Report: Machine Learning Assignment 1

Angadjeet Singh(2022071)

September 14, 2024

1 (Section-A) Theoretical Questions

1.1 Effect of Increasing Model's complexity

Increasing the model's complexity by adding more features or by including higher-order polynomial terms in a regression model will reduce bias, corresponding to an increase in the variance. This situation happens since bias and variance are negatively related, or we can say they are inversely proportional; hence, increasing one leads to a decrease in the other. This relationship between bias and variance is known as the **bias-variance tradeoff**.

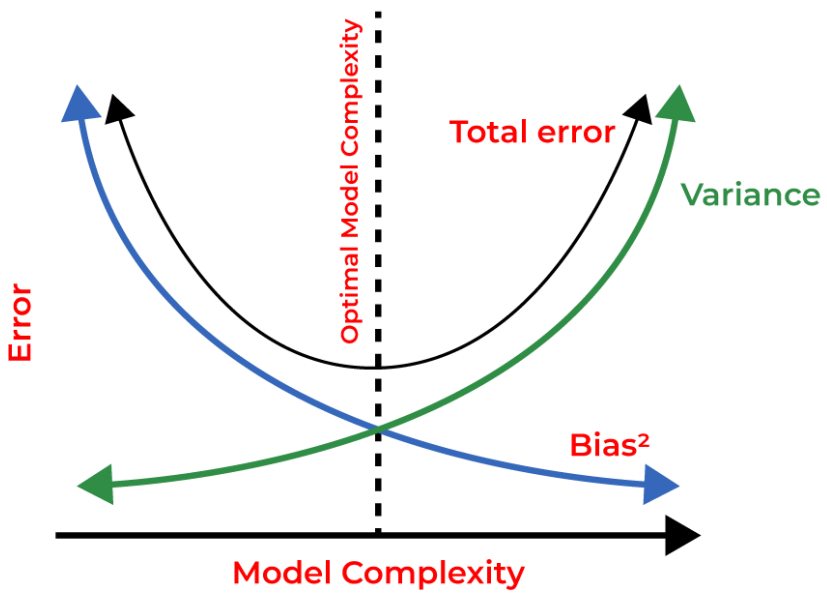


Figure 1: The Bias-Variance Tradeoff

1.2 Evaluation Metrics

Here, **TP**, **TN**, **FP**, and **FN** refer to true positives, true negatives, false positives, and false negatives, respectively. From the given data, we observe that there are **200 TPs** (spam emails that the model correctly predicted), **730 TNs** (regular emails that the model correctly predicted), **20 FPs** (regular emails that the model incorrectly predicted as spam), and **50 FNs** (spam emails that the model failed to predict as spam).

Metric	Count
TP	200
TN	730
FP	20
FN	50

Accuracy is the fraction of predictions our model got right out of all the predictions.

$$\textbf{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$$
$$= \frac{200+730}{200+730+50+20}$$
$$= \frac{930}{1000}$$
$$= 0.93 = 93\%$$

Precision tells what proportion of positive predictions was actually correct.

$$\textbf{Precision} = \frac{TP}{TP+FP}$$
$$= \frac{200}{200+20}$$
$$= \frac{200}{220}$$
$$= 0.909 \approx 90.9\%$$

Similarly to Precision, Recall aims at measuring what proportion of actual positives was identified correctly.

$$\textbf{Recall} = \frac{TP}{TP+FN}$$
$$= \frac{200}{200+50}$$
$$= \frac{200}{250}$$
$$= 0.8 = 80\%$$

The F1 score is a less known performance metric, indicating the harmonic mean of Precision and Recall.

$$\textbf{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$
$$= 2 \times \frac{0.909 \times 0.8}{0.909 + 0.8}$$
$$= \frac{1.4544}{1.709}$$
$$\approx 0.851 = 85.1\%$$

1.3 Regression Coefficients calculation

Table 1: Given Data

X	Y
3	15
6	30
10	55
15	85
18	100

To predict the future values, we need to get the best fit line or the regression line. For that, we need to calculate the slope m and intercept b for that line. These can be calculated using the formula given in the tutorials:

$$m = \frac{n(\sum xy) - (\sum x)(\sum y)}{n(\sum x^2) - (\sum x)^2}, \quad b = \frac{\sum y - m\sum x}{n}$$

We need to calculate:

$$\sum x = 3+6+10+15+18 = 52$$
$$\sum y = 15+30+55+85+100 = 285$$
$$\sum xy = (3\times15)+(6\times30)+(10\times55)+(15\times85)+(18\times100) = 45+180+550+1275+1800 = 3850$$
$$\sum x^2 = 3^2+6^2+10^2+15^2+18^2 = 9+36+100+225+324 = 694$$
$$n = 5$$

