CYBER LABS

JANUARY 2025

PROJECT REPORT

PRESENTED TO

Machine Learning Division

PRESENTED BY

Arnav Tripathi

TABLE OF CONTENTS

CYBER LABS

DECEMBER 2024

PROJECT REPORT

LINEAR REGRESSION (MULTIPLE LINEAR REGRESSION)

PRESENTED TO

Machine Learning Division

PRESENTED BY

Arnav Tripathi
24JE0090

COMMENCEMENT OF PROJECT

Starting the project by downloading a dataset from 'Kaggle' for implementing Linear Regression.

The dataset forms the foundation for building and testing the Linear Regression model.

It has been strategically decided that the pre-developed model will undergo a comprehensive training process utilizing the official dataset once it becomes available.

This approach ensures that the model will be fine-tuned and time is utilized to conduct extensive evaluations and assessments, ensuring the model meets the desired level.

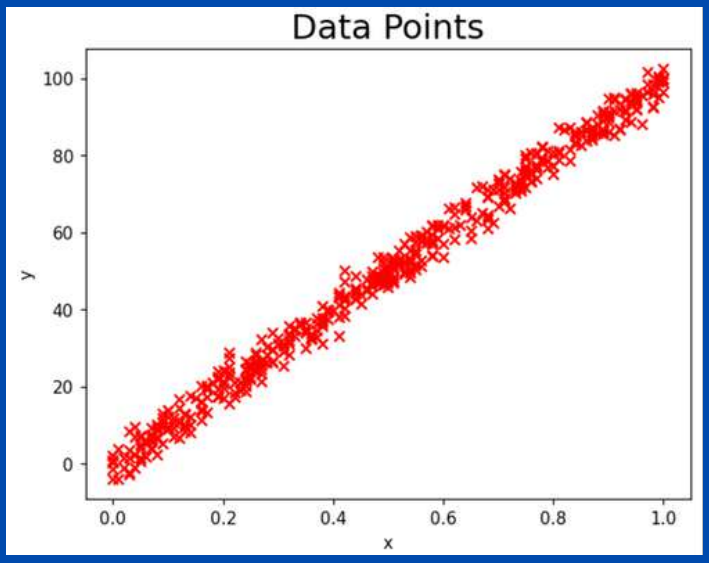
WORKING ON THE MODEL

(Utilizing the downloaded dataset)

1. Dataset Partitioning: The dataset was divided into training and testing subsets with an approximate ratio of 7:1 to ensure balanced model evaluation.

2. Data Preparation: The data was organized into NumPy arrays to facilitate efficient computational operations.

3. Data Visualization: The training dataset was visualized through plots to gain insights and inform the strategic planning of the model.



4. Definition of Gradient and Cost Functions: The gradient and cost functions were defined, followed by the implementation of the gradient descent optimization algorithm.

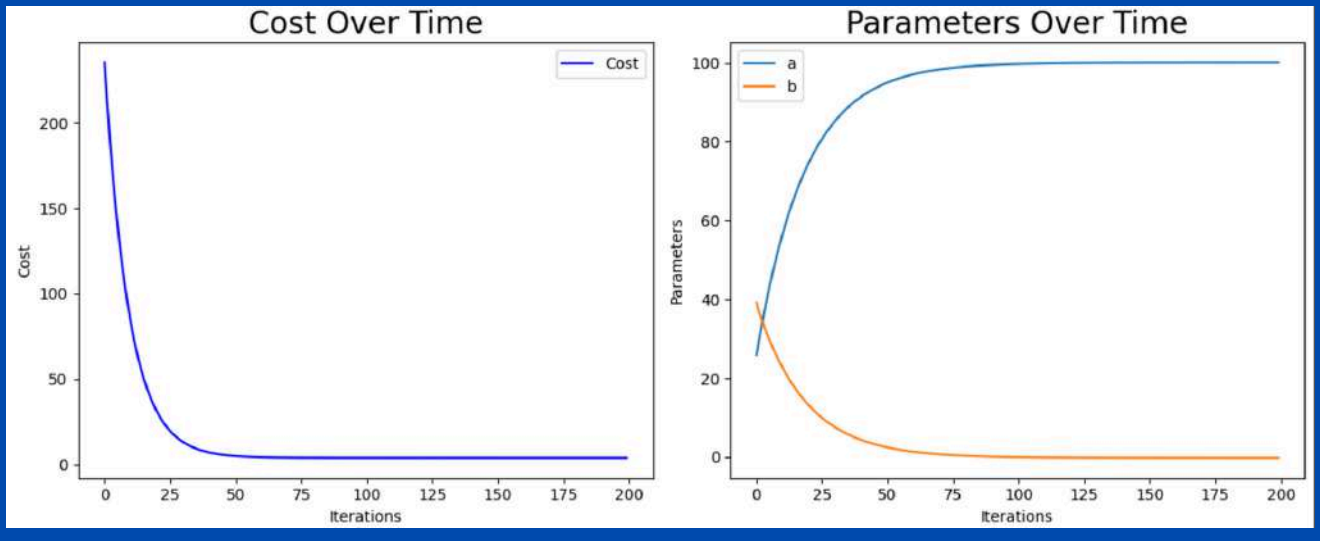
5. Tracking Cost and Parameters: During the execution of gradient descent, both cost values and model parameters were recorded to facilitate the plotting of the learning curve and parameter progression over iterations.

6. Addressing Cost Overflow: An issue with cost overflow was encountered despite testing various learning rates. To mitigate this, feature scaling was applied to the feature x, which helped stabilize the cost function and prevent divergence.

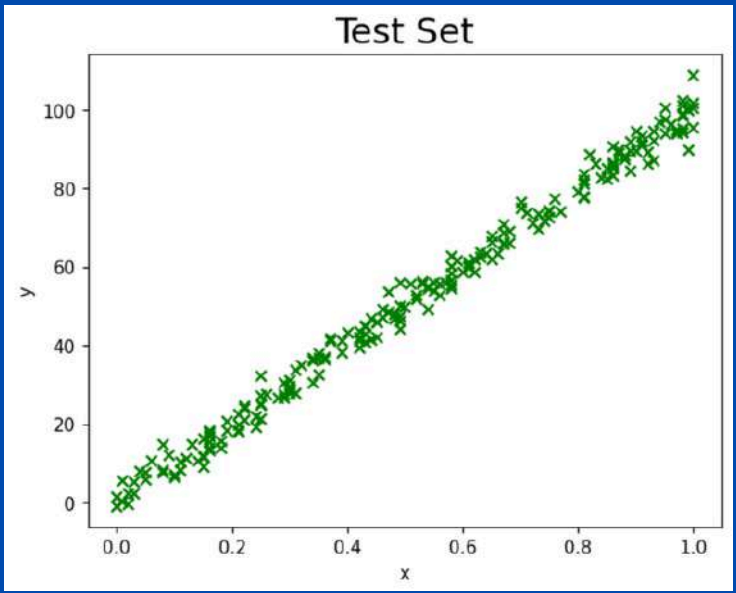
7. Incremental Adjustment of Learning Rate: The learning rate was gradually increased, starting from 0.01, to identify the optimal value for stable convergence.

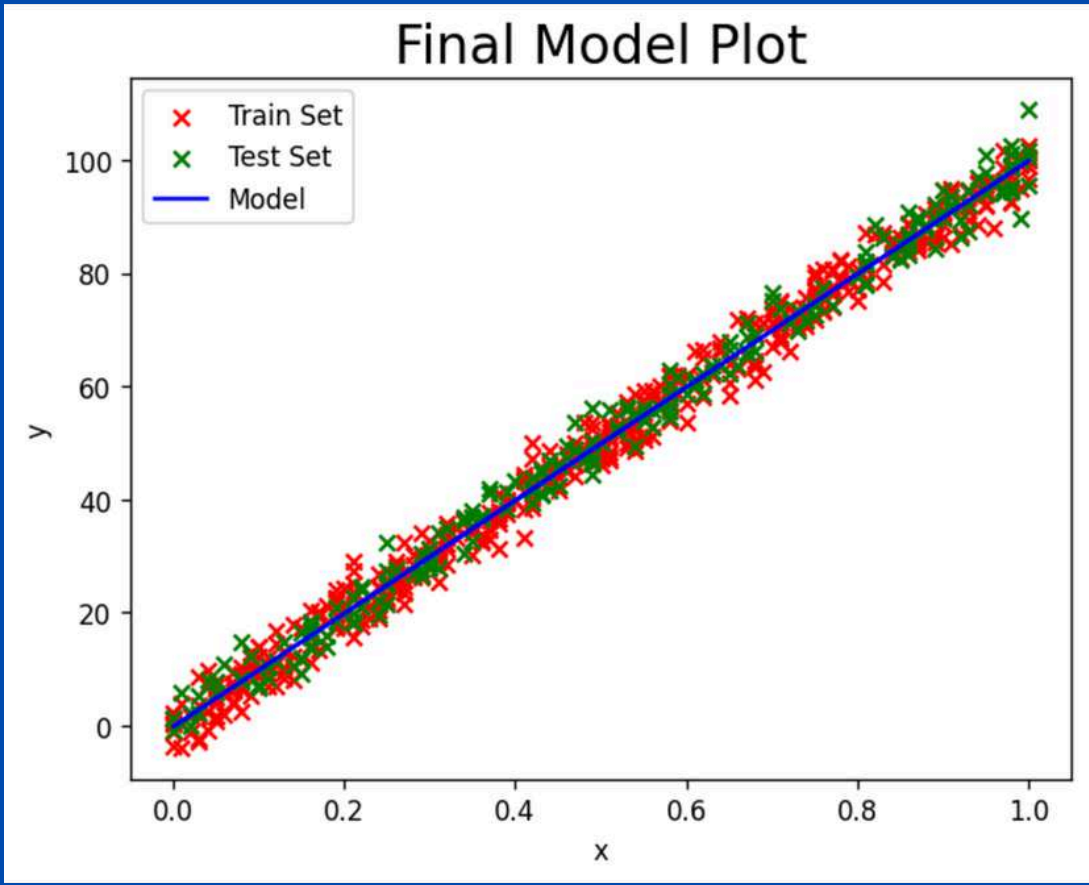
8. Optimal Learning Rate Selection: At a learning rate of approximately 1.0, the parameters vs. iterations graph revealed oscillatory behavior, with parameters fluctuating before converging. Therefore, the learning rate was adjusted to 0.80, resulting in smoother convergence and more stable parameter values.

9. Completion of Training: Both the cost and parameters vs. iterations curves exhibited smooth trajectories and stabilized over time. The training process was considered complete once the cost dropped below the threshold of 4. The final estimated training cost was approximately 3.9129.



10. Evaluation of Test Cost: The estimated test cost was approximately 4.0063, which is comparable to the final training cost of 3.9129. This indicates that the model generalizes well to unseen data; consistent performance across both training and testing datasets.





WORKING ON THE MODEL

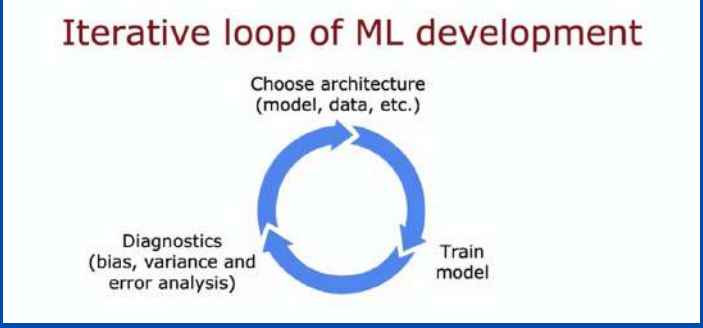
(Utilizing the official dataset)

Problem Statement:
Develop a linear regression model to predict the target variable using the given dataset with 25 features. The objective is to analyze the relationship between the features and the target variable and create an accurate predictive model.

1. Data Partitioning: The official dataset was divided into three distinct subsets: a training set, a cross-validation set, and a test set, following a typical 80%-10%-10% split. This ensures that each subset serves its intended purpose, and enough data is available for training the model.

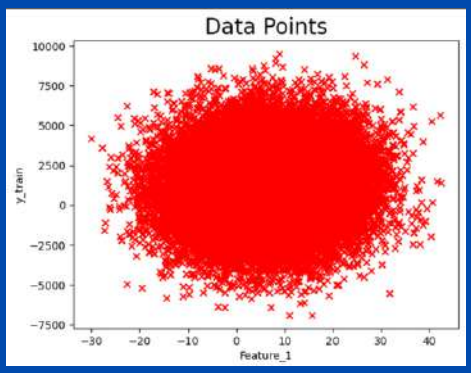
2. Identifying the Type of Target Variable : The task of the provided dataset is to predict a continuous variable rather than a categorical value. Therefore, a linear regression approach would be an appropriate model for this scenario (as given in the problem statement), as opposed to any classification method.

