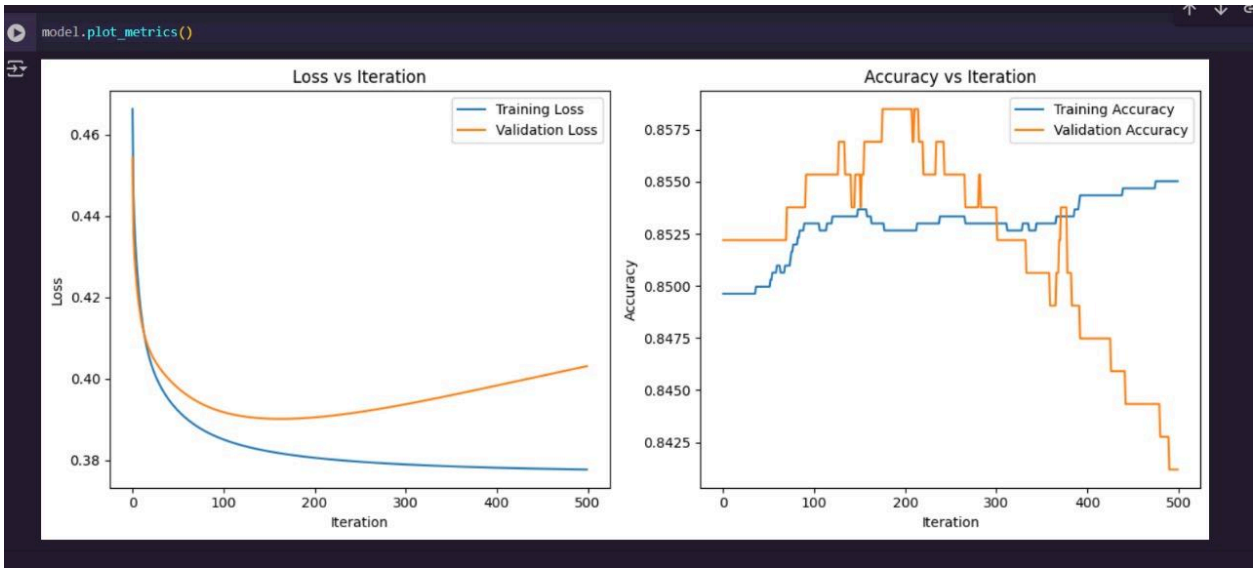
CSE343: Machine Learning
Assignment-1

REPORT

Anushka Srivastava (2022086)

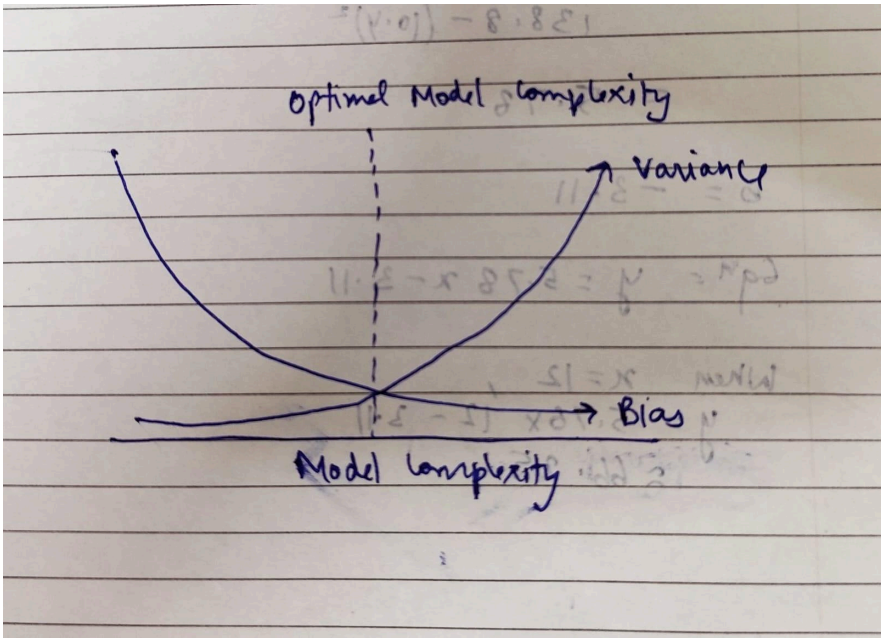
SECTION A

a. On increasing the complexity of the model, the model will most likely end up overfitting, that is it will perform extremely well on training data but underperform on the testing dataset as the model would fail to generalize on unseen data. In terms of bias and variance, the bias of the model decreases significantly but the variance of model performance increases.



In this graph, we see that while the training loss is decreasing and training accuracy is increasing, the validation loss is increasing, and the validation accuracy is decreasing. This indicates that the model has overfitted.

Bias and Variance graph vs model complexity

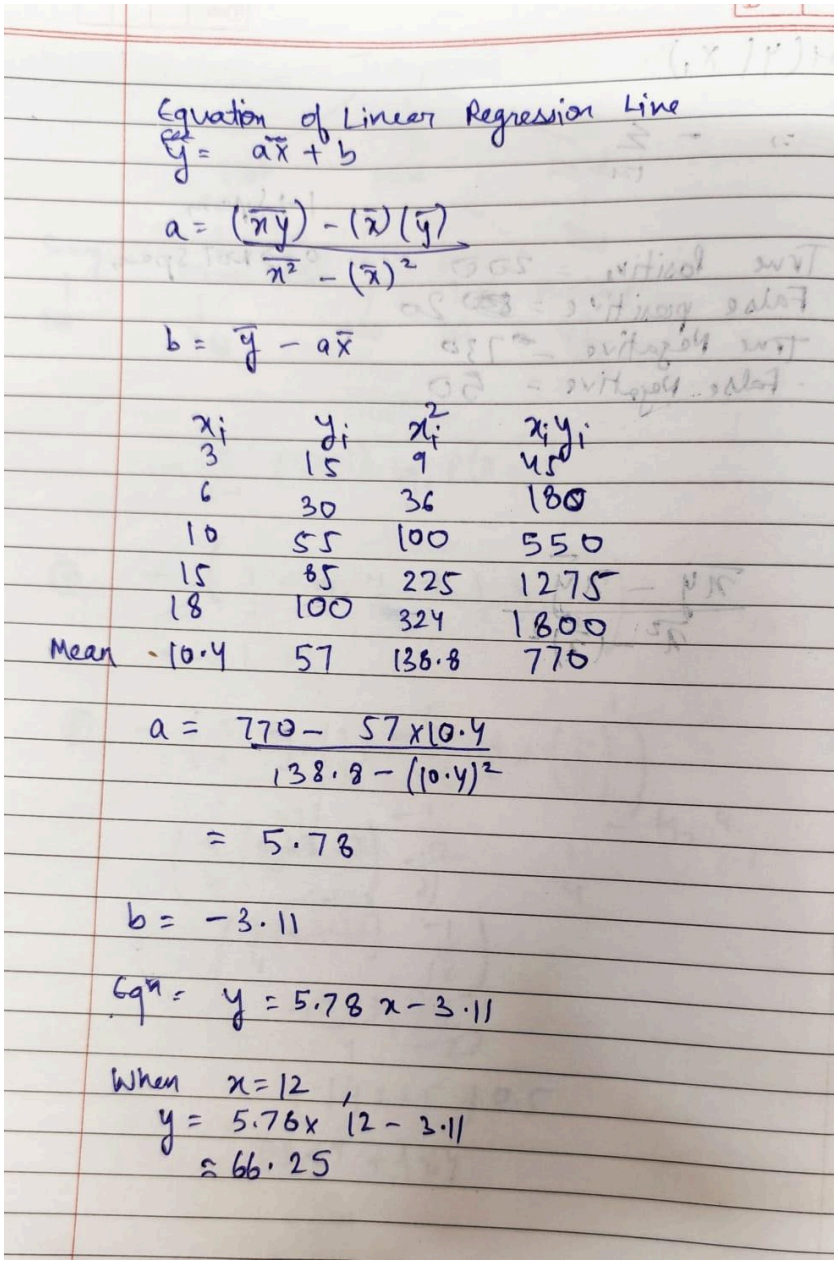


b. We define classes such that label 1 means it is a spam email and 0 means it is a legitimate email. Hence, we get the following classification report:

- True Positive: 200
- False Positive: 20
- True Negative: 730
- False Negative: 50
- Precision: (TP/TP + FP) = 0.909
- Recall: (TP/TP+FN) = 0.8
- Accuracy: (TP + TN/ TP + TN + FP + FN) = 0.93
- F1-Score: (2 * Precision * Recall/Precision + Recall) = 0.851

The overall classification performance is strong, considering the high precision and accuracy.

c. Answer for part c



The equation of the regression line is y = 5.78x - 3.11.
The value of y when x = 12 is 66.25.

d. Let us take an example training dataset.

X	Expected Y
1	2
2	4

3	7
4	9
5	10

We will use the MSE to calculate loss in this case.

Model f1
Suppose f1 trains on the data and fits the data nearly perfectly.
Let the regression equation be $y = 0.0833x^4 - 1.1667x^3 + 5.4167x^2 - 7.3333x + 5$

The prediction for the training dataset comes out to be the following:

X	**Predicted Y**
1	2
2	3.99
3	6.99
4	8.99
5	9.98

We calculate the empirical loss using MSE.
Loss = (2-2)^2 + (4 - 3.99)^2 + (7 - 6.99)^2 + (9 - 8.99)^2 + (10 - 9.98)^2
Loss = 0.0007

Model f2
Let f2 be a simpler model.
Let the regression equation be y = 2x

X	**Predicted Y**
1	2
2	4

3	6
4	8
5	10

Loss = (2 - 2)^2 + (4 - 4)^2 + (7 - 6)^2 + (9 - 8)^2 + (10 - 10)^2
Loss = 2

Hence, we notice that the empirical loss is significantly lower in the case of f1 than in f2.

Let's take a test dataset.