Now, substituting the values into the slope m formula:

$$m = \frac{5(\sum xy) - (\sum x)(\sum y)}{5(\sum x^2) - (\sum x)^2} = \frac{5(3850) - (52)(285)}{5(694) - (52)^2}$$
$$m = \frac{19250 - 14820}{3470 - 2704} = \frac{4430}{766} \approx 5.78$$

Now, for b:

$$b = \frac{\sum y - m\sum x}{n} = \frac{285 - 5.78 \times 52}{5} = \frac{285 - 300.56}{5} = \frac{-15.56}{5} \approx -3.11$$

Hence, the equation for the regression line can be written as:

$$y = 5.78x - 3.11$$

Prediction for $x = 12$:

$$y = 5.78 \times 12 - 3.11 = 69.36 - 3.11 = 66.25$$

The value of y for $x = 12$ according to this regression line is approximately 66.25.

1.4 Empirical Risk Testing for Complex and Non-Complex Models

$$\hat{f}_1(X) = 3X$$
$$\hat{f}_2(X) = 2X + 0.5X^2$$

Assume we have a training dataset $\{(X_i, Y_i)\}$ where the true underlying relationship between X and Y is somewhat linear, i.e., $Y = 3X$.

In this scenario, if the data points X_i are a just a little off the line defined by $Y = 3X$ but training data follows a more complex relation, such as a quadratic trend, the quadratic model f_2 might fit the training data very well, showing a lower empirical risk compared to the linear model f_1. However, this better fit is due to overfitting.

$$L(\hat{f}_1(X_i), Y_i) = (3X_i - Y_i)^2$$
$$L(\hat{f}_2(X_i), Y_i) = (2X_i + 0.5X_i^2 - Y_i)^2$$

Since f_2 includes a quadratic term, it may have a lower empirical risk if the training data exhibits a quadratic trend. But if the true relationship is linear, f_2 might not generalize well to new data, whereas f_1, being simpler, is more aligned with the true underlying linear relationship.

Thus, although f_2 has a lower empirical risk on the training set, f_1 may generalize better to new data, demonstrating that lower empirical risk does not always imply better generalization performance.

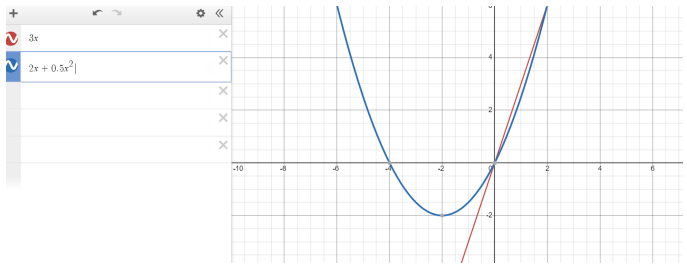


Figure 2: Plot for a linear and quadratic mode

2 Section B (Scratch Implementation)

2.1 Logistic Regression Using Batch Gradient Descent

The performance of logistic regression trained using batch gradient descent can be seen in Figure below.

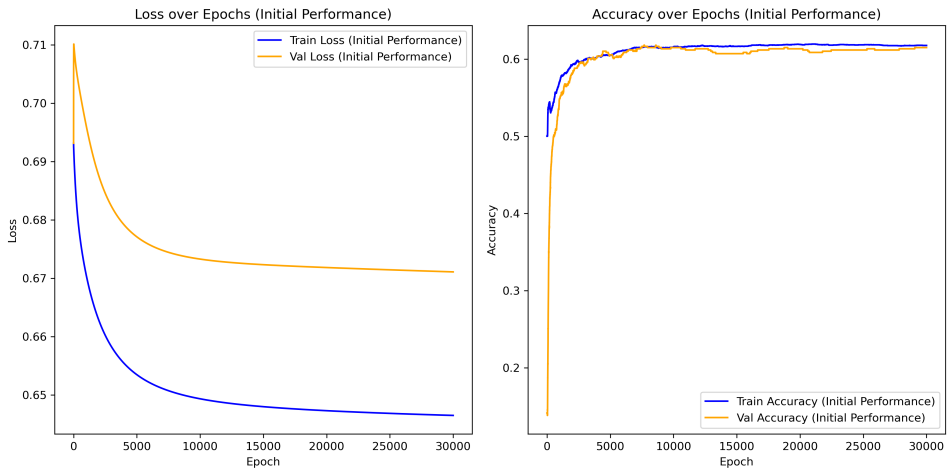


Figure 3: Performance of Logistic Regression

The plot indicates that the model begins to converge between 10,000 to 15,000 iterations. During this range, the training loss stabilizes between 63 and 65, while the validation loss settles between 67 and 68. A key observation is that the validation loss exceeds the training loss, suggesting that the model is performing better on the training data compared to the unseen validation data. This discrepancy may indicate overfitting, where the model is learning the specific patterns and noise within the training set rather than generalizing and capturing the true essence of the underlying distribution. For the accuracy we can see it converges just above 60 percent and Here also training accuracy is a bit better than validation accuracy since the model can capture the training distribution much easily.