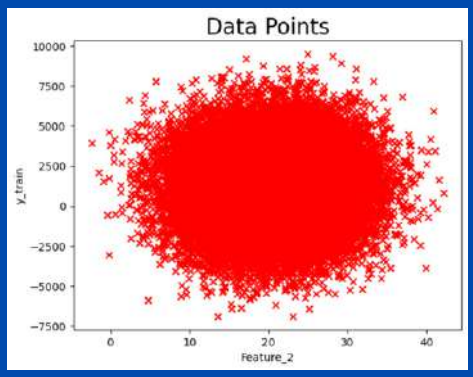
3. Model Training Loop: The model training process was strategically planned to follow the loop outlined below for successful development.
This approach was designed to ensure consistent training and accurate evaluation of the model.

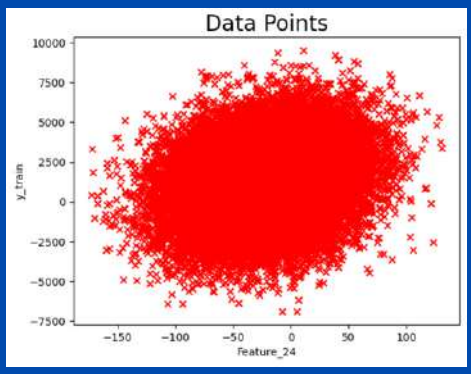


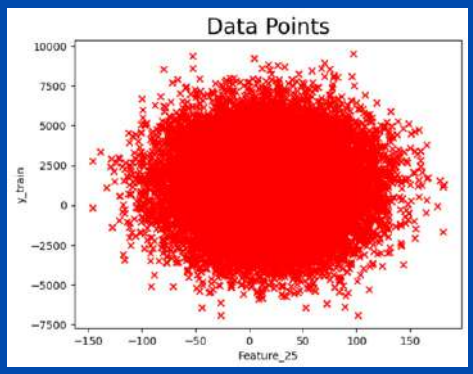
4. Data Organization: The training set was organized into NumPy arrays to facilitate efficient data manipulation and model operations.

5. Feature vs Target Visualization: The features from the train set were plotted against the target variable to inspect potential patterns or relationships visually. This plotting was intended to provide insights into how the features correlated with the target, which could aid in model development and understanding of the data.









6. Plotting Observations and Limitations: All scatter plots show a circular or elliptical spread of points, Some scatter plots showed trends such as an increasing or decreasing target variable with an increasing or decreasing feature, while for others, the relationship wasn't clear.

Despite this, they provide an intuitive sense of how a linear regression model might fit the data, even though the relationships are not immediately apparent.

Also, the plots provided insights into the range of values for the features, highlighting the variation in their scales. This highlights the need for normalization to ensure equal feature contributions during training.

7. Z-Score Normalization: Z-score normalization was applied to the training set to scale the features such that each column has a mean of 0 and a standard deviation of 1. This transformation ensures that no feature dominates the model due to differing scales, speeds up gradient descent and helps prevent numerical instability during the computation of the cost function. The normalization process will be similarly applied to the test set later, ensuring consistency between the training and evaluation stages of the model.

8.Model Selection: We begin by selecting the initial model as a simple multiple linear regression hypothesis: $f_{ab} = a_1.x_1 + a_2.x_2 + \ldots + a_{25}.x_{25} + b$
where $a_1, a_2, \ldots, a_{25}$ are the coefficients for the features $x_1, x_2, \ldots, x_{25}$, and b is the intercept term.

9. Vectorized Approach: Gradient computation took significantly longer than cost computation, so a more vectorized approach was implemented instead of using a for loop.

Defining Gradient

```
def gradient(x_train, y_train, a, b):
  m = x_train.shape[0]
  n = x_train.shape[1]
  dJ_da = np.zeros(n)
  dJ_db = 0

  for i in range(m):
    f_ab = np.dot(a, x_train[i]) + b

    for j in range(n):
      dJ_da[j] += (f_ab - y_train[i]) * x_train[i, j]

    dJ_db += (f_ab - y_train[i])

  dJ_da = dJ_da / m
  dJ_db = dJ_db / m
  return dJ_da, dJ_db                    # gradients
```

BEFORE

Defining Gradient

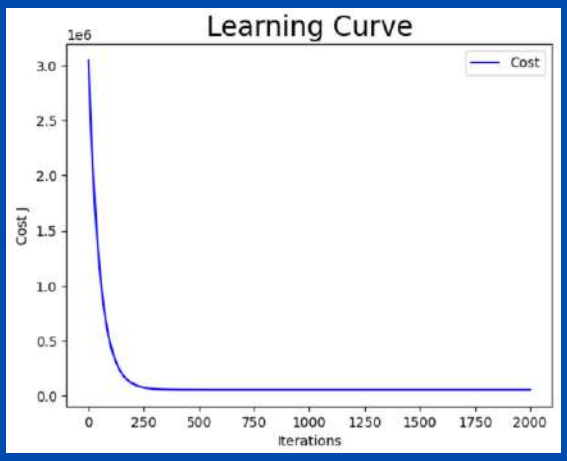
```
[13] def gradient(x_train, y_train, a, b):
       m = x_train.shape[0]
       n = x_train.shape[1]

       f_ab = np.dot(x_train, a) + b
       dJ_da = (1 / m) * np.dot(x_train.T, (f_ab - y_train))
       dJ_db = (1 / m) * np.sum((f_ab - y_train))

       return dJ_da, dJ_db
```

AFTER

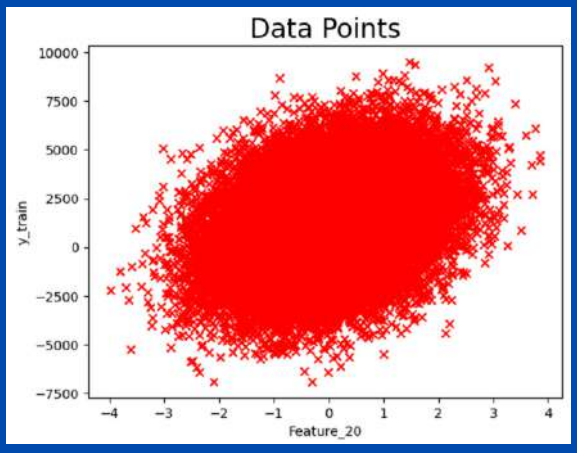
10. Issue with High Cost Despite Gradient Descent Optimization: After performing gradient descent with multiple learning rates (starting at 0.01) and varying the number of iterations, the cost remained significantly high, with a final value of 57947.5534 .



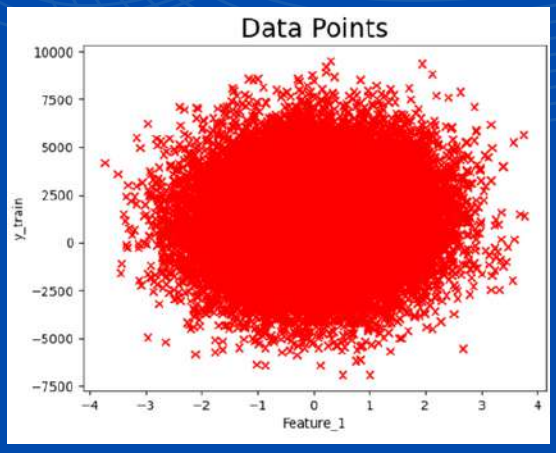

```
Iteration      0: Cost 3043863.7419
Iteration    200: Cost 113014.3971
Iteration    400: Cost 58967.1263
Iteration    600: Cost 57966.5023
Iteration    800: Cost 57947.9068
Iteration   1000: Cost 57947.5600
Iteration   1200: Cost 57947.5535
Iteration   1400: Cost 57947.5534
Iteration   1600: Cost 57947.5534
Iteration   1800: Cost 57947.5534
TRAIN COST estimated is =57947.5534
Time taken = 3.30657958984375 seconds
```

Estimated Cross Validation cost was 58699.4558 , there is a High Bias problem. The current approach wasn't working and the model needed modifications to get better results.

11. Checking again for trends in features: As mentioned earlier, while some features exhibited clear trends, others did not. Therefore, a method was needed to analyze the performance of the model on a feature-by-feature basis.

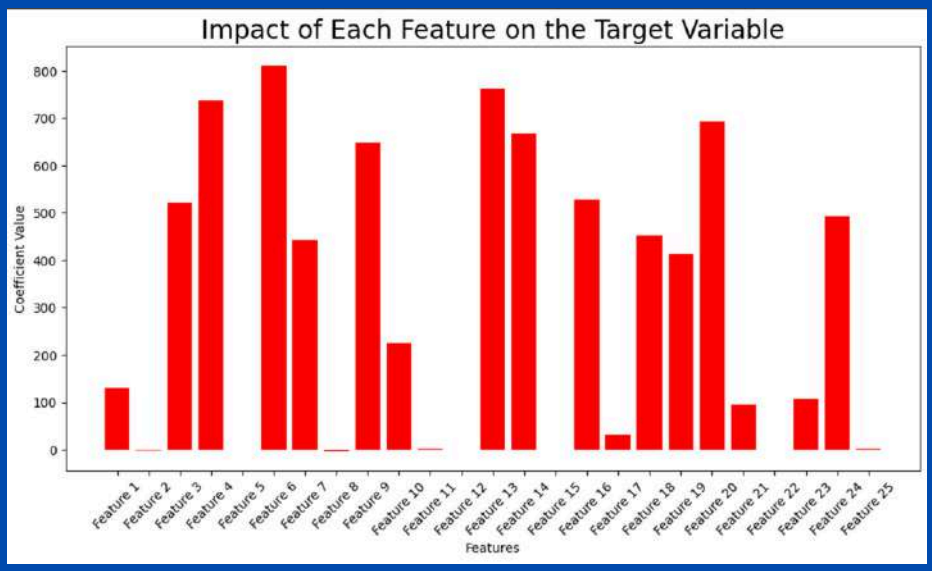


Trend can be easily observed (increasing)



Trend cannot be easily observed through plot.

12. Visualising Feature Impacts: A bar graph was plotted to visualize the impact of each individual feature on the target variable, after running gradient descent once.



13. Conclusion : It can be observed from the bar graph that Feature 6 has the highest impact (as its coefficient takes the highest value), while Features 2, 5, 8, 11, 12, 15, 22 and 25 have much lower impact. And, for these features the trend in the Feature vs Target plot was unclear too.

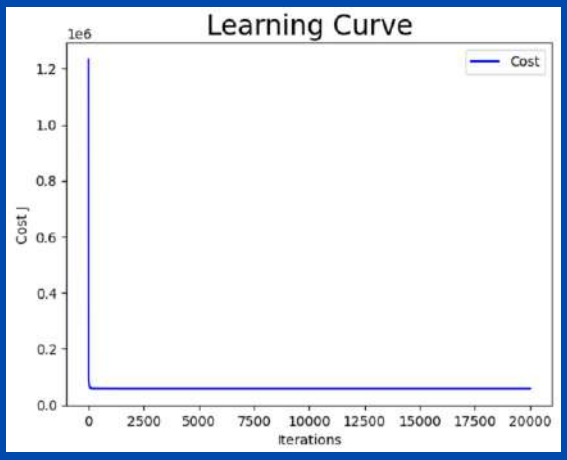
14. Deciding what to try Next: Deciding what to try next was crucial, as the model required a significant adjustment.

1 5. Feature Engineering: I decided to enhance the model by introducing interaction features derived from the most impactful existing features. To guide this process, I revisited and analyzed the bar graph for reference.

16. Interacting Features: I created interaction features by multiplying the most impactful features with Feature_6, which had the highest impact among all. This process increased the total number of features to 36, providing the model with richer representations and enabling it to better capture complex relationships within the data.

17. Issue with High Cost: This above approach didn't prove to be effective, after increasing features and even regularising the features with low impact, The Train Cost remained significantly high and overflowed for a learning rate more than 0.3 with number of iterations set to 20,000 .