X	**Expected Y**
10	20
20	41

Model f1

X	**Predicted Y**
10	139.64
20	6019.41

MSE Loss = (139.64 - 20)^2 + (6019.41 - 41)^2
Loss = 35755699.9

Model f2

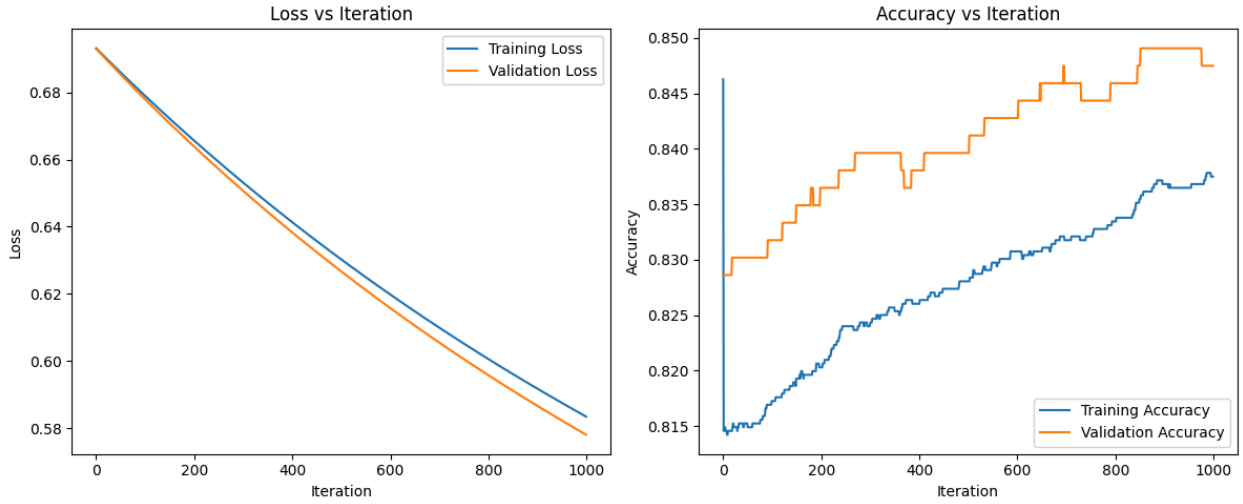
X	**Predicted Y**
10	20
20	40

MSE Loss = (20 - 20)^2 + (40 - 41)^2
Loss = 1

Hence, we notice that the testing loss is significantly lower for f2. This means that the model f1 is not generalizing well on the testing dataset and has overfitted.
So, in this example, we see that even though model f1 has significantly lower empirical risk than model f2 on the training dataset, it does not generalize better than f2 on testing dataset.

SECTION B

a. Batch Gradient Descent

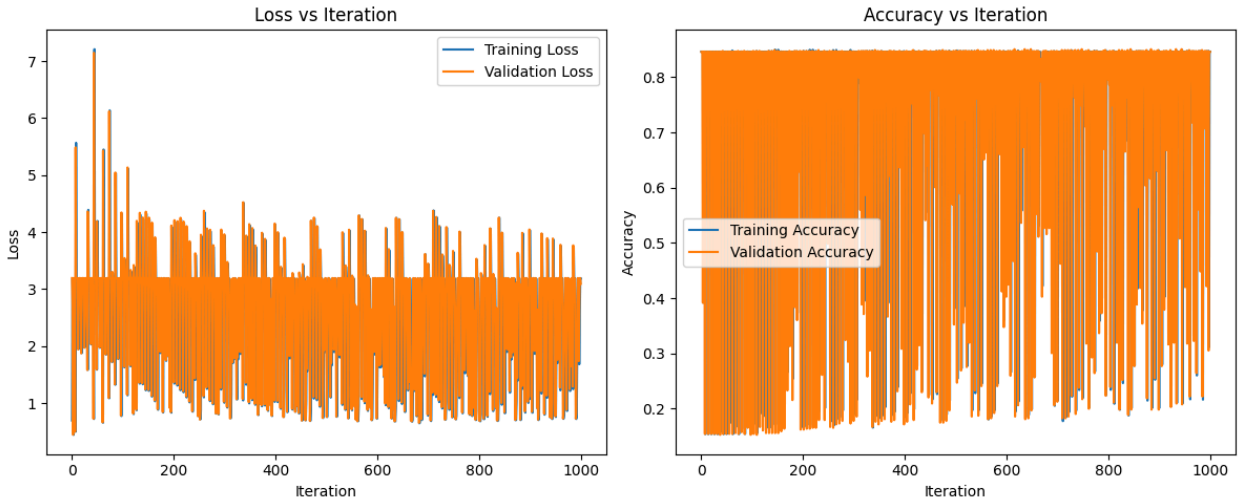


Training vs validation loss
We notice that the training and validation loss decreases steadily over time, which indicates that the model is generalizing well on unseen data. The parallel trend of both losses decreasing indicates that the model is converging appropriately without overfitting, as the validation loss is not increasing after a certain point.

Training vs validation accuracy
The training and validation accuracy increases, indicating that the model improves performance over time.

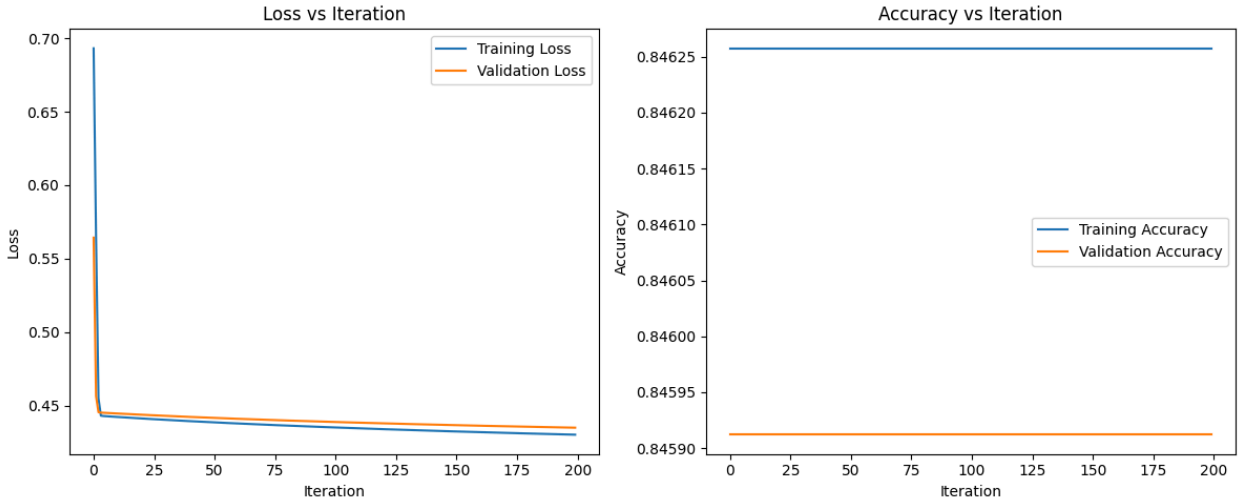
This suggests that the model is converging well over time and is not showing signs of underfitting or overfitting.

b. No Scaling

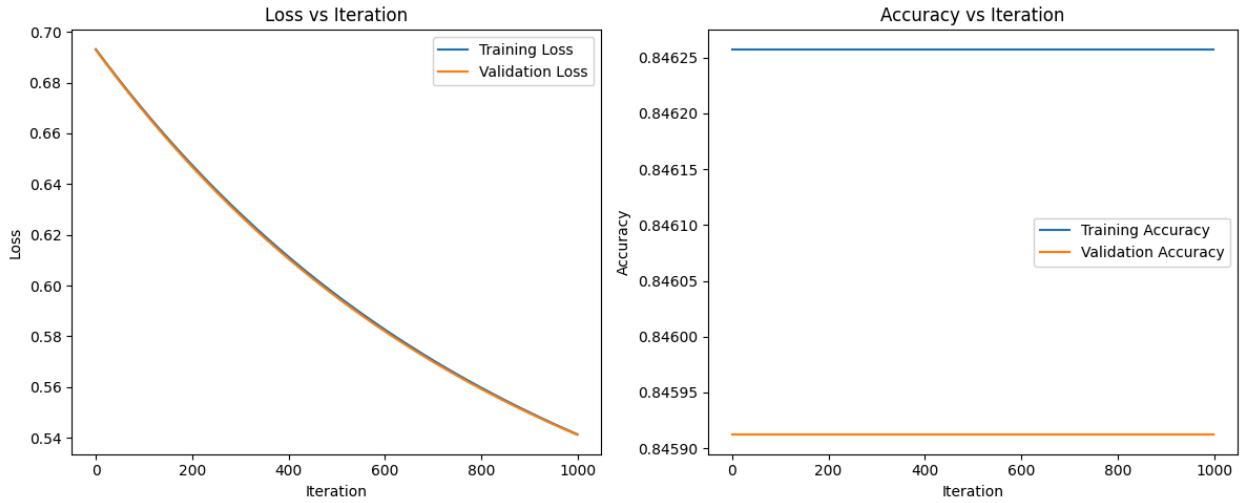


We notice heavy fluctuations in our loss and accuracy curves when using data with no scaling. This suggests that the model is not converging correctly and is struggling to settle into a consistent learning pattern.

On decreasing the learning rate and number of epochs, the fluctuation decreases and accuracy comes out to be constant.



Min Max Scaling

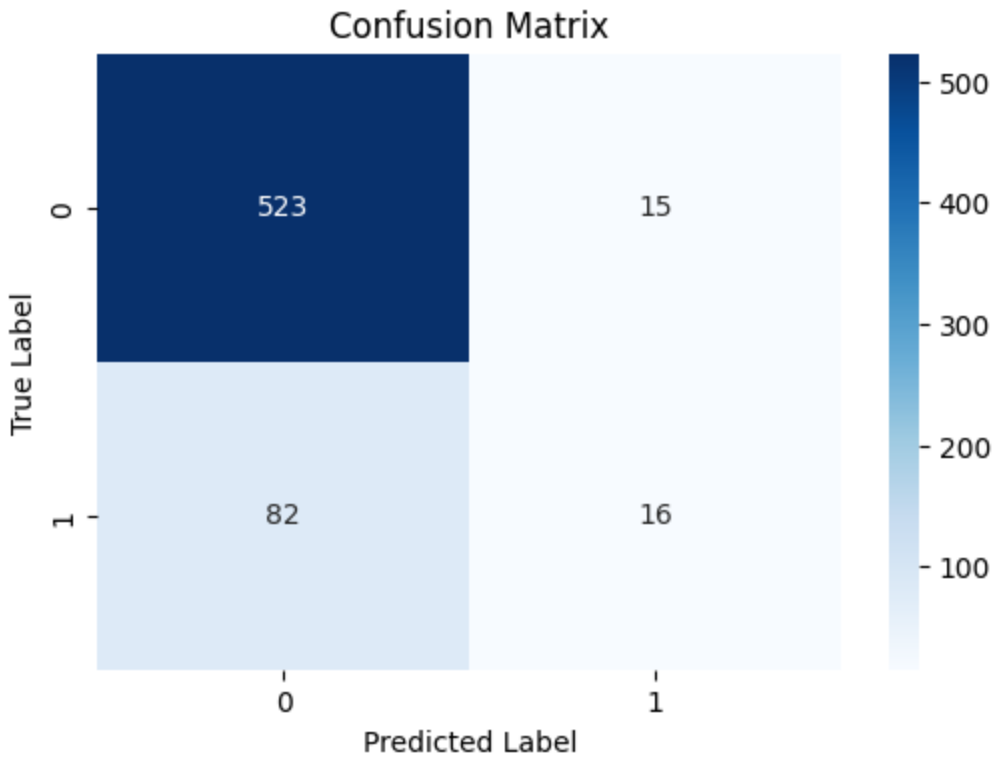


With min-max scaling, training and validation loss decrease steadily over time, indicating that the model is generalizing well over unseen data. The accuracy of training and validation stays constant over time. Overall, the model converges well and does not show indications of overfitting and underfitting.

