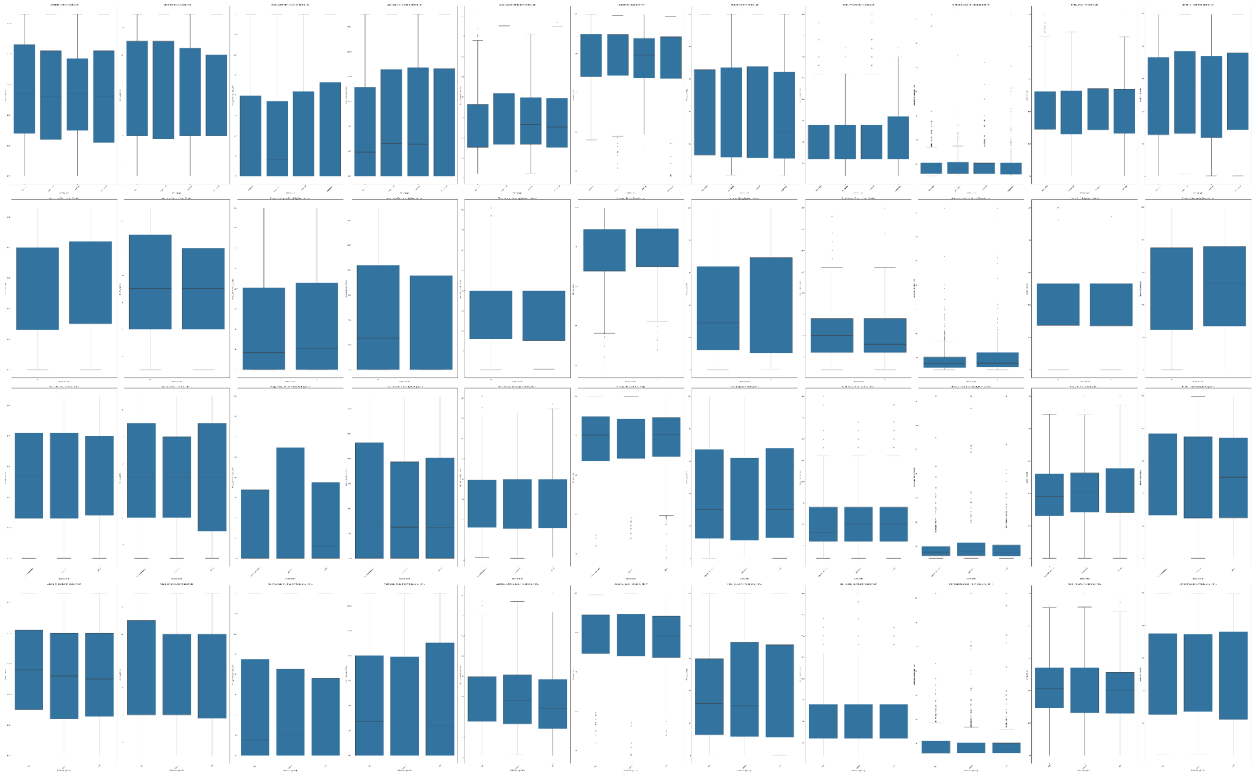
# Section C

part a.

