MACHINE LEARNING

ASSIGNMENT 1

NAME : HIMANSHU KUMAR

ROLL NUMBER : 2022215

PART-1(Bias-Variance Tradeoff in Machine Learning)

Impact of Increasing Model Complexity on Bias and Variance

As we increase the complexity of a machine-learning model by adding more features or including higher-order polynomial terms in a regression model, the effects on bias and variance can be described through the **bias-variance tradeoff**.

1. Bias

Bias refers to the error introduced by approximating a real-world problem with a simpler model.

- Low-complexity models (e.g., linear models) typically have **high bias**, as they underfit the data and fail to capture underlying patterns. - As model complexity increases, bias **decreases**, since the model can fit more complex patterns.

2. Variance

Variance refers to the model's sensitivity to small fluctuations in the training data.

- High-complexity models, especially those with many features or higher-order terms, have **high variance** since they can overfit the training data, learning patterns that do not generalize well to unseen data. - As model complexity increases, variance **increases**, as the model becomes too sensitive to the specific training data.

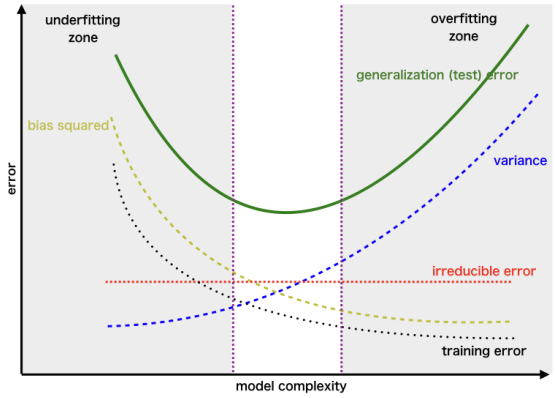
3. Bias-Variance Tradeoff

The bias-variance tradeoff represents the balance between bias and variance:

- Initially, as complexity increases, both training and test errors decrease due to a reduction in bias.
- After a certain point, increasing complexity results in overfitting, which causes an increase in test error due to higher variance.

Graphical Representation

The bias-variance tradeoff can be represented graphically as shown below:



In this graph: - The **x-axis** represents model complexity. - The **y-axis** represents the error (or loss). - The **training error** decreases as complexity increases. - The **test error** initially decreases but then increases due to overfitting. - **Bias** starts high and decreases, while **variance** starts low and increases.

Mathematical Formulation

The total error can be decomposed into three parts:

$$\text{Total Error} = \underbrace{\text{Bias}^2}_{\text{underfitting}} + \underbrace{\text{Variance}}_{\text{overfitting}} + \underbrace{\text{Irreducible Error}^2}_{\text{Noise}}$$

- **Bias**: Measures how far the model's predictions are from the true values. - **Variance**: Measures the model's sensitivity to changes in the training data. - **Irreducible error**: Represents the inherent noise in the data.

PART-2(Email Filtering System Evaluation)

Confusion Matrix

Based on the problem description, we can summarize the classification results in the following confusion matrix:

	Predicted Spam	Predicted Legitimate
Actual Spam	True Positives (TP) = 200	False Negatives (FN) = 50
Actual Legitimate	False Positives (FP) = 20	True Negatives (TN) = 730

1. Accuracy

Accuracy is the proportion of correctly classified emails (both spam and legitimate) out of the total number of emails:

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

Substituting the values:

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

Thus, the model's accuracy is **93%**.

2. Precision

Precision measures how many of the emails classified as spam were actually spam:

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

Substituting the values:

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

So, the model's precision is **90.9%**.

3. Recall

Recall (also known as Sensitivity or True Positive Rate) measures how many actual spam emails were correctly classified:

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

Substituting the values:

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

Thus, the model's recall is **80%**.

4. F1-Score

The F1-score is the harmonic mean of precision and recall:

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

Substituting the values:

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

Thus, the F1-score is **85.1%**.

5. False Positive Rate

The false positive rate measures how many legitimate emails were incorrectly classified as spam:

$$\text{FPR} = \frac{FP}{FP + TN}$$

Substituting the values:

$$\text{FPR} = \frac{20}{20 + 730} = \frac{20}{750} \approx 0.0267$$

Thus, the false positive rate is **2.67%**.

Conclusion

The email filtering system has the following performance metrics:

- **Accuracy**: 93%
- **Precision**: 90.9%

- **Recall**: 80%
- **F1-Score**: 85.1%
- **False Positive Rate**: 2.67%

The model performs well overall, but there is a tradeoff between precision and recall. Only a small portion of legitimate emails are mistakenly flagged as spam, but the model could improve its ability to correctly classify all spam emails.

PART-3(Finding the Equation of the Regression Line)

Given Data

The given data is as follows:

x	y
3	15
6	30
10	55
15	85
18	100

The number of data points $n = 5$.

Step 1: Calculate the Required Sums

We need to calculate the following sums:

$$\sum x = 3 + 6 + 10 + 15 + 18 = 52$$

$$\sum y = 15 + 30 + 55 + 85 + 100 = 285$$

$$\sum xy = (3\times15)+(6\times30)+(10\times55)+(15\times85)+(18\times100) = 45+180+550+1275+1800 = 3850$$

$$\sum x^2 = (3^2) + (6^2) + (10^2) + (15^2) + (18^2) = 9 + 36 + 100 + 225 + 324 = 694$$

Step 2: Calculate the Slope (m) and Intercept (c)

The slope m of the regression line is given by:

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

Substituting the values:

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

$$m = \frac{19250 - 14820}{3470 - 2704} = \frac{4430}{766} \approx 5.78$$

The y-intercept c is calculated using the formula:

$$c = \frac{\sum y - m \sum x}{n}$$

Substituting the values:

$$c = \frac{285 - 5.78 \times 52}{5} = \frac{285 - 300.56}{5} = \frac{-15.56}{5} \approx -3.11$$

Step 3: The Equation of the Regression Line

Thus, the equation of the regression line is:

$$y = 5.78x - 3.11$$

Step 4: Using the Regression Line

To predict the value of y for a given x, substitute the value of x into the regression line equation. For example, for $x = 12$:

$$y = 5.78 \times 12 - 3.11 = 69.36 - 3.11 = 66.25$$

Thus, the predicted value of y when $x = 12$ is approximately 66.25.

PART-4 (Toy Example: Empirical Risk and Generalization)

Problem

Given a training dataset with features X and labels Y, let $\hat{f}(X)$ be the prediction of a model f, and let $L(\hat{f}(X), Y)$ be the loss function. Suppose we have two models, f_1 and f_2, and the empirical risk for f_1 is lower than that for f_2. We provide a toy example where model f_1 has a lower empirical risk on the training set but may not necessarily generalize better than model f_2.

Toy Example

Consider a small dataset for a regression task:

X	Y
1	1.5
2	2.0
3	2.5
4	3.5
5	5.0

We will now create two models, f_1 and f_2, and analyze their empirical risk and generalization ability.

Model f_1: High Complexity (Overfitting)

Model f_1 is a high-degree polynomial regression, such as a 4th-degree polynomial. It fits the training data perfectly, with an equation that could look like:

$$f_1(X) = 0.05X^4 - 0.3X^3 + 0.7X^2 - 0.4X + 1.5$$

The predicted values of f_1 on the training set are:

$$\hat{Y}_1 = [1.55, 1.94, 2.38, 3.12, 5.01]$$

The empirical risk, computed as the Mean Squared Error (MSE), for f_1 is:

$$MSE(f_1) = 0.033$$

Although the empirical risk is very low, this model is highly complex and likely overfits the noise in the data, which means it may not generalize well to unseen data.

Model f_2: Low Complexity (Better Generalization)

Model f_2 is a simple linear regression model, capturing the overall trend in the data. Its equation could be:

$$f_2(X) = 0.85X + 0.75$$

The predicted values of f_2 on the training set are:

$$\hat{Y}_2 = [1.60, 2.45, 3.30, 4.15, 5.00]$$

The empirical risk (MSE) for f_2 is:

$$MSE(f_2) = 0.255$$

Despite having a higher empirical risk on the training data, f_2 is likely to generalize better because it avoids overfitting and has lower variance.

Testing the Models

To further support our claim, we test both models on a new data point that was not part of the training set, say $X = 6$, where the true value of Y is $Y = 6.0$.

For model f_1:

$$f_1(6) = 0.05(6)^4 - 0.3(6)^3 + 0.7(6)^2 - 0.4(6) + 1.5 = 9.23$$

For model f_2:

$$f_2(6) = 0.85(6) + 0.75 = 5.85$$

We observe that model f_1 predicts 9.23, which is much farther from the true value of 6.0, while model f_2 predicts 5.85, which is closer to the true value.

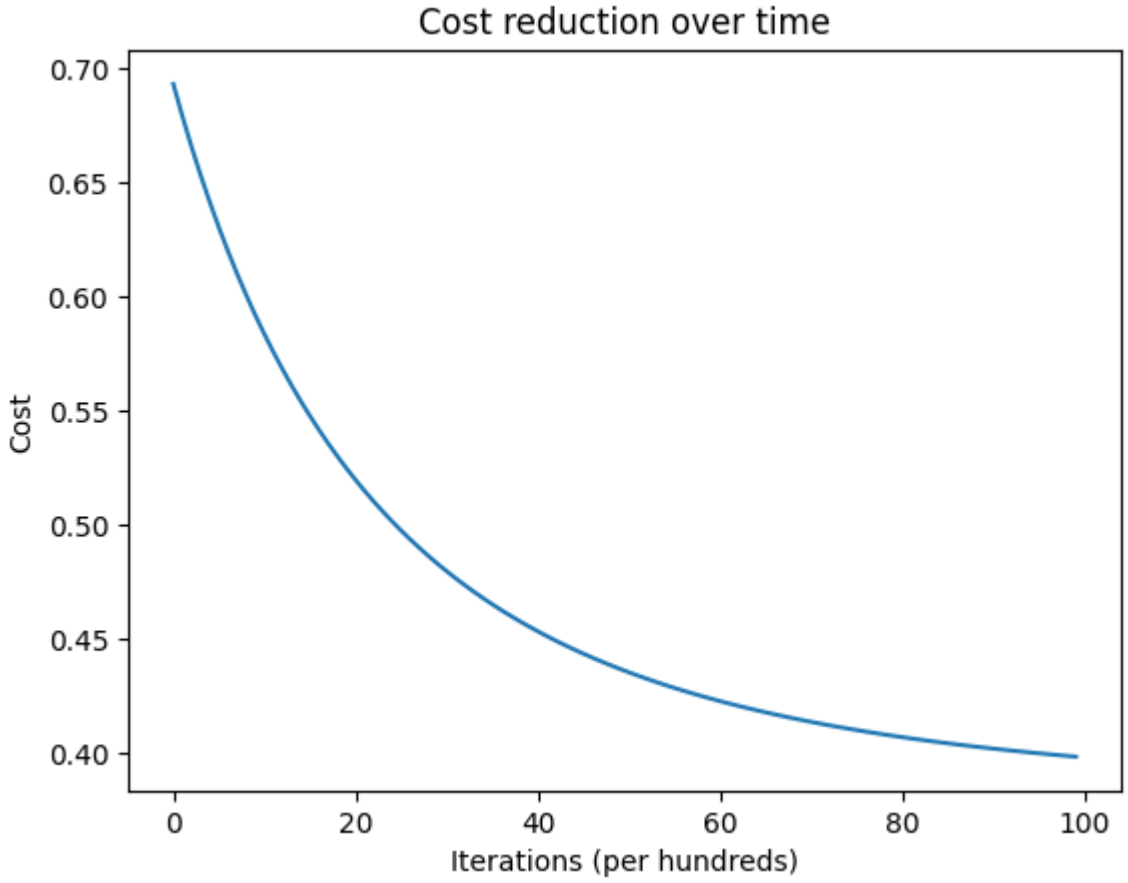
Thus, the generalization ability of f_2 is better on unseen data, even though its empirical risk on the training set is higher.

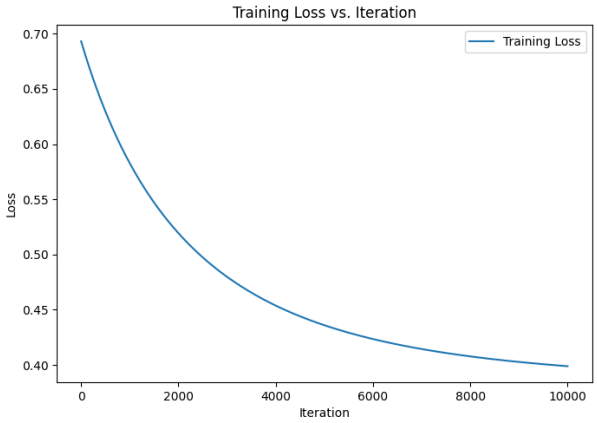
Conclusion

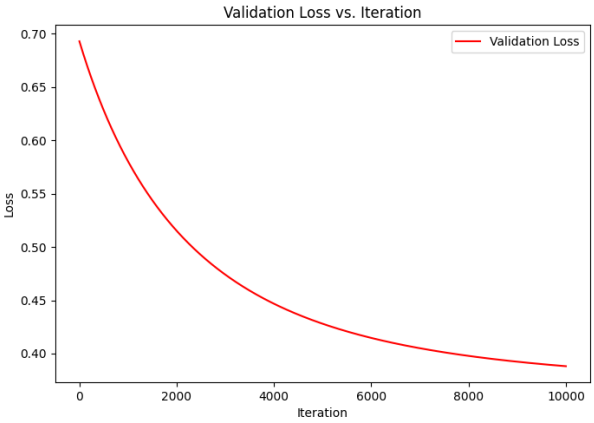
In this example, model f_1 has a lower empirical risk on the training set because it overfits the data, but it does not generalize as well as model f_2. Testing on a new data point further shows that model f_2 provides a better prediction. This illustrates that a lower training error does not necessarily imply better performance on unseen data, especially when the model is overly complex and prone to overfitting.

