Assignment 1 : Report

Name: Lakshay Kumar Roll no. : 2022266
Date: 13-09-2024

Section A

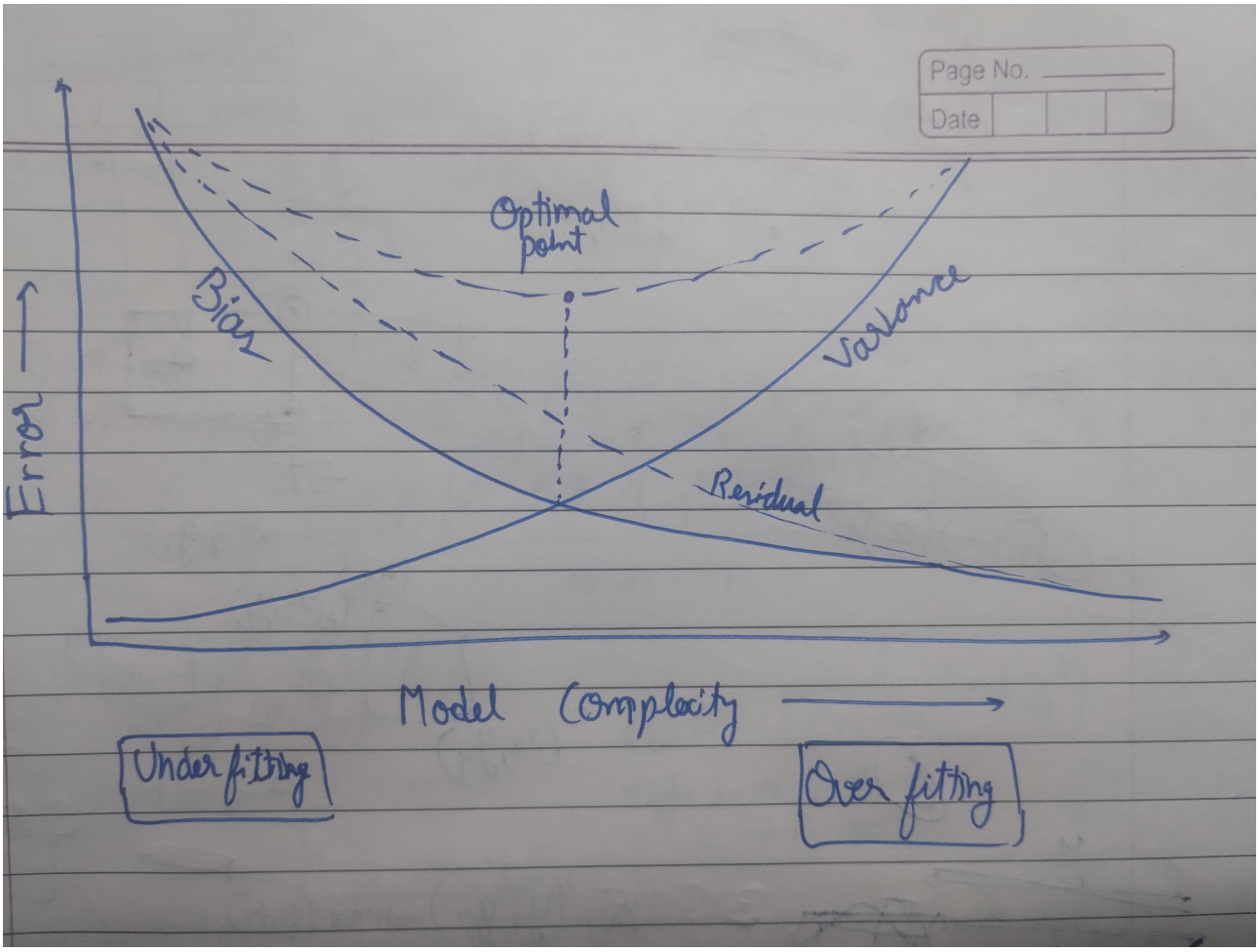
a) The Model's Complexity and Overfitting

A machine-learning model's likelihood of overfitting increases with its complexity. As a result, when applied to fresh or unfamiliar data, the model may learn too much from the training set and perform poorly. A polynomial term-rich or complex model may readily commit the training set to memory, leading to a high training accuracy but a limited ability to generalize. Early stopping techniques, cross-validation, and regularization can all be utilized to lessen overfitting and enhance the model's performance with fresh data.

High Variance: The models exhibit a notable degree of sensitivity to the particular training data that they encounter. Different model behaviors may arise from different training sets, which can lead to a substantial degree of variance.

- **Low Bias**: Even in the absence of overfitting, complicated models can capture intricate patterns in the data with effectiveness. It appears that they are not very biased since they

refrain from making snap judgments.



b) Model Performance Metrics

Assuming a mail classification task where the positive class is the mail being classified:

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

From this, we can calculate:

- **Total correct predictions**:
  [200 + 730 = 930]
- **Precision**:
  [$\frac{TP}{TP+FP}$ = $\frac{200}{200+20}$ = 0.9091]
- **Recall**:
  [$\frac{TP}{TP+FN}$ = $\frac{200}{200+50}$ = 0.80]

- **Accuracy**:
  [$\frac{TP+TN}{TP+FP+TN+FN}$ = $\frac{200+730}{200+20+730+50}$ = 0.93]
- **Negative Predicted Value (NPV)**:
  [$\frac{TN}{TN+FN}$ = $\frac{730}{730+50}$ ≈ 0.9355]
- **Specificity**:
  [$\frac{TN}{TN+FP}$ = $\frac{730}{730+20}$ = 0.9730]
- **F1-Score**:
  [2 × $\frac{Precision \times Recall}{Precision+Recall}$ = 2 × $\frac{0.9091 \times 0.80}{0.9091+0.80}$ ≈ 0.85]

In terms of precision, specificity, recall, and accuracy, the model performs admirably. The robust F1-Score, which indicates efficient classification with the lowest percentage of errors, demonstrates a solid balance between recall and precision. One aspect that can be enhanced by modifying model parameters or experimenting with different strategies is recall. All things considered, the model performs quite well.

c) Linear Regression Calculation

- (Σ x = 52\
- (Σ y = 285\
- (Σ x^2 = 694\
- (Σ xy = 3850\

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

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

further putting values and solving we obtain :
a = 5.78
b = -3.11

The equation for linear regression is:
y = ax + b
y = 5.78x - 3.11

Now, to predict (y) when (x = 12):
y = 5.78 x 12 - 3.11 = 66.25

d) example :

Height	weight
160 cm	50 kg
170 cm	60 kg
180 cm	70 kg
182 cm	80 kg
190 cm	85 kg

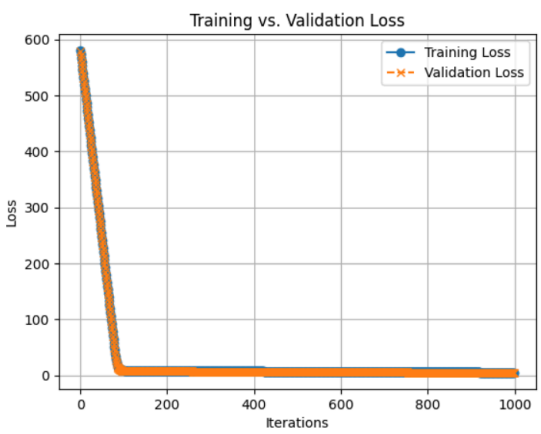
F1: High-capacity models, such as polynomial regression, tend to have lower training error due to their flexibility, but they may struggle to generalize well to unseen data.
F2: Low-capacity models, like linear regression or regularized models, may show higher training error, but their simplicity often leads to better performance on new data by reducing the risk of overfitting.

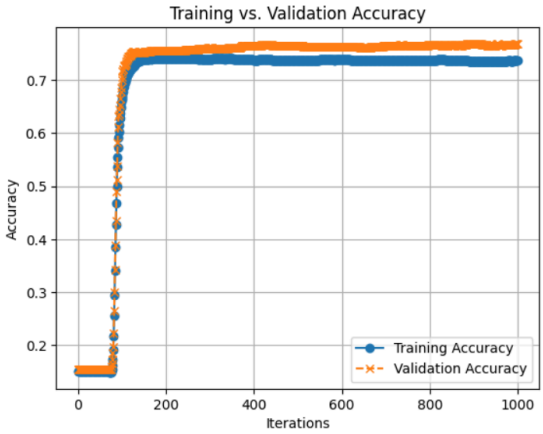
Section B :

(a) (3 marks) Implement Logistic Regression using Batch Gradient Descent. Plot training 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.

Plots without any scaling

Learning Rate : 0.0001
Iterations : 1000





Convergence:

1. **Initial Loss (High):** At the start (near iteration 0), both the training and validation losses are very high, around **600**, indicating a model that is initially poorly fit to the data.
2. **Rapid Decrease:** The loss drops sharply within the first **100 iterations** for both training and validation losses. This suggests that the model is learning effectively and quickly in the early phase of training.
3. **Stabilization:** After approximately **200 iterations**, both the training and validation losses flatten out near **0**, indicating that the model has converged, i.e., both losses stop decreasing significantly, and the model is not overfitting (as the validation loss remains close to the training loss).

Comparison of Training vs. Validation Loss:

- The two curves are almost identical throughout the iterations, which is a good sign that:
    - **No Overfitting:** The model generalizes well, as the validation loss does not diverge from the training loss.
    - **No Underfitting:** The low final loss values indicate the model fits the data adequately.

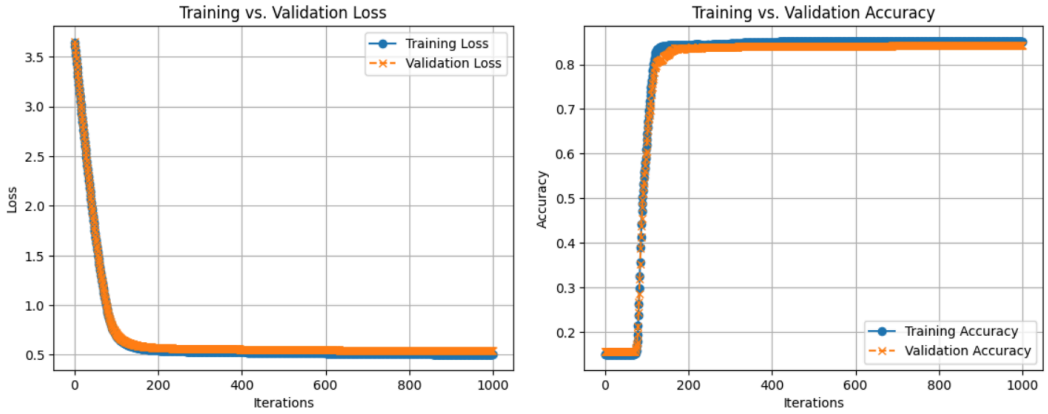
Conclusion:

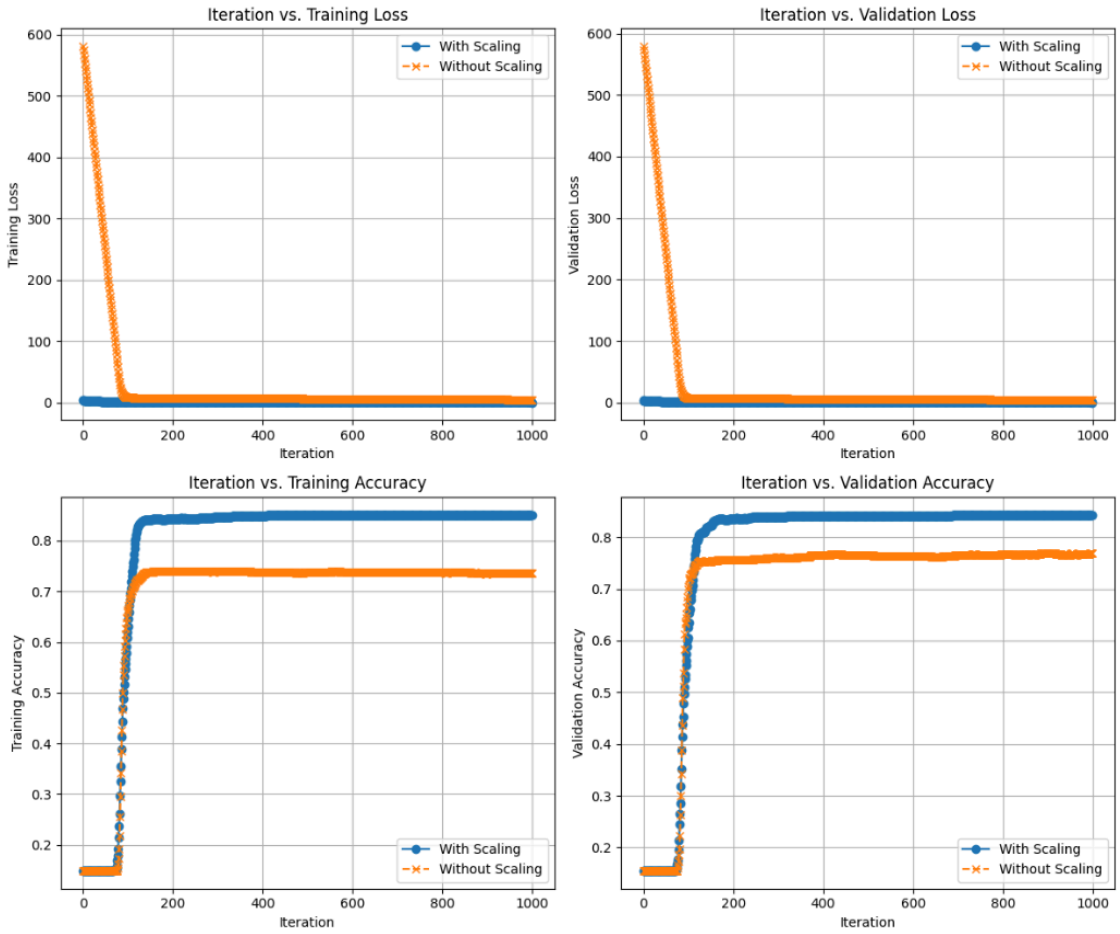
- The model has successfully converged after around **200 iterations**.
- There is no noticeable overfitting or underfitting, as both losses are almost equal and drop to near-zero values.
- Further training may not yield significant improvements

(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.

Plots with min-max scaling

Learning Rate : 0.05
Iterations : 1000





Training Loss (Top Left Plot):

- Without Scaling: The loss starts extremely high (around 600) and quickly drops to near-zero in less than 100 iterations.
- With Scaling: The initial loss starts significantly lower (around 3.5) and drops gradually to near 0.5.

Impact of Scaling:

- Feature scaling leads to a much smoother and more controlled reduction in loss. Without scaling, the model starts with an extreme loss, suggesting instability in gradients due to large variations in feature values. With scaling, the optimization is more stable, leading to a lower initial loss and better control.

Validation Loss (Top Right Plot):

- Without Scaling: Like the training loss, the validation loss starts very high and drops quickly, stabilizing at near zero.
- With Scaling: The validation loss starts lower, similarly to the training loss, and stabilizes around 0.5.

Impact of Scaling:

- The validation loss behaves similarly to the training loss, and feature scaling helps in reducing large variations. The loss converges smoothly and maintains a more realistic, generalizable level. A near-zero validation loss without scaling suggests overfitting, where the model fits too closely to training data.

Training Accuracy (Bottom Left Plot):

- Without Scaling: The training accuracy climbs quickly to about 0.72 and stabilizes around that value.
- With Scaling: Training accuracy starts lower but rises quickly to about 0.82, showing higher accuracy than the model without scaling.

Impact of Scaling:

- With scaling, the model achieves higher training accuracy (around 82%) compared to without scaling (around 72%). This indicates that feature scaling improves the model's ability to learn from the data.

Validation Accuracy (Bottom Right Plot):

- Without Scaling: The validation accuracy also quickly reaches about 0.72 and stabilizes there.
- With Scaling: Validation accuracy rises to around 0.82, reflecting better generalization.

Impact of Scaling:

- The scaled model generalizes better to unseen data, as shown by the higher validation accuracy (~82%) compared to the unscaled model (~72%). This is a significant improvement, suggesting that feature scaling improves not just training performance but also the model's ability to generalize to new data.

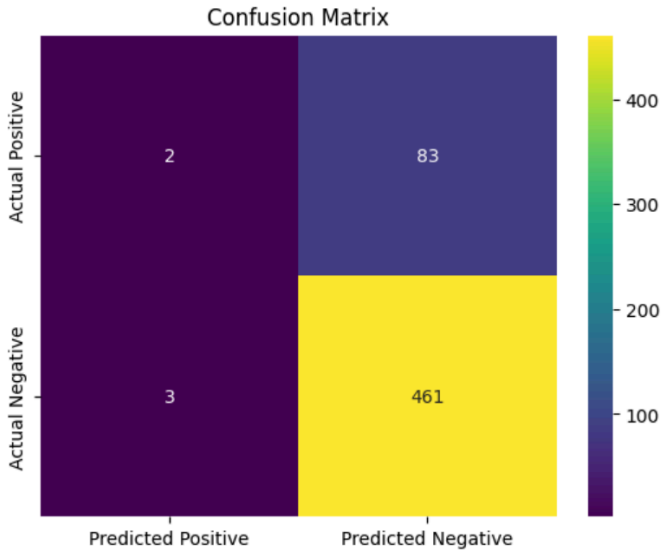
Summary of Feature Scaling's Impact:

1. **Lower Initial Loss**: Feature scaling reduces the starting loss, leading to smoother, more stable training.
2. **Improved Generalization**: The validation accuracy is significantly higher with scaling, indicating better performance on unseen data.
3. **Better Convergence**: Scaling leads to better convergence behavior, with both losses and accuracies stabilizing at more reasonable levels. The model without scaling shows signs of overfitting (zero validation loss), while the scaled model avoids this.
4. **Higher Accuracy**: Both training and validation accuracy are higher with scaling, highlighting the importance of normalization for improving overall model performance.

