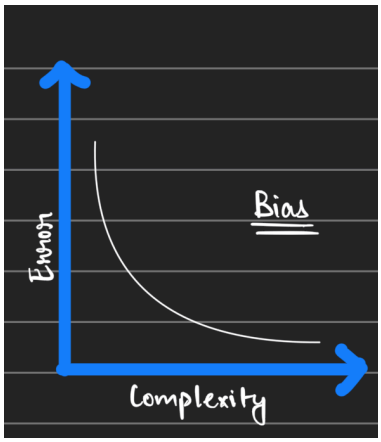
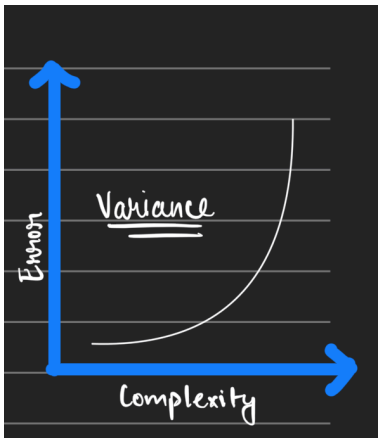
ML Assignment 1

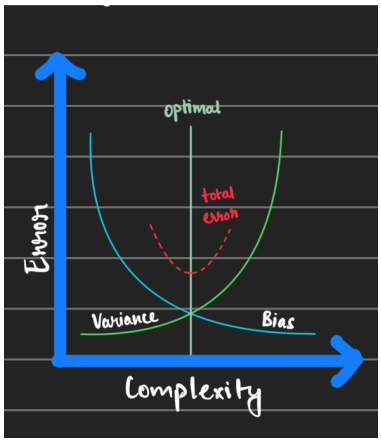
Monsoon 2024
Dr. Jainendra Shukla
Rahul Oberoi
2021555

Section A:

Ans 1. As we increase the complexity of the model the bias of the model decreases and the variance of the model increases. Therefore, the model is much more sensitive to outliers and changes predictions even when there are slight changes in the features. Hence, the total error of the model increases and the model overfits.







Ans 2.
Spam emails identified as spam or True Positives (TP) = 200
Spam emails identified as legitimate or False Positives (FN) = 50
Legitimate emails identified as legit or True Negatives (TN) = 730
Legitimate emails identified as spam or False Negatives (FP) = 20

Precision = $\frac{TP}{TP+FP} = \frac{200}{200+20} = \frac{200}{220}$ = **0.909**

Recall = $\frac{TP}{TP+FN} = \frac{200}{200+50} = \frac{200}{250}$ = **0.8**

Specificity = $\frac{TN}{TN+FP} = \frac{730}{730+20} = \frac{730}{750}$ = **0.9733**

F1 Score = $\frac{2 * Precision * Recall}{Precision + Recall} = \frac{2 * 0.909 * 0.8}{0.909 + 0.8} = \frac{1.4544}{1.709}$ = **0.851**

Accuracy = $\frac{TP + TN}{TP+TN+FP+FN} = \frac{(200 + 730)}{200+730+20+50} = \frac{930}{1000}$ = **0.93**

Ans 3.

	x	y	x^2	xy
	3	15	9	45
	6	30	36	180
	10	55	100	550
	15	85	225	1275
	18	100	324	1800
Σ	52	285	694	3850
Mean	10.4	57	138.8	770
	$\bar{x}$	$\bar{y}$	$\bar{x}^2$	$\overline{xy}$

$$y = mx + c$$

$$m = \frac{(\overline{xy}) - (\bar{x})(\bar{y})}{\bar{x}^2 - (\bar{x})^2} = \frac{770 - (10.4)(57)}{138.8 - 108.16}$$

$$\Rightarrow \frac{177.2}{30.64} = 5.78$$

$$c = \bar{y} - m\bar{x} = 57 - 5.78\,(10.4)$$

$$\Rightarrow 57 - 60.11 = -3.11$$

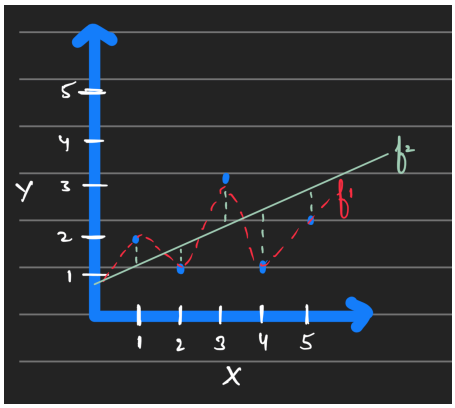
$$y = 5.78x - 3.11$$

$$x = 12 \Rightarrow 5.78\,(12) - 3.11$$

$$= 66.25$$

Ans 4. Let the following dataset be the sample points. The two models “f1” and “f2”, where “f1“ is a higher order polynomial curve and “f2” is a linear curve. The f1 curve fits the data almost perfectly and will give extremely good performance for the training set but might struggle with newer unseen data hence it is **not generalizable.** On the other hand, the f2 linear model may give worse performance on the training set but is much **more generalizable** for unseen data.