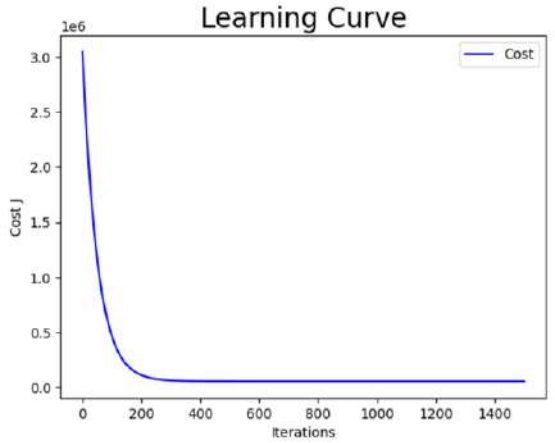
The estimated Train Cost was 57996.361154 .
The estimated Cross Validation Cost was 58802.4313565 .




```
CPU times: user 4 µs, sys: 0 ns, total: 4 µs
Wall time: 8.11 µs
Iteration      0: Cost 1233039.6750
Iteration   2000: Cost 57997.8432
Iteration   4000: Cost 57996.3832
Iteration   6000: Cost 57996.3618
Iteration   8000: Cost 57996.3612
Iteration  10000: Cost 57996.3612
Iteration  12000: Cost 57996.3612
Iteration  14000: Cost 57996.3612
Iteration  16000: Cost 57996.3612
Iteration  18000: Cost 57996.3612
TRAIN COST estimated is = 57996.361154
```

18. Judging by Absolute Error: I felt that the high cost might not be due to a high bias problem, but rather a result of the data itself, as the cost wasn't decreasing despite trying various approaches. Therefore, I switched from using mean squared error to mean absolute error. This will offer a more direct measure of how far the model is from achieving optimal performance on the data.

19. Training Result: The estimated training MAE was 272.3557, indicating that the model was performing reasonably well. Given that the values of y_train were quite large (in the thousands), this level of mean absolute error is considered acceptable.




```
CPU times: user 3 µs, sys: 0 ns, total: 3 µs
Wall time: 7.15 µs
Iteration      0: Cost 3043863.7419
Iteration    150: Cost 207320.7718
Iteration    300: Cost 65436.9198
Iteration    450: Cost 58323.8748
Iteration    600: Cost 57966.5023
Iteration    750: Cost 57948.5095
Iteration    900: Cost 57947.6017
Iteration   1050: Cost 57947.5558
Iteration   1200: Cost 57947.5535
Iteration   1350: Cost 57947.5534
TRAIN COST estimated is = 57947.5534
```

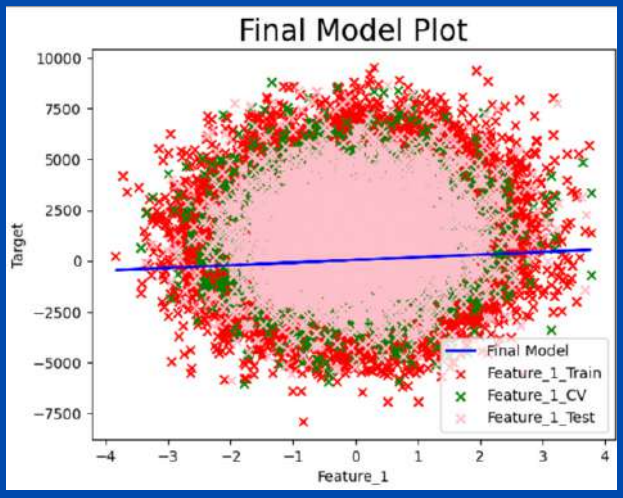
20. Hyperparameter Tuning: I adjusted the hyperparameters like learning rate (0.5) and number of iterations (10000) so the cost converges precisely and quickly.

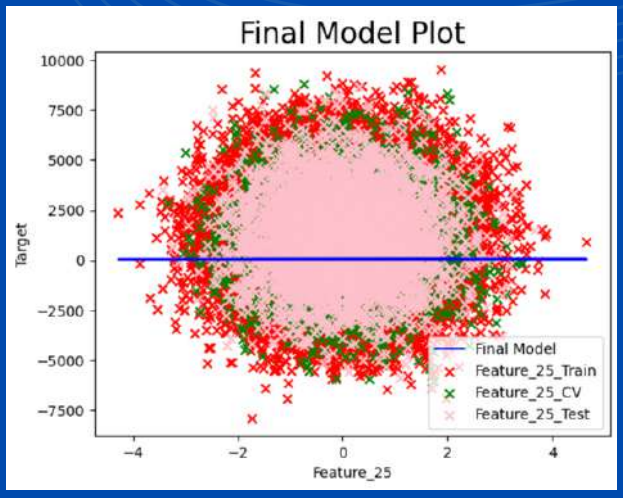
21. Further Results:

The estimated Cross Validation Set MAE was 274.2360
The estimated Test Set MAE was 273.0987 .

The R2 score of Test Set was 97.46% .
This confirmed the model's performance, ensuring that it captured the major variance in the data and generalized well to unseen data.

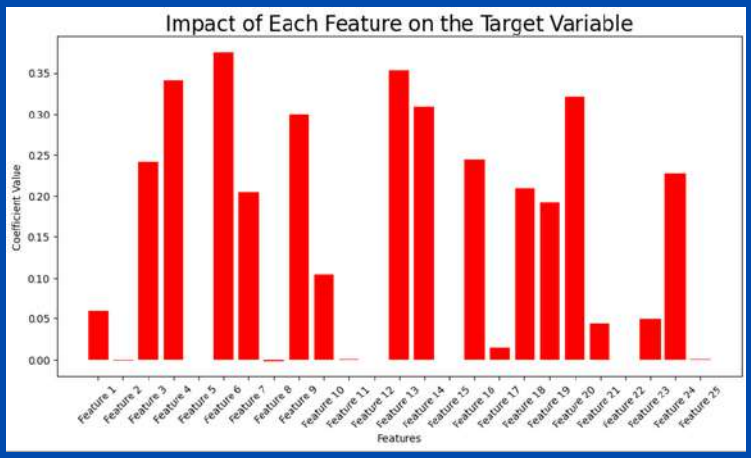
Hence, Training was complete.





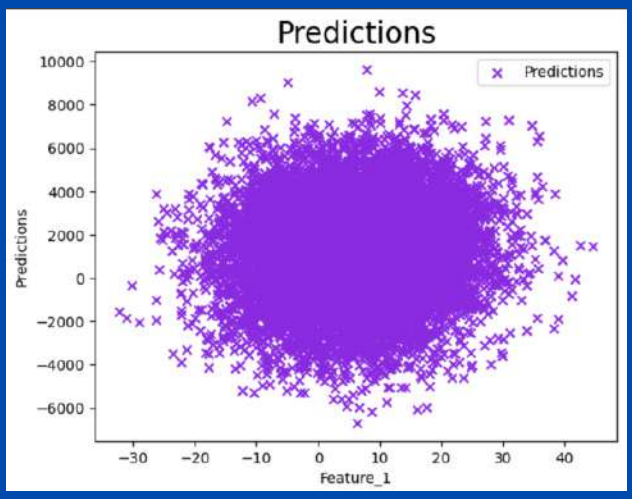
Note: The intercept for these plots has been taken as (b_final/n) for individual features, which is just an approximation used for visualization purposes. These plots show the contributions of individual features to the target variable, but under simplified assumptions. While the model was trained with all features together, these plots isolate each feature and do not capture the full joint behavior of the multiple linear regression model, which accounts for the interactions among all features. Thus, these plots are intended solely for interpretability and have limited physical significance

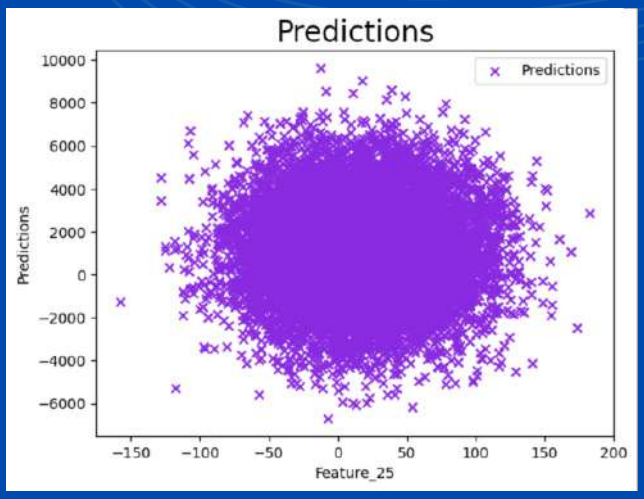
22. Visualizing Feature Impacts: A bar graph was plotted to visualize the impact of each individual feature on the target variable.

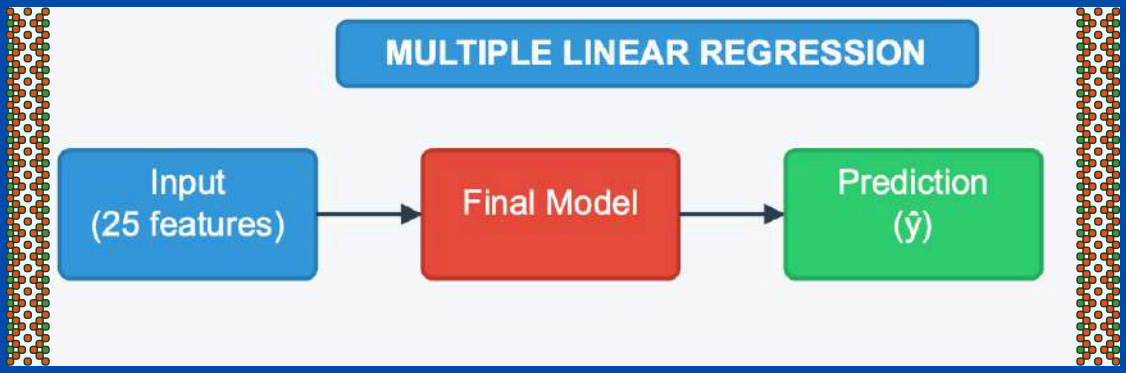


It can be observed from the bar graph that Feature 6 has the highest impact (as its coefficient takes the highest value), while Features 5, 12, 15 and 22 have a low impact.

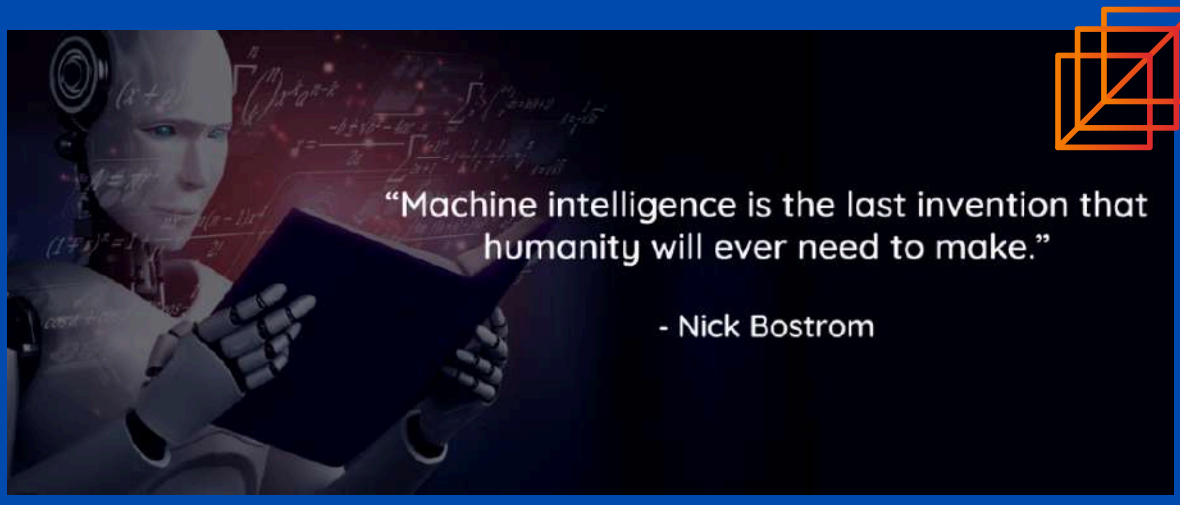
23. Predictions for Test Set: Predictions for the official linear_regression_test.csv were computed using the trained model and saved as a CSV file in Google Drive.







>>>Flow Chart showing working of Final Model



CYBER LABS

DECEMBER 2024

PROJECT REPORT

POLYNOMIAL REGRESSION

PRESENTED TO

Machine Learning Division

PRESENTED BY

Arnav Tripathi
24JE0090

COMMENCEMENT OF THE MODEL

Problem Statement:

Develop a Polynomial Regression Model that can predict a target variable using a dataset with five input features. The model should be designed to capture non-linear relationships by including polynomial terms of the input features, ensuring improved predictive performance compared to a simple linear regression model.