Conclusion:

Feature scaling significantly improves model convergence, stability, accuracy, and generalization. The comparison clearly shows that using scaling leads to better and more robust performance in both training and validation phases.

(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.




```
F1 Score: 0.044
Recall: 0.024
Precision: 0.400
ROC-AUC Score: 0.452
```

F1 Score: 0.044

Interpretation: The F1 Score is very low, indicating poor performance. The F1 Score is the harmonic mean of Precision and Recall, and a low F1 Score suggests that either the Precision or Recall (or both) are very low. This typically points to issues with the model's ability to correctly identify positive cases (if it's a classification problem) or an imbalance between Precision and Recall.

Implication: The model likely struggles with making accurate predictions and balancing false positives and false negatives.

Recall: 0.024

Interpretation: A Recall of 0.024 is extremely low, meaning the model identifies only a very small fraction of actual positive cases. Recall measures the proportion of actual positives that are correctly identified by the model.

Implication: The model is failing to capture most of the positive cases. This could be due to the model being too conservative or not sufficiently sensitive.

Precision: 0.400

Interpretation: A Precision of 0.400 is relatively low, indicating that among the cases the model predicts as positive, only 40% are truly positive. Precision measures the proportion of true positives among all predicted positives.

Implication: The model's predictions are often incorrect, resulting in a significant number of false positives. This is somewhat better than Recall, but still indicates that the model's predictions are not very reliable.

ROC-AUC Score: 0.452

Interpretation: An ROC-AUC score of 0.452 is close to 0.5, which is equivalent to random guessing. ROC-AUC measures the model’s ability to distinguish between classes, and a score near 0.5 suggests that the model has minimal discriminative power.

Implication: The model does not perform well in distinguishing between positive and negative cases. This further reinforces the idea that the model might not be learning useful patterns from the data.

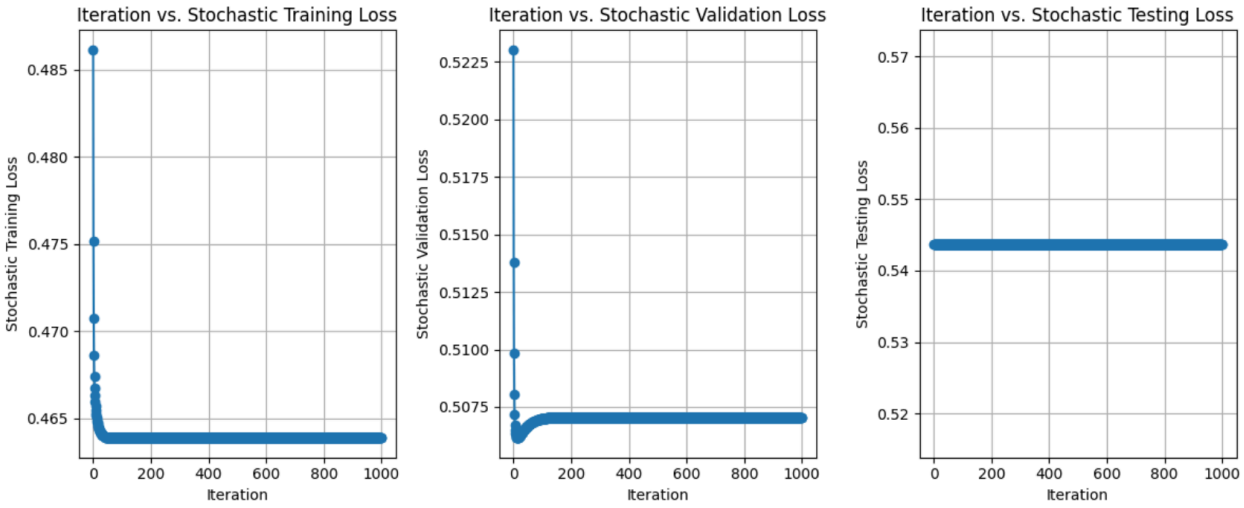
Conclusion:

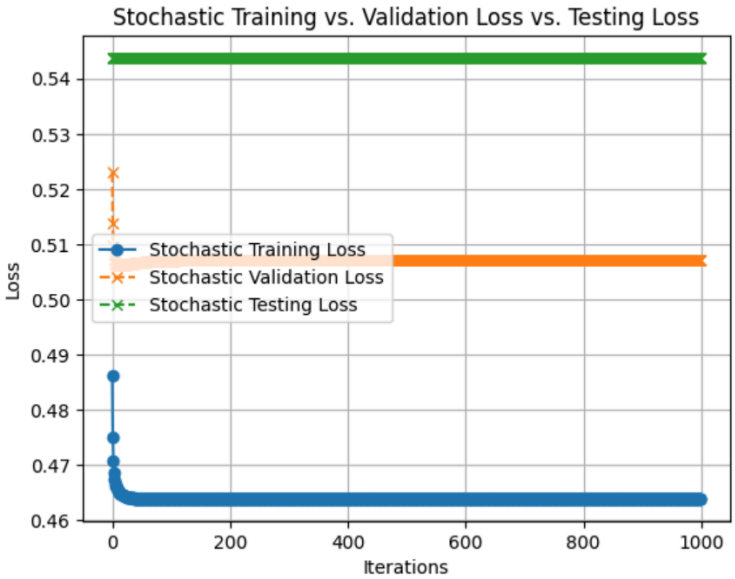
The metrics suggest that the model has significant performance issues. The very low F1 Score and Recall indicate that it fails to identify most positive cases and has poor overall performance. While Precision is somewhat better, it is still not sufficient. The ROC-AUC score further confirms that the model’s ability to differentiate between classes is weak, implying that improvements are needed in the model or data to achieve better performance.

(d) (3 marks) 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.

Stochastic, mini batch (batch size 32) and mini batch (batch size 64) all use:
learning rate : 0.05 Iterations : 1000

Stochastic Gradient Descent





Convergence Speed

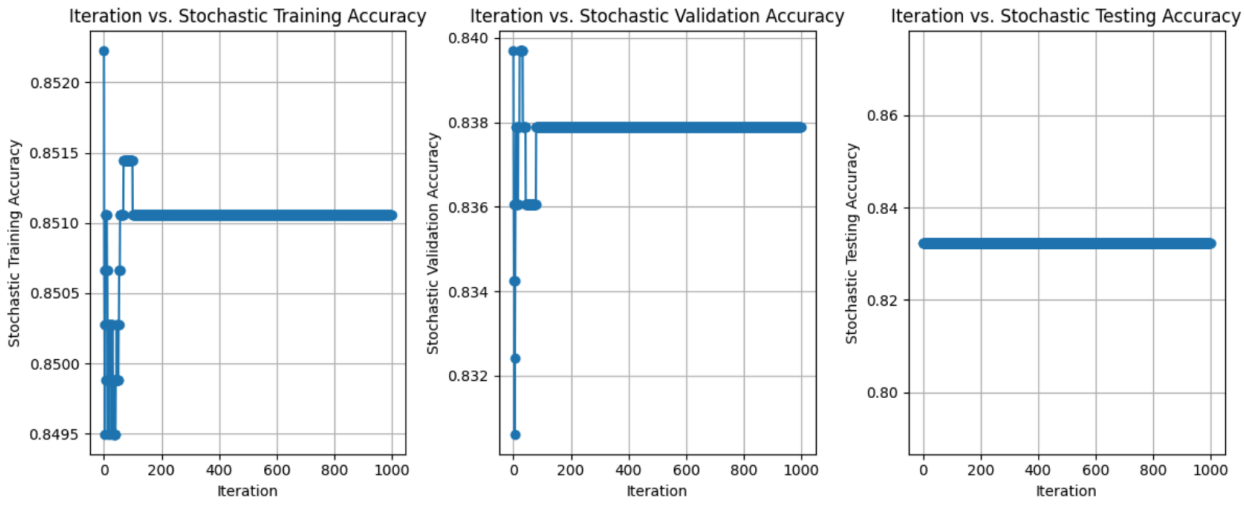
- **Stochastic Training Loss:** Typically converges the fastest due to frequent updates based on small batches.
- **Stochastic Validation Loss:** Converges slower as it's calculated less frequently and on a larger dataset.
- **Stochastic Testing Loss:** Converges the slowest as it's only calculated at the end of training.

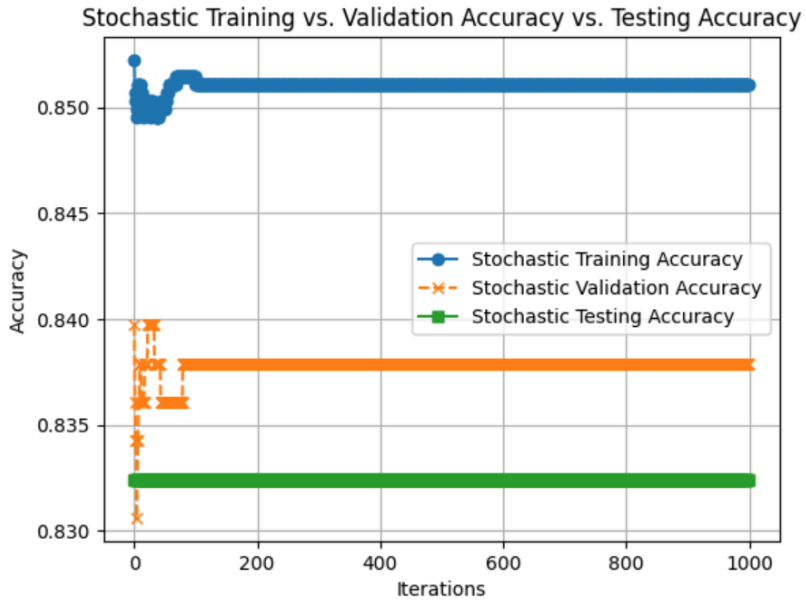
Stability

- **Stochastic Training Loss:** Can be noisy due to the small batch size, leading to fluctuations in the loss.
- **Stochastic Validation Loss:** More stable than training loss as it's calculated on a larger dataset, providing a smoother estimate.
- **Stochastic Testing Loss:** Most stable as it's calculated on a fixed dataset, providing a definitive measure of performance.

Key Trade-offs

- **Speed vs. Stability:** Stochastic training is faster but can be less stable, while validation and testing loss are slower but more stable.
- **Generalization vs. Overfitting:** While stochastic training can lead to faster convergence, it might also lead to overfitting if not monitored carefully. Validation and testing loss can help detect overfitting.





Convergence Speed

- **Stochastic Training Accuracy:** Typically converges the fastest due to frequent updates based on small batches.
- **Stochastic Validation Accuracy:** Converges slower as it's calculated less frequently and on a larger dataset.
- **Stochastic Testing Accuracy:** Converges the slowest as it's only calculated at the end of training.

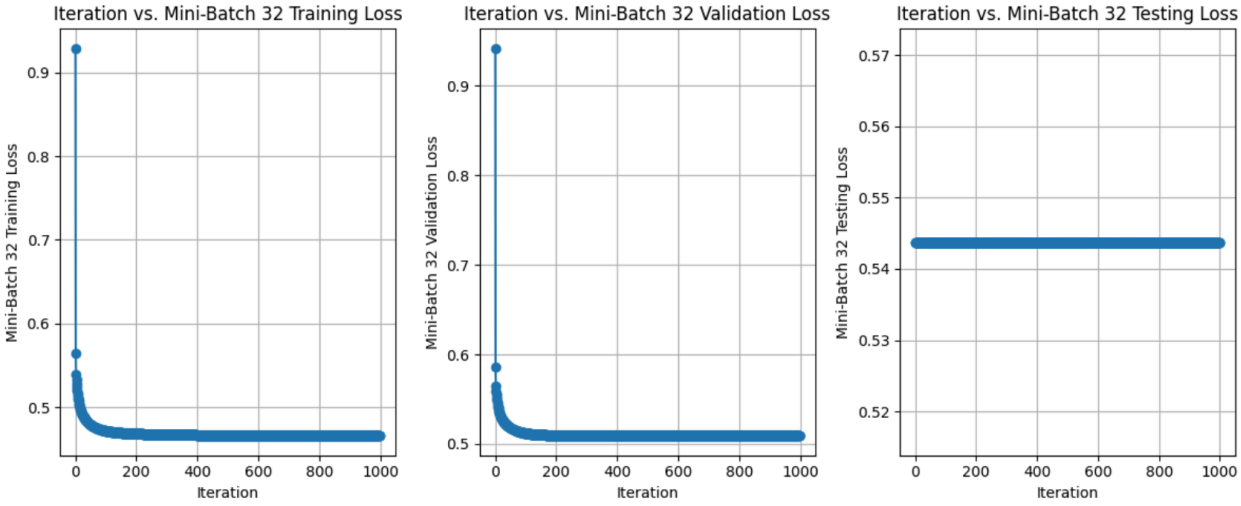
Stability

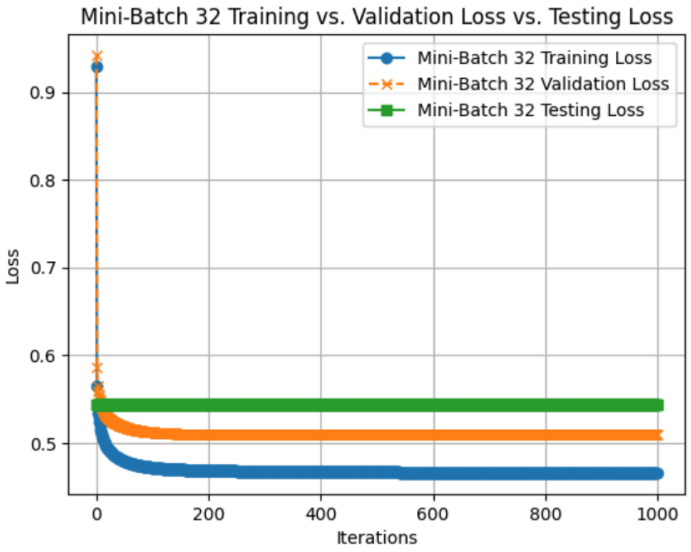
- **Stochastic Training Accuracy:** Can be noisy due to the small batch size, leading to fluctuations in the accuracy.
- **Stochastic Validation Accuracy:** More stable than training accuracy as it's calculated on a larger dataset
- **Stochastic Testing Accuracy:** Converges the slowest as it's only calculated at the end of training.

Key Trade-offs

- **Speed vs. Stability:** Stochastic training is faster but can be less stable, while validation and testing accuracy are slower but more stable.
- **Generalization vs. Overfitting:** While stochastic training can lead to faster convergence, it might also lead to overfitting if not monitored carefully. Validation and testing accuracy can help detect overfitting.

Mini Batch Gradient Descent (batch size = 32)





Convergence Speed

- **Mini-Batch 32 Training Loss:** Converges relatively quickly due to frequent updates based on mini-batches.
- **Mini-Batch 32 Validation Loss:** Converges slower as it's calculated less frequently and on a larger dataset.
- **Mini-Batch 32 Testing Loss:** Converges the slowest as it's only calculated at the end of training.