Scaling the features has improved the loss and accuracy trend. Feature scaling ensures that all features are within the same range, leading to smoother convergence of the model and preventing any single feature from disproportionately influencing the gradient updates.

c. Standardised data

Confusion Matrix

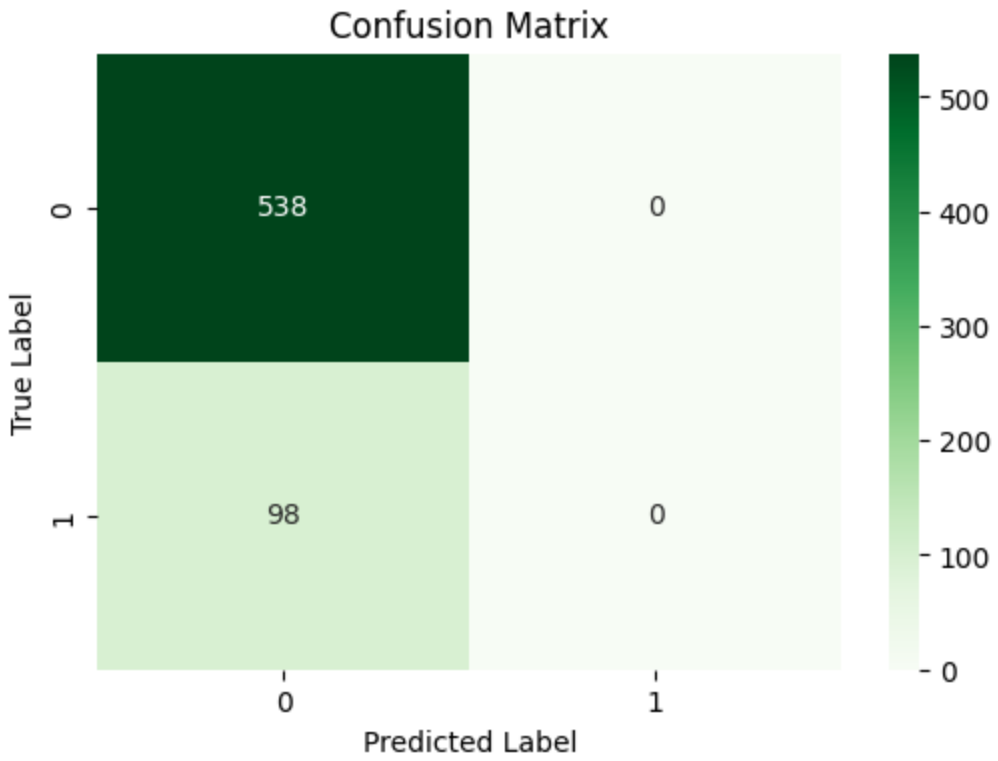


Evaluation Metrics

```
get_evaluation_metrics(tp, tn, fp, fn, y_val, y_val_pred)

Precision: 0.5161290322580645
Recall: 0.16326530612244897
F1-Score: 0.24806201550387597
ROC AUC Score: 0.5676921326151279
```

Unscaled data

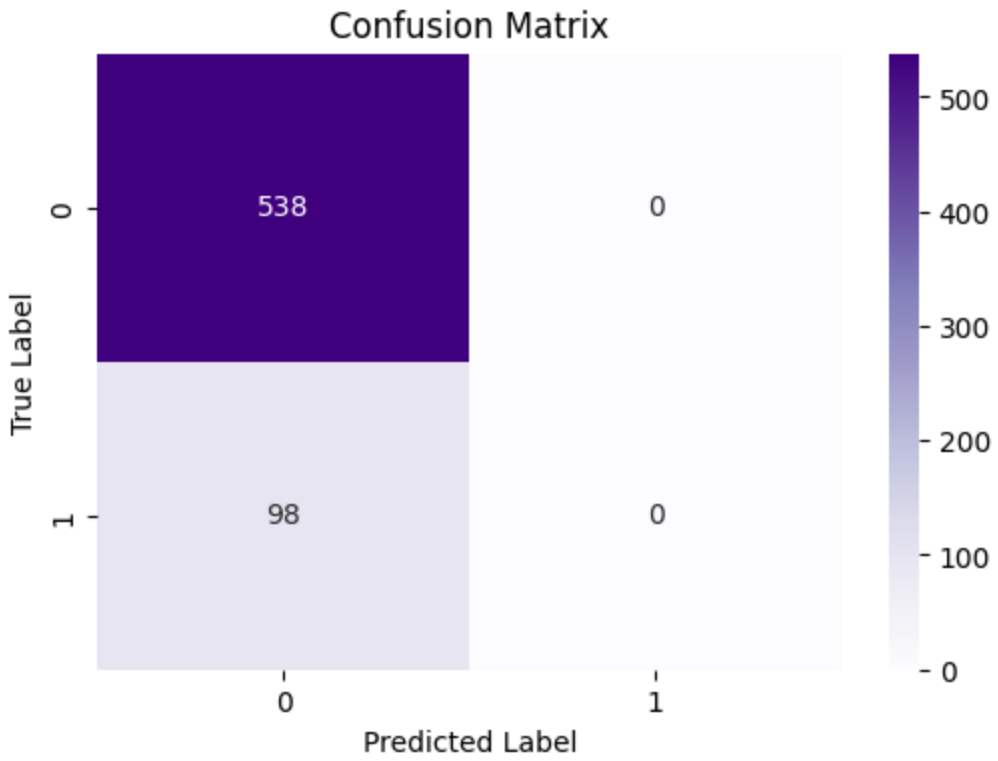
Confusion Matrix



Evaluation Metrics

```
[ ]  get_evaluation_metrics(tp, tn, fp, fn, y_val, y_val_pred)

     Precision: 0.0
     Recall: 0.0
     F1-Score: 0.0
     ROC AUC Score: 0.5
```

Min Max Scaled dataset

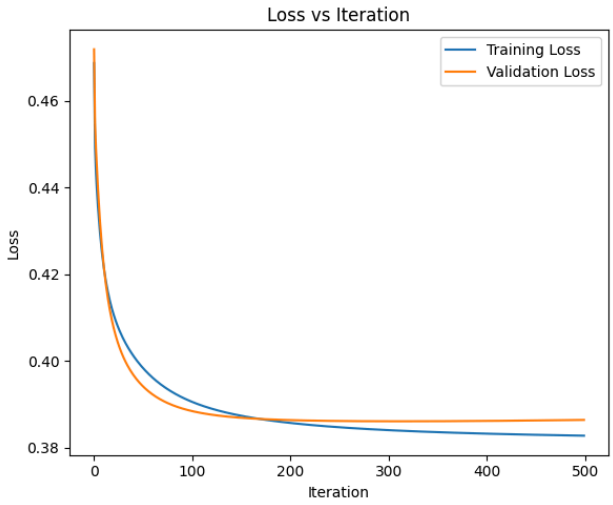


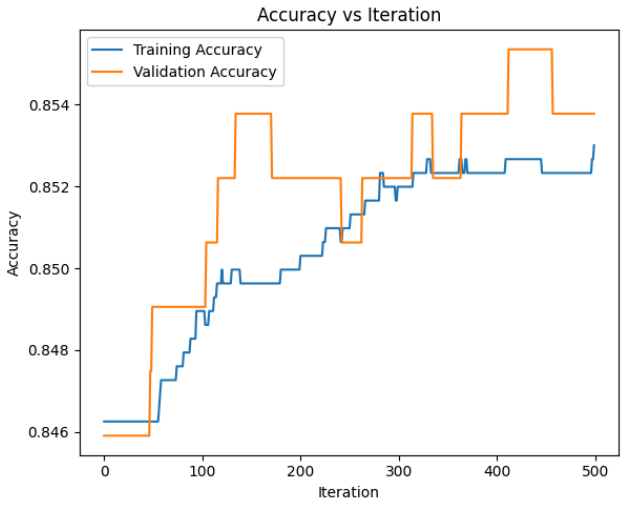
Evaluation Metrics

```
[ ] get_evaluation_metrics(tp, tn, fp, fn, y_val, y_val_pred)

    Precision: 0.0
    Recall: 0.0
    F1-Score: 0.0
    ROC AUC Score: 0.5
```

- **Precision** provides insights into the reliability of positive predictions out of all positive predictions. High precision indicates higher prediction accuracy for positive values.
- **Recall** provides insights into the actual percentage of positive predictions out of expected positive values.
- **F1 score** provides a balanced metric that considers both precision and recall, especially to balance the trade-off.
- **ROC AUC score** provides an overall picture of how well the model can differentiate between classes across different thresholds.

