CSE343: Machine Learning

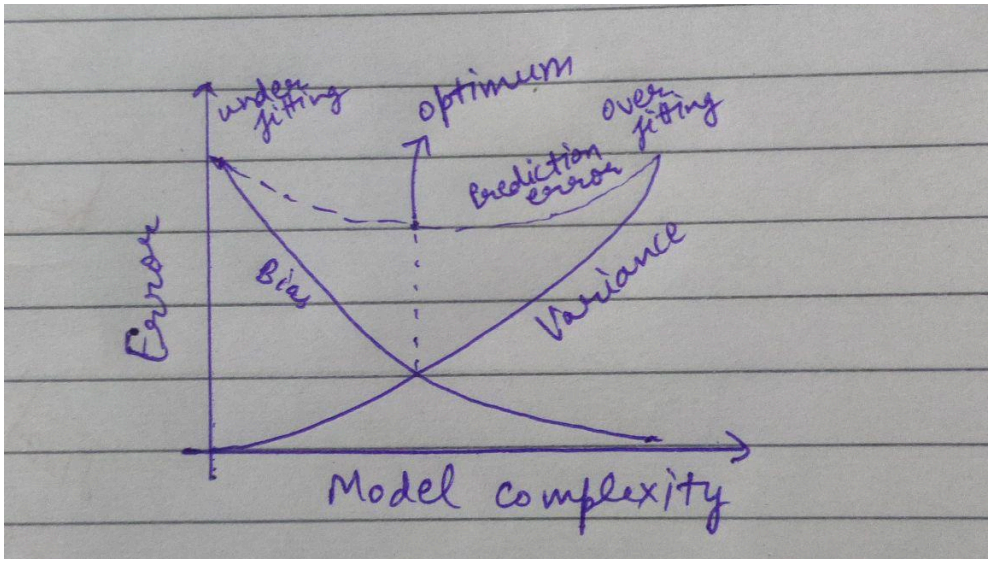
Assignment-1

Himanshu Raj (2022216)

September 13, 2024

Section A

a) As I increase the complexity of my model by adding more features or by including higher-order polynomial terms in a regression model, the model will begin to perform better on the training set. This means that the model is adapting and will eventually lead to overfitting. This means bias is low (we are not assuming data follows a linear pattern) and variance is high (the model performs too well on training data due to overfitting, and predictions on testing data will vary a lot).



b) The case where we identify a mail as spam is a positive instance. We can use F1 score to assess the model performance as it is a combination of both precision and recall.
TP = 200, FN = 50, TN = 730, FP = 20
Precision = TP / (TP + FP) = 200 / (200+20) = 0.909
Recall = TP / (TP + FN) = 200 / (200+50) = 0.8
F1 score = 2 * Precision * Recall / (Precision + Recall) = 2 * 0.909 * 0.8 / (0.909+0.8) = 0.851

c) Equation of regression line: $y_i = ax_i + b$, where $a = \frac{\overline{xy} - \bar{x}\bar{y}}{\overline{x^2} - \bar{x}\bar{x}}$ and $b = \bar{y} - a\bar{x}$.

	x	y	x^2	y^2
	3	15	9	45
	6	30	36	180
	10	55	100	550
	15	85	225	1275
	18	100	324	1800
sum	52	285	694	3850
mean	10.4	57	138.8	770

$a = \frac{770-(10.4)(57)}{138.8-10.4^2} = 5.78$ and $b = 57 - (5.78)(10.4) = -3.112$

For $x = 12$, $y_{12} = (5.78)(12) - 3.112 = 66.248$

d) Example:
X = [11, 15, 19, 23, 27, 31, 35, 39, 43, 47]
Y = [9, 18, 16, 26, 24, 29, 32, 43, 47, 44]
Loss function = Mean Squared Error

f1: A complex model that fits the training set very well, i.e. passes through all the points, as it assumes the data distribution is some high-degree polynomial function. Therefore empirical risk for f1 is 0.
f2: A model that doesn't fit the training set very well as it assumes the data distribution is some simple linear polynomial. The empirical risk is some positive real number, greater than 0.

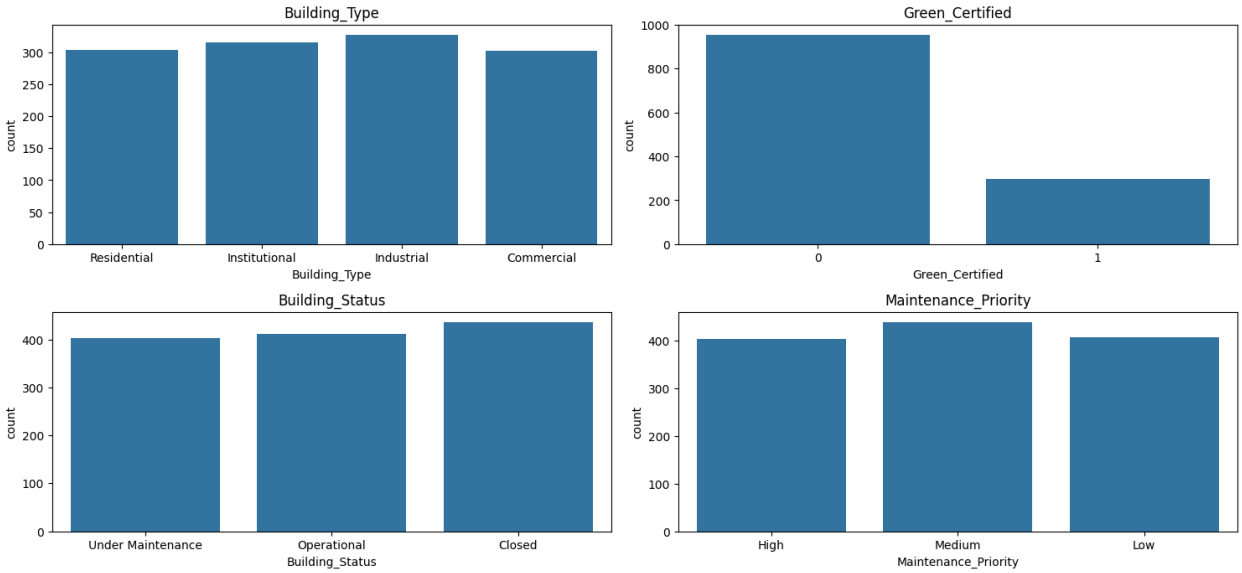
We can see that f1 has overfitted the training data and will not generalise over the unseen data, while f2 will generalise better.