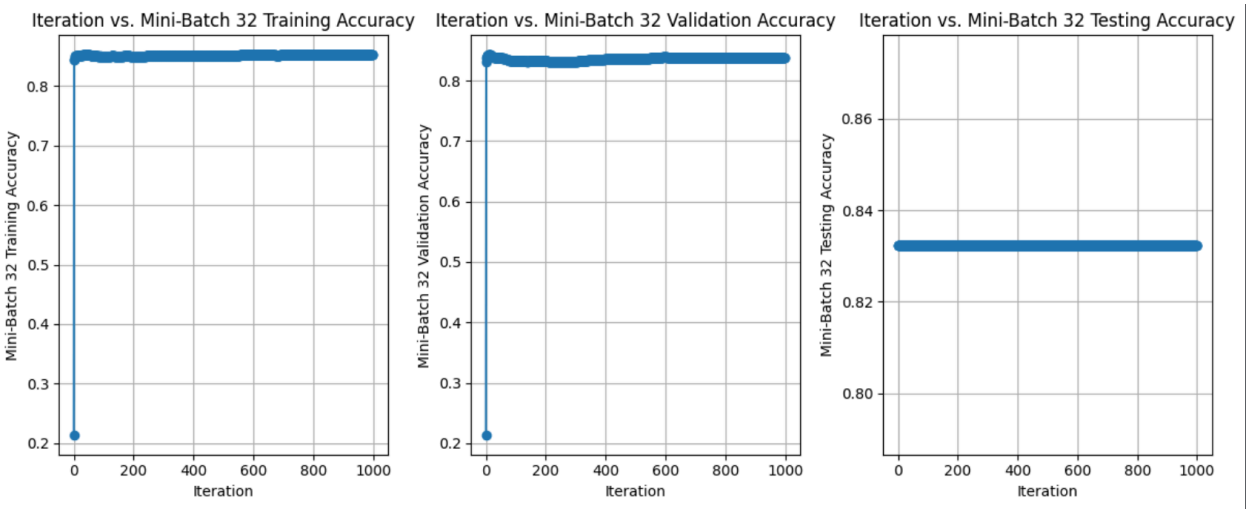
Stability

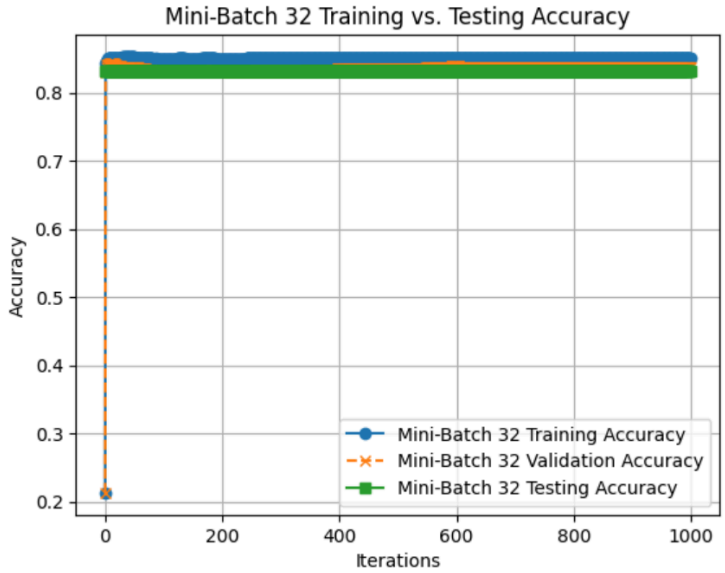
- **Mini-Batch 32 Training Loss:** Can be noisy due to the small batch size, leading to fluctuations in the loss.

- **Mini-Batch 32 Validation Loss:** More stable than training loss as it's calculated on a larger dataset,
- **Mini-Batch 32 Testing Loss:** Most stable as it's calculated on a fixed dataset, providing a definitive measure of performance.

Key Trade-offs

- **Speed vs. Stability:** Mini-batch training is faster but can be less stable, while validation and testing loss are slower but more stable.
- **Generalization vs. Overfitting:** While mini-batch training can lead to faster convergence, it might also lead to overfitting if not monitored carefully. Validation and testing loss can help detect overfitting.





Convergence Speed

- **Mini-Batch 32 Training Accuracy:** Converges relatively quickly due to frequent updates based on mini-batches.
- **Mini-Batch 32 Validation Accuracy:** Converges slower as it's calculated less frequently and on a larger dataset.
- **Mini-Batch 32 Testing Accuracy:** Converges the slowest as it's only calculated at the end of training.

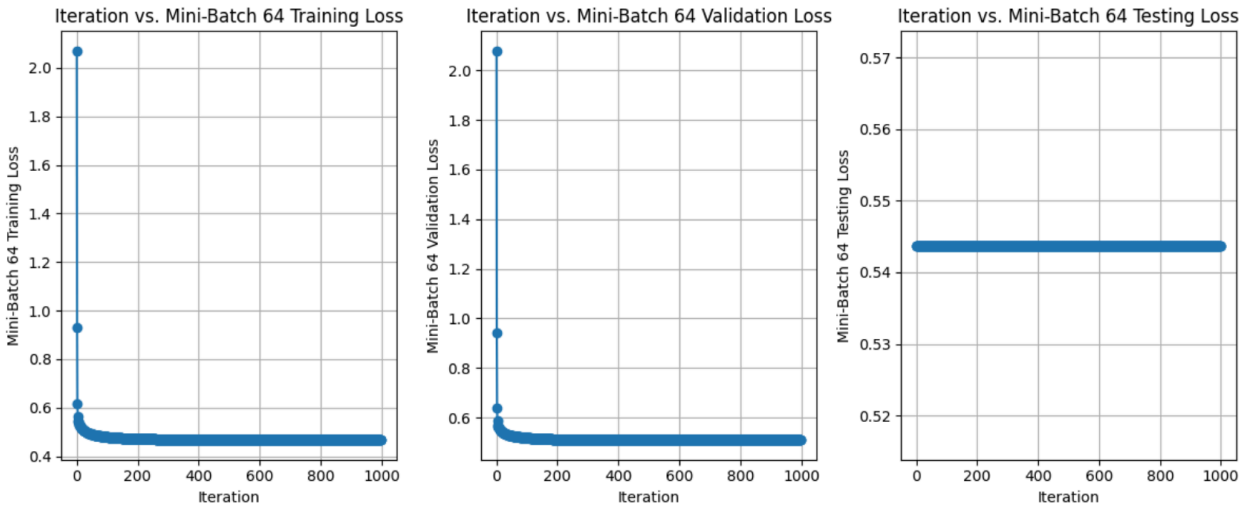
Stability

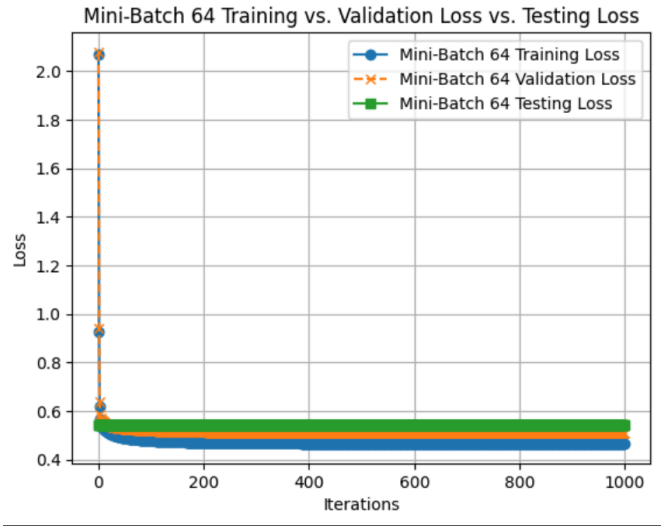
- **Mini-Batch 32 Training Accuracy:** Can be noisy due to the small batch size, leading to fluctuations in the accuracy.
- **Mini-Batch 32 Validation Accuracy:** More stable than training accuracy as it's calculated on a larger dataset, providing a smoother estimate.
- **Mini-Batch 32 Testing Accuracy:** Most stable as it's calculated on a fixed dataset, providing a definitive measure of performance.

Key Trade-offs

- **Speed vs. Stability:** Mini-batch training is faster but can be less stable, while validation and testing accuracy are slower but more stable.
- **Generalization vs. Overfitting:** While mini-batch training can lead to faster convergence, it might also lead to overfitting if not monitored carefully. Validation and testing accuracy can help detect overfitting.

Mini Batch Gradient Descent (Batch Size = 64)





Convergence Speed

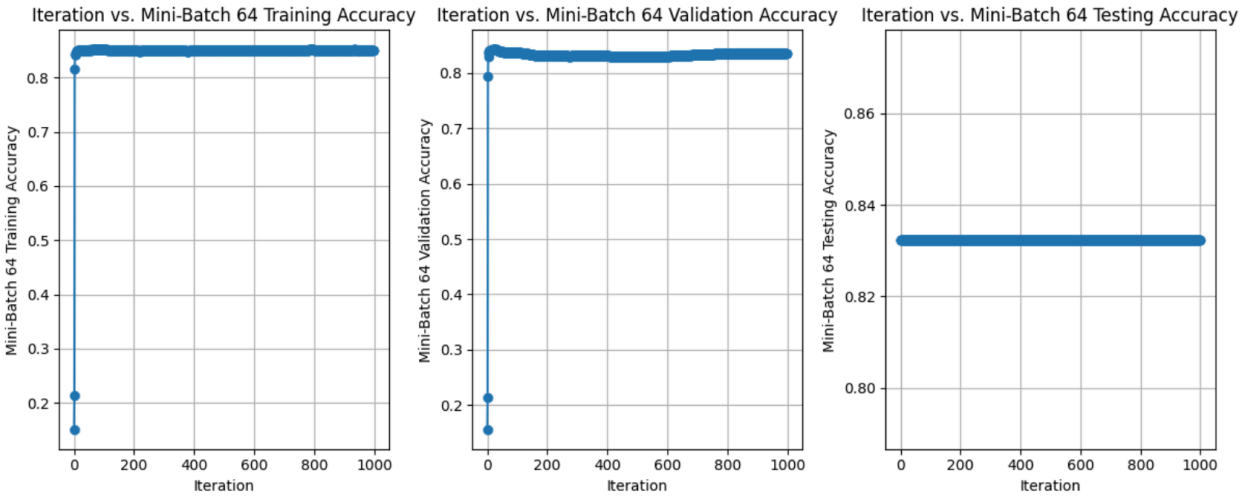
- **Mini-Batch 64 Training Loss:** Converges relatively quickly due to frequent updates based on mini-batches.
- **Mini-Batch 64 Validation Loss:** Converges slower as it's calculated less frequently and on a larger dataset.
- **Mini-Batch 64 Testing Loss:** Converges the slowest as it's only calculated at the end of training.

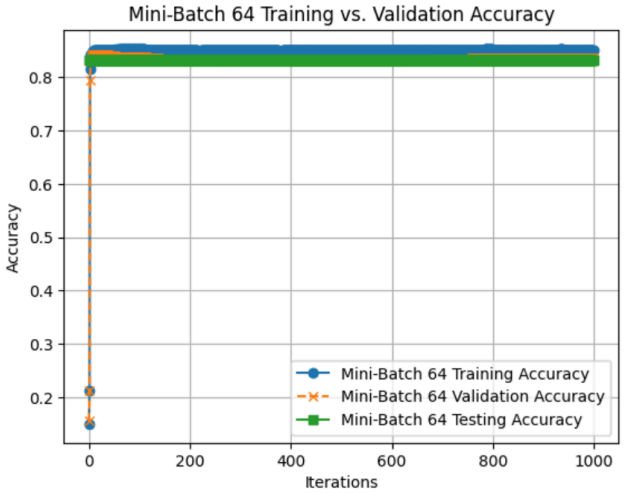
Stability

- **Mini-Batch 64 Training Loss:** Can be noisy due to the small batch size, leading to fluctuations in the loss.
- **Mini-Batch 64 Validation Loss:** More stable than training loss as it's calculated on a larger dataset, providing a smoother estimate.
- **Mini-Batch 64 Testing Loss:** Most stable as it's calculated on a fixed dataset, providing a definitive measure of performance.

Key Trade-offs

- **Speed vs. Stability:** Mini-batch training is faster but can be less stable, while validation and testing loss are slower but more stable.
- **Generalization vs. Overfitting:** While mini-batch training can lead to faster convergence, it might also lead to overfitting if not monitored carefully. Validation and testing loss can help detect overfitting.





Convergence Speed

- **Mini-Batch 64 Training Accuracy:** Converges relatively quickly due to frequent updates based on mini-batches.
- **Mini-Batch 64 Validation Accuracy:** Converges slower as it's calculated less frequently and on a larger dataset.
- **Mini-Batch 64 Testing Accuracy:** Converges the slowest as it's only calculated at the end of training.

Stability

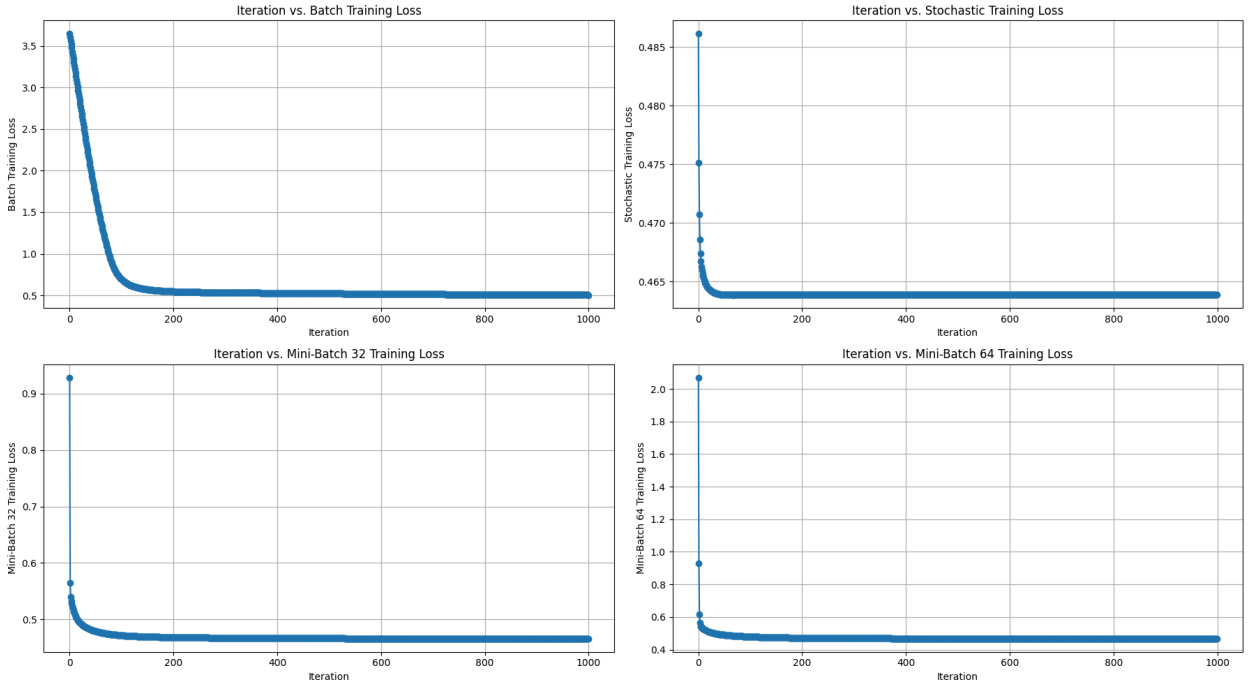
- **Mini-Batch 64 Training Accuracy:** Can be noisy due to the small batch size, leading to fluctuations in the accuracy.
- **Mini-Batch 64 Validation Accuracy:** More stable than training accuracy as it's calculated on a larger dataset, providing a smoother estimate.

- **Mini-Batch 64 Testing Accuracy:** Most stable as it's calculated on a fixed dataset, providing a definitive measure of performance.

Key Trade-offs

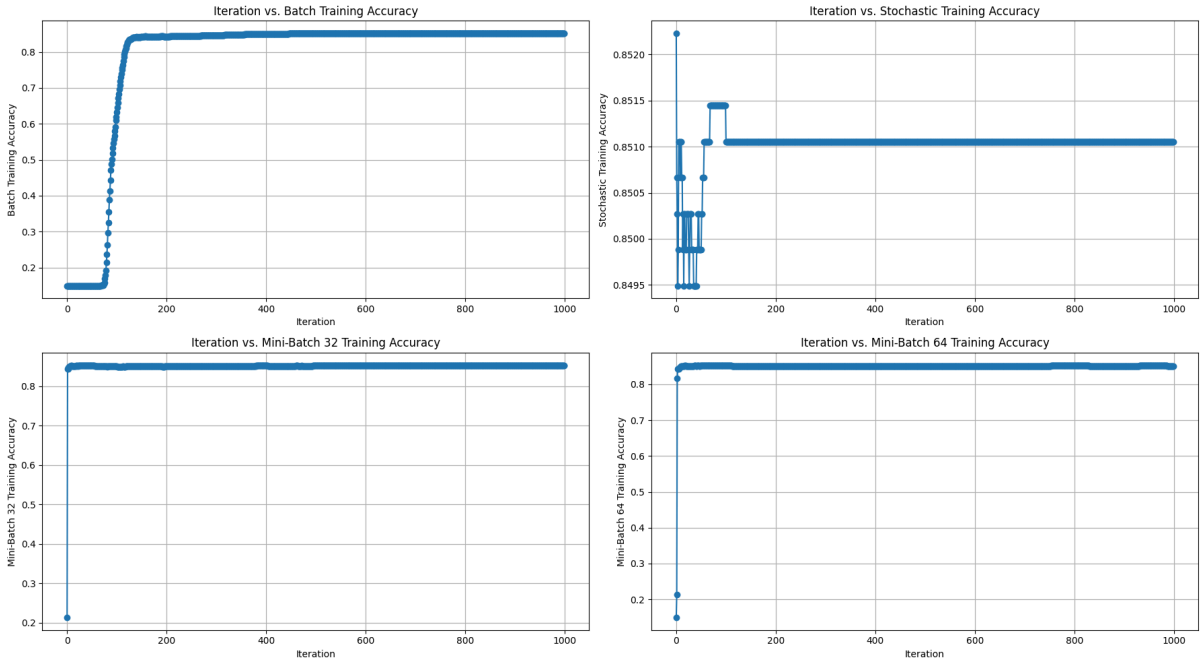
- **Speed vs. Stability:** Mini-batch training is faster but can be less stable, while validation and testing accuracy are slower but more stable.
- **Generalization vs. Overfitting:** While mini-batch training can lead to faster convergence, it might also lead to overfitting if not monitored carefully. Validation and testing accuracy can help detect overfitting.

