Assignment-1
Machine Learning

Section A - Theoretical

(a) Increasing the complexity of the model that is designed for the task of prediction, either by increasing the number of features or by including higher-order polynomial terms in a regression model will possibly affect the bias-variance tradeoff.
Bias is the error introduced by the model in approximating complex problem by a simplified model.
Variance is the change in model's prediction when the model is with different datasets.
Understanding Bias-Variance Tradeoff by considering following scenarios:

- **Low Complexity Model:**
  - Low complexity models are generally, too simple to analyse and capture underlying patterns in the data and understand the complexity of the relationships between features and the target.
  - Thus model makes strong assumptions about some new data, try to fit in the model that does not generlizes well to the unseen data, known as underfitting, and hence, the model is highly biased.
  - Thus, low complexity model are highly biased.
  - Also, simple models tend to have low variance because they are not capable of identifying underlying patterns in the data and in the noise. Thus, they remain stable when the model is trained on different datasets.
  - When we increase the complexity of the model(by increasing the number of features or by including higher-order polynomial terms):
    - **Bias decreases:** Because now the model is capable and complex enough to identify underlying

patterns in the data, it starts to better fit the training data and generalizes well to the unseen data, as it's assumptions becomes more aligned to the data's complexity.
        - **Variance increases:** As model's complexity increases, it will start identifying patterns in the data well and in some cases, it will starts to identify patterns and observations in the noise as well. Thus, model starts to overfit the data and becomes sensitive to the fluctuations in the training data.
- **High Complexity Model:**
    - High complexity model, usually, becomes too complex for the training data, such that it starts to identify patterns in the noise as if it were a true pattern.
    - Thus model starts to overfit the data, and make generalizations that may not reflect and represent true relationship between the output feature and the input features.
    - These type of models are highly sensitive to the training data, as there performance show fluctuations when trained with different datasets. Thus, these type of models show high variance.
    - Increasing the complexity of model will have the following implications:
        - **Variance Increases:** Increasing the model complexity can lead to model being more sensitive to smaller fluctutations/variations in the training data, thus increasing the variance of the model.
        - **Bias Decreases:** Increasing the complexity tend to make models capable of capturing underlying patterns in the data, thereby reducing bias as model fits the training data extremely well.

(b) We can evaluate the performance of the model by performance metrics like Precision and Sensitivity(Recall) and Accuracy.

Here, we have to define several terms with reference to the current scenario:

- **True Positive(TP):** When model correctly classifies spam email as spam. In our case, 200 emails are correctly classified as spam. Therefore, True Positive(TP) = 200.
- **True Negative(TN):** These are the cases when model correctly classifies legitimate emails as legitimate. In our case, 730 emails are correctly classified as spam. Therefore, True Negative(TN) = 730.
- **False Negative(FN):** These are the cases when model incorrectly classifies spam emails as legitimate. In our case, 50 spam emails are incorrectly classified as legitimate. Therefore, False Negative(FN) = 50.
- **False Positive(FP):** These are the cases when model incorrectly classifies legitimate emails as spam. In our case, 20 legitimate emails are incorrectly flagged as spam. Therefore, False Positive(FP) =20.

Computing Performance Metrics:

- **Precision:**
  - It measures out of all the instances where our model predicted positive, in how many of those cases, it is actually positive.
  - In our context, Precision will be of all the cases where the email is flagged as spam, how many of them were actually spam.
  - Precision(for spam emails) = TP/(TP + FP) =
    =200/(200+20) = 200/220 = 0.9099
- **Sensitivity(Recall):**
  - This performance metric measures how many actual spam emails are correctly identified.
  - Recall(for spam emails) = TP/(TP + FN) = 200/(200 + 50)
    = 200/250 = 0.8
- **Specificity:**
  - This performance metric measures how many actual legitimate emails are correctly identified.
  - Specificity(for legitimate emails) = TN/(TN + FP)

= 730/(730 + 20)
= 730/750 = 0.9733

- **Accuracy:**
  - This performance metric measures how many emails(including spam and legitimate) are correctly classified.
  - Accuracy = (TP + TN)/(TP + TN + FP + FN)
    = (
- **Negative Predicitive Value:**
  - It measure out of all the emails that are predicted as legitimate, how many of them were actually legitimate.
  - Negative Predictive Value= TN/(TN + FN)
    = 730/(730 + 50) = 730/780 = 0.9359
- **F1-Score:**
  - It measures the balance between Precison and Recall.
  - F1-score = 2*(precision*recall)/(precison + recall)
    = 2*(0.9099*0.8)/(0.9099 + 0.8)
    = 1.4558/1.7099
    = 0.8514

(c) Independent variable or Input Feature in this case = x
Dependent Variable or Target variable in this case = y
Because there is a linear relationship between input feature and output feature, we can use linear regression to find the best fit line that fits the data well.
Let the equation of such best fit line will be: $y = \beta_0 + \beta_0 x$

- For any input point $x^{(i)}$, its output can be given by — $y^{(i)} = \beta_0 + \beta_1 x^{(i)}$ where $i = 1, 2, \ldots, m$ and m — Total no. of samples in the training data.
- Matrix for Y (The Output feature):-

$$Y = \begin{bmatrix} 15 \\ 30 \\ 55 \\ 85 \\ 100 \end{bmatrix}$$

- Matrix for X (The Input feature):—

$$X = \begin{bmatrix} 1 & 3 \\ 1 & 6 \\ 1 & 10 \\ 1 & 15 \\ 1 & 18 \end{bmatrix}$$

Column For the Coefficient β_0

- Coefficients:

$$\hat{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix}$$

We can find the coefficents vector using the Least Square Method by which:

$$\Rightarrow \hat{\beta} = \left[(X^T X)^{-1} X^T\right] Y$$

$$\Rightarrow X^T = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 3 & 6 & 10 & 15 & 18 \end{bmatrix}$$

$$\Rightarrow X^T X = \underset{2\times5}{\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 3 & 6 & 10 & 15 & 18 \end{bmatrix}} \underset{5\times2}{\begin{bmatrix} 1 & 3 \\ 1 & 6 \\ 1 & 10 \\ 1 & 15 \\ 1 & 18 \end{bmatrix}} = \underset{2\times2}{\begin{bmatrix} 5 & 52 \\ 52 & 694 \end{bmatrix}}$$

$$\Rightarrow (X^T X)^{-1} = \frac{1}{5\times694 - 52\times52} \left\{ \begin{bmatrix} 694 & -52 \\ -52 & 5 \end{bmatrix} \right\} \text{adj}(X^T X)$$

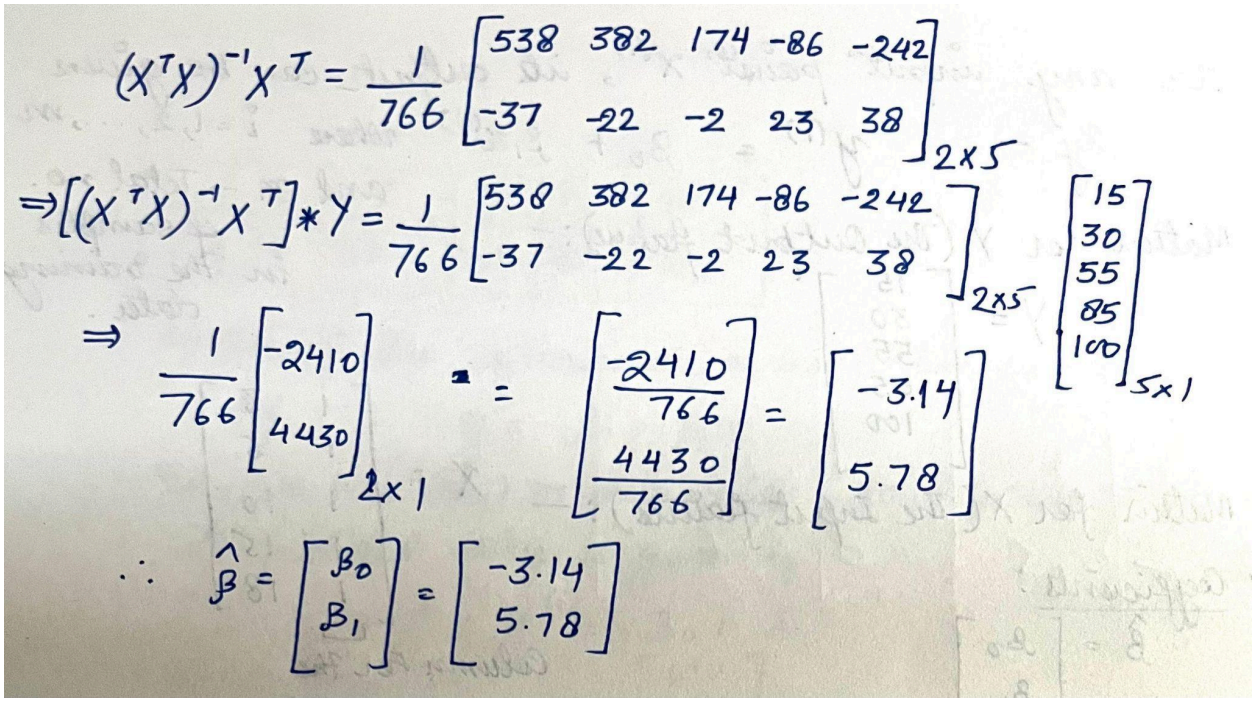
$$= \frac{1}{5 * 694 - 52 * 52} \begin{bmatrix} 694 & -52 \\ -52 & 5 \end{bmatrix}$$

$$= \frac{1}{3470 - 2704} \begin{bmatrix} 694 & -52 \\ -52 & 5 \end{bmatrix}$$

$$= \frac{1}{766} \begin{bmatrix} 694 & -52 \\ -52 & 5 \end{bmatrix}$$

$$\Rightarrow (X^T X)^{-1} X^T = \frac{1}{766} \underset{2\times2}{\begin{bmatrix} 694 & -52 \\ -52 & 5 \end{bmatrix}} * \underset{2\times5}{\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 3 & 6 & 10 & 15 & 18 \end{bmatrix}}$$

$$= \frac{1}{766} \underset{2\times5}{\begin{bmatrix} 538 & 382 & 174 & -86 & -242 \\ -37 & -22 & -2 & 23 & 38 \end{bmatrix}}$$