Section C

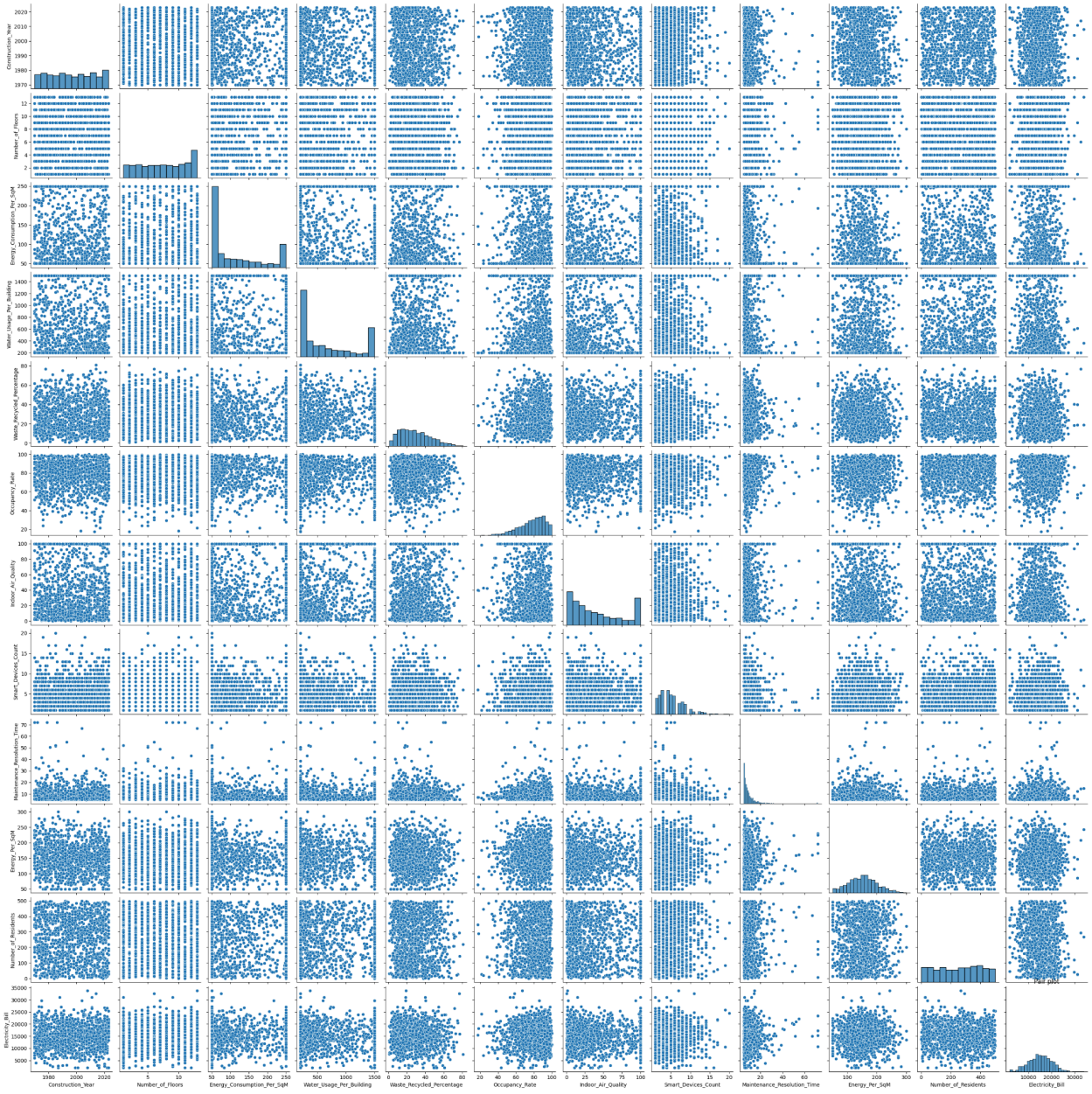
The problem is about implementing various Regression Algorithms using the 'scikit-learn' package. The data analysis and model training can be found in the notebook. The Electricity Bill dataset has no missing value; and has 4 categorical features and 12 non-categorical features.

a) Extensive Data Analysis

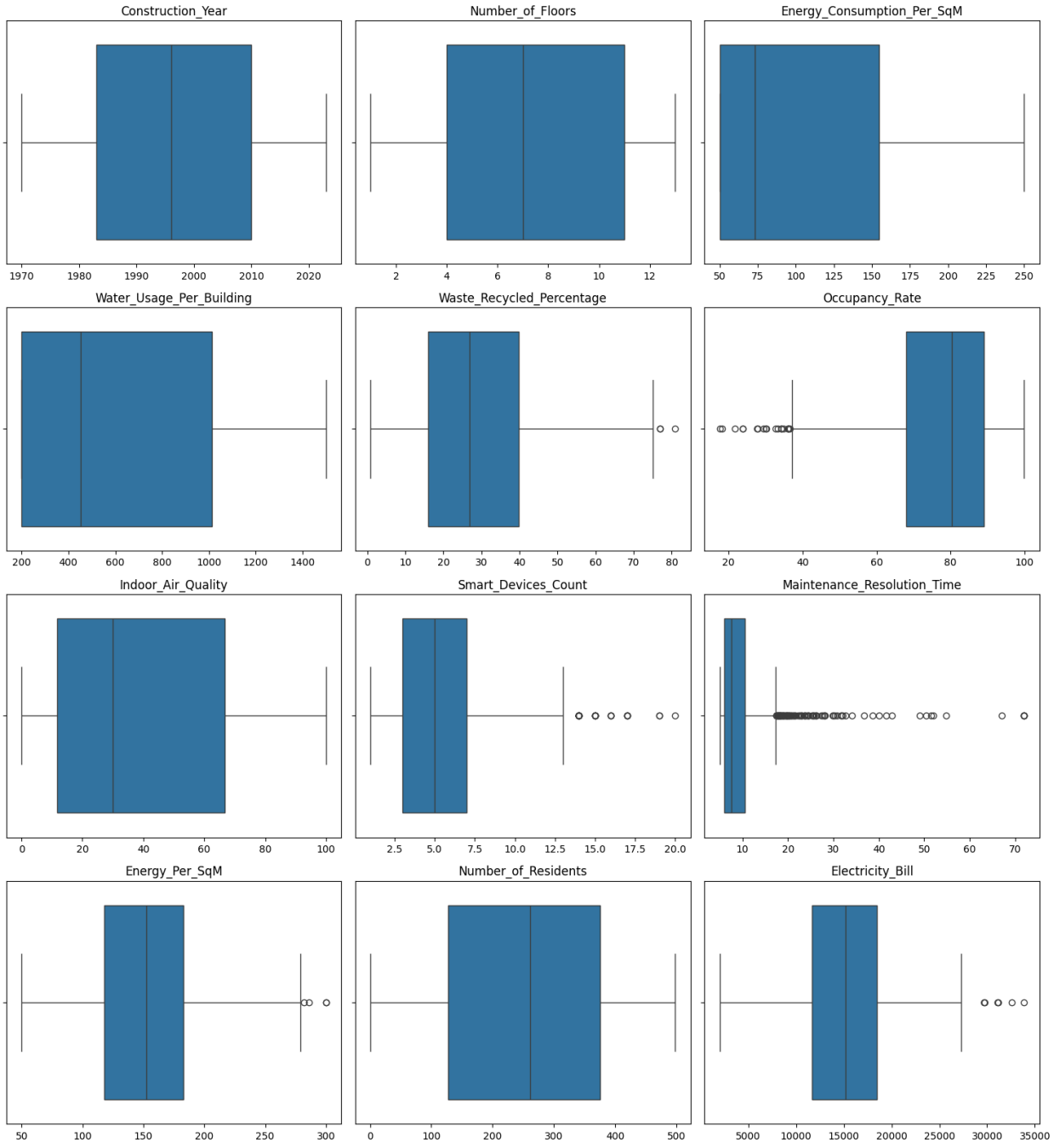
1. Count plots for categorical data:



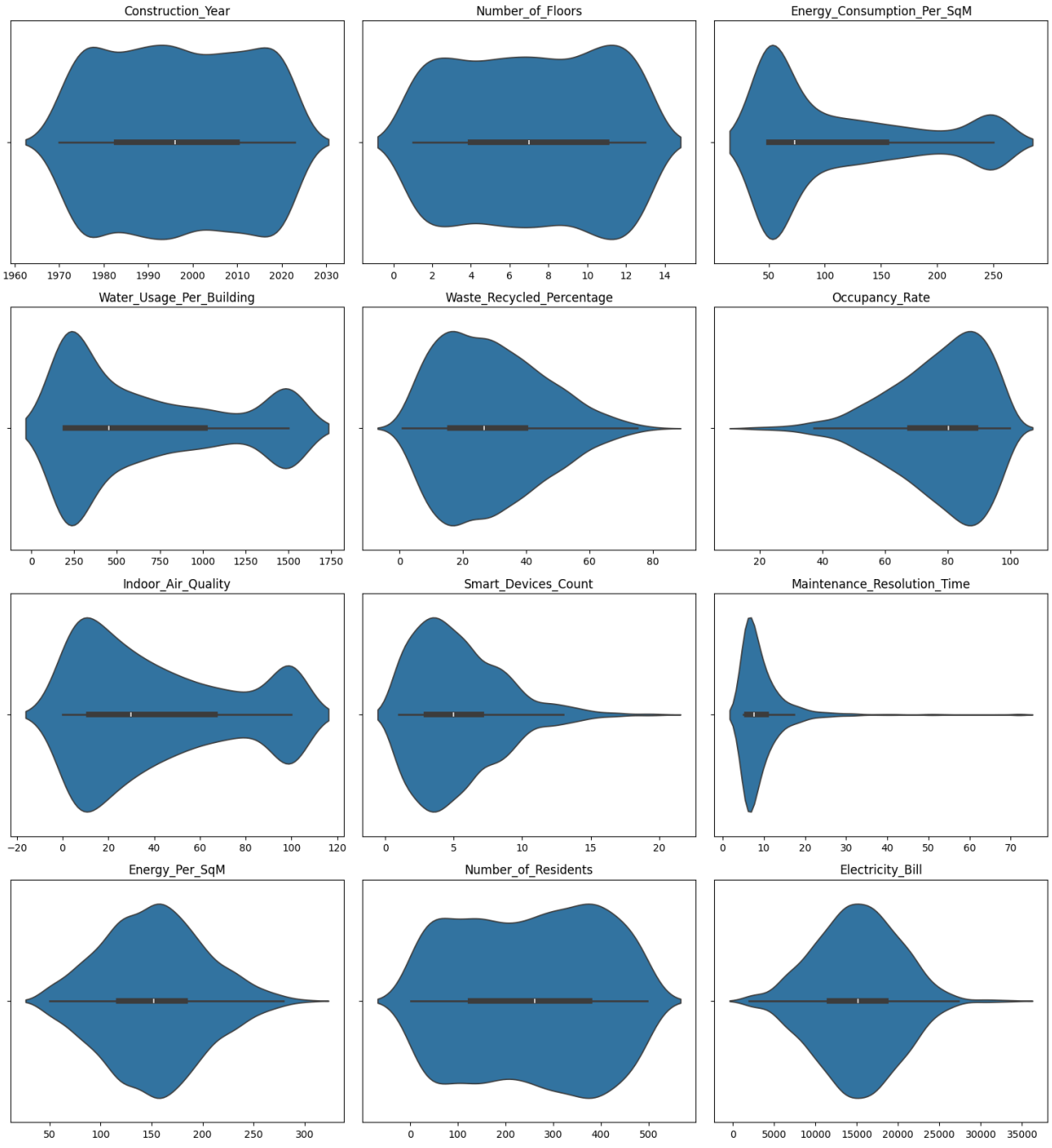
2. Pair plots for non-categorical data:



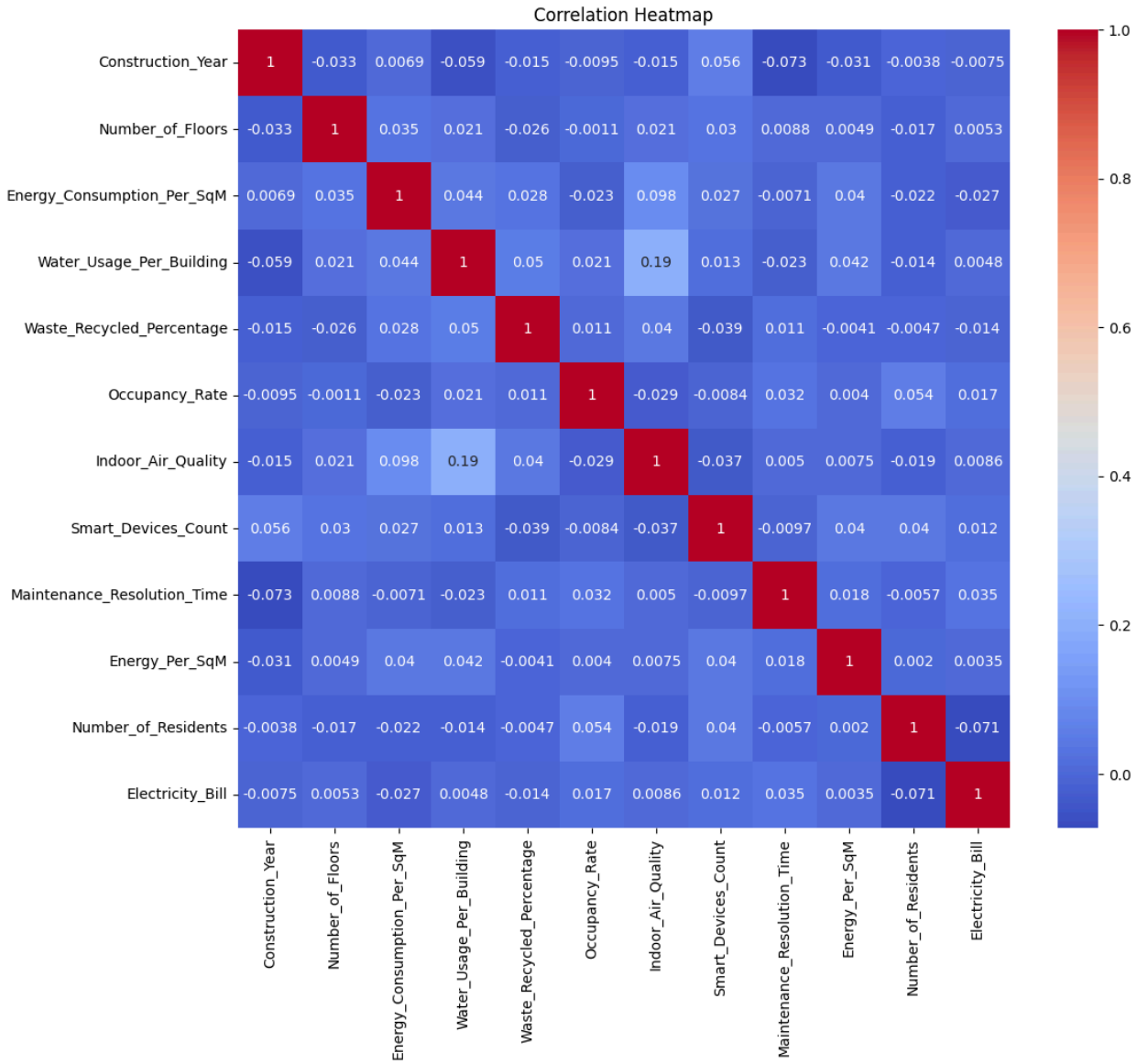
3. Box plots for non-categorical features:



4. Violin plots for non-categorical features:


Construction_Year
1960 1970 1980 1990 2000 2010 2020 2030
Number_of_Floors
0 2 4 6 8 10 12 14
Energy_Consumption_Per_SqM
50 100 150 200 250
Water_Usage_Per_Building
0 250 500 750 1000 1250 1500 1750
Waste_Recycled_Percentage
0 20 40 60 80
Occupancy_Rate
20 40 60 80 100
Indoor_Air_Quality
−20 0 20 40 60 80 100 120
Smart_Devices_Count
0 5 10 15 20
Maintenance_Resolution_Time
0 10 20 30 40 50 60 70
Energy_Per_SqM
50 100 150 200 250 300
Number_of_Residents
0 100 200 300 400 500
Electricity_Bill
0 5000 10000 15000 20000 25000 30000 35000

5. Correlation heatmap for non-categorical features:



Key insights on the dataset:

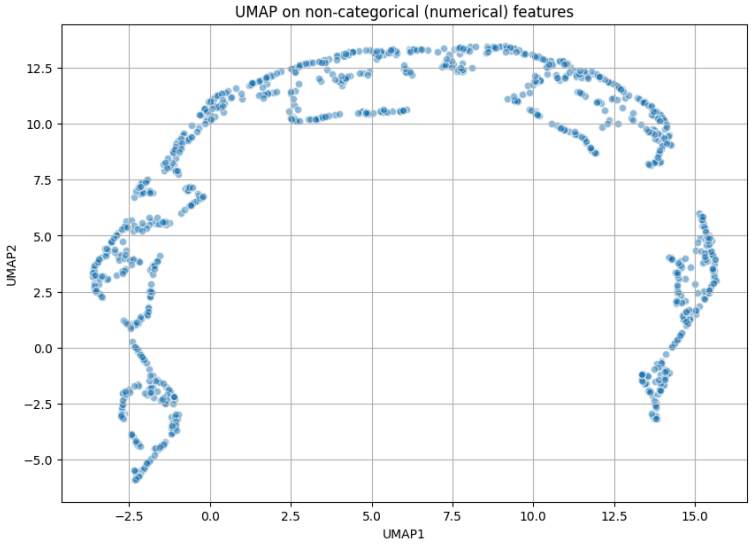
1. Count plots show that all categorical features except Green_certified are uniformly distributed.
2. All pair plots wrt Electricity_Bill have many scattered points, which means they won't be very helpful in model training and predictions.
3. Box plots show that Maintenance_Resolution_Time and Occupancy_Rate have a lot of outliers.
4. Electricity_Bill, Maintenance_Resolution_Time, Occupancy_Rate and Smart_Devices_Count have skewed distributions because their violin plots have a longer tail on one side.

5. The features are very weakly correlated with each other, the highest correlation we see is 0.19 between Indoor_Air Quality & Water_Usage_Building.

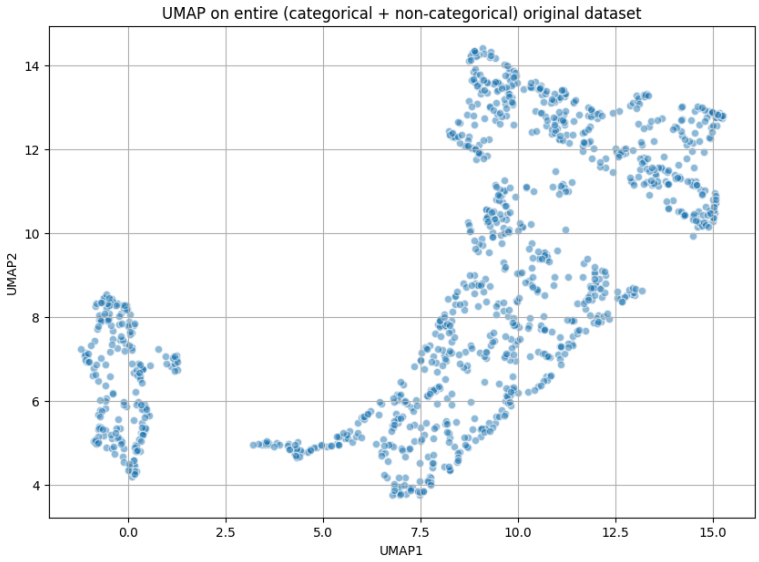
b) UMAP

We reduce the data dimensions to 2 with UMAP algo and then plot the scatter plots.

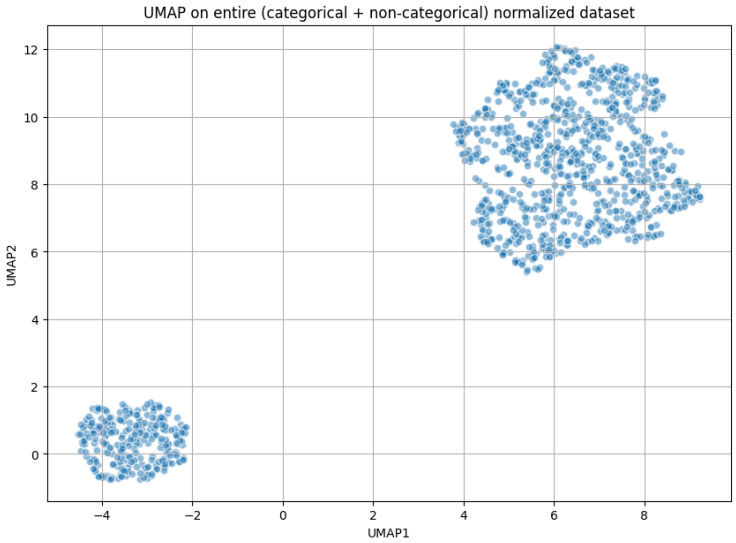
Firstly, I have plotted a UMAP for non-categorical (numerical) features. The plot shows no clustering and no signs of separability.



Secondly, I plotted a UMAP for all features after using label encoding on categorical features. The plot shows that data is separable but still clustering is not good.



Thirdly, I plotted a UMAP after applying Min-Max scaling on all features. The plot shows clear clustering and good separability. The separability and cluster size are directly dependent on the distribution of the Green_Certified feature.



c) Linear Regression

I applied LinearRegression model from scikit-learn on the dataset after preprocessing. I found that there were no missing values in the dataset before and applied label encoding while plotting UMAPs. Below are the performance metrics I found.

```
Training metrics:
MSE: 24475013.168475, RMSE: 4947.222773, R2: 0.013923, Adjusted R2: -0.001109, MAE: 4006.328469

Testing metrics:
MSE: 24278016.155743, RMSE: 4927.272689, R2: 0.000037, Adjusted R2: -0.064063, MAE: 3842.409313
```

d) Correlation Analysis and Related Results

I performed Correlation analysis on the original dataset which is label encoded and scaled (labelling is necessary and scaling doesn't affect the relevance of features). I found the correlation matrix and the top 3 relevant features were:

```
Top 3 Features:  ['Number_of_Residents' 'Green_Certified' 'Building_Type']
```

Now, training another model only on these 3 features, and the performance metrics are:

```
Training metrics:
MSE: 24569032.906898, RMSE: 4956.715940, R2: 0.010135, Adjusted R2: -0.004955, MAE: 4006.473378

Testing metrics:
MSE: 23941409.062998, RMSE: 4892.995919, R2: 0.013902, Adjusted R2: -0.049310, MAE: 3813.948128
```

The performance metrics in part c) and d) are only slightly different. We see that testing set MSE and MAE in d) are lesser than c) signifying model performs better after removing less important features.