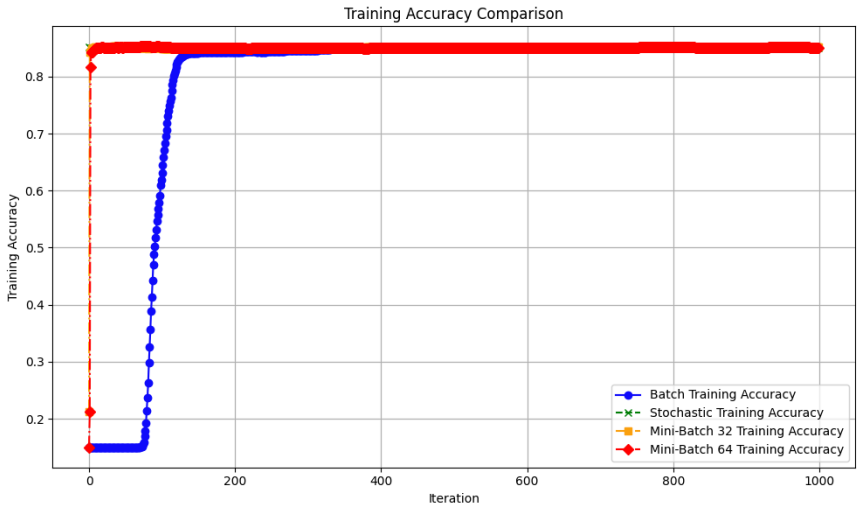
Comparing Performance on Training Set



Key Trade-offs

- **Speed vs. Stability:** Batch training is the slowest but most stable, while stochastic training is the fastest but can be noisy. Mini-batch training offers a balance between the two.
- **Computational Efficiency:** Stochastic training is the most efficient, while batch training can be computationally expensive for large datasets.
- **Generalization:** While batch training can provide better generalization, it may be more prone to getting stuck in local minima. Stochastic and mini-batch training can help avoid this issue.

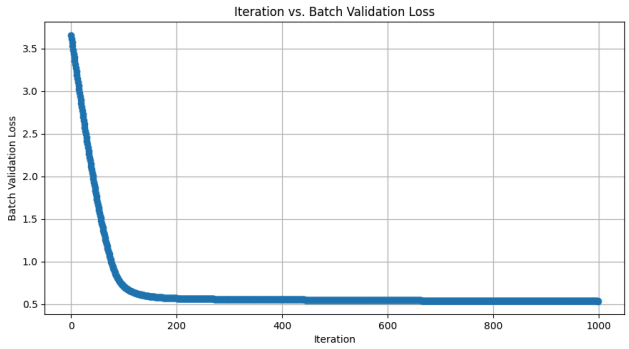


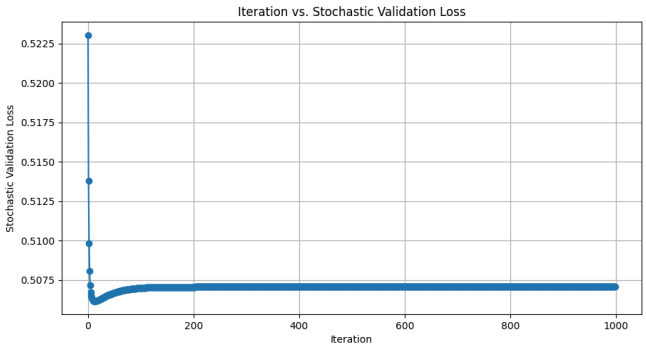


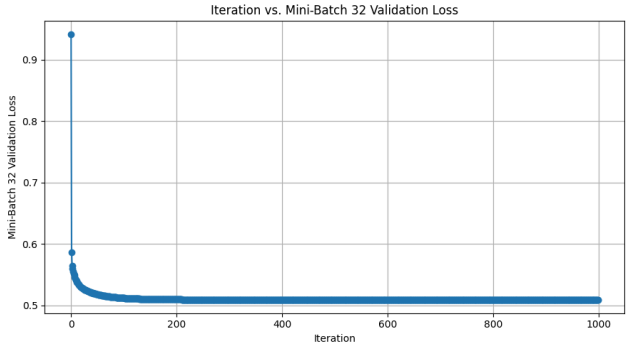
Key Trade-offs

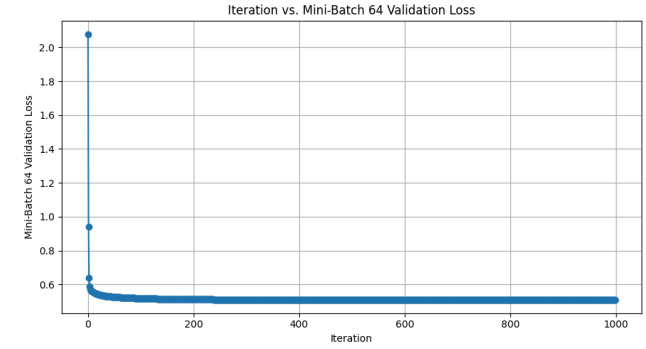
- **Speed vs. Stability:** Batch training is the slowest but most stable, while stochastic training is the fastest but can be noisy. Mini-batch training offers a balance between the two.
- **Computational Efficiency:** Stochastic training is the most efficient, while batch training can be computationally expensive for large datasets.
- **Generalization:** While batch training can provide better generalization, it may be more prone to getting stuck in local minima. Stochastic and mini-batch training can help avoid this issue.

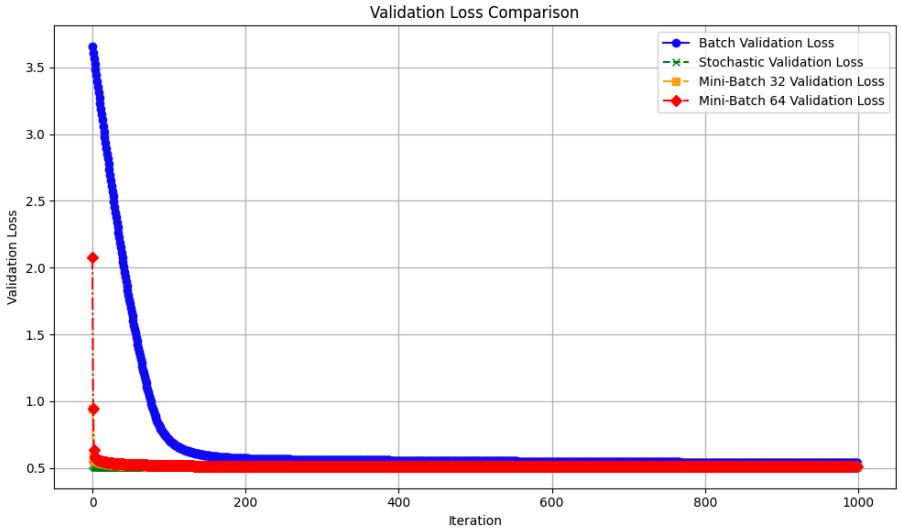
Comparing Performance on Validation Set







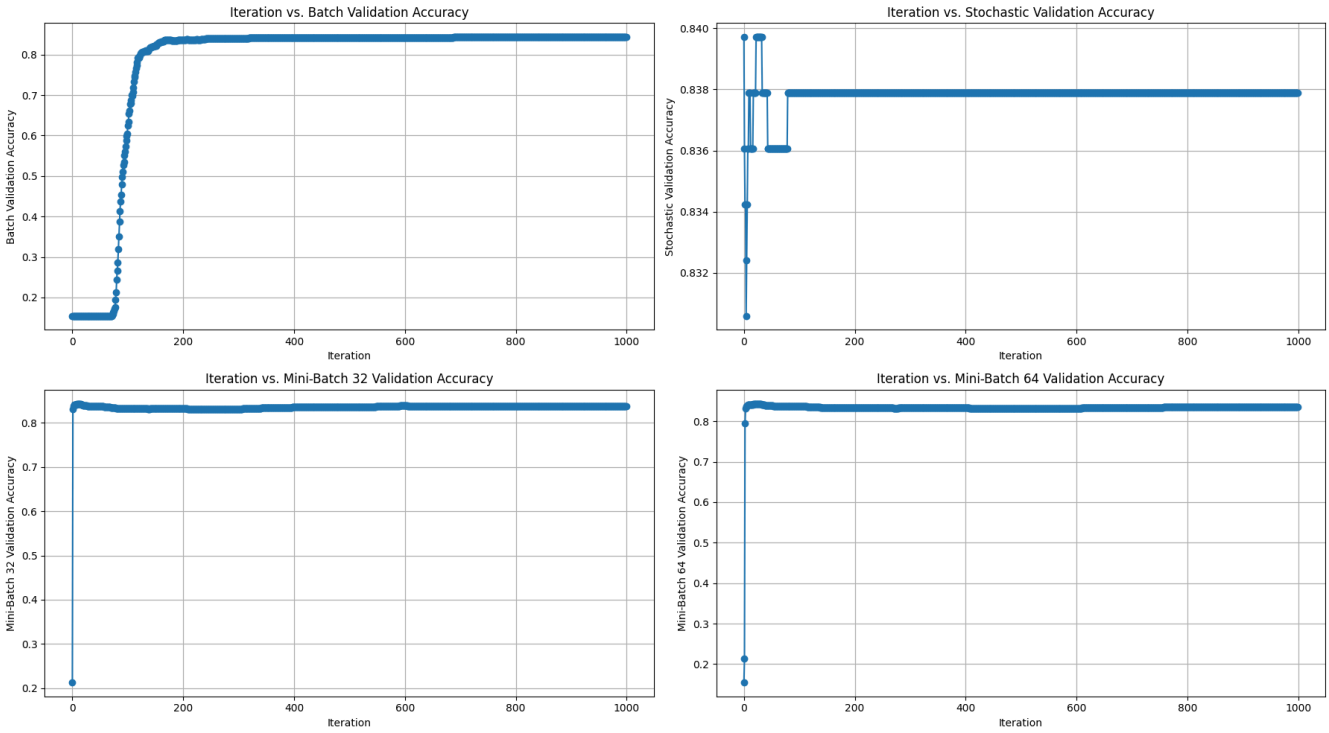


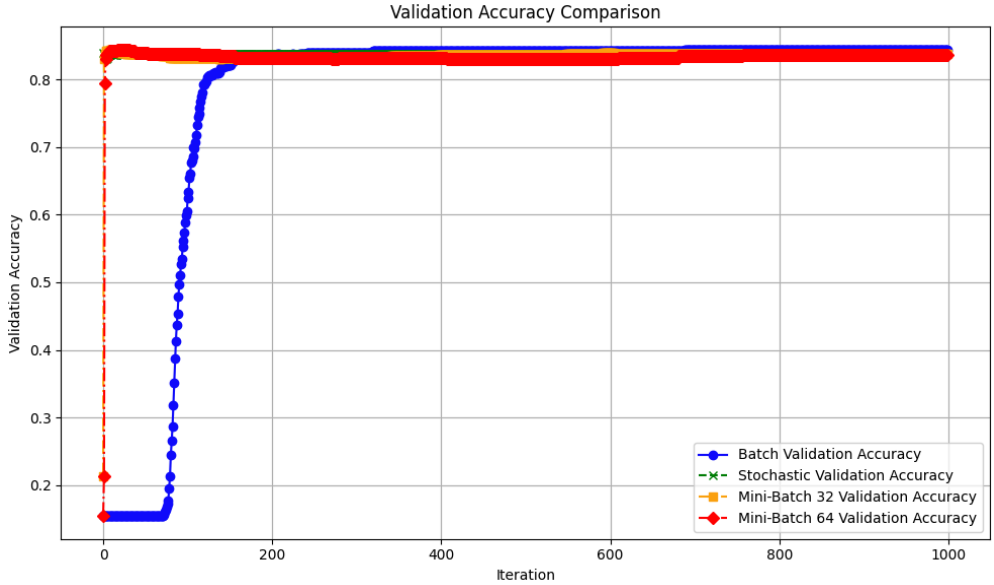


Key Trade-offs

- **Speed vs. Stability:** Batch validation loss is the slowest but most stable, while stochastic validation loss is the fastest but can be noisy. Mini-batch validation loss offers a balance between the two.
- **Computational Efficiency:** Stochastic validation loss is the most efficient, while batch validation loss can be computationally expensive for large datasets.

- **Generalization:** While batch validation loss can provide a more accurate estimate of generalization performance, it may be more computationally expensive.



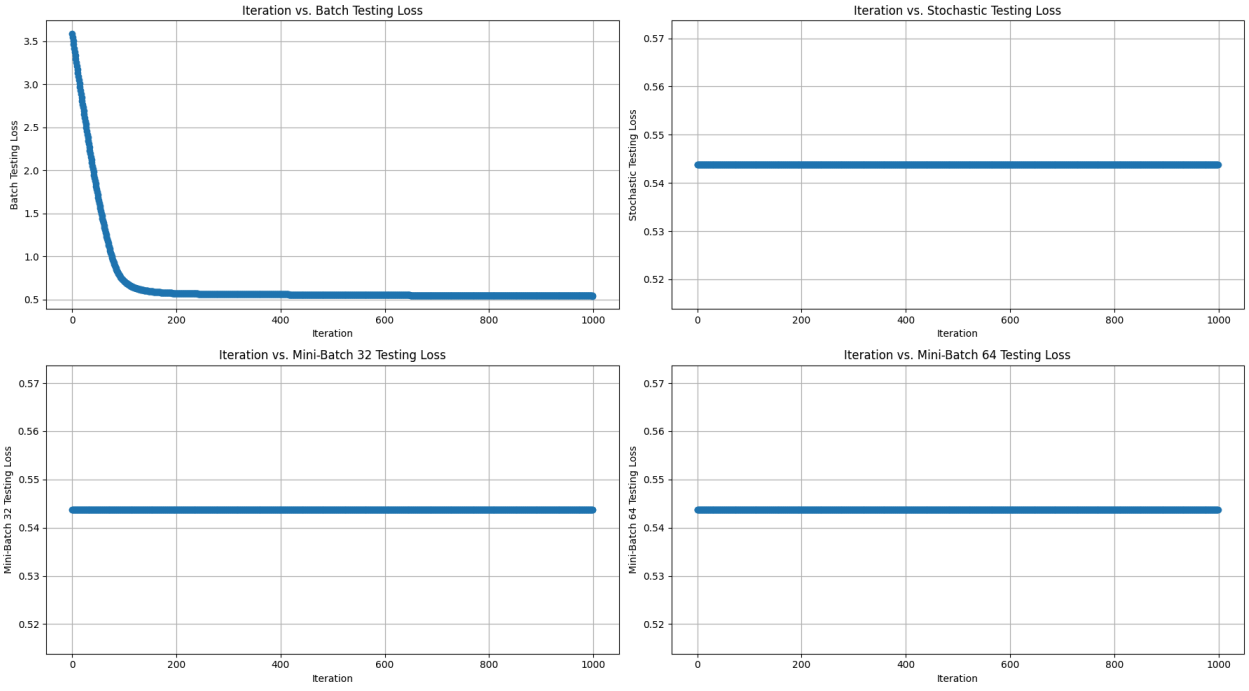


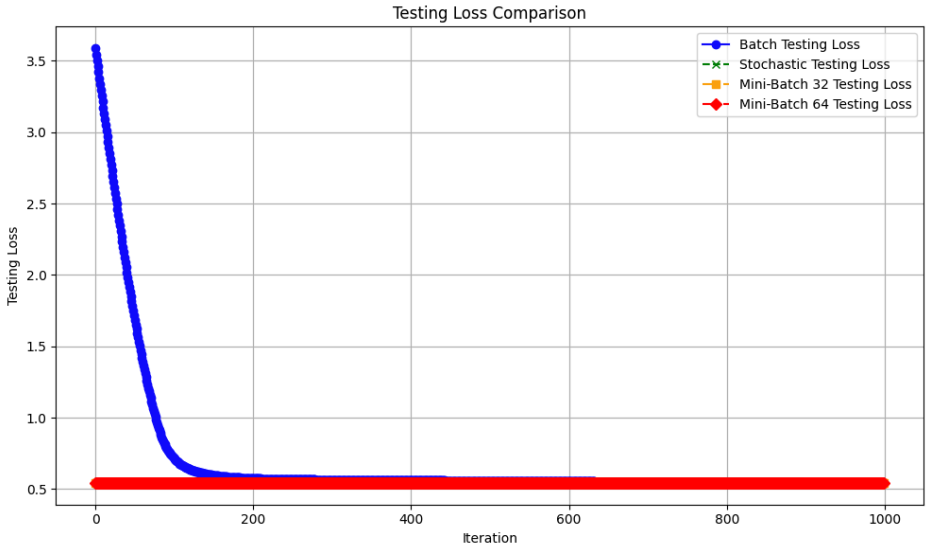
Key Trade-offs

- **Speed vs. Stability:** Batch validation is the slowest but most stable, while stochastic validation is the fastest but can be noisy. Mini-batch validation offers a balance between the two.

- **Computational Efficiency:** Stochastic validation is the most efficient, while batch validation loss can be computationally expensive for large datasets.
- **Generalization:** While batch validation can provide a more accurate estimate of generalization performance, it may be more computationally expensive.