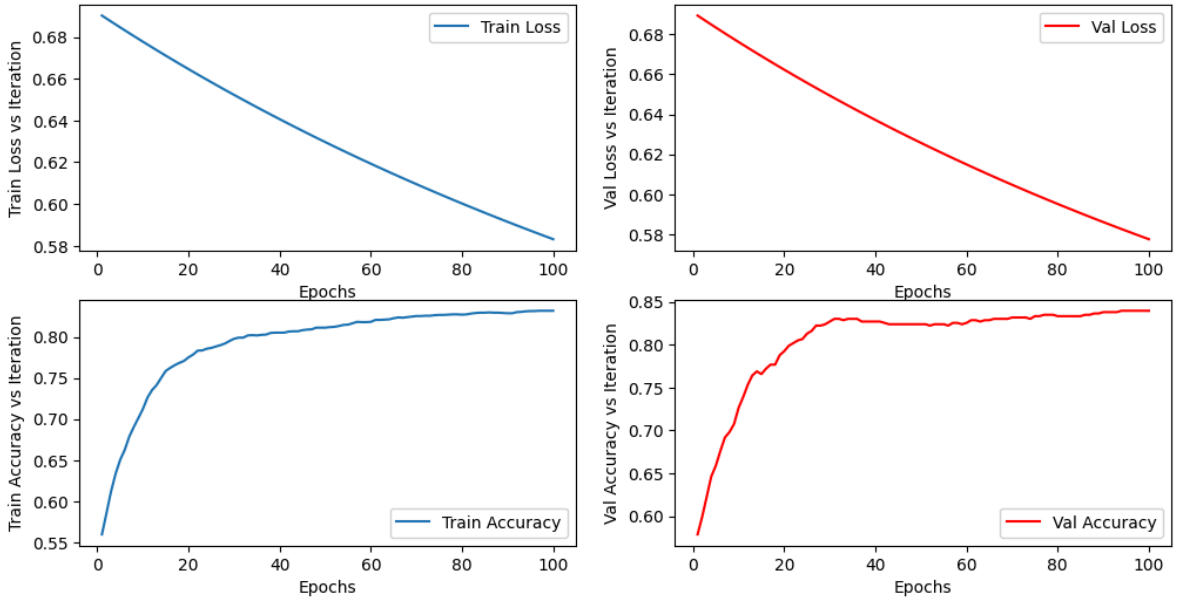
X	Y
1	2
2	1
3	3
4	1
5	2.5



Section B:

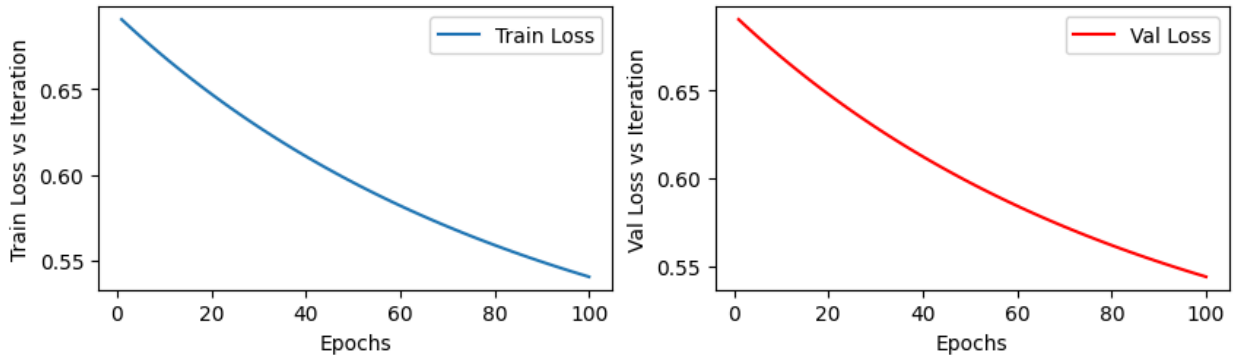
Note: There is a significant imbalance in the dataset and class 0 has 3594 entries(~84%) and class 1 has 644 entries (~16%).

Part a. The model converges smoothly overall as evident from the consistent increase in the accuracies and smooth loss curves. Additionally, the gap between train loss and val loss is minimal suggesting low overfitting and good generalization. The validation accuracy is slightly higher than the training accuracy, the reason for it could be that the data samples in the validation split are present in the training set and there are fewer unseen data samples which helps the model to make correct predictions. From the graphs, it is clear that the improvement in the accuracies slows down around the 30th epoch mark which is likely to happen as the model approaches minima.



Part b. After applying min-max scaling I am able to reduce the losses even further and increase the accuracies a bit, from **83.81%** in the val set to **84.59%.** Since, there is a significant imbalance in the dataset the model is more likely to predict 0 and scaling makes the decision boundary even more sensitive to small differences.

The plots <u>without scaling</u> are in **Part a**



Part c. As mentioned in **Part b** the dataset imbalance leads to high accuracies as the model just predicts 0 for majority of the entries but low F1 scores as the model isn't able to distinguish properly between the 2 classes. After applying min-max scaling the decision boundary becomes more sensitive which reduces the f1 score to 0 which was already pretty low (0.14) as the model just predicts 0 for the majority of the prediction. The model becomes worse after scaling is clearly evident from ROC-AUC.

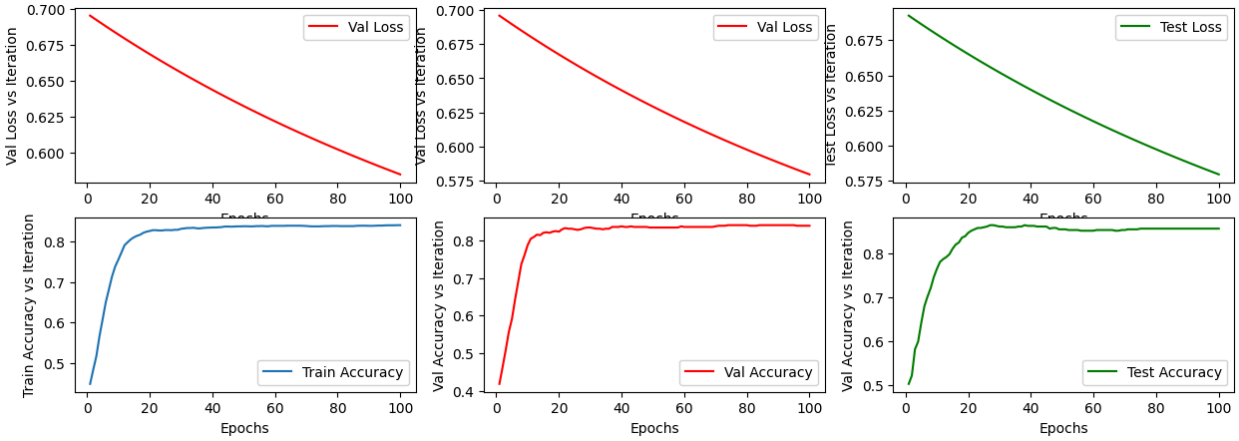
	Scores <u>before</u> scaling	**Scores <u>after</u> scaling**
Confusion Matrix	[522 16 89 09]	[538 0 98 0]
F1 Score	0.14634146341463417	0.0
Precision	0.36	0.0
Recall	0.09183673469387756	0.0
ROC-AUC	0.7291935361505197	0.31763523253167436

Part d.