Therefore, β_0 = -3.14 and β_0 = 5.78.
Therefore, equation of best fit line can be given by:

$$y = 5.78x - 3.14$$

Therefore, for any input point x = 12, the predicted value will be :

y = 5.78*12 - 3.14 = 69.36 - 3.14 = 66.22

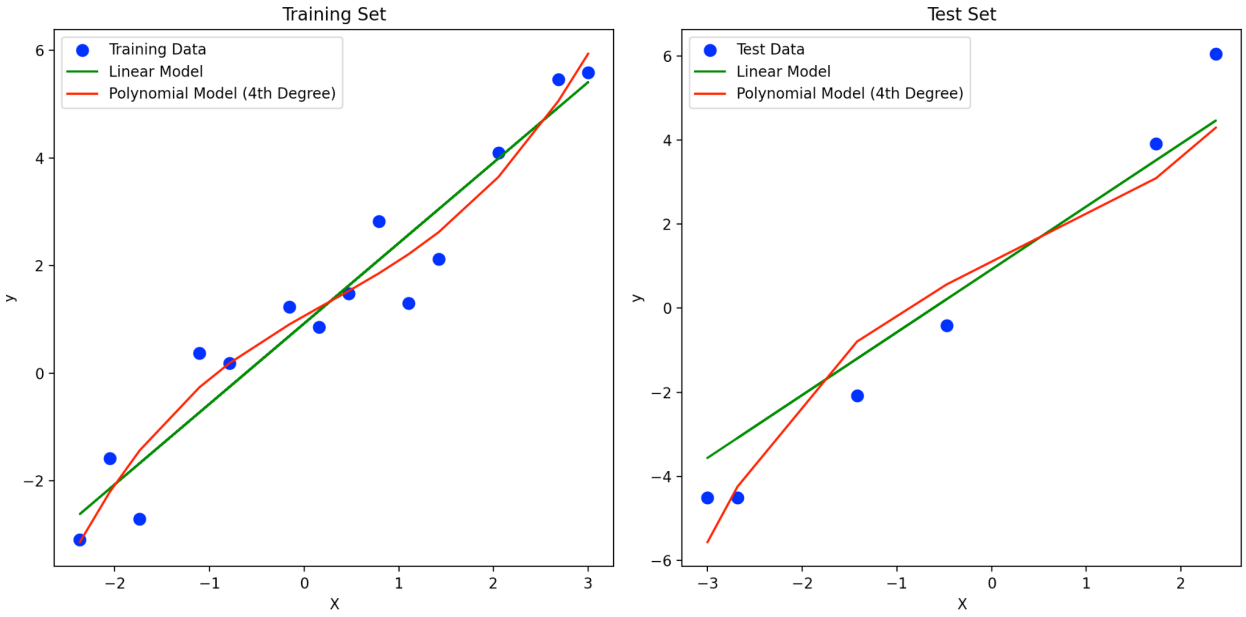
(d) Empirical Risk is the expected loss of a model when it is trained on the training dataset. It basically measures how well a data perform on a training data.

The lower the empirical risk, the better the model fits the data. However, it may also lead to overfitting, which makes it to find the patterns in the training data which may not generalize well to the unseen data.

Thus, it is not necessary that model that fits training data too well or have lower empirical risk minimization will generalize necessarily better than the model with higher empirical risk.

Let us consider an example, where we have labeled data points:
(X, Y), where for each input x, its output y is labeled as :
$y = 2*x + 1 + \varepsilon$, where ε - Noise ~ $N(0, \sigma^2)$.

After training this model on training set and test set, we get these plots:



- Linear Regression Equation: *y = 1.49x + 0.92*
- Polynomial Regression Equation (4th Degree):
  y = -0.00x^4 + 0.10x^3 + -0.07x^2 + 0.99x^1 + 0.00 + 1.06

For a unseen data with label, (2.5, 6):

➔ The predicted value by linear model: y* = 4.645

➔ The predicted value by polynomial model: y* = 4.584

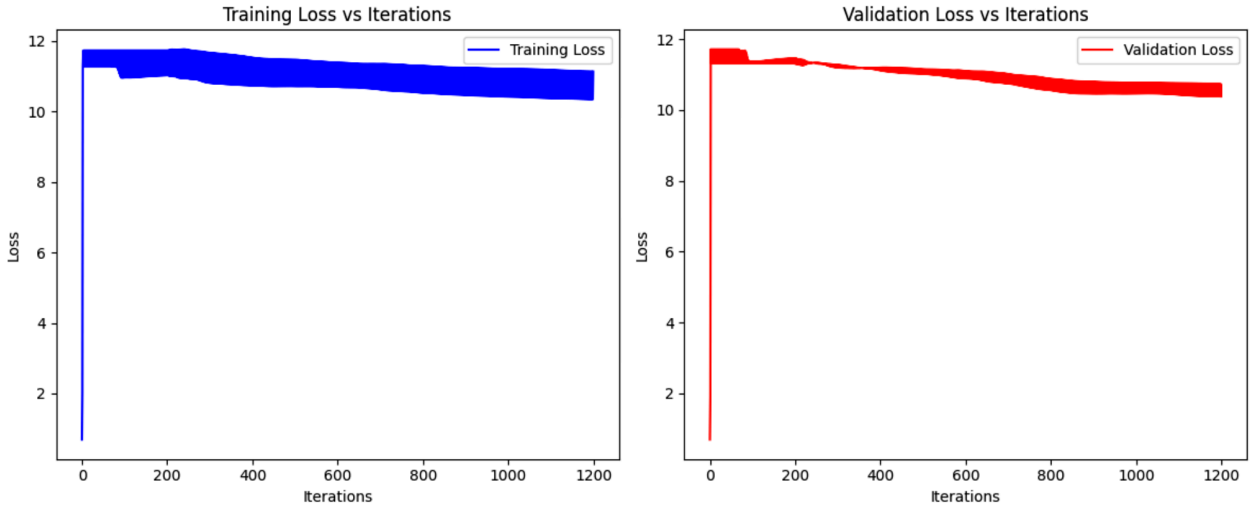
As, we see linear model predicted value closer to the ground truth value in comparison to polynomial mode.

Also, quadratic polynomial model performs well on the training data and have lower risk minimization as compared to the linear model.

However, when the model is evaluated on test data(which is unseen data), it is pretty evident that higher degree polynomial model performs does not generalize well as compared to the training data.

Section B:

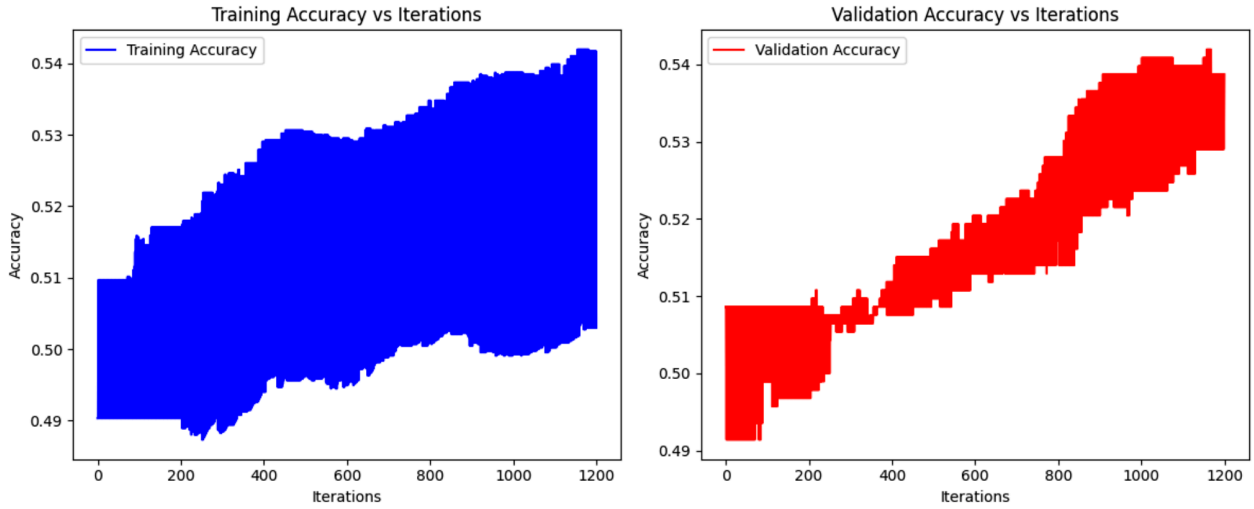
(a)



(i) Plot for Training Loss vs Iterations and Validation Loss vs Iterations

Plot Analysis:

- Both training loss and validation loss show similar trends .
- Both the training loss and validation loss spike as the iterations begins and then they stabilizes after some initial iterations.
- After that, both of the losses does not show significant improvement.



(ii) Plot for Training Accuracy vs Iterations and Validation Accuracy vs Iterations

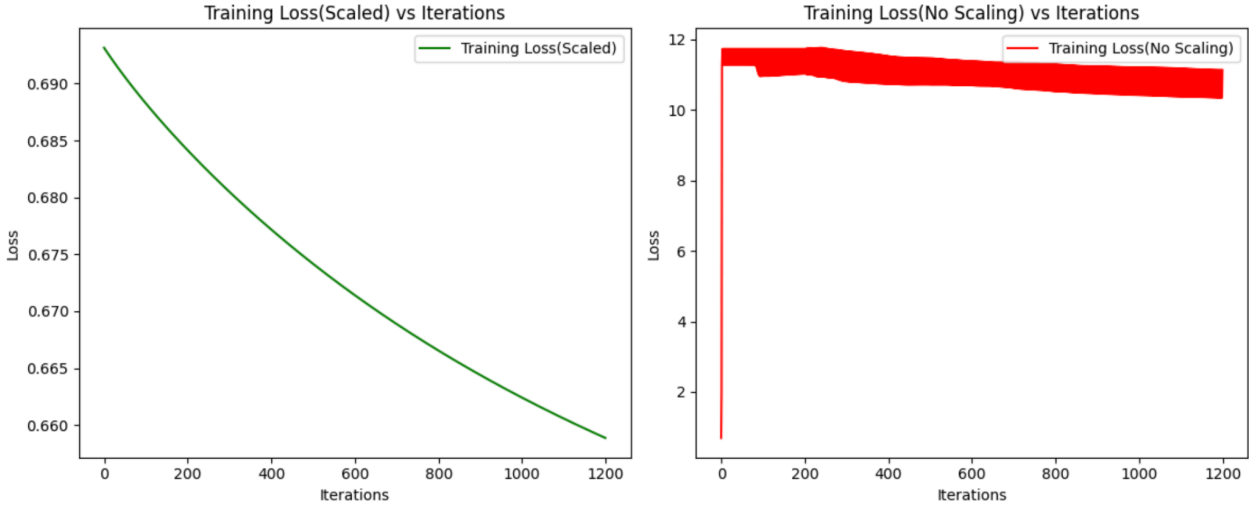
Plot Analysis:

- Both Training accuracy and Validation accuracy show an increase as the number of iterations increases.
- While training accuracy show a steady increase with iterations.
- Validation accuracy also increases with iterations but in a more volatile way as it shows larger fluctuations than the training accuracy.

