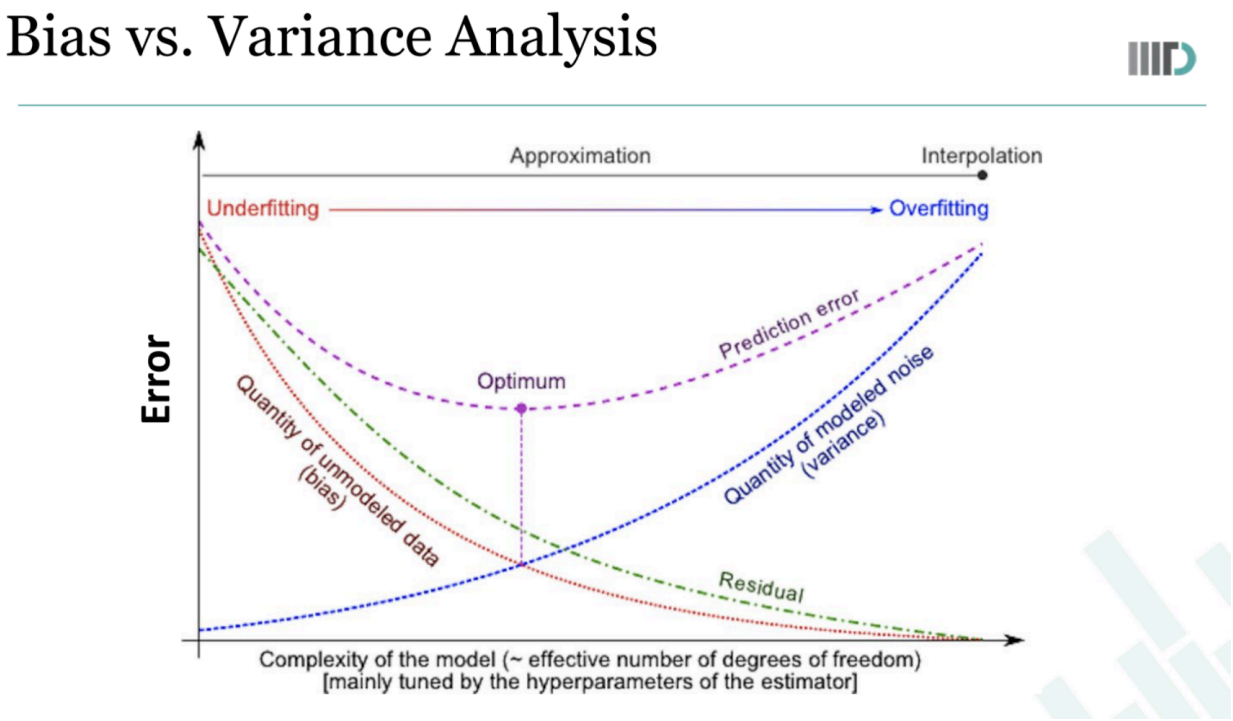
Q1
A)



Bias refers to the error introduced by oversimplifying the model. In simple models, bias tends to be high, as the model is unable to capture the underlying patterns in the data, leading to underfitting. However, as model complexity increases, bias decreases since the model becomes better at fitting the training data. On the other hand, **variance** refers to how sensitive the model is to small fluctuations in the training data. A more complex model tends to have higher variance, as it can capture the noise in the training data, leading to overfitting, where the model performs well on the training set but poorly on unseen data.
As you increase the complexity of your model, such as by adding more features or incorporating higher-order polynomial terms in a regression model, what is most likely to occur is an increase in **variance** and a decrease in **bias**. With higher complexity, the model can better capture the patterns in the training data, reducing bias and fitting the training set more accurately. However, this also makes the model more sensitive to fluctuations in the data, leading to higher variance. As a result, the model may overfit the training data and struggle to generalize well to new, unseen data.

b)
To evaluate the model's classification performance, we will use a confusion matrix. In this case, I will consider legitimate (genuine) emails as the positive class, and spam emails as the negative class.

For the given problem, the confusion matrix values are as follows:

True Positive (TP): Legitimate emails correctly classified as legitimate – 730
False Positive (FP): Legitimate emails incorrectly classified as spam – 20
True Negative (TN): Spam emails correctly classified as spam – 200
False Negative (FN): Spam emails incorrectly classified as legitimate – 50

Here is the confusion matrix-

	Predicted as legit	Predicted as spam
Actual email	730	20
Spam email	200	50

Accuracy measures the proportion of correctly classified emails (both legitimate and spam) out of the total number of emails.

Accuracy=(TP+TN)/(TP+TN+FP+FN)=(730+200)/(730+20+200+50)=93

Accuracy = 93%.

Precision is the proportion of correctly classified legitimate emails (TP) out of all emails classified as legitimate (TP + FP).

Precision=(TP)/(TP+FP)=(730)/(730+20)=97.33

Precision = 97.33%.

Recall (also known as sensitivity or true positive rate) is the proportion of correctly classified legitimate emails (TP) out of all actual legitimate emails (TP + FN).

Recall=(TP)/(TP+FN)=(730)/(730+50)=93.53

Recall = 93.59%.

Negative Precision (Precision for Spam) is given by:

NP=TN/(TN+FN)=200/(200+50)=0.8

Negative Precision = 80%.

Specificity is given by

Specificity=TN/(TN+FP)=200/(200+20)=0.909

Specificity ≈ 90.9%.

c)

To find the equation of the regression line for the given data, we use the formula for simple linear regression:

Here is the equation of line $y = m * x + b$

And here are the formula of b and m

[we take submission for all the values of x and ,here $\bar{x}$ is mean of all values of x and $\bar{y}$ is the mean of all the values of y

slope(m)= $m = \Sigma((x_i - \bar{x}) * (y_i - \bar{y})) / \Sigma((x_i - \bar{x})^2)$

intercept(b)= $b = \bar{y} - m * \bar{x}$

$\bar{x} = (3 + 6 + 10 + 15 + 18) / 5 = 52 / 5 = 10.4$

$\bar{y} = (15 + 30 + 55 + 85 + 100) / 5 = 285 / 5 = 57$

$\Sigma((x_i - \bar{x}) * (y_i - \bar{y})) = 310.8 + 118.8 + 0.8 + 128.8 + 326.8 = 886$

$\Sigma((x_i - \bar{x})^2) = 54.76 + 19.36 + 0.16 + 21.16 + 57.76 = 153.2$

$m = 886 / 153.2 \approx 5.78$

$b = \bar{y} - m * \bar{x} = 57 - (5.78 * 10.4) = 57 - 60.11 \approx -3.11$

So our line is $y = 5.78 * x - 3.11$

And for x=12 we have predicted value of

$y = 5.78 * 12 - 3.11 = 69.36 - 3.11 = 66.25$

d)

Consider a dataset where the true relationship is approximately y=10x with training points such as (1, 11), (2, 19), (3, 32), (4, 40), and (5, 49).

empirical risk refers to the average loss or error of a model over a given training dataset. It is used to measure how well a model performs on the observed data.

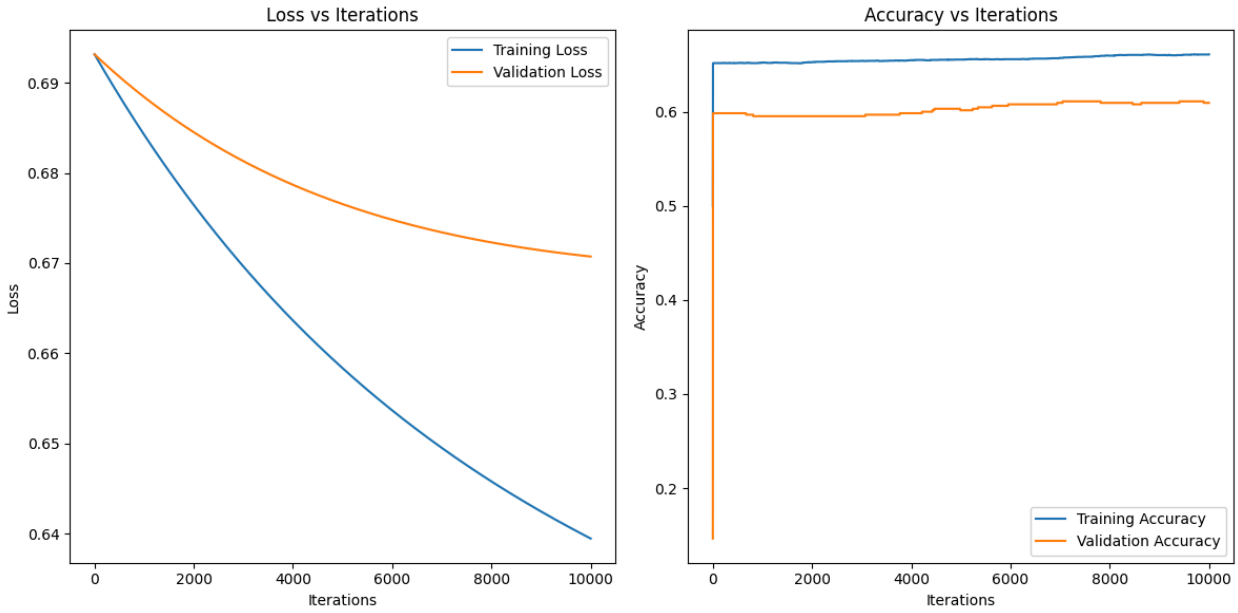
Model f1 is a high-degree polynomial regression model (e.g., degree 4) that fits the training data with very low empirical risk. However, this model overfits the training data and does not generalize well to new data.

Model f2 is a simple linear regression model that captures the true linear relationship. Although it may have a slightly higher empirical risk on the training data, it generally performs better on new data because it represents the true relationship more accurately.