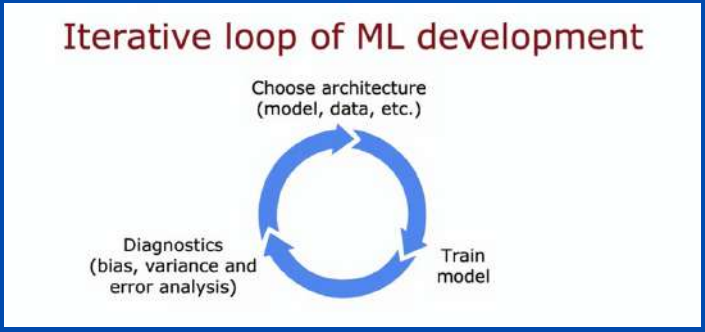
WORKING ON THE MODEL

(Utilizing the official dataset)

1. Data Partitioning: The official dataset was divided into 80% for training, 20% for cross-validation (CV), and 20% for testing. The model is trained on the training set and evaluated using both the CV set and test set.

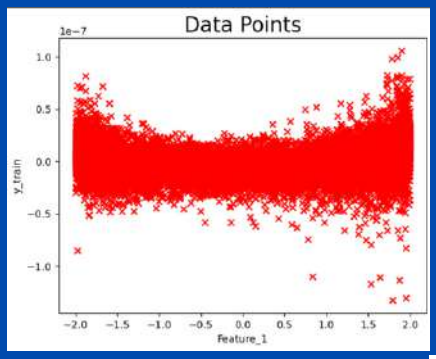
2. Identifying the Type of Target Variable : The task of the provided dataset is to predict a continuous variable rather than a categorical value. Therefore, a polynomial regression approach would be an appropriate model for this scenario (as given in the problem statement), as opposed to any classification method.

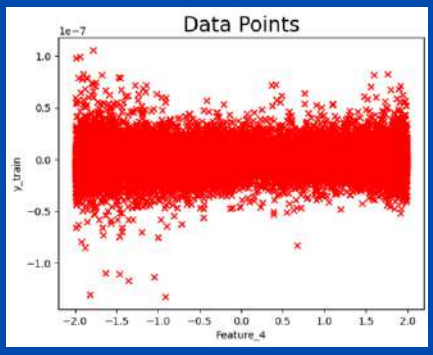
3. Iterative Refinement and Training Loop: The development and training process will follow an iterative loop to ensure systematic progression and refinement of the polynomial regression model. Each iteration will address critical aspects of model development, from feature preprocessing to evaluation. The loop is formally described below :

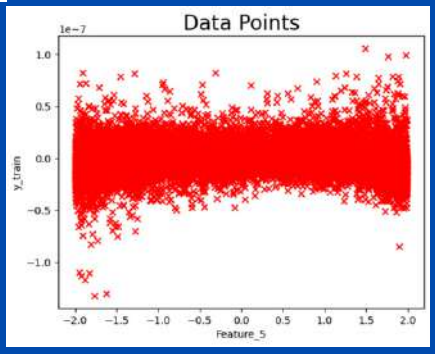


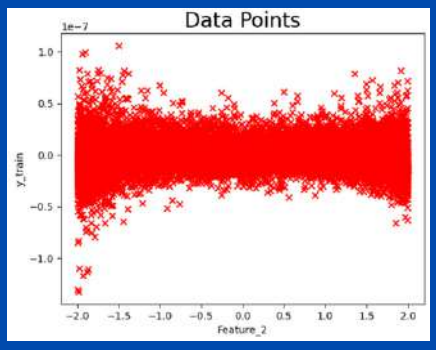
4. Data Organization: The training set was organized into NumPy arrays to facilitate efficient data manipulation and model operations.

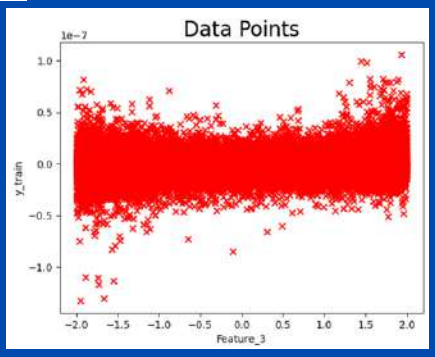
5. Feature vs Target Visualization: The features from the train set were plotted against the target variable to inspect potential patterns or relationships visually. This plotting was intended to provide insights into how the features correlated with the target, which could aid in model development and understanding of the data.







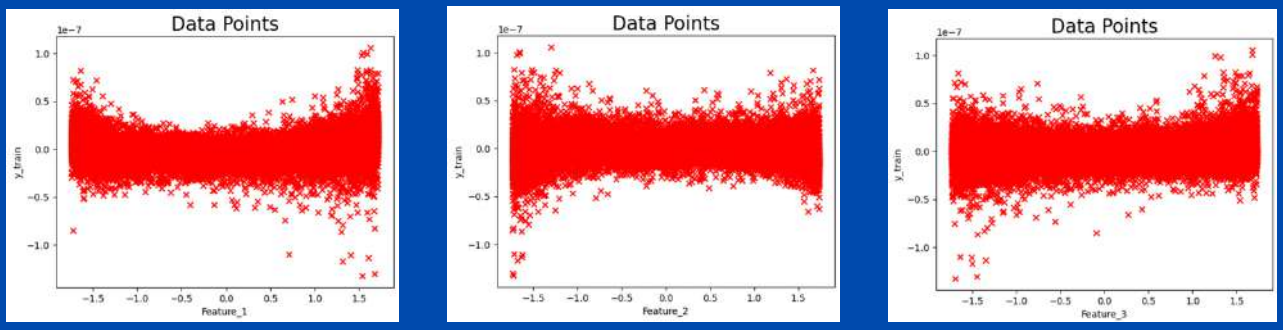




6. Plotting Observations : The scatter plot of features against target variable reveals a non-linear pattern. The points exhibit a curved pattern, especially at the extremes of features (around -2 and 2), where the spread of y_train increases. This confirms that polynomial regression is required to capture the underlying relationship.

The plots also highlight that the target variable takes very low values.

7. Feature Normalization: z-score normalization was applied to the features. This ensures all features lie in a similar scale, improving model performance. It works well with distance-based algorithms and gradient-based methods. The technique reduces the impact of outliers and ensures consistency between training and testing datasets. Overall, it leads to faster convergence and better model accuracy.



PLOTS POST NORMALIZATION

8.Model Selection: Model selection is crucial to ensure that the regression model adequately captures the underlying patterns in the data while avoiding overfitting.

Based on the observation that the plots exhibited U-shapes and curves bending at the ends, it became clear that there was a non-linear relationship between the features and the target variable. However, to start the analysis, I initially implemented multiple linear regression to observe how the model performed under the assumption of a linear relationship.

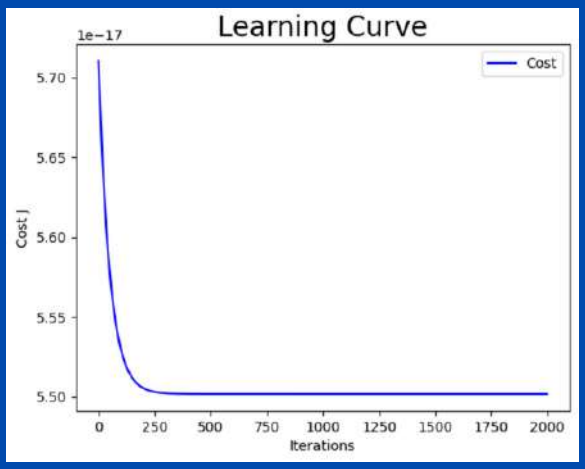
So the initial model is looking like,
f_ab = a_1.x_1 + ... + a_5.x_5 + b

where a_1, a_2, ... , a_5 and b are parameters & x_1, x_2, ... , x_5 are features.

9. Training Results : The Train Cost for the initial model was 5.5018e-17, and the Cross Validation Cost was 5.9006e-17.

Using higher-order features can further refine the model and reduce the costs. By adding features like squares, cubes, or interaction terms, the model can better capture the non-linear relationships between the features and the target variable, leading to improved performance and potentially lowering both training and cross-validation costs.

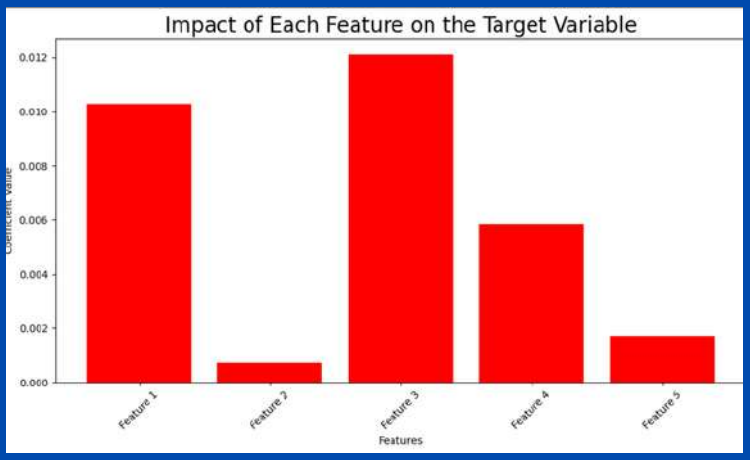
alpha = 0.01




```
Iteration      0: Cost  5.7104e-17
Iteration    200: Cost  5.5054e-17
Iteration    400: Cost  5.5018e-17
Iteration    600: Cost  5.5018e-17
Iteration    800: Cost  5.5018e-17
Iteration   1000: Cost  5.5018e-17
Iteration   1200: Cost  5.5018e-17
Iteration   1400: Cost  5.5018e-17
Iteration   1600: Cost  5.5018e-17
Iteration   1800: Cost  5.5018e-17
TRAIN COST estimated is = 5.5018e-17
Total time for gradient descent: 2.4198 seconds
```

10. Exploring Alternative Models: I will repeat the training process with different and more complex models until a well-performing model is achieved. By iteratively increasing complexity and fine-tuning hyperparameters, I aim to strike a balance between underfitting and overfitting, ensuring the model captures the underlying patterns in the data while maintaining good generalization.

11. Visualizing Feature Impacts: A bar graph was plotted to visualize the impact of each individual feature on the target variable, this will give help us in understanding the effect of individual features in the data.



It can be observed from the bar graph that Feature 3 and Feature 1 have the highest impact (as their coefficients takes the highest values), while Features 2, 4 and 5 have comparatively lower impacts.

12. Conclusion: In the next model, to capture non-linear relationships, higher-order features should be introduced for those features that have a strong impact on the target variable. By focusing on high-impact features, we can better capture complex patterns that the linear model might have missed. Adding higher-order terms for low-impact features, on the other hand, may introduce unnecessary complexity and noise, leading to overfitting without improving predictive performance. Therefore, selecting features with significant correlation to the target variable ensures that the new features meaningfully contribute to the model's accuracy.

13. Second Model Training and Evaluation:

I decided to introduce square terms for all Features by replacing the linear Features

a). MODEL :

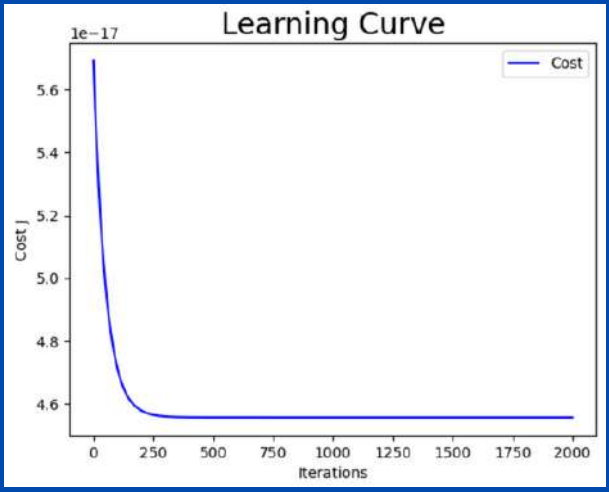
f_ab = a_1.(x_1)^2 + ... + a_5.(x_5)^2 + b

where a_1, a_2, ... , a_5 and b are parameters & x_1, x_2, ... , x_5 are features given in the dataset.

b). COST COMPUTATION (alpha = 0.01) :

Estimated Training Cost for this model is 4.5571e-17 .