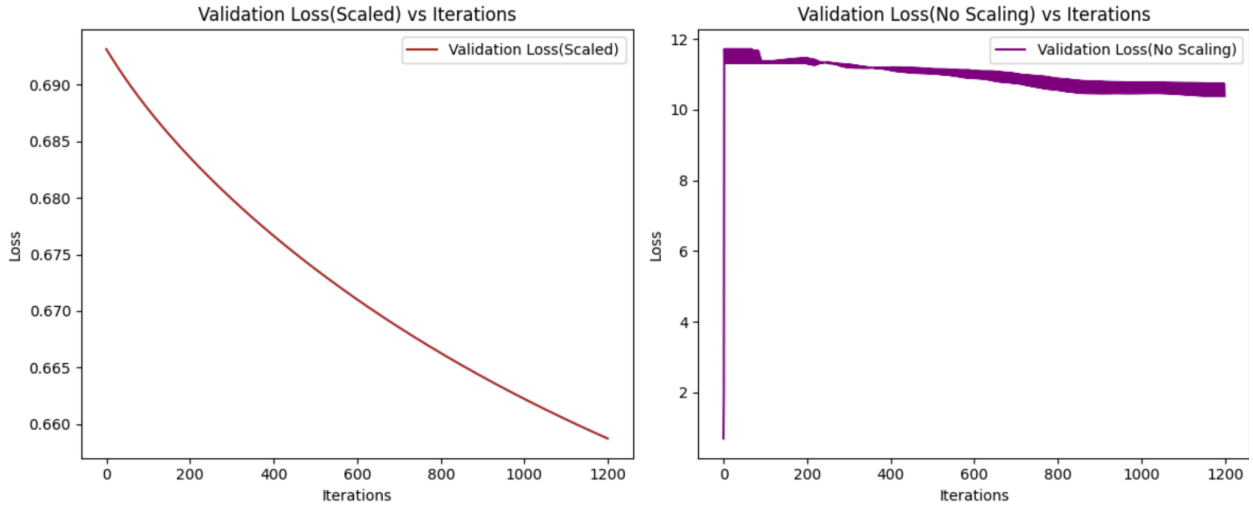
Convergence Analysis:

- From, the plots it is clear that both the graphs are relatively close, which indicates that model is not sensitive to the training datasets. Thus, we can say that the model has low variance, hence, it is not overfitting the data.
- Both the training loss and validation loss tends to stabliize after some time and converges to a certain level, indicating that the model has likely reached its capacity to reduce error on the training set.

(b)

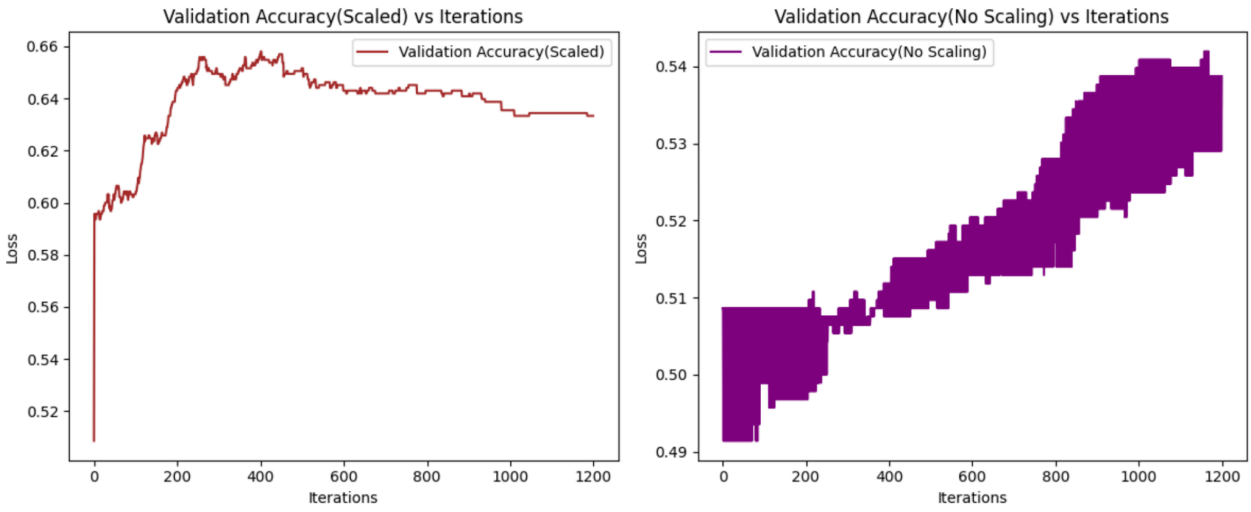


(i) Plot of Training Loss vs Iterations when scaling is allowed as comparison to when scaling is not allowed.



(ii) Plot of Validation Loss vs Iterations when scaling is allowed in comparison to when scaling is not allowed.

(iii)Plot of Training Accuracy vs Iterations when scaling is allowed in comparison to when scaling is not allowed.



(iv) Plot of Training Accuracy vs Iterations when scaling is allowed in comparison to when scaling is not allowed.

- As we can observe from the plot, Min-max model has a significant impact over model's performance.
- Applying min-max scaling has accelerated convergence, unlike when no scaling is applied as logistic regression works more efficiently when feature values are on the similar scale.
- It also normalizes feature values as Min-max scaling rescales all the features to a common range.
- Min-max scaling has also increased model's performance as model's accuracy is increased by more than 10%.

(c) The output that we are getting from our program where we calculate different performance metric for our model with scaling is given by:

```
Confusion Matrix(With Scaling):  [[298 159]
 [182 291]]
Recall/Sensitivy(With Scaling):  0.6152219873150105
Precision(With Scaling):  0.6466666666666666
F1-Score(With Scaling):  0.6305525460455037
ROC Score(With Scaling):  0.6336503809660392
```

From the performance metric, it is pretty evident that:

- Confusion Matrix:
    - Model predicted a fair share of positive and negative cases, i.e. the model correctly predicted 298 positive cases and 291 negative cases. However, it also makes a fair amount of mistakes as it has predicted incorrectly 159 negative cases and 182 positive cases.
    - Although, model is able to predict majority of cases correctly, a significant improvement is required to improve its accuracy.
- Recall:
    - Model is moderately capable of identifying positive cases, but still misses a significant chunk of them also.
- Precision:
    - Model is capable of making correct positive predictions, however, still, it fails 35.4% of time in making correct positive predictions.
- F1-Score:
    - The F1-score suggests the model is moderately balanced between precision and recall, but there is room for improvement in both areas.
- ROC Score:
    - The score suggests that our model is moderately effective in distinguishing between positive and negative cases.

(d)

On comparing the graphs of **Training Loss vs Iterations**, **Testing Loss vs Iterations**, and **Validation Loss vs Iterations**, along with **Testing Accuracy vs Iterations** and **Validation Accuracy vs Iterations**, it is revealed that with **SGD**, we observe significant fluctuations. However, in the case of **Mini-Batch Gradient Descent** with batch sizes of 32 and 64, the fluctuations are minimized. Overall, **Mini-Batch 64** performs the best in our case.

(e) The result after applying k-fold cross validation in our logistic regression model are as follows:

Average Metrics :-

- **Accuracy -** 0.6413 ± 0.0099
- **Precision -** 0.6241 ± 0.0231
- **Sensitivity -** 0.7172 ± 0.0314
- **F1 Score -** 0.6663 ± 0.0057

Fold-wise Metrics -

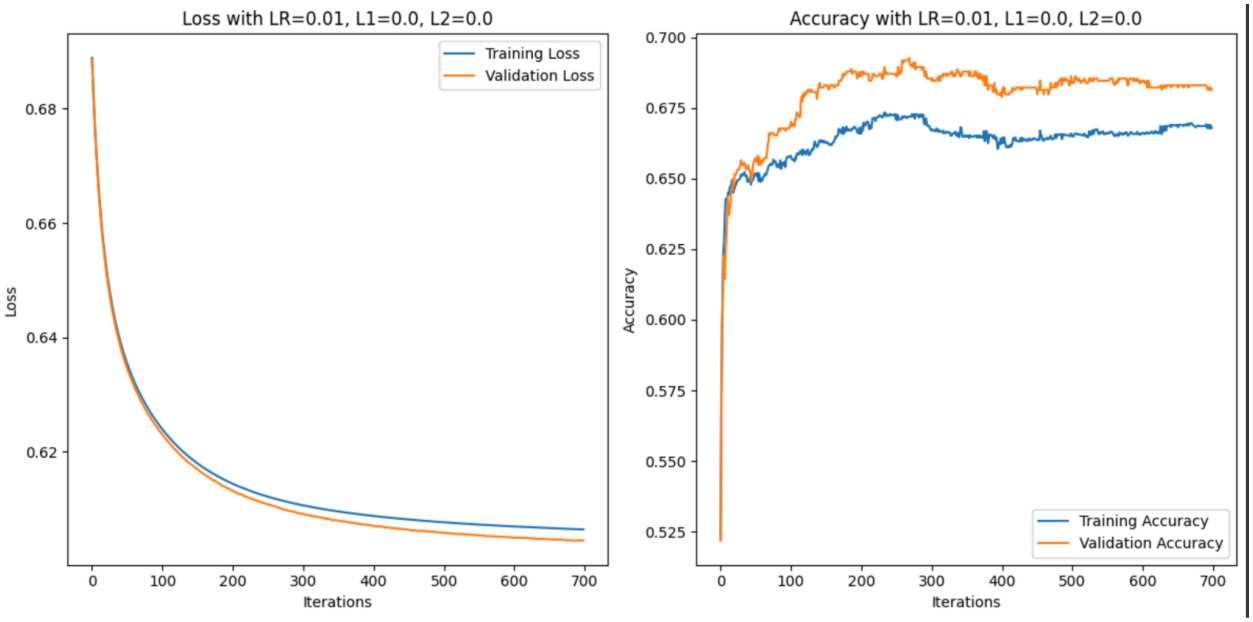
- **Fold 1 :**
  - Accuracy - 0.6449
  - Precision - 0.6134
  - Sensitivity - 0.7504
  - F1 Score - 0.6750
- **Fold 2 :**
  - Accuracy - 0.6336
  - Precision - 0.6334
  - Sensitivity - 0.7003
  - F1 Score - 0.6652
- **Fold 3:**
  - Accuracy - 0.6263
  - Precision - 0.5882
  - Sensitivity - 0.7487
  - F1 Score - 0.6588
- **Fold 4:**
  - Accuracy - 0.6489
  - Precision - 0.6270
  - Sensitivity - 0.7199
  - F1 Score - 0.6702
- **Fold 5:**
  - Accuracy - 0.6529
  - Precision - 0.6583
  - Sensitivity -0.6667
  - F1 Score - 0.6625

