SECTION - B

Comment on the convergence of the model

The model converges instantly in less than 10 epochs. Loss converges to around 0.44 whereas accuracy converges to about 84%

Compare and analyze the plots

Both training and validation loss converge to similar values. At iteration 50, Training Loss: 0.4352082880352969
and Validation Loss: 0.4411242792992432. So we can say that the model has both, low bias and low variance.
Similarly, the model records similar accuracy on both training and validation accuracies with training accuracy being slightly higher than the validation accuracy.

Compare and discuss the impact of feature scaling on model convergence

As compared to the previous plot where the model was applied to unscaled data, this time the loss converges much more gradually and consistently decreases till the last epoch. Again, the training loss is slightly lesser than the validation loss but both are similar in value. Training and validation accuracies follow a similar trend with training accuracy being slightly higher than the validation accuracy.
In conclusion, feature scaling results in a more stable convergence.

Precision = 0.1832 --- Precision measures the proportion of correctly predicted positive observations to the total predicted positives. Low precision score indicates that the model reports a high number of false positives (FP) which is true. As we can see our model reports 312 False Positives.

Recall (or Sensitivity) = 0.8235 --- Recall measures the proportion of correctly predicted positive observations to all actual positives. A high recall value indicates that most of the actual postives are identifies by our model as positives. This means a lesser number of False Negatives (15).

F1 Score = 0.2998 --- The F1 Score is the harmonic mean of Precision and Recall, balancing both metrics. It is especially useful when we need a single score that balances both precision and recall. A low F1 score means that the model either has poor precision, recall, or both.

ROC-AUC score = 0.6556 --- The ROC-AUC (Receiver Operating Characteristic - Area Under the Curve) score measures the model's ability to distinguish between classes by plotting the True Positive Rate (Recall) against the False Positive Rate (1 - Specificity). An ROC-AUC score of 0.5 indicates that the model is performing no better than random guessing, while a score closer to 1.0 indicates strong discriminatory power.

Trade Off in terms of convergence speed and stability between these methods

Both Stochastic Gradient Descent (SGD) and Mini-Batch Gradient Descent (MBGD) are optimization algorithms that minimize the loss function by updating the model's parameters iteratively based on the gradient of the loss function. The key difference lies in how much data they use in each iteration to compute the gradient.
In SGD, the gradient of the loss function is computed using only one sample at a time.
In MBGD, the gradient is computed based on a small subset (mini-batch) of the training data rather than a single point or the entire dataset.

Convergence speed - From the graphs we can see that SGD results in a faster initial convergence as compared to MBGD. This might be because of frequent updates in SGD and slower updates in MBGD as the gradient is averaged over a mini-batch.

Stability - From the graphs we can see that SGD shows a higher variance in updates specially after converging. It seems to oscillate around the optimum due to noisy gradients. Whereas, MBGD shows a smoother and more stable convergence.

Accuracy:
Average Accuracy: 0.7934 ± 0.1057
Accuracy is relatively high in most folds, indicating that the model correctly classifies a large proportion of both spam and legitimate emails. However, the high standard deviation (0.1057) suggests that the model's performance is not consistent across all folds. This variability indicates that the model's accuracy may depend heavily on the particular split of the dataset, meaning it may generalize poorly on unseen data.

Precision:
Average Precision: 0.1020 ± 0.1344
Precision is quite low on average, with a high standard deviation (0.1344). This suggests that when the model predicts an email as spam, it often makes incorrect predictions, i.e., it frequently flags legitimate emails as spam (false positives). A higher variance in precision indicates the model struggles to consistently differentiate between spam and legitimate emails across different data splits.

Recall:
Average Recall: 0.1143 ± 0.2215
Recall is low overall, meaning the model is not effectively identifying spam emails (many spam emails are missed and classified as legitimate). The high variance (0.2215) suggests that the model's ability to detect spam varies significantly across folds, further indicating that the model's ability to identify spam is unstable and inconsistent across different dataset splits.

F1 Score:
Average F1 Score: 0.0592 ± 0.1050
The F1 score, which balances precision and recall, is quite low, indicating poor performance when it comes to balancing the trade-off between false positives and false negatives. A high standard deviation (0.1050) again reflects variability in the model's performance across different data splits.
Analyze the effect of early stopping on overfitting and generalization.

Effect of Early Stopping on Overfitting:

In a typical training process, training loss decreases steadily as the model improves its predictions on the training set.
Initially, validation loss also decreases as the model improves its ability to generalize. However, after a certain point, the validation loss starts increasing even though the training loss continues to decrease. This is a sign of overfitting, where the model is memorizing the training data rather than learning generalized patterns.