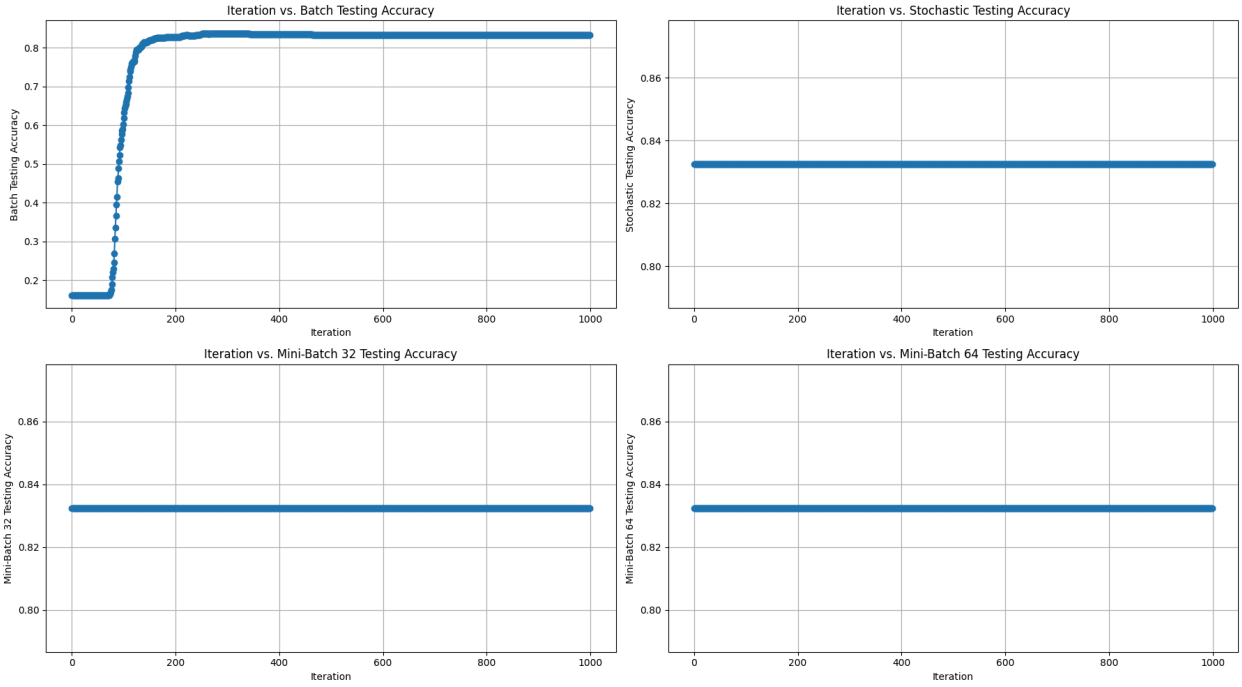
Comparing Performance on Testing Set

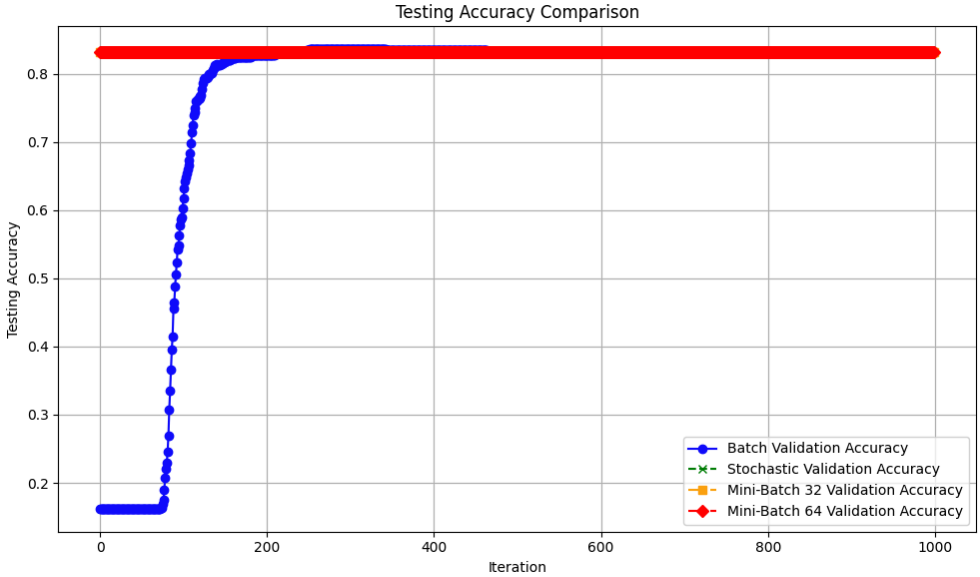




Key Trade-offs

- **Speed vs. Stability:** Batch testing loss is the slowest but most stable, while stochastic testing loss is the fastest but can be noisy. Mini-batch testing loss offers a balance between the two.
- **Computational Efficiency:** Stochastic testing loss is the most efficient, while batch testing loss can be computationally expensive for large datasets.
- **Generalization:** While batch testing loss can provide a more accurate estimate of generalization performance, it may be more computationally expensive.





Key Trade-offs

- **Speed vs. Stability:** Batch testing is the slowest but most stable, while stochastic testing loss is the fastest but can be noisy. Mini-batch testing loss offers a balance between the two.
- **Computational Efficiency:** Stochastic testing is the most efficient, while batch testing loss can be computationally expensive for large datasets.
- **Generalization:** While batch testing can provide a more accurate estimate of generalization performance, it may be more computationally expensive.

(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.

```
Accuracy for each fold : [0.83310534 0.86867305 0.84404925 0.8495212  0.83994528]
Precision for each fold : [0.75       1.         0.         0.16666667 0.54545455]
Recall for each fold : [0.02419355 0.01030928 0.         0.00943396 0.05084746]
F1 Score for each fold : [0.046875   0.02040816 0.         0.01785714 0.09302326]

Accuracy: Mean = 0.847, Std = 0.012
Precision: Mean = 0.492, Std = 0.368
Recall: Mean = 0.019, Std = 0.018
F1 Score: Mean = 0.036, Std = 0.032
```

Stability:

- **Accuracy**: The model shows high stability with an accuracy mean of 0.847 and a low standard deviation of 0.012. This indicates that the model consistently performs well in classifying instances correctly across different folds. The low fluctuation in accuracy suggests that the model's overall performance is reliable and not overly sensitive to the specific data subset used in each fold.
- **Precision**: The precision metric has a mean of 0.492 but a high standard deviation of 0.368. This significant fluctuation indicates lower stability in the model's ability to correctly identify positive instances. The variability in precision suggests that the model's performance in minimizing false positives varies considerably depending on the data subset.
- **Recall**: With a mean recall of 0.019 and a standard deviation of 0.018, the model demonstrates high stability in terms of recall. The low standard deviation indicates that

the model's ability to identify all relevant positive instances is consistent across different folds. However, the consistently low recall values across folds highlight that the model is persistently poor at capturing positive cases.
- **F1 Score**: The F1 Score, with a mean of 0.036 and a standard deviation of 0.032, shows some fluctuation but remains relatively stable. The consistency in the F1 Score reflects that the model's ability to balance precision and recall is somewhat steady, but the low values indicate persistent difficulty in achieving a good balance between precision and recall.

Variance:

- **Accuracy**: The low variance in accuracy, indicated by a small standard deviation, reflects that the model's overall classification performance is not highly sensitive to the specific data subsets used in each fold. This suggests a robust model with stable performance in terms of correct classifications.
- **Precision**: The high variance in precision, as evidenced by the large standard deviation, indicates that the model's performance in identifying positive instances fluctuates significantly across folds. This suggests that the model's ability to reduce false positives is inconsistent, possibly due to varying data distributions or characteristics in different folds.
- **Recall**: The low variance in recall, despite its low values, shows that the model's performance in capturing positive instances is stable across folds. However, the consistently low recall values indicate that the model struggles to identify relevant instances effectively.
- **F1 Score**: The slight variance in the F1 Score, coupled with its low values, suggests some sensitivity to the data used in each fold. The variability in the F1 Score indicates that the model's performance in balancing precision and recall is somewhat unstable, which can be attributed to the fluctuating precision and recall values across different data subsets.

Conclusion

The model demonstrates high stability in terms of accuracy and recall, performing consistently well in overall classification and identifying relevant instances. However, precision and F1 Score show considerable fluctuation, indicating less stability in minimizing false positives and balancing precision with recall. The low variance in accuracy and recall suggests that the model's overall performance is consistent, while the high variance in precision and the slight variance in F1 Score point to instability in the model's ability to effectively predict positive cases and balance classification metrics.

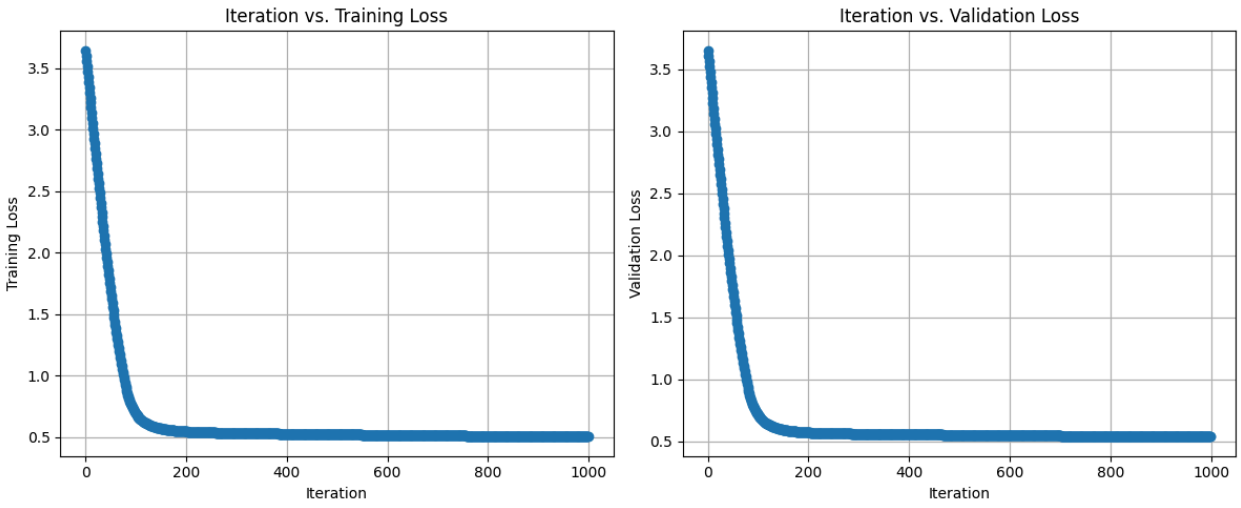
(f) (3 marks) Implement early stopping in your best Gradient Descent method to avoid overfitting. Define and use appropriate stopping criteria. Experiment with different learning rates and regularization techniques (L1 and L2). Plot and compare the performance with and without early stopping. Analyze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

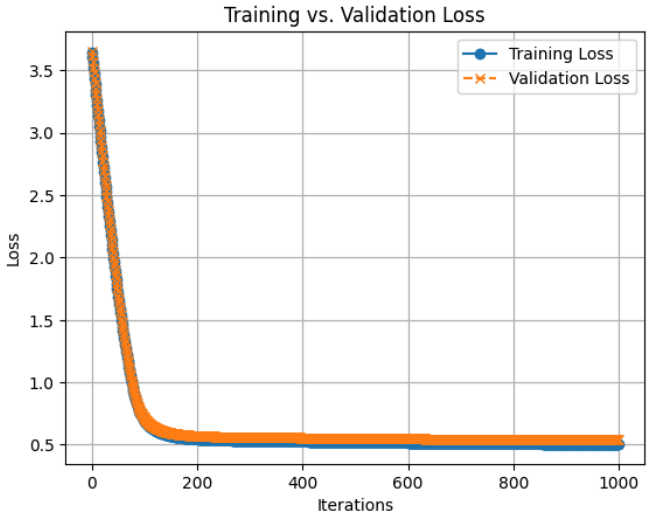
Early Stopping Criteria

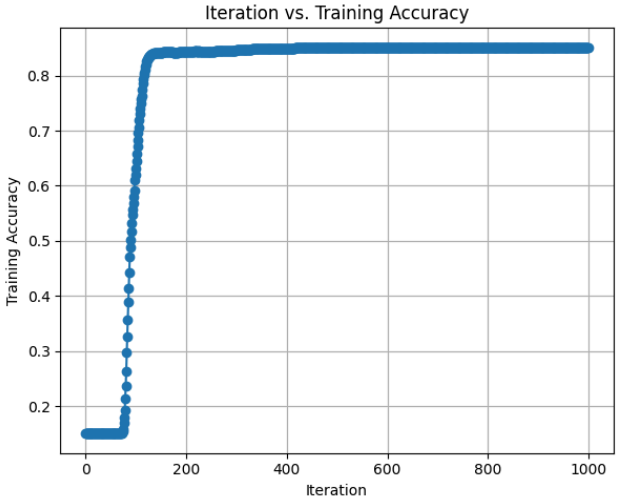
Validation Loss : Using validation loss for early stopping helps prevent overfitting by stopping the training process when the model's performance on the validation set starts to degrade, even if performance on the training set continues to improve. This approach ensures that the model generalizes well to unseen data and doesn't just memorize the training data. It helps in finding the optimal point where the model has the best balance between fitting the training data and maintaining generalization.

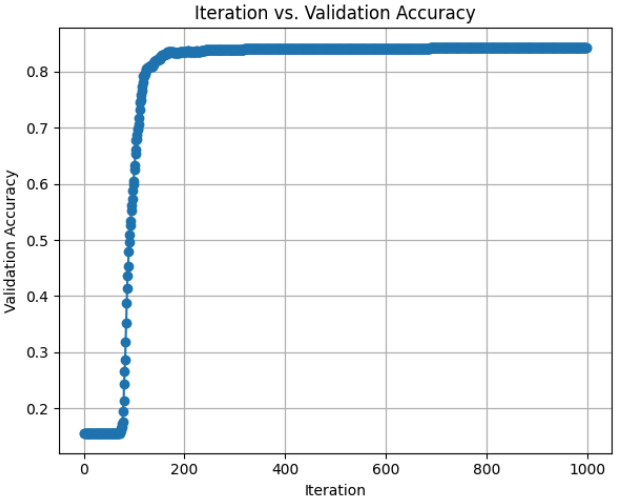
Early Stopping

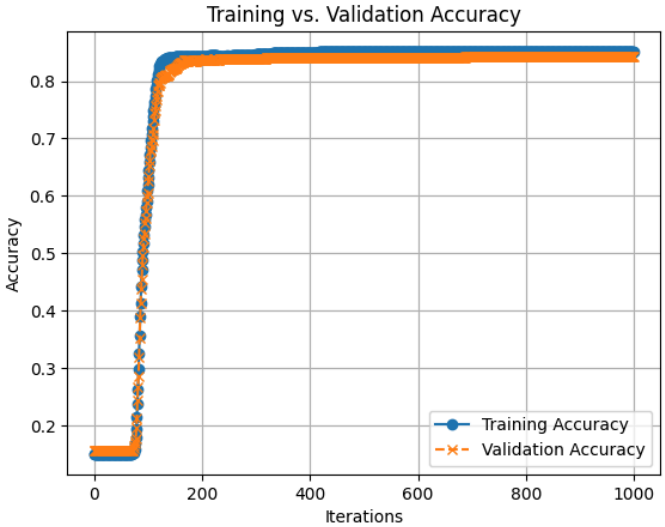
Learning Rate : 0.05
Iterations : 1000




Training vs. Validation Loss
Training Loss
Validation Loss
Loss
Iterations


Iteration vs. Training Accuracy
Training Accuracy
Iteration


Iteration vs. Validation Accuracy
Validation Accuracy
Iteration


Training vs. Validation Accuracy
Accuracy
Training Accuracy
Validation Accuracy
Iterations

```
F1 Score: 0.044
Recall: 0.024
Precision: 0.400
ROC-AUC Score: 0.452
```

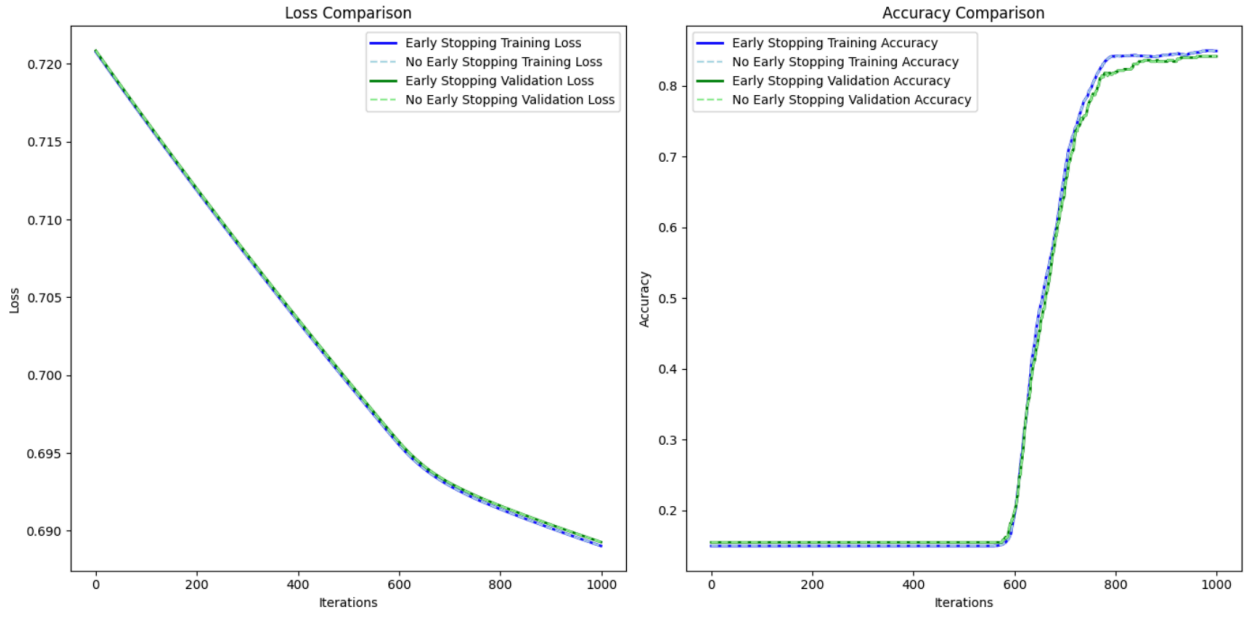
Conclusion

- **F1 Score (0.044)**: Indicates a significant imbalance between precision and recall.
- **Recall (0.024)**: Shows the model is failing to identify most positive cases.
- **Precision (0.400)**: Suggests that 60% of positive predictions are incorrect.
- **ROC-AUC Score (0.452)**: Close to random guessing, reflecting weak class differentiation.