Model	**Train Loss**	**Train Acc.**	**Val Loss**	**Val Acc.**	**Test Loss**	**Test Acc.**
Full Batch	0.5922	0.8392	0.5874	0.8412	0.5874	0.8553
Stochastic	0.5955	0.7829	0.5942	0.7814	0.6120	0.8396
Batch = 16	0.5921	0.8328	0.5862	0.8412	0.5808	0.8443
Batch = 4	0.5878	0.7930	0.5829	0.8003	0.5842	0.7925

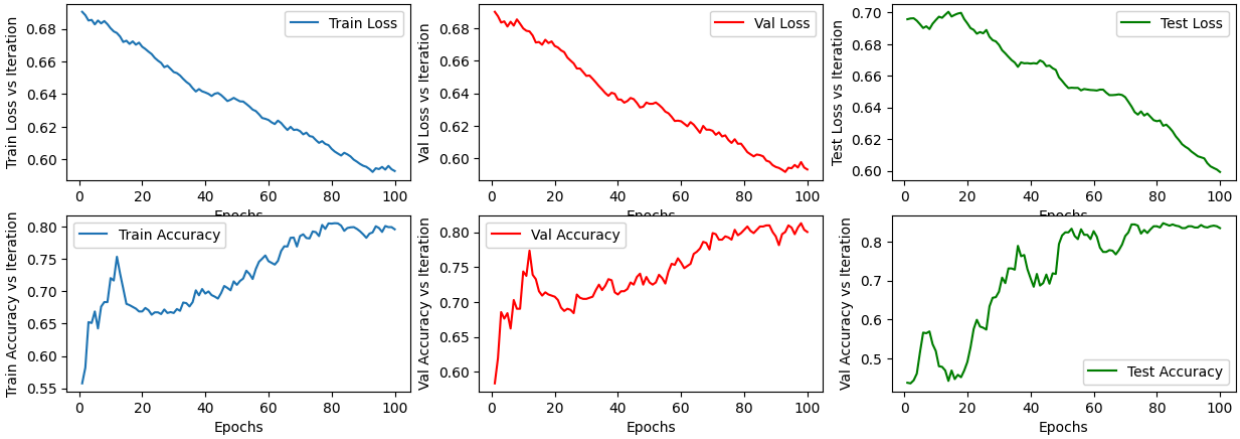
Full Batch Gradient Descent:

It converges the slowest but is the most stable. This is evident from the smooth loss and accuracy graphs.



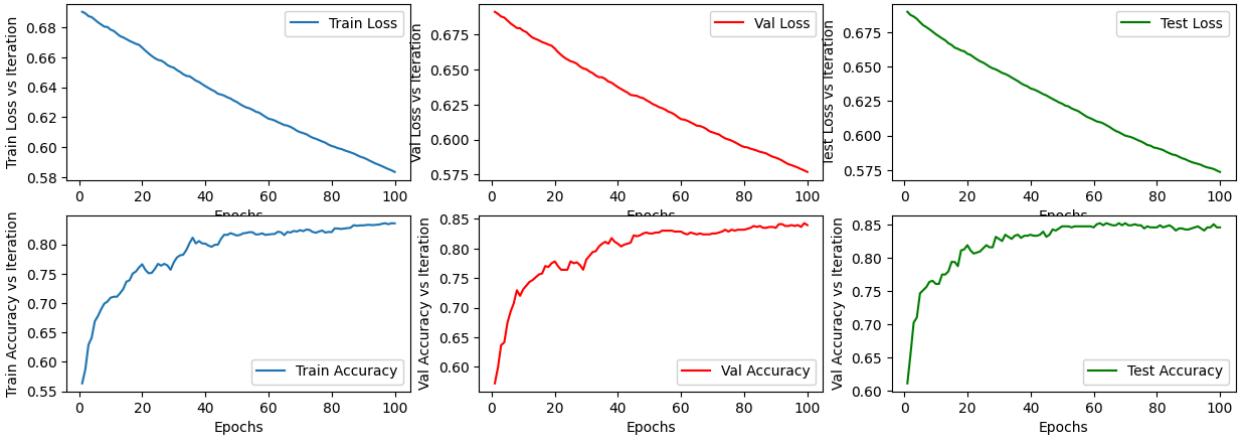
Stochastic Gradient Descent:

It converges the fastest but is the least stable. This is evident from the jagged loss and accuracy graphs.

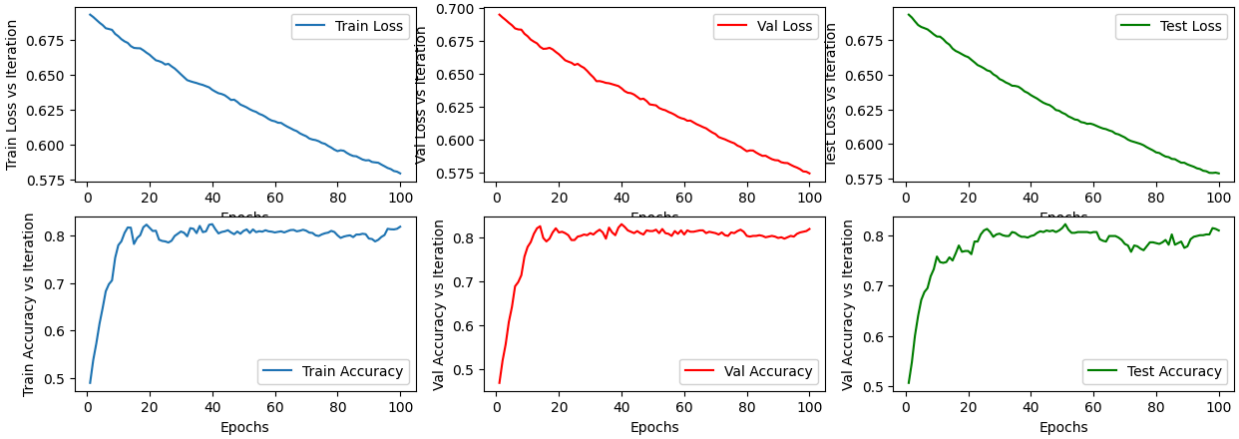


Mini Batch (Batch Size = 16)

It is faster and less stable than full batch gradient descent but slower and more stable than stochastic gradient descent.