2.2 Feature Scaling Comparison

Feature scaling can significantly impact model convergence, as shown in Figure.

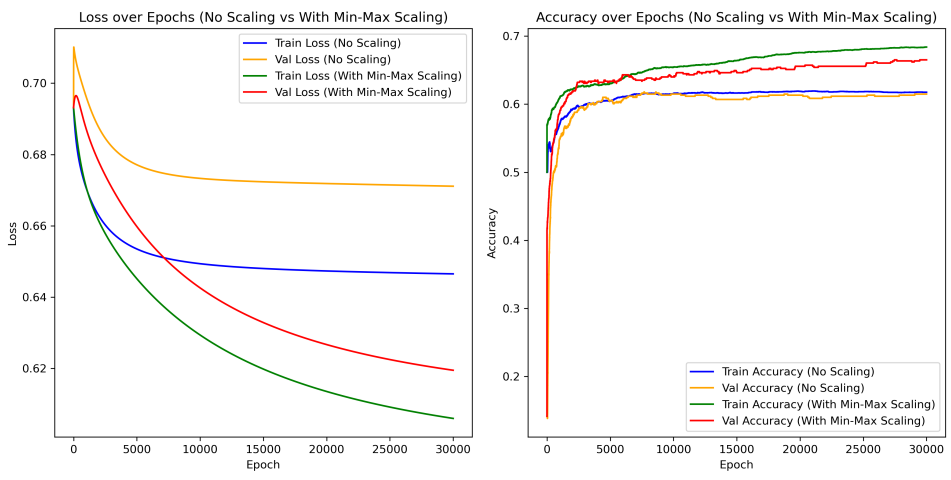


Figure 4: Effect of Scaling on Model Performance

After applying feature scaling, the model's performance improves, with both training and testing losses decreasing and accuracies increasing. Scaling ensures that all features contribute equally, allowing the optimization algorithm to converge more effectively and enhance generalization [1].

2.3 Performance Metrics on Validation Set

The confusion matrix for the validation set is shown in Figure 5. The precision, recall, F1-score, and ROC-AUC score are discussed based on this matrix.