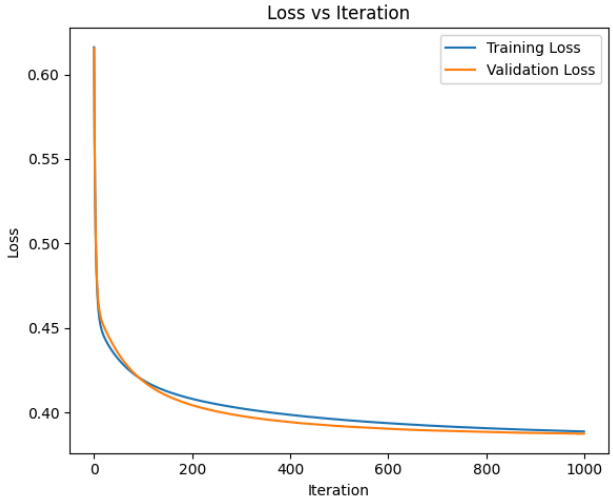
d. Stochastic Gradient Descent

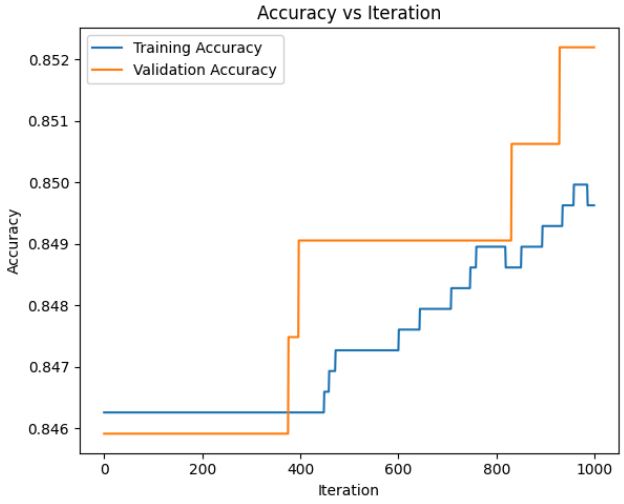




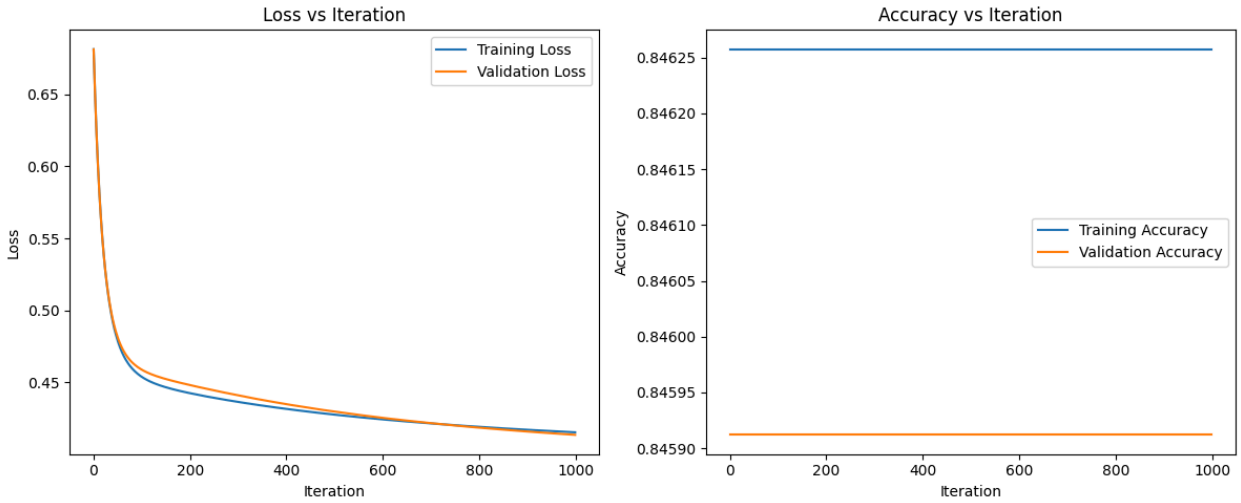
Mini Batch Gradient Descent

Batch size = 8





Batch size = 64



We notice that the convergence of the Stochastic Gradient Descent model is faster as the training loss and validation loss decrease faster than the Mini Batch but there is more fluctuation in the training and validation accuracy for SGD indicating less stability as compared to the Mini Batch. Hence, we notice that as the convergence speed increases, the stability of training decreases.

Further, in the Mini Batch model, we notice that as we increase the batch size, the convergence slows down, the model training becomes more stable, and accuracy saturates quicker than when the batch size is smaller.

e. Evaluation Metrics for all folds

```
Fold 1
Accuracy: 0.8394332939787486
Precision: 0.6875
Recall: 0.07746478873239436
F1-Score: 0.13924050632911392

Fold 2
Accuracy: 0.8630460448642266
Precision: 0.5
Recall: 0.08620689655172414
F1-Score: 0.14705882352941177

Fold 3
Accuracy: 0.8406139315230224
Precision: 0.43902439024390244
Recall: 0.13846153846153847
F1-Score: 0.2105263157894737

Fold 4
Accuracy: 0.8536009445100354
Precision: 0.6
Recall: 0.11627906976744186
F1-Score: 0.19480519480519481

Fold 5
Accuracy: 0.8406139315230224
Precision: 0.4
Recall: 0.12598425196850394
F1-Score: 0.19161676646706585
```

Mean and standard deviation for the evaluation metrics across all folds

```
K-Fold Cross-Validation Scores: [0.8394332939787486, 0.8630460448642266, 0.8406139315230224, 0.8536009445100354, 0.8406139315230224]
Mean Accuracy: 0.8474616292798112
Standard Deviation of Accuracy: 0.009368044592622119
Mean Precision: 0.5253048780487805
Standard Deviation of Precision: 0.10552022623157101
Mean Recall: 0.10887930909632056
Standard Deviation of Recall: 0.023338184175958496
Mean F1-Score: 0.176649521384052
Standard Deviation of F1-Score: 0.028200453573808345
```

We notice that the model's accuracy lies within a narrow range of 83 to 86% with a standard deviation of 0.009 only, indicating low variance in the model's accuracy within different folds.
However, we notice that the standard deviation of Precision is relatively higher than other metrics, indicating that the model's ability to correctly predict positive cases is inconsistent across folds.
F1-Score and Recall have comparable standard deviations, which are higher than the standard deviation of Accuracy.
Hence, in k-fold cross-validation, accuracy is a stable metric, while precision, recall, and f1-score show high variance.

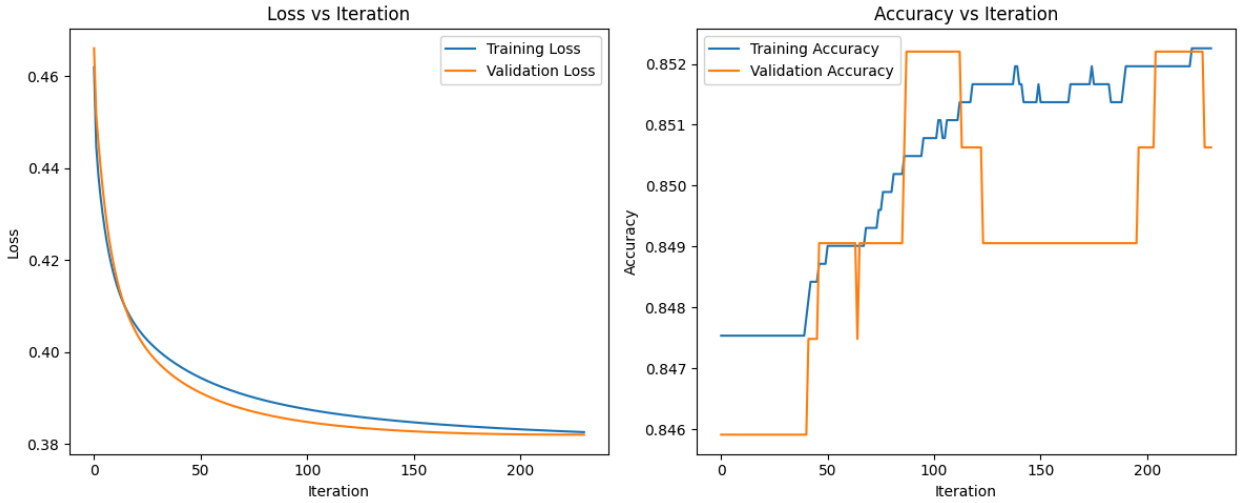
f. We implement early stopping on our stochastic gradient descent model as it is our best-performing model.

Criteria Used:

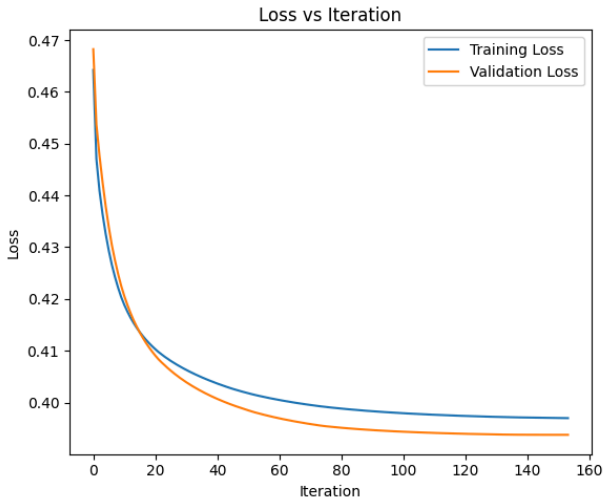
- We set a patience level. Our default patience level is 5.
- We track our best validation loss. If our validation loss does not improve over epochs equal to patience level, we stop the training of our model as this indicates that our model will not be performing better than this and further training could lead to overfitting.
- We set our weights and bias to the best weights and bias obtained during the best validation loss epoch.