Estimated Cross Validation Cost for this model is 4.9636e-17 .




```
Iteration      0: Cost  5.6919e-17
Iteration    200: Cost  4.5784e-17
Iteration    400: Cost  4.5575e-17
Iteration    600: Cost  4.5571e-17
Iteration    800: Cost  4.5571e-17
Iteration   1000: Cost  4.5571e-17
Iteration   1200: Cost  4.5571e-17
Iteration   1400: Cost  4.5571e-17
Iteration   1600: Cost  4.5571e-17
Iteration   1800: Cost  4.5571e-17
TRAIN COST estimated is = 4.5571e-17
Total time for gradient descent: 1.7285 seconds
```

14. Third Model Training and Evaluation:

Since the previous model performed well with low training and cross-validation costs, and both were comparable, I decided to again test the model by taking the cubes of all features.

This approach will help capture potential non-linear relationships between the features and the target variable. Improving the fit of the model, as the previous model showed strong generalization. I will monitor whether this transformation leads to improved accuracy or any signs of overfitting.

a). MODEL :

f_ab = a_1.(x_1)^3 + ... + a_5*(x_5)^3+ b

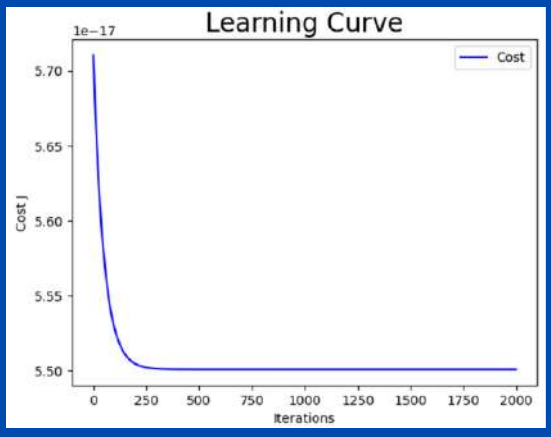
where a_1, a_2, ... , a_4, a_5 and b are parameters & x_1, x_2, ... , x_4, x_5 are features.

b). COST COMPUTATION (alpha = 0.01) :

Estimated Training Cost for this model is 5.5008e-17 .

Estimated Cross Validation Cost for this model is 6.5939e-17 .

Here, the CV Cost went up.




```
Iteration      0: Cost  5.7104e-17
Iteration    200: Cost  5.5045e-17
Iteration    400: Cost  5.5009e-17
Iteration    600: Cost  5.5008e-17
Iteration    800: Cost  5.5008e-17
Iteration   1000: Cost  5.5008e-17
Iteration   1200: Cost  5.5008e-17
Iteration   1400: Cost  5.5008e-17
Iteration   1600: Cost  5.5008e-17
Iteration   1800: Cost  5.5008e-17
TRAIN COST estimated is = 5.5008e-17
Total time for gradient descent: 2.0273 seconds
```

15. Fourth Model Training and Evaluation:

In this model I again raised the powers of the features to increase the complexity of model so that it can better capture the complex patterns in the data.

a). MODEL :

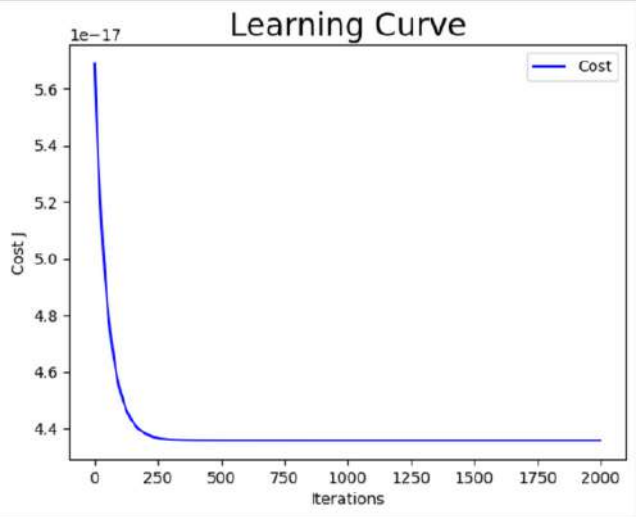
f_ab = a_1.(x_1)^4 + ... + a_5*(x_5)^4 + b

where a_1, a_2, ... , a_4, a_5 and b are parameters & x_1, x_2, ... , x_4, x_5 are features.

b). COST COMPUTATION (alpha = 0.03) :

Estimated Training Cost for this model is 4.3567e-17 .

Estimated Cross Validation Cost for this model is 1.9400e-16 .




```
Iteration      0: Cost  5.6879e-17
Iteration    200: Cost  4.3814e-17
Iteration    400: Cost  4.3572e-17
Iteration    600: Cost  4.3567e-17
Iteration    800: Cost  4.3567e-17
Iteration   1000: Cost  4.3567e-17
Iteration   1200: Cost  4.3567e-17
Iteration   1400: Cost  4.3567e-17
Iteration   1600: Cost  4.3567e-17
Iteration   1800: Cost  4.3567e-17
TRAIN COST estimated is = 4.3567e-17
Total time for gradient descent: 1.2336 seconds
```

16. Fifth Model Training and Evaluation:

In this model I again raised the powers of the features to increase the complexity of model so that it can better capture the complex patterns in the data.

a). MODEL :

f_ab = a_1.(x_1)^5 + ... + a_5*(x_5)^5 + b

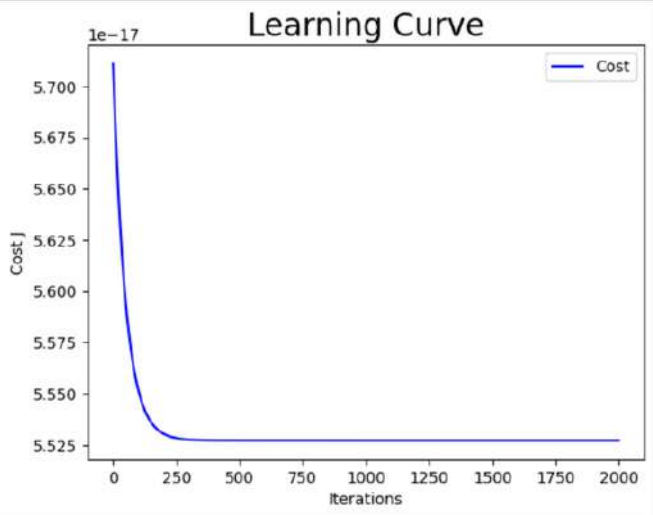
where a_1, a_2, ... , a_4, a_5 and b are parameters & x_1, x_2, ... , x_4, x_5 are features.

b). COST COMPUTATION (alpha = 0.01) :

Estimated Training Cost for this model is 5.5271e-17 .

Estimated Cross Validation Cost for this model is 1.9135e-16 .

Increasing powers is increasing the Train and CV cost gap, this trend is showing overfitting of the model.




```
Iteration      0: Cost  5.7110e-17
Iteration    200: Cost  5.5304e-17
Iteration    400: Cost  5.5272e-17
Iteration    600: Cost  5.5271e-17
Iteration    800: Cost  5.5271e-17
Iteration   1000: Cost  5.5271e-17
Iteration   1200: Cost  5.5271e-17
Iteration   1400: Cost  5.5271e-17
Iteration   1600: Cost  5.5271e-17
Iteration   1800: Cost  5.5271e-17
TRAIN COST estimated is = 5.5271e-17
Total time for gradient descent: 1.8864 seconds
```

17. Sixth Model Training and Evaluation:

In this model I added squares of the features to the initial to increase the complexity of model so that it can better capture the complex patterns in the data.

a). MODEL :

f_ab = a_1.x_1 + ... + a_5.x_5 + a_6.(x_1)^2 + ... + a_10*(x_5)^2 + b

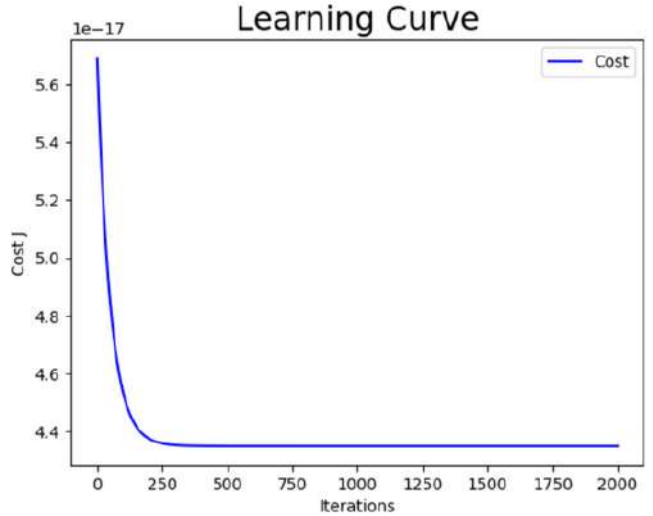
or equivalently, f_ab = a_1.x_1 + ... + a_10.x_10 + b

where a_1, a_2, ... , a_9, a_10 and b are parameters & x_1, x_2, ... , x_9, x_10 are including both features and their squares.

b). COST COMPUTATION (alpha = 0.03) :

Estimated Training Cost for this model is 4.3499e-17 .

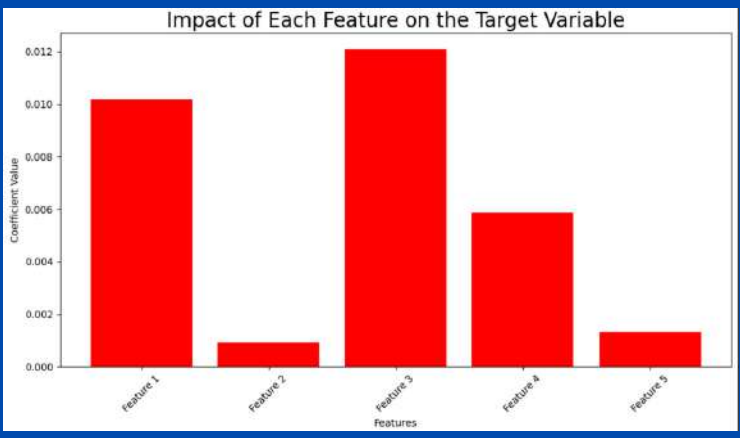
Estimated Cross Validation Cost for this model is 4.6969e-17 .




```
Iteration      0: Cost  5.6877e-17
Iteration    200: Cost  4.3747e-17
Iteration    400: Cost  4.3503e-17
Iteration    600: Cost  4.3499e-17
Iteration    800: Cost  4.3499e-17
Iteration   1000: Cost  4.3499e-17
Iteration   1200: Cost  4.3499e-17
Iteration   1400: Cost  4.3499e-17
Iteration   1600: Cost  4.3499e-17
Iteration   1800: Cost  4.3499e-17
TRAIN COST estimated is = 4.3499e-17
Total time for gradient descent: 1.7027 seconds
```

18. Seventh Model Training and Evaluation:

Since further increasing the polynomial order will not improve the error. Instead, I will introduce interaction features to capture more complex patterns in the dataset. These features represent the combined effect of two or more variables, enabling the model to learn relationships that higher-order polynomials alone may miss.
Initially I would introduce features by just multiplying the features having higher impact.

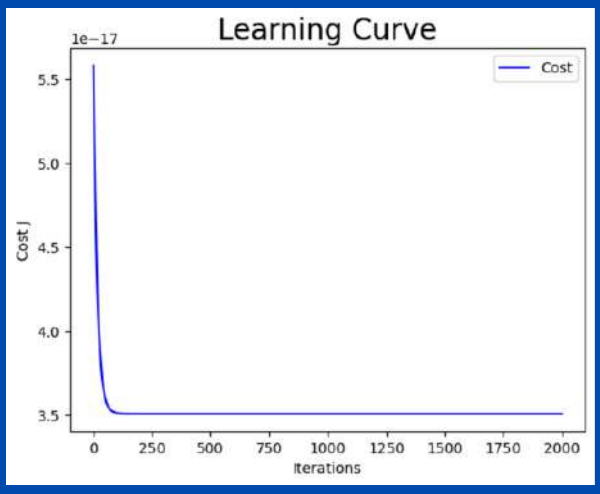


'This bar graph shows feature impacts after sixth model.'

a). MODEL :
f_ab = a_1.x_1 + ... + a_9*(x_4)^2 + a_10(x_5)^2 + a_11.(x_1*x_3) + a_12.(x_3*x_4) +a_13(x_1*x_4) + b
or equivalently, f_ab = a_1.x_1 + ... + a_13.x_13 + b
where a_1, a_2, ... , a_12, a_13 and b are parameters & x_1, x_2, ... , x_12, x_13 are features including x_1, ... , x_5, their squares and the interacting features.

b). COST COMPUTATION (alpha = 0.03) :
Estimated Training Cost for this model is 3.5086e-17.
Estimated Cross Validation Cost for this model is 3.7155e-17 .