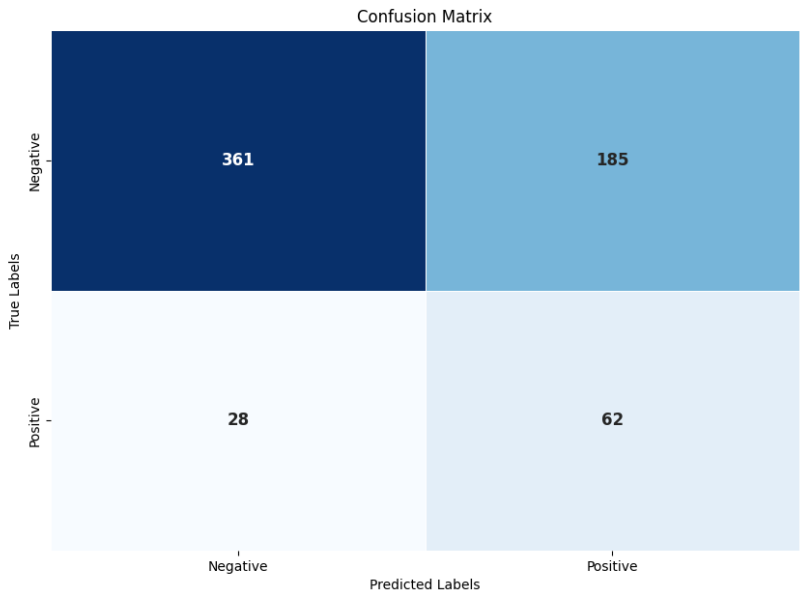


Figure 5: Confusion Matrix for Validation Set

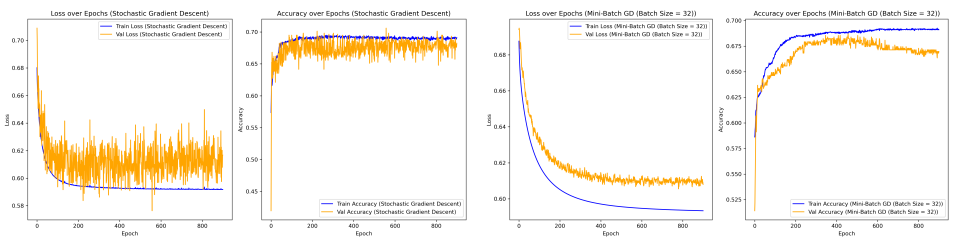
Table 2: Performance Metrics

Metric	Value
Precision	0.251
Recall	0.689
F1 Score	0.368
ROC AUC	0.675

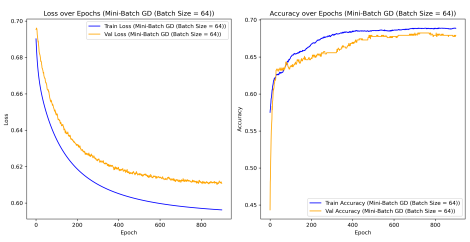
- *Precision tells what proportion of positive predictions was actually correct.*
- *Recall measures what proportion of actual positives was identified correctly.*
- *The F1 score is a less known performance metric that indicates the harmonic mean of Precision and Recall.*
- *A ROC curve shows the performance of a classification model at all classification thresholds. The AUC (Area Under the Curve) quantifies the overall ability of the model to discriminate between positive and negative classes.*

2.4 Comparison of Different Variants of Gradient Descent

Here is comparision of batch gradient descent (BGD), stochastic gradient descent (SGD), and mini-batch gradient descent (MBGD) with different batch sizes.



(a) SGD Performance (b) MBGD (Batch Size = 32)



(c) MBGD (Batch Size = 64)

Figure 6: Comparison of Optimization Algorithms

From the plots, we observe that stochastic gradient-based approaches exhibit comparable and, in some cases, even superior performance in terms of both validation loss and training loss, as well as accuracy, compared to batch-based methods. These methods also converge in a significantly lower number of epochs. Below, we discuss the performance of Stochastic Gradient Descent (SGD) and Mini-batch Gradient Descent (MBGD) and there effect on convergence speed and stability of the model.

2.4.1 Stochastic Gradient Descent (SGD)

In SGD, a single randomly selected data point is used to update the model's weights in each iteration of an epoch. As seen in the plots, there are numerous spikes, indicating that the method takes many random jumps during the convergence process. This randomness helps SGD to escape local minima, but it also results in a less stable path toward convergence. However, despite the instability, SGD tends to converge in the fewest number of epochs [2]. The stochastic nature of this approach makes it computationally inexpensive per iteration, although the convergence path can be erratic.

2.4.2 Mini-batch Gradient Descent (MBGD)

Mini-batch gradient descent provides a middle ground between full-batch and stochastic methods by using a small batch of data points to update the weights

in each iteration. From the plots, we observe that the number of spikes increases when smaller batch sizes, such as 32, are used, as the method behaves more stochastically, akin to SGD. Conversely, as the batch size increases, the method transitions toward full-batch gradient descent, and the spikes reduce [3]. The trade-off here is between stability and computation. While larger batches offer smoother convergence, they become more computationally expensive, as they require multiplying the entire data matrix. Smaller batches, on the other hand, lead to more randomness but at a lower computational cost per update.

2.4.3 Trade-offs in Stochastic Methods

A key consideration in stochastic approaches is that, since weights are updated randomly, there is no guarantee of finding a single optimal set of parameters, unlike batch gradient descent. This lack of determinism is a compromise for the computational efficiency that stochastic methods provide. Although batch gradient descent guarantees convergence to the optimal parameters (under convexity assumptions), the high computational cost makes it impractical for large datasets [4]. On the other hand, stochastic methods offer similar performance while requiring fewer computations per update, making them more efficient for large-scale learning.

2.5 Result of K-Fold Cross Validation

Note: The data used is significantly imbalanced, which may affect the performance metrics and their interpretation.

K-Fold Cross Validation Results on Initial Data without Resampling:

Table 3: K-Fold Cross Validation Metrics (Initial Data)

Fold	Accuracy	Precision	Recall	F1 Score
1	0.83353	0.00000	0.00000	0.00000
2	0.86305	0.75000	0.02542	0.04918
3	0.86423	0.57143	0.03448	0.06504
4	0.85124	0.85714	0.04580	0.08696
5	0.83589	0.50000	0.00719	0.01418

K-Fold Cross Validation Results on Resampled Data:

Table 4: K-Fold Cross Validation Metrics (Resampled Data)

Fold	Accuracy	Precision	Recall	F1 Score
1	0.83117	0.00000	0.00000	0.00000
2	0.86895	1.00000	0.01770	0.03478
3	0.84534	0.28571	0.01563	0.02963
4	0.84534	0.42857	0.02308	0.04380
5	0.85124	1.00000	0.02326	0.04545

Comparison and Insights:

- **Accuracy**: Resampling did not significantly improve accuracy. The values are comparable across folds with slight variations, indicating that resampling did not provide a substantial advantage in terms of overall accuracy.

- **Precision**: The precision on resampled data is notably higher in some folds, especially in Folds 2 and 5, reaching 1.00000. This suggests that resampling improved the model's ability to correctly identify positive instances in these cases.

- **Recall**: Recall values are generally low in both cases, but resampling shows slight improvements in some folds, particularly in Fold 4, with a recall of 0.02308 compared to lower values without resampling.