Effect of Early Stopping on Generalization:

Early stopping helps the model maintain a balance between underfitting and overfitting. By halting training early, the model is prevented from fitting too closely to the training data, thus maintaining its ability to generalize to new data.

In our case, we don't witness early stopping as the val loss continuously decreases and never increases (at least for the epochs that it was feasible for me to run). But we use 2 conditions for early stopping. One is if the val loss is more than the best val loss for more than (patience) no. of times. Other criteria is if the val loss decreases but decreases less than a given small value. For eg. 1e-5, this means that the model has almost converged and more training does not result in better performance.

SECTION - C

a)

1. The pair plots indicate weak relationship between the features, The data points are fairly scattered, which shows that the majority of relationships between the variables are not strongly linear.
2. Some scatterplots show potential outliers, which should be removed to make it easier for the model to learn the true pattern and captuare the true performance of the model during evalutation.
3. The values for Construction_Year are tightly packed, indicating that most buildings in the dataset were built within a narrow range of years. This is consistent in both the violin and box plots.
4. Both the violin and box plots indicate that the majority of values for Waste_Recycled_Percentage are concentrated towards the lower end, likely indicating that a smaller percentage of waste is recycled in most cases.
5. The correlation heat map indicates that the target variable Electricity_Bill shows very low correlation with all other features i.e. none of the features have a strong linear relationship with the target.

b)

The features when reduced to 2 dimensions and plotted as a scatter plot separate out into 2 major clusters, one comparatively larger than the other.
Showing 'Electricity_Bill' as the colour of the sample points denoted by the 2 dimensional features, we see that there is no clear separability of the data based on the target value or label i.e. the 'Electricity_Bill'.

c)

Results obtained in part c) –

```
Train Set Metrics:
  MSE: 24475013.1685
  RMSE: 4947.2228
  MAE: 4006.3285
  R2 Score: 0.0139
  Adjusted R2 Score: -0.0011


Test Set Metrics:
  MSE: 24278016.1557
  RMSE: 4927.2727
  MAE: 3842.4093
  R2 Score: 0.0000
  Adjusted R2 Score: -0.0641
```

d)

Results obtained in part d) —

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


Model with Top 3 Features Selected using RFE:


  MSE (Train): 24673540.3115, MSE (Test): 24181190.6472
  RMSE (Train): 4967.2468, RMSE (Test): 4917.4374
  MAE (Train): 4006.7840, MAE (Test): 3825.6516
  R2 Score (Train): 0.0059, R2 Score (Test): 0.0040
  Adjusted R2 Score (Train): 0.0029, Adjusted R2 Score (Test): -0.0081
```

The performance metrics (MSE, RMSE, MAE, R^2 score, and adjusted R^2 score) did not change significantly after selecting the top 3 features using RFE. This suggests that the remaining features might not have had a significant contribution to the model's predictive power.

e)

Results obtained in part e) –

```
Ridge Regression with One-Hot Encoded Features:
Train Set Metrics:
  MSE: 24188934.3377
  RMSE: 4918.2247
  MAE: 3976.7359
  R2 Score: 0.0254
  Adjusted R2 Score: 0.0066

Test Set Metrics:
  MSE: 24150009.7223
  RMSE: 4914.2659
  MAE: 3799.7460
  R2 Score: 0.0053
  Adjusted R2 Score: -0.0769
```

The R^2 score for Ridge on the train set is 0.0254, slightly better than Linear Regression's 0.0139, indicating Ridge is capturing slightly more variance.

On the test set, Ridge still shows a very marginal improvement (R^2 of 0.0053) compared to 0.0000 for Linear Regression, but the improvement is minimal.

The adjusted R^2 score is slightly better for Ridge on the train set (0.0066 vs. -0.0011). However, both are quite low, meaning the models aren't performing well overall.

On the test set, both models have negative adjusted R^2 scores (Ridge: -0.0769, Linear: -0.0641), which confirms that neither model generalizes well to the test data.

f)

Results obtained in part f) –

```
Results for 4 Components:
Train MSE: 24800891.1751, Test MSE: 24361808.6516
Train RMSE: 4980.0493, Test RMSE: 4935.7683
Train MAE: 4008.4981, Test MAE: 3841.8032
Train R²: 0.0008, Test R²: -0.0034
Adjusted Train R²: -0.0032, Adjusted Test R²: -0.0198
```

```
Results for 5 Components:
Train MSE: 24686731.6985, Test MSE: 24409298.1348
Train RMSE: 4968.5744, Test RMSE: 4940.5767
Train MAE: 4010.5033, Test MAE: 3844.6135
Train R²: 0.0054, Test R²: -0.0054
Adjusted Train R²: 0.0004, Adjusted Test R²: -0.0260