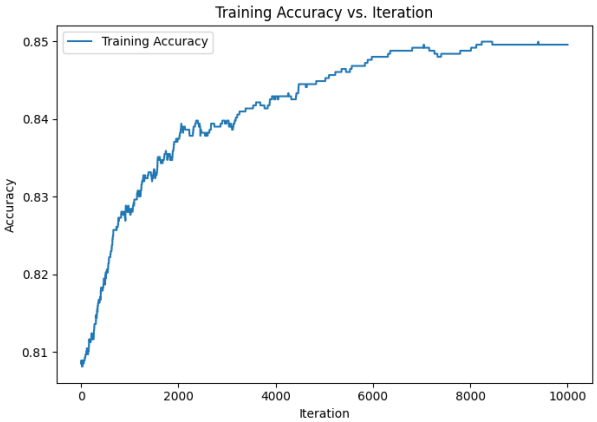
Section B (Scratch Implementation)

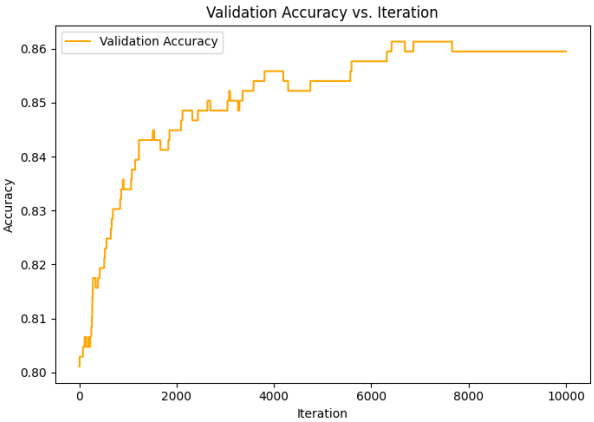
a)



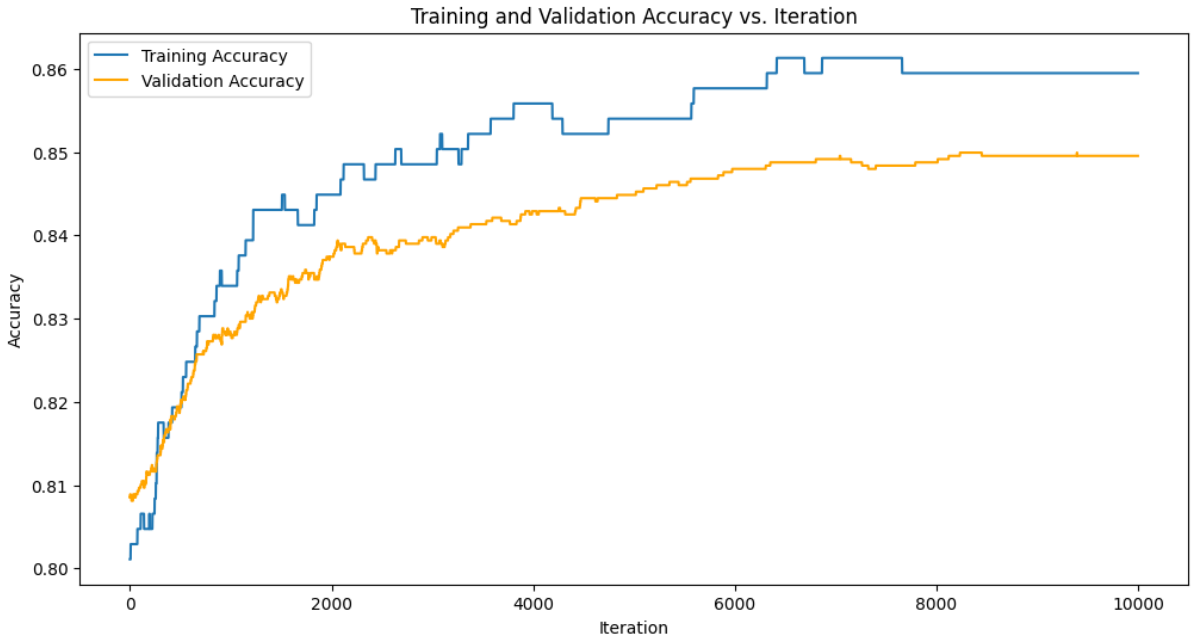

Training Loss vs. Iteration
Training Loss
Loss
Iteration


Validation Loss vs. Iteration
Validation Loss
Loss
Iteration


Training Accuracy vs. Iteration
Training Accuracy
Accuracy
Iteration


Validation Accuracy vs. Iteration
Validation Accuracy
Accuracy
Iteration


Training and Validation Loss vs. Iteration
Training Loss
Validation Loss
Loss
Iteration



1. Convergence of the Model:

Training Loss vs. Iteration:

Initial Decrease: we see a rapid decrease in training loss in the early iterations, indicating that the model is learning from the data.

Over time, the training loss should gradually decrease and eventually plateau. A well-behaved convergence is characterized by a smooth curve that flattens out, indicating that the model is approaching a minimum loss.

Validation Loss vs. Iteration:

Initial Decrease: Similar to training loss, validation loss should initially decrease as the model improves.

Overfitting Indicator: If validation loss starts increasing after a certain point while training loss continues to decrease, it suggests overfitting. This is when the model starts to memorize the training data rather than generalizing well to unseen data.

Training Accuracy vs. Iteration:

Improvement: Training accuracy should generally increase as the training progresses, reflecting better performance on the training set.

Saturation: Eventually, the training accuracy should stabilize, indicating that the model has achieved its best performance on the training data.

Validation Accuracy vs. Iteration:

Initial Improvement: Validation accuracy should initially increase, reflecting improvements in model generalization.

Plateau or Decrease: If validation accuracy plateaus or decreases after some iterations, it may indicate overfitting, where the model's performance on unseen data starts to decline.

2. Comparing and Analyzing the Plots:

Training Loss vs. Validation Loss:

Convergence: If both training and validation losses decrease and stabilize, it indicates good convergence. The difference between the two should be minimal.

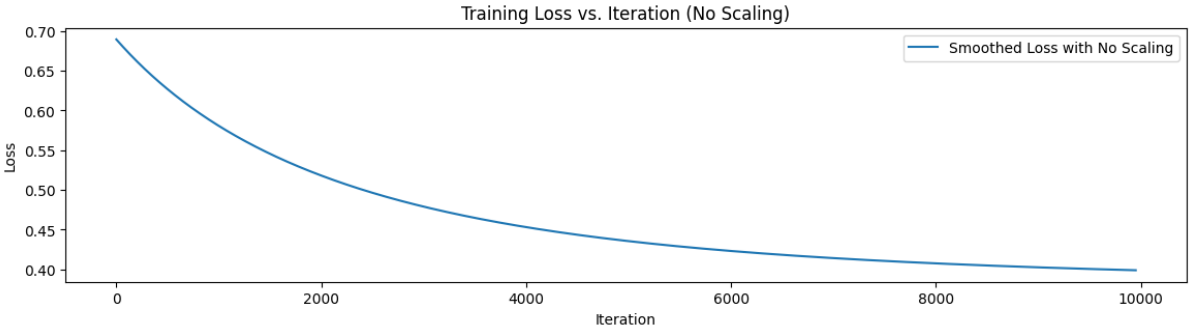
Overfitting: A significant gap where validation loss increases while training loss continues to decrease signals overfitting.

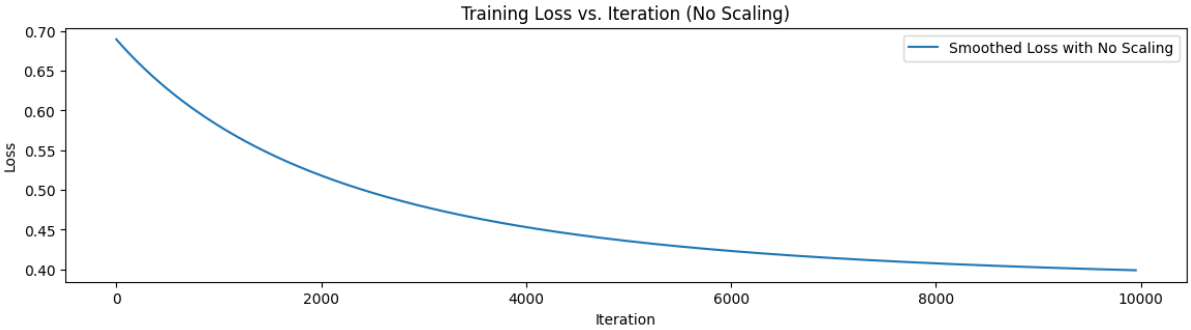
Training Accuracy vs. Validation Accuracy:

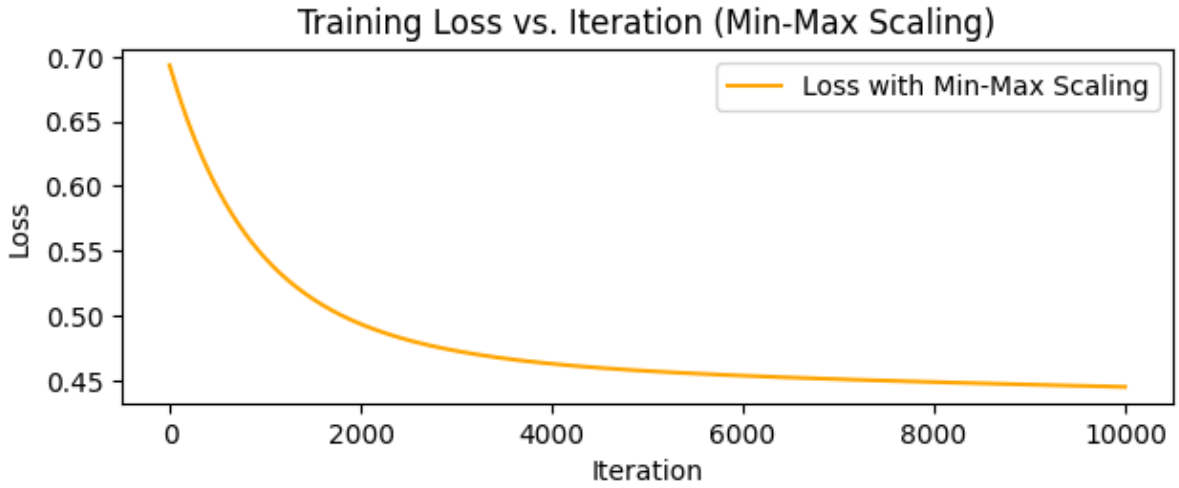
Consistency: Ideally, both training and validation accuracy should improve and stabilize. A large gap between training and validation accuracy suggests that the model might be overfitting.

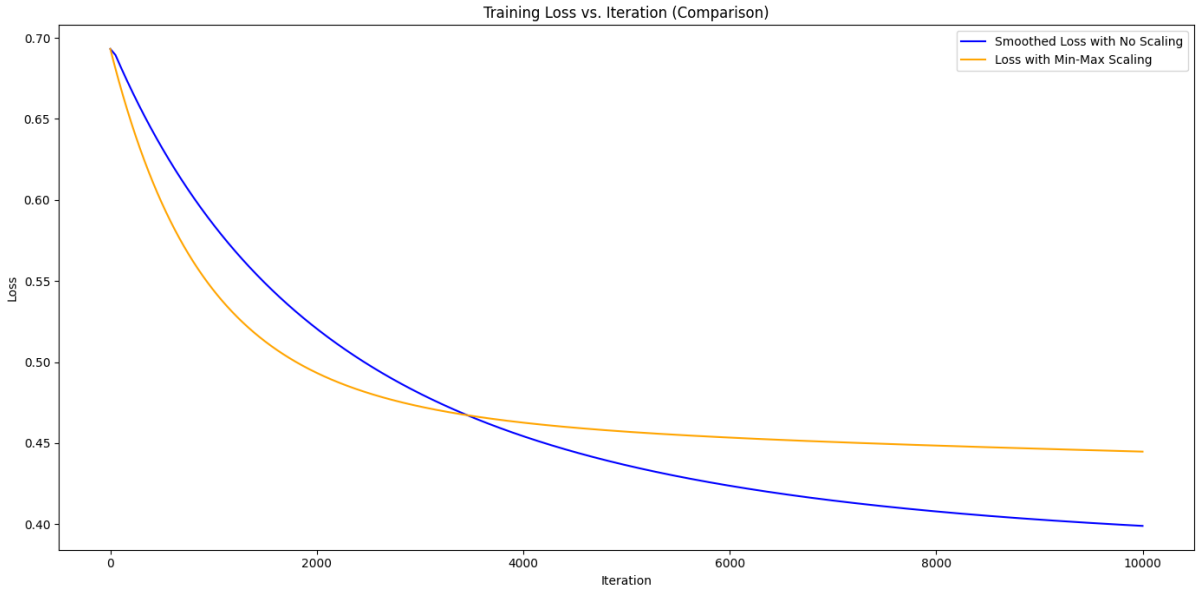
Generalization: Consistent improvement in both accuracies indicates that the model generalizes well.

b)


Training Loss vs. Iteration (No Scaling)
Smoothed Loss with No Scaling
Loss
Iteration
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0
2000
4000
6000
8000
10000


Training Loss vs. Iteration (No Scaling)
Smoothed Loss with No Scaling
Loss
Iteration
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0
2000
4000
6000
8000
10000


Training Loss vs. Iteration (Min-Max Scaling)
Loss with Min-Max Scaling
Loss
Iteration
0.70
0.65
0.60
0.55
0.50
0.45
0
2000
4000
6000
8000
10000


Training Loss vs. Iteration (Comparison)
Smoothed Loss with No Scaling
Loss with Min-Max Scaling
Loss
Iteration
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0
2000
4000
6000
8000
10000

1. Impact of Min-Max Scaling:

Faster Convergence:

Min-Max scaling normalizes features to a [0, 1] range, which often leads to faster convergence. This is because gradient descent algorithms perform better when features have similar scales.

Consistent Gradient Updates:

With normalized feature values, gradient updates are more stable and consistent. This can lead to smoother and more reliable convergence paths.