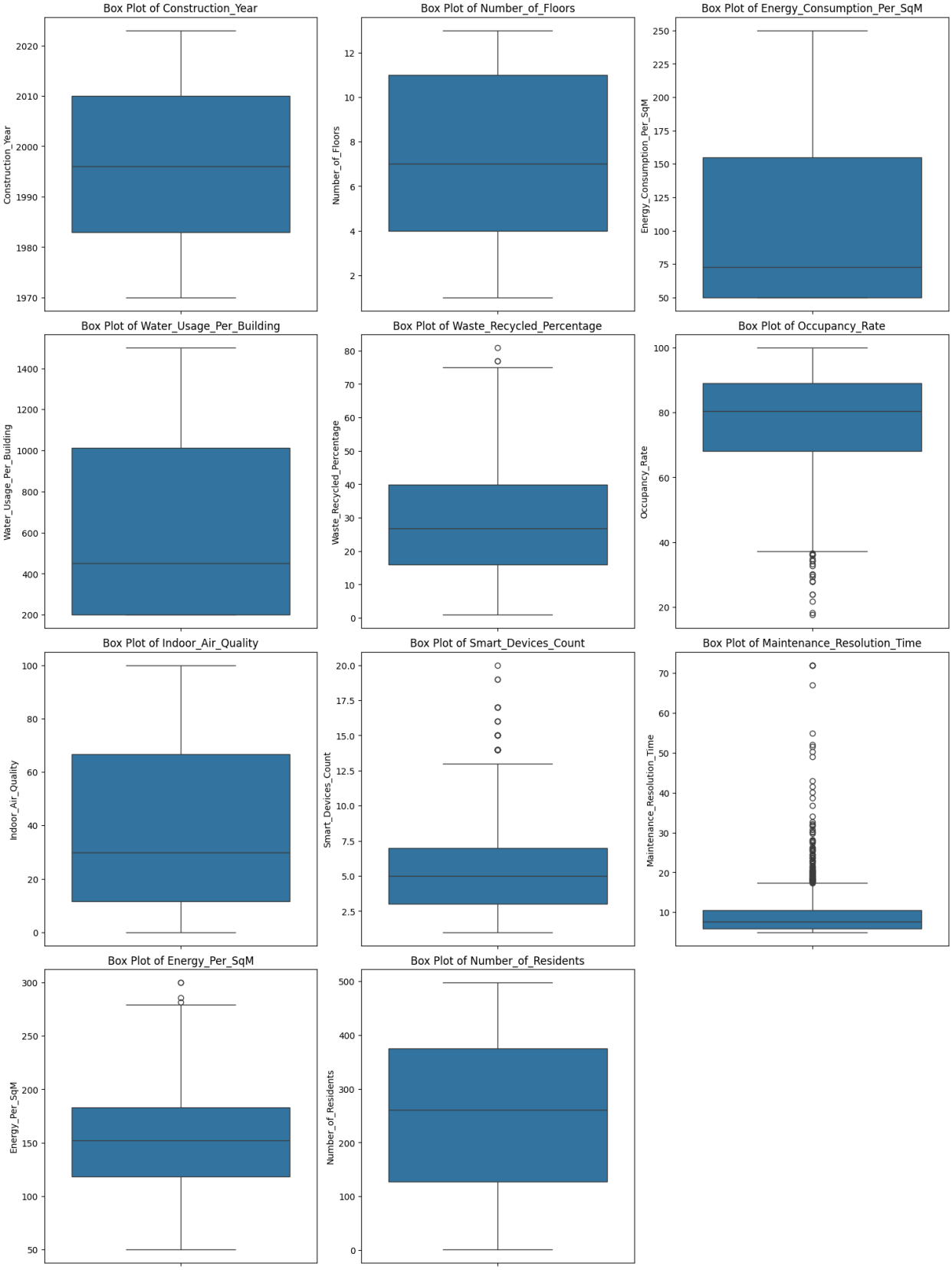
Pair Plot



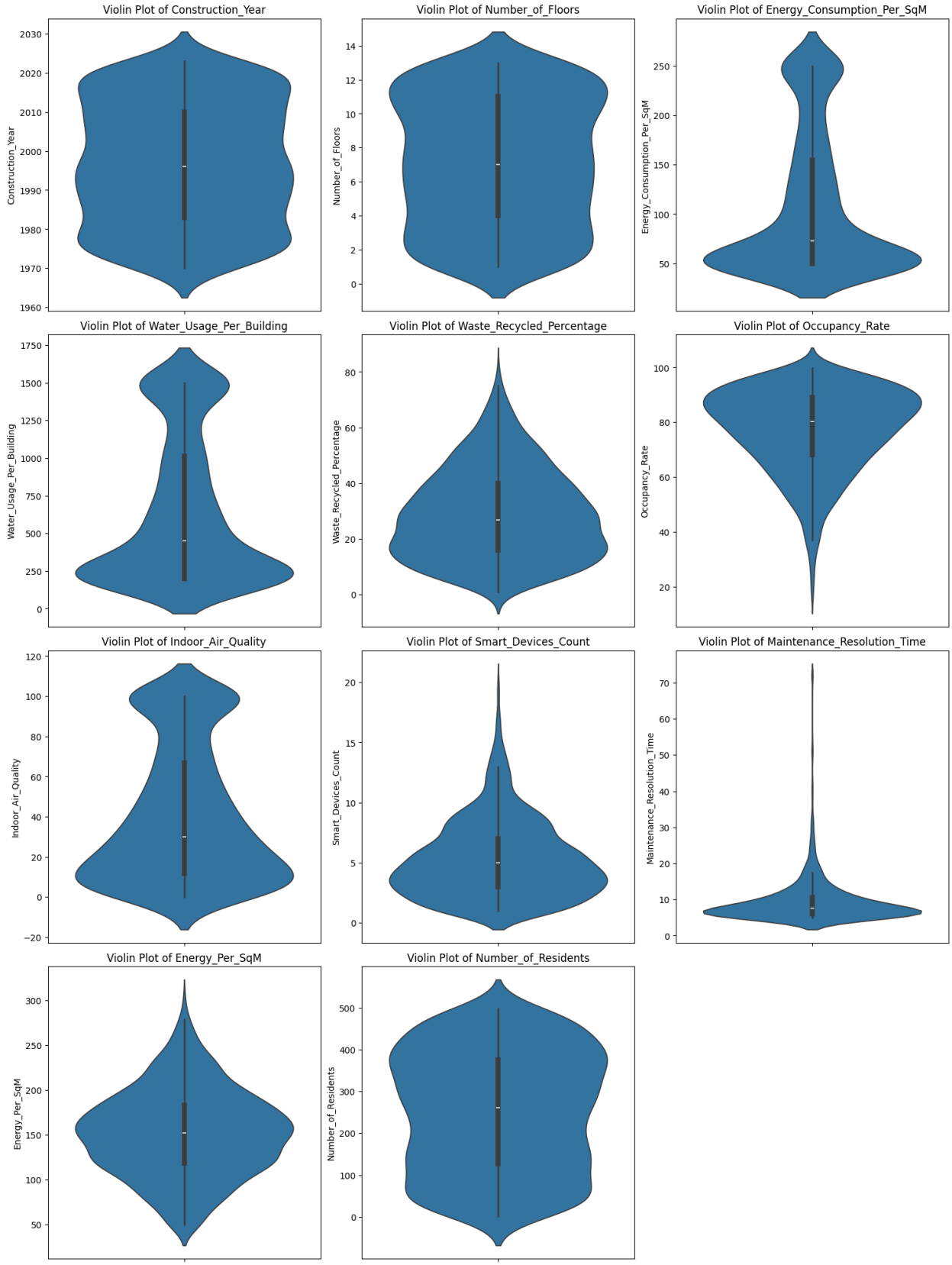
Box Plot



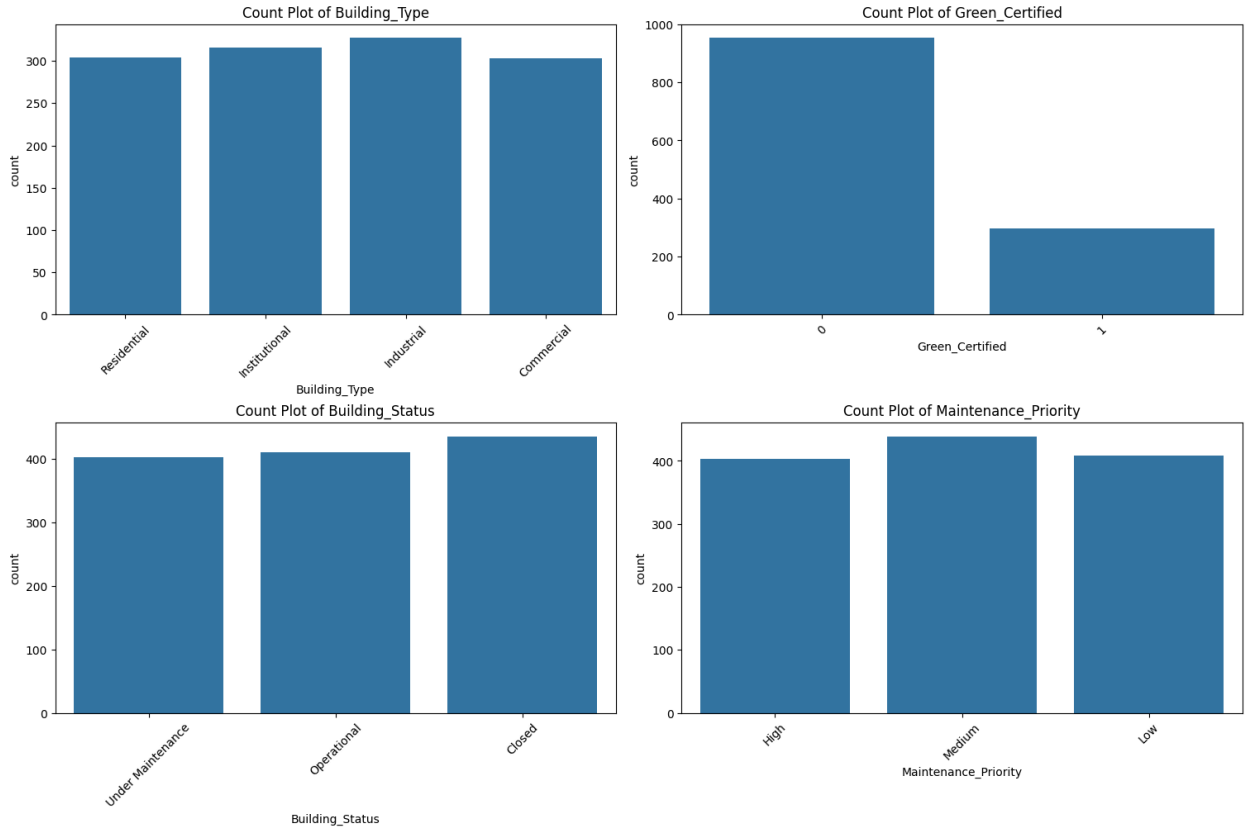
# Box-Plot


Box Plot of Construction_Year
Construction_Year
Box Plot of Number_of_Floors
Number_of_Floors
Box Plot of Energy_Consumption_Per_SqM
Energy_Consumption_Per_SqM
Box Plot of Water_Usage_Per_Building
Water_Usage_Per_Building
Box Plot of Waste_Recycled_Percentage
Waste_Recycled_Percentage
Box Plot of Occupancy_Rate
Occupancy_Rate
Box Plot of Indoor_Air_Quality
Indoor_Air_Quality
Box Plot of Smart_Devices_Count
Smart_Devices_Count
Box Plot of Maintenance_Resolution_Time
Maintenance_Resolution_Time
Box Plot of Energy_Per_SqM
Energy_Per_SqM
Box Plot of Number_of_Residents
Number_of_Residents