e) One-Hot encoding and Ridge regression

I performed one-hot encoding on categorical features from original dataset, applied min-max scaling and applied Ridge regression on this data. The performance metrics are:

```
Training metrics:
MSE: 24189170.350024, RMSE: 4918.248708, R2: 0.025439, Adjusted R2: 0.002473, MAE: 3976.484672

Testing metrics:
MSE: 24119869.980905, RMSE: 4911.198426, R2: 0.006551, Adjusted R2: -0.094552, MAE: 3796.700550
```

The performance metrics in part c) and e) have still not improved much. We see that testing set MSE and MAE have improved slightly. Ridge regression is used to solve overfitting and no significant differences in metrics indicate that there is no overfitting in our model.

f) ICA

I performed ICA on the data with one-hot encoding. The results for different number of components are:

```
Results for  4  ICA components:
Training metrics:
MSE: 24664017.747731, RMSE: 4966.288126, R2: 0.006308, Adjusted R2: 0.002313, MAE: 4008.626465

Testing metrics:
MSE: 24429934.088252, RMSE: 4942.664675, R2: -0.006220, Adjusted R2: -0.022648, MAE: 3843.406824


Results for  5  ICA components:
Training metrics:
MSE: 24662402.316144, RMSE: 4966.125483, R2: 0.006373, Adjusted R2: 0.001375, MAE: 4007.623820

Testing metrics:
MSE: 24457063.548279, RMSE: 4945.408330, R2: -0.007337, Adjusted R2: -0.027979, MAE: 3845.490059


Results for  6  ICA components:
Training metrics:
MSE: 24657926.933578, RMSE: 4965.674872, R2: 0.006553, Adjusted R2: 0.000550, MAE: 4007.267720

Testing metrics:
MSE: 24503742.086319, RMSE: 4950.125462, R2: -0.009260, Adjusted R2: -0.034180, MAE: 3844.312378


Results for  8  ICA components:
Training metrics:
MSE: 24407603.429804, RMSE: 4940.405189, R2: 0.016638, Adjusted R2: 0.008700, MAE: 3978.601023

Testing metrics:
MSE: 24048353.738371, RMSE: 4903.912085, R2: 0.009497, Adjusted R2: -0.023383, MAE: 3777.973255
```

As we increase the number of components, we see a slight decrease in train and test MAE. We alse see a slight increase in train R square error which means with increasing components the model complexity increases. But this doesn't provide much improvement and insight into model's performance.

g) ElasticNet

I used the ElasticNet model on the dataset which is label encoded and min-max scaled from part c). The results for different values of alpha are:

```
ElasticNet with alpha=0.1:
Testing metrics:
MSE: 24179392.906361, RMSE: 4917.254611, R2: 0.004099, Adjusted R2: -0.059740, MAE: 3831.727821


ElasticNet with alpha=0.5:
Testing metrics:
MSE: 24233436.978265, RMSE: 4922.746894, R2: 0.001873, Adjusted R2: -0.062109, MAE: 3831.919426


ElasticNet with alpha=1.0:
Testing metrics:
MSE: 24276861.557231, RMSE: 4927.155524, R2: 0.000085, Adjusted R2: -0.064012, MAE: 3835.424757


ElasticNet with alpha=2.0:
Testing metrics:
MSE: 24312795.461590, RMSE: 4930.800692, R2: -0.001395, Adjusted R2: -0.065587, MAE: 3838.272971


ElasticNet with alpha=5.0:
Testing metrics:
MSE: 24342155.658763, RMSE: 4933.777018, R2: -0.002604, Adjusted R2: -0.066874, MAE: 3840.554178
```

As we increase alpha, we see a slight increase in test MSE, R square error, MAE. This indicates that the model's performance degrades as we increase the alpha (controls overall regularization strength).

h) Gradient Boosting Regressor

I used the Gradient Boosting Regressor to perform regression on the dataset which is label encoded and min-max scaled from part c). The performance metrics are:

```
Training metrics:
MSE: 14926446.257308, RMSE: 3863.475929, R2: 0.398626, Adjusted R2: 0.389459, MAE: 3092.748189

Testing metrics:
MSE: 24424706.773919, RMSE: 4942.135851, R2: -0.006005, Adjusted R2: -0.070492, MAE: 3820.286031
```

The training MSE is the least encountered so far in all model metrics indicating better performance on testing data. The test MAE and test MSE are more than part g) but less than part c) which means that even after gradient boosting model was not able to generalise well.

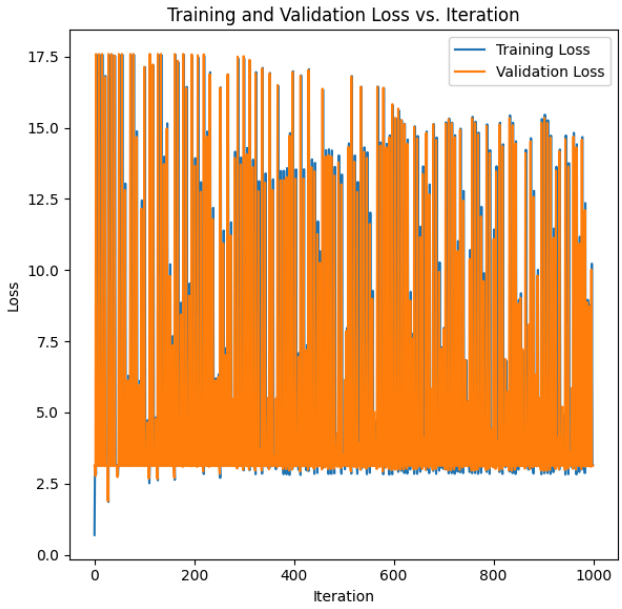
Section B

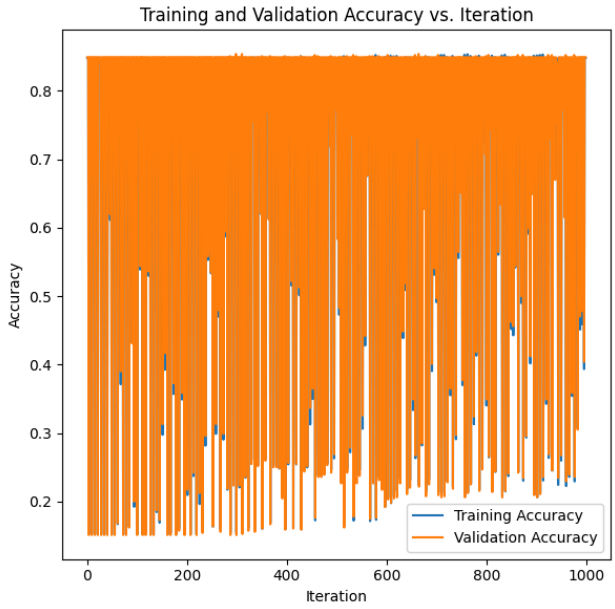
The problem is about implementing Logistic Regression using the various gradient descent algorithms. The data analysis and model training can be found

in the notebook. The Heart Disease dataset has some missing value and they are replaced with an appropriate estimate of the features.

a) Logistic Regression with batch gradient descent

I implemented this from scratch, the functions take parameters as training set (set means features and target are separated), validation set, learning rate and number of iterations. It sets bias and weights initially as zero. The loss function used is cross-entropy loss. Three helper functions are also used in the implementation. Below are the plots for the model:

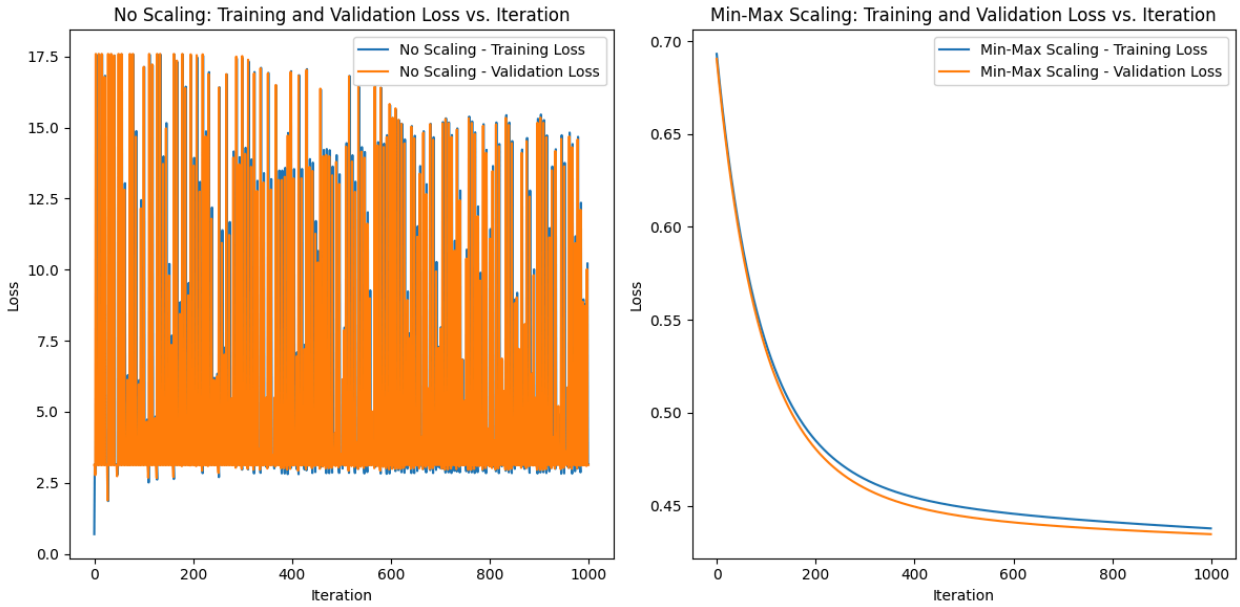




We cannot comment on the convergence of the model from these plots as they don't converge at all. (From the plots) Even after training the model for significant iterations, we still see large losses and low accuracy in further iterations. The model's performance is quite similar in testing and validation data.

b) Applying scaling

As the plots above were not quite insightful on model performance, we will now apply Min-Max scaling on the dataset and train the model again. I have implemented the Min-Max scalar from scratch. Below are the plots for the model after scaling the data:



In this case, feature scaling proved to be quite indispensable to comment on model convergence. Only after scaling, we can comprehend the plots where we see decreasing losses with every next iteration.
The model converges at around 800th iteration, after this losses become too small and model may begin to overfit testing data after that.

c) Performance analysis

The model obtained after scaling the dataset performed better (loss plots were comprehensible), so we will find the confusion matrix for the model from part b).

Below are the performance metrics when prediction threshold was 0.5:

```
Confusion Matrix:
[[540   0]
 [ 96   0]]
True Positives: 0
True Negatives: 540
False Positives: 0
False Negatives: 96
Precision:  0.0
Recall:  0.0
F1 Score:  0.0
ROC-AUC Score:  0.5
```

This indicates that the model predicts no for all instances in the validation set. The large number of true negatives also signify that the dataset is not distributed

uniformly and has a lot of no instances, leading to bad performance on yes instances.

Below are the performance metrics when prediction threshold was 0.25:

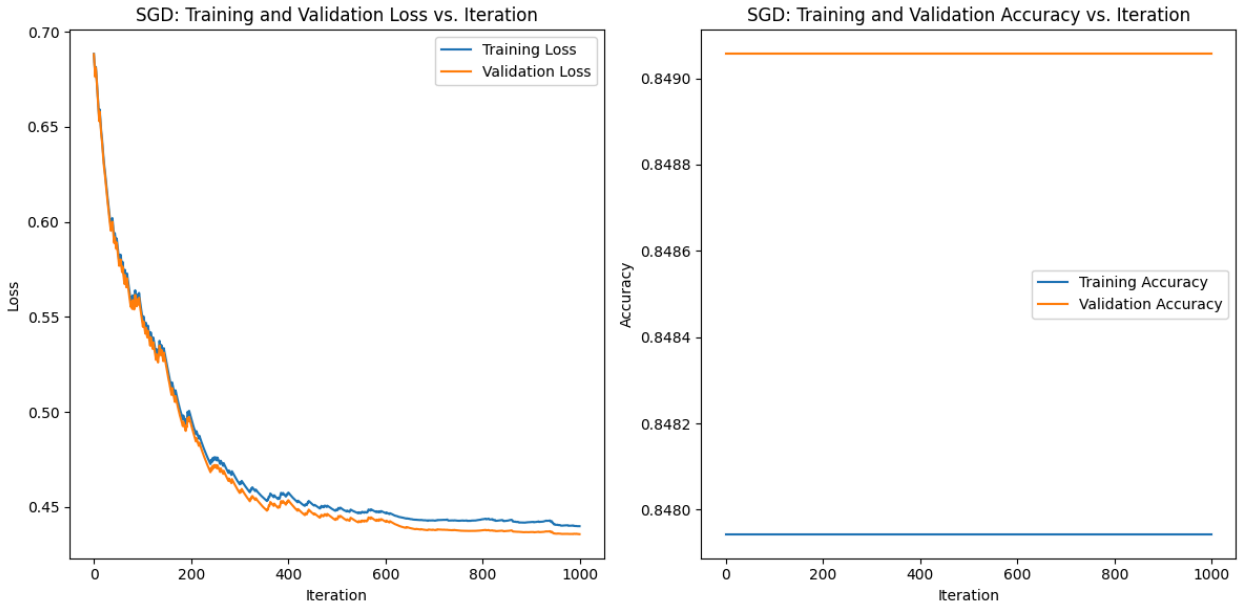
```
Confusion Matrix:
[[539   1]
 [ 96   0]]
True Positives: 0
True Negatives: 539
False Positives: 1
False Negatives: 96
Precision:  0.0
Recall:  0.0
F1 Score:  0.0
ROC-AUC Score:  0.499074074074074 06
```

Even after reducing the threshold to half of before, the model predicted one yes instance which is wrong signifies that the model cannot figure out yes instances and needs more dataset.