Results for 6 Components:
Train MSE: 24684778.5818, Test MSE: 24387370.2384
Train RMSE: 4968.3779, Test RMSE: 4938.3570
Train MAE: 4010.7220, Test MAE: 3841.9533
Train R²: 0.0055, Test R²: -0.0045
Adjusted Train R²: -0.0005, Adjusted Test R²: -0.0293

Results for 8 Components:
Train MSE: 24674722.1792, Test MSE: 24416610.3106
Train RMSE: 4967.3657, Test RMSE: 4941.3167
Train MAE: 4012.3616, Test MAE: 3848.6541
Train R²: 0.0059, Test R²: -0.0057
Adjusted Train R²: -0.0021, Adjusted Test R²: -0.0391
```

Across all components (4, 5, 6, and 8), the Train MSE, RMSE, and MAE are very close to each other, indicating that increasing the number of components does not significantly change the training performance of the model.

There is no significant improvement when increasing the no. of components beyond 4. In Fact the performance deteriorates upon adding more components. Thus, adding more components does not help the model capture more variance or reduce the error.

Given that Train R^2 is marginally improving, but Test R^2 is decreasing, this indicates that adding components may lead to overfitting on the training set while not improving generalization to the test set.

g)

Results obtained in g) —

```
Results for alpha = 0.01:
Train MSE: 24189141.6183, Test MSE: 24142794.2823
Train RMSE: 4918.2458, Test RMSE: 4913.5318
Train MAE: 3976.8955, Test MAE: 3799.6651
Train R²: 0.0254, Test R²: 0.0056
Adjusted Train R²: 0.0065, Adjusted Test R²: -0.0765

Results for alpha = 0.1:
Train MSE: 24202509.1014, Test MSE: 24094484.7977
Train RMSE: 4919.6046, Test RMSE: 4908.6133
Train MAE: 3979.2026, Test MAE: 3800.4843
Train R²: 0.0249, Test R²: 0.0076
Adjusted Train R²: 0.0060, Adjusted Test R²: -0.0744

Results for alpha = 1.0:
Train MSE: 24388144.1435, Test MSE: 24106705.5742
Train RMSE: 4938.4354, Test RMSE: 4909.8580
Train MAE: 3993.3637, Test MAE: 3812.2908
Train R²: 0.0174, Test R²: 0.0071
Adjusted Train R²: -0.0016, Adjusted Test R²: -0.0749

Results for alpha = 10.0:
Train MSE: 24708524.8509, Test MSE: 24289066.8934
Train RMSE: 4970.7670, Test RMSE: 4928.3939
Train MAE: 4005.2215, Test MAE: 3835.0942
Train R²: 0.0045, Test R²: -0.0004
Adjusted Train R²: -0.0148, Adjusted Test R²: -0.0831
```

Observations -

As the alpha value increases, the regularization penalty becomes stronger, and the model is forced to fit a simpler solution. This can be observed in the decline of Train R^2 from 0.0254 (for alpha = 0.01) to 0.0045 (for alpha = 10.0).

The model becomes too constrained which leads to underfitting, where it cannot capture the complexity of the data adequately. This results in worse performance on both training and test sets as regularization increases.

The best Test R^2 is only around 0.0076, indicating that even at lower alpha values, the model is not performing well.

h)

Results obtained in h) —

```
Gradient Boosting Regressor Results:
Train MSE: 15548098.7804, Test MSE: 25053601.6302
Train RMSE: 3943.1078, Test RMSE: 5005.3573
Train MAE: 3155.7775, Test MAE: 3832.7926
Train R²: 0.3736, Test R²: -0.0319
Adjusted Train R²: 0.3614, Adjusted Test R²: -0.1172
```

Observations

Gradient Boosting Regressor has the lowest Train MSE (15,548,098) compared to both Simple Linear Regression (24,475,013) and ElasticNet (24,189,141). This suggests that Gradient Boosting fits the training data much better.

The Train R^2 for Gradient Boosting is 0.3736, which indicates that the model explains 37.36% of the variance in the training set. In contrast, Simple Linear Regression ($R^2 = 0.0139$) and ElasticNet ($R^2 = 0.0254$) explain much less variance in the training data.

Test MSE for Gradient Boosting is 25,053,601, which is slightly higher than both Simple Linear Regression (24,278,016) and ElasticNet (24,142,794). This shows that Gradient Boosting overfits the training data to some extent, resulting in worse generalization on the test data.

Test R^2 for Gradient Boosting is -0.0319, which is negative and worse than ElasticNet (0.0056) and Simple Linear Regression (0.0000).