- **F1 Score**: The F1 Score, which balances precision and recall, is also improved in the resampled data for some folds. The best F1 Scores are observed in Folds 4 and 5, indicating better performance in handling class imbalances post-resampling.

Summary: Resampling improved precision and F1 scores in certain folds, but overall accuracy and recall showed modest improvements. This suggests that while resampling helps with precision and balancing class distributions, it may not dramatically enhance overall model performance in every aspect.

2.6 Effects of Regularization Techniques and Early Stopping on Convergence and Performance

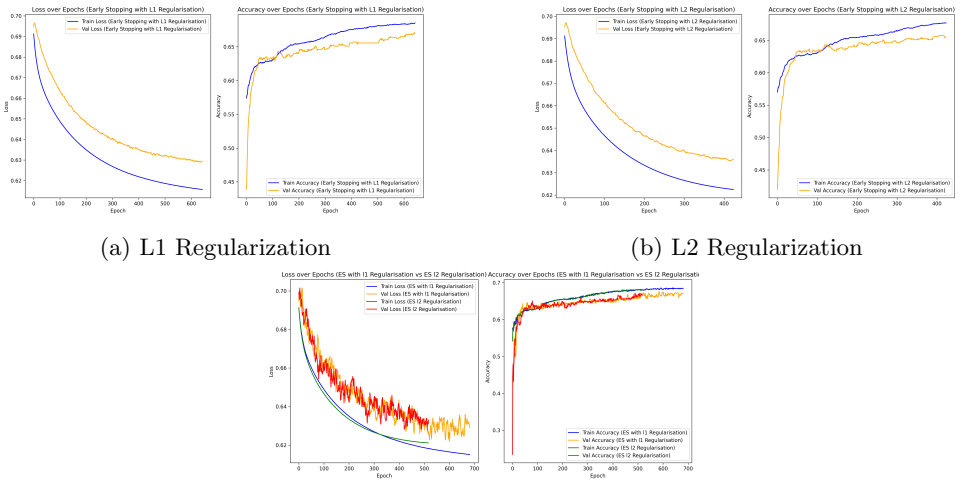
2.6.1 Regularization Techniques: L1 vs L2

Regularization helps in preventing overfitting by adding a penalty to the loss function based on the magnitude of the model coefficients. We experimented with two common regularization techniques:

- **L1 Regularization:** Also known as Lasso regularization, it adds the absolute value of the coefficients to the loss function. L1 tends to produce sparse models by driving some coefficients to zero, effectively performing feature selection [5].

- **L2 Regularization:** Also known as Ridge regularization, it penalizes the squared magnitude of the coefficients. Unlike L1, L2 tends to shrink the coefficients towards zero but not exactly zero, thus retaining all features in the model [6].

Figure 7 provides visual comparisons of L1 and L2 regularization across different metrics. From the loss and accuracy plots, we observe that L2 regularization generally results in smoother convergence compared to L1 regularization.



(a) L1 Regularization (b) L2 Regularization

(c) Combined Plots

Figure 7: Comparison of Loss and Accuracy with L1 and L2 Regularization

2.6.2 Early Stopping Criteria

Early stopping was applied to prevent overfitting by halting training after 200 epochs of no improvement in validation loss [7]. Experimenting with learning rates and regularization revealed that a learning rate of `0.0001` and L2 regularization with an alpha of `0.001` provided stable results, minimizing overfitting and underfitting while ensuring smooth convergence.

2.6.3 Impact of Early Stopping on Overfitting and Generalization

Early stopping helps balance underfitting and overfitting by halting training when validation performance stabilizes. Without it, models may overfit by continuing to optimize training loss even as validation loss increases. Early stopping ensures better generalization to unseen data, leading to stable performance across training and validation datasets.

3 (Section-C) Package Based Implementaion

Plotted all the required plots such as pair plots, violin plots, box plots, count plots, correlation metrics and attached them in the submitted zip folder by the name of plots folder in part c section. I did'nt include all plots here as it would be a lot of clutter and since they all in png and 300 dpi it would exceed the latex's limit.