  The lack of improvement with early stopping suggests the model's performance issues are not being addressed by additional training.

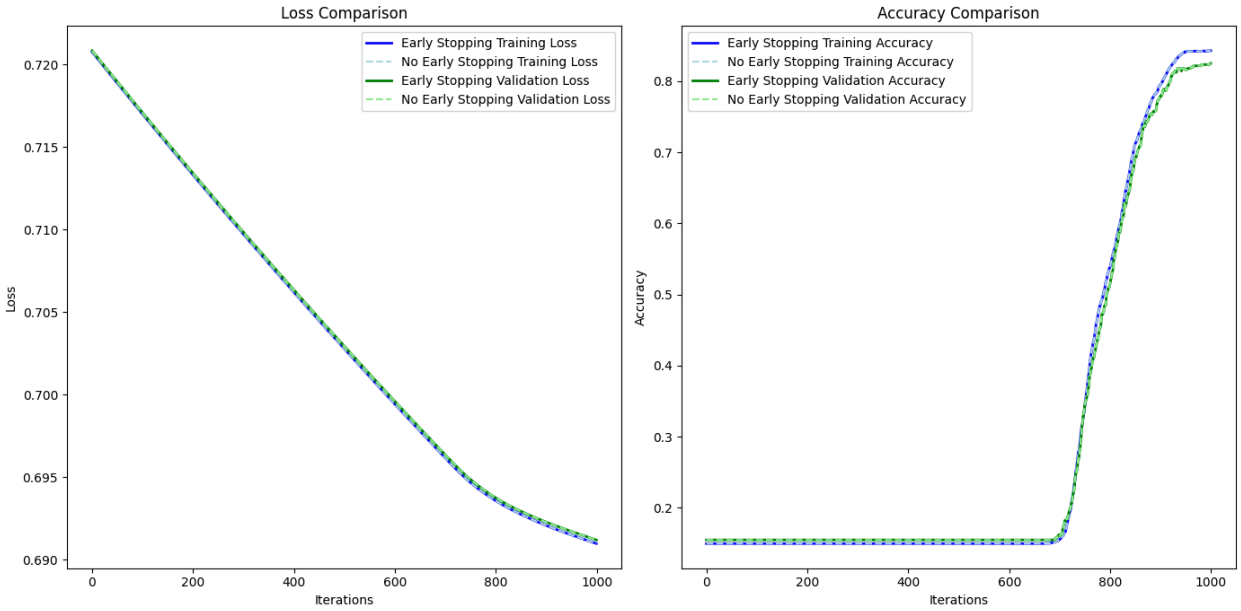
L2 (Ridge) Regularization

1. Learning rate : 0.00012



```
Early Stopping Metrics:
F1 Score: 0.044
Recall: 0.024
Precision: 0.333
ROC-AUC Score: 0.422

No Early Stopping Metrics:
F1 Score: 0.044
Recall: 0.024
Precision: 0.333
ROC-AUC Score: 0.422
```

2. Learning rate : 0.0001



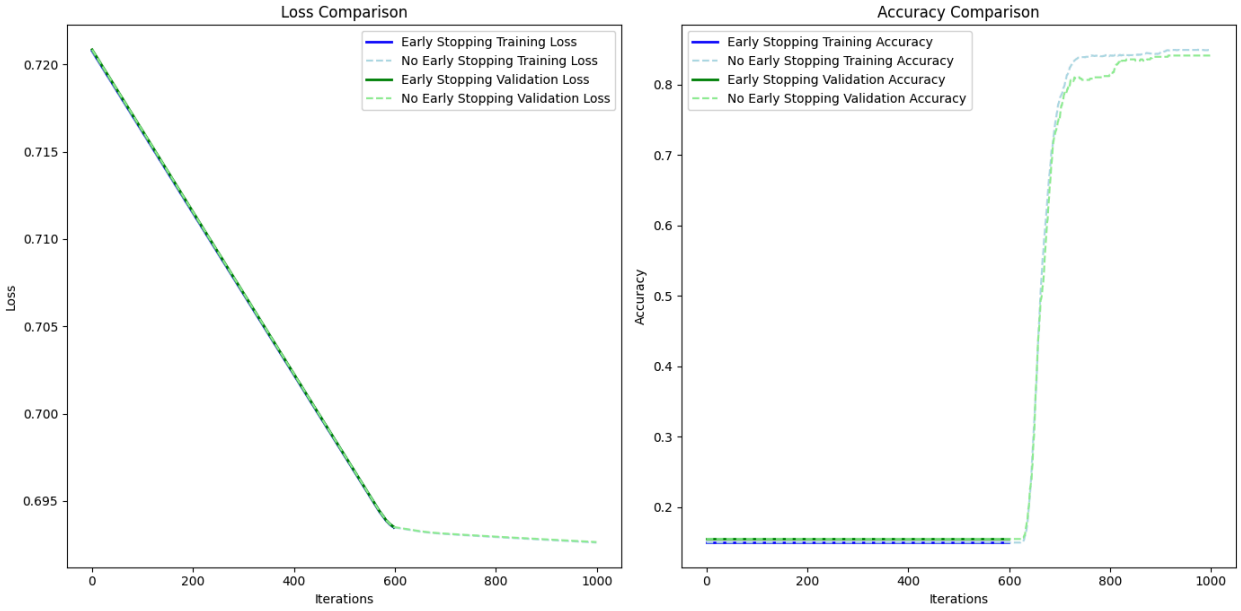

```
Early Stopping Metrics:
F1 Score: 0.094
Recall: 0.059
Precision: 0.238
ROC-AUC Score: 0.442

No Early Stopping Metrics:
F1 Score: 0.094
Recall: 0.059
Precision: 0.238
ROC-AUC Score: 0.442
```

Conclusion :

The lack of improvement with early stopping and L2 regularization suggests that the model's performance issues are not being addressed by additional training.

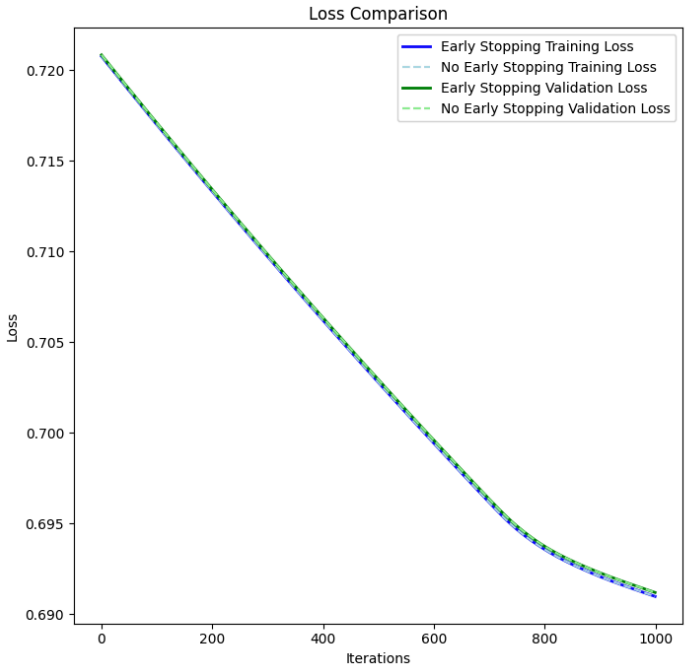
L1 (Lasso) Regularization

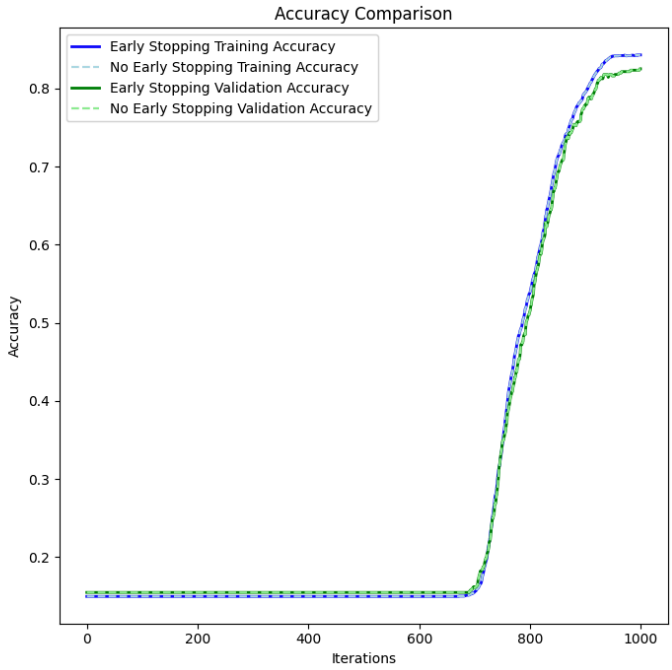
1. Learning rate : 0.000015




```
Early Stopping Metrics:
F1 Score: 0.094
Recall: 0.059
Precision: 0.238
ROC-AUC Score: 0.442

No Early Stopping Metrics:
F1 Score: 0.094
Recall: 0.059
Precision: 0.238
ROC-AUC Score: 0.442
```

2. Learning rate : 0.00001






```
Early Stopping Metrics:
F1 Score: 0.094
Recall: 0.059
Precision: 0.238
ROC-AUC Score: 0.442

No Early Stopping Metrics:
F1 Score: 0.094
Recall: 0.059
Precision: 0.238
ROC-AUC Score: 0.442
```

Conclusion :

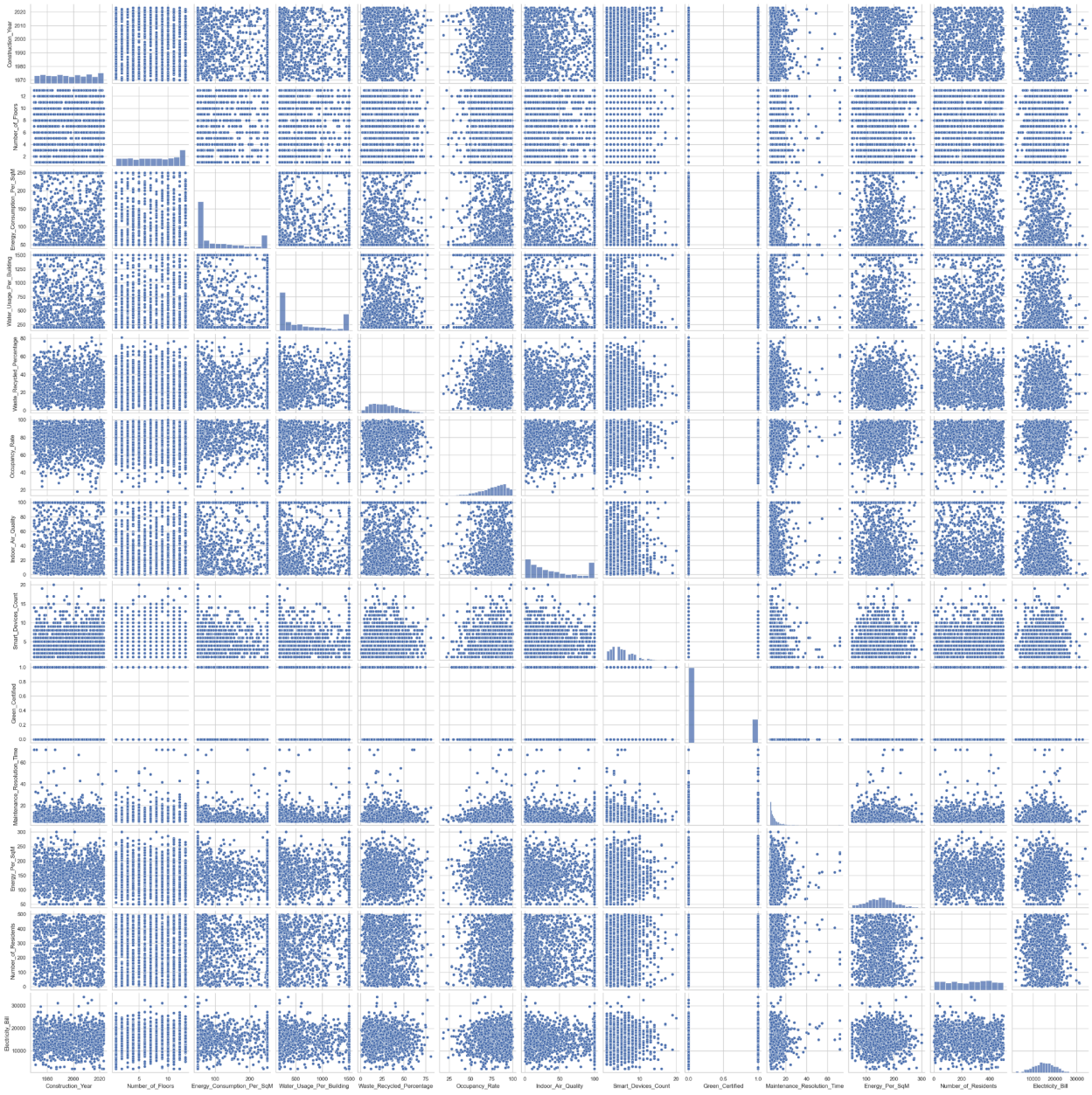
The lack of improvement with early stopping and L1 regularization suggests that the model's performance issues are not being addressed by additional training.

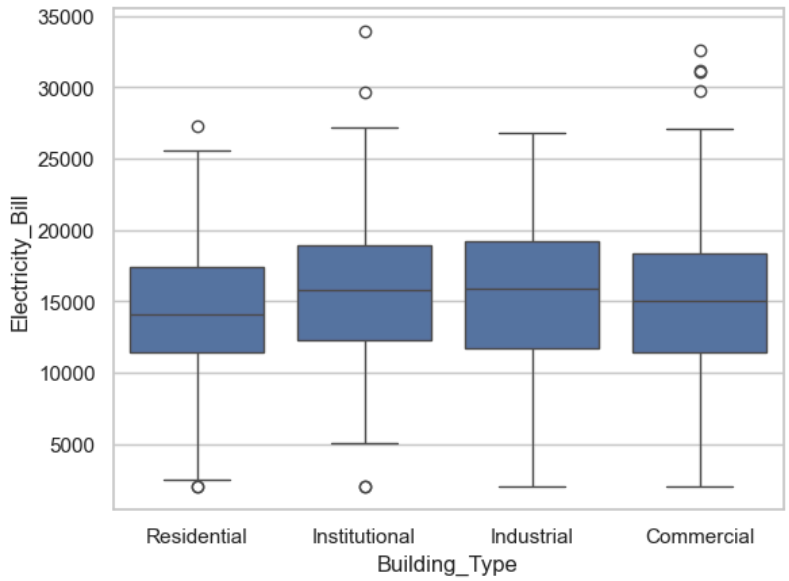
Final verdict on Early stopping using L1 and L2 Regularization :

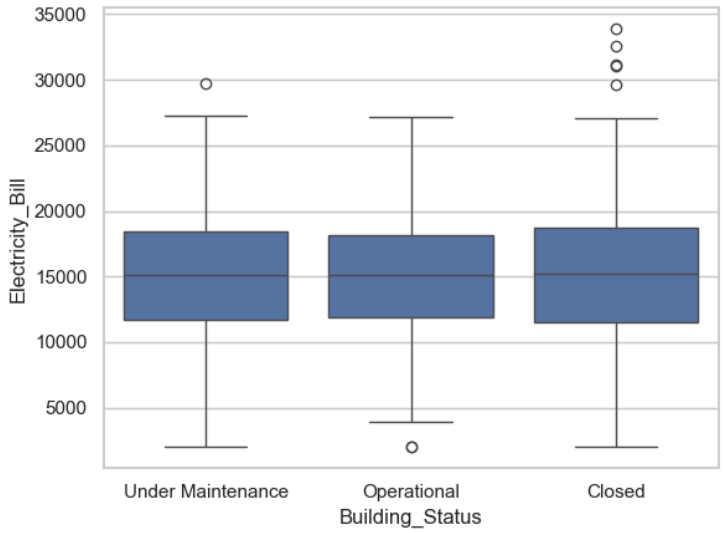
Early stopping combined with L1 and L2 regularization did not yield significant improvements in model performance. Despite incorporating these techniques to address overfitting and promote generalization, the results remained largely unchanged, suggesting that the underlying model or data may need further refinement for better performance.