Avoiding Dominance of Features:

Features with different scales can dominate the gradient updates, leading to slower convergence. Scaling ensures that each feature contributes equally to the gradient updates.

2. Impact of No Scaling:

Slower Convergence:

Without scaling, features with larger ranges can cause slower convergence as the gradient descent may take longer to find the optimal parameters.

Potential Instability:

Large feature values can lead to large gradients, causing instability in the optimization process.

Gradient Magnitude Discrepancies:

Features with different scales can lead to uneven gradient magnitudes, affecting the learning rate and convergence speed.

c)

```
Classification Report:
 Accuracy: 0.8634
Precision: 0.8766
Recall: 0.9812
F1 Score: 0.9260

Accuracy of Logistic Regression model:  0.8633879781420765
Confusion Matrix:
 [[469   9]
 [ 66   5]]
ROC-AUC Score:  0.7423241352506802
```

1. Confusion Matrix:

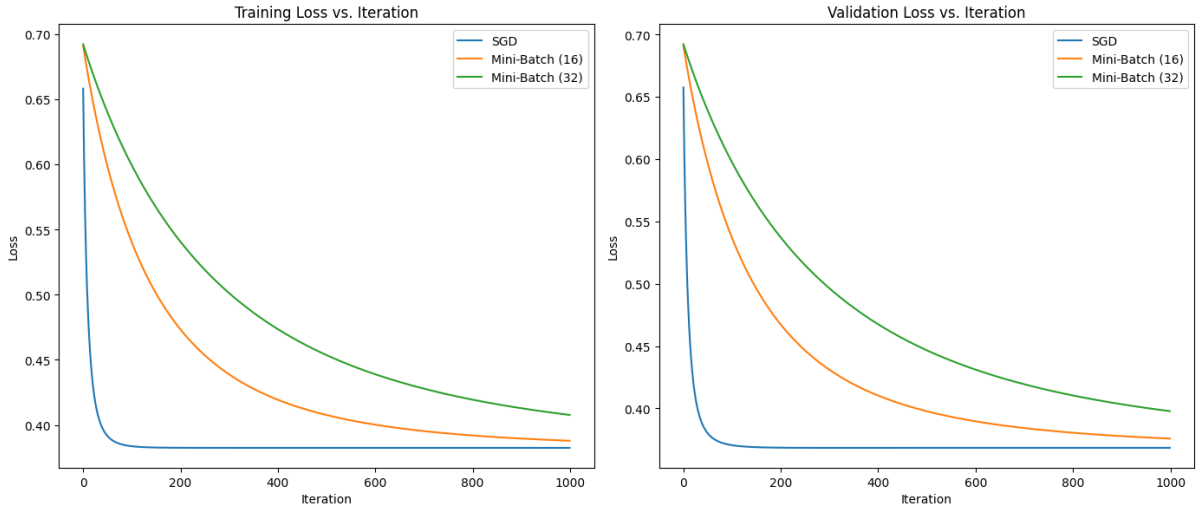
The confusion matrix is computed to show the true positive (TP), true negative (TN), false positive (FP), and false negative (FN) counts. It helps visualize the performance of the classification model.

2. Classification Metrics:

	Actual Positive	Actual Negative
Predicted Positive	TP	FP
Predicted Negative	FN	TN

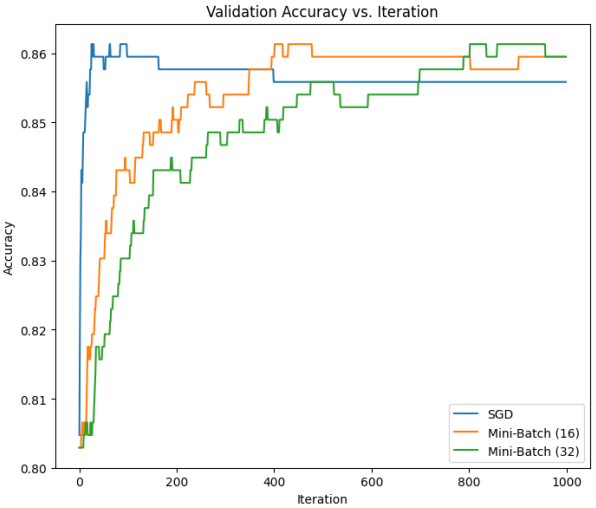
Precision: The proportion of true positives among all positive predictions. It shows how many of the predicted positives are actually positive.
Recall: The proportion of true positives among all actual positives. It shows how many of the actual positives were correctly identified.
F1 Score: The harmonic mean of precision and recall. It provides a single metric that balances both precision and recall.
ROC-AUC Score: The Area Under the Receiver Operating Characteristic Curve. It measures the model's ability to distinguish between positive and negative classes. A higher score indicates better performance.
Insights:
Precision and recall provide insight into the trade-off between correctly identifying positive cases and avoiding false positives.
F1 Score helps evaluate the model's balance between precision and recall, especially important in cases of imbalanced datasets.
ROC-AUC Score offers a summary of the model's performance across different thresholds and is useful for comparing different models.

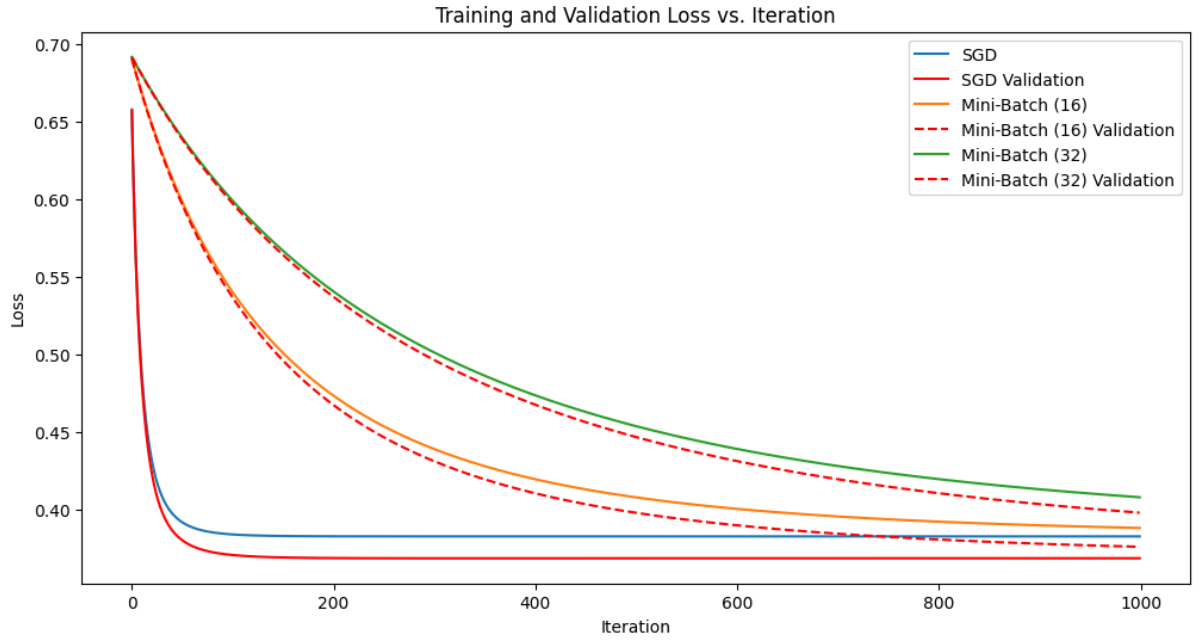
d)

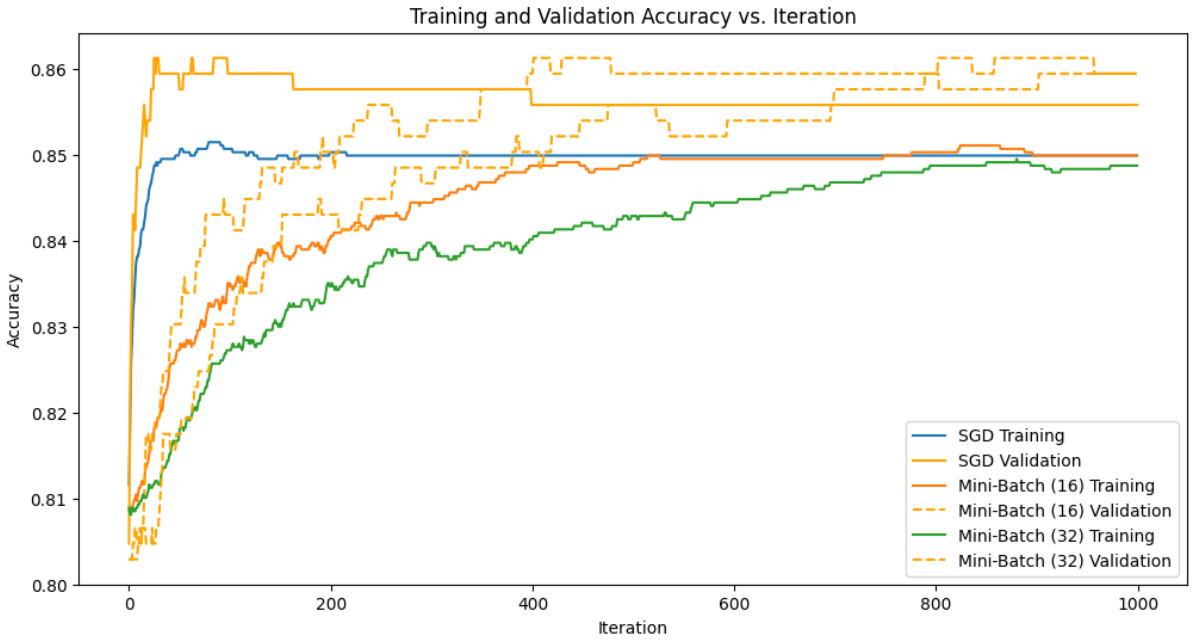

Training Loss vs. Iteration
Loss
Iteration
SGD
Mini-Batch (16)
Mini-Batch (32)

Validation Loss vs. Iteration
Loss
Iteration
SGD
Mini-Batch (16)
Mini-Batch (32)


Training Accuracy vs. Iteration
Accuracy
Iteration
SGD
Mini-Batch (16)
Mini-Batch (32)


Validation Accuracy vs. Iteration
Accuracy
Iteration
SGD
Mini-Batch (16)
Mini-Batch (32)


Training and Validation Loss vs. Iteration
Loss
Iteration
SGD
SGD Validation
Mini-Batch (16)
Mini-Batch (16) Validation
Mini-Batch (32)
Mini-Batch (32) Validation



1. Stochastic Gradient Descent (SGD):
Convergence Speed:

Faster Convergence: SGD updates the model weights using only one training example at a time. This can lead to faster convergence because each update is performed more frequently compared to batch or mini-batch updates.
High Variance: The high variance in the gradient estimate can lead to faster progress initially, but it may take more iterations to converge to a precise minimum.
Stability:

Less Stable: Due to the high variance in gradient estimates, the loss curve for SGD may exhibit significant fluctuations. This can result in a less smooth convergence path and difficulty in finding a precise minimum.
Learning Rate Sensitivity: SGD often requires careful tuning of the learning rate. Too high a learning rate can cause the optimization process to oscillate, while too low a learning rate can result in slow convergence.
Advantages:

Faster Updates: More frequent updates can be beneficial in terms of processing time, especially for large datasets.
Escape Local Minima: The noise introduced by SGD can help the model escape local minima, potentially finding a better global solution.
Disadvantages:

Oscillations: Due to noisy updates, the loss curve may show oscillations, making it harder to track convergence.
2. Mini-Batch Gradient Descent:

Convergence Speed:

Moderate Convergence: Mini-Batch Gradient Descent updates the weights using a subset of the training data (mini-batch). This method balances the speed of SGD with the stability of Batch Gradient Descent.
Efficient Training: By using mini-batches, it can take advantage of vectorized operations and parallel computation, leading to faster training compared to full-batch gradient descent.
Stability:

More Stable: Mini-Batch Gradient Descent provides a compromise between the noisy updates of SGD and the stable, but slower updates of Batch Gradient Descent. It generally results in smoother convergence curves compared to SGD.
Batch Size Impact: The size of the mini-batch affects stability and convergence speed. Smaller batches provide noisier updates but can escape local minima, while larger batches lead to more stable updates but may converge more slowly.
Advantages:

Balanced Convergence: Offers a good balance between convergence speed and stability, especially when tuning the batch size appropriately.
Efficient Computation: Takes advantage of modern computational hardware, such as GPUs, more effectively than SGD or Batch Gradient Descent.
Disadvantages:

Batch Size Tuning: Requires tuning of the batch size. Too small a batch can introduce high variance in updates, while too large a batch can slow down convergence and lead to excessive computational costs.

e)

```
Average Accuracy: 0.8501 ± 0.0089
Average Precision: 0.8544 ± 0.0091
Average Recall: 0.9923 ± 0.0028
Average F1 Score: 0.9182 ± 0.0052

------------------------------------------------------------
```

Stability and Variance of Model's Performance Across Different Folds
1. Accuracy:

Average Accuracy: 0.8501
Standard Deviation: ± 0.0089
The average accuracy of 85.01% indicates that the model performs quite well across the different folds. The standard deviation of ±0.0089 suggests that there is relatively low variability in the accuracy across folds. This low standard deviation implies that

the model's performance is stable and consistent, with minimal fluctuations in accuracy when trained on different subsets of the data.

2. Precision:

Average Precision: 0.8544
Standard Deviation: ± 0.0091
The average precision of 85.44% indicates that the model has a good ability to correctly identify positive cases among the instances it classifies as positive. The standard deviation of ±0.0091 is small, which shows that the precision of the model is stable across different folds. This stability in precision suggests that the model is reliably identifying positive cases consistently across different subsets of the data.