3.1 EDA Insights

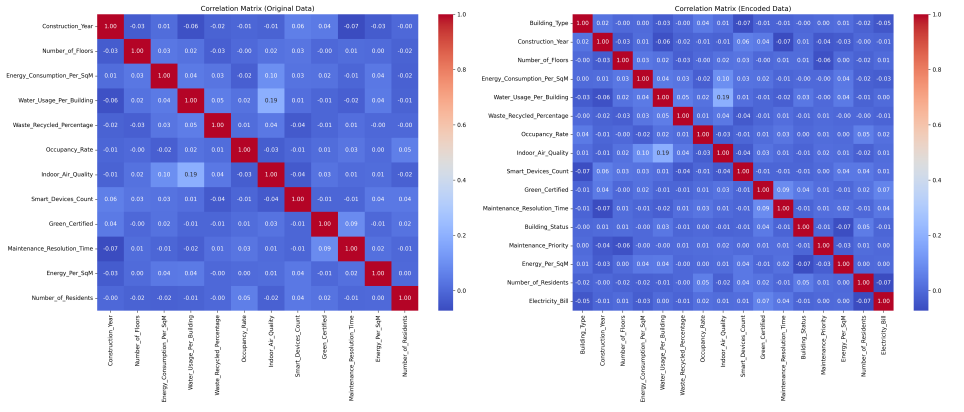


Figure 8: Correlation Heatmap

3.1.1 Key Insights

- **Water Usage Outliers:** From the violin plot of *Water_Usage_Per_Building*, significant variability and a few high outliers suggest that certain buildings consume much more water than average. This anomaly may require further investigation to identify potential inefficiencies or special conditions.

- **Occupancy Rate Outliers:** The box and violin plots of *Occupancy_Rate* indicate that most buildings have occupancy rates clustered between 60% and 90%, but several outliers below 20% raise questions about underutilization, which may correlate with maintenance issues or building condition.

- **Maintenance Resolution Time Delays:** Violin and box plots of *Maintenance_Resolution_Time* highlight extended resolution times for a subset of buildings, with some delays exceeding 50 days. These outliers suggest that certain buildings may face operational challenges that impact maintenance efficiency.

- **Weak Correlations Among Variables:** The overall correlation heatmap reveals mostly weak relationships between the dataset variables, with few correlations exceeding 0.1. This suggests a high degree of independence among variables, which could complicate efforts to find strong predictive models.

- **Positive Correlation Between Indoor Air Quality and Water Usage:** A moderate positive correlation (0.19) exists between *Indoor_Air_Quality* and *Water_Usage_Per_Building*, potentially indicating that buildings with higher water usage (e.g., for HVAC systems) may maintain better indoor air quality.

3.2 Dimensionality Reduction Using UMAP

UMAP Observations: The UMAP plot reveals clear stratification in electricity bills, with clusters indicating distinct energy consumption levels. A smooth gradient from low to high bills (yellow to pink) suggests a continuous spectrum, while the isolated pink cluster may represent outliers with unique energy demands. The circular shape hints at complex underlying structures.

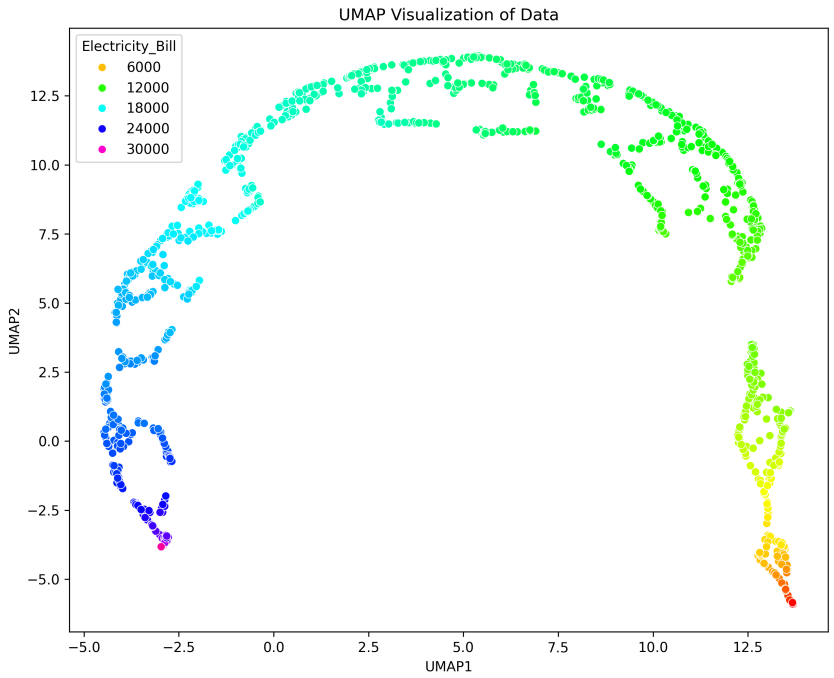


Figure 9: Umap Plot

3.3 Data Preprocessing and Model Evaluation

The evaluation metrics for Linear Regression reveal that the model struggles to explain the variance in the data, as evidenced by the low R2 and Adjusted R2 values. The Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) indicate that the prediction errors are relatively large.

Table 5: Evaluation Metrics for Linear Regression

Metric	Training Data	Testing Data
MAE	4006.33	3842.41
MSE	24475013.17	24278016.16
RMSE	4947.22	4927.27
R2	0.01392	0.00004
Adjusted R2	-0.00111	-0.06406

MAE: The MAE for both training and testing is around 4000, meaning the average prediction error is 4000 units.
MSE and RMSE: The high MSE and RMSE values (24 million and 4900, respectively) indicate significant prediction errors.
R2 and Adjusted R2: The near-zero R2 and negative Adjusted R2 on the test set show the model has very little predictive power, and adding features doesn't improve model performance.