d) Stochastic and Mini-Batch Gradient Descents

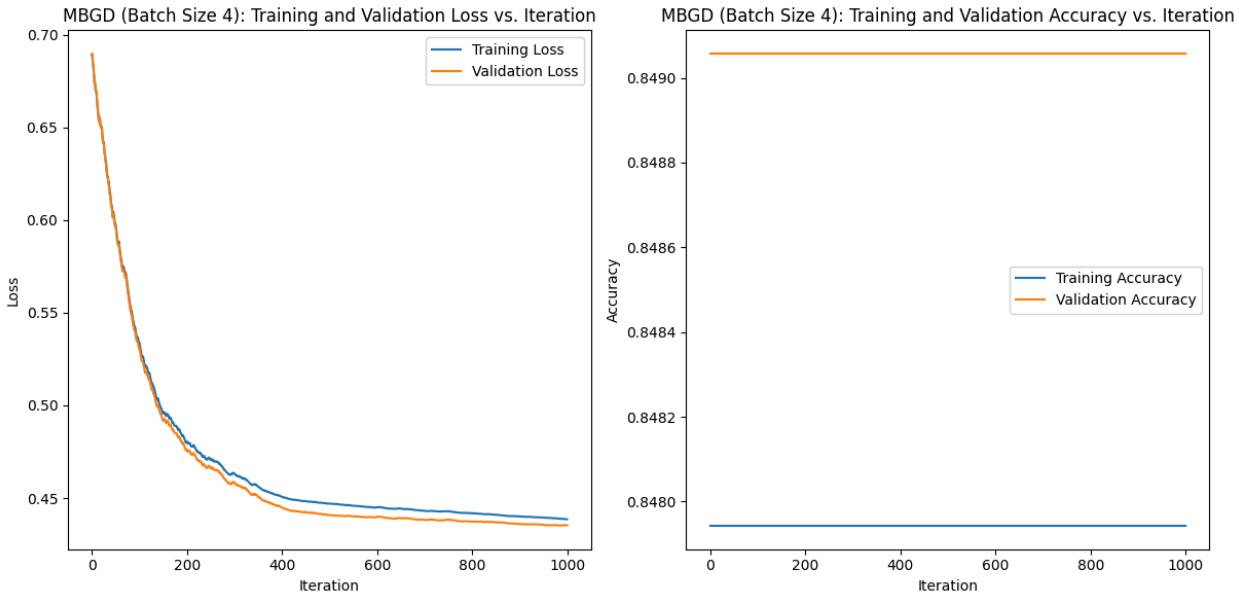
Both of these algorithms have two variations, here we will talk about the variation that is commonly used.

Below are the plots after training our model with SGD on scaled data:

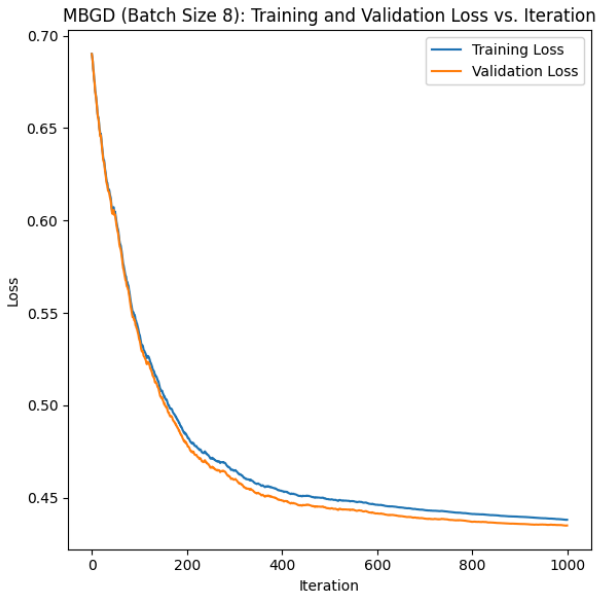


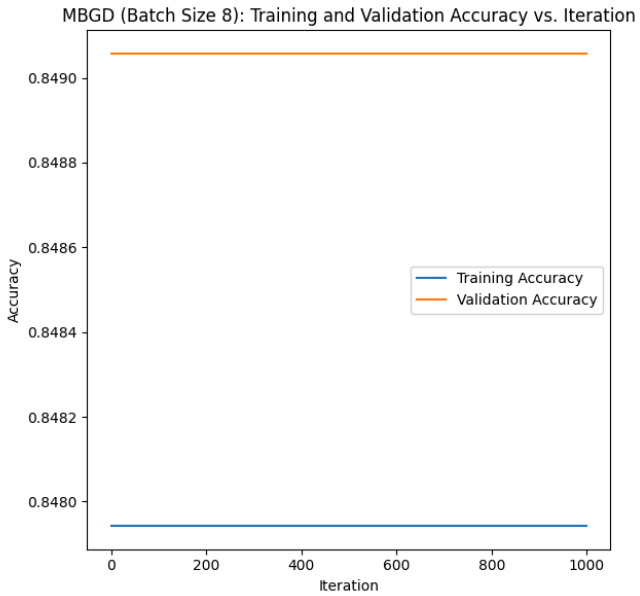
The graph shows very noisy behaviour due to the updation of weights and bias using only one data point. This indicates that the algorithm is unstable and the convergence is reached at around 650th iteration which is faster than BGD (part b).

Below are the plots after training our model with MBGD on scaled data with batch sizes 4 and 8:



The graph shows little noise. This means that the algorithm is more stable than SGD but less stable than BGD (smooth curve: no noise) and the convergence is reached at around the 650th iteration which is the same as SGD.





As we increase batch size, we see that noise reduces and the algorithm somewhat stabilizes. The algorithm is more stable than SGD but less stable than BGD (smooth curve: no noise) and the convergence is reached at around the 650th iteration which is the same as SGD and MBGD with batch size 4.

From the above plots, we conclude that we reached convergence earlier than the Batch Gradient Descent but the plots have more noise compared to the Batch Gradient Descent.

e) k-fold cross-validation

We are implementing k-fold cross-validation for k=5, which means we create 5 5-fold partition of the original dataset (with missing values filled with appropriate estimates) and experiment 5 times by training with 4 folds and validating on the remaining one. (train : validation = 80 : 20)
Below are the performance metrics we found:

```
Testing metrics for folds when model is trained with rest folds:
Fold 1: Accuracy = 0.8323, Precision = 0.3514, Recall = 0.0992, F1 Score = 0.1548
Fold 2: Accuracy = 0.8418, Precision = 0.0000, Recall = 0.0000, F1 Score = 0.0000
Fold 3: Accuracy = 0.8418, Precision = 0.0000, Recall = 0.0000, F1 Score = 0.0000
Fold 4: Accuracy = 0.1936, Precision = 0.1521, Recall = 0.9760, F1 Score = 0.2632
Fold 5: Accuracy = 0.8583, Precision = 0.0000, Recall = 0.0000, F1 Score = 0.0000

Accuracy: Average = 0.7136, Standard Deviation = 0.2601
Precision: Average = 0.1007, Standard Deviation = 0.1385
Recall: Average = 0.2150, Standard Deviation = 0.3824
F1 Score: Average = 0.0836, Standard Deviation = 0.1080
```