For example, predicting y for x=6 with **Model f1** results in inaccuracies, while **Model f2** will predict y=60 which will be better then **Model f1**. This demonstrates that a model with lower empirical risk on the training set (like f1) may not necessarily generalize better than a simpler model (like f2).

Q2

a)



Loss vs Iterations: This plot illustrates how the training loss and validation loss change as the number of iterations increases.

- **Training Loss:** This line represents the loss calculated on the training set during each iteration. It is decreasing as the model learns.
- **Validation Loss:** This line represents the loss calculated on the validation set during each iteration. It can help identify overfitting if it starts to increase while the training loss continues to decrease.In our case

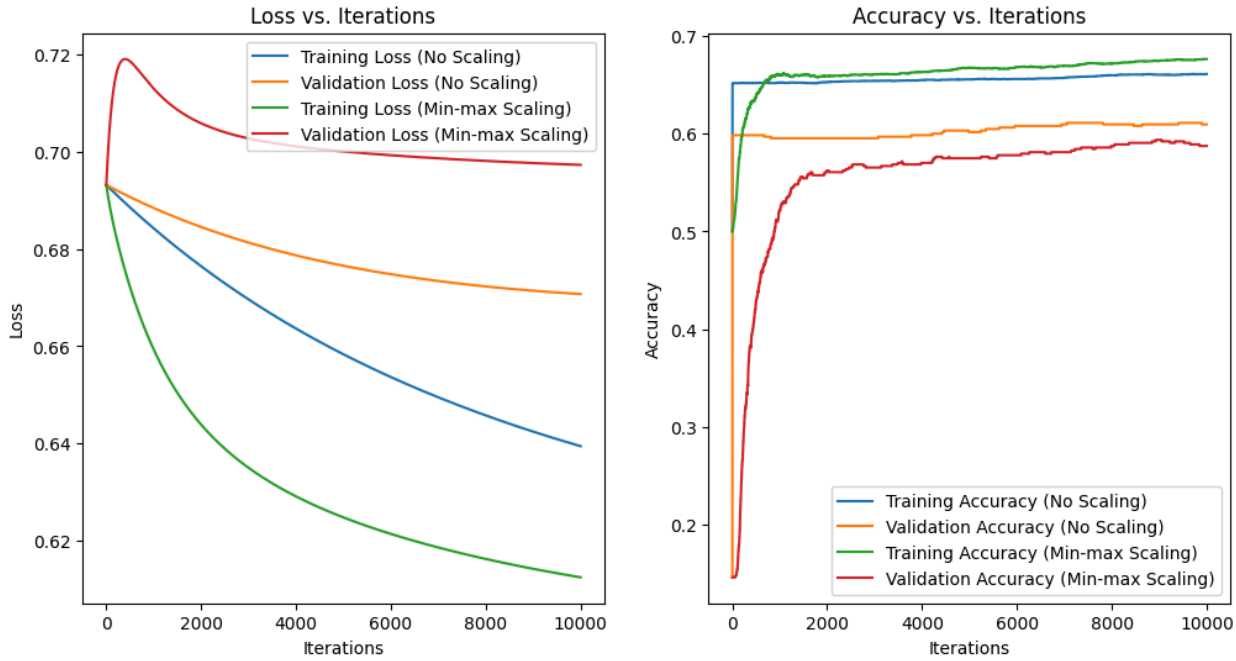
Accuracy vs Iterations: This plot shows how the training accuracy and validation accuracy change as the number of iterations increases.

- **Training Accuracy:** This line represents the accuracy on the training set during each iteration. It generally increases as the model learns.
- **Validation Accuracy:** This line represents the accuracy on the validation set during each iteration. It can help identify overfitting if it starts to decrease while the training accuracy continues to increase.

Analysis of the Plots:

- **Convergence:** The model seems to be converging as both the training and validation loss are decreasing and the training and validation accuracy are increasing. However, there might be a slight overfitting issue as the validation loss starts to increase slightly towards the end of the training.
- **Comparison:** The training loss and validation loss are similar in the beginning but diverge as the training progresses, indicating some overfitting. The training accuracy increases rapidly initially but plateaus, while the validation accuracy increases more gradually and might start to decrease slightly.

b)



- **Loss vs. Iterations:** This plot shows how the model's prediction error decreases over time (iterations). A lower loss indicates better model performance.
- **Accuracy vs. Iterations:** This plot shows how accurately the model predicts the correct class for each data point. A higher accuracy indicates better performance.

Comparing the Effects of Feature Scaling:

1. **Loss:**
    - **No Scaling:** The training loss initially decreases rapidly but then plateaus, indicating potential overfitting. The validation loss also starts decreasing but eventually increases, suggesting that the model is learning noise from the training data.
    - **Min-max Scaling:** Both training and validation losses decrease steadily and converge to a lower value compared to no scaling. This indicates better generalization and reduced overfitting.
2. **Accuracy:**
    - **No Scaling:** The training accuracy increases rapidly but then plateaus. The validation accuracy increases initially but then decreases, again suggesting overfitting.
    - **Min-max Scaling:** Both training and validation accuracies increase steadily and converge to a higher value. This indicates better model performance and generalization.
    -

- **Feature scaling is crucial for model convergence and generalization.**
- **Min-max scaling** is particularly effective in this case. By bringing all features to a common scale, it helps the optimization algorithm converge faster and prevents features with larger magnitudes from dominating the learning process.
- **Without scaling,** the model might struggle to learn meaningful patterns, leading to overfitting and poor generalization.

Conclusion:

Based on the analysis, **Min-max scaling** significantly improves the performance of the model compared to no scaling. It helps the model converge faster, reduces overfitting, and leads to better generalization on unseen data. Therefore, it is generally recommended to apply feature scaling techniques when training machine learning models, especially when dealing with features that have different scales or units.

c)

```
Metrics for model without scaling:
Confusion Matrix:
 [[320 222]
 [ 26  67]]
Precision: 0.23183391003460208
Recall: 0.7204301075268817
F1 Score: 0.3507853403141361
ROC-AUC Score: 0.6554180057929612

Metrics for model with Min-Max Scaling:
Confusion Matrix:
 [[306 236]
 [ 26  67]]
Precision: 0.22112211221122113
Recall: 0.7204301075268817
F1 Score: 0.3383838383838384
ROC-AUC Score: 0.6425028766416696
```

Recall: Both models have similar recall, indicating that they can correctly identify most of the positive cases.

Precision: The model without scaling has slightly higher precision, suggesting it makes fewer false positive predictions.

F1 Score: The model without scaling also has a slightly higher F1 score, reflecting the balance between precision and recall.

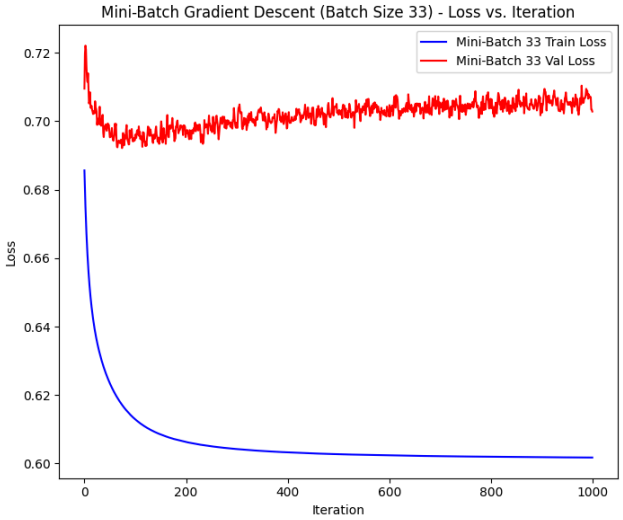
ROC-AUC Score: The model without scaling has a slightly higher ROC-AUC score, indicating better overall classification performance.

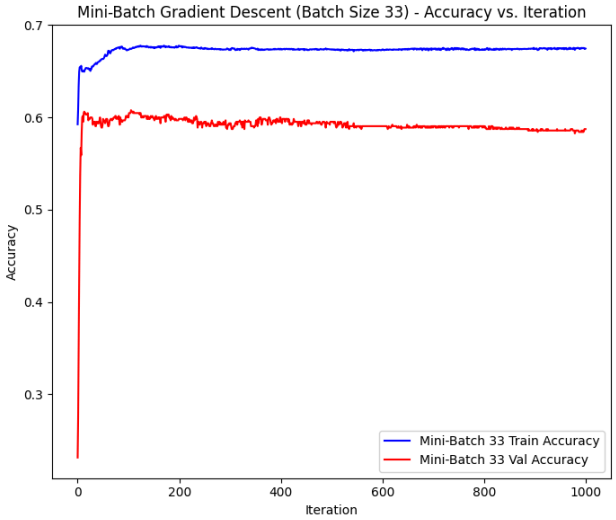
While both models perform reasonably well, the model **without scaling** appears to have a slight edge in terms of precision and F1 score.

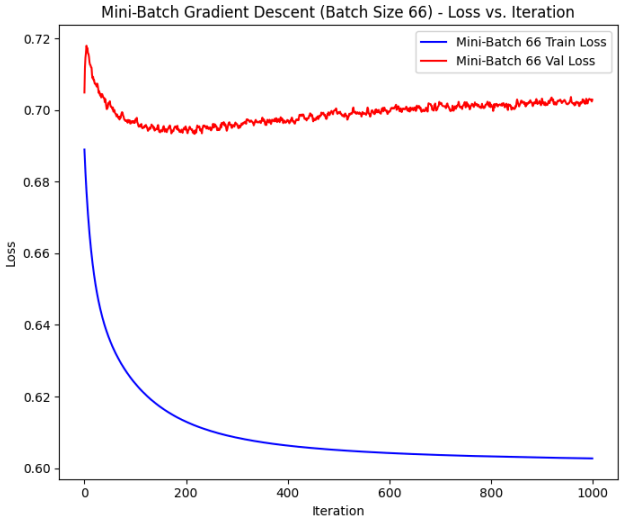
Min-max scaling can potentially reduce the importance of certain features, especially when dealing with features that have a wide range of values, which might be the case here.