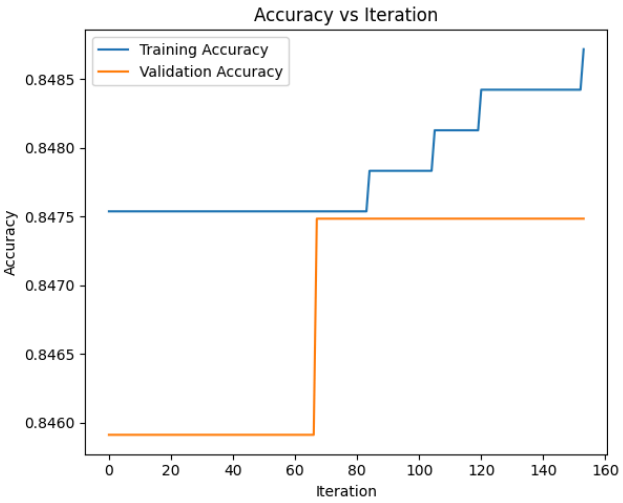
Early Stopping

L1 = 0, L2 = 0

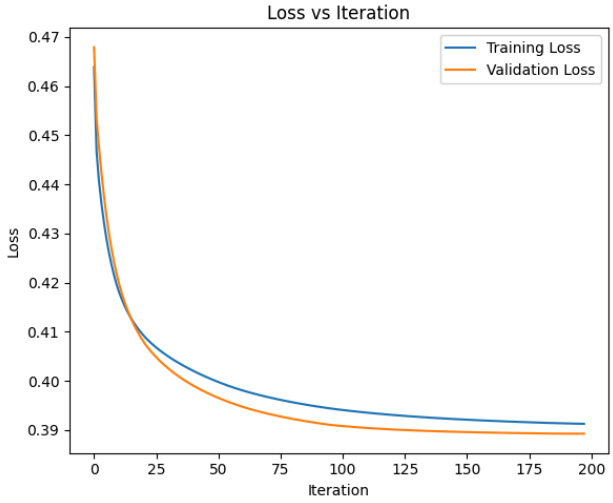


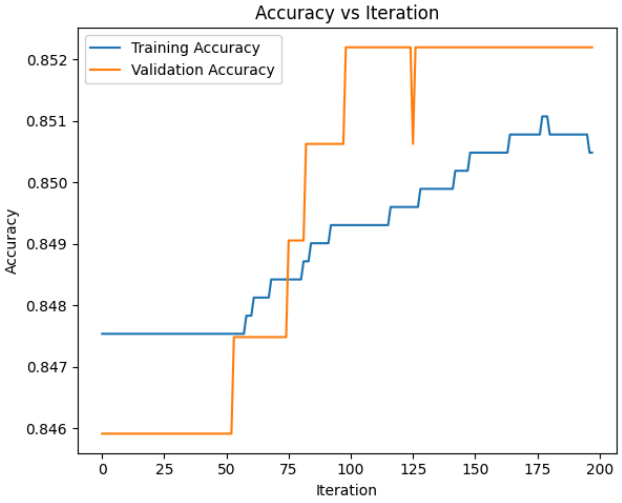
L1 = 0.001, L2 = 0.001



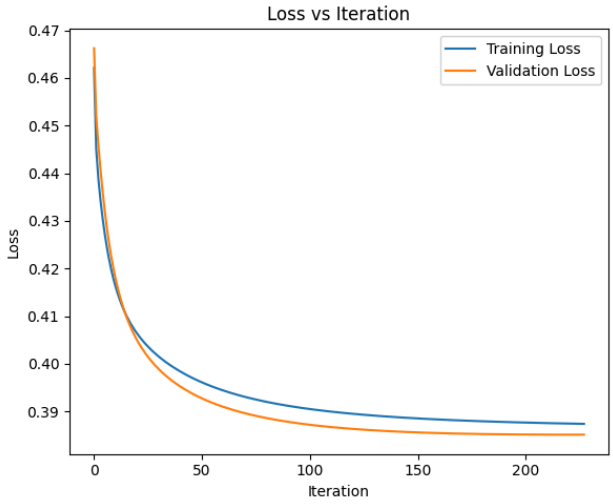


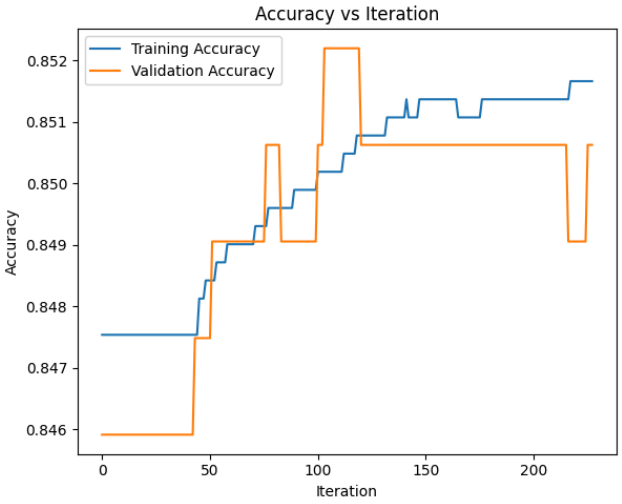
L1 = 0.001, L2 = 0





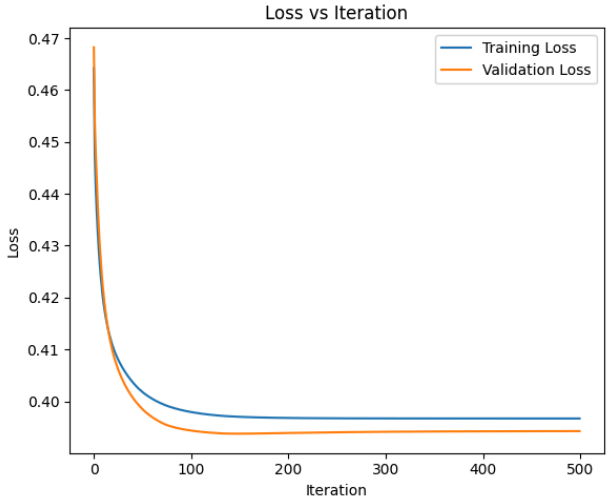
L1 = 0, L2 = 0.0005

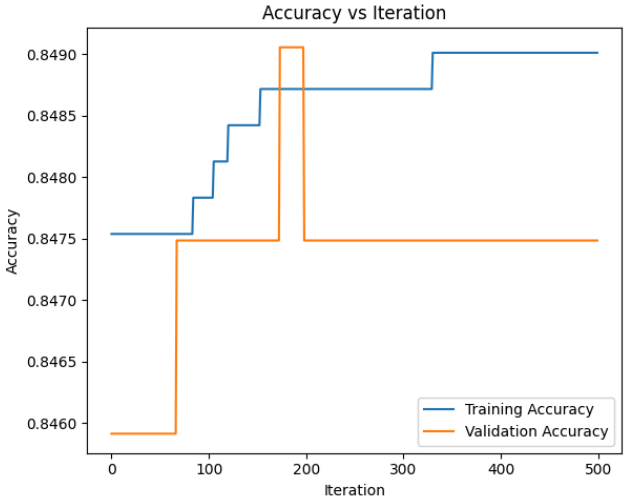




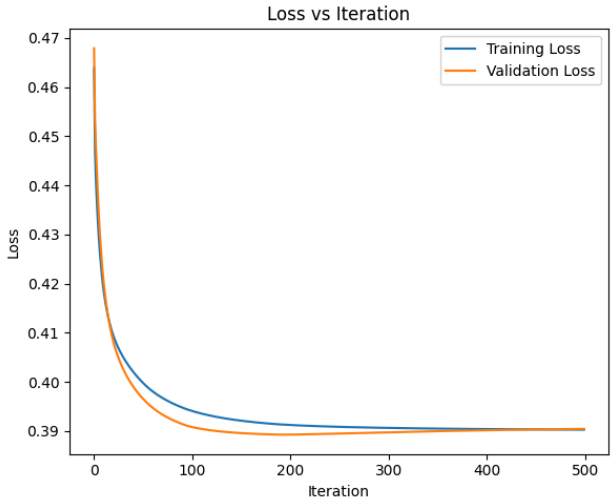
Without early stopping

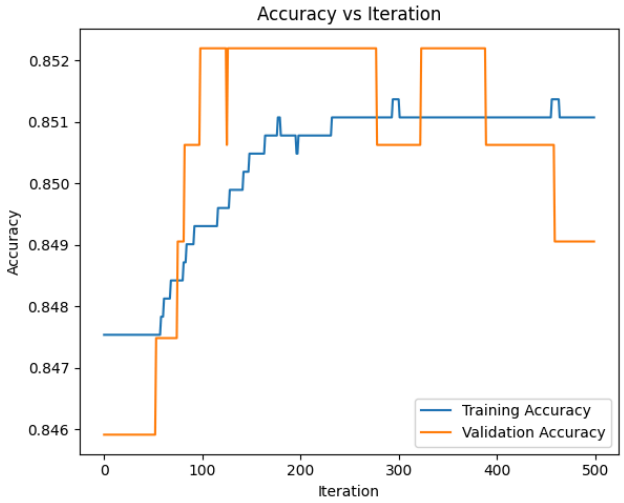
L1 = 0.001, L2 = 0.001



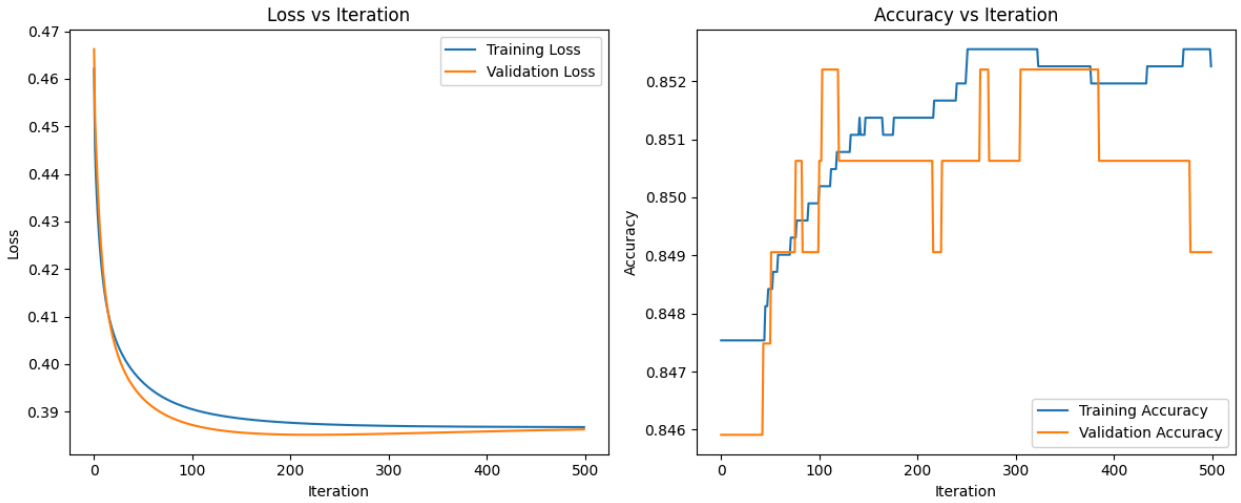


