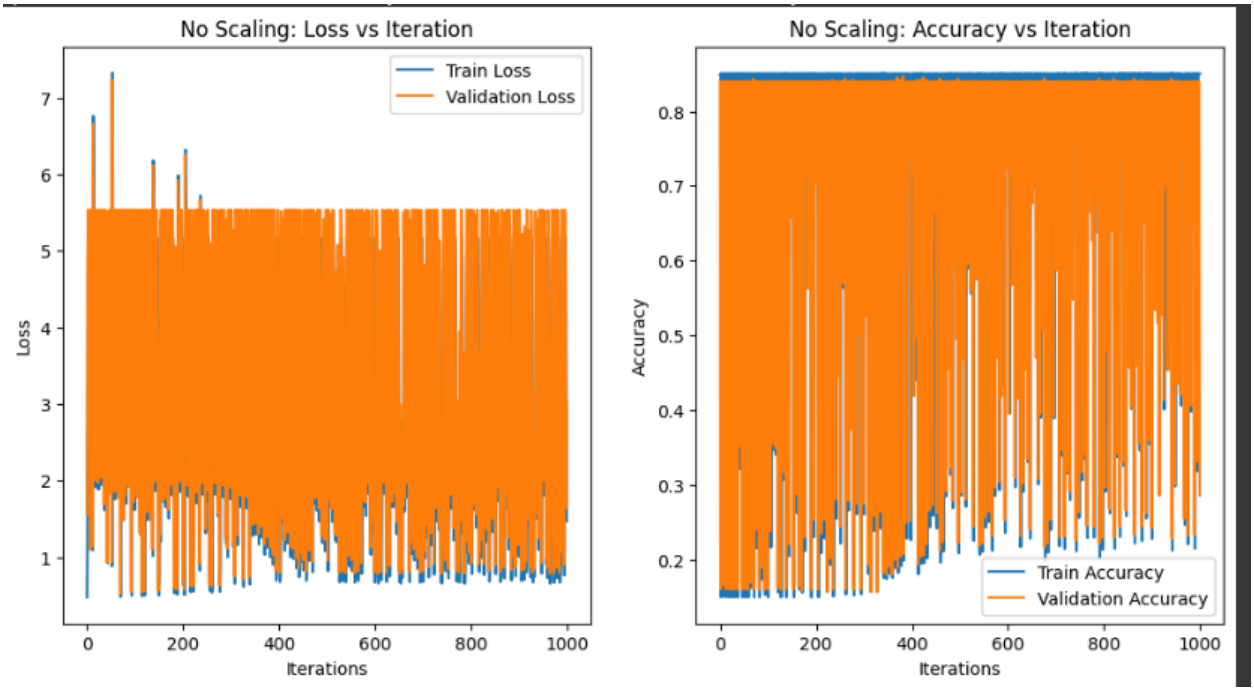
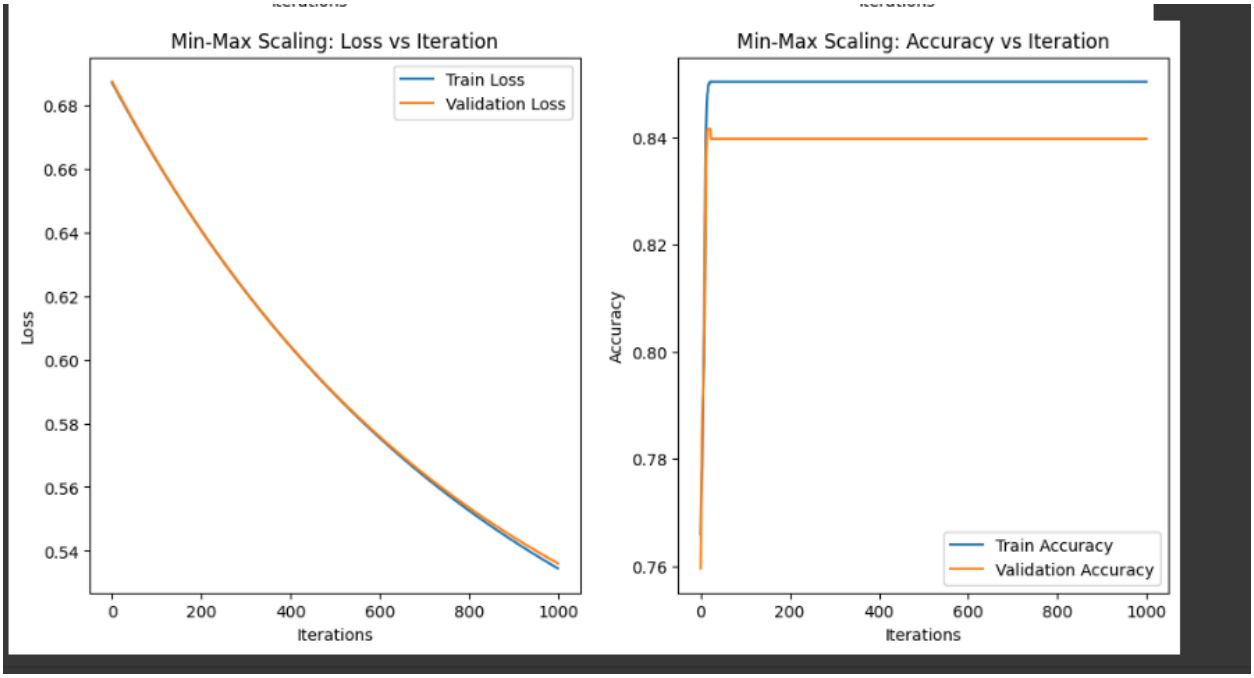
ML ASSIGNMENT REPORT
DEV UTKARSH PAL
2022150





No convergence, so I had to scale the data, but after using min max scaling, the graphs for accuaracy (to 0.9) and loss are converging to a value (to 0.54).

```
Evaluation of Standard Scaled Model:
Confusion Matrix:
[[239 222]
 [ 33  55]]
Precision: 0.1986
Recall: 0.6250
F1 Score: 0.3014
ROC-AUC Score: 0.6143
```