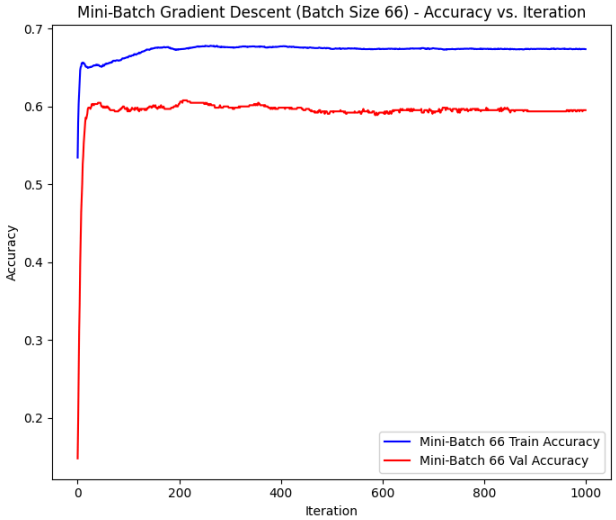
d)

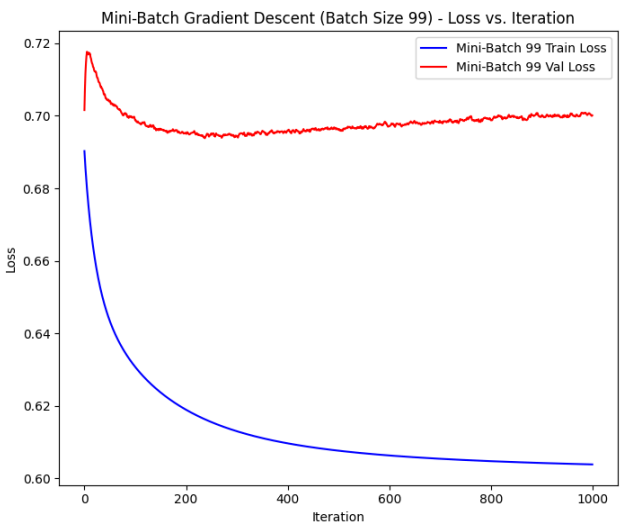
The mini batch plots-

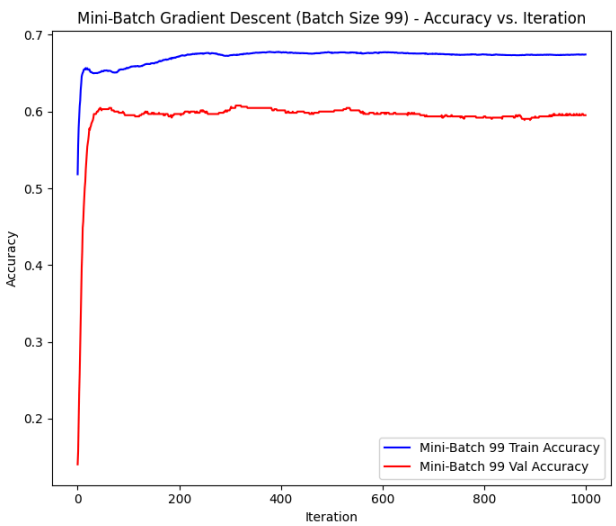

Mini-Batch Gradient Descent (Batch Size 33) - Loss vs. Iteration
Mini-Batch 33 Train Loss
Mini-Batch 33 Val Loss
Loss
Iteration


Mini-Batch Gradient Descent (Batch Size 33) - Accuracy vs. Iteration
Mini-Batch 33 Train Accuracy
Mini-Batch 33 Val Accuracy
Accuracy
Iteration

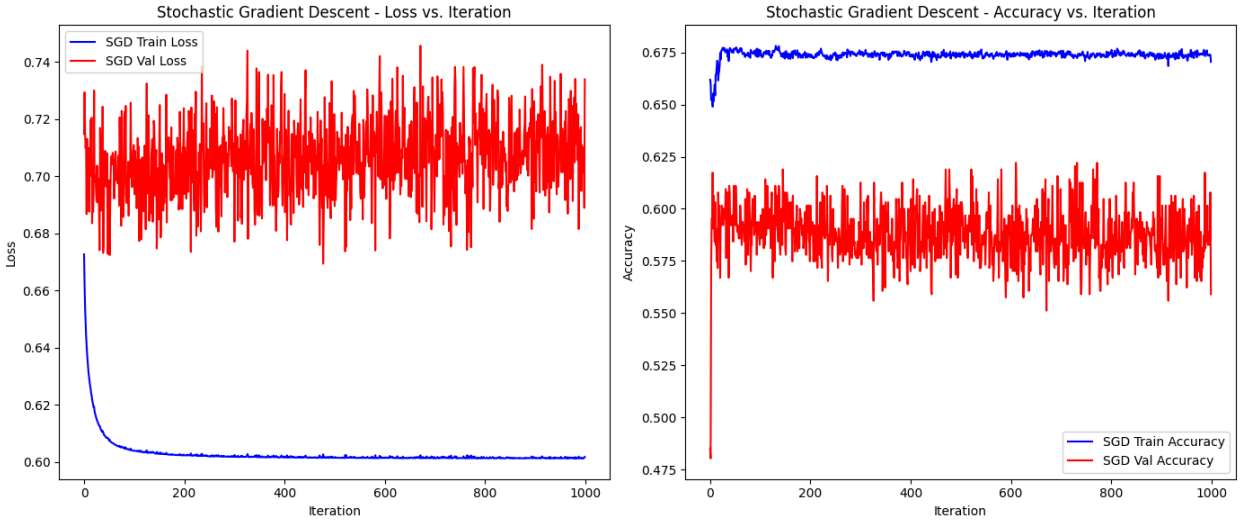

Mini-Batch Gradient Descent (Batch Size 66) - Loss vs. Iteration
Mini-Batch 66 Train Loss
Mini-Batch 66 Val Loss
Loss
Iteration


Mini-Batch Gradient Descent (Batch Size 66) - Accuracy vs. Iteration
Mini-Batch 66 Train Accuracy
Mini-Batch 66 Val Accuracy
Accuracy
Iteration


Mini-Batch Gradient Descent (Batch Size 99) - Loss vs. Iteration
Mini-Batch 99 Train Loss
Mini-Batch 99 Val Loss
Loss
Iteration


Mini-Batch Gradient Descent (Batch Size 99) - Accuracy vs. Iteration
Mini-Batch 99 Train Accuracy
Mini-Batch 99 Val Accuracy
Accuracy
Iteration

The stochastic batch plot-



Loss vs. Iteration:

- **SGD:** The loss function exhibits a more erratic and noisy pattern, with frequent fluctuations.Despite the initial oscillations, SGD appears to converge faster than Mini-Batch Gradient Descent in the early iterations.
- **Mini-Batch Gradient Descent:** The loss function shows a smoother and more consistent decrease, with fewer significant oscillations.

Accuracy vs. Iteration:

- **SGD:** The accuracy fluctuates more significantly, with occasional peaks and troughs.
- **Mini-Batch Gradient Descent:**The accuracy curve is generally more stable and reaches a higher peak compared to SGD, indicating better overall performance.

Observations:

- **Convergence:** SGD initially converges faster than Mini-Batch Gradient Descent, but the latter exhibits a more stable and consistent convergence path over time.
- **Oscillations:** SGD exhibits more pronounced oscillations in both loss and accuracy, indicating higher variance in the gradients.
- **Noise:** The overall noise level is lower in Mini-Batch Gradient Descent, suggesting a more consistent update direction.
- **Accuracy:** Mini-Batch Gradient Descent consistently achieves higher accuracy levels compared to SGD.

While SGD initially converges faster, Mini-Batch Gradient Descent demonstrates a more stable and consistent training process overall, especially in terms of long-term convergence, accuracy, and stability.

e)

Report Metrics

Using the results you provided, you would summarize the cross-validation results as follows:

- **Accuracy**: Mean = 0.5495, Std Dev = 0.0399
- **Precision**: Mean = 0.5280, Std Dev = 0.0287
- **Recall**: Mean = 0.9765, Std Dev = 0.0163
- **F1 Score**: Mean = 0.6846, Std Dev = 0.0214

Discuss Stability and Variance

Based on the results:

1. **Accuracy**:
    - **Mean**: 0.5495
    - **Std Dev**: 0.0399

    The accuracy is relatively moderate, with some variability between folds. A standard deviation of 0.0399 indicates that the model's performance in terms of accuracy varies somewhat across different subsets of the data.

2. **Precision**:
    - **Mean**: 0.5280
    - **Std Dev**: 0.0287

    The precision is lower compared to recall, and the standard deviation suggests a moderate level of variability. This indicates that the model might struggle with consistently identifying positive cases correctly across different folds.

3. **Recall**:
    - **Mean**: 0.9765
    - **Std Dev**: 0.0163

    The recall is high and shows very little variation across folds (low standard deviation). This suggests that the model is very effective at identifying most of the positive cases in the data, regardless of the subset used for testing.