From the data, it can be observed that:

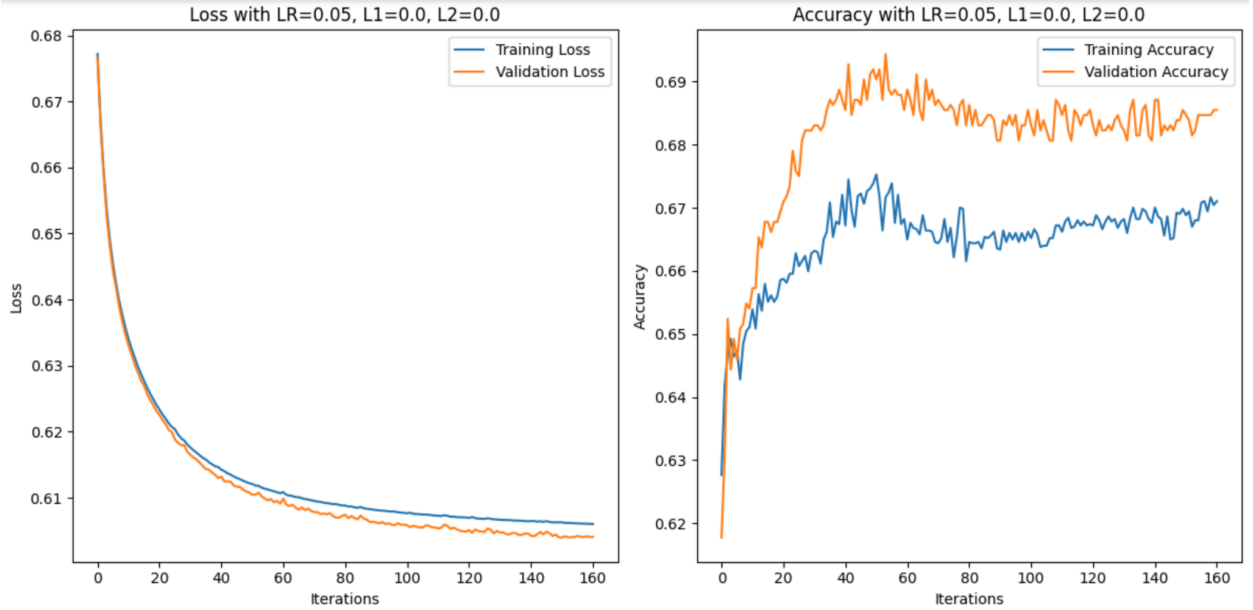
- The relative difference between performance metrics across all the folds lies within a range of 0 to 0.6.
- This difference is relatively smaller suggesting that the variance across all the five folds are lower. Thus, model performs with low variance, thus avoiding the possibility of overfitting.
- The relatively smaller standard deviation across accuracy and f1-score suggest that model performs consistently across different folds of the data.
- However, the standard deviation is relatively larger across performance metrics like precision and recall score suggests that there is variability which could be attribute to the imbalance present in the data, that results in imbalance distribution of positive and negative class in the data.
- Overall, the model is stable and has low variance, however, further tuning and validation may help in model's performance further.

(f) Early stopping criteria in our case will be validation loss. As validation loss measures the performance of the model before final evaluation on the data that it has not seen. By monitoring the validation loss across the iterations, we can halt the training process at the moment where the model starts to overfit the data, thus, preventing the model from overfitting. When we observe that validation loss stabilizes after some iterations and it is not showing further significant improvements, we can halt our training process then, thus, saving unnecessary training and computational cost and improving model's performance.

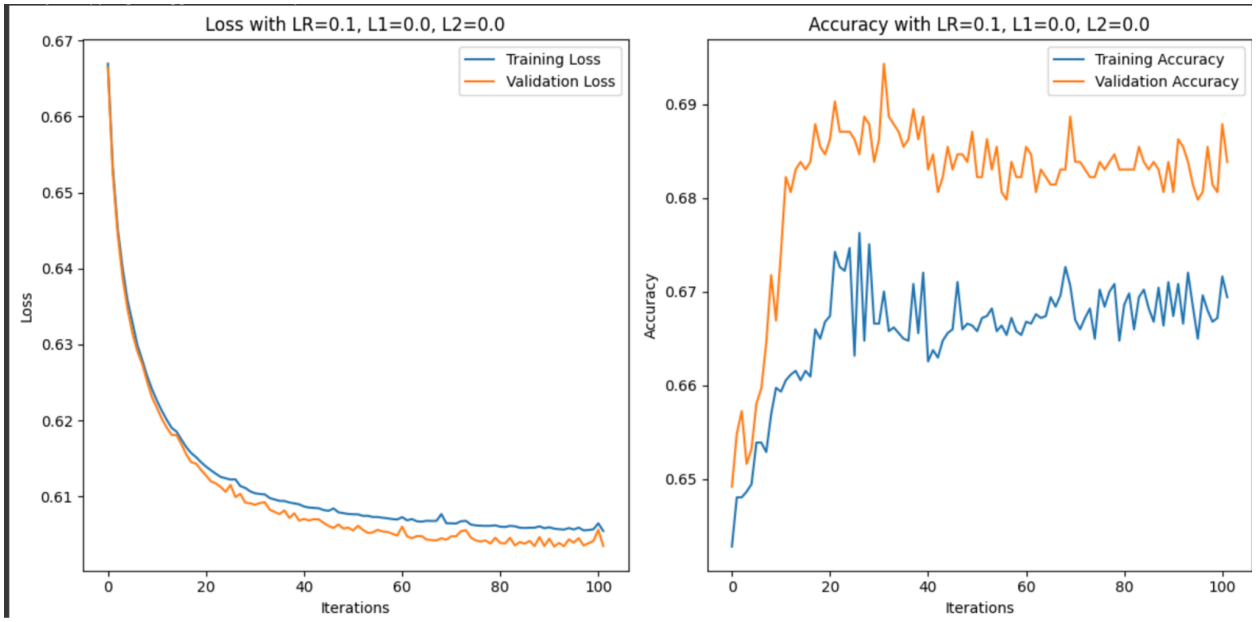
Plots showing effect of variation in learning rate(With Early Stopping) on Loss and Accuracy:



Plot showing Loss vs Iterations and Accuracy vs Iterations for the case when learning rate is 0.01



Plot showing Loss vs Iterations and Accuracy vs Iterations for the case when learning rate is 0.05

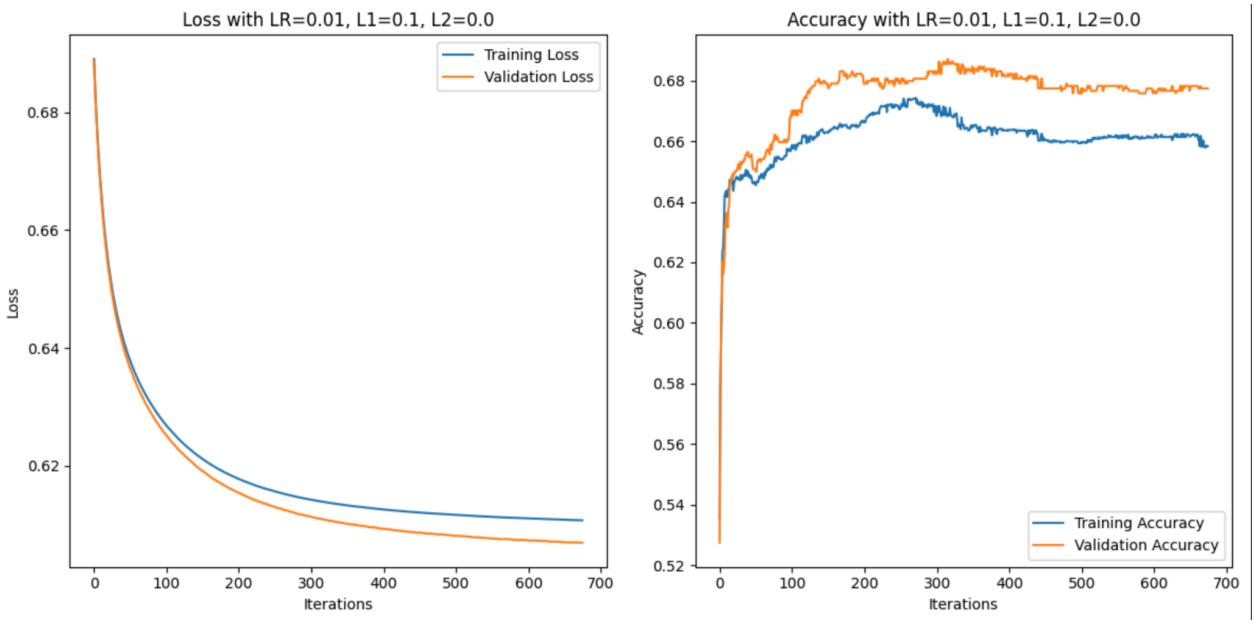


Plot showing Loss vs Iterations and Accuracy vs Iterations for the case when learning rate is 0.1