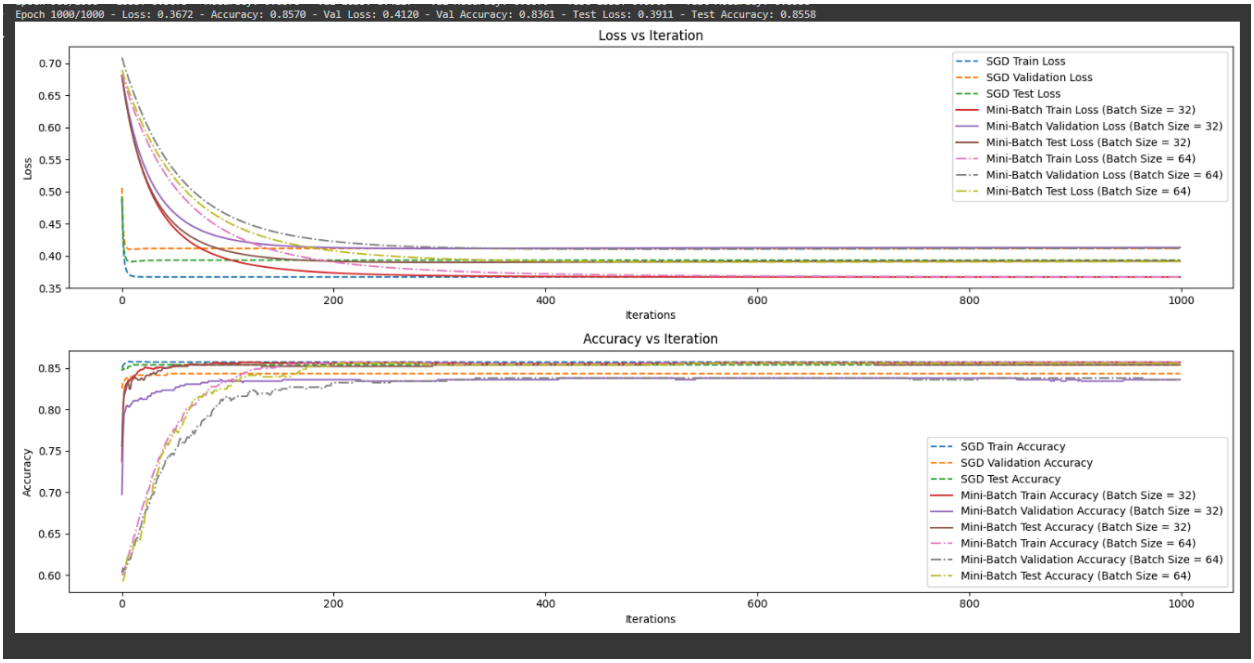
The Recall is hihg => my model is detecting true positives more (TP/TP+FP) though the precision is low and F1 is 30% => 30% were correct, ROC is 62% telling that it differs between true and false cases on a fine level.