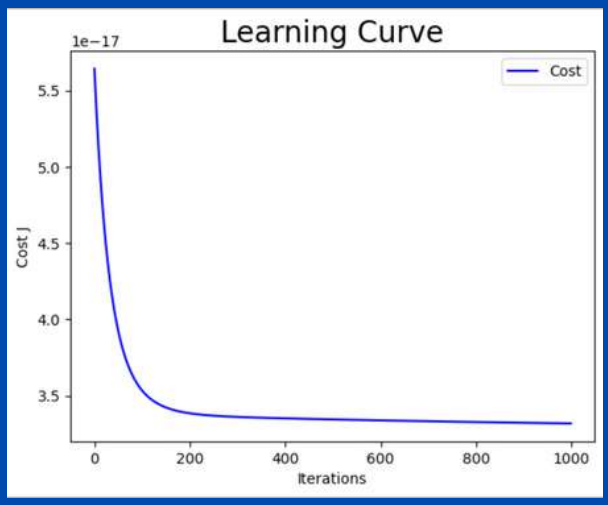
This model did show improved costs.




```
Iteration      0: Cost  5.5837e-17
Iteration    200: Cost  3.5086e-17
Iteration    400: Cost  3.5086e-17
Iteration    600: Cost  3.5086e-17
Iteration    800: Cost  3.5086e-17
Iteration   1000: Cost  3.5086e-17
Iteration   1200: Cost  3.5086e-17
Iteration   1400: Cost  3.5086e-17
Iteration   1600: Cost  3.5086e-17
Iteration   1800: Cost  3.5086e-17
TRAIN COST estimated is = 3.5086e-17
Total time for gradient descent: 3.5618 seconds
```

19. Eighth Model Training and Evaluation:

I further added fourth power of features to the above model, to increase complexity as fourth model performed better than third.




```
Iteration      0: Cost  5.0620e-17
Iteration    100: Cost  3.3170e-17
Iteration    200: Cost  3.2884e-17
Iteration    300: Cost  3.2760e-17
Iteration    400: Cost  3.2707e-17
Iteration    500: Cost  3.2684e-17
Iteration    600: Cost  3.2673e-17
Iteration    700: Cost  3.2669e-17
Iteration    800: Cost  3.2667e-17
Iteration    900: Cost  3.2666e-17
TRAIN COST estimated is = 3.2666e-17
Total time for gradient descent: 1.0667 seconds
```

COST COMPUTATION:

Estimated Training Cost for this model is 3.2666e-17.

Estimated Cross Validation Cost for this model is 3.5229e-17 .

20. Ninth Model Training and Evaluation:

Following the pattern I also added the sixth powers of the features to the model.

a). MODEL :

f_ab = a_1.x_1 + ... + a_23*(x_5)^6 + b

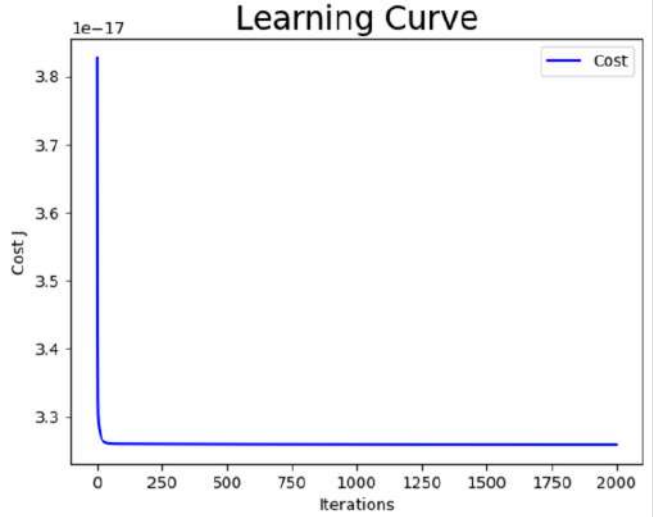
or equivalently, f_ab = a_1.x_1 + ... + a_23.x_23 + b

where a_1, a_2, ... , a_22, a_23 and b are parameters & x_1, x_2, ... , x_22, x_23 are including both features, their squares, the interacting features, fourth and sixth powers of the features.

b). COST COMPUTATION (alpha = 0.1) :

Estimated Training Cost for this model is 3.2590e-17 .

Estimated Cross Validation Cost for this model is 3.5212e-17 .




```
Iteration      0: Cost  3.8267e-17
Iteration    200: Cost  3.2599e-17
Iteration    400: Cost  3.2597e-17
Iteration    600: Cost  3.2595e-17
Iteration    800: Cost  3.2593e-17
Iteration   1000: Cost  3.2592e-17
Iteration   1200: Cost  3.2592e-17
Iteration   1400: Cost  3.2591e-17
Iteration   1600: Cost  3.2591e-17
Iteration   1800: Cost  3.2590e-17
TRAIN COST estimated is = 3.2590e-17
Total time for gradient descent: 2.3795 seconds
```

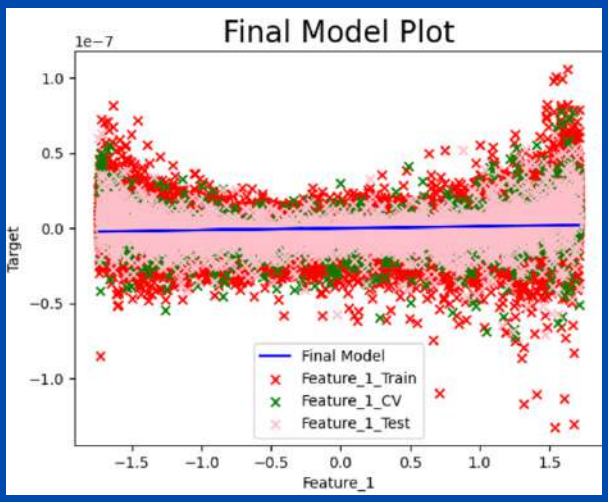
Thus, I finalised the model with the lowest cross-validation and train cost for better generalization.
That is the ninth model,
f_ab = a_1.x_1 + … + a_9*(x_4)^2 + a_10(x_5)^2 + a_11.(x_1*x_3) + a_12.(x_3*x_4) +a_13(x_1*x_4) + … + a_23.(x_5)^6 + b
or equivalently, f_ab = a_1.x_1 + … + a_23.x_23 + b
where a_1, a_2, … , a_22, a_23 and b are parameters & x_1, x_2, … , x_22, x_23 are including both features, their squares, the interacting features, fourth and sixth powers of the features.

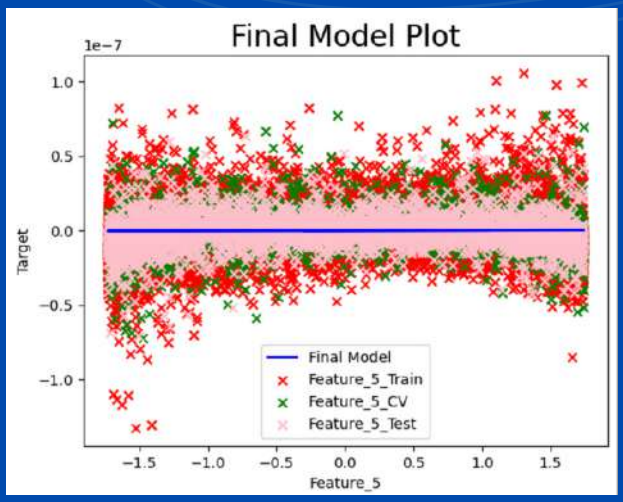
21. Computing Test Cost: The Test Cost for the selected model is 2.9931e-17 . Which is comparable to train and cross validation cost so model generalizes to unseen data.

22. Computing R - squared:
Estimated R2 score for Test Set is 42.30% .

23. Conclusion: The selected model, showed low test cost, demonstrating strong generalization and minimal overfitting. By focusing on key features and avoiding unnecessary complexity, it balances predictive accuracy with simplicity. This approach ensures reliable performance on unseen data, making it the most suitable choice for the current problem.

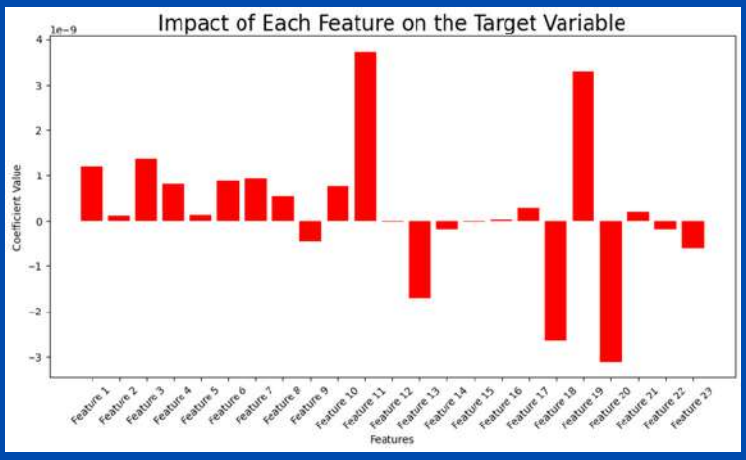
Thus, training process is complete.





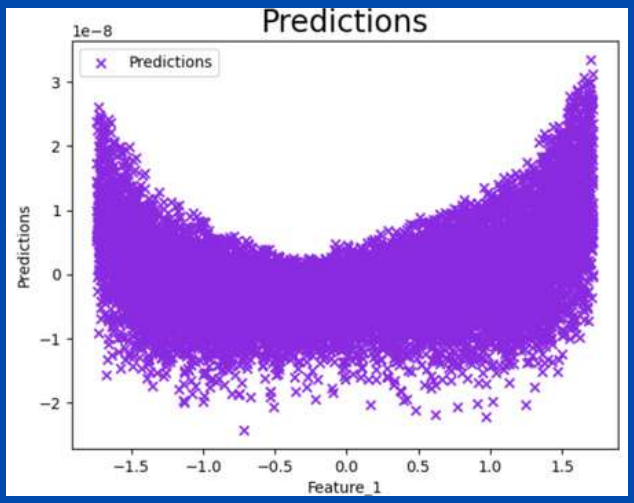
Note: (b_final/n) has been taken for individual features, which is just an approximation used for visualization purposes. These plots show the contributions of individual features to the target variable, but under simplified assumptions. While the model was trained with all features together, these plots isolate each feature and do not capture the full joint behavior of the polynomial regression model, which accounts for the interactions among all features. Thus, these plots are intended solely for interpretability and have limited physical significance

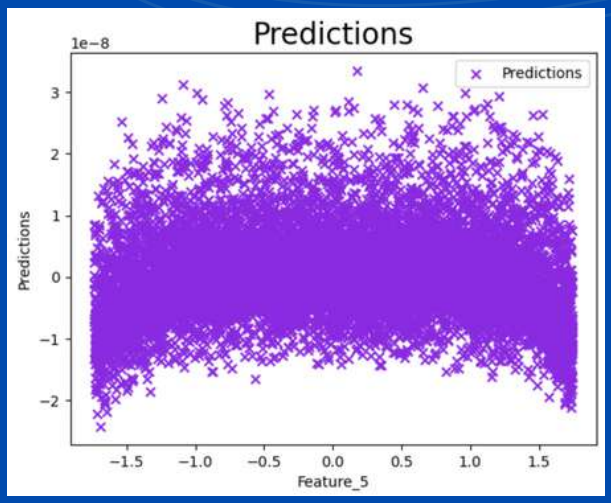
24. Visualizing Feature Impacts: A bar graph was plotted to visualize the impact of each individual feature on the target variable.