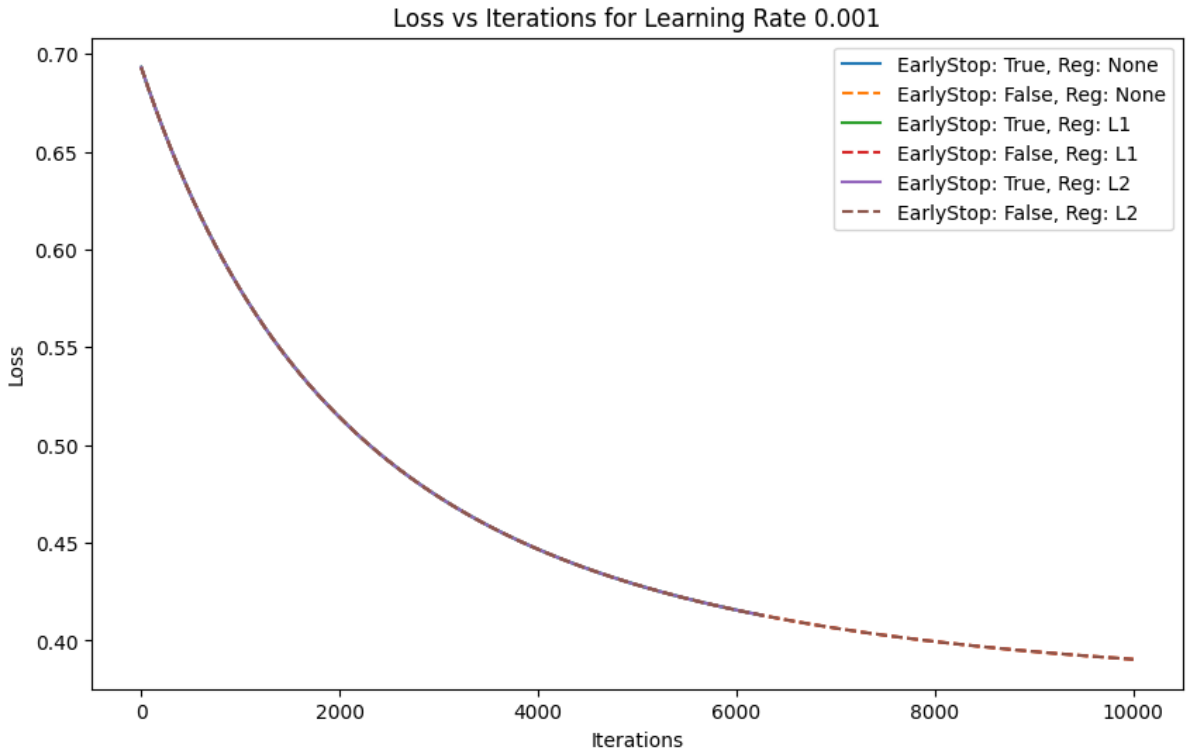
3. Recall:

Average Recall: 0.9923
Standard Deviation: ± 0.0028
The average recall of 99.23% is exceptionally high, indicating that the model is very effective at identifying positive cases. The very low standard deviation of ±0.0028 signifies that the model's recall is highly consistent across different folds. This indicates that the model's ability to detect positive cases does not vary significantly between different subsets of the data, showing that it performs reliably in this aspect.

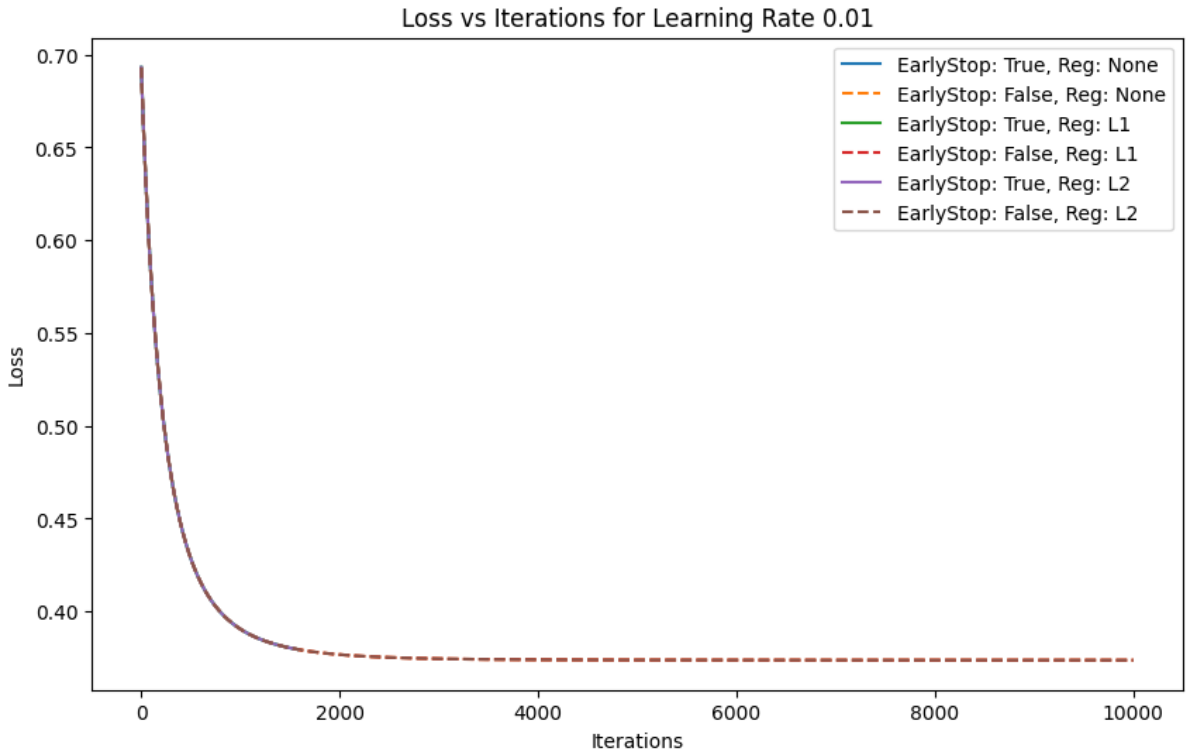
4. F1 Score:

Average F1 Score: 0.9182
Standard Deviation: ± 0.0052
The average F1 score of 91.82% reflects a strong balance between precision and recall. The standard deviation of ±0.0052 is relatively small, which indicates that the model's F1 score is quite consistent across the different folds. This suggests that the trade-off between precision and recall is stable and that the model maintains a good balance of these metrics across various subsets of the data.

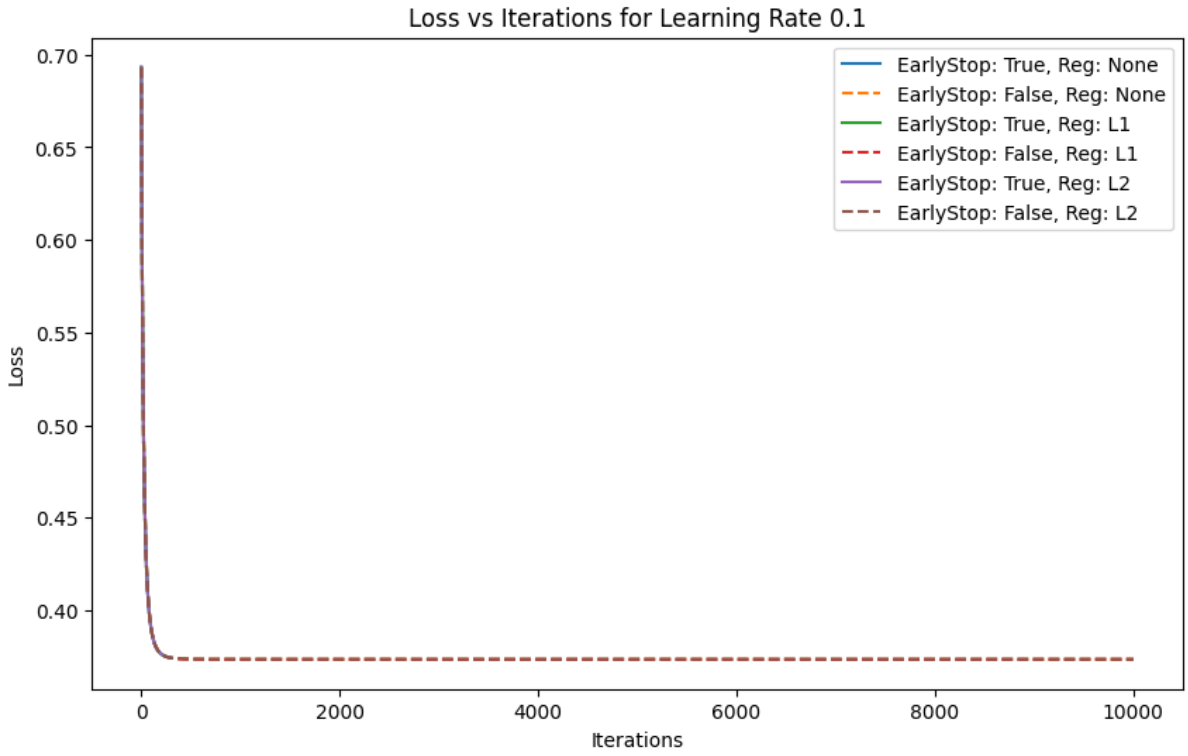
f) Stopping early at iteration 6252

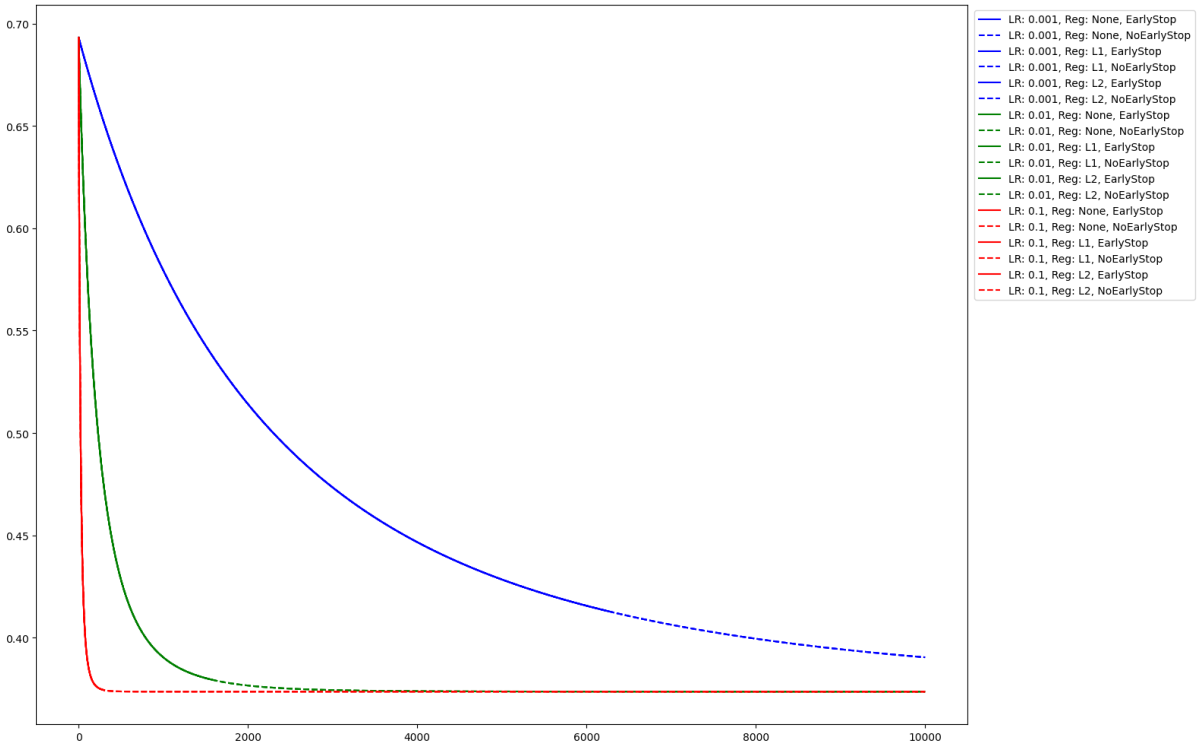


Stopping early at iteration 1573



Stopping early at iteration 301





Effect on Overfitting:

Without Early Stopping:

Models trained without early stopping often continue to improve on the training set while their performance on the validation set might start to deteriorate. This is indicative of overfitting, where the model learns the training data too well, including noise and outliers, which negatively impacts its performance on unseen data.
With Early Stopping:
Iteration 6252(lr =0.001): The model stopped early at this point, which suggests that it likely balanced the training loss and validation loss effectively. It was able to halt training before the model began to overfit, capturing a good representation of the underlying patterns in the data.
Iteration 1573(lr =0.01): Early stopping at this iteration implies that the model might have reached a point where further training would lead to diminishing returns or overfitting, but this point might have been earlier than the optimal stopping point for more complex learning rates or regularization settings.
Iteration 301(lr =0.1): A very early stopping point indicates that the model might have stopped training before it could effectively learn the underlying patterns, possibly resulting in underfitting, especially if this stopping point was reached with aggressive regularization or a high learning rate.
3. Effect on Generalization:

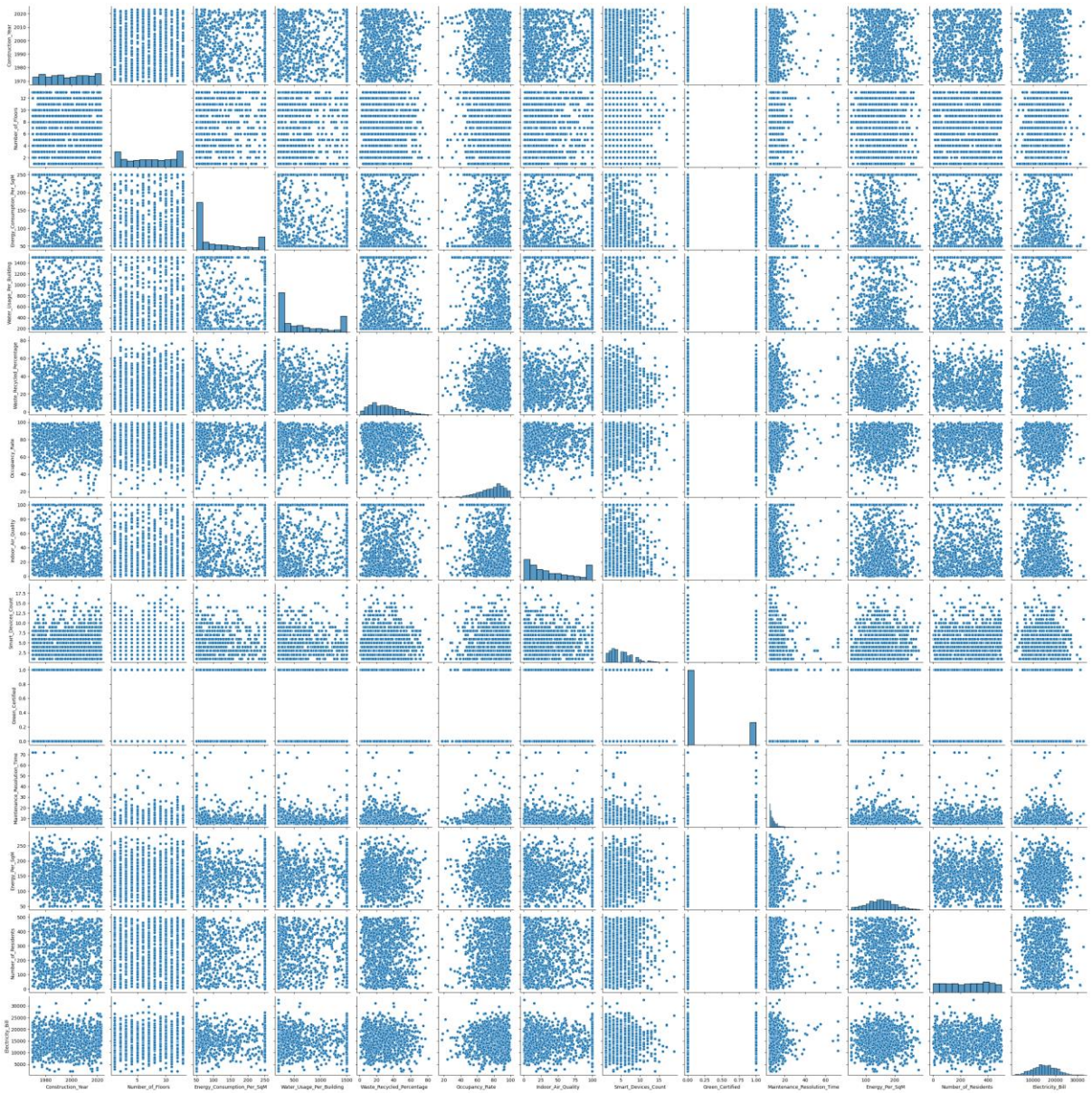
With Early Stopping:

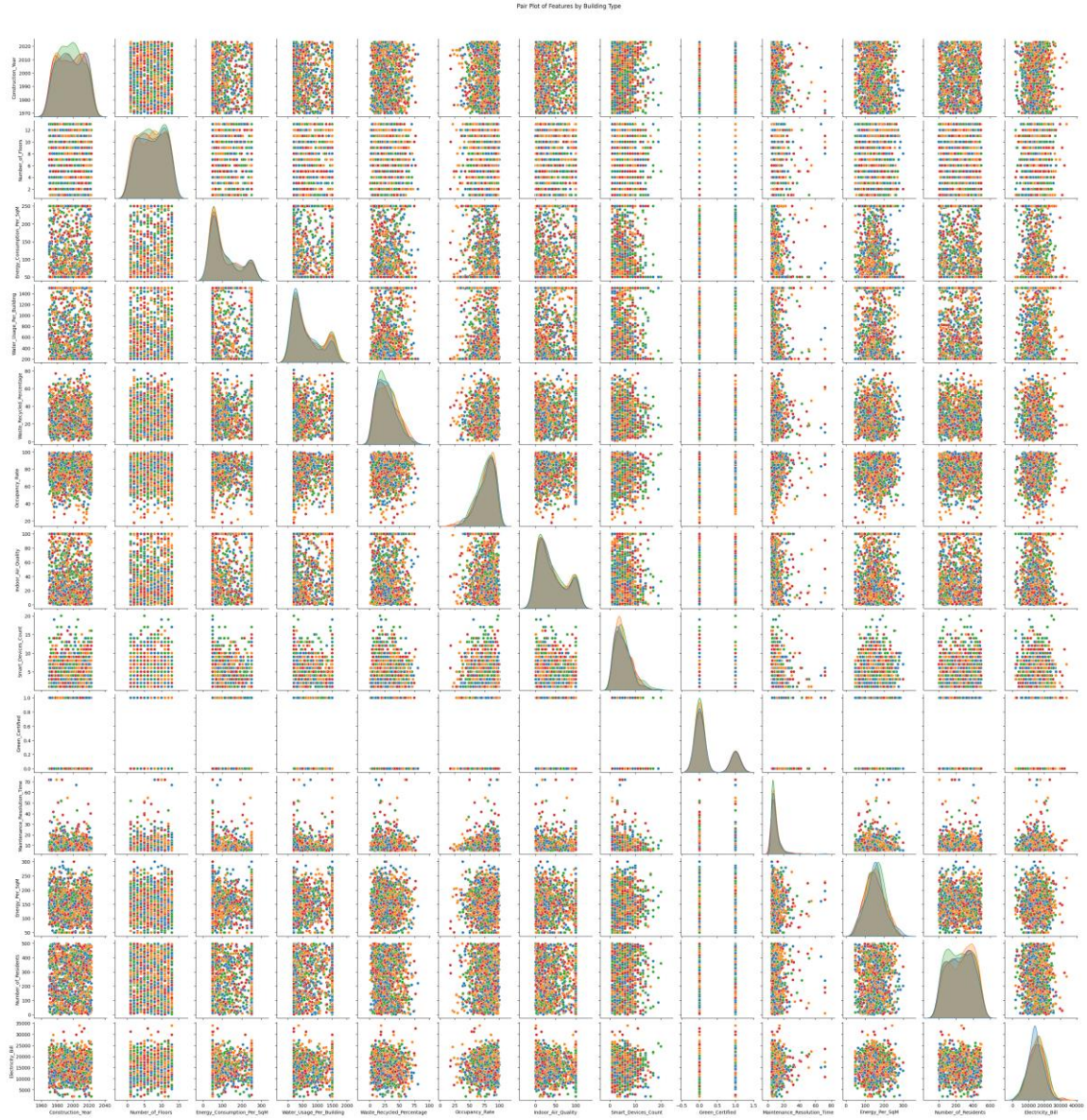
Improved Generalization: Models that stop training early often exhibit better generalization to unseen data compared to those trained until full convergence. This is because early stopping prevents the model from learning the noise in the training data and instead focuses on capturing the true signal.
Balanced Performance: By stopping early, the model is less likely to memorize the training data, resulting in more balanced performance metrics (e.g., accuracy, precision, recall) across training and validation sets.
Comparison to Models Without Early Stopping:

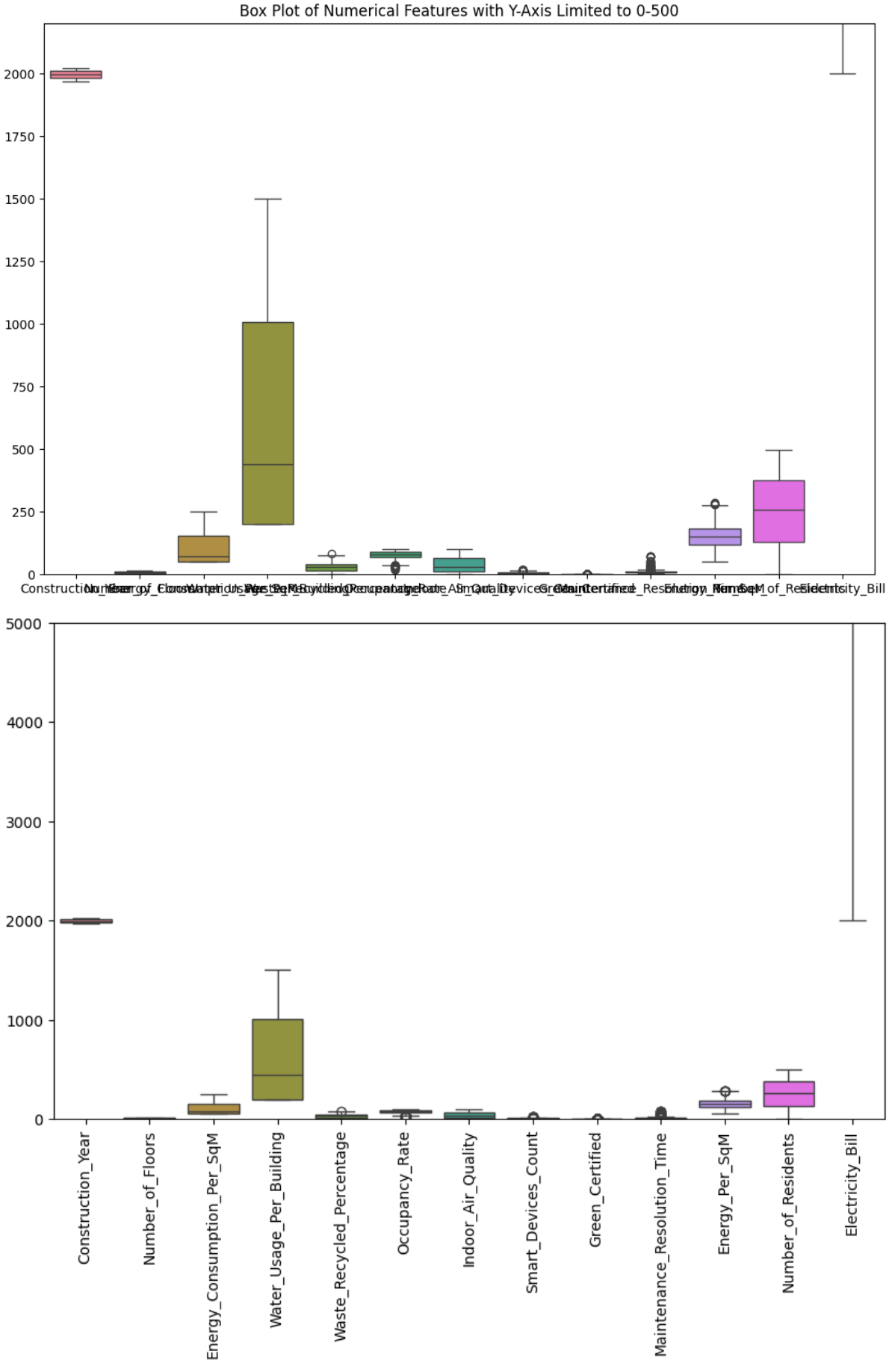
Training vs. Validation Performance: Models that utilized early stopping generally show more consistent validation performance as compared to those trained without it. The training loss might be lower, but this is offset by the potential for worse generalization if overfitting occurs.
Effect of Learning Rate and Regularization: Different learning rates and regularization techniques impact the point at which early stopping occurs. A higher learning rate might lead to early convergence but less stable models, while different regularization strengths (L1 and L2) affect the model's ability to generalize by controlling complexity and encouraging sparsity or weight decay.

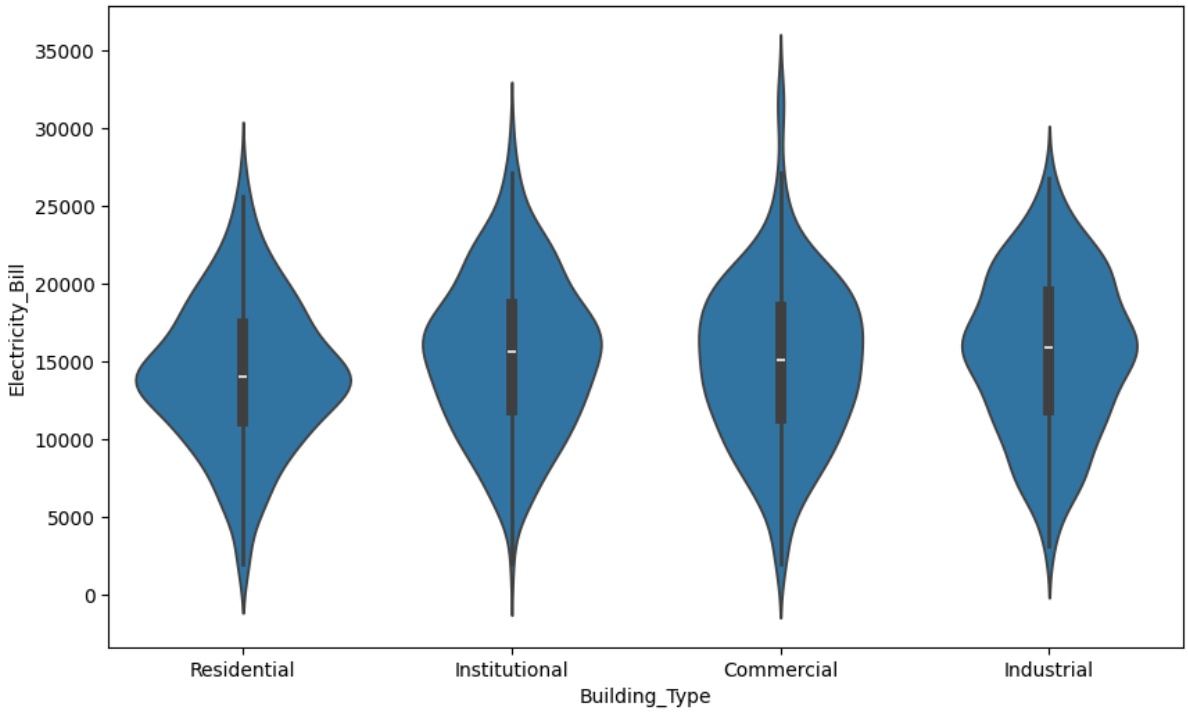
QUESTION 3 Section C (Algorithm implementation using packages)