4. **F1 Score**:
    - **Mean**: 0.6846
    - **Std Dev**: 0.0214

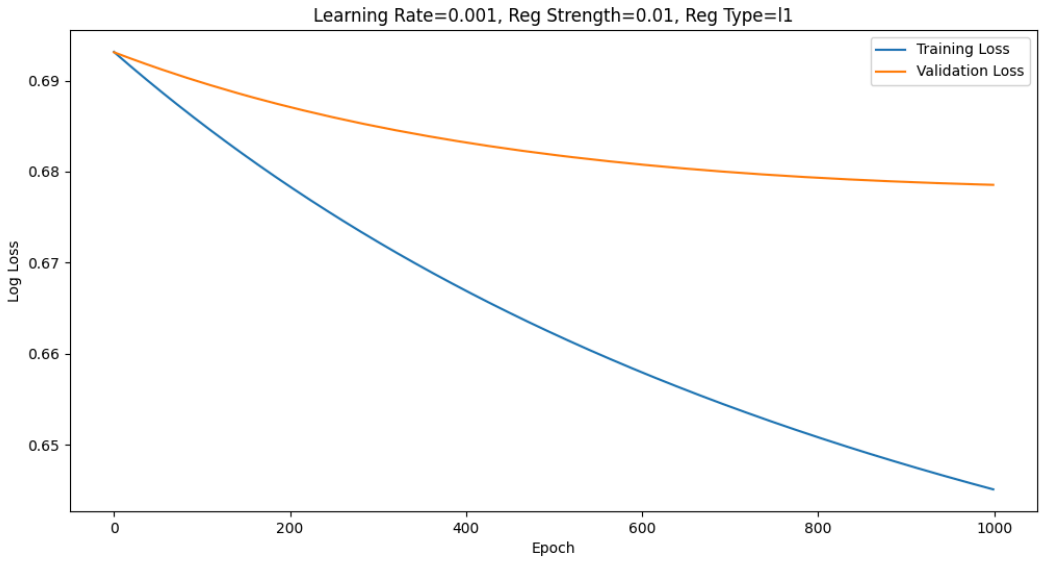
    The F1 score, which balances precision and recall, is fairly high with a low standard deviation. This suggests that the model maintains a good balance between precision and recall across different folds.

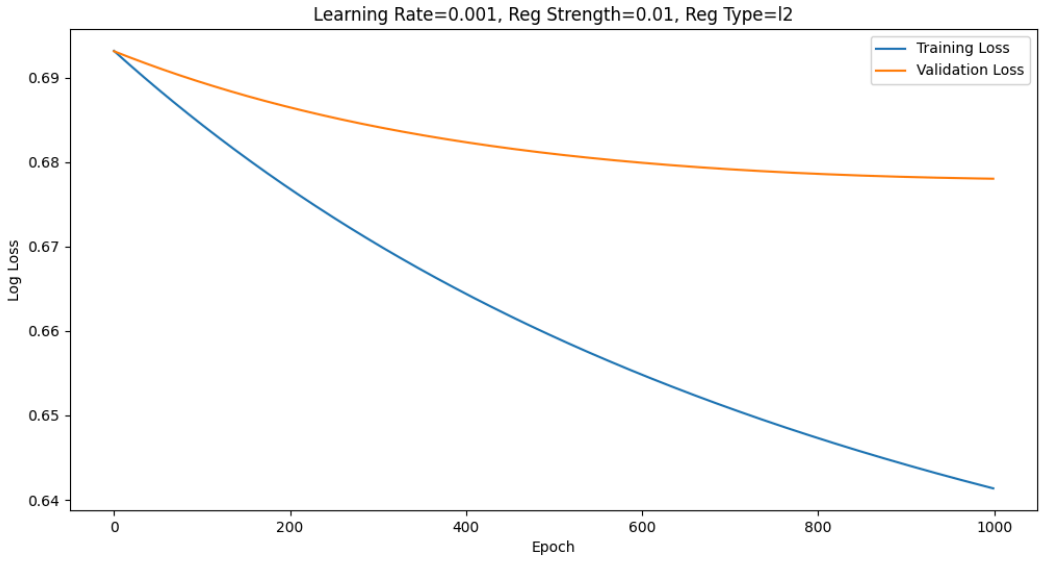
Summary

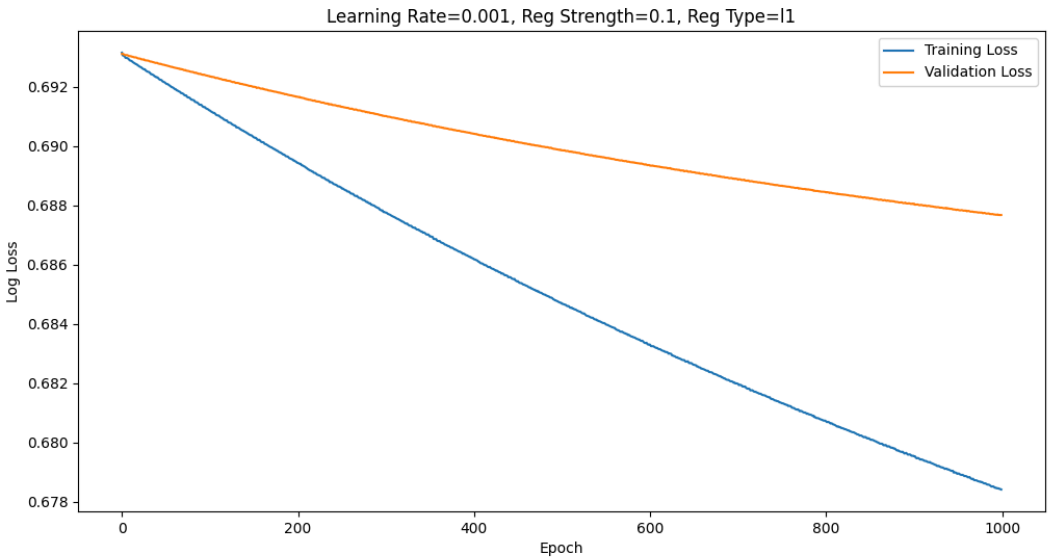
The model exhibits good recall with low variance, suggesting that it is consistently good at identifying positive cases. However, precision and accuracy have moderate variability, indicating that while the model is effective at identifying positives, there is some inconsistency in how many of these are truly positive versus false positives. The F1 score indicates a good balance between precision and recall, with stability across folds.

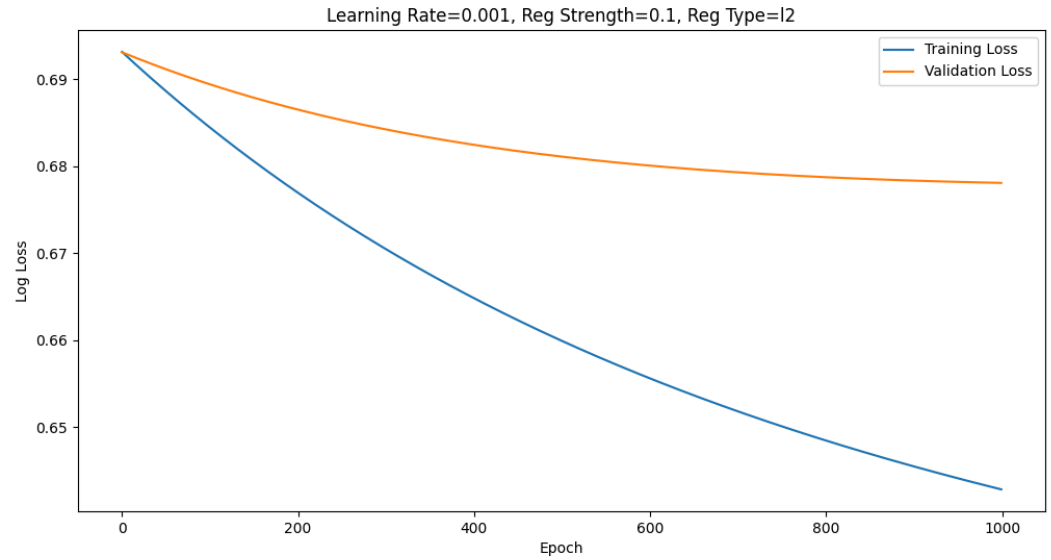
f)