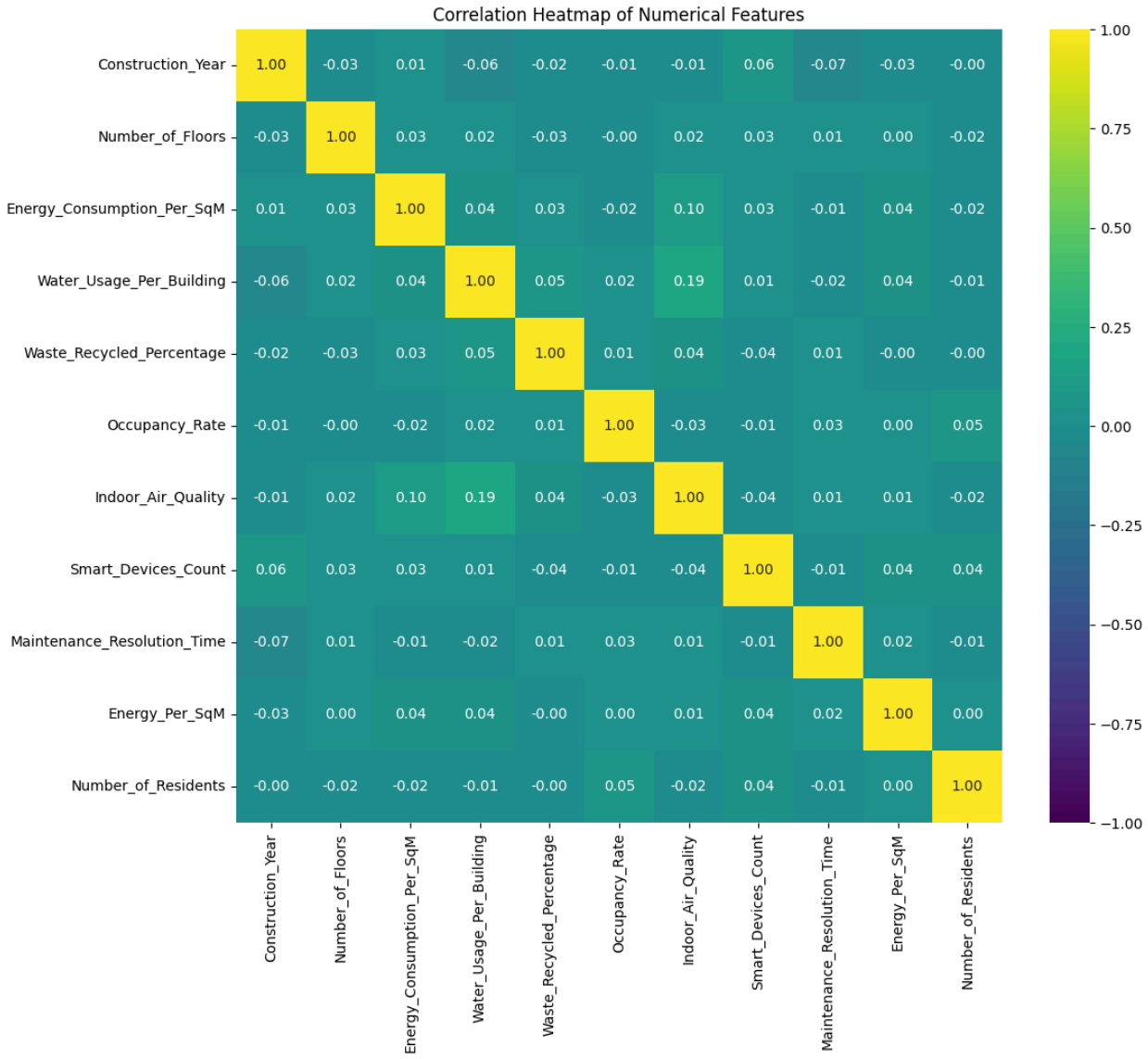
# Violin-Plot


Violin Plot of Construction_Year
Construction_Year
Violin Plot of Number_of_Floors
Number_of_Floors
Violin Plot of Energy_Consumption_Per_SqM
Energy_Consumption_Per_SqM
Violin Plot of Water_Usage_Per_Building
Water_Usage_Per_Building
Violin Plot of Waste_Recycled_Percentage
Waste_Recycled_Percentage
Violin Plot of Occupancy_Rate
Occupancy_Rate
Violin Plot of Indoor_Air_Quality
Indoor_Air_Quality
Violin Plot of Smart_Devices_Count
Smart_Devices_Count
Violin Plot of Maintenance_Resolution_Time
Maintenance_Resolution_Time
Violin Plot of Energy_Per_SqM
Energy_Per_SqM
Violin Plot of Number_of_Residents
Number_of_Residents

# Count Plot


Count Plot of Building_Type
count
Residential
Institutional
Industrial
Commercial
Building_Type
Count Plot of Green_Certified
count
0
1
Green_Certified
Count Plot of Building_Status
count
Under Maintenance
Operational
Closed
Building_Status
Count Plot of Maintenance_Priority
count
High
Medium
Low
Maintenance_Priority