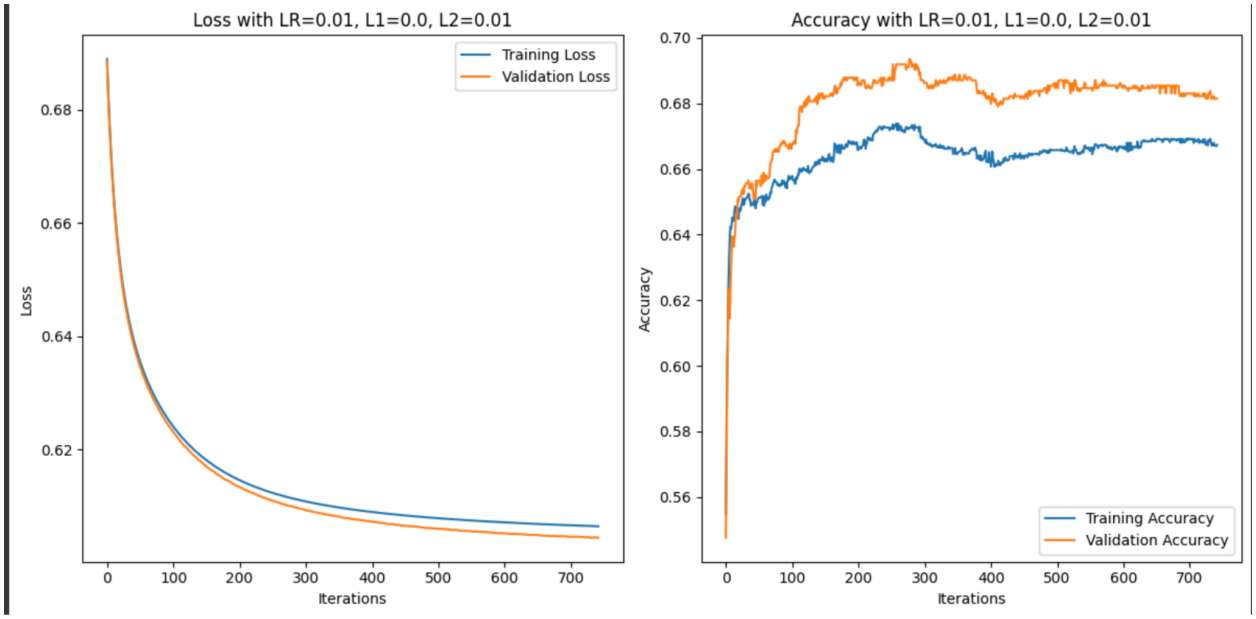
Plot Analysis:

- With lower learning rate, the training and validation loss decreases in a slow manner, but ultimately achieved convergence in a steadily fashion indicating that there is no overfitting condition.
- With moderate learning rate, the convergence rate is quicker, but the possibility of overfitting increased as training accuracy and validation accuracy are diverging.
- With high learning rate, smaller fluctuations are observable in loss vs iterations graph with more volate accuracy vs iterations graph. This indicate that model is struggling to converge properly. This can lead to poor performance of the model.

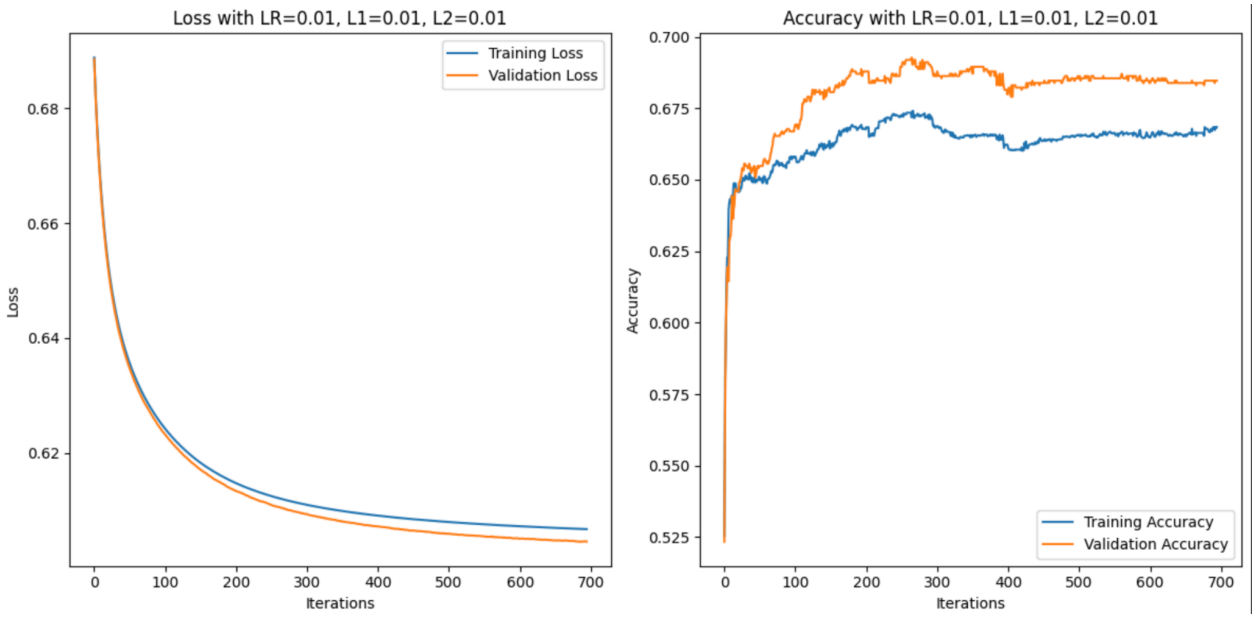
Plots showing Effect of Regularization(With Early Stopping) on Loss and Accuracy of the model:



Plots showing effect of L1 regularization(lasso regularization)



Plots showing effect of L2 regularization(Ridge regularization)

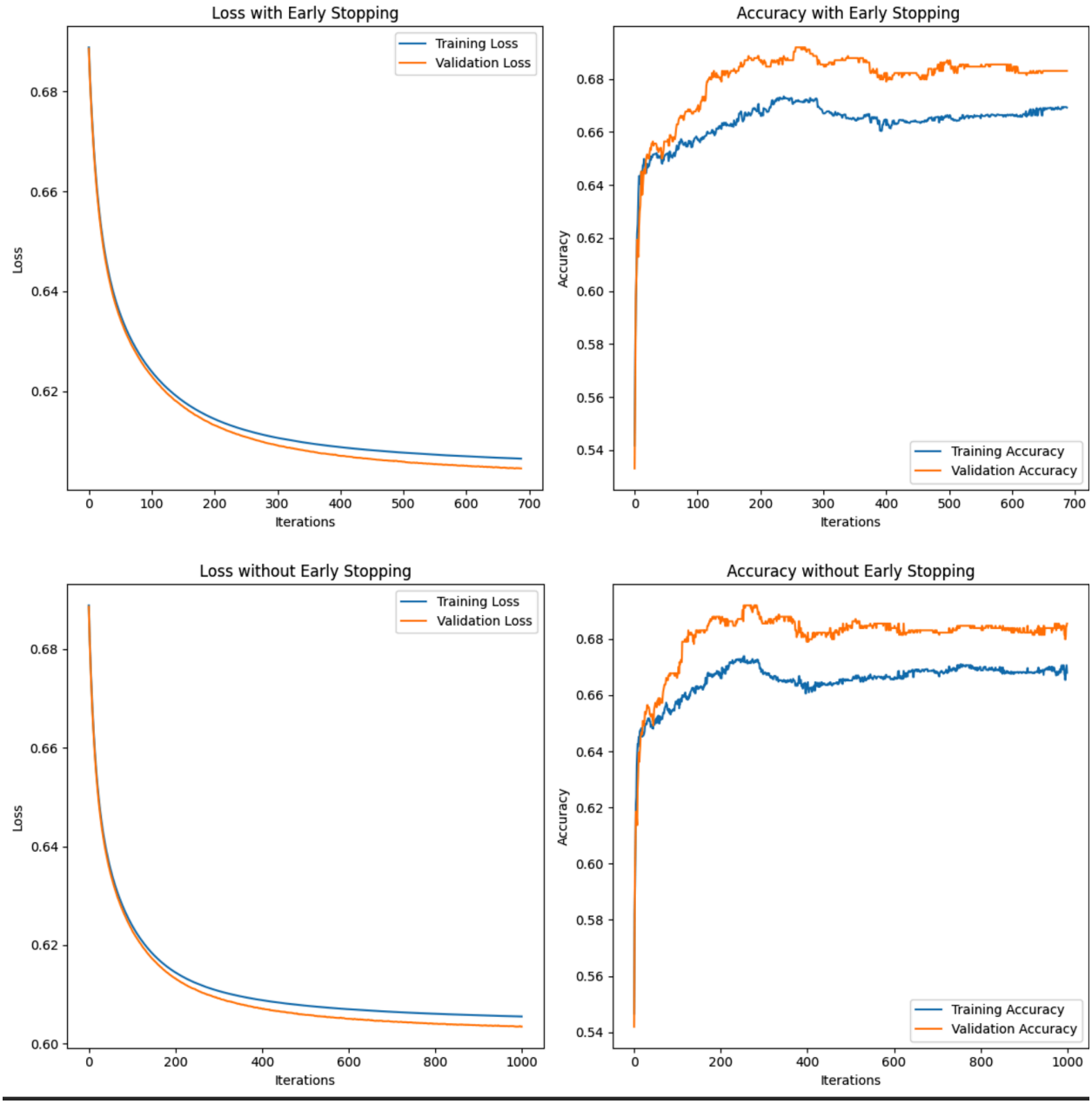


Plots showing effect of L1 and L2 regularization(ElasticNet regularization)

Plot Analysis:

- The L1 and L2 case performs best as compared to using other regularization techniques as it shows the lowest loss and highest accuracy.
- The L1 regularization technique shows poor performance out of three. L2 regularization technique shows moderate performance.
- There is not much effect on the training loss and validation loss as both converges after some regular iterations.