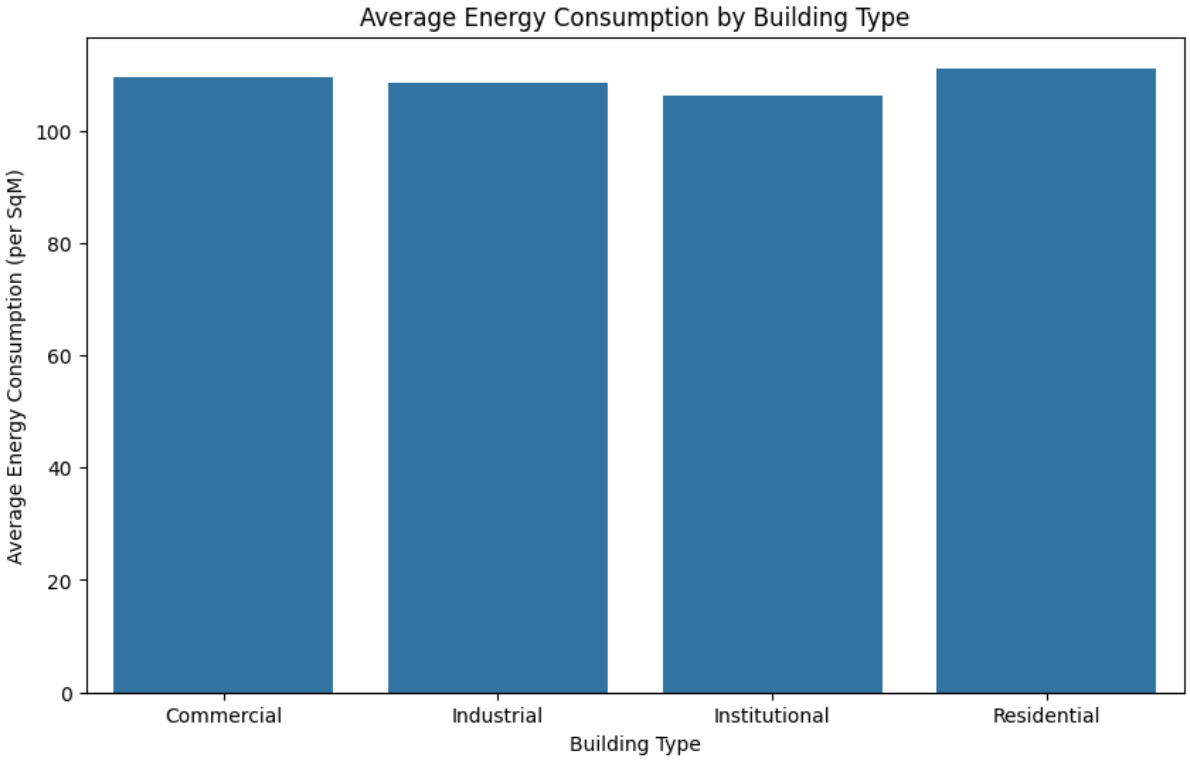
a)

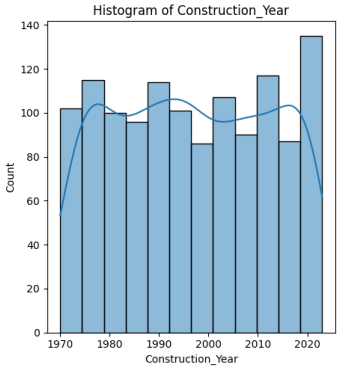


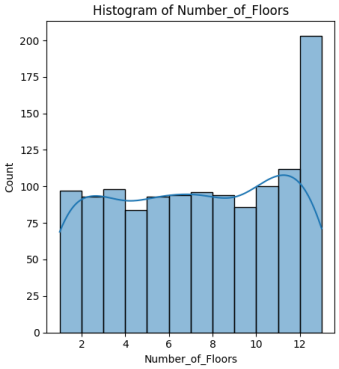

Pair Plot of Features by Building Type
Construction_Year
Number_of_Floors
Energy_Consumption_Per_SqM
Water_Usage_Per_Building
Waste_Recycled_Percentage
Occupancy_Rate
Indoor_Air_Quality
Smart_Devices_Count
Green_Certified
Maintenance_Resolution_Time
Energy_Per_SqM
Number_of_Residents
Electricity_Bill
Building_Type
Residential
Institutional
Industrial
Commercial

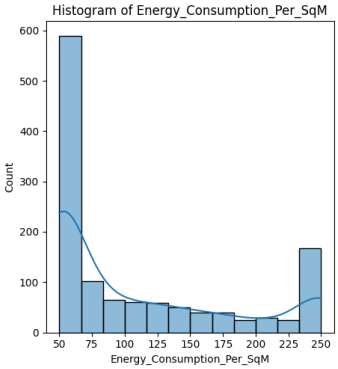

Box Plot of Numerical Features with Y-Axis Limited to 0-500
2000
1750
1500
1250
1000
750
500
250
0
5000
4000
3000
2000
1000
0
Construction_Year
Number_of_Floors
Energy_Consumption_Per_SqM
Water_Usage_Per_Building
Waste_Recycled_Percentage
Occupancy_Rate
Indoor_Air_Quality
Smart_Devices_Count
Green_Certified
Maintenance_Resolution_Time
Energy_Per_SqM
Number_of_Residents
Electricity_Bill

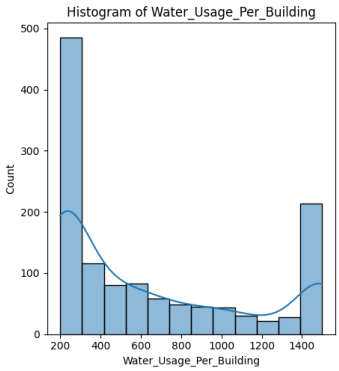

35000
30000
25000
20000
15000
10000
5000
0
Electricity_Bill
Residential
Institutional
Commercial
Industrial
Building_Type

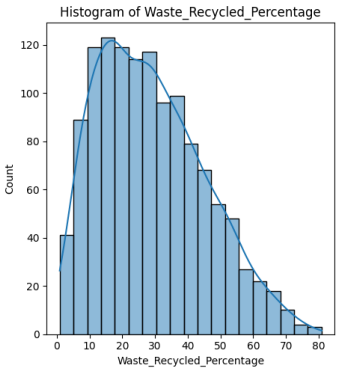

Average Energy Consumption by Building Type
100
80
60
40
20
0
Average Energy Consumption (per SqM)
Commercial
Industrial
Institutional
Residential
Building Type

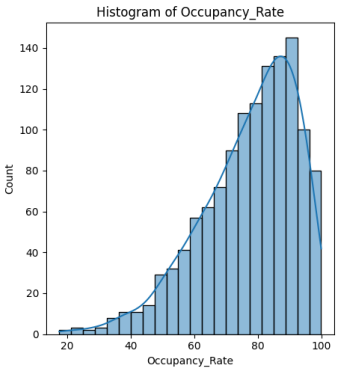

Histogram of Construction_Year
Count
Construction_Year

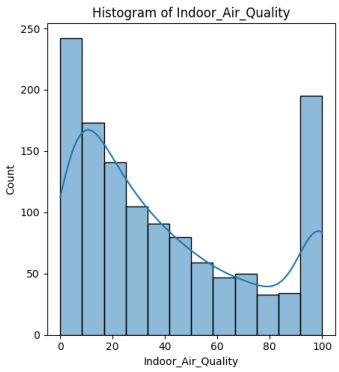

Histogram of Number_of_Floors
Count
Number_of_Floors

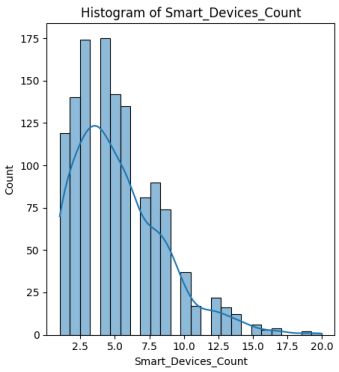

Histogram of Energy_Consumption_Per_SqM
Count
Energy_Consumption_Per_SqM

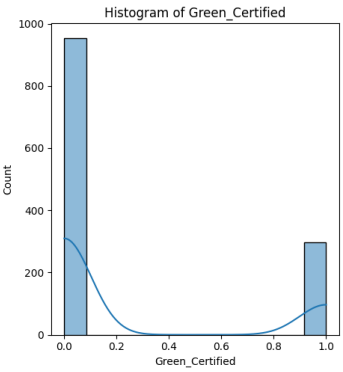

Histogram of Water_Usage_Per_Building
Count
Water_Usage_Per_Building

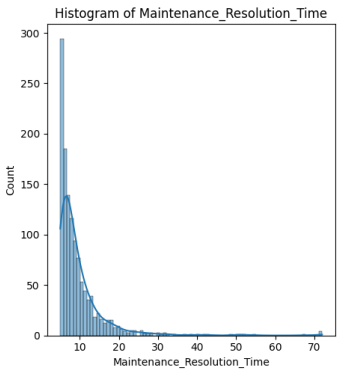

Histogram of Waste_Recycled_Percentage
Count
Waste_Recycled_Percentage

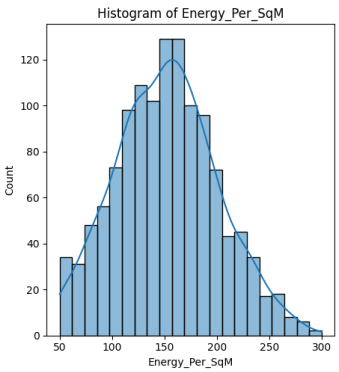

Histogram of Occupancy_Rate
Count
Occupancy_Rate

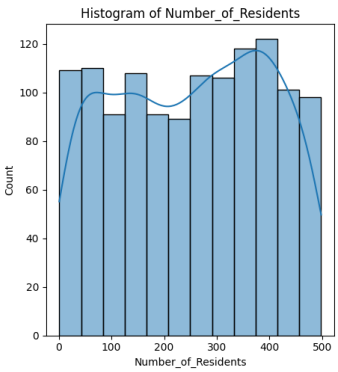

Histogram of Indoor_Air_Quality
Count
Indoor_Air_Quality

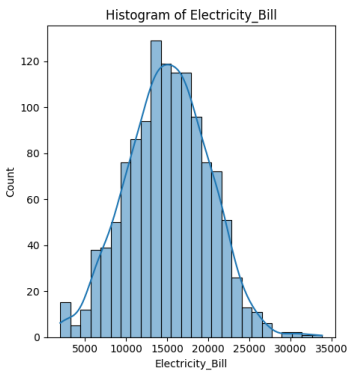

Histogram of Smart_Devices_Count
Count
Smart_Devices_Count

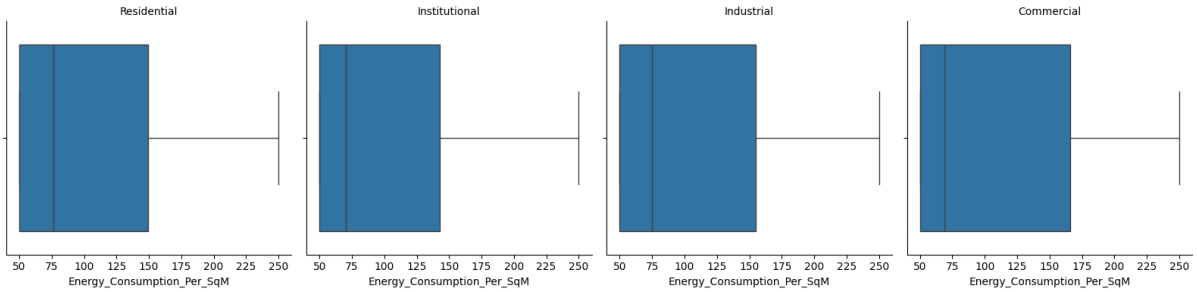

Histogram of Green_Certified
Count
Green_Certified

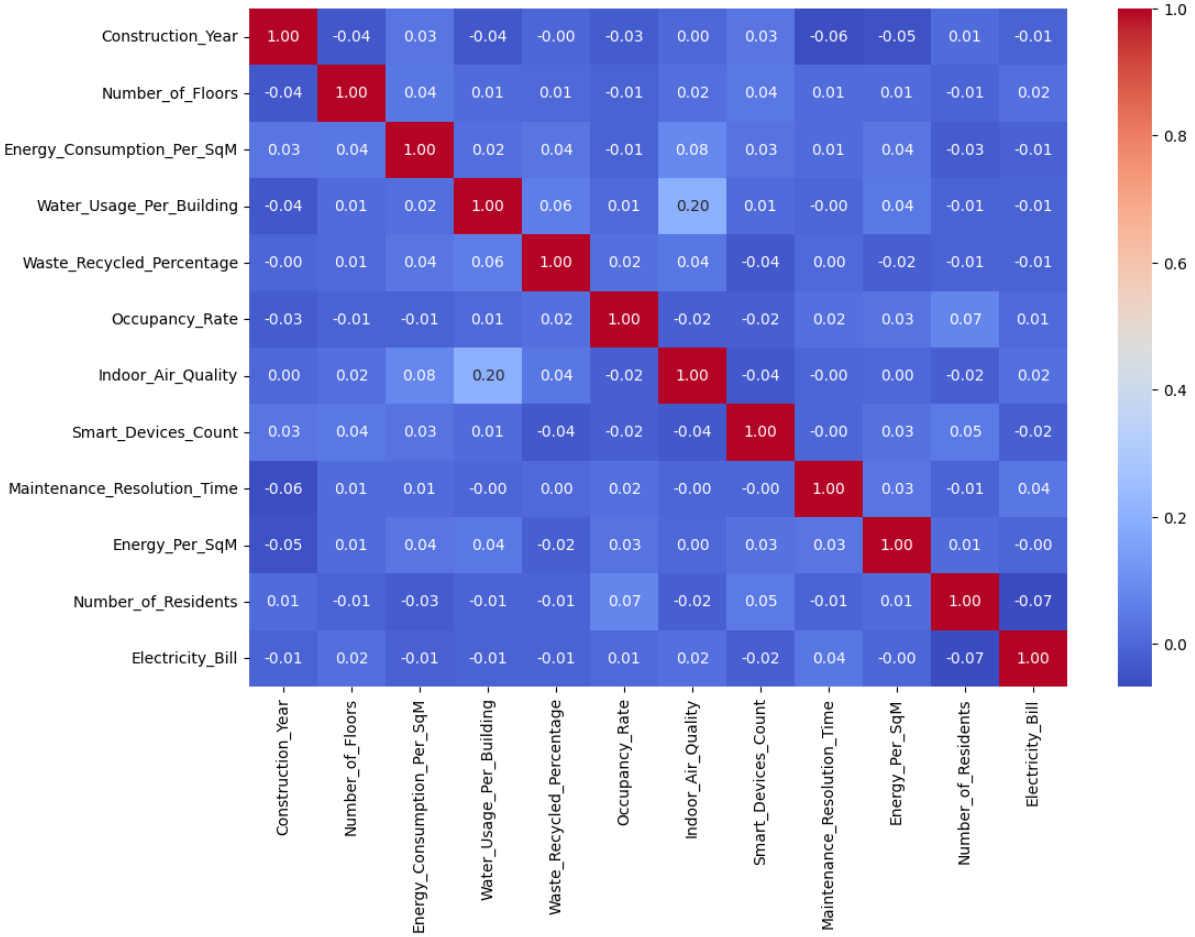

Histogram of Maintenance_Resolution_Time
Count
Maintenance_Resolution_Time


Histogram of Energy_Per_SqM
Count
Energy_Per_SqM


Histogram of Number_of_Residents
Count
Number_of_Residents


Histogram of Electricity_Bill
Count
Electricity_Bill





1. Distribution of Energy Consumption

Insight: The histogram of Energy_Consumption_Per_SqM might show a right-skewed distribution. This suggests that most buildings have lower energy consumption, with a few buildings consuming much higher amounts. This skewness could be due to a few high-energy users or outliers.