L1 = 0.001, L2 = 0





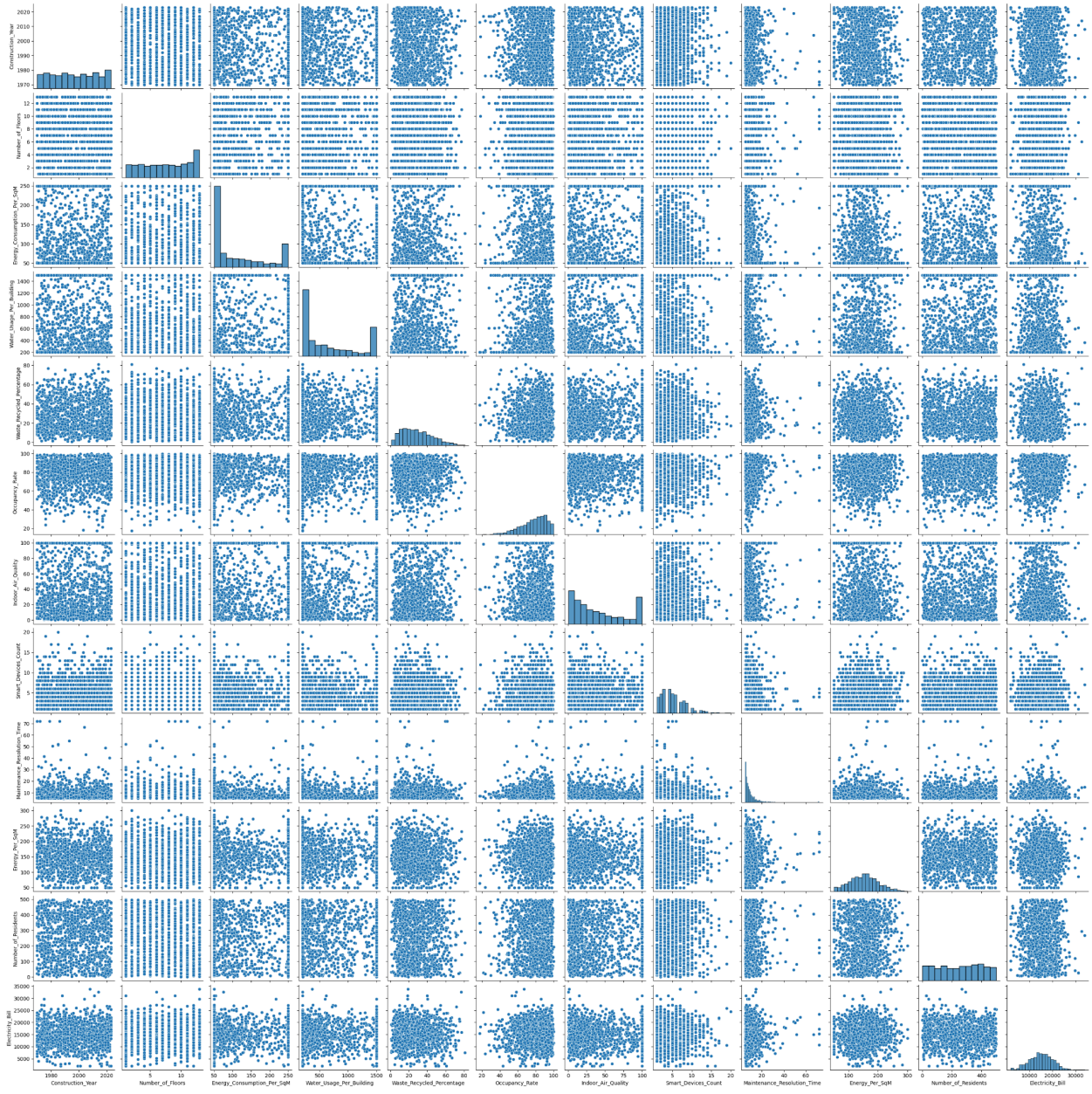
L1 = 0, L2 = 0.0005



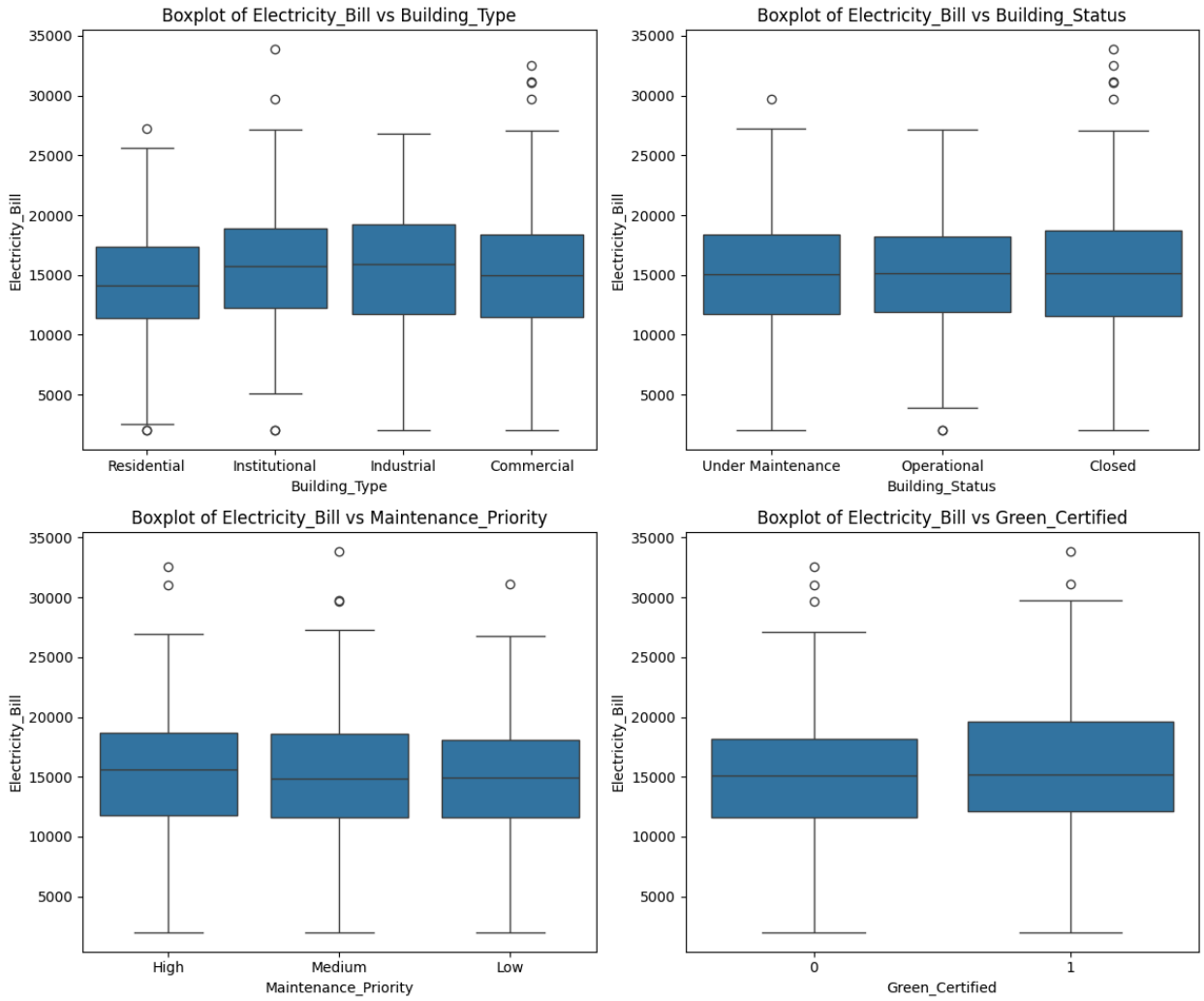
When the model starts to overfit, that is, the validation loss does not seem to be getting better, early stopping causes the model to stop the training process, hence preventing the model from overfitting. Therefore, this model now performs better on unseen data and generalizes better. While early stopping prevents overfitting, it also ensures that the model is trained sufficiently to avoid underfitting.