Correlation HeatMap

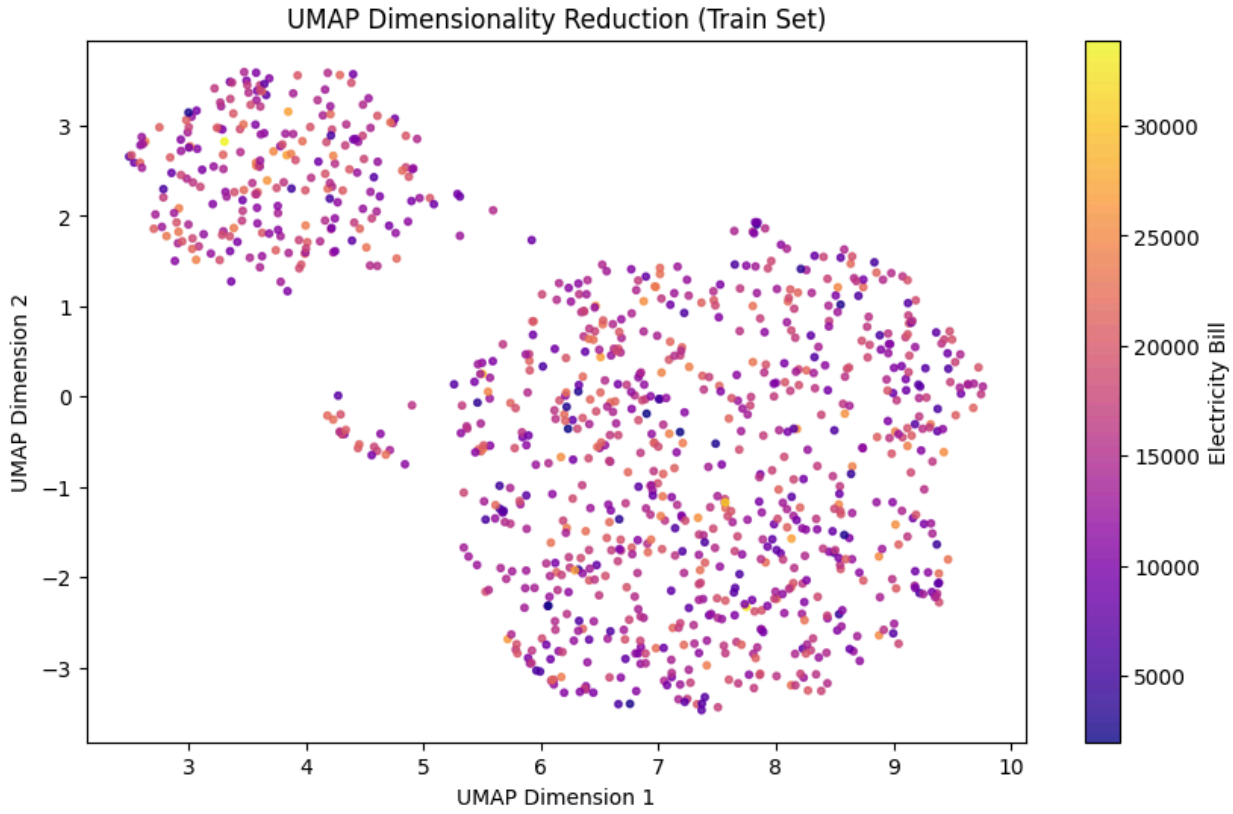


5 insights about the dataset based on the above visualizations:

1. Building Types: Industrial buildings are most common, followed by institutional, and residential.
2. Feature Relationships: Most features are weakly correlated, with some moderate relationships between energy consumption, number of residents, and maintenance.
3. The best correlation is between Water Usage Per Building and Indoor Air Quality.

part b.

UMAP




The UMAP plot effectively visualized the high-dimensional data into a 2D space, revealing distinct clusters. This suggests that the data points are naturally grouped based on their inherent characteristics, and UMAP successfully preserved the underlying structure of the data.

part c.

---- Training Data ----

MSE: 24475013.16847547

RMSE: 4947.222773281538

MAE: 4006.3284693293604

R2 Score: 0.013922520844610209

Adjusted R2 Score: -0.0011091480449536562

---- Test Data ----

MSE: 24278016.155742623

RMSE: 4927.272689403604

MAE: 3842.409312558516

R2 Score: 3.7344733075372893e-05

Adjusted R2 Score: -0.015205988426481465

part d.

---- Training Data ----

MSE: 24673540.31152836

RMSE: 4967.246753638112

MAE: 4006.784035347106

R2 Score: 0.005924030979948536

Adjusted R2 Score: 0.0029298262539845243

---- Test Data ----

MSE: 24181190.647202764

RMSE: 4917.437406536332

MAE: 3825.6515746669897

R2 Score: 0.004025392685427787

Adjusted R2 Score: -0.008120639111091288

It is evident that feature selection did not lead to significant improvements in model performance. Both approaches yielded similar metrics, with the training and test Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and Mean Absolute Error (MAE) showing marginal differences. The $R^2$ and Adjusted $R^2$ scores remain very low across both models, indicating that neither approach effectively explains the variance in the target variable, Electricity_Bill. This suggests that the selected features or the chosen model may not be adequately capturing the underlying patterns in the data, highlighting the need for further exploration into more complex models or feature engineering techniques.