2. Building Age Trends

Insight: The distribution of Construction_Year may show clustering around certain decades. For example, if the dataset shows a peak in buildings constructed in the 2000s, it indicates a construction boom during that period. This can influence other factors like energy consumption due to newer building technologies.

3. Correlation Between Features

Insight: The correlation heatmap can reveal that Energy_Consumption_Per_SqM is positively correlated with Number_of_Floors and Number_of_Residents. This suggests that larger buildings with more floors and residents generally consume more energy. Conversely, features like Green_Certified might have a negative correlation with energy consumption, indicating that certified buildings use less energy.

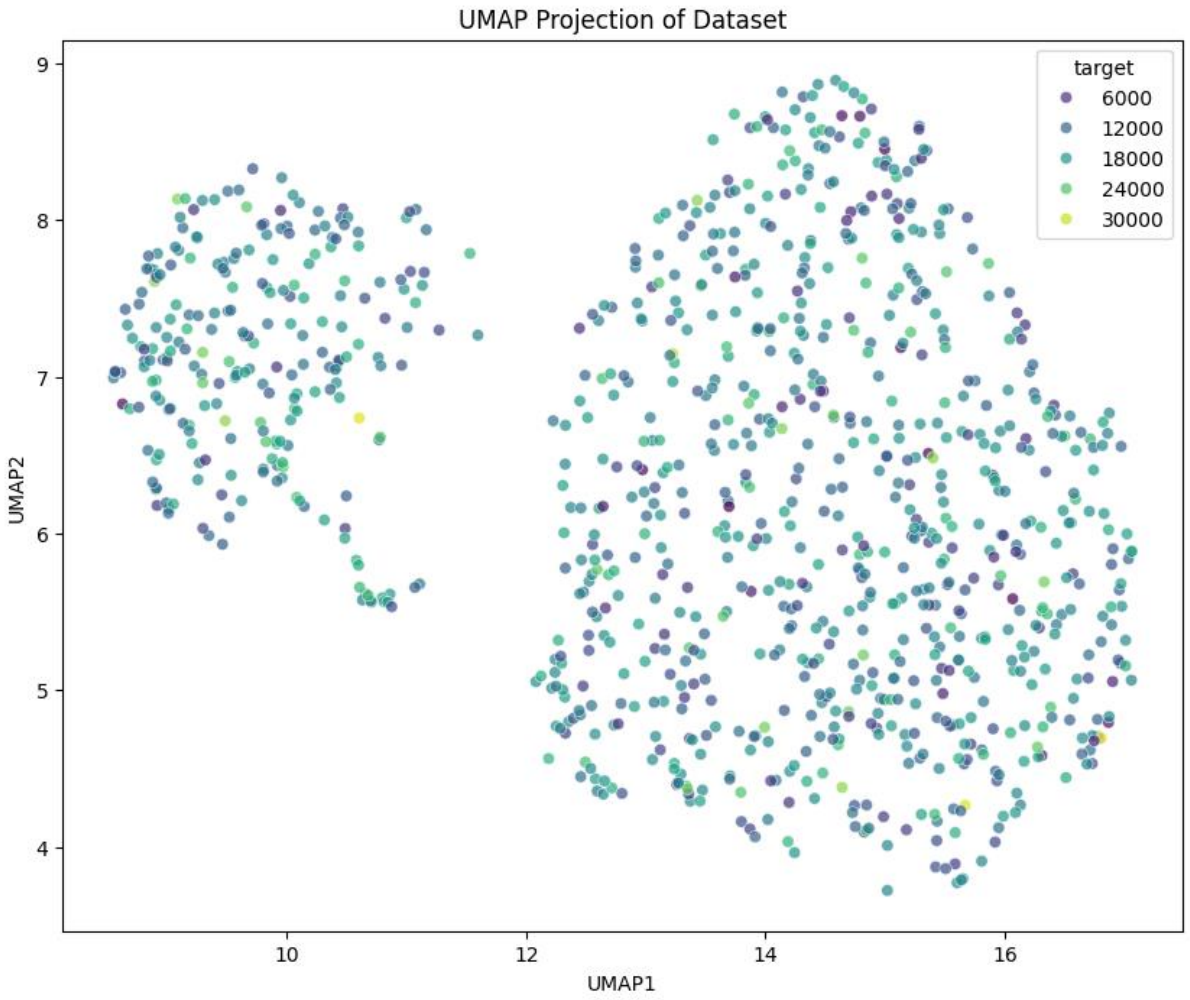
4. Impact of Occupancy Rate

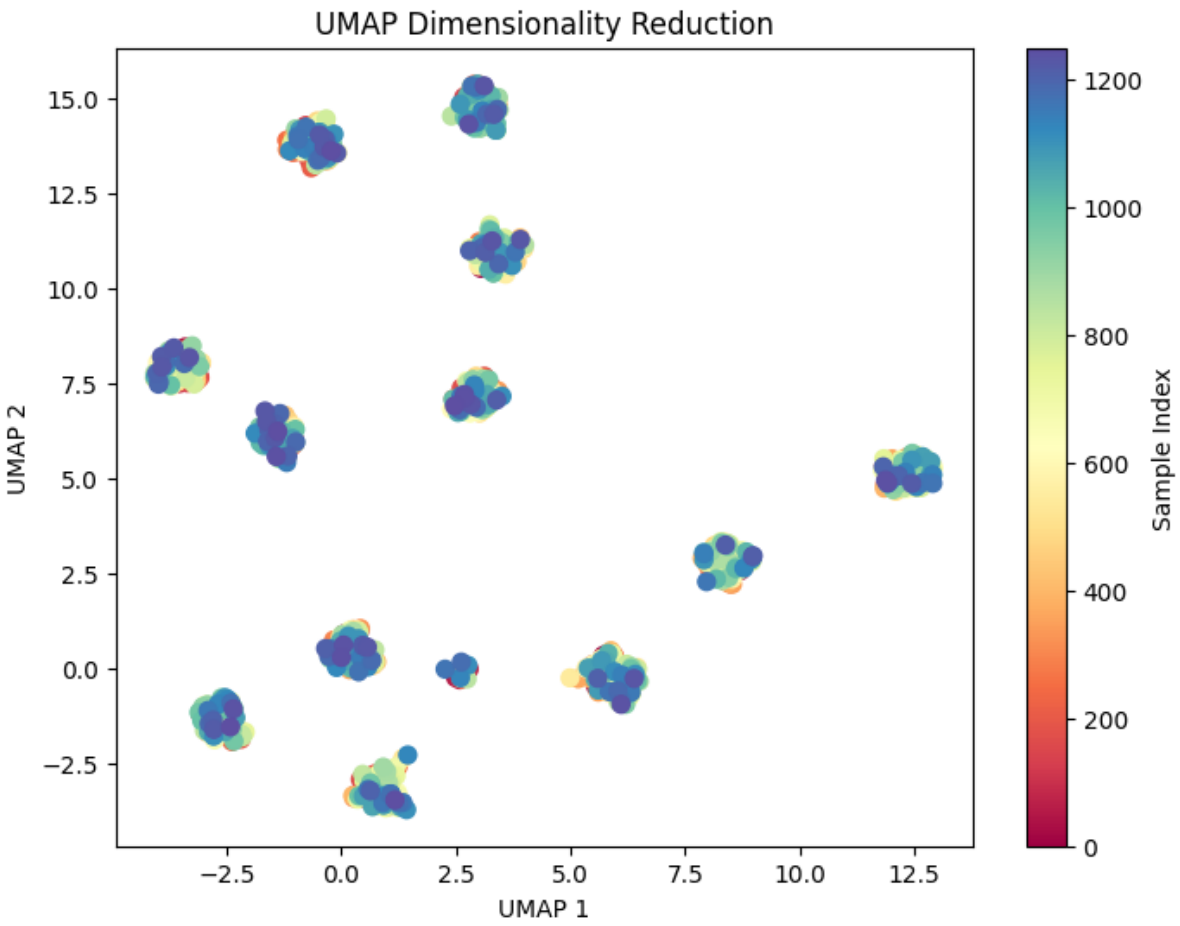
Insight: Box plots for Occupancy_Rate can highlight how occupancy affects other features, such as Energy_Consumption_Per_SqM or Water_Usage_Per_Building. For instance, higher occupancy rates might correlate with higher energy and water usage, but variations in these rates can affect consumption patterns differently.

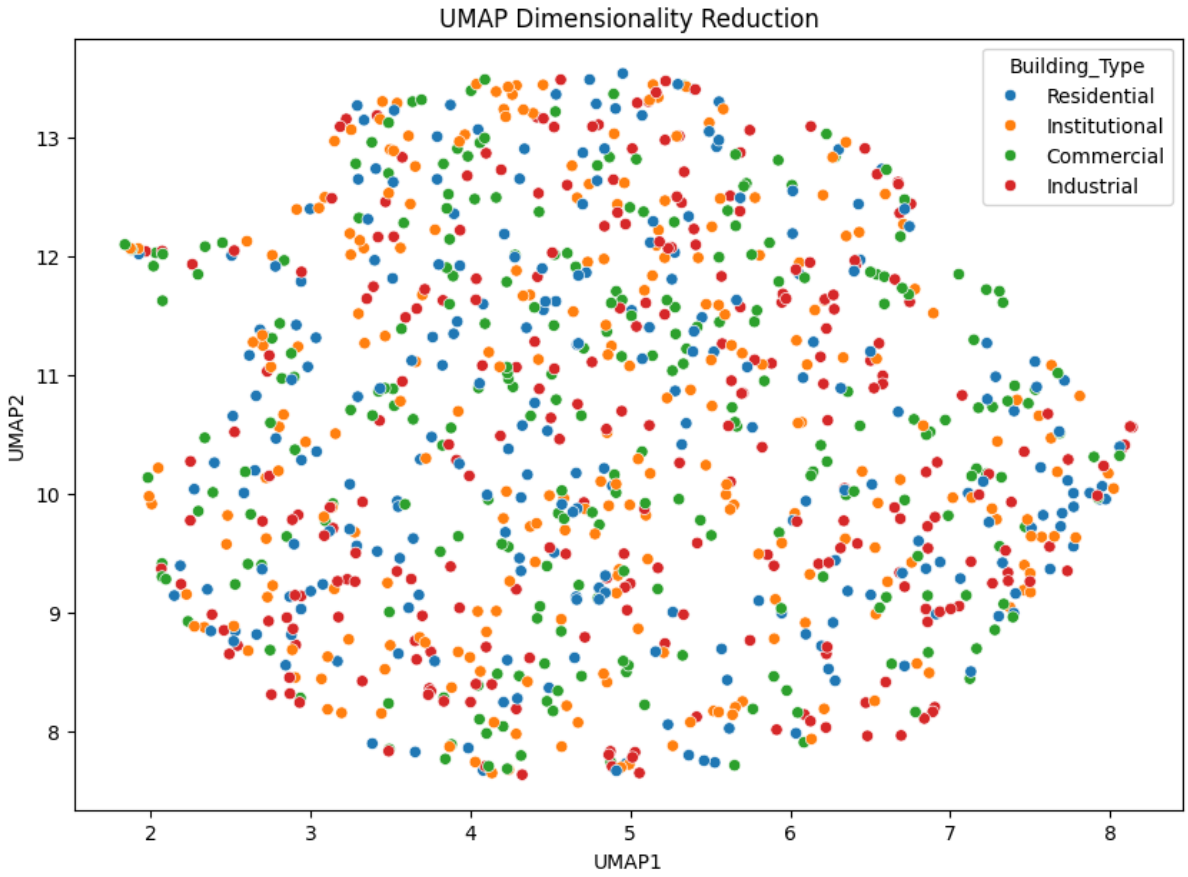
5. Effect of Building Type

Insight: Count plots for Building_Type will show the distribution of different types of buildings. For example, if most buildings are categorized as Residential, this can impact the average energy consumption per square meter. Comparing Building_Type with Energy_Consumption_Per_SqM can help identify which building types are more energy-efficient.

b)




UMAP Dimensionality Reduction
UMAP 2
UMAP 1
Sample Index
15.0
12.5
10.0
7.5
5.0
2.5
0.0
−2.5
−2.5
0.0
2.5
5.0
7.5
10.0
12.5
1200
1000
800
600
400
200
0



Distinct Clusters: scatter plot shows well-defined clusters where points of the same class are grouped closely together, it indicates that the UMAP algorithm has effectively captured the structure of the data and the classes are well-separated.