Mini Batch (Batch Size = 4)
It is a middle ground between full batch and stochastic gradient descent like the previous one.



Part e.

Fold Number	Train Loss	Train Acc.	Val Loss	Val Acc.
1	0.5883	0.8334	0.5934	0.8153
2	0.5884	0.8390	0.5930	0.8181
3	0.5905	0.8376	0.5902	0.8458
4	0.5908	0.8352	0.5887	0.8389
5	0.5915	0.8319	0.5900	0.8476

	Average	Standard Deviation
Accuracy	0.8323	0.0199
Precision	0.4108	0.0442
Recall	0.1938	0.0389
F1 Score	0.2601	0.0351

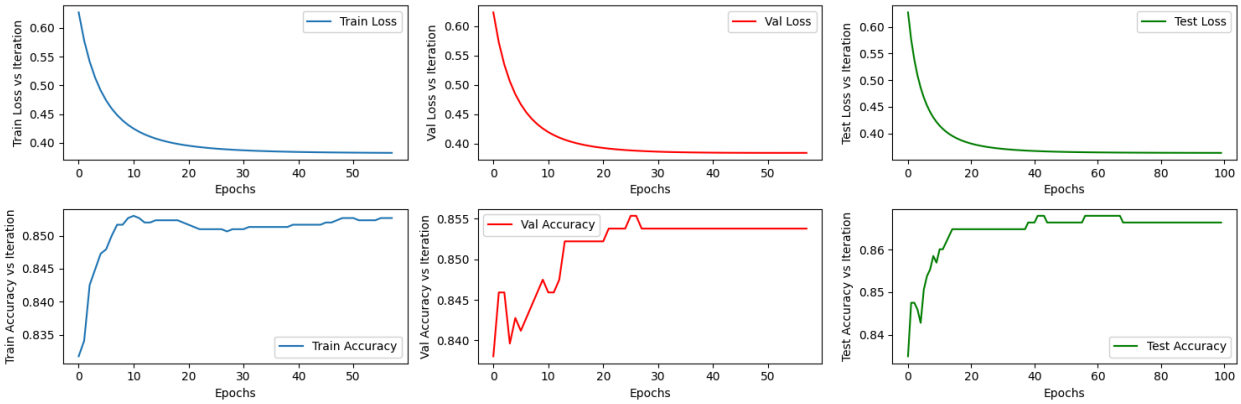
On test set:

Accuracy	0.8428
Precision	0.4038
Recall	0.2333
F1 Score	0.2958

Part f. With early stopping the model stops training at the 57th epoch saving time and compute for the rest of the 43 epochs. However, without early stopping the model keeps training till the 100th epoch (The graphs are above in **Part d**). Early stopping stops whenever the model starts to overfit and generalizes better than an overfitted model (Visible from the accuracy difference on the test set).

Accuracy without early stopping on the test set: **85.53%**
Accuracy with early stopping on the test set: **86.64%**

Therefore, we get better results with less computation.

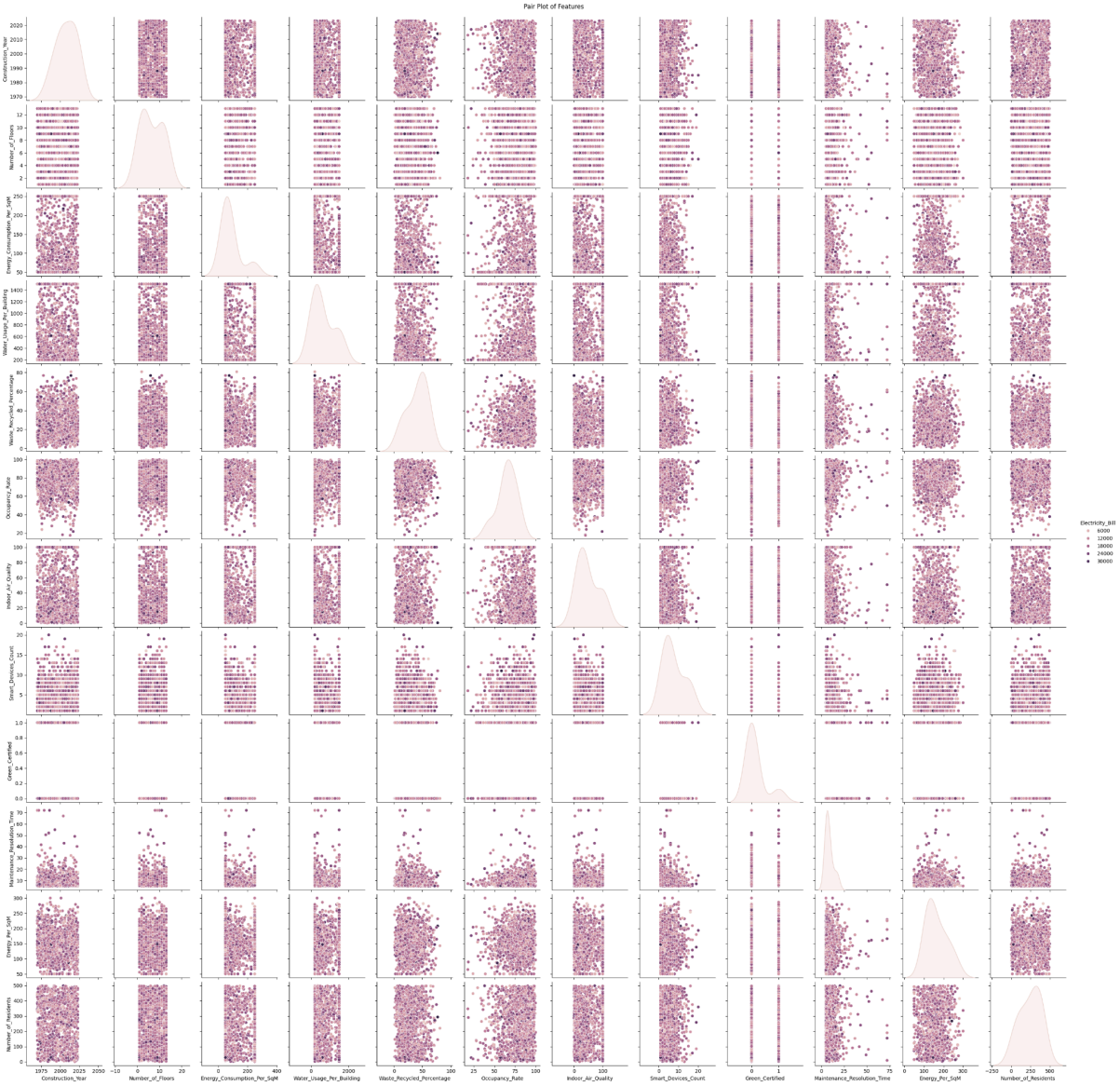


Section C:

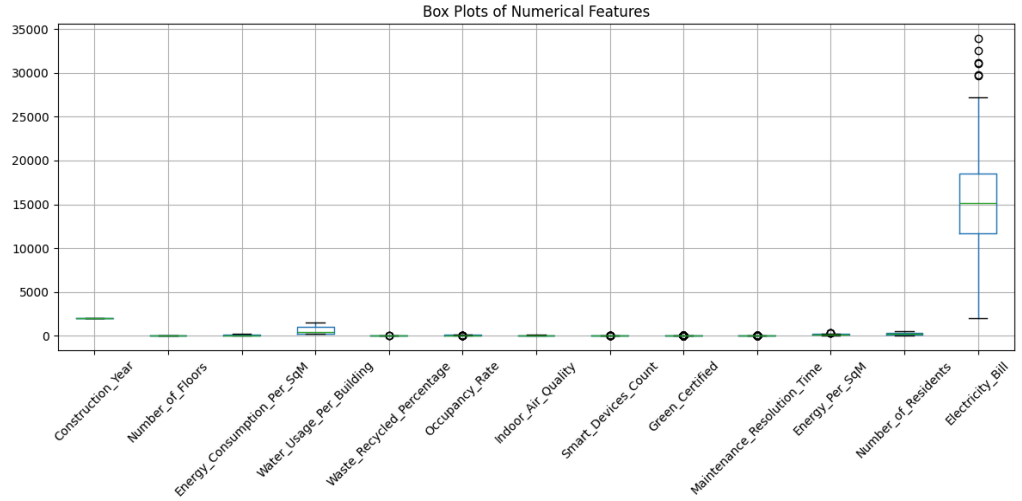
Part a.

1. From the pair plot, we can see that the majority of the features are tightly grouped, with relatively few outliers. This suggests that most of the data points are concentrated within specific ranges for these features, and there is limited variability or deviation from the norm.
2. From the box plot, we can see that there are circles in "*Water_Usage_Per_Building*", "*Water_Recycled_Percentage*", "*Indoor_Air_Quality*", "*Smart_Devices_Count*", "*Green_Certified*" and "*Maintenance_Resolution_Time*", meaning that they contain outliers.
3. Additionally, we can see that the range of "*Electricity Bill*" is very different from the range of the features.
4. From the count plots, it is clear that the "*Building_Type*", "*Building_Status*" and "*Maintenance_Priority*" are evenly distributed but "*Green_Certified*" is skewed towards 0.
5. From the correlation heatmap, we can see that the features aren't very dependent on some other feature.
6. From the violin plot, we can see that most of the features show very narrow distributions. Electricity bill has the widest range.