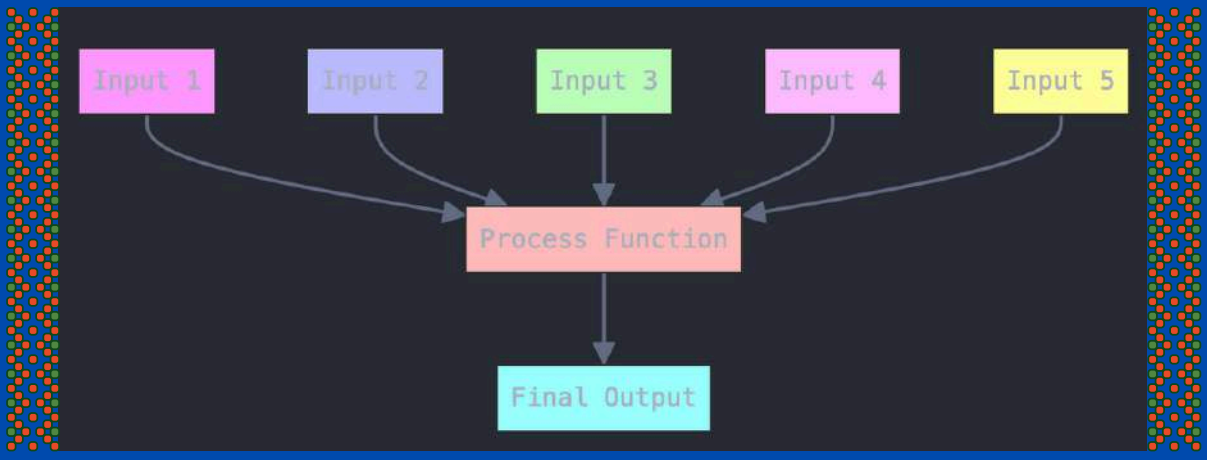


It can be observed from the bar graph that Feature 11 has the highest impact (as its coefficient take the highest value). And Feature 12, 15 and 16 have the low impact on the target variable.

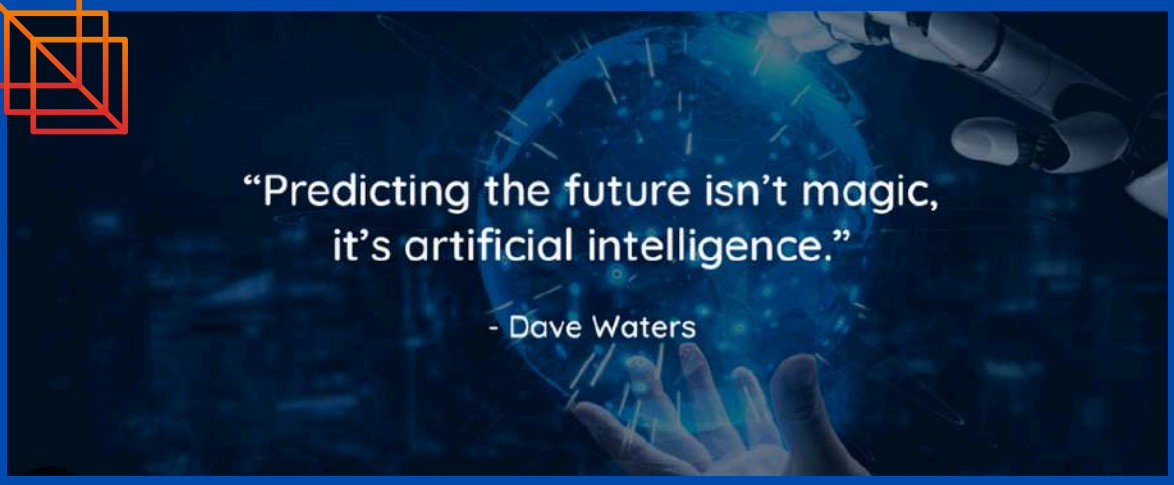
25. Predictions for Test Set: Predictions for the official polynomial_regression_test.csv were computed using the trained model and saved as a CSV file in Google Drive.







>>>Flow Chart showing working of Final Model



CYBER LABS

DECEMBER 2024

PROJECT REPORT

BINARY CLASSIFICATION (K Nearest Neighbor)

PRESENTED TO

Machine Learning Division

PRESENTED BY

Arnav Tripathi
24JE0090

COMMENCEMENT OF THE MODEL

Problem Statement:

Develop a binary classification model using a given dataset that contains 20 features to predict a binary target variable. The target variable has two possible outcomes(0 or 1). The goal is to design an effective model that can classify the instances correctly based on the provided features.

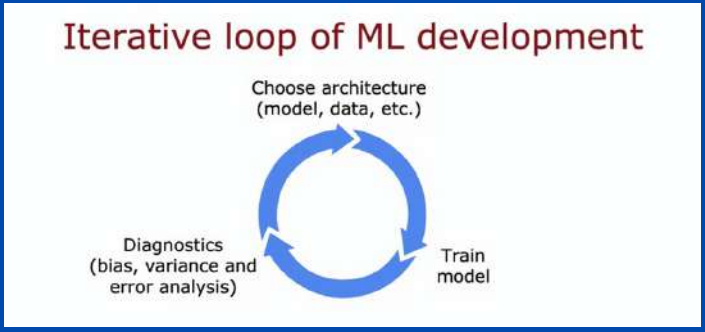
WORKING ON THE MODEL

(Utilizing the official dataset)

1. Data Partitioning: The official dataset was divided into 80% for training, 10% for cross-validation (CV), and 10% for testing. The model is trained on the training set and evaluated using both the CV set and test set.

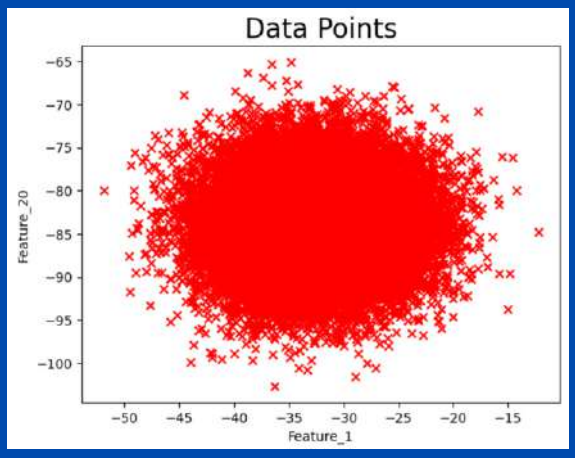
2. Identifying the Type of Target Variable : The target variable is binary, so a model like logistic regression would work well. Logistic regression is a powerful algorithm for binary classification tasks, as it estimates the probability that a given input belongs to a particular class.

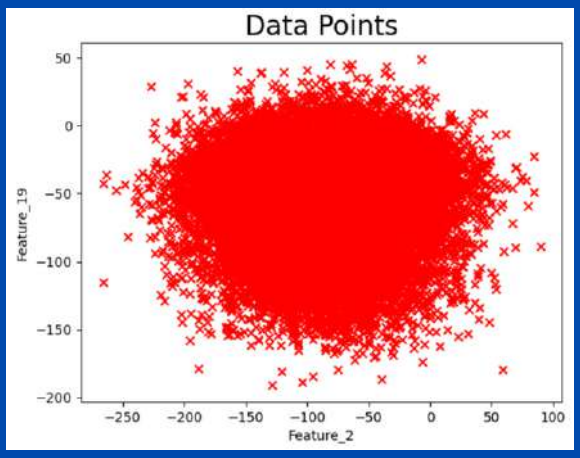
3. Iterative Refinement and Training Loop: The development and training process will follow an iterative loop to ensure systematic progression and refinement of the logistic regression model. Each iteration will address critical aspects of model development, from feature preprocessing to evaluation. The loop is formally described below :

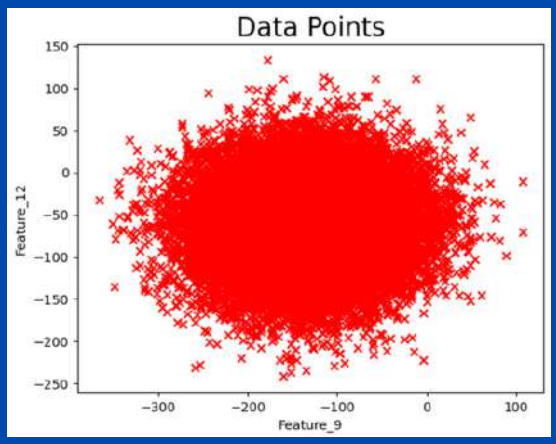


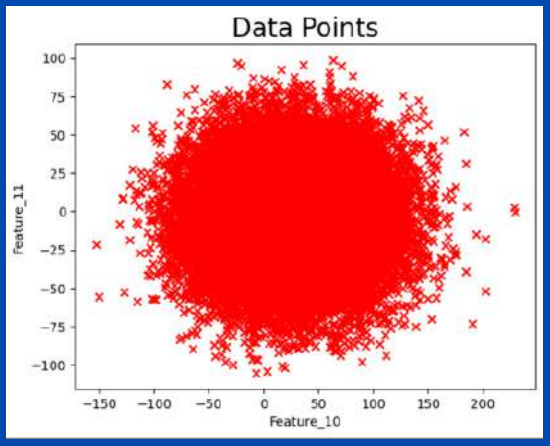
4. Data Organization: The training set was organized into NumPy arrays to facilitate efficient data manipulation and model operations.

5. Data Visualization: The features from the train set were plotted in pairs to inspect potential patterns or relationships visually. This plotting was intended to provide insights into how the features related with the each other, which could aid in model development and understanding of the decision boundary.









6. Plotting Observations : The scatter plot of features reveals a non-linear pattern. The points exhibit a circular or elliptical pattern. Insisting that the decision boundary to be non - linear.

7. Checking for skewed dataset: Checked the number of ones to ensure the data wasn't skewed. But, train set alone had 12277 (31.97%) ones ensuring that data wasn't very skewed.

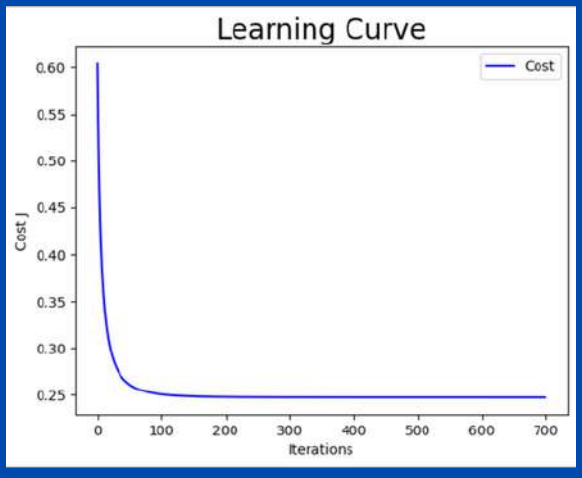
8. Initial Model: For the initial model I still took linear features to check how the model performed on a linear decision boundary.

So the initial model is looking like,
f_ab = a_1.x_1 + … + a_20.x_20 + b

where a_1, a_2, … , a_20 and b are parameters & x_1, x_2, … , x_20 are features.

9. Addressing RuntimeWarning and Overflow: A RuntimeWarning showed up in gradient descent that value passed in np.log() was either zero or negative, and also afterwards overflow was encountered in calculating sigmoid function.
To address both these issues I applied feature-wise z-score normalization to in input. This would bring in numerical stability, faster and stable gradient descent and ensure not only few features dominate the output.

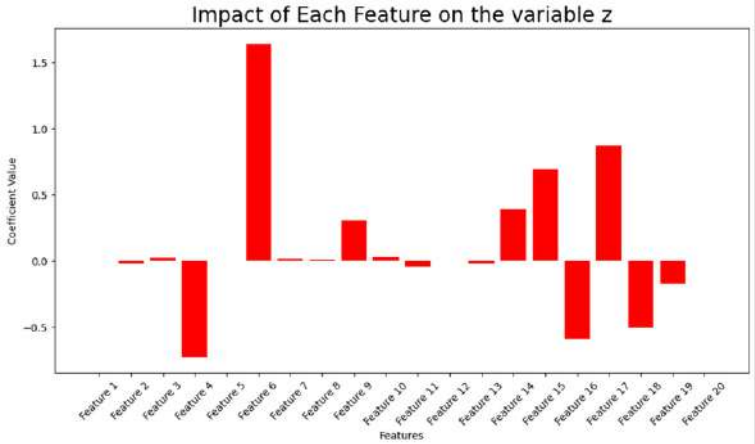
10. Results post Normalization: Estimated Train Cost is 0.2470 .
Estimated Cross Validation Cost is 0.2501 .




```
Iteration      0: Cost   0.6038  dJ_db:  1.803e-01   b:-5.40859e-02
Iteration     70: Cost   0.2543  dJ_db:  1.466e-02   b:-1.01273e+00
Iteration    140: Cost   0.2483  dJ_db:  5.211e-03   b:-1.19765e+00
Iteration    210: Cost   0.2474  dJ_db:  2.299e-03   b:-1.27114e+00
Iteration    280: Cost   0.2471  dJ_db:  1.121e-03   b:-1.30519e+00
Iteration    350: Cost   0.2471  dJ_db:  5.827e-04   b:-1.32231e+00
Iteration    420: Cost   0.2470  dJ_db:  3.173e-04   b:-1.33141e+00
Iteration    490: Cost   0.2470  dJ_db:  1.792e-04   b:-1.33645e+00
Iteration    560: Cost   0.2470  dJ_db:  1.041e-04   b:-1.33934e+00
Iteration    630: Cost   0.2470  dJ_db:  6.177e-05   b:-1.34103e+00
TRAIN COST estimated is =  0.2470
Total time for gradient descent: 3.3971 seconds
```

11. Exploring Alternative Models: I will repeat the training process with different and more complex models until a well-performing model is achieved. By iteratively increasing complexity and fine-tuning hyperparameters, I aim to strike a balance between underfitting and overfitting, ensuring the model captures the underlying patterns in the data while maintaining good generalization.