SECTION C

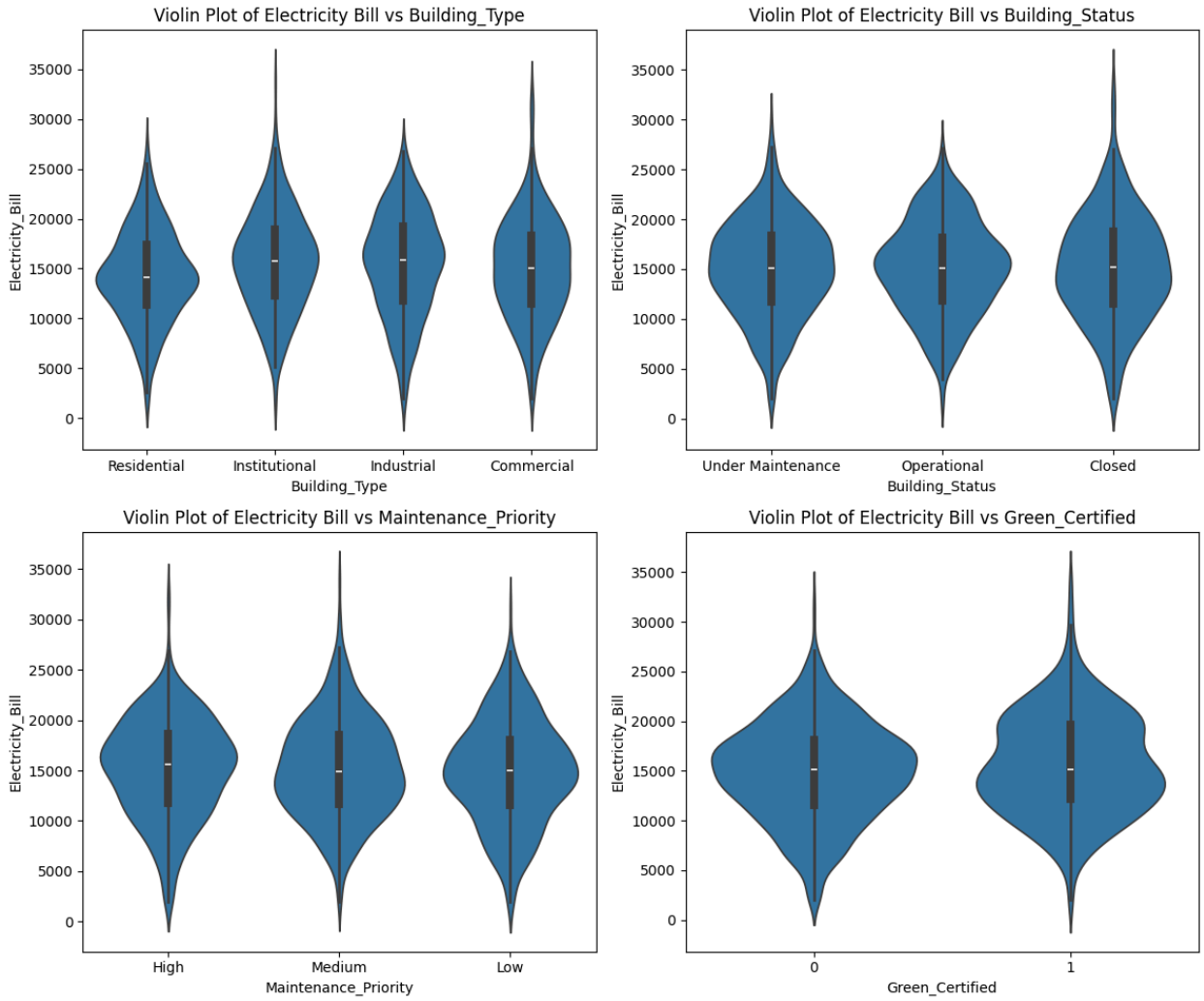
a. Pair Plot


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

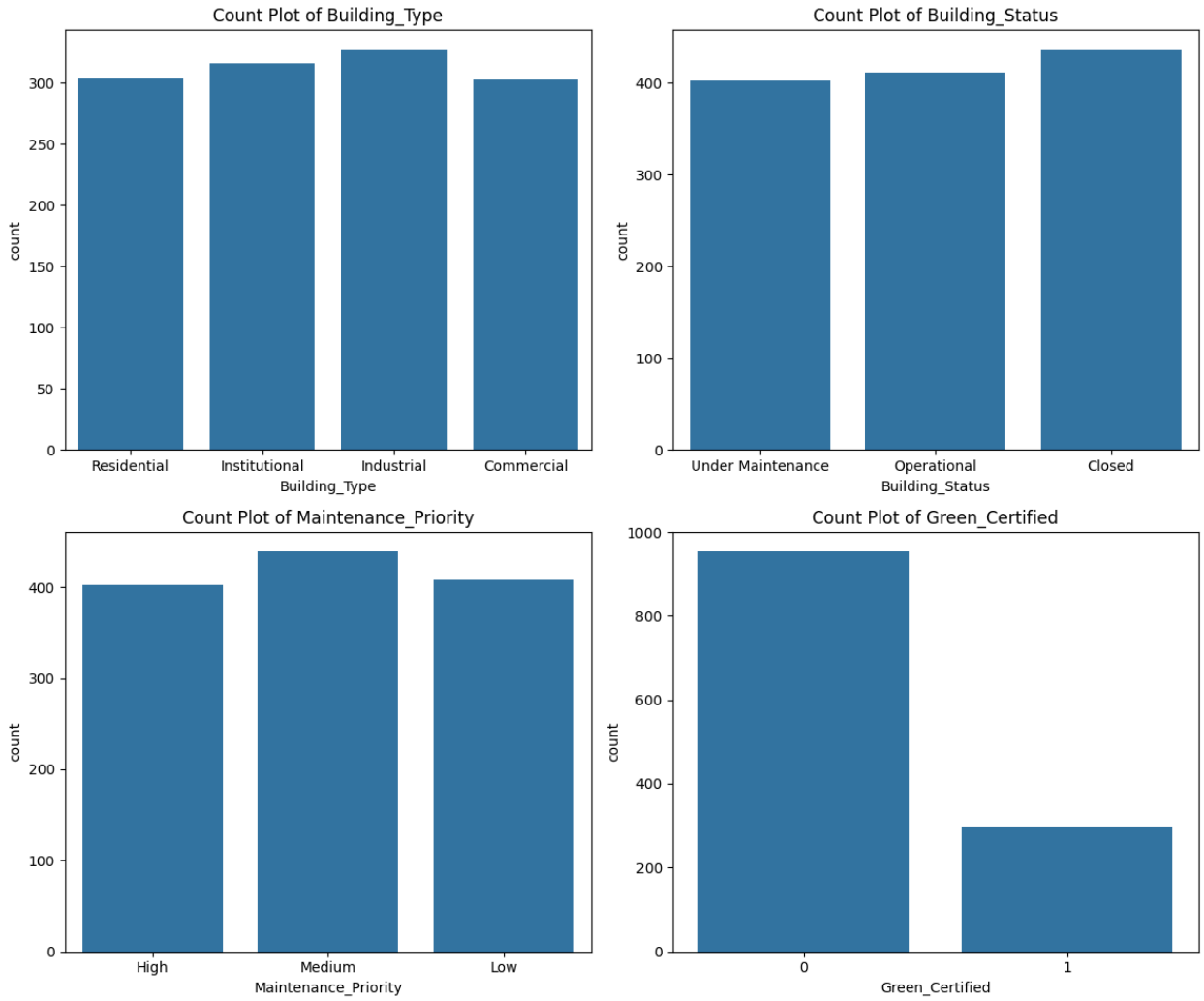
Box Plot


Boxplot of Electricity_Bill vs Building_Type
Electricity_Bill
Residential
Institutional
Industrial
Commercial
Building_Type
Boxplot of Electricity_Bill vs Building_Status
Under Maintenance
Operational
Closed
Building_Status
Boxplot of Electricity_Bill vs Maintenance_Priority
High
Medium
Low
Maintenance_Priority
Boxplot of Electricity_Bill vs Green_Certified
0
1
Green_Certified

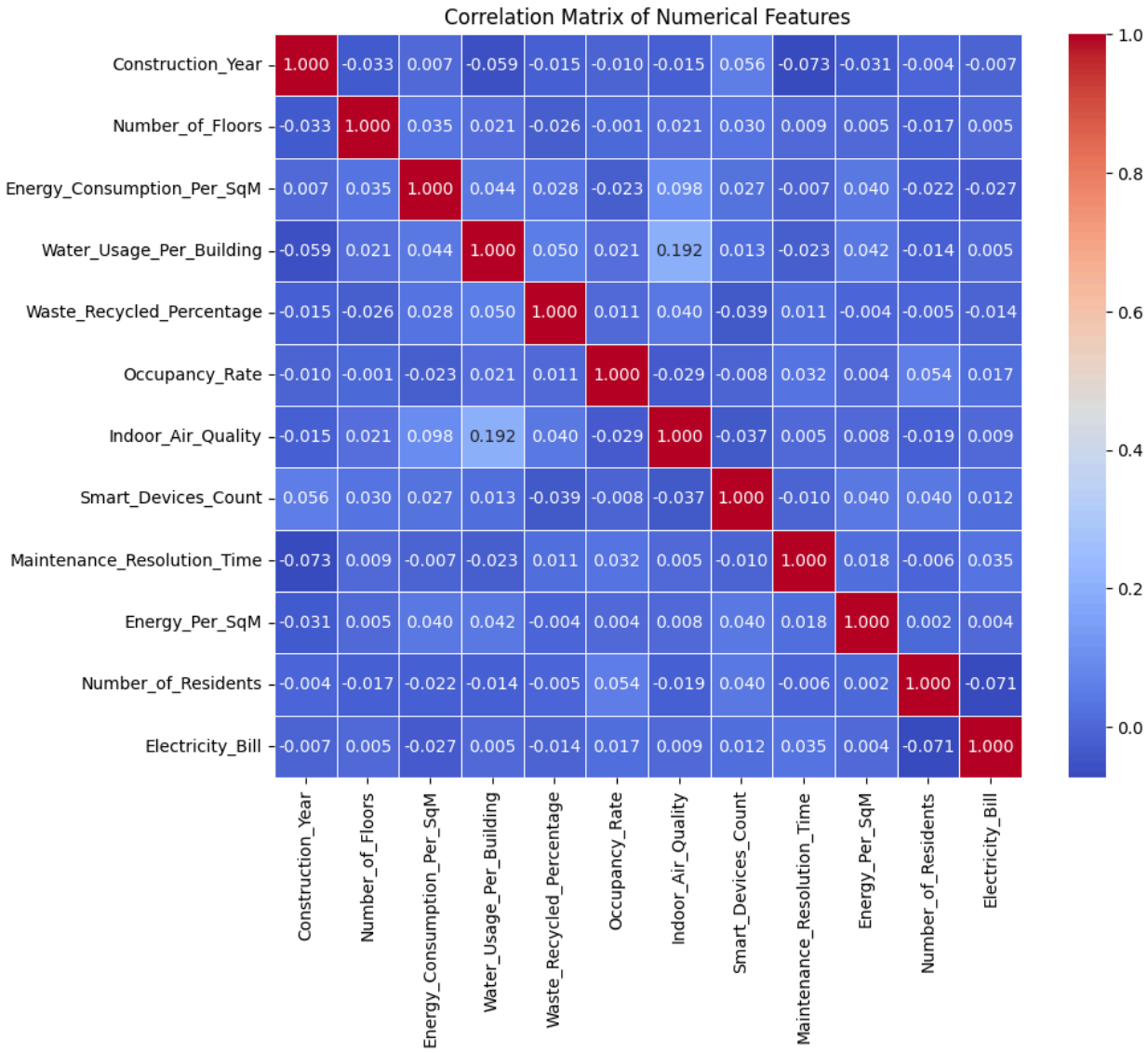
Violin Plot


Violin Plot of Electricity Bill vs Building_Type
Electricity_Bill
Residential
Institutional
Industrial
Commercial
Building_Type
Violin Plot of Electricity Bill vs Building_Status
Under Maintenance
Operational
Closed
Building_Status
Violin Plot of Electricity Bill vs Maintenance_Priority
High
Medium
Low
Maintenance_Priority
Violin Plot of Electricity Bill vs Green_Certified
0
1
Green_Certified

Count Plot



Correlation Matrix

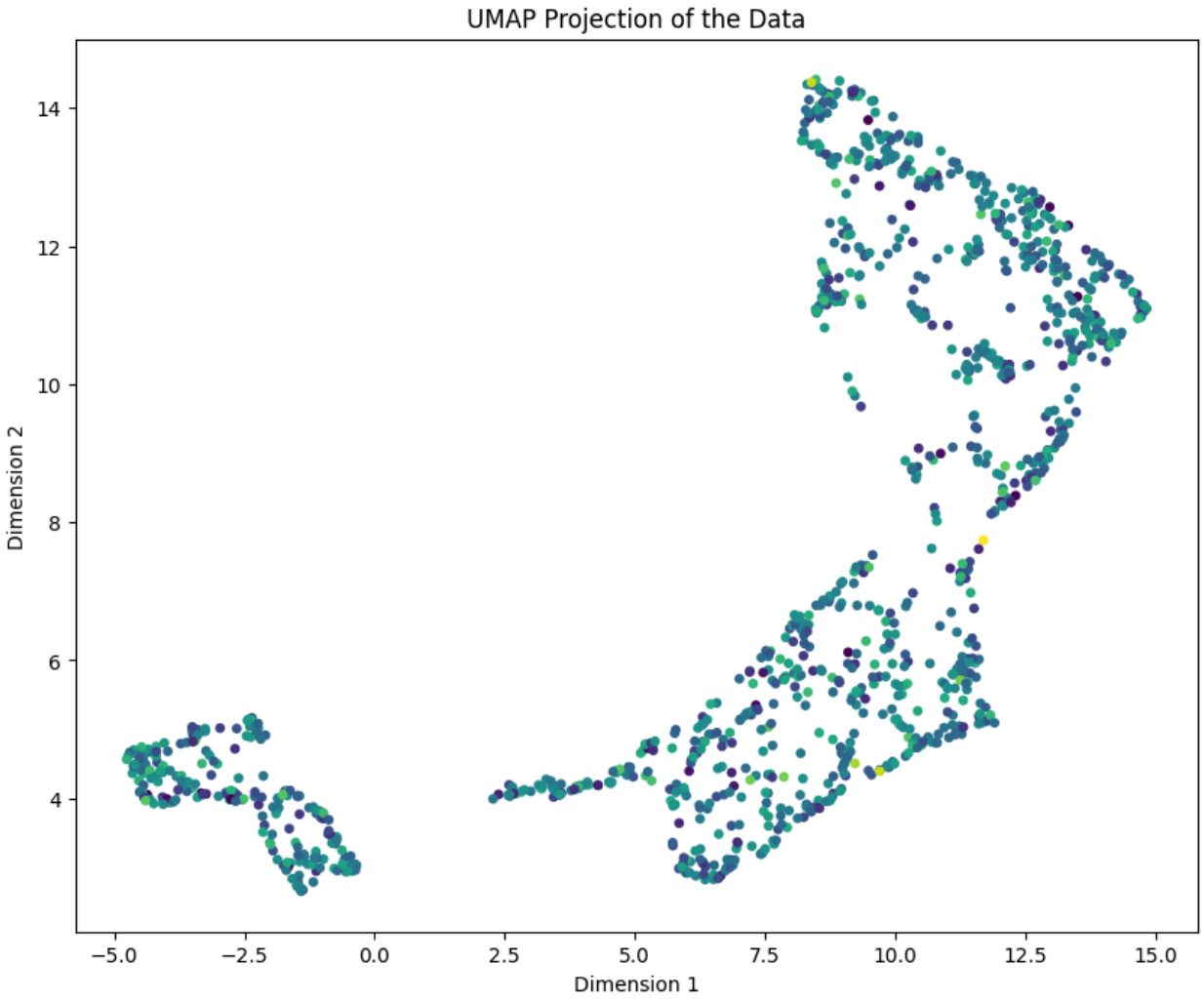


Based on these visualizations, we get the following insights about our dataset:

1. From our pair plots, we notice no strong correlation between the features as the points are scattered randomly. We notice horizontal and vertical alignments for the features "Number_of_Floors" and "Smart_Devices_Count." This indicates that these features are discrete categories. This represents a relationship between the discrete variable and the continuous variable. We noticed some of the points separated from the main cluster, indicating the presence of outliers in the data for these numerical features.
2. From the box plot, we notice outliers in our categorical features in their relationship with Electricity_Bill. We also see a higher range of electricity bills for buildings that are Green-Certified or Closed or of Institutional Type. The range for all bills based on maintenance priority is similar.