Here are the experimented plots, Here i have used 100 as a stopping criteria

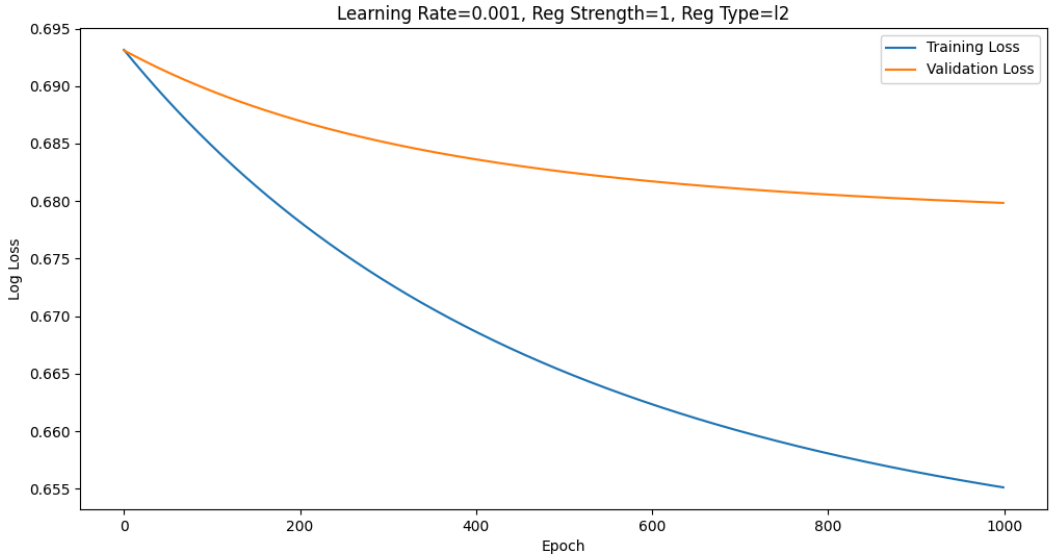


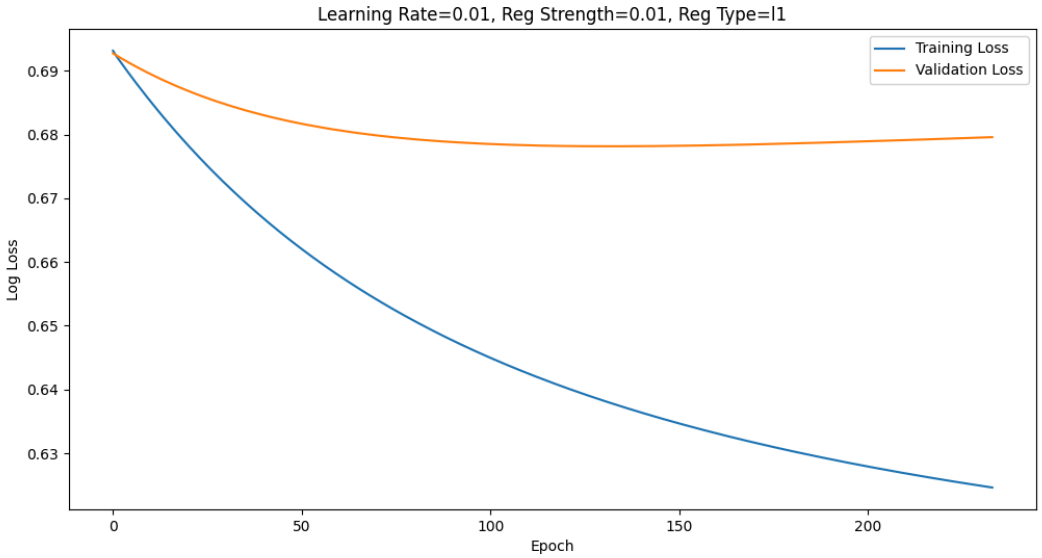


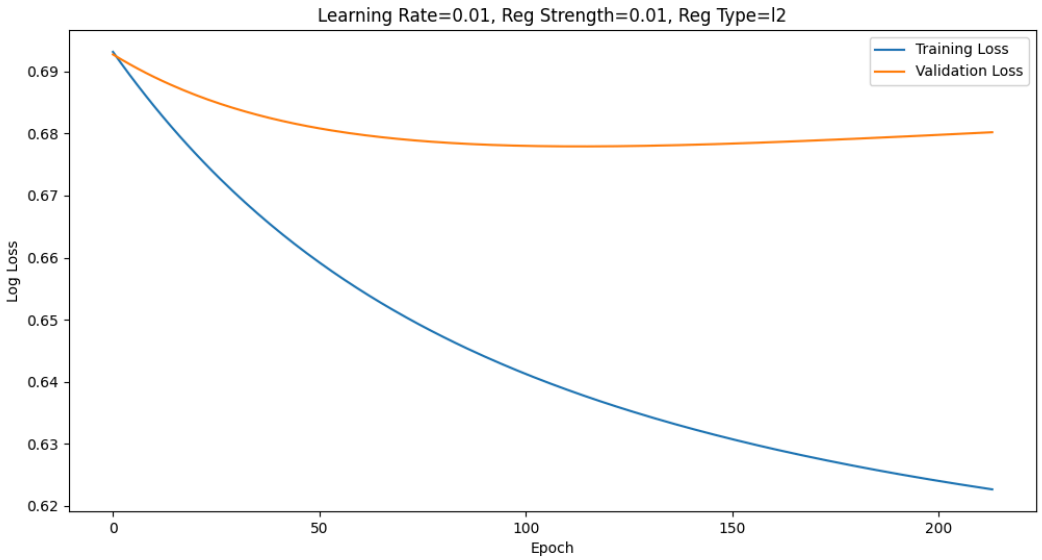

Learning Rate=0.001, Reg Strength=0.1, Reg Type=l1
Training Loss
Validation Loss
Log Loss
0.692
0.690
0.688
0.686
0.684
0.682
0.680
0.678
0
200
400
600
800
1000
Epoch

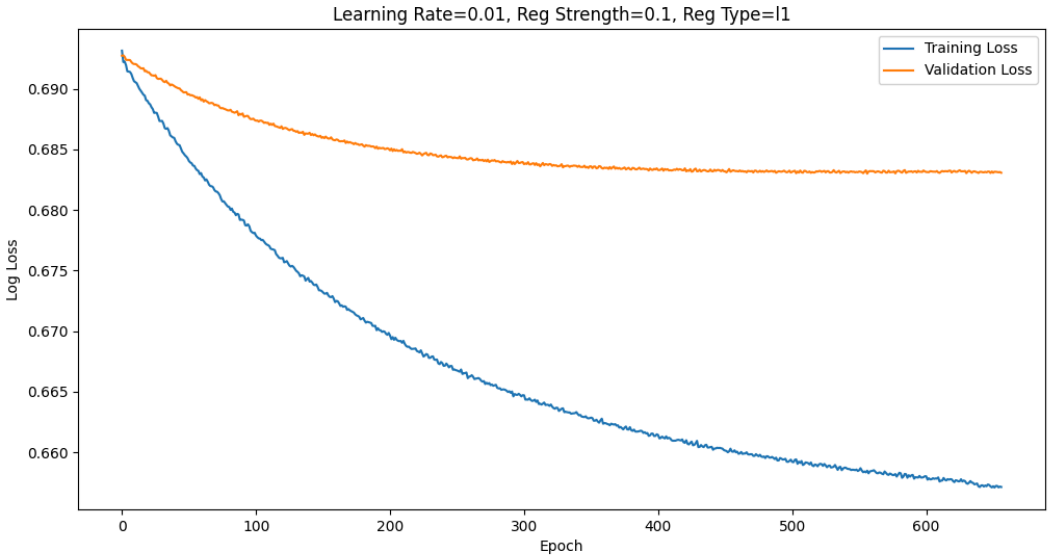

Learning Rate=0.001, Reg Strength=0.1, Reg Type=l2
Training Loss
Validation Loss
Log Loss
0.69
0.68
0.67
0.66
0.65
0
200
400
600
800
1000
Epoch

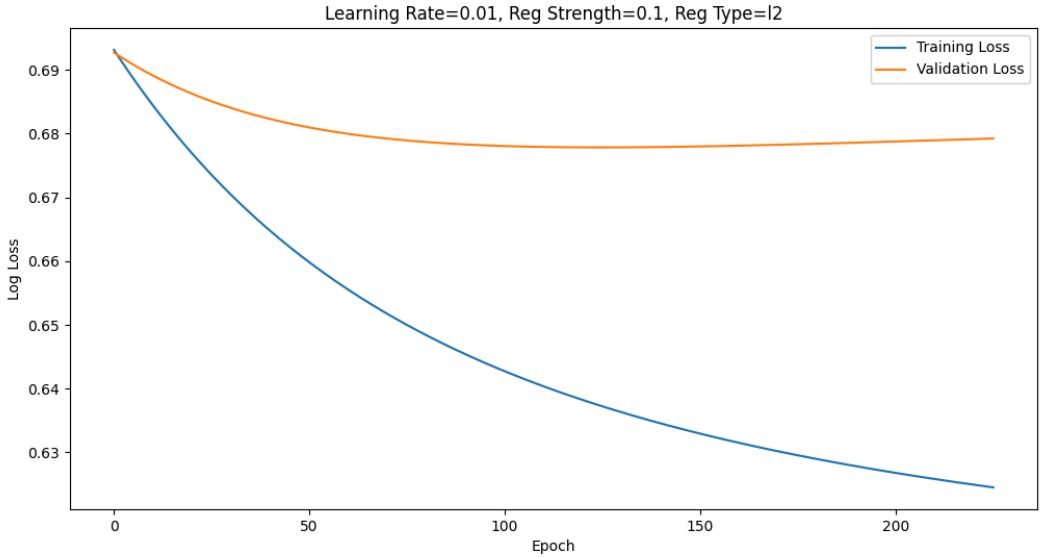

Learning Rate=0.001, Reg Strength=1, Reg Type=l1
Training Loss
Validation Loss
Log Loss
Epoch
0.6938
0.6936
0.6934
0.6932
0.6930
0.6928
0.6926
0
20
40
60
80
100
120

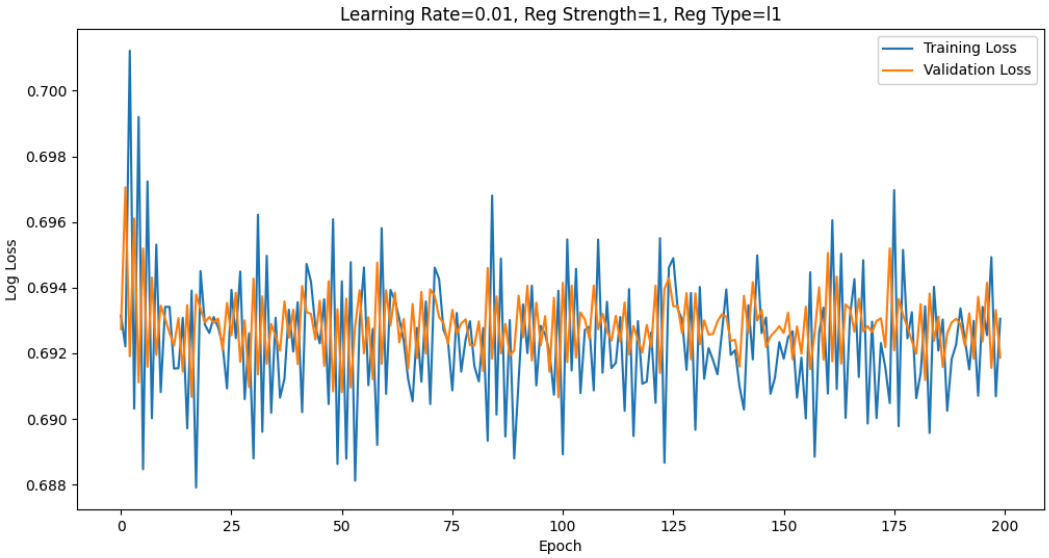

Learning Rate=0.001, Reg Strength=1, Reg Type=l2
Training Loss
Validation Loss
Log Loss
Epoch
0.695
0.690
0.685
0.680
0.675
0.670
0.665
0.660
0.655
0
200
400
600
800
1000