## part e.

---- Training Data ----

MSE: 24188931.451950934

RMSE: 4918.224420657411

MAE: 3976.711053643824

R2 Score: 0.025448510061495067

Adjusted R2 Score: 0.0024826450322065208

---- Test Data ----

MSE: 24129617.719904963

RMSE: 4912.190725114911

MAE: 3797.5717657896607

R2 Score: 0.00614957783259007

Adjusted R2 Score: -0.0949944916800225

In comparing Ridge Regression with the previous Linear Regression model, Ridge Regression shows a modest improvement in performance metrics. Specifically, Ridge Regression achieves slightly lower Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and Mean Absolute Error (MAE) on both training and test datasets, indicating better overall predictive accuracy. The $R^2$ and Adjusted $R^2$ scores also improved, with Ridge Regression yielding higher values, though they still remain relatively low, reflecting limited explanatory power of the models. Overall, Ridge Regression provides better handling of multicollinearity and regularization effects.

## part f.

--- ICA with 4 Components ---

Training Data:

MSE: 24701058.83595767

RMSE: 4970.015979446914

MAE: 4010.9839044323808

R2 Score: 0.00481533301878545

Adjusted R2 Score: 0.0008145906389613877

Testing Data:

MSE: 24167219.465412047

RMSE: 4916.016625827464

MAE: 3818.893490105248

R2 Score: 0.004600837563256355

Adjusted R2 Score: -0.011650577333670142

--- ICA with 5 Components ---

Training Data:

MSE: 24683781.19389808

RMSE: 4968.277487610579

MAE: 4008.4359205885153

R2 Score: 0.005511434533035486

Adjusted R2 Score: 0.0005089769602639738

Testing Data:

MSE: 24261490.949717674

RMSE: 4925.595491889044

MAE: 3831.4951996885943

R2 Score: 0.0007179847322226207

Adjusted R2 Score: -0.019759105744576066

--- ICA with 6 Components ---

Training Data:

MSE: 24682728.660109933

RMSE: 4968.171561058448

MAE: 4009.3988550415106

R2 Score: 0.00555384022480776

Adjusted R2 Score: -0.0004548979007221732

Testing Data:

MSE: 24253810.821500693

RMSE: 4924.815815997659

MAE: 3829.8593447145345

R2 Score: 0.001034313766496875

Adjusted R2 Score: -0.023631505646676043

--- ICA with 8 Components ---

Training Data:

MSE: 24674427.276686326

RMSE: 4967.336034202471

MAE: 4009.038279586307

R2 Score: 0.005888295907569008

Adjusted R2 Score: -0.002136823802561638

Testing Data:

MSE: 24222142.010129824

RMSE: 4921.599537765118

MAE: 3830.142401265849

R2 Score: 0.0023386884116249895

Adjusted R2 Score: -0.03077869952491863

Applying Independent Component Analysis (ICA) with 4, 5, 6, and 8 components shows little variation in model performance metrics. MSE and RMSE remain consistent, while $R^2$ and Adjusted $R^2$ scores are low across all component choices, indicating limited predictive improvement. The slight decrease in performance with more components suggests that ICA does not significantly enhance model accuracy for this dataset.

part g.

Alpha: 0.01
  MSE: 24276857.894035783
  RMSE: 4927.155152218751
  MAE: 3842.2249727301682
  R2 Score: 8.505120317725545e-05
  Adjusted R2 Score: -0.06401206089918321
---
Alpha: 0.1
  MSE: 24267611.164532654
  RMSE: 4926.216719200715
  MAE: 3841.121423809543
  R2 Score: 0.000465905393534749
  Adjusted R2 Score: -0.06360679297867455
---
Alpha: 1
  MSE: 24236425.434417684
  RMSE: 4923.050419650167
  MAE: 3835.0470411464416
  R2 Score: 0.0017503828933501664
  Adjusted R2 Score: -0.062239977177589
---
Alpha: 10
  MSE: 24307006.5318607
  RMSE: 4930.213639575946
  MAE: 3837.5891179007876
  R2 Score: -0.0011567105511067766
  Adjusted R2 Score: -0.06533342276592125
---
Alpha: 100
  MSE: 24358231.74488439
  RMSE: 4935.405935167278
  MAE: 3841.697130377902
  R2 Score: -0.0032665740466735205
  Adjusted R2 Score: -0.0675785339214603

As the alpha value increases in the ElasticNet regularization, the performance metrics on the test dataset generally worsen. For alpha values ranging from 0.01 to 1, the Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and Mean Absolute Error (MAE) show slight improvements or remain stable, while the R2 Score and Adjusted R2 Score gradually increase but remain close to zero, indicating minimal explanatory power of the model. However, when alpha is set to 10 or higher, the model performance deteriorates significantly, with MSE, RMSE, and MAE increasing, and both R2 and Adjusted R2 Scores becoming negative. This suggests that higher regularization strengths are leading to underfitting, where the model fails to capture the underlying patterns in the data effectively.

part h.