3.4 Feature Selection Using RFE or Correlation Analysis (2 Marks)

After applying RFE, the selected features improve the model slightly, as shown by a small improvement in the R2 values. However, the errors remain relatively high.

Table 6: Evaluation Metrics for Feature Selection

Metric	Training Data	Testing Data
MAE	4006.47	3813.95
MSE	24569032.91	23941409.06
RMSE	4956.72	4892.99
R2	0.01013	0.01390
Adjusted R2	0.00715	0.00188

MAE, MSE, and RMSE: Errors remain nearly the same, indicating the model's performance hasn't improved much with the selected features.
R2 and Adjusted R2: The small positive R2 on both train and test data suggests a marginal improvement in the model's ability to explain variance, but it is still weak overall.

3.5 Ridge Regression with One-Hot Encoding (2 Marks)

Introducing Ridge Regression with One-Hot Encoding shows minor improvements in performance metrics compared to plain Linear Regression. However, the errors are still high, and the R2 values remain low.

Table 7: Evaluation Metrics for Ridge Regression

Metric	Training Data	Testing Data
MAE	3976.70	3797.50
MSE	24188925.37	24130089.94
RMSE	4918.22	4912.24
R2	0.02545	0.00613
Adjusted R2	0.00655	-0.07597

MAE, MSE, and RMSE: The errors are slightly lower compared to plain Linear Regression, suggesting that Ridge Regression helps reduce prediction errors, although the improvement is marginal.
R2 and Adjusted R2: The R2 values show a minor improvement in capturing variance, but it's still far from satisfactory, especially on the test data where the model struggles to generalize.

3.6 Independent Component Analysis (ICA) (2 Marks)

Independent Component Analysis (ICA) was performed on the one-hot encoded dataset with different numbers of components (4, 5, 6, and 7). The following metrics were evaluated for each setting. Overall, the results show that ICA does not significantly improve model performance, as R2 values remain low and errors remain high across different numbers of components.

Table 8: Evaluation Metrics for ICA (n_components=4)

Metric	Training Data	Testing Data
MAE	4010.98	3877.44
MSE	24701058.76	24726610.06
RMSE	4970.02	4972.59
R2	0.00482	-0.01844
Adjusted R2	0.00081	-0.03507

MAE, MSE, RMSE: Errors remain high, and there is little difference between the training and testing data.
R2 and Adjusted R2: The negative R2 on the testing data indicates that the model performs worse than a simple mean-based prediction, showing that ICA with 4 components does not improve predictive ability.

Table 9: Evaluation Metrics for ICA (n_components=5)

Metric	Training Data	Testing Data
MAE	4008.44	3881.23
MSE	24683781.38	24829340.47
RMSE	4968.28	4982.90
R2	0.00551	-0.02267
Adjusted R2	0.00051	-0.04363

MAE, MSE, RMSE: Errors are almost identical to the case of 4 components, showing minimal improvement.
R2 and Adjusted R2: The R2 remains negative for the testing data, indicating that the model's predictions are still poor with 5 components.

Table 10: Evaluation Metrics for ICA (n_components=6)

Metric	Training Data	Testing Data
MAE	4009.40	3824.80
MSE	24682728.73	24315105.41
RMSE	4968.17	4931.03
R2	0.00555	-0.00149
Adjusted R2	-0.00045	-0.02622

MAE, MSE, RMSE: The MAE decreases slightly on the testing data, indicating some improvement with 6 components.
R2 and Adjusted R2: The R2 value on the testing data increases to nearly zero, showing a very slight improvement, though the model still explains almost no variance.

Table 11: Evaluation Metrics for ICA (n_components=7)

Metric	Training Data	Testing Data
MAE	4008.40	3875.73
MSE	24677540.75	24692967.25
RMSE	4967.65	4969.20
R2	0.00576	-0.01705
Adjusted R2	-0.00125	-0.04647

MAE, MSE, RMSE: Similar to the other components, the errors remain high and consistent across training and testing data.
R2 and Adjusted R2: The R2 remains negative for testing data, confirming that ICA with 7 components does not provide a significant improvement in model performance.

Summary: Across all the component settings (4 to 7), ICA does not meaningfully improve the predictive power of the model. The MAE, MSE, and RMSE remain high, and the R2 values show no substantial improvement. The testing data consistently yields negative R2, indicating that none of the ICA configurations help the model generalize well.

3.7 ElasticNet Regularization (1.5 Marks)

ElasticNet regularization (a combination of L1 and L2 penalties) was applied while training a linear model on the preprocessed dataset from part (c). The evaluation metrics are compared for different values of the mixing parameter, alpha. The results indicate that increasing alpha has a minimal impact on model performance, with only slight variations in MAE, MSE, and R2 values.