part d:

```
Training using Stochastic Gradient Descent (SGD):
Epoch 100/1000 - Loss: 0.3672 - Accuracy: 0.8574 - Val Loss: 0.4116 - Val Accuracy: 0.8434 - Test Loss: 0.3933 - Test Accuracy: 0.8540
Epoch 200/1000 - Loss: 0.3672 - Accuracy: 0.8574 - Val Loss: 0.4116 - Val Accuracy: 0.8434 - Test Loss: 0.3933 - Test Accuracy: 0.8540
Epoch 300/1000 - Loss: 0.3672 - Accuracy: 0.8574 - Val Loss: 0.4116 - Val Accuracy: 0.8434 - Test Loss: 0.3933 - Test Accuracy: 0.8540
Epoch 400/1000 - Loss: 0.3672 - Accuracy: 0.8574 - Val Loss: 0.4116 - Val Accuracy: 0.8434 - Test Loss: 0.3933 - Test Accuracy: 0.8540
Epoch 500/1000 - Loss: 0.3672 - Accuracy: 0.8574 - Val Loss: 0.4116 - Val Accuracy: 0.8434 - Test Loss: 0.3933 - Test Accuracy: 0.8540
Epoch 600/1000 - Loss: 0.3672 - Accuracy: 0.8574 - Val Loss: 0.4116 - Val Accuracy: 0.8434 - Test Loss: 0.3933 - Test Accuracy: 0.8540
Epoch 700/1000 - Loss: 0.3672 - Accuracy: 0.8574 - Val Loss: 0.4116 - Val Accuracy: 0.8434 - Test Loss: 0.3933 - Test Accuracy: 0.8540
Epoch 800/1000 - Loss: 0.3672 - Accuracy: 0.8574 - Val Loss: 0.4116 - Val Accuracy: 0.8434 - Test Loss: 0.3933 - Test Accuracy: 0.8540
Epoch 900/1000 - Loss: 0.3672 - Accuracy: 0.8574 - Val Loss: 0.4116 - Val Accuracy: 0.8434 - Test Loss: 0.3933 - Test Accuracy: 0.8540
Epoch 1000/1000 - Loss: 0.3672 - Accuracy: 0.8574 - Val Loss: 0.4116 - Val Accuracy: 0.8434 - Test Loss: 0.3933 - Test Accuracy: 0.8540

Training using Mini-Batch Gradient Descent (Batch Size = 32):
Epoch 100/1000 - Loss: 0.3953 - Accuracy: 0.8558 - Val Loss: 0.4269 - Val Accuracy: 0.8342 - Test Loss: 0.4077 - Test Accuracy: 0.8540
Epoch 200/1000 - Loss: 0.3736 - Accuracy: 0.8558 - Val Loss: 0.4127 - Val Accuracy: 0.8361 - Test Loss: 0.3916 - Test Accuracy: 0.8522
Epoch 300/1000 - Loss: 0.3691 - Accuracy: 0.8562 - Val Loss: 0.4116 - Val Accuracy: 0.8361 - Test Loss: 0.3899 - Test Accuracy: 0.8540
Epoch 400/1000 - Loss: 0.3678 - Accuracy: 0.8554 - Val Loss: 0.4119 - Val Accuracy: 0.8361 - Test Loss: 0.3902 - Test Accuracy: 0.8540
Epoch 500/1000 - Loss: 0.3673 - Accuracy: 0.8558 - Val Loss: 0.4123 - Val Accuracy: 0.8379 - Test Loss: 0.3908 - Test Accuracy: 0.8558
Epoch 600/1000 - Loss: 0.3671 - Accuracy: 0.8566 - Val Loss: 0.4126 - Val Accuracy: 0.8379 - Test Loss: 0.3912 - Test Accuracy: 0.8558
Epoch 700/1000 - Loss: 0.3670 - Accuracy: 0.8566 - Val Loss: 0.4128 - Val Accuracy: 0.8379 - Test Loss: 0.3916 - Test Accuracy: 0.8558
Epoch 800/1000 - Loss: 0.3670 - Accuracy: 0.8566 - Val Loss: 0.4129 - Val Accuracy: 0.8379 - Test Loss: 0.3918 - Test Accuracy: 0.8540
Epoch 900/1000 - Loss: 0.3670 - Accuracy: 0.8570 - Val Loss: 0.4130 - Val Accuracy: 0.8361 - Test Loss: 0.3919 - Test Accuracy: 0.8540
Epoch 1000/1000 - Loss: 0.3670 - Accuracy: 0.8570 - Val Loss: 0.4130 - Val Accuracy: 0.8361 - Test Loss: 0.3920 - Test Accuracy: 0.8540

Training using Mini-Batch Gradient Descent (Batch Size = 64):
Epoch 100/1000 - Loss: 0.4379 - Accuracy: 0.8343 - Val Loss: 0.4645 - Val Accuracy: 0.8160 - Test Loss: 0.4534 - Test Accuracy: 0.8248
Epoch 200/1000 - Loss: 0.3907 - Accuracy: 0.8558 - Val Loss: 0.4226 - Val Accuracy: 0.8324 - Test Loss: 0.4080 - Test Accuracy: 0.8540
Epoch 300/1000 - Loss: 0.3771 - Accuracy: 0.8558 - Val Loss: 0.4132 - Val Accuracy: 0.8342 - Test Loss: 0.3958 - Test Accuracy: 0.8540
Epoch 400/1000 - Loss: 0.3720 - Accuracy: 0.8554 - Val Loss: 0.4109 - Val Accuracy: 0.8379 - Test Loss: 0.3919 - Test Accuracy: 0.8540
Epoch 500/1000 - Loss: 0.3697 - Accuracy: 0.8558 - Val Loss: 0.4105 - Val Accuracy: 0.8379 - Test Loss: 0.3906 - Test Accuracy: 0.8540
Epoch 600/1000 - Loss: 0.3685 - Accuracy: 0.8558 - Val Loss: 0.4107 - Val Accuracy: 0.8379 - Test Loss: 0.3903 - Test Accuracy: 0.8558
Epoch 700/1000 - Loss: 0.3678 - Accuracy: 0.8558 - Val Loss: 0.4111 - Val Accuracy: 0.8379 - Test Loss: 0.3904 - Test Accuracy: 0.8558
Epoch 800/1000 - Loss: 0.3675 - Accuracy: 0.8570 - Val Loss: 0.4114 - Val Accuracy: 0.8361 - Test Loss: 0.3906 - Test Accuracy: 0.8558
Epoch 900/1000 - Loss: 0.3673 - Accuracy: 0.8570 - Val Loss: 0.4117 - Val Accuracy: 0.8379 - Test Loss: 0.3909 - Test Accuracy: 0.8558
Epoch 1000/1000 - Loss: 0.3672 - Accuracy: 0.8570 - Val Loss: 0.4120 - Val Accuracy: 0.8361 - Test Loss: 0.3911 - Test Accuracy: 0.8558
```



With and without early stopping: Because of early stopping, the loss isn't going as low as with no early stopping, so it is affecting it a bit.

But the accuracy is reaching at the saturation value and then stopping, in case of early stopping, as same as the case without stopping.

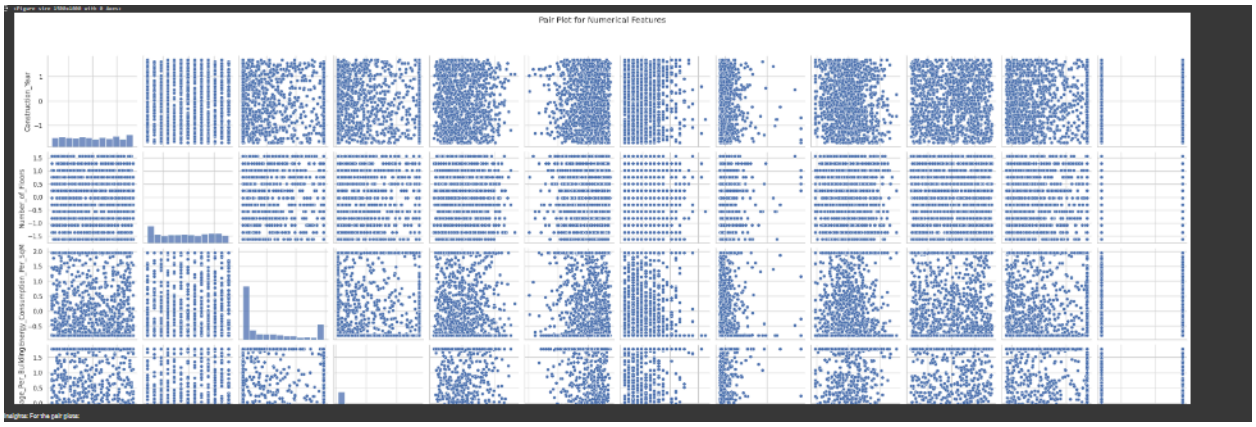
Effect on overfitting and generalization: Early stopping reduces overfitting and doing the same in our case, this helps the model generalize better to unseen data by preventing it from overfitting to specific details.