---- Training Data ----

MSE: 14926446.25730777

RMSE: 3863.4759294329465

MAE: 3092.748188686501

R2 Score: 0.398626166333897

Adjusted R2 Score: 0.38945888228410885

---- Test Data ----

MSE: 24405496.61674575

RMSE: 4940.1919615279885

MAE: 3813.630549423027

R2 Score: -0.005213319055167753

Adjusted R2 Score: -0.06965007027665293

**Observation:** The Gradient Boosting Regressor outperforms both Linear Regression and Elastic Net models in terms of training metrics, demonstrating a lower MSE, RMSE, MAE, and higher $R^2$ and Adjusted $R^2$ scores. However, on the test set, while it still maintains better performance compared to Linear Regression and Elastic Net (particularly for lower alpha values), all models show relatively poor performance, with $R^2$ values close to zero or negative, indicating challenges in predicting the target variable effectively. The Gradient Boosting Regressor shows the best balance between bias and variance among the tested models.

# part a.

As we increase the complexity of a machine learning model, such as by adding more features or higher-order polynomial terms in a regression model, the behavior of the model can be described in terms of the **bias-variance trade-off**.

## Bias

Bias refers to the error introduced by approximating a real-world problem, which may be complex, by a simplified model. A high-bias model makes strong assumptions about the data, leading to underfitting. In such cases, the model is too simple and does not capture the underlying patterns in the data well, resulting in high training and test errors.

## Variance

Variance refers to the sensitivity of the model to the training data. A high-variance model is too complex and fits the noise in the training data rather than the underlying pattern, leading to overfitting. In such cases, the model performs well on the training set but generalizes poorly to unseen data, resulting in low training error but high test error.

## Effect of Increasing Model Complexity

As the complexity of the model increases, the bias decreases, but the variance increases. Initially, as the model becomes more flexible, it fits the training data better, reducing both bias and error. However, after a certain point, the model starts to fit the noise in the training data, causing the variance to increase.

### Lower Bias and Higher Variance

As the model complexity increases, the model's ability to fit the training data improves, which reduces bias. However, this also leads to overfitting, where the model becomes too specialized to the training data and loses its ability to generalize to new data, increasing variance.

### Overfitting

When a model becomes too complex and fits the training data too closely, it can struggle to perform well on new, unseen data. This is known as overfitting.

## Bias-Variance Trade-off Curve

The bias-variance trade-off can be visualized in the following image:

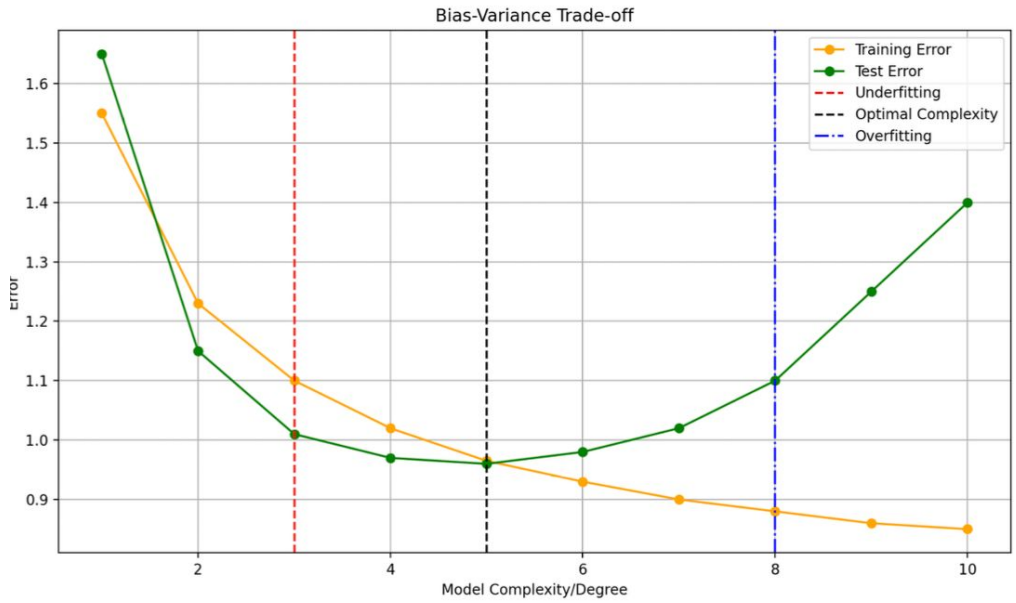



Figure 1: Bias-Variance Tradeoff Curve

*(Replace 'your_image.png' with the path to your bias-variance trade-off curve image.)*

## Conclusion

*In summary, increasing model complexity reduces bias but increases variance. The goal is to find an optimal balance between the two to minimize the total error and achieve better generalization performance. Overfitting occurs when the model becomes too complex and fits the training data too closely, leading to poor performance on unseen data.*

# part b.

We are tasked with evaluating the performance of an email filtering system using standard classification metrics based on the following confusion matrix:

| | **Predicted Spam** | **Predicted Legitimate** |
|---|---|---|
| **Actual Spam** | 200 (TP) | 50 (FN) |
| **Actual Legitimate** | 20 (FP) | 730 (TN) |

Where:

- TP (True Positive) = 200
- FN (False Negative) = 50
- FP (False Positive) = 20
- TN (True Negative) = 730