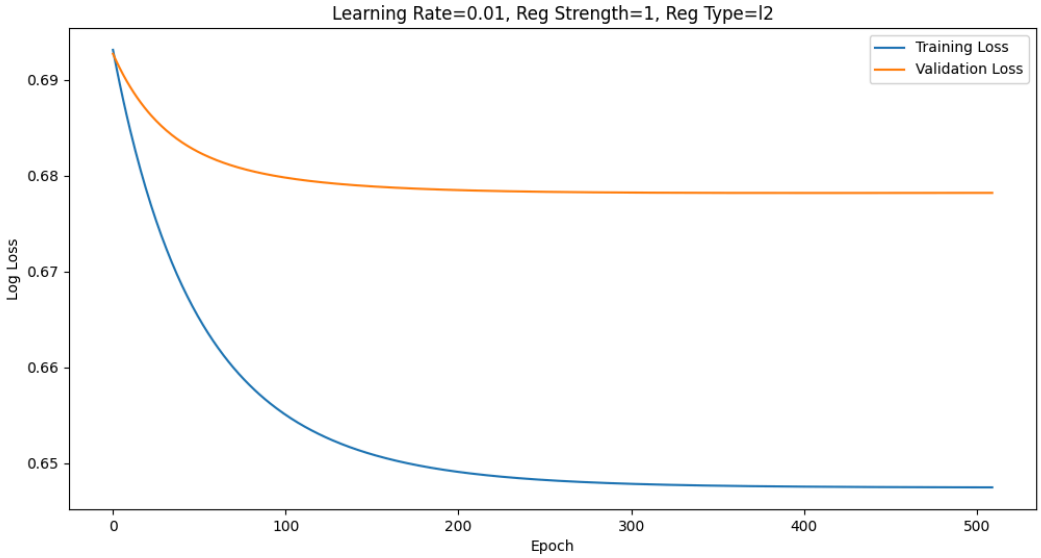

Learning Rate=0.01, Reg Strength=0.01, Reg Type=l1
Training Loss
Validation Loss
Log Loss
Epoch
0.69
0.68
0.67
0.66
0.65
0.64
0.63
0
50
100
150
200

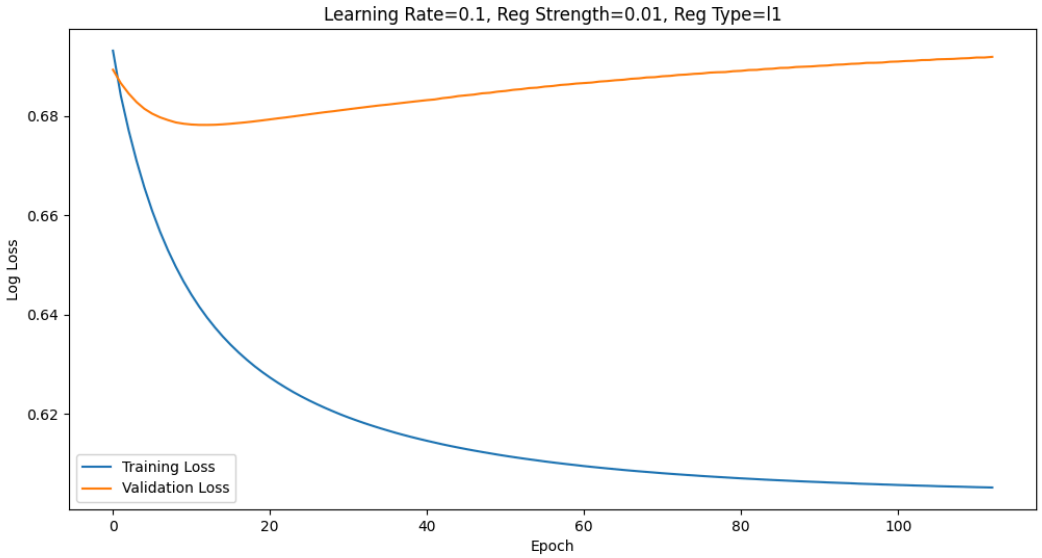

Learning Rate=0.01, Reg Strength=0.01, Reg Type=l2
Training Loss
Validation Loss
Log Loss
Epoch
0.69
0.68
0.67
0.66
0.65
0.64
0.63
0.62
0
50
100
150
200

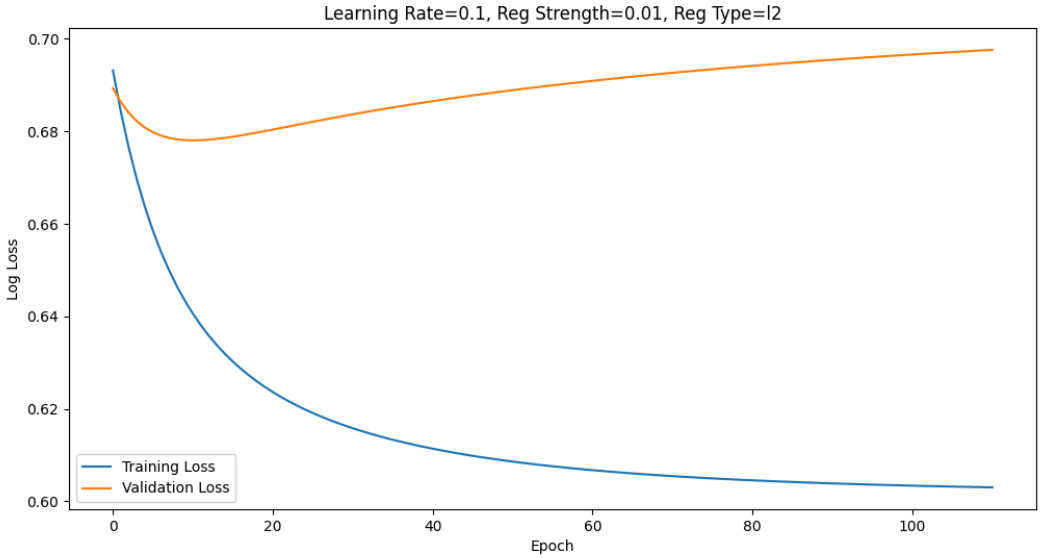

Learning Rate=0.01, Reg Strength=0.1, Reg Type=l1
Training Loss
Validation Loss
Log Loss
Epoch
0.690
0.685
0.680
0.675
0.670
0.665
0.660
0
100
200
300
400
500
600


Learning Rate=0.01, Reg Strength=0.1, Reg Type=l2
Training Loss
Validation Loss
Log Loss
Epoch
0.69
0.68
0.67
0.66
0.65
0.64
0.63
0
50
100
150
200

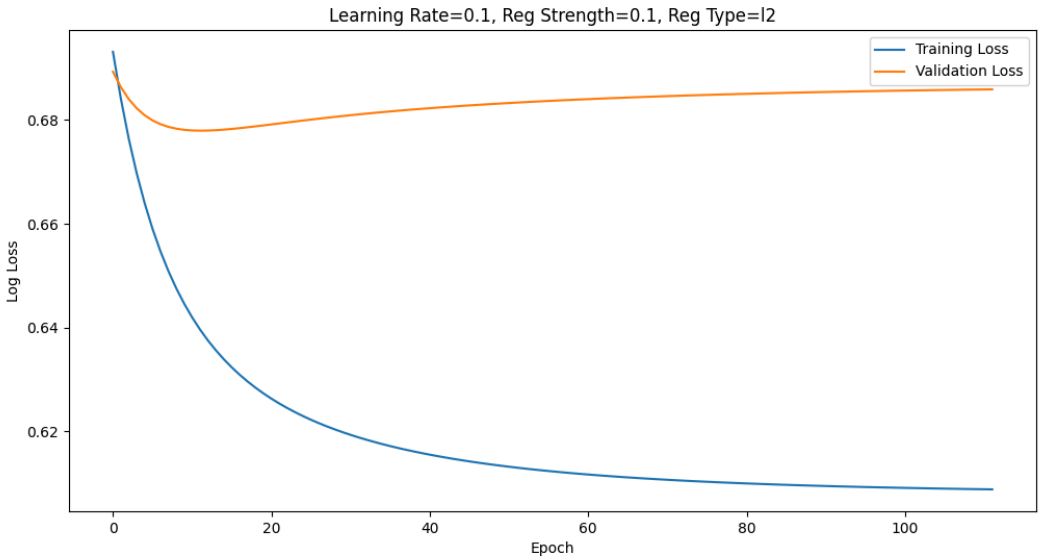

Learning Rate=0.01, Reg Strength=1, Reg Type=l1
Training Loss
Validation Loss
Log Loss
Epoch
0.700
0.698
0.696
0.694
0.692
0.690
0.688
0
25
50
75
100
125
150
175
200

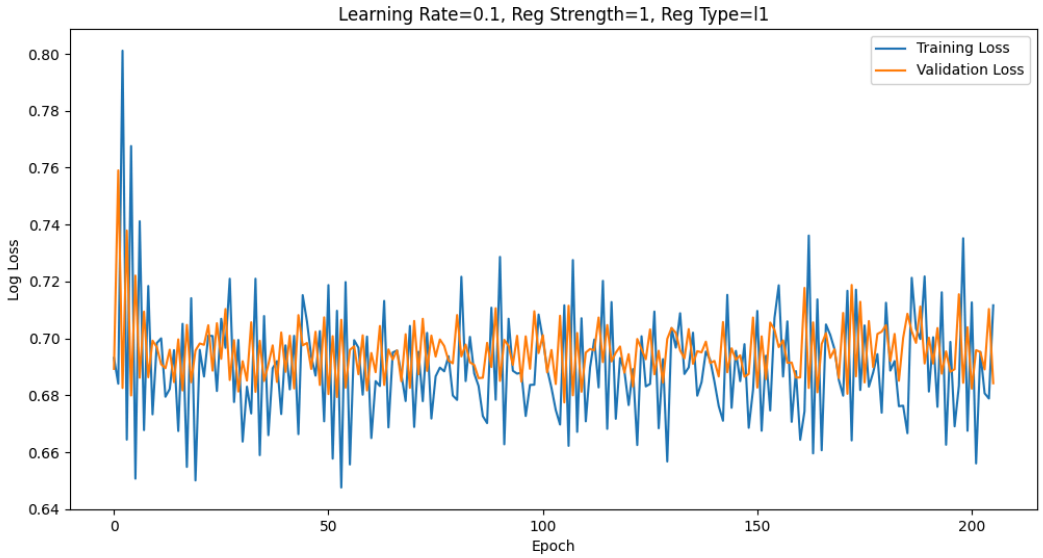

Learning Rate=0.01, Reg Strength=1, Reg Type=l2
Training Loss
Validation Loss
Log Loss
Epoch
0.69
0.68
0.67
0.66
0.65
0
100
200
300
400
500