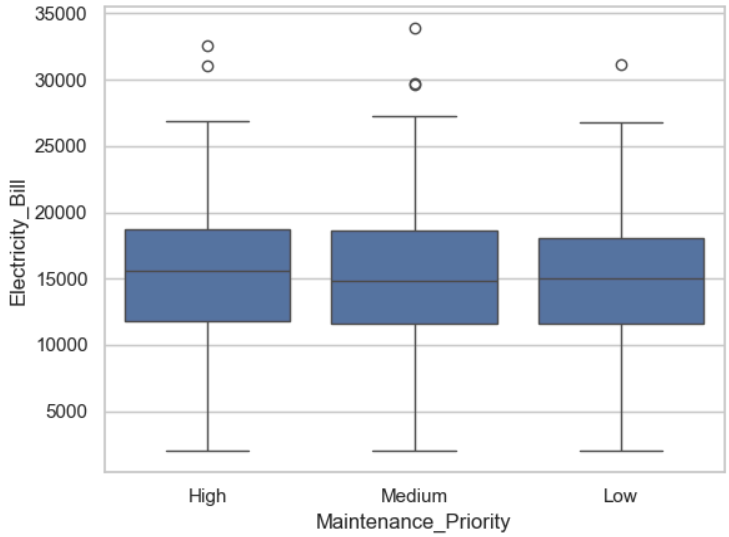
Section C

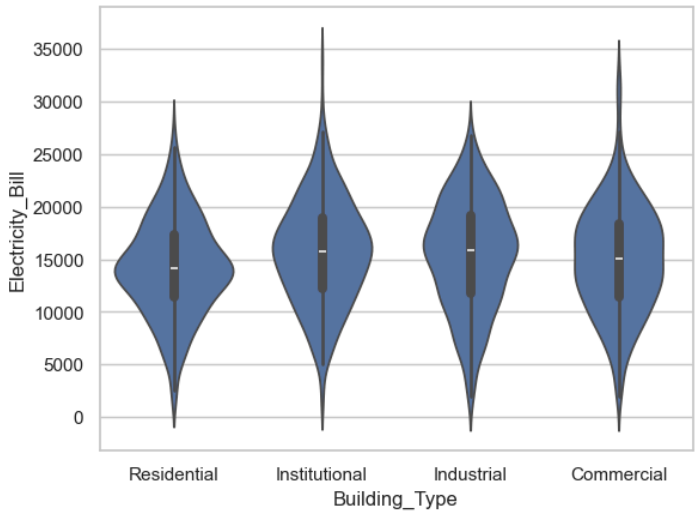
(a) (2.5 marks) Perform EDA by creating pair plots, box plots, violin plots, count plots for categorical features, and a correlation heatmap. Based on these visualizations, provide at least five insights on the dataset.

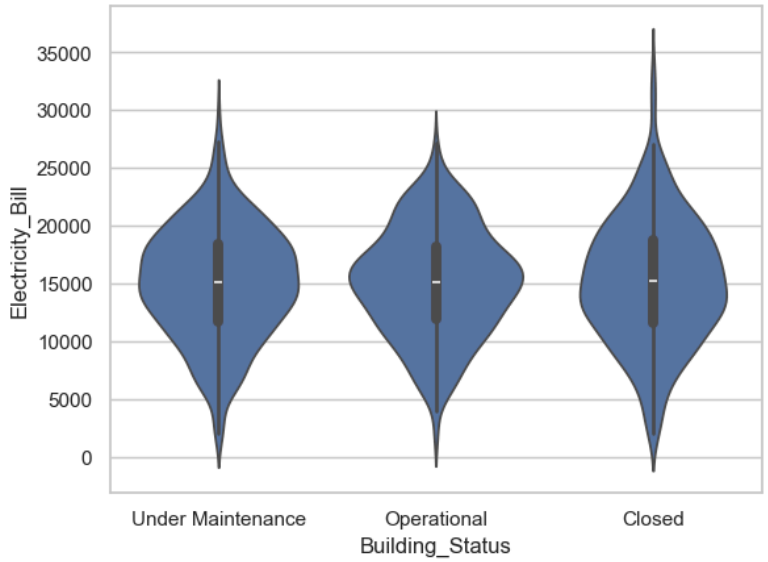


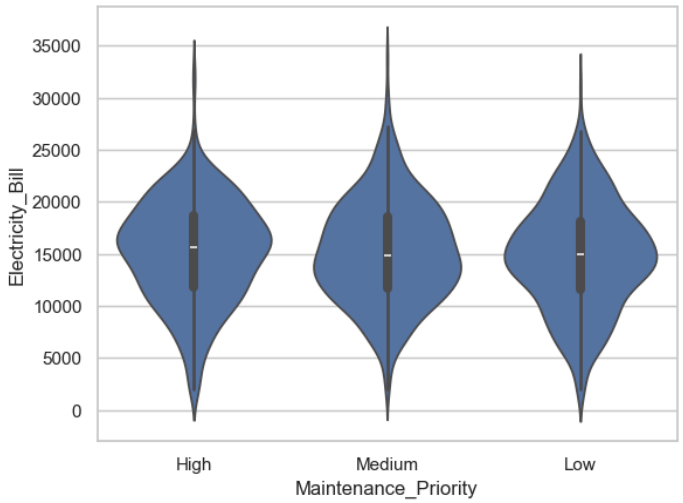

35000
30000
25000
20000
15000
10000
5000
Electricity_Bill
Residential
Institutional
Industrial
Commercial
Building_Type

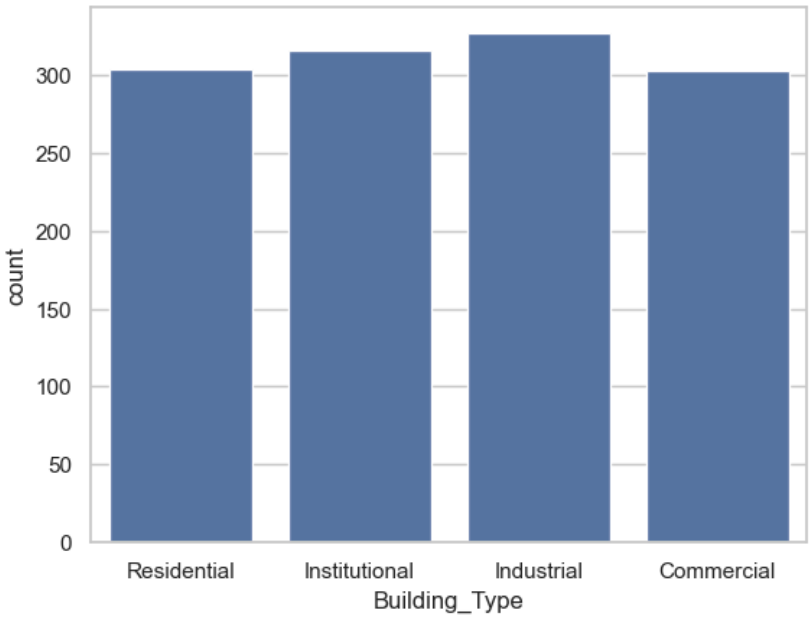

35000
30000
25000
20000
15000
10000
5000
Electricity_Bill
Under Maintenance
Operational
Closed
Building_Status

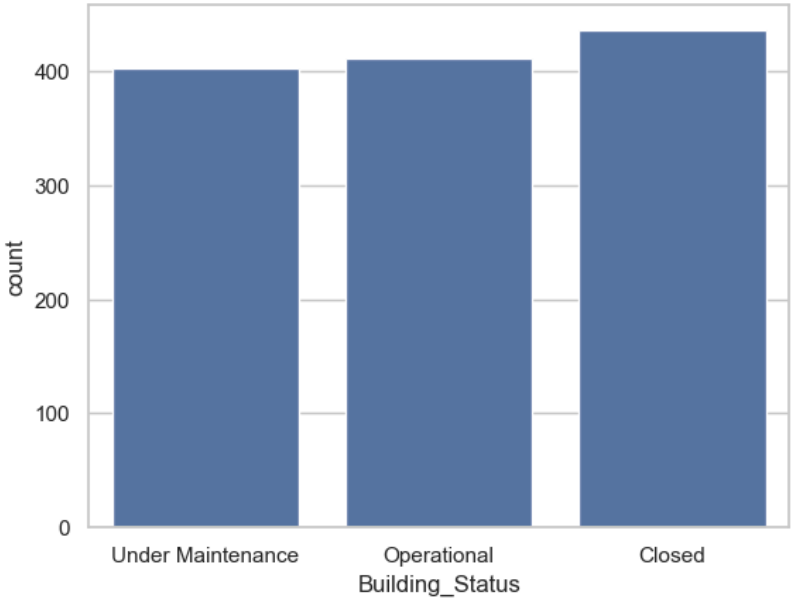

35000
30000
25000
20000
15000
10000
5000
Electricity_Bill
High
Medium
Low
Maintenance_Priority

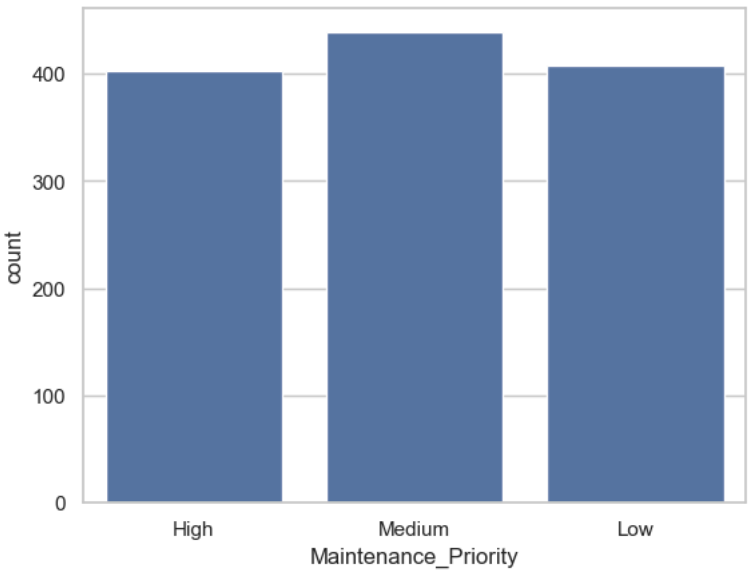

35000
30000
25000
20000
15000
10000
5000
0
Electricity_Bill
Residential
Institutional
Industrial
Commercial
Building_Type

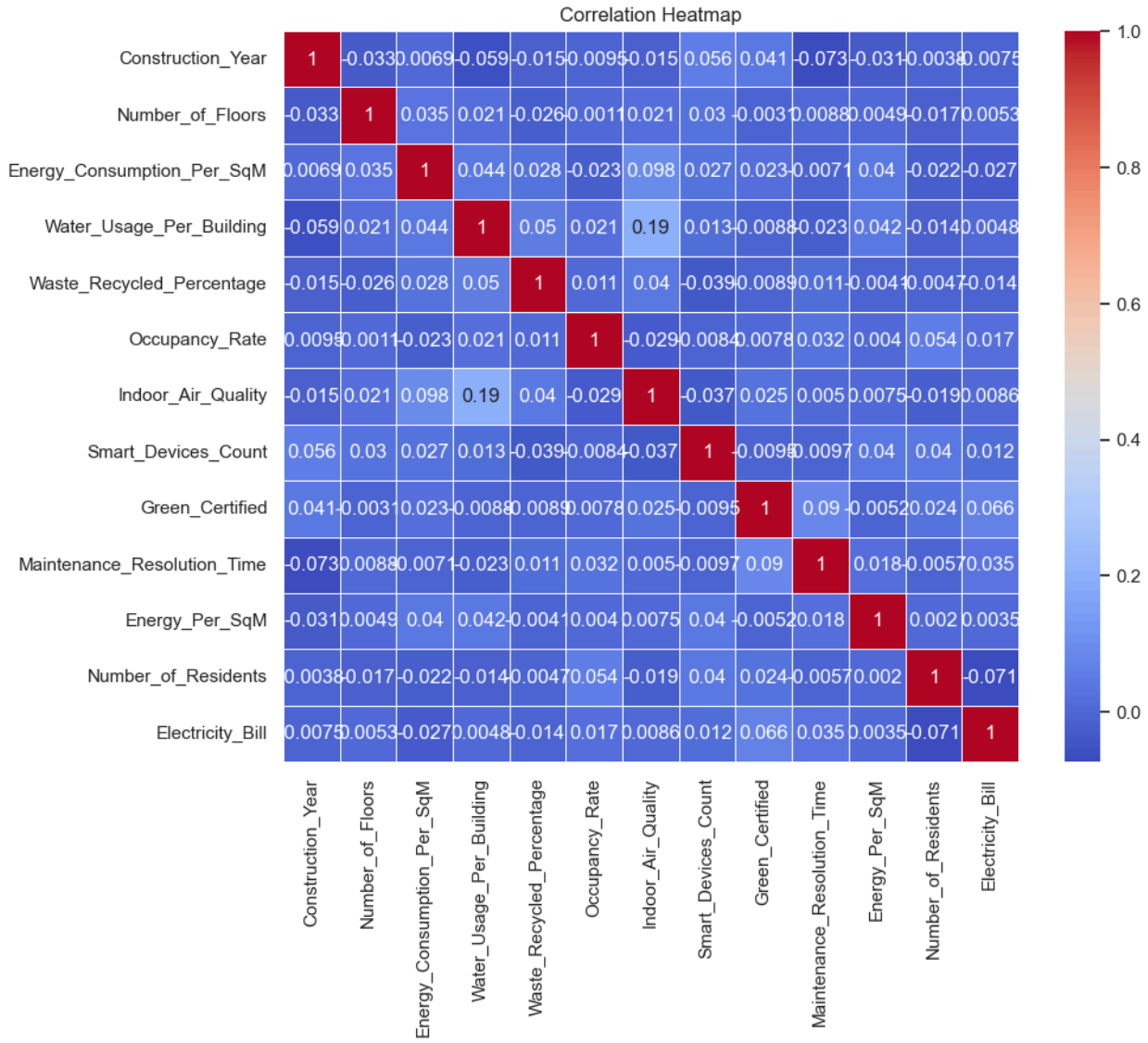

35000
30000
25000
20000
15000
10000
5000
0
Electricity_Bill
Under Maintenance
Operational
Closed
Building_Status


35000
30000
25000
20000
15000
10000
5000
0
Electricity_Bill
High
Medium
Low
Maintenance_Priority


300
250
200
150
100
50
0
count
Residential
Institutional
Industrial
Commercial
Building_Type

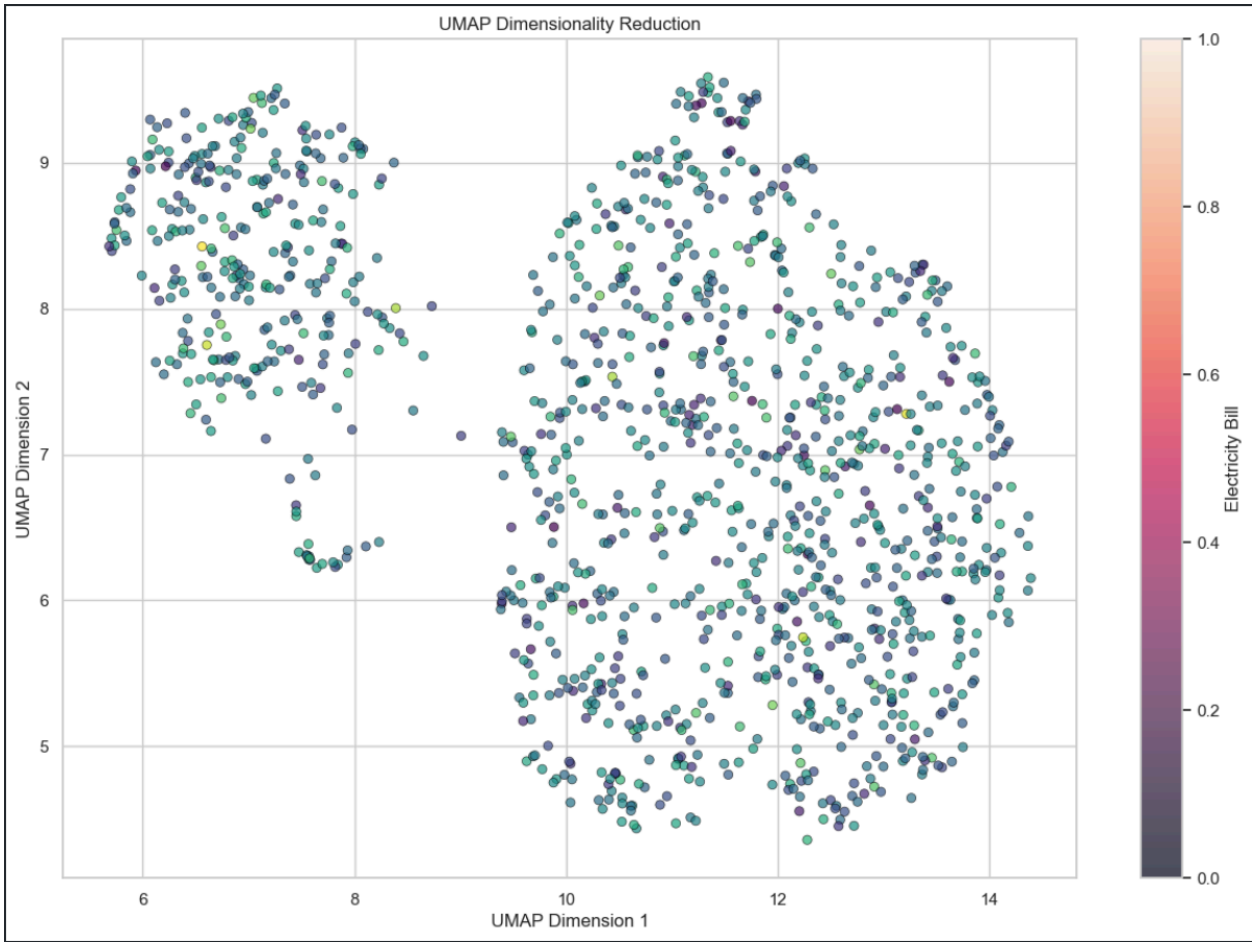

400
300
200
100
0
count
Under Maintenance
Operational
Closed
Building_Status


400
300
200
100
0
count
High
Medium
Low
Maintenance_Priority



Numerous noteworthy correlations between the variables are shown by the correlation analysis. Electricity bills are closely associated with energy consumption, yet indoor air quality may be positively impacted by water usage. The existence of smart devices and age (year of construction) do not appear to have a significant influence on the factors taken into account. better occupancy rates are typically correlated with better waste recycling percentages in buildings, and green certification is frequently linked to faster maintenance resolution times.