## 1. Accuracy

Accuracy is the proportion of correctly classified emails out of the total number of emails.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Substituting the values:

$$\text{Accuracy} = \frac{200 + 730}{200 + 730 + 20 + 50} = \frac{930}{1000} = 0.93$$

## 2. Precision

Precision measures the proportion of correctly classified spam emails out of all emails classified as spam.

$$\text{Precision} = \frac{TP}{TP + FP}$$

Substituting the values:

$$\text{Precision} = \frac{200}{200 + 20} = \frac{200}{220} \approx 0.909$$

## 3. Recall

Recall (also known as sensitivity) measures the proportion of actual spam emails that were correctly classified.

$$\text{Recall} = \frac{TP}{TP + FN}$$

Substituting the values:

$$\text{Recall} = \frac{200}{200 + 50} = \frac{200}{250} = 0.8$$

## 4. F1-Score

The F1-Score is the harmonic mean of precision and recall, providing a balance between the two.

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Substituting the values:

$$\text{F1-Score} = 2 \times \frac{0.909 \times 0.8}{0.909 + 0.8} = 2 \times \frac{0.7272}{1.709} \approx 0.851$$

## Conclusion

The model achieves:

- **Accuracy**: 93%
- **Precision**: 90.9%
- **Recall**: 80%
- **F1-Score**: 85.1%

These metrics suggest that the model performs well in classifying emails, with high accuracy and precision, although there is room for improvement in recall.

Date ..../..../.....

Part c

regression equation

$\Rightarrow y = mx + b$

$m$ ↙ slope, $b$ ↘ intercept

$$m = \frac{n\Sigma xy - \Sigma x \Sigma y}{n\Sigma x^2 - (\Sigma x)^2}$$

$$b = \frac{\Sigma y - m\Sigma x}{n}$$

2) $$m = \frac{\overline{x \cdot y} - \bar{x} \cdot \bar{y}}{\overline{x^2} - (\bar{x})^2}$$

$$b = \bar{y} - m\bar{x}$$

acc. to given data in the question,

$$\bar{x} = \frac{3+6+10+15+18}{5} = \frac{52}{5}$$

$$\bar{y} = = \frac{15+30+55+85+100}{5} = \frac{285}{5}$$

Date ..../..../.....

$$\overline{x^2} = \frac{3^2 + 5^2 + 10^2 + 15^2 + 18^2}{5} = \frac{694}{5}$$

$$\overline{xy} = \frac{3(15) + 5(30) + 10(55) + 15(85) + 18(100)}{5}$$

$$= \frac{3850}{5}$$

∴ we have $\Rightarrow m = \frac{5 \times 3850 - 52 \times 285}{5 \times 694 - 52^2}$

$$= \frac{19250 - 14820}{3470 - 2704}$$

$$= \frac{4430}{766} = 5.78$$

& $b = \frac{285 - 5.7 \times 52}{5} \Rightarrow \frac{-15.56}{5} = -3.1$

∴ we have

$\Rightarrow y = 5.78x - 3.1$

when $x = 12$, predicted value of $y = 5.78 \times 12 - 3.1 \Rightarrow 66.2$ Sprint

∴ 66.2 units in 1 hour.

# Toy Example: Empirical Risk vs Generalization

Let's consider a toy example where two models, $f_1$ and $f_2$, are trained on a small dataset. We define the dataset as follows:

$$X = \{(1,1),(2,4),(3,9),(4,16)\}, \quad Y = \{1,4,9,16\}$$

Here, the input $X$ represents some numerical features, and the output $Y$ is the corresponding label.

## Model $f_1$ (Overfitting Model)

Model $f_1$ is a polynomial model of degree 3, which fits the training data perfectly. The predicted values $\hat{f}_1(X)$ for the training points match exactly with $Y$. Hence, the empirical risk (training loss) is:

$$L(\hat{f}_1(X), Y) = 0$$

However, due to its complexity, $f_1$ may overfit the training data and fail to generalize well to new data points outside the training set.

## Model $f_2$ (Simpler Model)

Model $f_2$ is a simple linear model:

$$\hat{f}_2(x) = 3x - 2$$

While this model does not fit the training data perfectly, the empirical risk on the training set is non-zero. Let's compute the predictions and the corresponding loss:

$$\hat{f}_2(1) = 1, \quad \hat{f}_2(2) = 4, \quad \hat{f}_2(3) = 7, \quad \hat{f}_2(4) = 10$$

The squared loss on the training set is:

$$L(\hat{f}_2(X), Y) = (1-1)^2 + (4-4)^2 + (7-9)^2 + (10-16)^2 = 0 + 0 + 4 + 36 = 40$$

Thus, the empirical risk for model $f_2$ is higher than that of $f_1$. However, since $f_2$ is simpler and avoids overfitting, it is more likely to generalize better to new data.

## Conclusion

In this toy example:

- Model $f_1$ achieves zero empirical risk on the training set, but may overfit and not generalize well.
- Model $f_2$ has a higher empirical risk on the training set, but due to its simplicity, it may generalize better to new, unseen data.

This demonstrates that a model with lower empirical risk on the training data (e.g., $f_1$) does not necessarily generalize better than a model with higher empirical risk (e.g., $f_2$).

Date ..../.../.....

part d