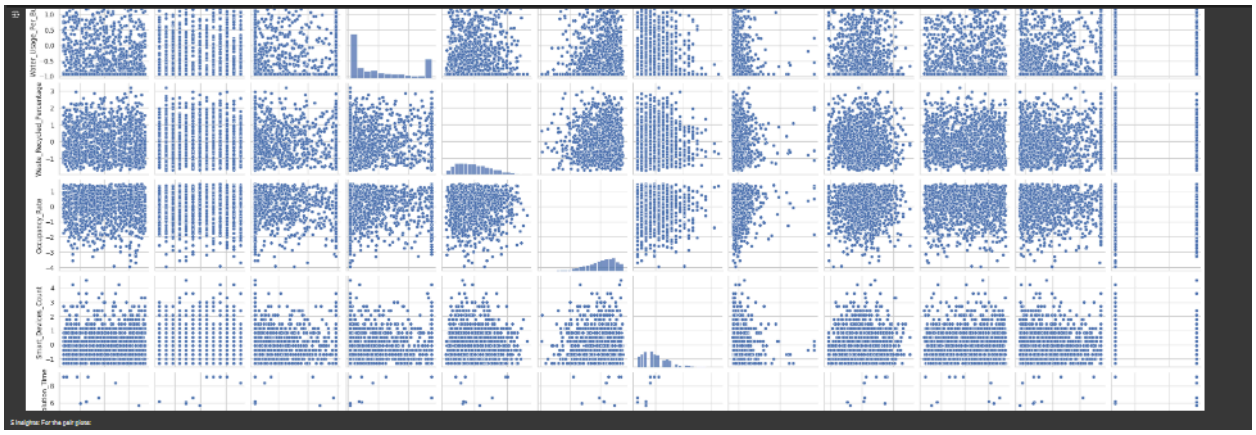
stopping too early can lead to underfitting, where the model fails to learn important patterns, so seeing the correct stopping point is necessary.

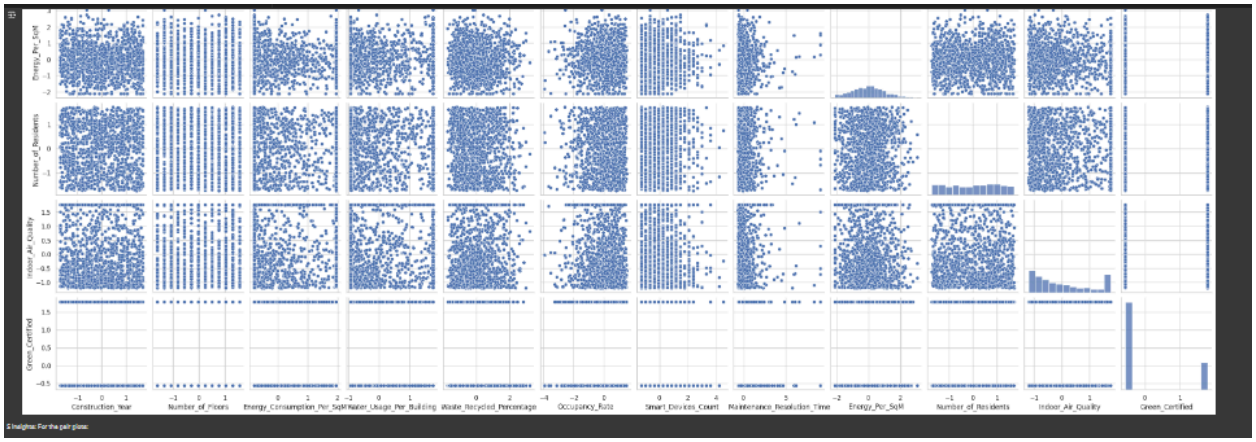
I used early stopping using a delta value of 0.01 which doesn't let the model overfit over the dataset, the threshold value can be set by checking the loss details empirically.

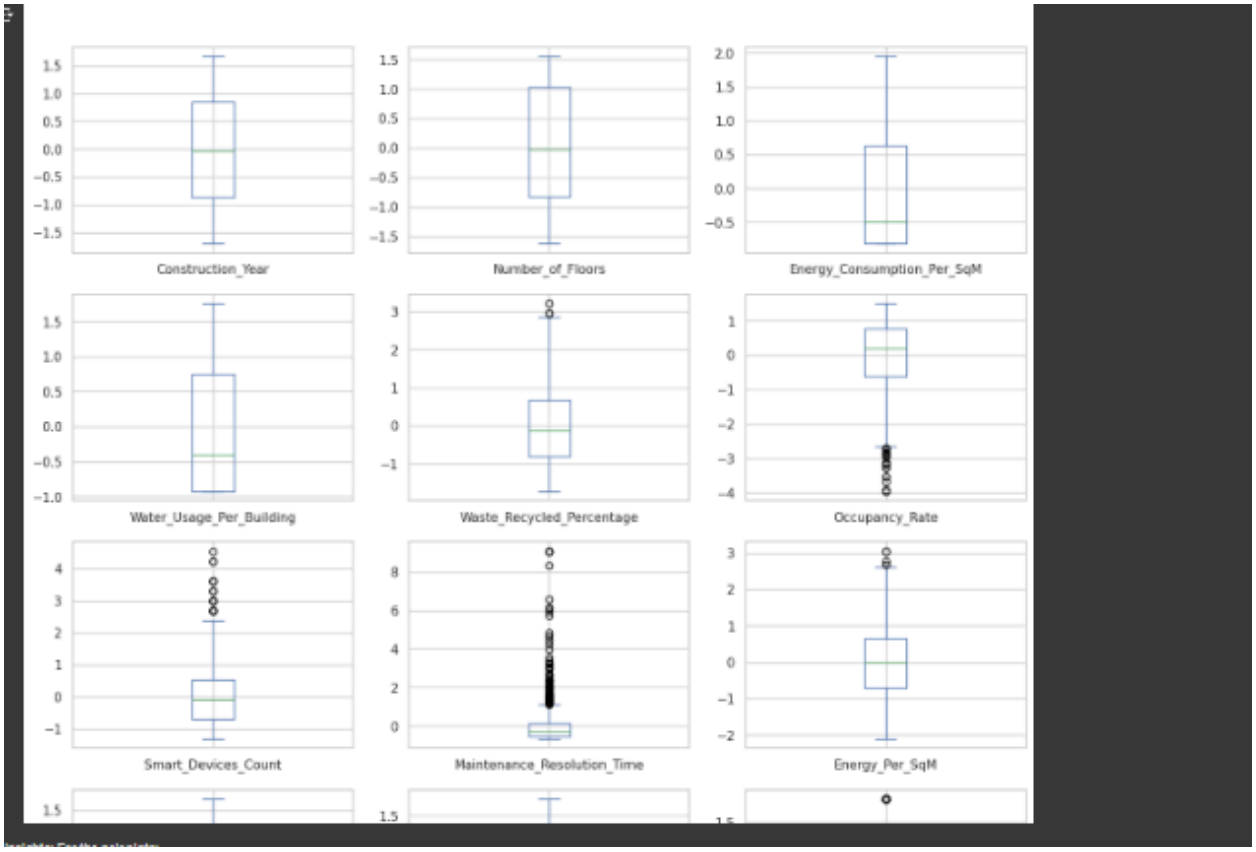
It can be seen from the plots that the accuracy stops early at near a saturation value and hence require less iterations.