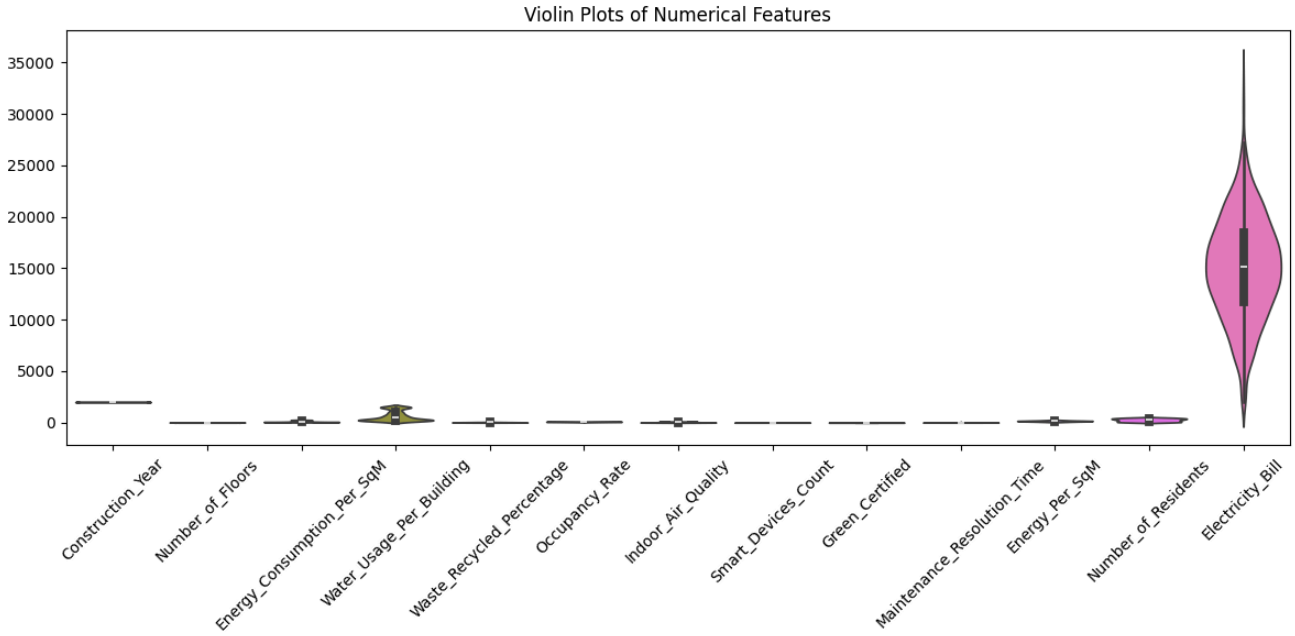
Pair Plot



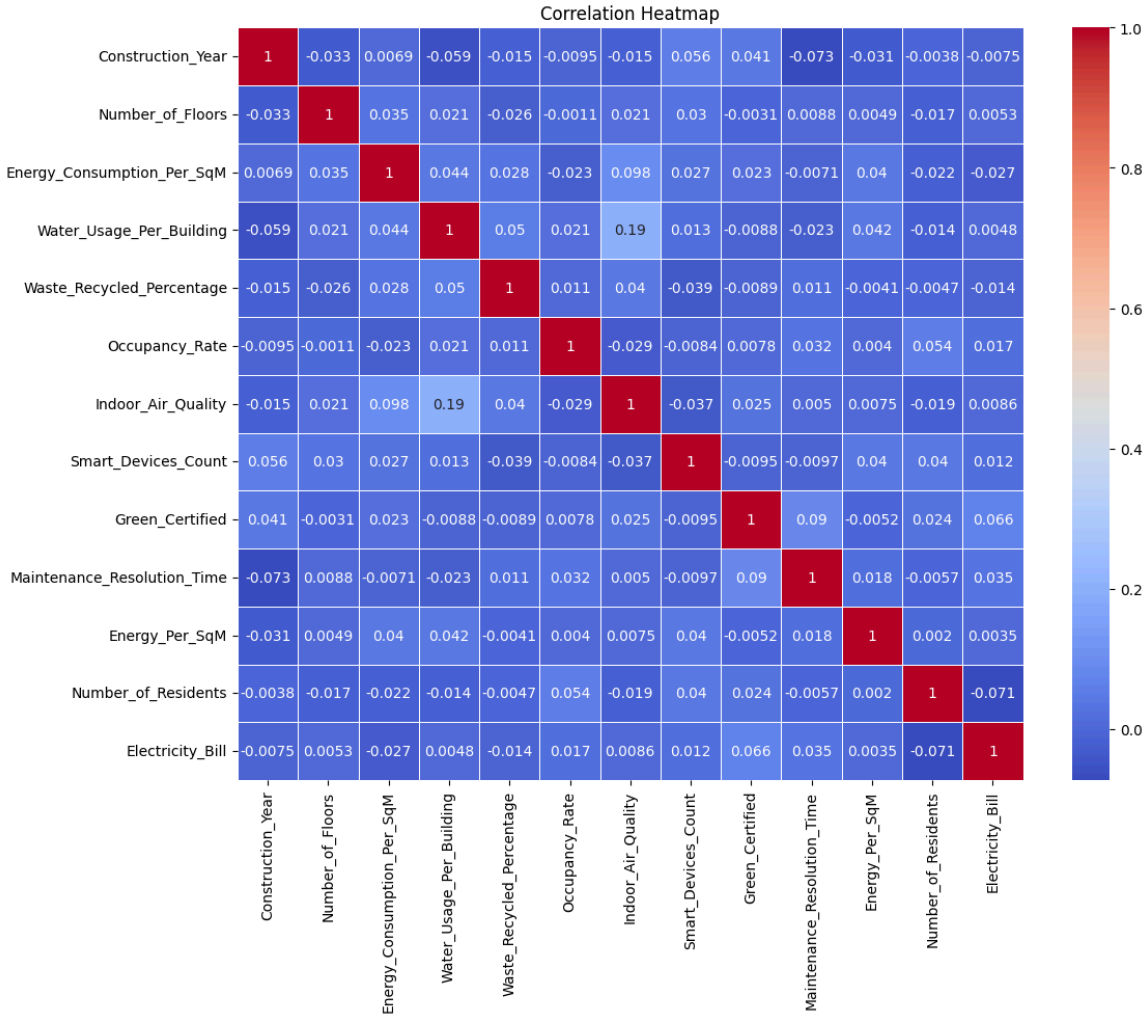
Box Plot



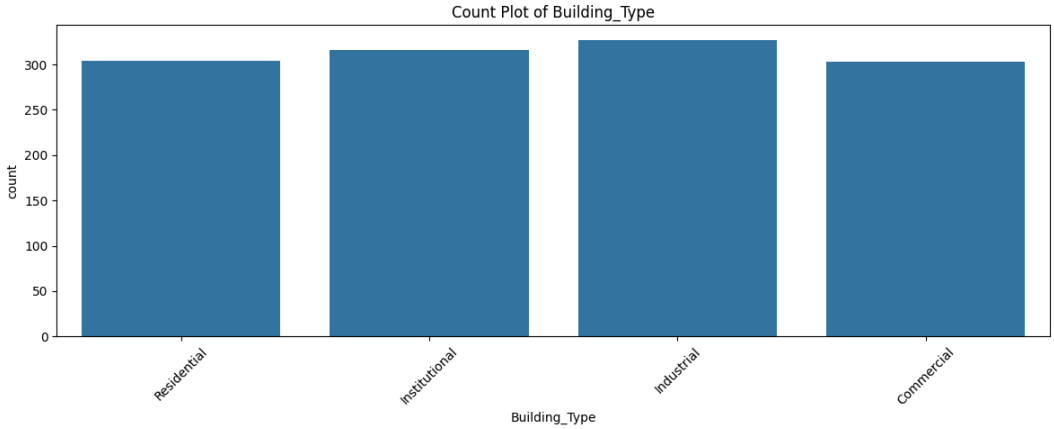
Violin Plot

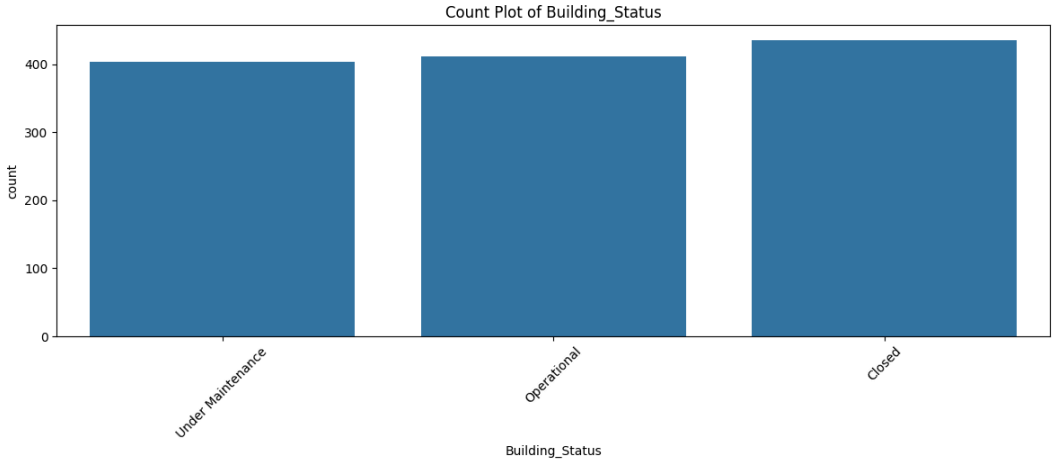


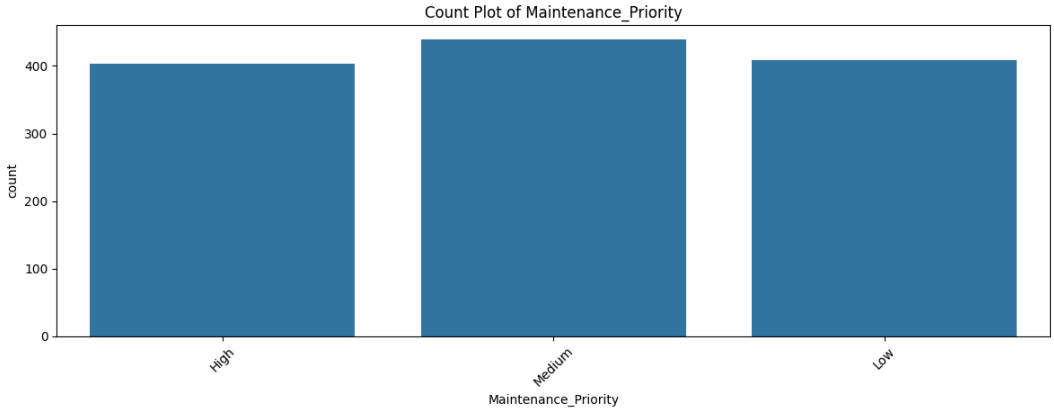