3. Our violin plot shows that the electricity bill is concentrated around the median for residential buildings, high-maintenance priority buildings, and buildings that are not green-certified. For other categories, the probability distribution is comparable.
4. From our count plots, we notice that the count of different categories in Building_Type, Building_status, and Maintenance_Priority are comparable, while more buildings are not green-certified compared to green-certified buildings.
5. From the diagonal of the pair plot, we notice that the Energy_Per_SqM and Electricity_Bill follow a normal distribution while others either follow a skewed distribution or have similar values.

b.



From our UMAP plot, we notice two distinct clusters, indicating that the data has two significant groupings after dimensionality reduction. We also see some sub-clusters arising in the larger cluster, indicating the presence of sub-groups.

The smaller cluster is distinctly separated from the larger cluster, indicating that it is an entirely different group. Within the clusters, we notice that the sub-clusters are overlapping.

c. The following are the scores for train and test data for different evaluation metrics.

```
Train MSE: 24475013.16847547, Test MSE: 24278016.155742623
Train RMSE: 4947.222773281538, Test RMSE: 4927.272689403604
Train R2: 0.013922520844610209, Test R2: 3.7344733075372893e-05
Train Adjusted R2: -0.0011091480449536562, Test Adjusted R2: -0.0640628254763429
Train MAE: 4006.32846932936, Test MAE: 3842.4093125585155
```

d. We select features using both RFE and Correlation Analysis. We see that they give similar results.

RFE

We first analyse our results by selecting top 3 features through RFE.

```
lr_model = LinearRegression()

rfe = RFE(estimator=lr_model, n_features_to_select=3)
rfe.fit(X_train_le, y_train_le)

selected_features = X_le.columns[rfe.support_]
print(f"Selected Features: {selected_features}")

X_selected = X_le[selected_features]
```

```
Selected Features: Index(['Building_Type', 'Green_Certified', 'Number_of_Residents'], dtype='object')
```

```
[84] train_and_evaluate_selected(lr_model, X_selected, y_le)
```

```
Train MSE: 24569032.906897984, Test MSE: 23941409.062998377
Train RMSE: 4956.715939702212, Test RMSE: 4892.995918964002
Train R2: 0.010134545491284008, Test R2: 0.01390151386794114
Train Adjusted R2: 0.007153023037944517, Test Adjusted R2: 0.0018759225736477703
Train MAE: 4006.4733775147365, Test MAE: 3813.948128176773
```

Correlation Analysis

We now select top 3 features using correlation analysis.

```
correlation_matrix = data_le.corr()

correlation_with_target = correlation_matrix['Electricity_Bill'].abs().sort_values(ascending=False)

top_3_features_corr = correlation_with_target.index[1:4]
print(f"Selected Features: {top_3_features_corr}")

X_selected = X_le[top_3_features_corr]
```

```
Selected Features: Index(['Number_of_Residents', 'Green_Certified', 'Building_Type'], dtype='object')
```

```
train_and_evaluate_selected(lr_model, X_selected, y_le)
```

```
Train MSE: 24569032.90689799, Test MSE: 23941409.062998384
Train RMSE: 4956.715939702212, Test RMSE: 4892.995918964002
Train R2: 0.010134545491283897, Test R2: 0.013901513867940807
Train Adjusted R2: 0.007153023037944406, Test Adjusted R2: 0.0018759225736473262
Train MAE: 4006.473377514736, Test MAE: 3813.948128176773
```

By comparing the results of d with c, we notice that our model performs better during training for the model which is trained on all features, while our model performs better on test data when it is trained on the top 3 selected features. This gives the possibility of overfitting the model trained on all features and better generalizing the model trained on selected features.

e. The following are the scores for train and test data for different evaluation metrics.

```
Train MSE: 24188925.91329433, Test MSE: 24129688.934199765
Train RMSE: 4918.223857582565, Test RMSE: 4912.1979738402
Train R2: 0.02544873320926233, Test R2: 0.00614664465902004
Train Adjusted R2: 0.006554371914339829, Test Adjusted R2: -0.0759542846952348
Train MAE: 3976.6837323582586, Test MAE: 3797.462833733082
```

We notice that our model trained using Ridge Regression with one hot encoding performs better than the model trained using Linear Regression with label encoding. This is likely due to Ridge's regularization, which mitigates overfitting by shrinking less important feature weights.

f. The scores of different evaluation metrics for different number of components are as follows:

```
Components: 4
Train MSE: 24589773.9306, Test MSE: 24232749.7311
Train RMSE: 4958.8077, Test RMSE: 4922.6771
Train R2: 0.0093, Test R2: 0.0019
Train Adjusted R2: 0.0053, Test Adjusted R2: -0.0144
Train MAE: 3978.0974, Test MAE: 3802.1953


Components: 5
Train MSE: 24588480.2463, Test MSE: 24254297.9900
Train RMSE: 4958.6773, Test RMSE: 4924.8653
Train R2: 0.0094, Test R2: 0.0010
Train Adjusted R2: 0.0044, Test Adjusted R2: -0.0195
Train MAE: 3978.7462, Test MAE: 3804.2047


Components: 6
Train MSE: 24587094.9113, Test MSE: 24236046.1667
Train RMSE: 4958.5376, Test RMSE: 4923.0119
Train R2: 0.0094, Test R2: 0.0018
Train Adjusted R2: 0.0034, Test Adjusted R2: -0.0229
Train MAE: 3978.8786, Test MAE: 3802.7388


Components: 8
Train MSE: 24379674.0159, Test MSE: 24235407.5064
Train RMSE: 4937.5777, Test RMSE: 4922.9470
Train R2: 0.0178, Test R2: 0.0018
Train Adjusted R2: 0.0098, Test Adjusted R2: -0.0313
Train MAE: 3976.0493, Test MAE: 3800.5082
```

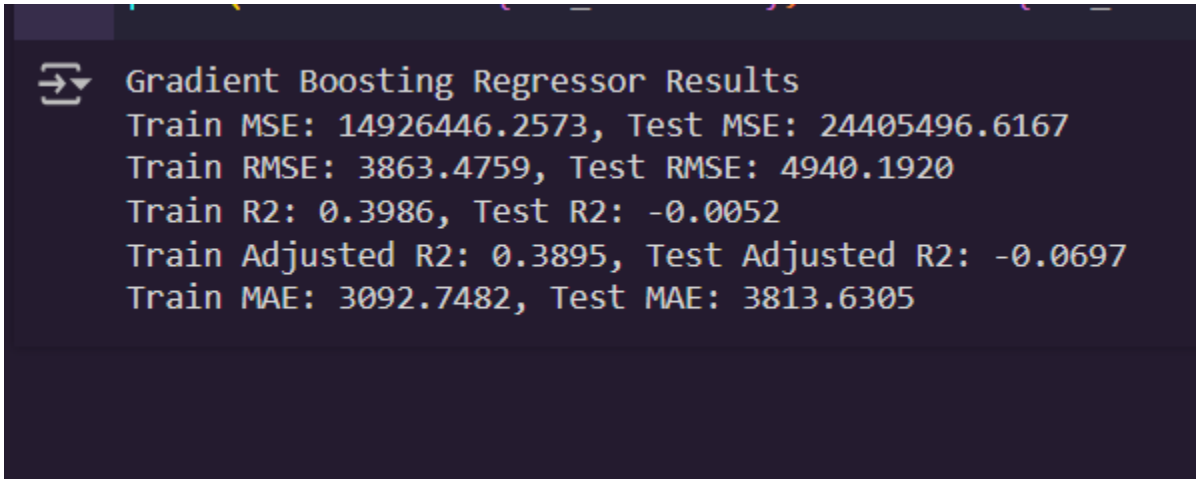
We notice that the model performs better for the training dataset as the number of components increases, but there is no noticeable trend between the performance of the model and the number of components for the training dataset. This might indicate that while more number of components capture the complexity of the data, it might not necessarily generalise well on unseen data.

g. The score for different values of alpha are as follows:

```
Alpha: 0.0001, L1 Ratio: 0.5
Test MSE: 24278004.4327
Test RMSE: 4927.2715
Test R²: 0.0000
Test Adjusted R²: -0.0641
Test MAE: 3842.4075


Alpha: 0.0005, L1 Ratio: 0.5
Test MSE: 24277957.5702
Test RMSE: 4927.2667
Test R²: 0.0000
Test Adjusted R²: -0.0641
Test MAE: 3842.4001


Alpha: 0.001, L1 Ratio: 0.5
Test MSE: 24277899.0563
Test RMSE: 4927.2608
Test R²: 0.0000
Test Adjusted R²: -0.0641
Test MAE: 3842.3908


Alpha: 0.01, L1 Ratio: 0.5
Test MSE: 24276857.8940
Test RMSE: 4927.1552
Test R²: 0.0001
Test Adjusted R²: -0.0640
Test MAE: 3842.2250
```

We notice that with a constant L1 ratio and increasing learning rate, the model performs better on the testing dataset, which means that it can generalize well on unseen data.

h. The scores for different evaluation metrics for the Gradient Boosting Regressor are as follows:



We notice that GBR model performs significantly better than the models used in c and g part on the training dataset but it performs slightly worse than both the models on the testing dataset.

REFERENCES

- https://medium.com/@koushikkushal95/logistic-regression-from-scratch-dfb8527a4226
- https://developer.ibm.com/articles/implementing-logistic-regression-from-scratch-in-python/
- https://www.datacamp.com/tutorial/normalization-in-machine-learning
- https://medium.com/@avijit.bhattacharjee1996/implementing-k-fold-cross-validation-from-scratch-in-python-ae413b41c80d
- https://www.greydongilmore.com/courses/ml_intro/02_05_plotting_categorical/
- https://www.geeksforgeeks.org/umap-uniform-manifold-approximation-and-projection/
- https://stackoverflow.com/questions/47272033/standardization-before-or-after-categorical-encoding
- https://stackoverflow.com/questions/51038820/how-to-calculated-the-adjusted-r2-value-using-scikit