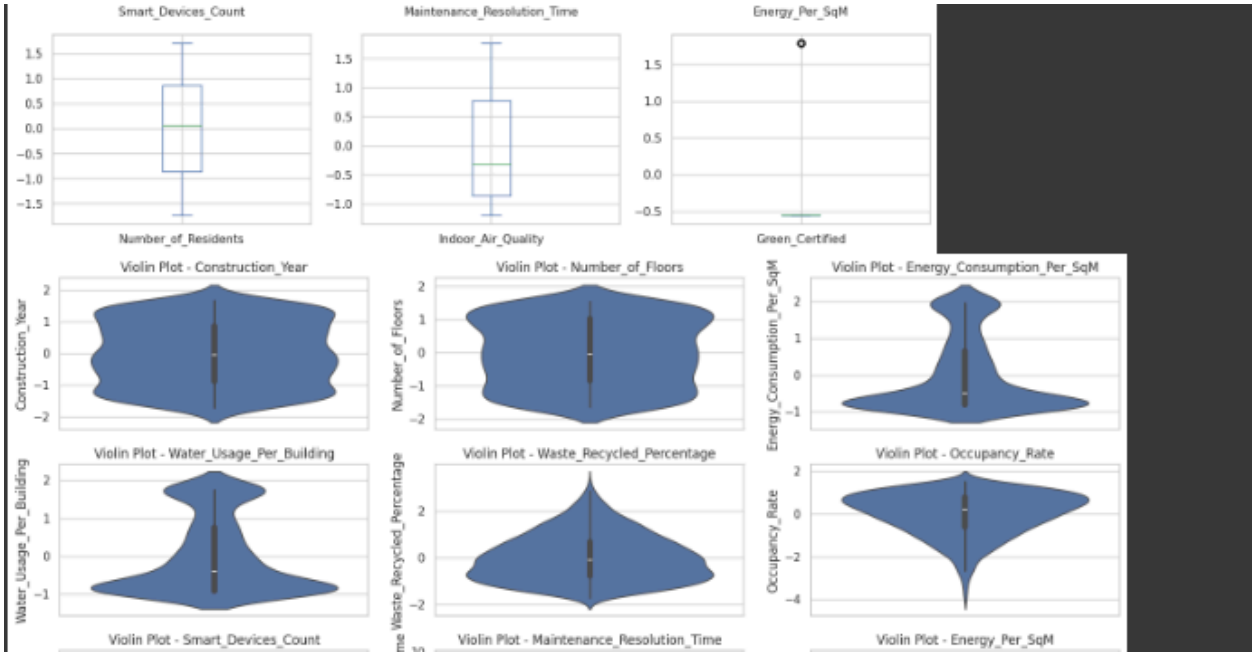
QUESTION 3:

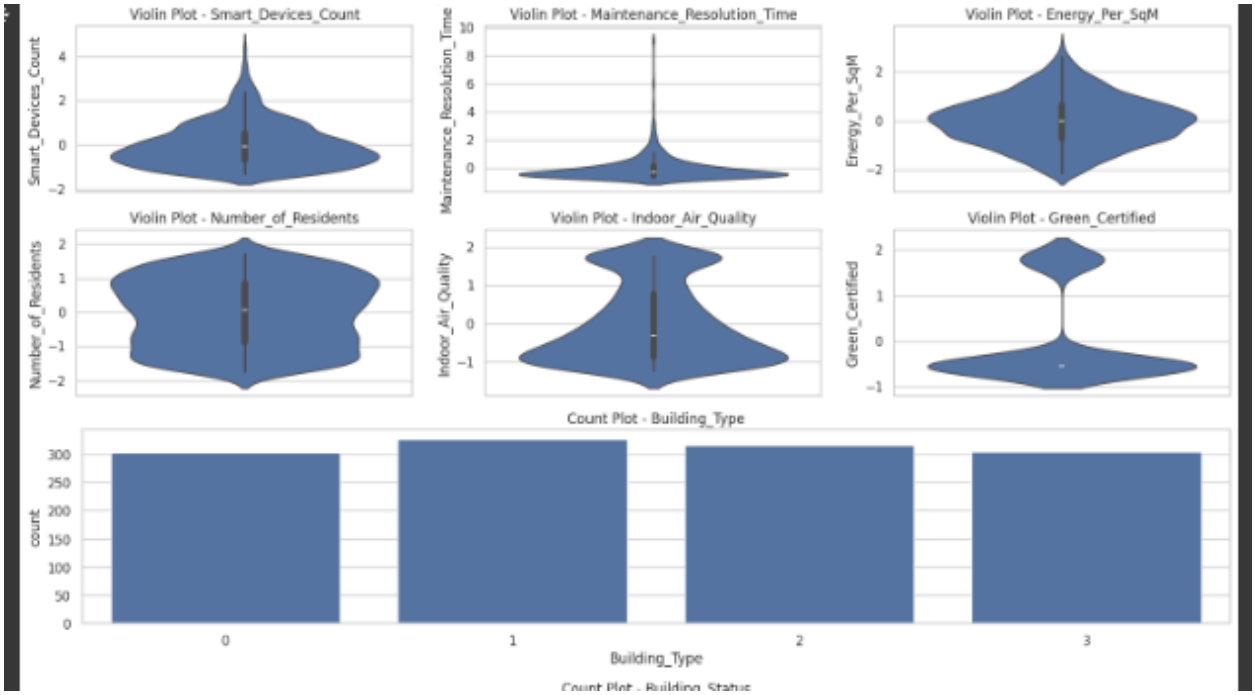


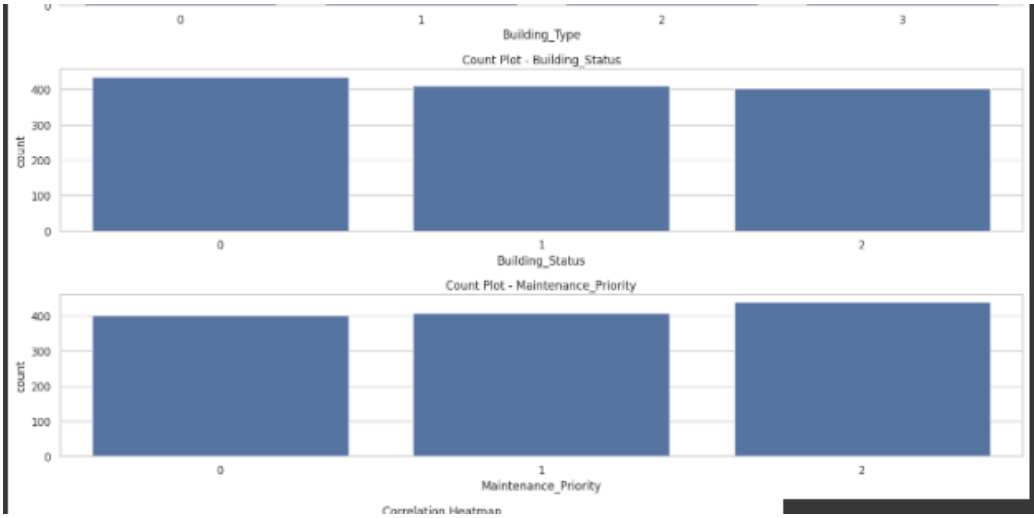


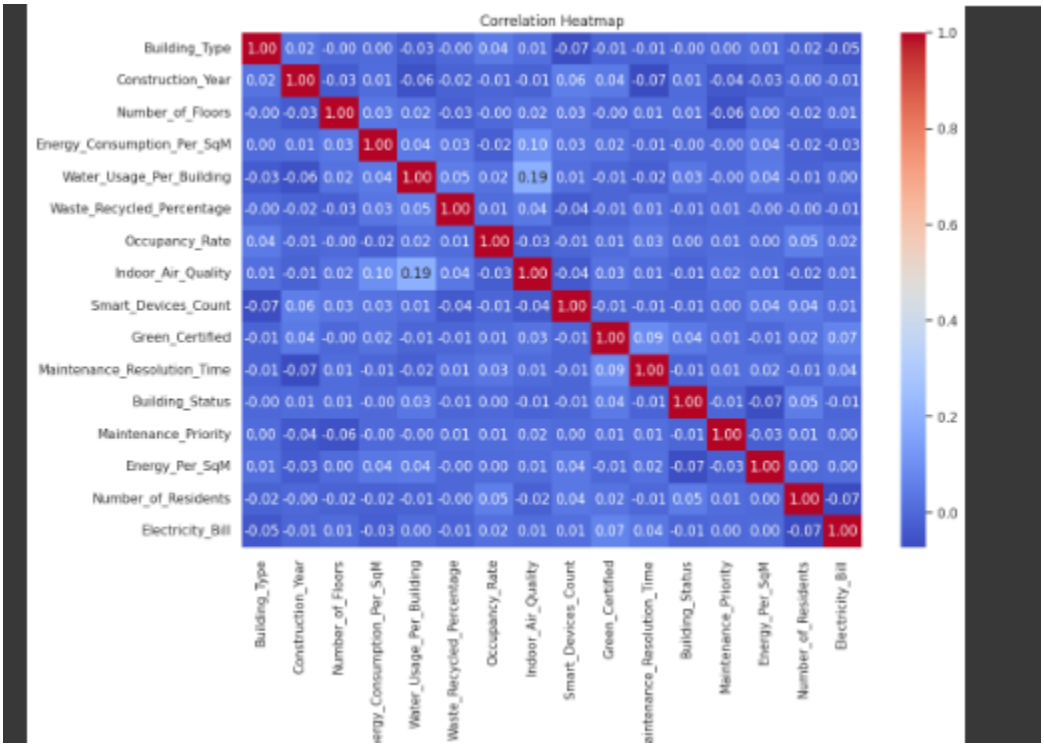

Energy_Per_SqM
Number_of_Residents
Indoor_Air_Quality
Green_Certified
Construction_Year
Number_of_Floors
Energy_Consumption_Per_SqM
Water_Usage_Per_Building
Waste_Recycled_Percentage
Occupancy_Rate
Smart_Devices_Count
Maintenance_Resolution_Time
Energy_Per_SqM
Number_of_Residents
Indoor_Air_Quality
Green_Certified


Construction_Year
Number_of_Floors
Energy_Consumption_Per_SqM
Water_Usage_Per_Building
Waste_Recycled_Percentage
Occupancy_Rate
Smart_Devices_Count
Maintenance_Resolution_Time
Energy_Per_SqM


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


Violin Plot - Smart_Devices_Count
Violin Plot - Maintenance_Resolution_Time
Violin Plot - Energy_Per_SqM
Violin Plot - Number_of_Residents
Violin Plot - Indoor_Air_Quality
Violin Plot - Green_Certified
Count Plot - Building_Type
count
Building_Type
Count Plot - Building_Status


Building_Type
Count Plot - Building_Status
count
Building_Status
Count Plot - Maintenance_Priority
count
Maintenance_Priority


Correlation Heatmap
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
Building_Status
Maintenance_Priority
Energy_Per_SqM
Number_of_Residents
Electricity_Bill

5 insights: For the pair plots:

1. Occupancy_rate and air_index_quality shows clustering.
2. many of the features had scattered points with respect to the electricity bill which shows that they are not going to be much helpful in determining the target data while others such as Green certified and number of floors and smart_devices_count had a pattern which might be useful.