(b) (1 marks)Use the Uniform Manifold Approximation and Projection (UMAP) algorithm to reduce the data dimensions to 2 and plot the resulting data as a scatter plot. Comment on the separability and clustering of the data after dimensionality reduction.



1. Two Different Clusters: Two distinct clusters formed by the data points indicate the presence of two underlying patterns or groups in the dataset.
2. Electricity Bill Distribution: There is no obvious distinction between high and low electricity bills depending on cluster membership; instead, the color gradient indicating the electricity bill is distributed throughout both clusters.
3. Intra-cluster fluctuation: A variety of colors represent the large fluctuation in the electricity bill inside both clusters. This implies that other factors may have an impact on the electricity bill even among similar groups.
4. Sparse Middle Region: There aren't many data points in the region between the clusters, suggesting that the two groups don't overlap. This may indicate a clear feature difference between both groups in the dataset.
5. Gradual Color Change: The values of the electrical bills transition gradually in most places, suggesting that changes in the bills are not sudden but rather widely distributed, possibly due to continuous variables.

Conclusion: With two primary clusters and separate feature sets, the dataset may be efficiently grouped. The feature that deals with power bills, however, seems to differ between the two groups, indicating that factors other than the primary ones that drive the clustering might have a

greater impact on electricity expenses. This necessitates more research into the particular characteristics that influence the variation in electricity bills among each group.

(c) (2.5 marks) Perform the necessary pre-processing steps, including handling missing values and normalizing numerical features. For categorical features, use LabelEncoding. Apply Linear Regression on the preprocessed data. Report Mean Squared Error (MSE), Root Mean Squared Error (RMSE), R2 score, Adjusted R2 score, and Mean Absolute Error (MAE) on the train and test data.

```
Training Set Metrics:
Mean Squared Error (MSE): 24475013.168
Root Mean Squared Error (RMSE): 4947.223
R2 Score: 0.014
Adjusted R2 Score: -0.001
Mean Absolute Error (MAE): 4006.328

Test Set Metrics:
Mean Squared Error (MSE): 24278016.156
Root Mean Squared Error (RMSE): 4927.273
R2 Score: 0.000
Adjusted R2 Score: -0.064
Mean Absolute Error (MAE): 3842.409
```

The model's performance is poor, as indicated by the low R^2 scores (close to zero) and high error metrics (MSE, RMSE, MAE) on both the training and test sets. This suggests that the model is not effectively capturing the patterns in the data, performing only marginally better than guessing the average. The negative adjusted R^2 on the test set implies that adding features may have worsened the model, indicating potential overfitting or irrelevant features. Overall, the model requires improvement, either by tuning, selecting better features, or trying a different algorithm.

(d) (2 marks) Perform Recursive Feature Elimination (RFE) or Correlation analysis on the original dataset to select the 3 most important features. Train the regression model using the selected features. Compare the results (MSE, RMSE, R2 score, Adjusted R2 score, MAE) on the train and test dataset with the results obtained in part (c).

```
Top 3 selected features using RFE: Index(['Building_Type', 'Green_Certified', 'Building_Status'], dtype='object')

Results using RFE-selected features:
Training Set (RFE) Metrics:
Mean Squared Error (MSE): 24673540.312
Root Mean Squared Error (RMSE): 4967.247
R2 Score: 0.006
Adjusted R2 Score: 0.003
Mean Absolute Error (MAE): 4006.784

Test Set (RFE) Metrics:
Mean Squared Error (MSE): 24181190.647
Root Mean Squared Error (RMSE): 4917.437
R2 Score: 0.004
Adjusted R2 Score: -0.008
Mean Absolute Error (MAE): 3825.652
```

Training Set (All Features vs. RFE-Selected Features):
- MSE: 24,475,013 (All) vs. 24,673,540 (RFE) – Slightly worse with RFE.
- RMSE: 4947.2 (All) vs. 4967.2 (RFE) – Almost the same.
- R^2: 0.014 (All) vs. 0.006 (RFE) – Slightly worse with RFE.
- Adjusted R^2: -0.001 (All) vs. 0.003 (RFE) – Slightly better with RFE.
- MAE: 4006.3 (All) vs. 4006.8 (RFE) – Very similar.

Test Set (All Features vs. RFE-Selected Features):
- MSE: 24,278,016 (All) vs. 24,181,190 (RFE) – **Slightly better with RFE.**
- RMSE: 4927.3 (All) vs. 4917.4 (RFE) – **Slightly better with RFE.**
- R^2: 0.000 (All) vs. 0.004 (RFE) – **Slightly better with RFE.**
- Adjusted R^2: -0.064 (All) vs. -0.008 (RFE) – **Better with RFE.**
- MAE: 3842.4 (All) vs. 3825.7 (RFE) – **Slightly better with RFE.**

Conclusion:
Using the top 3 features selected by RFE gives similar results to using all features, but slightly better performance on the test set. It suggests that focusing on key features can simplify the model without losing accuracy.

(e) (2 marks) Encode the categorical features of the original dataset using One-Hot Encoding and perform Ridge Regression on the preprocessed data. Report the evaluation metrics (MSE, RMSE, R2 score, Adjusted R2 score, MAE). Compare the results with those obtained in part (c)

```
Results using Ridge Regression with One-Hot Encoded Features:
Training Set (Ridge) Metrics:
  Mean Squared Error (MSE):     24188933.389
  Root Mean Squared Error (RMSE): 4918.225
  R2 Score:                      0.025
  Adjusted R2 Score:             0.004
  Mean Absolute Error (MAE):     3976.677

Test Set (Ridge) Metrics:
  Mean Squared Error (MSE):     24130064.924
  Root Mean Squared Error (RMSE): 4912.236
  R2 Score:                      0.006
  Adjusted R2 Score:             -0.090
  Mean Absolute Error (MAE):     3797.607

Comparison with Original Model Metrics:

Original Model Metrics:
Training Set (Original) Metrics:
  Mean Squared Error (MSE):     24475013.168
  Root Mean Squared Error (RMSE): 4947.223
  R2 Score:                      0.014
  Adjusted R2 Score:             -0.001
  Mean Absolute Error (MAE):     4006.328

Test Set (Original) Metrics:
  Mean Squared Error (MSE):     24278016.156
  Root Mean Squared Error (RMSE): 4927.273
  R2 Score:                      0.000
  Adjusted R2 Score:             -0.064
  Mean Absolute Error (MAE):     3842.409
```

- **MSE & RMSE:** Ridge Regression with one-hot encoding shows a slight improvement in both training and test set metrics compared to the original model.
- **R^2 Score:** The Ridge Regression model has a marginally better R^2 score on the training set and test set, but both models still show low R^2 values.
- **Adjusted R^2 Score:** Ridge Regression improves the Adjusted R^2 score on the training set but performs worse on the test set compared to the original model.
- **MAE:** Ridge Regression slightly reduces the MAE on both training and test sets compared to the original model.

Conclusion: Ridge Regression with one-hot encoded features performs slightly better than the original model in terms of MSE, RMSE, and MAE on both training and test sets. However, both models exhibit similar R^2 scores, indicating that neither model explains a significant amount of variance in the data.

(f) (2 marks) Perform Independent Component Analysis (ICA) on the one-hot encoded dataset and choose the appropriate number of components (try 4, 5, 6, and 8 components). Compare the results (MSE, RMSE, R2 score, Adjusted R2 score, MAE) on the train and test dataset.

```
Number of Components: 4
  Train MSE: 24701058.640
  Train RMSE: 4970.016
  Train R2: 0.005
  Train Adjusted R2: 0.001
  Train MAE: 4010.995
  Test MSE: 24167148.348
  Test RMSE: 4916.009
  Test R2: 0.005
  Test Adjusted R2: -0.012
  Test MAE: 3818.895

Number of Components: 5
  Train MSE: 24683781.209
  Train RMSE: 4968.277
  Train R2: 0.006
  Train Adjusted R2: 0.001
  Train MAE: 4008.443
  Test MSE: 24261530.740
  Test RMSE: 4925.600
  Test R2: 0.001
  Test Adjusted R2: -0.020
  Test MAE: 3831.502
```

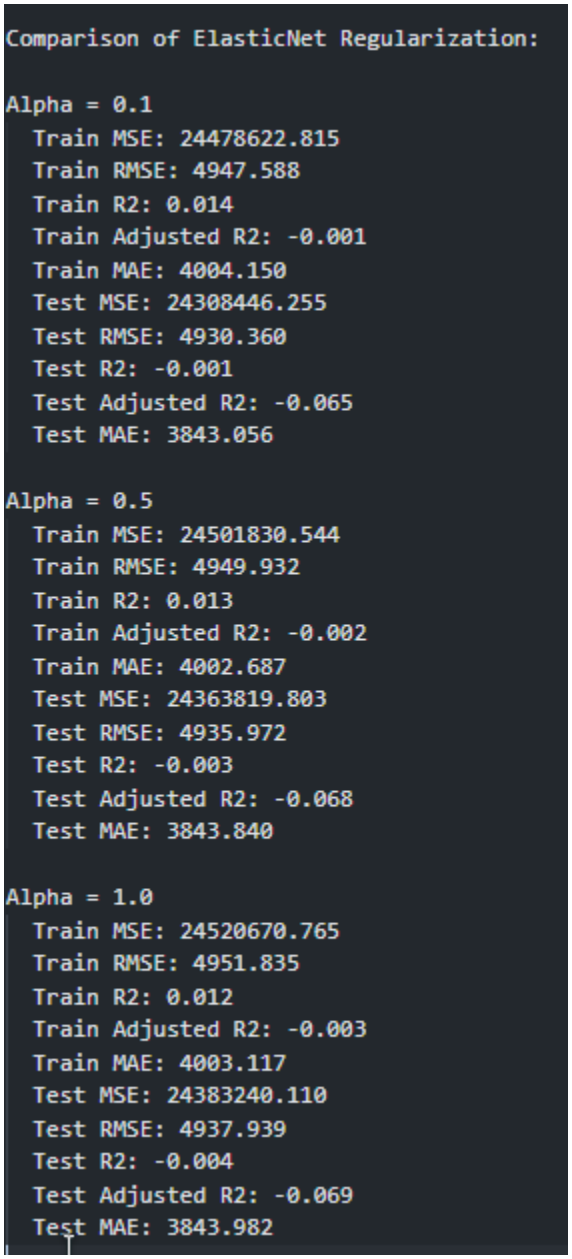
```
Number of Components: 6
  Train MSE: 24682728.783
  Train RMSE: 4968.172
  Train R2: 0.006
  Train Adjusted R2: -0.000
  Train MAE: 4009.407
  Test MSE: 24253844.061
  Test RMSE: 4924.819
  Test R2: 0.001
  Test Adjusted R2: -0.024
  Test MAE: 3829.863

Number of Components: 8
  Train MSE: 24674426.719
  Train RMSE: 4967.336
  Train R2: 0.006
  Train Adjusted R2: -0.002
  Train MAE: 4009.046
  Test MSE: 24222154.151
  Test RMSE: 4921.601
  Test R2: 0.002
  Test Adjusted R2: -0.031
  Test MAE: 3830.142
```

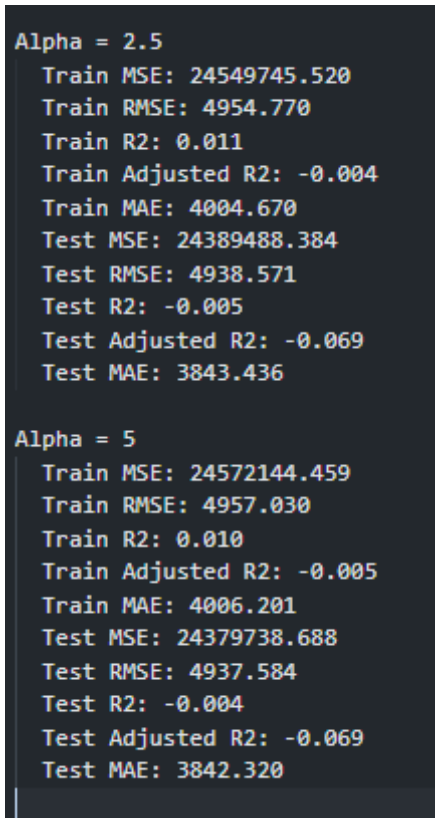
- **MSE & RMSE:** The model's performance improves slightly with more components, especially on the test set, with 8 components showing the lowest MSE and RMSE.
- **R^2 Score:** All component configurations yield similar R^2 scores, indicating minimal improvement in the model's ability to explain variance.
- **Adjusted R^2 Score:** The adjusted R^2 scores decrease with more components, suggesting that adding more components may not be beneficial for improving model performance.
- **MAE:** The MAE shows a slight improvement with more components, with 8 components giving the lowest MAE on the test set.

Conclusion: Using 8 components in ICA provides the best performance in terms of MSE, RMSE, and MAE on the test set. However, the R^2 and Adjusted R^2 scores are quite similar across different numbers of components, suggesting limited improvement in the explanatory power of the model.

(g) (1.5 marks) Use ElasticNet regularization (which combines L1 and L2) while training a linear model on the preprocessed dataset from part (c). Compare the evaluation metrics (MSE, RMSE,

R2 score, Adjusted R2 score, MAE) on the test dataset for different values of the mixing parameter (alpha).





- **MSE & RMSE:** The metrics slightly worsen with increasing alpha. The best performance on the test set in terms of MSE and RMSE is observed with alpha = 0.1.
- **R² Score:** All alpha values yield similar R² scores, which are negative, indicating poor performance in explaining the variance.
- **Adjusted R² Score:** The adjusted R² scores are also negative and show a slight worsening with higher alpha values.
- **MAE:** The MAE is lowest for alpha = 5 on the test set, indicating slightly better performance in terms of absolute errors.

Conclusion: ElasticNet regularization with alpha = 0.1 provides the best overall performance in terms of MSE and RMSE on the test set. However, the performance is generally similar across different alpha values, with small variations in MAE and R^2 scores.

(h) (1.5 marks) Use the Gradient Boosting Regressor to perform regression on the preprocessed dataset from part (c). Report the evaluation metrics (MSE, RMSE, R2 score, Adjusted R2 score, MAE). Compare the results with those obtained in parts (c) and (g).

```
Training Set (Gradient Boosting) Metrics:
  Mean Squared Error (MSE):     14926446.257
  Root Mean Squared Error (RMSE): 3863.476
  R2 Score:                      0.399
  Adjusted R2 Score:             0.389
  Mean Absolute Error (MAE):     3092.748

Test Set (Gradient Boosting) Metrics:
  Mean Squared Error (MSE):     24420565.348
  Root Mean Squared Error (RMSE): 4941.717
  R2 Score:                      -0.006
  Adjusted R2 Score:             -0.070
  Mean Absolute Error (MAE):     3819.610
```

- **MSE & RMSE:** Gradient Boosting Regressor shows a significant improvement in MSE and RMSE on the training set compared to Ridge Regression and ElasticNet. However, the test set performance is slightly worse than Ridge Regression, with a higher RMSE.
- **R^2 Score:** Gradient Boosting has a much higher R^2 score on the training set compared to other methods, indicating better fit to the training data. However, its R^2 score on the test set is negative, similar to the other models, indicating poor predictive performance.
- **Adjusted R^2 Score:** The Gradient Boosting model performs better on the training set in terms of adjusted R^2, but still has a negative adjusted R^2 score on the test set.
- **MAE:** Gradient Boosting achieves a lower MAE on the training set compared to the other methods but is slightly higher on the test set compared to Ridge Regression.

Conclusion: The Gradient Boosting Regressor shows strong performance on the training set with better MSE, RMSE, and MAE compared to Ridge Regression and ElasticNet. However, its performance on the test set is still not optimal, with higher error metrics compared to Ridge Regression. This suggests that while Gradient Boosting can fit the training data well, it may not generalize as effectively to unseen data compared to simpler models like Ridge Regression.