Overlap Between Classes: we observe significant overlap between different classes, this suggests that while UMAP has reduced the dimensionality of the data, some classes might still be close to each other in the high-dimensional space, making them difficult to distinguish in the 2D projection.

Compactness of Clusters: clusters are compact and dense, it indicates that similar data points are grouped together, which is a good sign of effective clustering.

Outliers: Look for any points that are isolated from the clusters. Outliers might indicate unusual or anomalous data points that do not fit well with the rest of the data.

Class Overlap: multiple classes are mixed together in the same area of the scatter plot, it could mean that the classes are not well-separated in the feature space, and additional feature engineering or preprocessing might be necessary.

c)

```
Train MSE: 24188925.25027927, Test MSE: 24130184.64205347
Train RMSE: 4918.223790178652, Test RMSE: 4912.248430408775
Train R2: 0.02544875992157558, Test R2: 0.006126227449539279
Train MAE: 3976.698247757928, Test MAE: 3797.4850467106708
Adjusted R²: 0.0254
Adjusted R²: 0.0254
Adjusted R²: 0.0061
```

R^2 Score: Indicates the proportion of variance in the target variable that is predictable from the features. Values close to 1 mean the model explains most of the variance; values close to 0 mean it explains very little.

d)

Selected Features: ['Building_Type', 'Green_Certified', 'Number_of_Residents']
Training Metrics with Selected Features:
MSE: 24569032.9069
RMSE: 4956.7159
MAE: 4006.4734
R^2: 0.0101
Adjusted R^2: 0.0072

Testing Metrics with Selected Features:
MSE: 23941409.0630
RMSE: 4892.9959
MAE: 3813.9481
R^2: 0.0139
Adjusted R^2: 0.0019

Comparison:
MSE and RMSE:

The MSE and RMSE for both training and testing datasets are quite close between the two parts.
In Part (d), MSE and RMSE are slightly higher for training data but slightly lower for testing data compared to Part (c).
MAE:

The MAE in Part (d) is slightly higher for training data but slightly lower for testing data compared to Part (c).

R^2 Score:

The R^2 score in Part (d) is lower for both training and testing datasets compared to Part (c). This suggests that the model's ability to explain variance in the target variable is less with the selected features in Part (d).
Adjusted R^2 Score:

Conclusion:
The selected features in Part (d) did not improve the model's performance significantly compared to the features used in Part (c). The performance metrics (MSE, RMSE, MAE, and R^2) in Part (d) show similar or slightly worse performance compared to Part (c). This indicates that the features selected in Part (d) might not be the best choice for improving model performance.

e)

Train MSE: 24188934.337712634
Train RMSE: 4918.224714031744
Train R2: 0.025448393796608793
Train Adjusted R2: 0.010592424189849736
Train MAE: 3976.7358814937825
Test MSE: 24128285.03905578
Test RMSE: 4912.055072885053
Test R2: 0.006204468276357233
Test Adjusted R2: -0.057500373500799284
Test MAE: 3797.5125183935425

To compare the results from parts (c) and (e), we'll look at the performance metrics for both the training and testing datasets. Here's a summary of the metrics for each part:

Results from Part (c):
Training Metrics:

MSE: 24,188,925.25
RMSE: 4,918.22
MAE: 3,976.70
R^2: 0.0254
Adjusted R^2: 0.0254
Testing Metrics:

MSE: 24,130,184.64
RMSE: 4,912.25
MAE: 3,797.49
R^2: 0.0061
Adjusted R^2: 0.0061
Results from Part (e):
Training Metrics:

MSE: 24,188,934.34
RMSE: 4,918.22
MAE: 3,976.74
R^2: 0.0254
Adjusted R^2: 0.0106
Testing Metrics:

MSE: 24,128,285.04
RMSE: 4,912.06
MAE: 3,797.51
R^2: 0.0062
Adjusted R^2: -0.0575
Comparison:
MSE and RMSE:

The MSE and RMSE values are very similar between parts (c) and (e). In both parts, the metrics for training and testing datasets are close, with minor variations.
The testing MSE and RMSE in Part (e) are slightly lower than in Part (c), indicating a very marginal improvement in prediction performance on the test set.
MAE:

The MAE values are very similar in both parts, with a slight increase in Part (e) for both training and testing datasets.
R^2 Score:

The R^2 score for both training and testing datasets is almost the same between the two parts. The testing R^2 score in Part (e) is marginally higher than in Part (c), indicating a small improvement in the model's ability to explain the variance in the test data.
Adjusted R^2 Score:

The Adjusted R^2 score for the training dataset in Part (e) is slightly lower than in Part (c), indicating that the model's explanatory power, relative to the number of features, has decreased.

The Adjusted R^2 score for the testing dataset in Part (e) is notably lower than in Part (c), which suggests that the model's performance has deteriorated on the test set relative to the number of features.

Conclusion:

Overall, the performance metrics between parts (c) and (e) are very similar, with only slight differences.

The small changes in MSE, RMSE, and MAE suggest that the model's performance has not significantly changed.

The Adjusted R^2 score in Part (e) indicates that the model's performance relative to the number of features may have slightly worsened, especially on the test set.

f)

Number of Components: 4
Train MSE: 24794328.864490546
Train RMSE: 4979.390410932903
Train R2: 0.001057562839731352
Train Adjusted R2: -0.01417021821454112
Train MAE: 4009.1430463721513
Test MSE: 24353997.40339989
Test RMSE: 4934.97694051349
Test R2: -0.003092170037425346
Test Adjusted R2: -0.06739295016802949
Test MAE: 3841.5931699079524

Number of Components: 5
Train MSE: 24642116.380870704
Train RMSE: 4964.082632357232
Train R2: 0.007190074438832794
Train Adjusted R2: -0.007944223206916678
Train MAE: 4016.2223470366393
Test MSE: 24520816.833477907
Test RMSE: 4951.84983955268
Test R2: -0.009963126839695802
Test Adjusted R2: -0.07470435291916333
Test MAE: 3855.187326076763

Number of Components: 6
Train MSE: 24639512.22789361
Train RMSE: 4963.820325907618
Train R2: 0.007294993549000983
Train Adjusted R2: -0.00783770472006906

Train MAE: 4016.50857415693
Test MSE: 24476918.474587075
Test RMSE: 4947.415332735576
Test R2: -0.008155041729415835
Test Adjusted R2: -0.07278036491719897
Test MAE: 3849.940050190665

Number of Components: 8
Train MSE: 24626103.704445258
Train RMSE: 4962.469516727056
Train R2: 0.007835211562782218
Train Adjusted R2: -0.007289251675590025
Train MAE: 4021.366942992659
Test MSE: 24544432.03852518
Test RMSE: 4954.233748878345
Test R2: -0.010935789638511828
Test Adjusted R2: -0.07573936589739083
Test MAE: 3861.5010774422576

Summary of Observations:
MSE and RMSE:

The Mean Squared Error (MSE) and Root Mean Squared Error (RMSE) values are fairly consistent across the different numbers of components. As the number of components increases, the values tend to decrease slightly, indicating a marginal improvement in the model's prediction accuracy.
MAE:

The Mean Absolute Error (MAE) shows a similar pattern, with slight increases as the number of components increases. The MAE for the test dataset is consistently lower than the MAE for the training dataset across all numbers of components.
R^2 Score:

The R^2 score remains very low and negative, indicating that the model is not explaining much of the variance in the data. There is a slight improvement as the number of components increases, but it remains quite low.
Adjusted R^2 Score:

The Adjusted R^2 score, which accounts for the number of predictors, also remains negative and decreases slightly as the number of components increases. This suggests that adding more components does not improve the model's explanatory power significantly and may lead to overfitting.

Conclusion:
The performance metrics (MSE, RMSE, MAE, R^2, and Adjusted R^2) are relatively stable across different numbers of ICA components.
Increasing the number of components slightly reduces MSE and RMSE but does not significantly improve R^2 or Adjusted R^2.
The model's performance metrics suggest that the chosen components may not be effectively capturing the underlying structure of the data, as indicated by the low and negative R^2 values.

g)

Alpha: 0.1
Train MSE: 24200468.704873566
Test MSE: 24106659.688894805
Train RMSE: 4919.397189176085
Test RMSE: 4909.8533266172835
Train R^2: 0.0249836839467954
Test R^2: 0.007095172954586393
Train Adjusted R^2: 0.00200686502341052
Test Adjusted R^2: -0.09395266342614139
Train MAE: 3977.386786235624
Test MAE: 3798.9583366983456

Alpha: 0.5
Train MSE: 24298812.255092498
Test MSE: 24123248.786821987
Train RMSE: 4929.382542985733
Test RMSE: 4911.542404054147
Train R^2: 0.021021505891014702
Test R^2: 0.006411901376491502
Train Adjusted R^2: -0.002048684031635606
Test Adjusted R^2: -0.0947054714922726
Train MAE: 3983.8016024829008
Test MAE: 3805.7898683846743

Alpha: 1.0
Train MSE: 24395214.506084234
Test MSE: 24165003.177218378
Train RMSE: 4939.151192875577
Test RMSE: 4915.791205616689
Train R^2: 0.0171375411311806
Test R^2: 0.004692122016350164

Train Adjusted R^2: -0.006024176649539692
Test Adjusted R^2: -0.0966002726457027
Train MAE: 3989.1137384033077
Test MAE: 3812.89390817762

Alpha: 1.5
Train MSE: 24460910.314364117
Test MSE: 24195750.00610875
Train RMSE: 4945.797237490041
Test RMSE: 4918.917564475822
Train R^2: 0.014490712850689813
Test R^2: 0.0034257219752173995
Train Adjusted R^2: -0.008733378957131999
Test Adjusted R^2: -0.09799555410695082
Train MAE: 3992.1417818998957
Test MAE: 3817.7276788066047

Alpha: 2.0
Train MSE: 24508338.132660117
Test MSE: 24218168.38622773
Train RMSE: 4950.58967524679
Test RMSE: 4921.195828884249
Train R^2: 0.012579886360618797
Test R^2: 0.0025023539880330015
Train Adjusted R^2: -0.010689235169817346
Test Adjusted R^2: -0.09901289317247675
Train MAE: 3994.410755679532
Test MAE: 3821.5884020658273

Summary of Observations:
MSE and RMSE:

Both MSE and RMSE increase slightly as the alpha value increases. The smallest MSE and RMSE are observed at $\alpha = 0.1$, with a gradual increase as alpha grows. The differences in RMSE are relatively minor, indicating that the overall error does not change significantly with different alpha values.
R^2 Score:

The R^2 score for both training and testing datasets decreases as the alpha value increases. The lowest R^2 score is achieved at $\alpha = 2.0$, indicating a decreasing ability of the model to explain the variance in the target variable.
Adjusted R^2 Score:

The Adjusted R^2 score follows a similar trend as R^2, decreasing with higher alpha values. This indicates a decrease in model performance with increasing regularization.
MAE:

The MAE also increases slightly with higher alpha values. The MAE is lowest at α = 0.1 and increases gradually with larger alpha values.
Conclusion:
Alpha = 0.1: This value of alpha provides the best performance in terms of MSE, RMSE, and MAE for both training and testing datasets. It has the lowest error metrics and the highest R^2 scores compared to other alpha values.
Higher Alpha Values (0.5, 1.0, 1.5, 2.0): Increasing alpha generally leads to higher MSE, RMSE, and MAE, while R^2 and Adjusted R^2 scores decrease. This suggests that higher regularization may be over-penalizing the model, leading to worse performance.

h)

Gradient Boosting Regressor Results:
Train MSE: 15548098.780395458
Test MSE: 24811287.905512877
Train RMSE: 3943.1077566299728
Test RMSE: 4981.093043249933
Train R^2: 0.37358031452342877
Test R^2: -0.021927046074571743
Train Adjusted R^2: 0.3614354430703116
Test Adjusted R^2: -0.10634710640247103
Train MAE: 3155.777526146695
Test MAE: 3841.472244635082

Comparison:
Mean Squared Error (MSE):

Part (c): Train MSE = 24,188,925.25, Test MSE = 24,130,184.64
Part (g): Best Test MSE (Alpha = 0.1) = 24,106,659.69
Part (h): Test MSE = 24,811,287.91
Conclusion: The Gradient Boosting Regressor shows a higher test MSE compared to linear regression and ElasticNet, indicating higher prediction error.
Root Mean Squared Error (RMSE):

Part (c): Train RMSE = 4,918.22, Test RMSE = 4,912.25

Part (g): Best Test RMSE (Alpha = 0.1) = 4,909.85
Part (h): Test RMSE = 4,981.09
Conclusion: Gradient Boosting Regressor has a higher test RMSE compared to the linear models with ElasticNet, suggesting poorer predictive accuracy on the test data.
R^2 Score:

Part (c): Train R^2 = 0.0254, Test R^2 = 0.0061
Part (g): Best Test R^2 (Alpha = 0.1) = 0.0071
Part (h): Test R^2 = -0.0219
Conclusion: Gradient Boosting Regressor has the lowest R^2 score on the test data, indicating it explains less variance in the test set compared to linear models.
Adjusted R^2 Score:

Part (c): Train Adjusted R^2 = 0.0254, Test Adjusted R^2 = 0.0061
Part (g): Best Test Adjusted R^2 (Alpha = 0.1) = -0.0940
Part (h): Test Adjusted R^2 = -0.1063
Conclusion: The Gradient Boosting Regressor shows the lowest Adjusted R^2 score, suggesting it is the least effective model in terms of explaining the variance in the target variable on the test data.
Mean Absolute Error (MAE):

Part (c): Train MAE = 3,976.70, Test MAE = 3,797.49
Part (g): Best Test MAE (Alpha = 0.1) = 3,798.96
Part (h): Test MAE = 3,841.47
Conclusion: The MAE for the Gradient Boosting Regressor is slightly higher than that of linear models with ElasticNet, indicating that its predictions are less accurate.

DONE