Violin Plots:

3. The density in the plots help us in understanding the consistency of the distribution.
4. Green_certified and building_type show a narrowing in the middle which shows that there are outliers and extreme values in the distributions of the dataset.

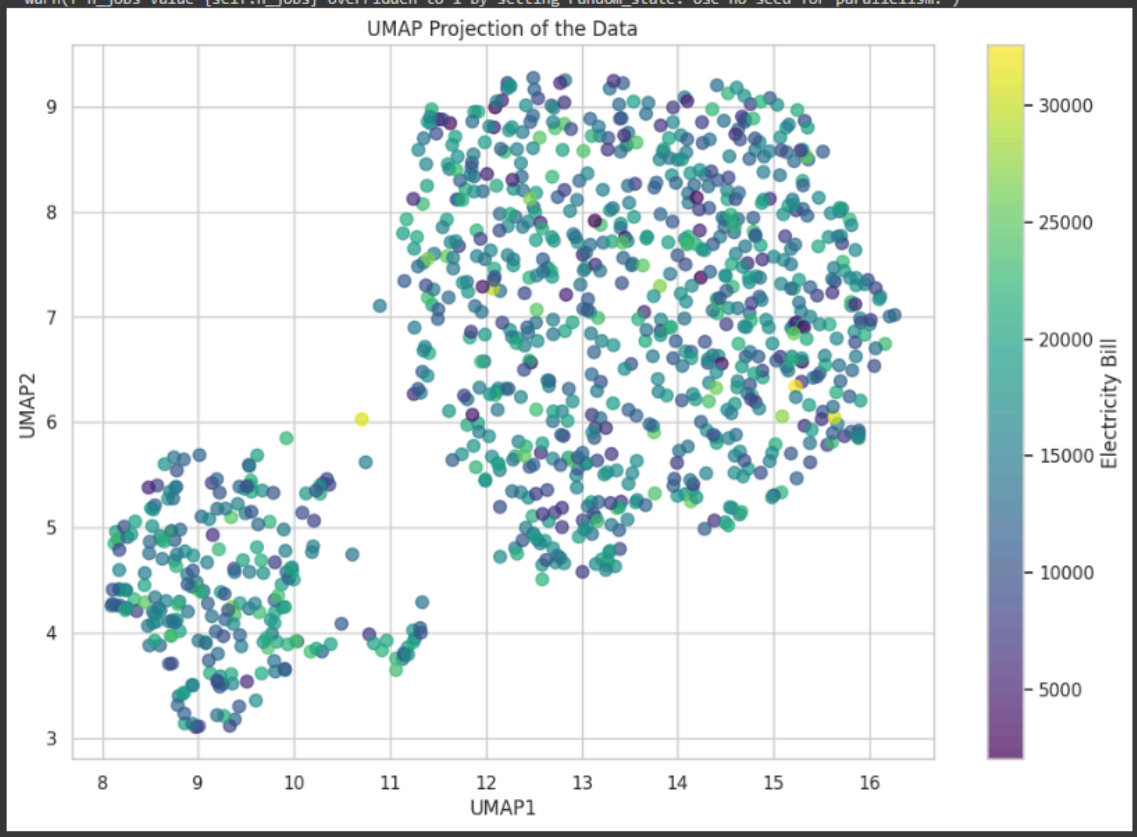
Box Plots:

5. The water_usage per building is centered around the mean with less outliers.
6. The data is symmetrix which means that water usage is balanced around the buildings.
7. occupancy rates are tightly packed around the median.

Heat Map:

8. There is a Strong Positive Correlation Between "Smart Devices Count" and "Green Certified" , the fields with the highest magnitude values are highly correlated.
9. The three features: number_of_residents, building_type and and Green_certified are the most dependent features for the electricity_bill as seen from the last row.

```
warn(f"n_jobs value {self.n_jobs} overridden to 1 by setting random_state. Use no seed for parallelism.")
```



There are two different clusters of data which shows that there is some kind of pattern.

But at the same time the high amount of scattering shows that the dataset is not good enough and the distribution is random.

```
Train Data:
MSE: 24475013.16847547
RMSE: 4947.222773281538
MAE: 4006.3284693293604
R2 Score: 0.013922520844610209
Adjusted R2 Score: -0.0011091480449536562

Test Data:
MSE: 24278016.155742623
RMSE: 4927.272689403604
MAE: 3842.409312558516
R2 Score: 3.7344733075372893e-05
Adjusted R2 Score: -0.0640628254763429
```

```
Top 3 selected features: ['Number_of_Residents', 'Green_Certified', 'Building_Type']

Train Data with Selected Features (RFE):
MSE: 24569032.906897984
RMSE: 4956.715939702212
MAE: 4006.473377514736
R2 Score: 0.010134545491284008
Adjusted R2 Score: 0.007153023037944517

Test Data with Selected Features (RFE):
MSE: 23941409.062998377
RMSE: 4892.995918964002
MAE: 3813.948128176773
R2 Score: 0.013901513867941l4
Adjusted R2 Score: 0.0018759225736477703
```

comparison with part c:

Mean Squared Error (MSE): Train Data: Slightly higher with RFE (24,569,032.91) compared to the part c model (24,475,013.17).

Test Data: Lower with RFE (23,941,409.06) compared to the part c model (24,278,016.16).

Root Mean Squared Error (RMSE): Train Data: Slightly higher with RFE (4,956.72) compared to the original model (4,947.22).

Test Data: Lower with RFE (4,892.99) compared to the original model (4,927.27).

R2 Score: Train Data: Slightly lower with RFE (0.0101) compared to the original model (0.0139).

Test Data: Higher with RFE (0.0139) compared to the original model (0.000037).