Plots with Early Stopping and without Early-stopping:



In Early Stopping case:

- Both the training and validation loss are decreasing steadily. Also, the divergence between training loss and validation loss is also minimal. This suggests that there is no condition of overfitting and our model is showing low variation when trained with different dataset.
- In this case, validation accuracy stabilizes and shows higher performance than the training accuracy. The training accuracy does

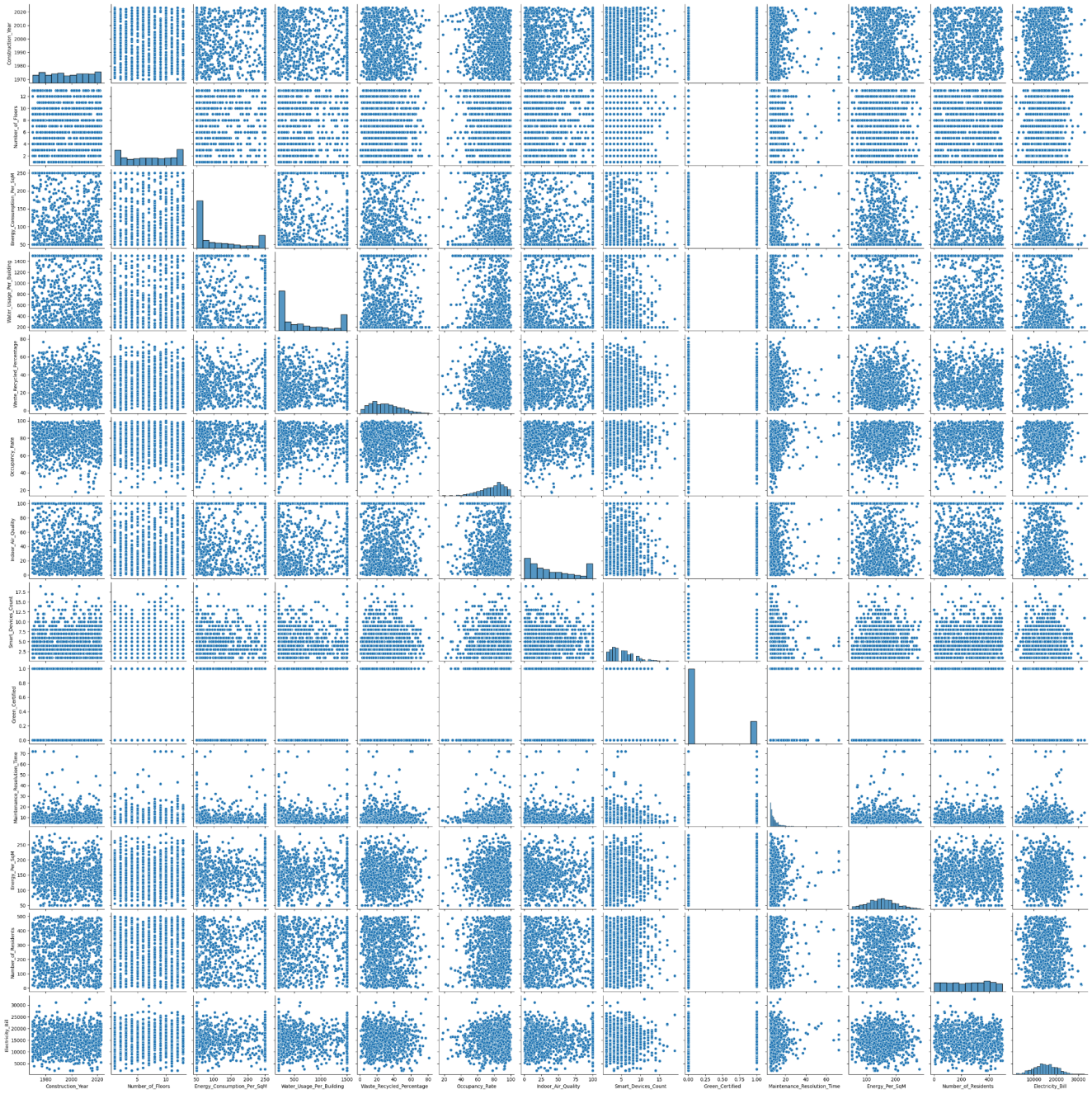
not show significant improvement after some time but doesn't catch up the validation accuracy, which indicates that early stopping has halt training when model is at the optimal point for genearlization.

In No-Early Stopping Case:

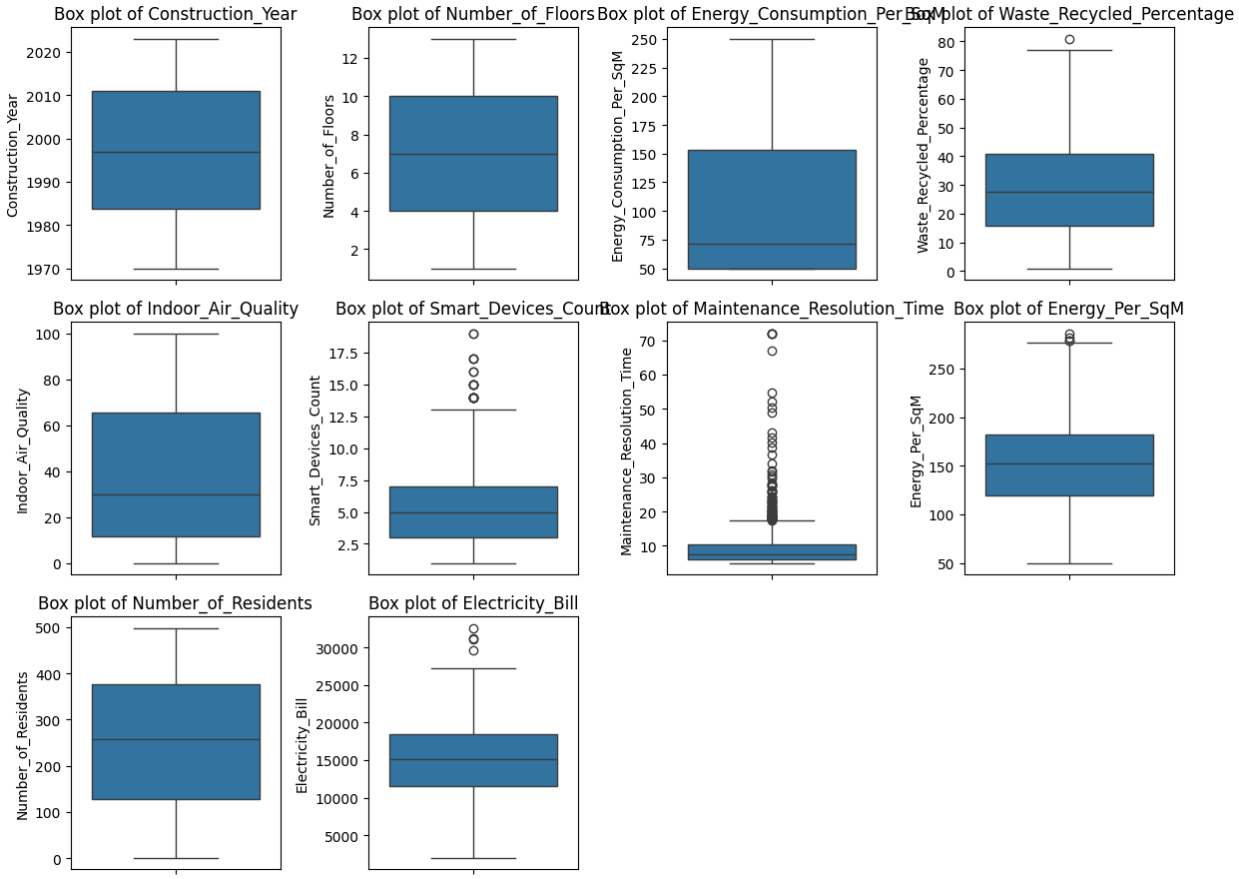
- The training accuracy continues to improve with iterations while validation accuracy stabilizes after some iterations and does not show significant improvement.
- The widened gap between training accuracy and early stopping suggests that model is overfitting to the training dataset without early stopping.
- Thus, model's genearlization to the new data is not that good.

Section C

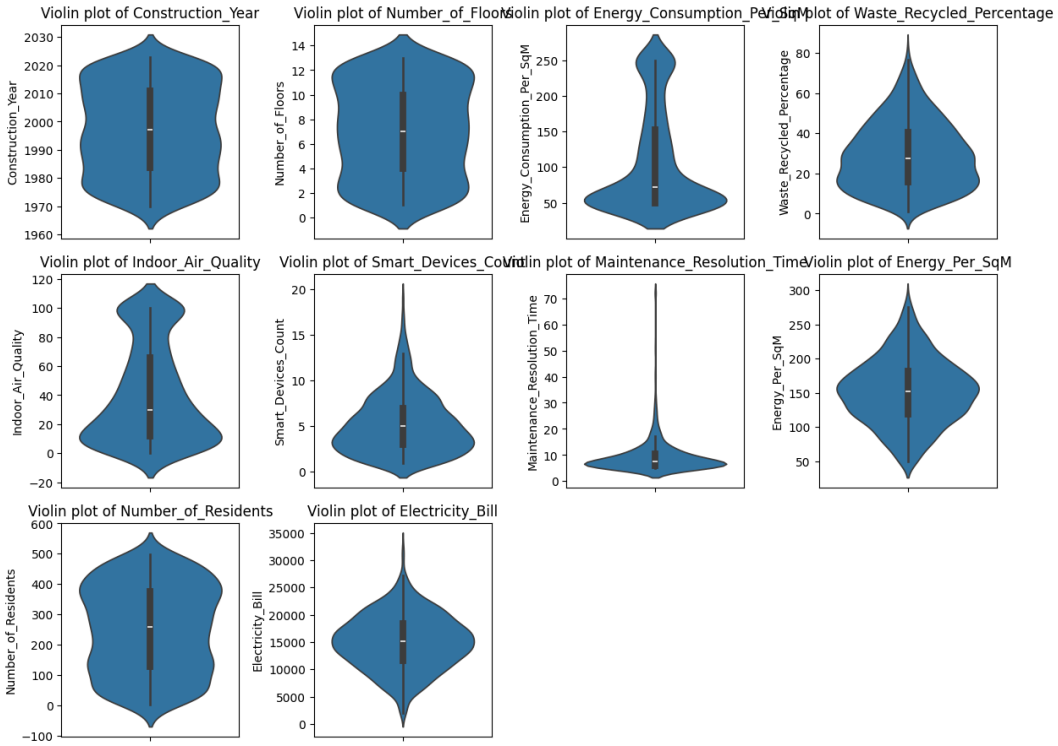
(a) Pair Plot which includes all the features:



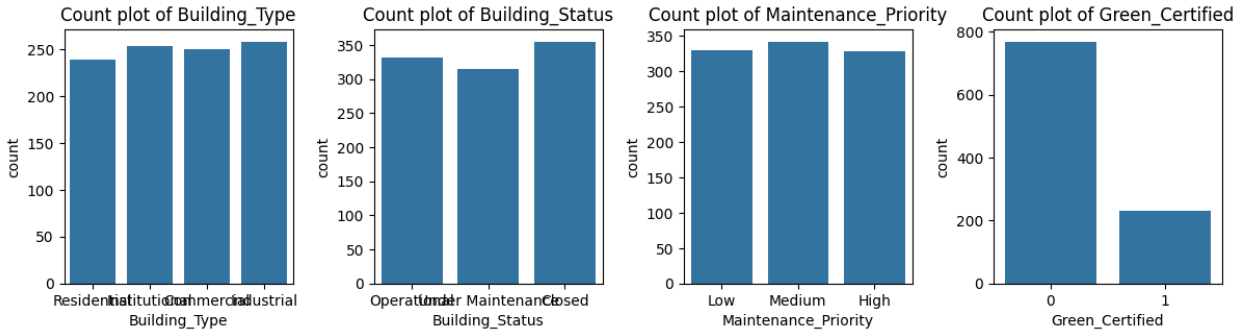
Box Plot for numerical features:



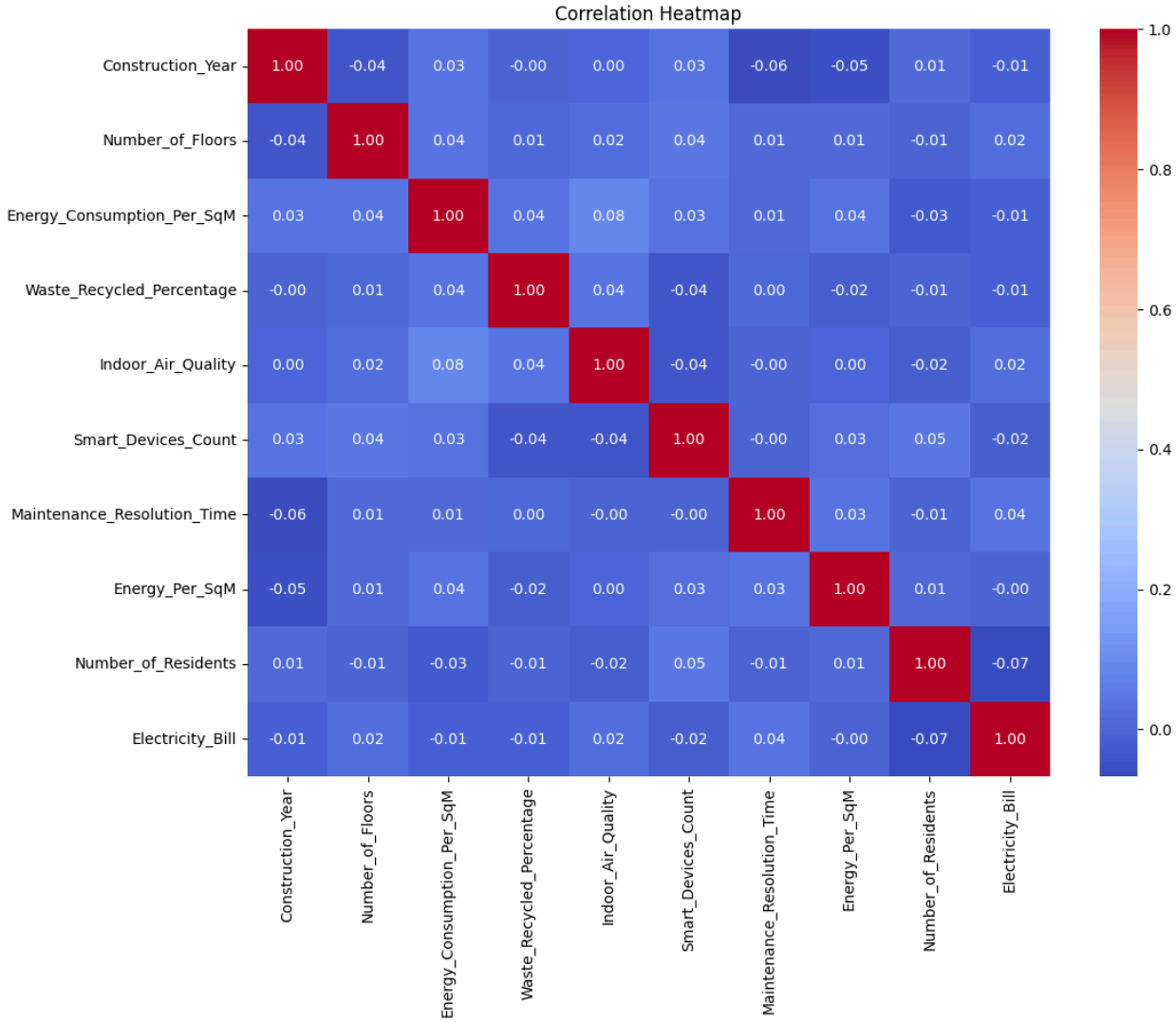
Violin Plot for numerical features:



Count plot for categorical features:



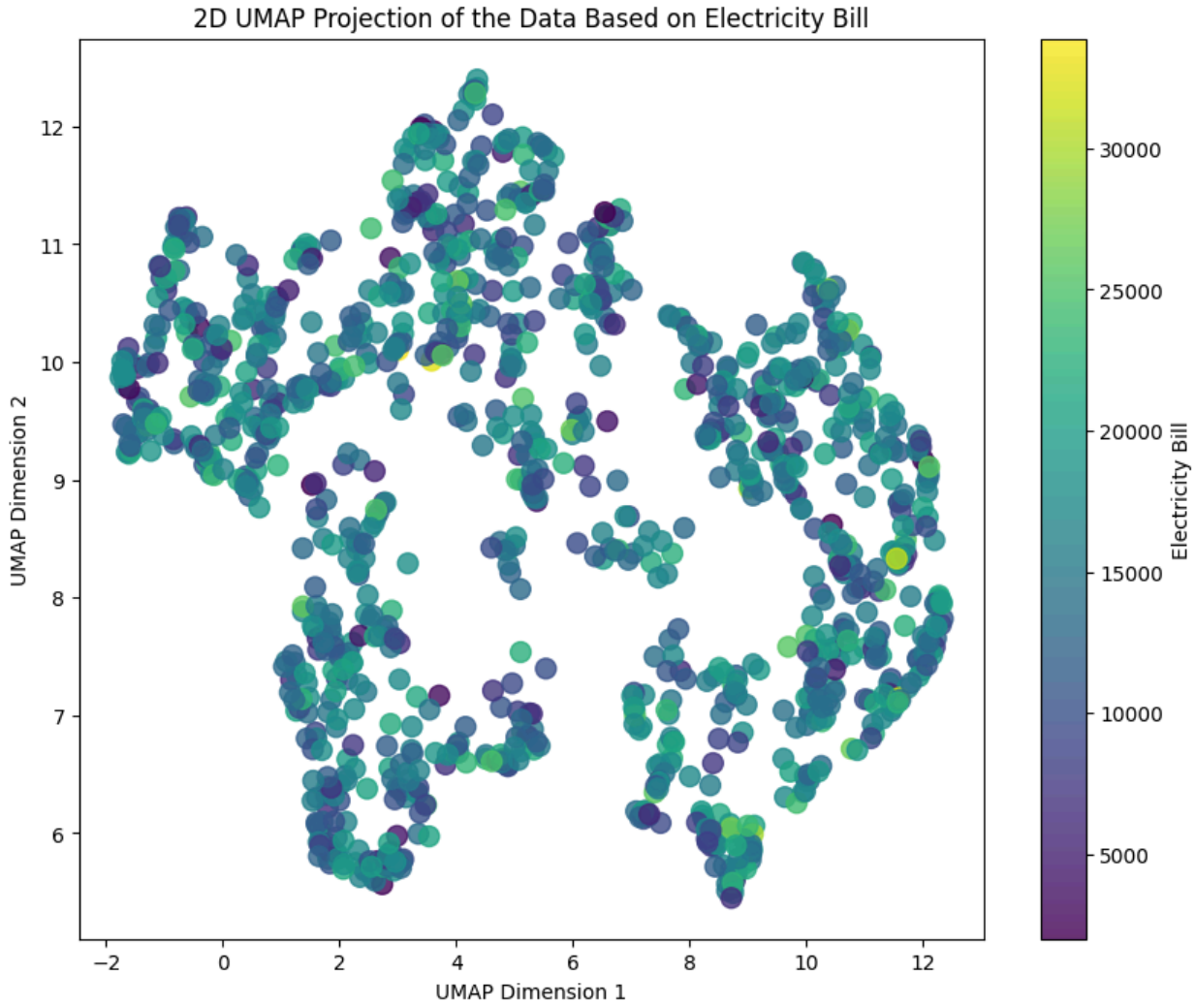
Correlation Heatpmap for numerical features:



Insights:

- From the correlation heatmap, it is evident that most of the variables have weak correlation with values being close to 0. This indicates that many of the characteristics are some way independent of each other or partially dependent.
- From the violin plot and box plot of the numerical variable, we can see that most of the buildings were constructed between the year 1980 and 2000 with the median between 1996 to 2001. There are also few cases where buildings are found to have construction years lesser than 1980 and greater than 2015.
- The count plot shows that there is fairly even distribution among Residential, Institutional, Commercial, and Industrial building types. For Building Status variable, most buildings are either "Operational" or "Closed," with fewer under maintenance.
- The feature named Energy_Consumption_Per_SqM and Energy_Per_SqM plots show a variety of range of values, suggests that significant variability in energy efficiency across different buildings. The majority of buildings consume energy within 50-200 units per square meter, but there are some outliers with much higher consumption.
- Indoor_Air_Quality shows a bimodal distribution in the violin plot, indicating that two distinct groups of buildings with different air quality levels.

(b)



Observations that can be made from the Umap representation:

- Although there are certain areas in the umap which are dense and highly populated, the data doesn't form very distinct and well separated clusters
- Some areas exhibits dense clusters indicating that certain groups of data share similar electricity bills.
- Separation between groups based on electricity bill is not sharp.

(c)

```
Train Data:
MSE :- 24553562.280891553
RMSE :- 4955.155121778888
MAE :- 4000.0400072070274
R2 Score :- 0.010757843864542038
Adjusted R2 Score :- -0.0012694163924240 61

Test Data:
MSE :- 24505841.579286285
RMSE :- 4950.337521754076
MAE -: 3850.786546806548
R2 Score :- -0.009346326239157099
Adjusted R2 Score :- -0.060452469339873804
```

Analysis:

- From the high MSE, RMSE, MAE, it can be said that model appears to be overfitting the training data.
- The negative adjusted R^2 and the poor generalization to the testing data indicate that adding more features has only increased complexity without improving predictive performance.
- The model fails to genearlize to the new and unseen data.
- The higher error of prediction on both training and testing dataset, suggests that odel is not genralizing well to the unseen data and is not robust.