Correlation Heatmap

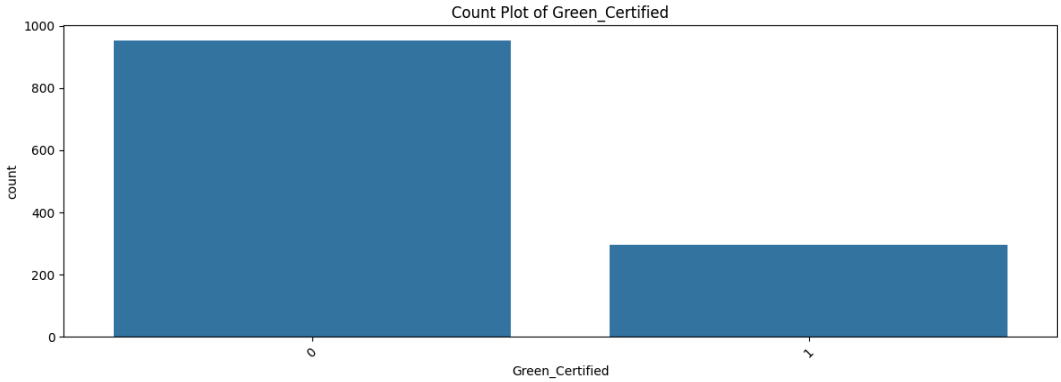


Count Plot

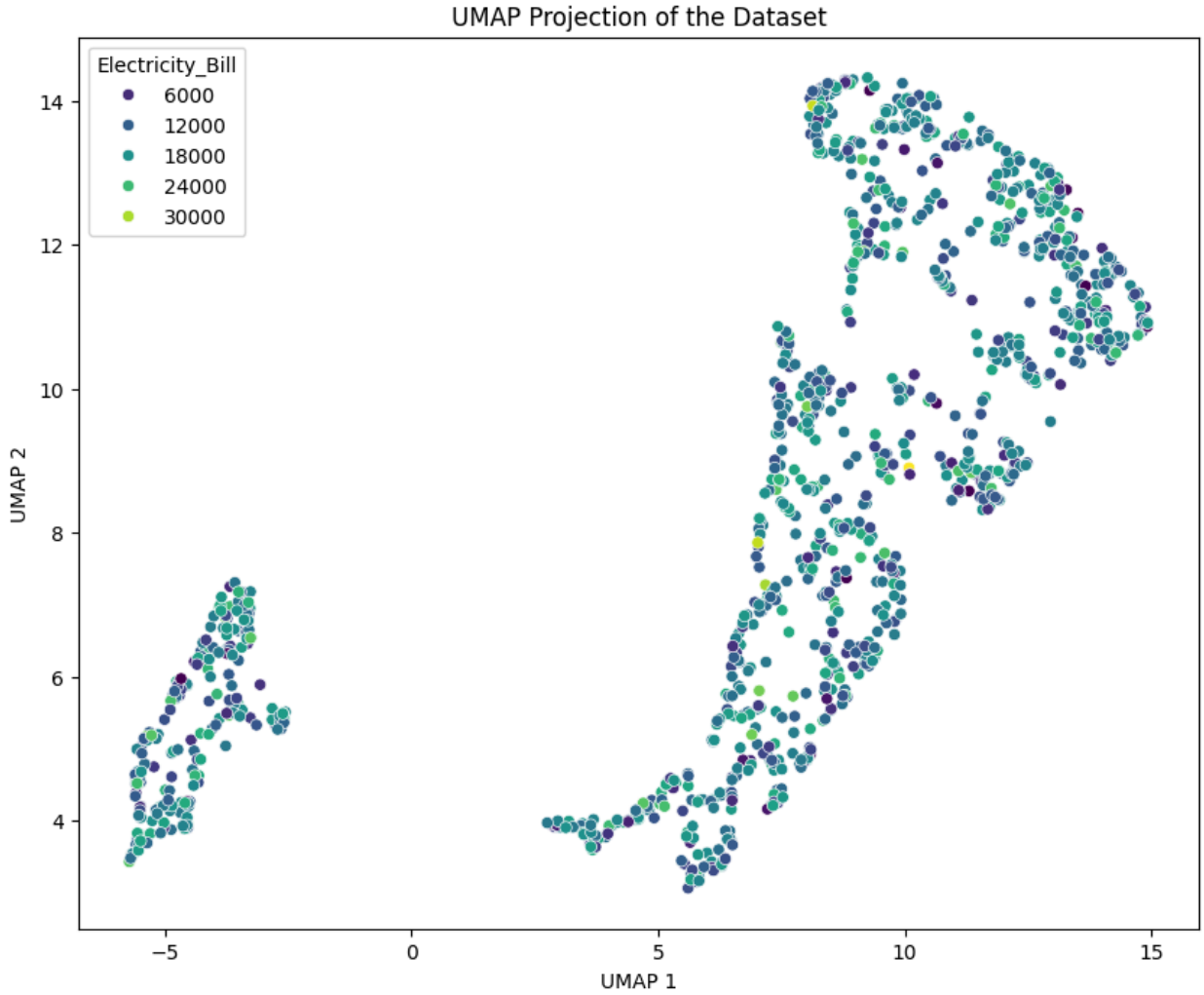

Count Plot of Building_Type
count
0
50
100
150
200
250
300
Residential
Institutional
Industrial
Commercial
Building_Type