Performance on Training Data: The RFE model shows a slightly higher MSE and RMSE compared to the original model but with very similar MAE. The R2 and Adjusted R2 scores are slightly lower with RFE.

Ant the performance on the Test Data:The RFE model shows improvements in MSE, RMSE, MAE, and R2 compared to the original model. The Adjusted R2 score is also better with RFE.

test e:

```
Train Data (Ridge Regression with One-Hot Encoded Categorical Features):
MSE: 24188934.340711236
RMSE: 4918.22471433659
MAE: 3976.735596585223
R2 Score: 0.025448393675797765
Adjusted R2 Score: 0.006554025798083618

Test Data (Ridge Regression with One-Hot Encoded Categorical Features):
MSE: 24128288.424410056
RMSE: 4912.055417481572
MAE: 3797.5125456363867
R2 Score: 0.006204328840423434
Adjusted R2 Score: -0.0758918352988460Z
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:975: FutureWarning: `sparse` was renamed to `sparse_output` in version 1.2 and will be removed in 1.4. `sparse_output` is ignored unless you leave `sparse` to its default value.
  warnings.warn(
```

Overall Performance: Ridge Regression with one-hot encoding generally outperforms the original model in terms of error metrics, particularly on the training data. The RFE model, while performing well on test data, does not consistently outperform Ridge Regression.

Feature Selection Impact: The Ridge Regression model benefits from incorporating one-hot encoded categorical features, showing improved performance over the original model. RFE did improve test data performance but was less effective on training data compared to Ridge Regression.

```
Evaluating ICA with 4 components:
MSE Train: 24691015.1735
MSE Test: 24445533.4369
RMSE Train: 4969.0055
RMSE Test: 4944.2425
MAE Train: 4013.6441
MAE Test: 3852.4464
R2 Train: 0.0052
R2 Test: -0.0069
Adjusted R2 Train: 0.0012
Adjusted R2 Test: -0.0233

Evaluating ICA with 5 components:
MSE Train: 24665150.1429
MSE Test: 24499301.9259
RMSE Train: 4966.4021
RMSE Test: 4949.6770
MAE Train: 4013.1603
MAE Test: 3850.9878
R2 Train: 0.0063
R2 Test: -0.0091
Adjusted R2 Train: 0.0013
Adjusted R2 Test: -0.0298

Evaluating ICA with 6 components:
MSE Train: 24663787.0959
MSE Test: 24473494.4960
RMSE Train: 4966.2649
RMSE Test: 4947.0693
MAE Train: 4013.2170
MAE Test: 3847.9924
R2 Train: 0.0063
R2 Test: -0.0080
Adjusted R2 Train: 0.0003
Adjusted R2 Test: -0.0329

Evaluating ICA with 8 components:
MSE Train: 24634815.0313
MSE Test: 24610827.6363
RMSE Train: 4963.3472
RMSE Test: 4960.9301
MAE Train: 4021.0477
MAE Test: 3867.3050
R2 Train: 0.0075
R2 Test: -0.0137
Adjusted R2 Train: -0.0005
Adjusted R2 Test: -0.0473
```

```
ElasticNet Regularization Evaluation Metrics:
   Alpha     MSE Train   RMSE Train    MAE Train  R2 Train  Adjusted R2 Train  \
0    0.1  2.420252e+07  4919.605309  3979.192304  0.024901           0.005996
1    0.5  2.429893e+07  4929.394570  3987.616848  0.021017           0.002036
2    1.0  2.438829e+07  4938.450376  3993.314394  0.017416          -0.001634
3    2.0  2.449485e+07  4949.227251  3998.095697  0.013123          -0.006010
4    5.0  2.463037e+07  4962.899854  4002.744192  0.007663          -0.011576

       MSE Test    RMSE Test     MAE Test   R2 Test  Adjusted R2 Test
0  2.407340e+07  4906.464753  3797.967118  0.008465         -0.073444
1  2.405711e+07  4904.805259  3803.691440  0.009136         -0.072718
2  2.409151e+07  4908.310642  3810.134469  0.007719         -0.074252
3  2.414857e+07  4914.119642  3819.139026  0.005369         -0.076796
4  2.423335e+07  4922.737818  3828.868990  0.001877         -0.080576
```

part h:

```
Train Data (Gradient Boosting Regressor):
MSE: 15548098.780395454
RMSE: 3943.107756299723
MAE: 3155.777526146695
R2 Score: 0.373580314523429
Adjusted R2 Score: 0.3614354430703118 3

Test Data (Gradient Boosting Regressor):
MSE: 24868467.60197402
RMSE: 4986.82941376322
MAE: 3850.400448176846
R2 Score: -0.02428216276672046
Adjusted R2 Score: -0.1088967762126691

Comparison of Results:
Linear Regression Test Data - MSE: 24233347.620866563, RMSE: 4922.737817603794, MAE: 3828.8689898388943, R2 Score: 0.0018771518431379697, Adjusted R2 Score: -0.0805764747437332
Ridge Regression Test Data - MSE: 24128288.424410056, RMSE: 4912.055417481572, MAE: 3797.5125456363867, R2 Score: 0.006204328840423434, Adjusted R2 Score: -0.0758918352988460 2
Gradient Boosting Regressor Test Data - MSE: 24868467.60197402, RMSE: 4986.82941376322, MAE: 3850.400448176846, R2 Score: -0.02428216276672046, Adjusted R2 Score: -0.1088967762126691
```

Performance on Test Data: The Gradient Boosting Regressor performs worse across all metrics compared to both Linear Regression and Ridge Regression. It has the highest MSE, RMSE, and MAE, and the lowest R2 and Adjusted R2 scores, indicating it is less effective at making accurate predictions and explaining the variance in the target variable.

Model Effectiveness: Ridge Regression shows the best performance among the models in terms of both MSE and MAE, with relatively low RMSE and better R2 scores. Linear Regression performs slightly worse than Ridge Regression but better than Gradient Boosting Regressor. Gradient Boosting Regressor, despite being a more complex model, does not show improvements over the simpler models in this case, possibly due to overfitting or inappropriate hyperparameters for this dataset. g In summary, Ridge Regression seems to be the most effective model for this particular dataset based on the metrics provided.