(d) After performing Recursive Featuer Eliminiation, the top 3 features that this algorithm selected are : 'Number_of_Floors', 'Smart_Devices_Count' and 'Maintenance_Resolution_Time'.

After training the model we got the following results.

```
Top 3 features selected by RFE: ['Number_of_Floors', 'Smart_Devices_Count', 'Maintenance_Resolution_Time']
RFE - Train Data:
MSE: 24757308.526249778
RMSE: 4975.671665840681
MAE: 4013.8708515415333
R2 Score: 0.0025490808037321733
Adjusted R2 Score: -0.00045528943481087936

RFE - Test Data:
MSE: 24619736.33685434
RMSE: 4961.827922938716
MAE: 3867.891642295807
R2 Score: -0.014037422227730856
Adjusted R2 Score: -0.026403732254898227
```

Analysis:

- Both the RFE model and general model performs poorly on the unseen data in terms of their predictive power as it was also indicated by the negative R2 score.
- Despite selecting only three features, the rfe model performs as poorly as original model indicating that, reducing the number of features has no significant impact over model's performance.
- It can also be interpreted as reducin the number of features also does not degrade system's performance and may mitigate overfitting.
- Both models have negative adjusted R^2 scores, but the original model's score is worse on the testing data. This indicates that the original model may be overfitting the training data by incorporating unnecessary features.

(e) After encoding the features using OneHot encoding and training it on Ridge Regression model, the evaluation metrics for the training and testing sets are as follows:

```
Training Set Evaluation Metrics:
MSE :- 24188936.71529504
RMSE. :- 4918.224955743183
MAE :- 3976.694459809908
R² Score:- 0.0254482980058343 7
Adjusted R² Score --: 0.0065539282732943 74

Testing Set Evaluation Metrics:
MSE :- 24129257.316123597
RMSE. :- 4912.154040349671
MAE :--  3797.611567420423
R²: 0.0061644221394673 29
Adjusted R² -: -0.0759350386403159 2
```

Analysis:

- The MSE in this case is slightly lower than that of in part (c), however it is still very large but it indicates a marginal improvement in error reduction.
- There is a low improvement in RMSE, which further supports the conclusion that Ridge Regression improves the training performance slightly.
- MAE has also decreased meaning the average absolute error is smaller.
- he R^2 score for Ridge Regression is higher than the original model, suggesting Ridge explains more variance in the target variable. However, both models still explain very little of the variability, as both R^2 values are low.
- Adjusted r2 Score indicates that Ridge Regression does a better job of balancing model complexity with performance, and adding features did not negatively affect the model.

These results are similar in Testing data as well such as :

- Slight improvement in MSE in the ridge regression model on the unseen data.

- Lower RMSE in ridge regression model indicating that that the magnitude of prediction errors has reduced slightly.
- Smaller MAE in ridge regression model indicating that ridge makes fewer error as compared to original model.
- The R^2 score for Ridge Regression is slightly positive, indicating the model explains some of the variance in the test data.
- The adjusted R^2 for Ridge Regression is negative, indicating that although Ridge has improved performance, it still doesn't generalize well to the test data.

(f) After performing ICA(Independent Component Analysis), we got the following results:

Training Set Evaluation Metrics:
MSE : 24701058.76240827
RMSE. : 4970.015972047602
MAE : 4010.983900100582
R^2. : 0.004815335982028102
Adjusted R^2. : 0.000814593614116621

Testing Set Evaluation Metrics:
MSE. : 24167219.341959227
RMSE : 4916.01661327128
MAE : 3818.8935324450963
R^2. : 0.004600842648029624
Adjusted R^2. : -0.011650572165880169

One of the results for ICA with components:
Training Set Evaluation Metrics:
MSE : 24683781.380049054
RMSE. : 4968.277506344533
MAE : 4008.435901682255
R^2. : 0.005511427033171 09
Adjusted R^2. : 0.0005089694226739772

Testing Set Evaluation Metrics:
MSE. : 24261490.340966236
RMSE : 4925.595430094339
MAE : 3831.4952084897286
R^2. : 0.0007180098054699879
Adjusted R^2. : -0.019759080157532694

We have performed **Independent Component Analysis (ICA)** on the one-hot encoded dataset with different numbers of components (4, 5, 6, and 8).

Analysis on Training Data:

- Across all the components, the relative difference between MSE, MAE and RMSE are minimal. These indicates that increasing the number of components does not necessarily change the training error metrics.
- Analysing R2 score and Adjusted R2 score all the components indicates that:
    - R2 score shows a slight improvement with the increase in the number of components.
    - While Adjusted R2 score begins with a positive values while worsen as the number of components increased.

Analysis on Testing Data:

- Again, the marginal difference between MSE, MAE and RMSE are minimal.The MSE values range between **24.16M** to **24.26M**, and the RMSE fluctuates around **4,916 to 4,926**. The MAE values are slightly lower than the training set, ranging from **3,818 to 3,831**, but the differences between the component sizes are minimal.
- Analysing R2 score and Adjusted R2 score all the components indicates that:
    - The **R^2** for the test data starts at **0.0046** for 4 components and drops slightly to **0.0007** with 5 components, then increases slightly for 6 components (**0.0010**) and 8 components (**0.0023**). While the differences are small, the higher component models explain slightly more variance.

    - However, the **Adjusted R²** declines across the board, from **-0.0117** (4 components) to **-0.0308** (8 components). This consistent decline indicates that the model is not benefiting from the additional complexity, as it struggles to generalize well to unseen data.

(g)

ElasticNet regularization combines L1 and L2 regularizaton terms, controlling the model complexity and reducing overfitting.
The mixing parameter α\alphaα adjusts the contribution of the L1 and L2 regularizationns.

In your case, you have trained a linear model using ElasticNet regularization on the preprocessed dataset from part (c) with different values of α and obtained the following results:

Training Set analysis:

- **MSE, RMSE and MAE:**
    - With increase in α, there is a slight improvement in the error metrics MSE and RMSE which indicates that error on the training set rises as the regularization penalty increases.
    - The MAE also increases with the rise of the value of α.
- **R2 score and Adusted R2 score:**
    - R2 score decreases as the α increases.
    - The **Adjusted R²** follows a similar trend, becoming more negative as α increases, showing that the additional regularization is reducing the model's ability to capture the complexity in the training data.

Testing Set Analysis:

- **MSE, RMSE and MAE:**
    - With increase in α, there is a slight improvement in the error metrics MSE and RMSE in the testing set as well which indicates that error on the testing set rises as the regularization penalty increases.

    - The MAE values also decrease as α\alphaα increases, from 3,849.36 to 3,840.45, indicating that the model's absolute prediction error on unseen data is slightly lower with stronger regularization.
- **R2 score and Adusted R2 score:**
    - The R2 scores on the test data are negative for all α\alphaα values, indicating that the model fails to explain the variance in the target variable for the test set.
    - However, with the increase of α, R2 also increases.
    - The Adjusted R2 is less negative with higher α\alphaα, suggesting that regularization helps reduce the model complexity, resulting in a modest improvement in generalization, though it is still not sufficient to produce a good fit.

(h) After using the Gradient Booster Regressor on the preprocessed data from part c, we get the following results:

Results for alpha=0.1:
Training Set Metrics:
MSE: 24554128.57851768
RMSE: 4955.212263719656
MAE: 3999.4968395226915
R^2: 0.0107350282144838
Adjusted R^2: -0.0012925094364040213

Testing Set Metrics:
MSE: 24487397.515020184
RMSE: 4948.474261327443
MAE: 3849.3580247371474
R^2: -0.00858665232844169
Adjusted R^2: -0.05965433092735006

Results for alpha=0.5:
Training Set Metrics:
MSE: 24563612.6567632
RMSE: 4956.169151346955
MAE: 3998.305110968403
R^2: 0.010352922762524197
Adjusted R^2: -0.0016792605473539002

Testing Set Metrics:
MSE: 24436187.861386012
RMSE: 4943.297266135834
MAE: 3845.030055312522
R^2: -0.006477429692834713
Adjusted R^2: -0.05743831220892748

Results for alpha=1.0:
Training Set Metrics:
MSE: 24581685.011365093
RMSE: 4957.992034217592
MAE: 3997.8401867057414
R^2: 0.009624802955373712
Adjusted R^2: -0.002416232874955959

Testing Set Metrics:
MSE: 24401051.58689911
RMSE: 4939.742056717042
MAE: 3842.872482346764
R^2: -0.005030237216039124
Adjusted R^2: -0.055917844163686725

Results for alpha=5.0:
Training Set Metrics:
MSE: 24686754.953174766
RMSE: 4968.5767532739965
MAE: 4000.755595060608

R^2: 0.0053916242991867 15
Adjusted R^2: -0.006700878748847394

Testing Set Metrics:
MSE: 24350934.24432863
RMSE: 4934.666578840826
MAE: 3839.7581683960407
R^2: -0.0029660046761885717
Adjusted R^2: -0.05374909352055246

Results for alpha=10.0:
Training Set Metrics:
MSE: 24737187.09889484
RMSE: 4973.6492738124225
MAE: 4003.9122352133804
R^2: 0.003359756010588688
Adjusted R^2: -0.008757450603264383

Testing Set Metrics:
MSE: 24351152.756909396
RMSE: 4934.688719352964
MAE: 3840.4520373604178
R^2: -0.00297500476989776
Adjusted R^2: -0.05375854931520907

Observations and Comparisons:
On Training Data:
The Gradient Boosting Regressor clearly outperforms both the linear regression model and the ElasticNet model on the training set:

- MSE, RMSE and MAE are lower in Gradient Boosting Regressor model as compared to linear model and elasticnet model.
- R2 for Gradient Boosting (0.4758) is much higher, indicating that it explains about 47.6% of the variance in the training data, compared to 0.0000 for linear regression and 0.0034 for ElasticNet.

On Testin Data:

- Similar trend is shown in MSE as it is lower in Gradient Boosting Regressor as compared to both the linear regression model and Elastic Net model.
- RMSE is also lower as compared to other models.
- MAE for Gradient Boosting is slightly lower compared to both **l**inear regression and ElasticNet.
- R2 for Gradient Boosting is 0.0375, a significant improvement compared to the negative R^2 scores for both linear regression (-0.0093) and ElasticNet (-0.0030). This indicates that Gradient Boosting performs better in capturing the variance of the test set data, although it still has room for improvement.