Count Plot of Building_Status
count
0
100
200
300
400
Under Maintenance
Operational
Closed
Building_Status


Count Plot of Maintenance_Priority
count
0
100
200
300
400
High
Medium
Low
Maintenance_Priority


Count Plot of Green_Certified
count
0
200
400
600
800
1000
0
1
Green_Certified

Part b. Since string values cannot be plotted on the UMAP, I have label encoded them as we were required to do so in **Part c.** The UMAP splits the data points in two clear clusters.



Part c. As the dataset did not have any NaN values so I did not have to do anything with the missing values. I have min-max scaled the numerical features (except the target variable) and label encoded the categorical features. Upon applying linear regression I get the following results:

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

Part d. After applying RFE and only selecting 3 features we can see that there is a slight improvement in the errors meaning that the rest of the features aren't providing much information to the model and in reality, negatively impacting the performance of the model. In comparison to the previous part the model gives better results.

Metric	Train	Test
MSE	24598921.045604337	23976300.350124482
RMSE	4959.729936761108	4896.560052743607
MAE	4017.1253534034668	3816.722345837314
R2	0.008930377784842625	0.012464411927795571
Adjusted R2	0.005945228320339169	0.00042129500008580845

Part e. One-hot encoding with ridge regression leads to a noticeable improvement in model performance across all metrics when compared to label encoding. This is most likely due to the penalizing term present in ridge regression.

Metric	Train	Test
MSE	24279780.494423226	23294434.679618455
RMSE	4927.451724210317	4826.430842726171
MAE	3981.71251379437	3733.689649935618
R2	0.021788279353217255	0.04054908746473351
Adjusted R2	-0.00023900606564586369	0.05243734458714244

Part f. For n = (4,5,6) the performance on the train set improves however it worsens on the test set. However, for n =8 the performance on both the sets improve and even beats the performance of the model in **Part c** by a small margin.

Dataset	N	MSE	RMSE	MAE	R2	Adjusted R2
Train	4	24723471.40007696	4972.270246082463	3999.4550092249247	0.003912349858938846	-9.202260393981199e-05
Test	4	24640324.962147772	4963.90219103356	3854.391804757433	-0.014885426293822945	-0.03145498427413029
Train	5	24680344.74145633	4967.931636149629	3993.7287932774107	0.005649886281260286	0.000648125145854106
Test	5	24698709.660744872	4969.77963905291	3839.8179581959985	-0.017290174600033999	-0.03813628473559283
Train	6	24616414.970208745	4961.493219808805	3986.3483138005904	0.008225562430680956	0.0022329676417425226
Test	6	24493802.30529969	4949.121367000378	3823.8870147468483	-0.008850452758148775	-0.03376034048057219

Train	8	24414396.16 8039948	4941.09260 8729364	3978.837103 1043093	0.01636472 8273548956	0.008424181 175858192
Test	8	24075459.2 53332824	4906.67496 9195822	3781.821091 0239695	0.00838025 6142187181	-0.02453658 1828196624

Part g. The errors reduce from 0.005 to 0.1 alpha value on the test set but then they start increasing.

Dataset	Alpha	MSE	RMSE	MAE	R2	Adjusted R2
Train	0.005	24476127.32 44174	4947.335376 181546	4005.76385 22018233	0.013877632 448635158	-0.00115472 07152575734
Test	0.005	24259681.02 6543718	4925.411762 13154	3840.597618 9762887	0.00079253 18274732973	-0.06325922 895281688
Train	0.01	24478391.96 42343	4947.564245 5893685	4005.22738 28058283	0.013786392 035137873	-0.00124735 19887168383
Test	0.01	24246459.5 5231687	4924.06940 9778549	3839.23780 0635025	0.00133709 77586271149	-0.06267975 494915312
Train	0.05	24500165.4 3664655	4949.764179 902569	4001.923319 767777	0.012909157 337001487	-0.00213795 9167007652
Test	0.05	24205628.3 5381715	4919.921580 047506	3834.80489 75247293	0.00301885 2701353491	-0.06089019 5202405906
Train	0.1	24523997.70 8067063	4952.171009 574191	4000.02986 60531626	0.011948975 376603	-0.00311277 80475341815
Test	0.1	24200434.8 0514653	4919.393743 658514	3832.94038 7265711	0.00323276 4525351594	-0.06066257 1081997685
Train	0.5	24626508.6 71742547	4962.510319 560308	3997.38662 58825997	0.00781889 5774557633	-0.00730581 6180098512
Test	0.5	24260358.3 57730027	4925.48052 0490364	3833.278937 04927	0.00076463 3918220236	-0.06328891 518958613
Train	1.0	24675958.5 5730296	4967.490166 804859	3998.26197 0984851	0.00582660 1912965801	-0.0093284 8037494626
Test	1.0	24295276.3 96878663	4929.02387 8708508	3835.69660 9308444	-0.0006735 6988392708 32	-0.06481931 154315324

Part h. We get the best performance on the train set using gradient boosting regressor but the worst performance out of the three parts. This is likely due to the complexity of the model which makes it overfit on the data.

Metric	Train	Test
MSE	14926446.25730777	24507135.23925415
RMSE	3863.4759294329465	4950.468183844247
MAE	3092.7481886865007	3839.7324393022955
R2	0.398626166333897	-0.009399609491305139
Adjusted R2	0.38945888228410885	-0.07410471266382457