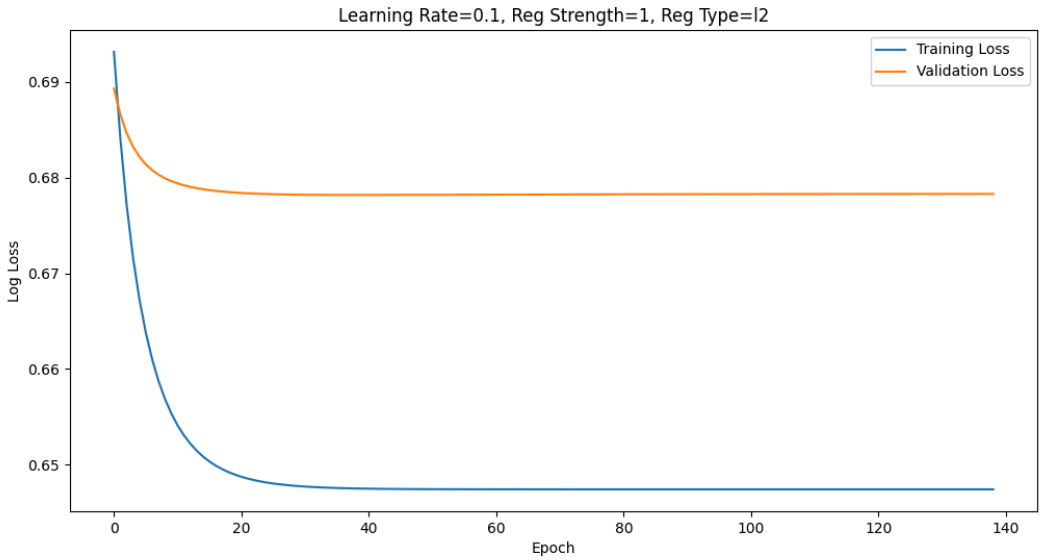

Learning Rate=0.1, Reg Strength=0.01, Reg Type=l1
Log Loss
Epoch
Training Loss
Validation Loss
0.68
0.66
0.64
0.62
0
20
40
60
80
100


Learning Rate=0.1, Reg Strength=0.01, Reg Type=l2
Log Loss
Epoch
Training Loss
Validation Loss
0.70
0.68
0.66
0.64
0.62
0.60
0
20
40
60
80
100


Learning Rate=0.1, Reg Strength=0.1, Reg Type=l1
Training Loss
Validation Loss
Log Loss
Epoch
0.69
0.68
0.67
0.66
0.65
0
25
50
75
100
125
150
175
200


Learning Rate=0.1, Reg Strength=0.1, Reg Type=l2
Training Loss
Validation Loss
Log Loss
Epoch
0.68
0.66
0.64
0.62
0
20
40
60
80
100





With Early Stopping:

- **Observed Issue:** Early Stopping is intended to prevent overfitting by halting training when the validation loss no longer improves. However, in this case, it has caused the model to stop training prematurely, resulting in a model that fails to achieve optimal performance.
- **Impact on Stabilization:** Instead of allowing the model to reach a stable and well-tuned state, Early Stopping may lead to a situation where:
    - **Validation Loss Instability:** The validation loss is not stabilized as expected, and is fluctuating or failing to improve adequately before training stops.

- **Suboptimal Model:** The model is underfitted, as it hasn't had sufficient opportunity to fully learn from the data, leading to poorer performance on unseen data.
- **Improper Patience Setting:** If the patience parameter (the number of epochs to wait before stopping) is not set appropriately, it may cause the training process to halt too early or too late.

Without Early Stopping:

- **Expected Behavior:** Without Early Stopping, the model may continue to improve on the training set but risk overfitting, as indicated by an increasing validation loss.

Effect of Regularization:

- **L1 Regularization**: Can lead to a sparse model by driving some weights to zero, which might be beneficial for feature selection.
- **L2 Regularization**: Helps to prevent large weights by adding a penalty proportional to the square of the weights, promoting smoother models.

Q3
a)
The images are provided in the code file
1: Correlation between features
From the correlation heatmap, it is clear that indoor air and Electricity usage per building are highly correlated, indicating multicollinearity which might need attention in modeling.

2: Skewness in numerical features
The box plot and violin plot reveal that certain numerical features are skewed, possibly indicating the need for transformation (e.g Energy consumption per sqm is left skewed).

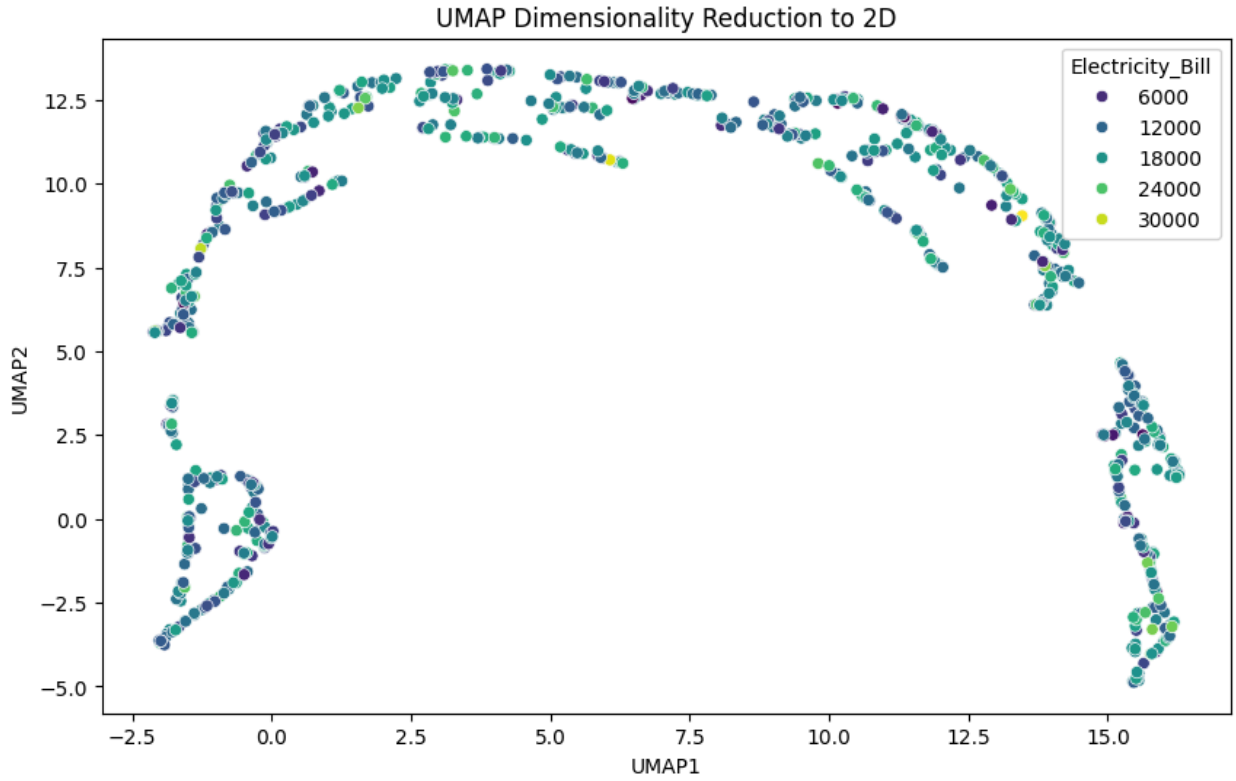
3: Outliers detected in certain features
The box plot highlights the presence of outliers in features like Electricity bill and Maintenance Resolution Time, which may affect the model's performance and could be considered for removal or treatment.

4: Imbalance in categorical features
The count plot shows no significant imbalance in any categorical columns, which indicates there is no need for using of balancing techniques such as SMOTE for better model performance.

5: Non-linear relationships between features

The pair plot suggests non-linear relationships between certain features (e.g., indoor air quality and water usage per building), which might benefit from non-linear models like decision trees.

b)



Separability: There is some distinction between the data points with lower electricity bill amounts (darker colors) and those with higher amounts (lighter colors). This suggests that UMAP has successfully captured some underlying structure in the data that separates the different bill amounts.

Clustering: The data points naturally form clusters based on their electricity bill amounts. The lower bill amounts are clustered on the left side of the plot, while the higher bill amounts are clustered on the right side. This visualization provides strong evidence of the existence of natural groupings within the data.

The UMAP dimensionality reduction has effectively preserved the structure of the data and has made it easier to visualize and identify the distinct clusters based on electricity bill amounts. The plot clearly shows the separation and clustering patterns.

c)

Train MSE: 24475013.16847547

Test MSE: 24278016.155742623

Train RMSE: 4947.222773281538

Test RMSE: 4927.272689403604

Train MAE: 4006.3284693293604

Test MAE: 3842.409312558516

Train R2: 0.013922520844610209

Test R2: 3.7344733075372893e-05

Train Adjusted R2: -0.0011091480449536562

Test Adjusted R2: -0.0640628254763429

Model Performance: The model performs poorly on both training and testing datasets. The metrics indicate that the model fails to capture the relationship between features and the target variable effectively.