Table 12: Evaluation Metrics for ElasticNet Regularization (alpha=0.1)

Metric	Training Data	Testing Data
MAE	4005.47	3841.13
MSE	24475813.08	24267836.22
RMSE	4947.30	4926.24
R2	0.01389	0.00046
Adjusted R2	-0.00114	-0.06362

MAE, MSE, RMSE: Errors are moderate but fairly consistent between training and testing, indicating stable model performance with a low alpha.
R2 and Adjusted R2: The R2 value remains near zero for testing, indicating that the model struggles to explain much variance in the data.

Table 13: Evaluation Metrics for ElasticNet Regularization (alpha=2.0)

Metric	Training Data	Testing Data
MAE	4002.21	3833.02
MSE	24561803.70	24243872.67
RMSE	4955.99	4923.81
R2	0.01043	0.00144
Adjusted R2	-0.00466	-0.06257

MAE, MSE, RMSE: Minor changes in error metrics are observed as alpha increases, but the differences are negligible.
R2 and Adjusted R2: The R2 values remain close to zero, showing that increasing alpha does not significantly affect model performance.

Table 14: Evaluation Metrics for ElasticNet Regularization (alpha=5.0)

Metric	Training Data	Testing Data
MAE	4003.29	3834.64
MSE	24652049.10	24277056.63
RMSE	4965.08	4927.18
R2	0.00679	0.00008
Adjusted R2	-0.00835	-0.06402

MAE, MSE, RMSE: The errors slightly increase with a higher alpha, though the change is still very small.
R2 and Adjusted R2: The R2 decreases marginally, indicating that increasing alpha does not improve the model's ability to fit the data.

Summary: Across different alpha values (0.1 to 5.0), ElasticNet regularization shows minimal impact on model performance. While errors remain consistent and low across the training and testing datasets, the R2 and Adjusted R2 values stay close to zero, suggesting that the model's ability to explain variance in the target variable does not improve significantly with ElasticNet regularization.

3.8 Gradient Boosting Regressor (1.5 Marks)

Use the Gradient Boosting Regressor to perform regression on the preprocessed dataset from part (c). Report the following evaluation metrics:

Table 15: Evaluation Metrics for Gradient Boosting Regressor

Metric	Train Data	Test Data
MAE	3092.75	3815.70
MSE	14926446.26	24392500.90
RMSE	3863.48	4938.88
R2	0.39863	-0.00468
Adjusted R2	0.38946	-0.06908

Comparison: The results for the Gradient Boosting Regressor can be compared with the linear models from parts (c) and (g):

- **MAE**: The Gradient Boosting Regressor achieves a significantly lower MAE on the training data (3092.75) compared to the linear models using ElasticNet, where MAE values ranged around 4000 for both training and testing. The test MAE (3815.70) is also comparable to ElasticNet results, indicating slightly better performance in generalization.

- **MSE and RMSE**: The MSE and RMSE on the training data for Gradient Boosting are lower (14926446.26 and 3863.48) compared to ElasticNet, where MSE values were around 24 million. However, on the test data, Gradient Boosting's MSE (24392500.90) and RMSE (4938.88) are similar to ElasticNet, showing no significant advantage in performance on unseen data.

- **R2 and Adjusted R2**: Gradient Boosting provides a better fit on the training data, with an R2 of 0.39863 compared to ElasticNet's lower values close to 0. However, the test R2 (-0.00468) indicates poor generalization, similar to ElasticNet, where test R2 was around zero. The Adjusted R2 for Gradient Boosting on the testing data is negative, suggesting potential overfitting.

Summary: Although Gradient Boosting shows improved performance on the training data, it does not offer significant advantages over ElasticNet on the test data. The lower training error suggests Gradient Boosting captures more complexity, but it struggles with generalization, similar to the linear models.

References

[1] Ng, A. (2011). *Machine Learning*. Stanford University.

[2] L. Bottou, *Stochastic Gradient Descent Tricks*. Neural Networks: Tricks of the Trade, Springer, 2012.

[3] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, P. T. P. Tang, *On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima*. arXiv preprint arXiv:1609.04836, 2016.

[4] I. Goodfellow, Y. Bengio, A. Courville, *Deep Learning*. MIT Press, 2016.

[5] A. Ng, *Feature selection, L1 vs. L2 regularization, and rotational invariance*. Proceedings of the Twenty-first International Conference on Machine Learning, ACM, 2004.

[6] A. E. Hoerl, R. W. Kennard, *Ridge Regression: Biased Estimation for Nonorthogonal Problems*. Technometrics, 1970.

[7] L. Prechelt, *Early Stopping — But When?*. Neural Networks: Tricks of the Trade, Springer, 1998.