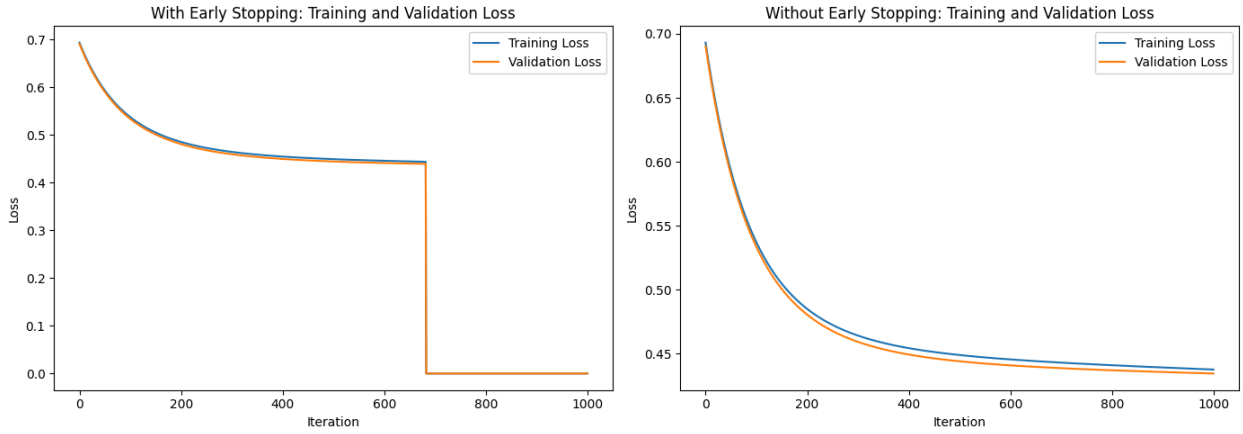
Precision, Recall, F1 score are 0 in Folds 2, 3 and 5 indicating that the model didn't predict a single positive outcome in these folds, yet these folds show a high accuracy implying that the dataset is not uniformly distributed (has very less positive instances).
Fold 4 shows a very high recall and very low accuracy indicates that model predicted many positive values correctly but failed to predict negative values this time.
The accuracy is stable across all folds except Fold 4, this is clearly depicted in high standard deviation as well. This implies that the dataset has a high variance.

Overall, we can conclude that the model consistently fails to predict positive instances as indicated by average precision and recall and the relatively higher standard deviations indicate instability across folds.

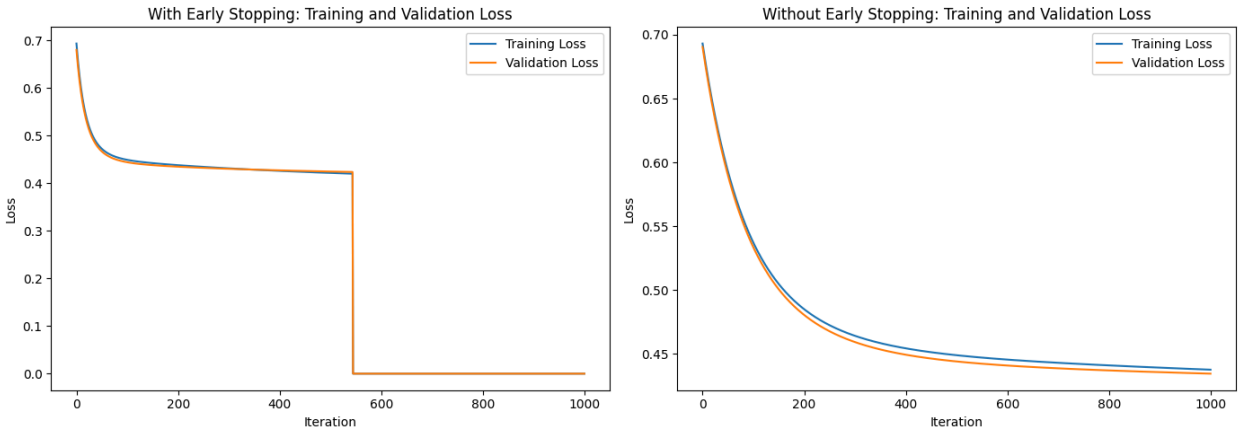
f) Early stopping

An appropriate criterion to stop early to avoid overfitting would involve comparing validation loss. Let's say that after an iteration, the improvement in validation loss is less than 0.0001 (1e-4) for threshold_iterations (any small number) iterations we should stop after that iteration to avoid overfitting.

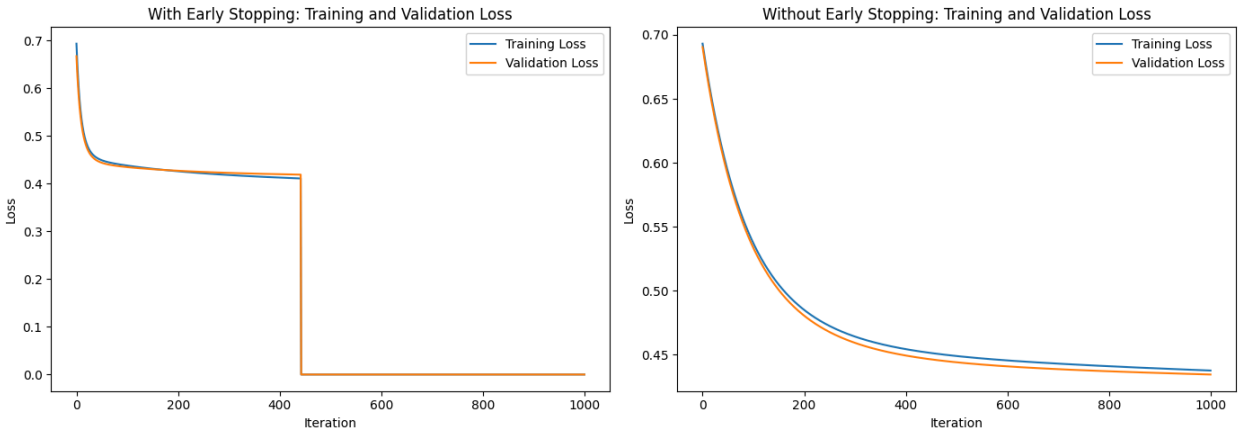
Below are the results we found for learning rate = 0.01:
Early stopping at iteration 681



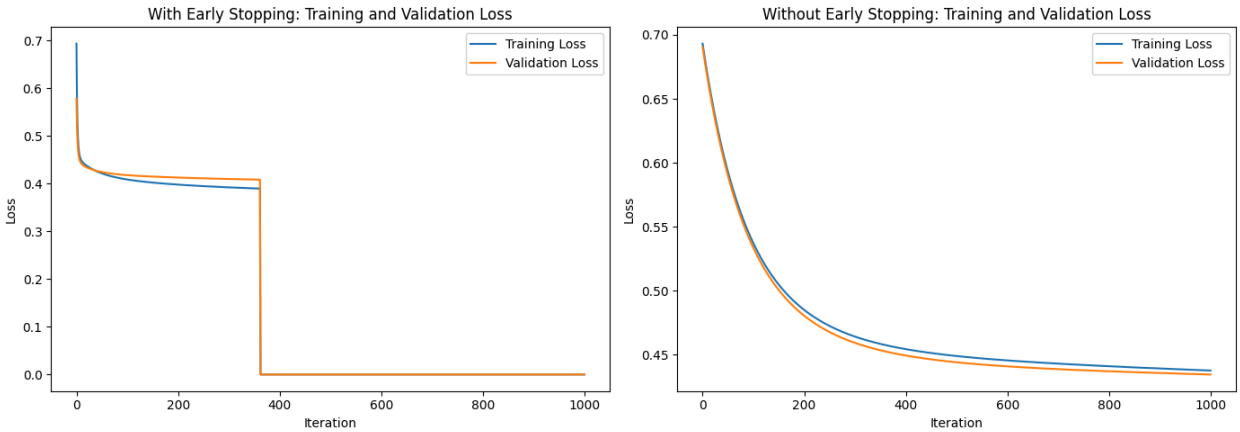
Below are the results we found for learning rate = 0.05:
Early stopping at iteration 543



Below are the results we found for learning rate = 0.1:
Early stopping at iteration 441



Below are the results we found for learning rate = 0.5:
Early stopping at iteration 361



We can see that the early stopping function stops the model training as the change in validation goes below 0.0001. We also observe that as we increase the learning rate, our model converges faster and without the incorporation of early stopping, the model would begin to overfit the training data and will not be able to generalise on unseen data.