Overfitting/Underfitting: Since the MSE, RMSE, and MAE are similar for both training and test sets, overfitting (where the model performs well on the training data but poorly on unseen data) is not a significant issue here. Instead, it appears the model is underfitting, meaning it is too simplistic to capture the complexity of the data.

The model's poor performance across metrics suggests that it is not effectively learning from the data, and further investigation into model choice, features, and hyperparameters is needed to achieve better predictive accuracy.

d)

The Selected Features: ['Building_Type' 'Green_Certified' 'Number_of_Residents']

Train MSE: 24569032.90689799

Test MSE: 23941409.062998377

Train RMSE: 4956.715939702212

Test RMSE: 4892.995918964002

Train MAE: 4006.473377514736

Test MAE: 3813.948128176773

Train R2: 0.010134545491283897

Test R2: 0.01390151386794114

Train Adjusted R2: 0.007153023037944406

Test Adjusted R2: 0.0018759225736477703

- The selected features slightly improved the MSE, RMSE, MAE, R^2, and Adjusted R^2 scores on the test dataset compared to the original features.
- The train metrics are slightly worse with the selected features, but the improvements in the test metrics indicate that the selected features may provide a better generalization to unseen data.
- Adjusted R^2, in particular, shows improvement, which suggests that the selected features are more appropriate for the model, considering the complexity of the dataset.

Overall, the selected features seem to improve the model's performance on the test dataset, providing a better fit to the unseen data.

e)

Train MSE: 24188925.91329433

Test MSE: 24129688.93419977

Train RMSE: 4918.223857582565

Test RMSE: 4912.197973840201

Train MAE: 3976.6837323582586

Test MAE: 3797.462833733082

Train R2: 0.02544873320926233

Test R2: 0.006146644659019929

Train Adjusted R2: 0.006554371914339829

Test Adjusted R2: -0.07595428469523502

- **Overall Performance**: Applying One-Hot Encoding followed by Ridge Regression slightly improved the performance on the train and test datasets compared to the original dataset.
- **MSE and RMSE**: Both metrics show slight improvements, indicating that the model is performing marginally better with One-Hot Encoding and Ridge Regression.
- **MAE**: Shows improvement in both train and test datasets.
- **R^2 and Adjusted R^2**: The R^2 score increased, suggesting better model performance. However, the Adjusted R^2 score on the test dataset is slightly worse with One-Hot Encoding and Ridge Regression.

Overall, One-Hot Encoding along with Ridge Regression appears to enhance the model's performance, especially in reducing the MSE, RMSE, and MAE, and improving the R^2 score. However, there is a minor trade-off in the Adjusted R^2 score on the test set.

f)

=== ICA with 4 components ===
Train MSE: 24589773.930624887
Test MSE: 24232749.731079873
Train RMSE: 4958.807712608434
Test RMSE: 4922.67708986481
Train MAE: 3978.097434702609
Test MAE: 3802.1953103700625
Train R2: 0.009298907273198487
Test R2: 0.0019017777209204834
Train Adjusted R2: 0.005316189312487674
Test Adjusted R2: -0.014393703459146145

=== ICA with 5 components ===
Train MSE: 24588480.246258456
Test MSE: 24254297.990001556
Train RMSE: 4958.67726780625
Test RMSE: 4924.86527633006
Train MAE: 3978.746224466576
Test MAE: 3804.2047264950625
Train R2: 0.009351028716814813
Test R2: 0.0010142482757846683
Train Adjusted R2: 0.004367884998086491
Test Adjusted R2: -0.019456771226760816

=== ICA with 6 components ===
Train MSE: 24587094.91133023
Test MSE: 24236046.1667434
Train RMSE: 4958.537577888286
Test RMSE: 4923.011899918931
Train MAE: 3978.878636668621
Test MAE: 3802.738812113786
Train R2: 0.009406842683671868
Test R2: 0.001766004166035673
Train Adjusted R2: 0.003421385539766564
Test Adjusted R2: -0.022881748817519032

=== ICA with 8 components ===
Train MSE: 24379674.015889145
Test MSE: 24235407.506404456
Train RMSE: 4937.577747832346
Test RMSE: 4922.947034694204
Train MAE: 3976.0493368931047
Test MAE: 3800.508170901315
Train R2: 0.017763654273221685
Test R2: 0.0017923093008640478
Train Adjusted R2: 0.009834400220936845
Test Adjusted R2: -0.03134321570159693

1. **Mean Squared Error (MSE)**
    - **Train**: Lowest with 8 components (24,379,674.02)
    - **Test**: Slightly lower with 8 components (24,235,407.51)
2. **Root Mean Squared Error (RMSE)**
    - **Train**: Lowest with 8 components (4,937.58)
    - **Test**: Slightly lower with 8 components (4,922.95)
3. **Mean Absolute Error (MAE)**

    - **Train**: Lowest with 8 components (3,976.05)
    - **Test**: Lowest with 8 components (3,800.51)
4. **R² Score**
    - **Train**: Highest with 8 components (0.0178)
    - **Test**: Highest with 8 components (0.0018)
5. **Adjusted R² Score**
    - **Train**: Highest with 8 components (0.0098)
    - **Test**: Slightly worse with 8 components (-0.0313)

Conclusion

- **ICA with 8 components** generally provides the best performance among the different component choices. It results in the lowest MSE, RMSE, and MAE for both train and test datasets.
- The R² and Adjusted R² scores are also highest with 8 components for the train dataset, indicating better explained variance in the model.
- For the test dataset, while the R² score is slightly improved with 8 components, the Adjusted R² score is slightly worse compared to some other component choices. However, the overall metrics suggest that 8 components perform better overall.

selecting the number of components should balance model complexity and performance.

g)
=== ElasticNet with alpha=0.1 ===
Train MSE: 24190404.855616156
Test MSE: 24108417.937015373
Train RMSE: 4918.374208579107
Test RMSE: 4910.0323763714
Train MAE: 3976.0822914657842
Test MAE: 3796.386129678228
Train R2: 0.025389147880074403
Test R2: 0.007022754250851304
Train Adjusted R2: 0.006493631359381968
Test Adjusted R2: -0.07500580083277408

=== ElasticNet with alpha=0.5 ===
Train MSE: 24213520.011309635
Test MSE: 24063500.556266405
Train RMSE: 4920.723525185055
Test RMSE: 4905.456202665192
Train MAE: 3975.2409206961875
Test MAE: 3793.834544489474
Train R2: 0.02445785790283883
Test R2: 0.008872810821917154
Train Adjusted R2: 0.005544285760138767
Test Adjusted R2: -0.07300291350148957

=== ElasticNet with alpha=1 ===
Train MSE: 24255567.395840995
Test MSE: 24051287.377304178
Train RMSE: 4924.994151858557
Test RMSE: 4904.211188081543
Train MAE: 3976.290815429076
Test MAE: 3795.203184932364
Train R2: 0.022763804516295005
Test R2: 0.00937584709909789
Train Adjusted R2: 0.0038173884814068915
Test Adjusted R2: -0.07245832205358549

=== ElasticNet with alpha=5 ===
Train MSE: 24496564.124766845
Test MSE: 24148101.960715506
Train RMSE: 4949.400380325565
Test RMSE: 4914.071831049634
Train MAE: 3987.505379947928
Test MAE: 3810.510010664519
Train R2: 0.013054251131866756
Test R2: 0.005388249130815925
Train Adjusted R2: -0.006080411346188974
Test Adjusted R2: -0.07677533028881234

=== ElasticNet with alpha=10 ===
Train MSE: 24615416.58607818
Test MSE: 24221960.53095781
Train RMSE: 4961.39260551694
Test RMSE: 4921.581100719342
Train MAE: 3995.282415275215
Test MAE: 3822.0436974549807
Train R2: 0.008265786478771187
Test R2: 0.002346163173712612
Train Adjusted R2: -0.01096171357929343
Test Adjusted R2: -0.08006871899889378

- **ElasticNet with α=0.5** generally performs the best on most metrics for the test dataset, showing the lowest MSE, RMSE, and MAE.
- **ElasticNet with α=1** provides the highest R^2 and Adjusted R^2 scores on the test dataset, indicating slightly better variance explanation but still quite low.
- **Higher values of α**(e.g., 5 and 10) tend to degrade model performance, as evidenced by increasing MSE, RMSE, and MAE.

The choice of α should balance the trade-off between model complexity and predictive performance. Here, α=0.5 seems to offer the best overall performance on the test dataset.

h)
=== Gradient Boosting Regressor ===
Train MSE: 15548098.780395458
Test MSE: 24763212.870917924
Train RMSE: 3943.1077566299728
Test RMSE: 4976.26495184068
Train MAE: 3155.777526146695
Test MAE: 3837.5378752506076
Train R2: 0.37358031452342877
Test R2: -0.019946932092554936
Train Adjusted R2: 0.3614354430703116
Test Adjusted R2: -0.10420341778715736

- **ElasticNet models** generally showed higher MSE and RMSE compared to the Gradient Boosting Regressor and the original regression models. The R2 scores and Adjusted R2 scores for ElasticNet are generally lower, indicating poorer fit and predictive performance.
- **Gradient Boosting Regressor** performed better on the training set (lower MSE and RMSE) but had worse performance on the test set compared to the ElasticNet models. The R2 score for the Gradient Boosting Regressor on the test set was negative, indicating that the model performed worse than a simple mean-based prediction.
- **Original Regression Model** (from part (c)) had more consistent metrics, but still showed room for improvement compared to more advanced models.

In conclusion, the Gradient Boosting Regressor, while fitting the training data better, did not generalize as well to the test data compared to the ElasticNet models. ElasticNet with lower alpha values generally performed better, though none of the models achieved high performance.