12. Visualizing Feature Impacts: A bar graph was plotted to visualize the impact of each individual feature on the target variable, this will give help us in introducing higher order features.



It can be observed from the bar graph that Feature 6, 9, 14, 15 and 17 have the higher impact (as their coefficients takes the higher values), while Features 4, 16 and 18 have impact in decreasing the variable inside sigmoid function. Other features have low impacts.

13. Conclusion: In the next model, to capture non-linear relationships, higher-order features should be introduced for those features that have a strong impact on the target variable. By focusing on high-impact features, we can better capture complex patterns that the linear model might have missed. Adding higher-order terms for low-impact features, on the other hand, may introduce unnecessary complexity and noise, leading to overfitting without improving predictive performance. Therefore, selecting features with significant correlation to the target variable ensures that the new features meaningfully contribute to the model's accuracy.

14. Second Model Training and Evaluation:

I decided to introduce square terms for Feature 6, 9, 14, 15 and 17 as they had comparatively higher impacts in increasing the variable z than other features.

a). MODEL :

f_ab = a_1.x_1 + … + a_24.(x_15)^2 + a_25.(x_17)^2 + b

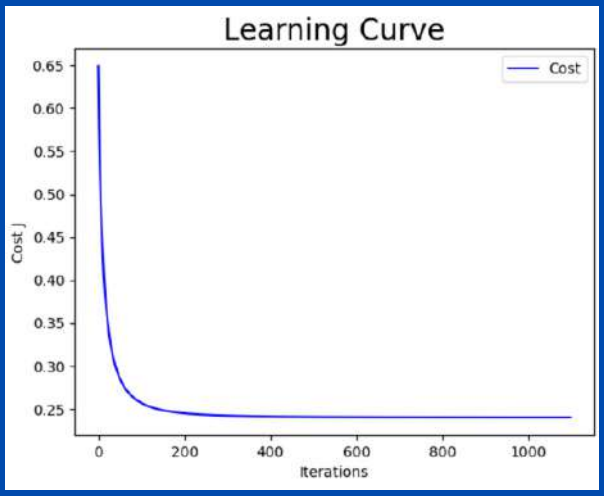
or equivalently, f_ab = a_1.x_1 + … + a_25.x_25 + b

where a_1, a_2, … , a_25 and b are parameters & x_1, x_2, … , x_25 are features including x_1, … , x_20 and squares of selected features.

b). COST COMPUTATION (alpha = 0.03) :

Estimated Training Cost for this model is 0.2403 .

Estimated Cross Validation Cost for this model is 0.0350 .




```
Iteration      0: Cost   0.6492  dJ_db:  1.803e-01   b:-1.80286e-02
Iteration    110: Cost   0.2545  dJ_db:  2.509e-02   b:-7.34039e-01
Iteration    220: Cost   0.2441  dJ_db:  1.028e-02   b:-9.09597e-01
Iteration    330: Cost   0.2418  dJ_db:  5.558e-03   b:-9.92585e-01
Iteration    440: Cost   0.2410  dJ_db:  3.437e-03   b:-1.04062e+00
Iteration    550: Cost   0.2407  dJ_db:  2.301e-03   b:-1.07154e+00
Iteration    660: Cost   0.2405  dJ_db:  1.622e-03   b:-1.09279e+00
Iteration    770: Cost   0.2404  dJ_db:  1.184e-03   b:-1.10804e+00
Iteration    880: Cost   0.2403  dJ_db:  8.857e-04   b:-1.11931e+00
Iteration    990: Cost   0.2403  dJ_db:  6.757e-04   b:-1.12782e+00
TRAIN COST estimated is =  0.2403
Total time for gradient descent: 7.5088 seconds
```

15. Third Model Training and Evaluation:

Next, I added interacting features to model, by using particularly those with higher impact.

I will monitor whether this transformation leads to improved accuracy or any signs of overfitting.

a). MODEL :

f_ab = a_1.x_1 + ... + a_31(x_4.x_16) + b

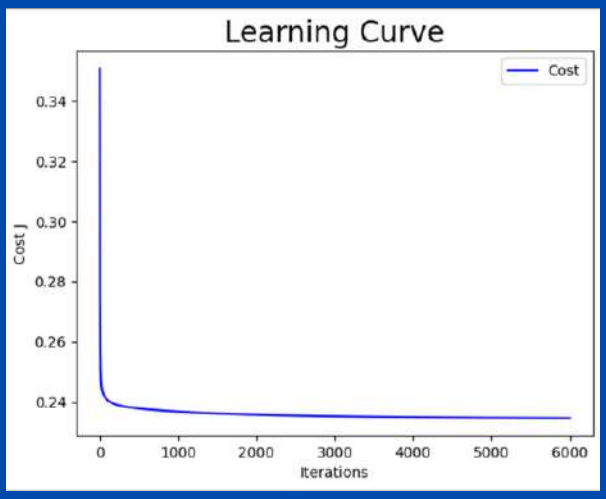
or equivalently, f_ab = a_1.x_1 + ... + a_31.x_31 + b

where a_1, a_2, ... , a_30, a_31 and b are parameters & x_1, x_2, ... , x_30, x_31 are features including x_1, ... , x_5, square and interacting features.

b). COST COMPUTATION (alpha = 1) :

Estimated Training Cost for this model is 0.2347 .

Estimated Cross Validation Cost for this model is 0.0297 .




```
Iteration      0: Cost   0.3507  dJ_db:  1.803e-01   b:-1.80286e-01
Iteration    600: Cost   0.2376  dJ_db:  6.338e-05   b:-1.14121e+00
Iteration   1200: Cost   0.2366  dJ_db:  1.600e-05   b:-1.15988e+00
Iteration   1800: Cost   0.2360  dJ_db:  9.193e-06   b:-1.16701e+00
Iteration   2400: Cost   0.2356  dJ_db:  6.476e-06   b:-1.17163e+00
Iteration   3000: Cost   0.2353  dJ_db:  4.831e-06   b:-1.17498e+00
Iteration   3600: Cost   0.2351  dJ_db:  3.712e-06   b:-1.17752e+00
Iteration   4200: Cost   0.2350  dJ_db:  2.907e-06   b:-1.17950e+00
Iteration   4800: Cost   0.2349  dJ_db:  2.302e-06   b:-1.18105e+00
Iteration   5400: Cost   0.2348  dJ_db:  1.835e-06   b:-1.18229e+00
TRAIN COST estimated is =  0.2347
Total time for gradient descent: 38.8563 seconds
```

16. Fourth Model Training and Evaluation:

I further introduced cube terms for selected features, to cover more complex patterns in the dataset.

a). MODEL :

f_ab = a_1.x_1 + … + a_37.(x_18)^2 + b

or equivalently, f_ab = a_1.x_1 + … + a_37.x_37 + b

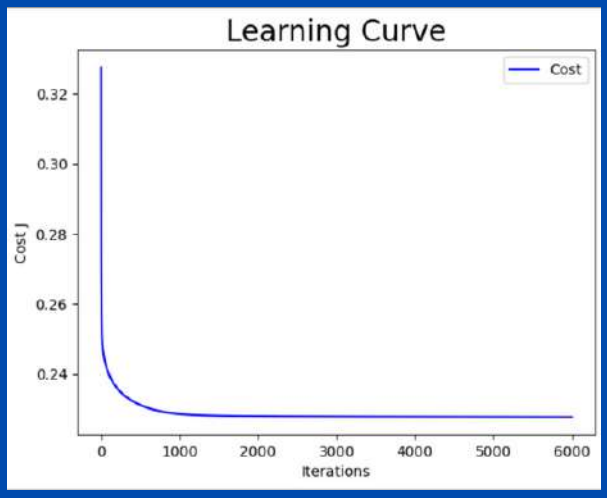
where a_1, a_2, … , a_37 and b are parameters & x_1, x_2, … , x_37 are features including x_1, … , x_20 and square terms, interacting features and cube terms

b). COST COMPUTATION (alpha = 0.08) :

Estimated Training Cost for this model is 0.2277 .

Estimated Cross Validation Cost for this model is 0.0290 .

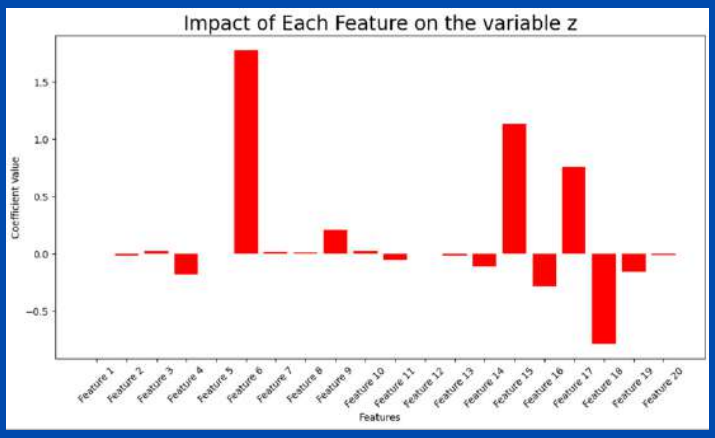
The continuous low values of cross validation cost can be because of small cv set but still train set wasn't compromised as these costs also tells about model performance.




```
Iteration      0: Cost   0.3276  dJ_db:  1.803e-01   b:-1.80286e-01
Iteration    600: Cost   0.2303  dJ_db:  6.506e-05   b:-1.17419e+00
Iteration   1200: Cost   0.2283  dJ_db: -1.379e-07   b:-1.18430e+00
Iteration   1800: Cost   0.2280  dJ_db:  5.092e-10   b:-1.18410e+00
Iteration   2400: Cost   0.2279  dJ_db:  1.929e-07   b:-1.18420e+00
Iteration   3000: Cost   0.2278  dJ_db:  2.614e-08   b:-1.18427e+00
Iteration   3600: Cost   0.2278  dJ_db: -1.460e-07   b:-1.18423e+00
Iteration   4200: Cost   0.2278  dJ_db: -2.620e-07   b:-1.18410e+00
Iteration   4800: Cost   0.2277  dJ_db: -3.304e-07   b:-1.18392e+00
Iteration   5400: Cost   0.2277  dJ_db: -3.670e-07   b:-1.18371e+00
TRAIN COST estimated is =  0.2277
Total time for gradient descent: 40.7932 seconds
```

17. Fifth Model Training and Evaluation:

Again, Analysed the feature impacts to see the contribution of features to the variable z, for further model improvement.



Feature 6, 15 and 17 still had high impact on increasing the variable z so I added their fourth power to the model.

a). MODEL :

f_ab = a_1.x_1 + ... + a_40(x_17)^4 + b

or equivalently, f_ab = a_1.x_1 + ... + a_40.x_40 + b

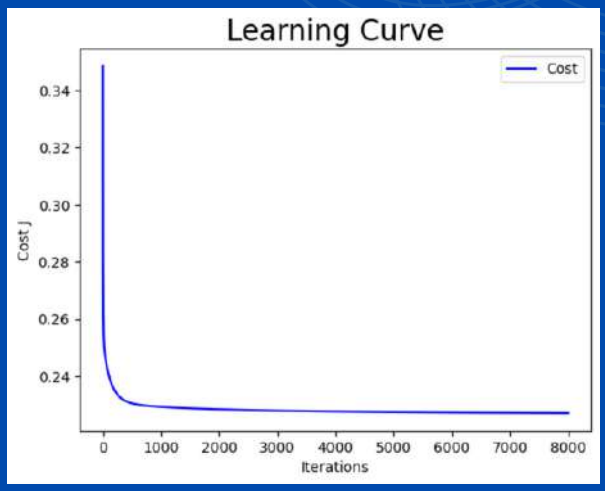
where a_1, a_2, ... , a_39, a_40 and b are parameters & x_1, x_2, ... , x_39, x_40 are features including x_1, ... , x_20, their squares, the interacting features, cube terms and fourth power terms.

b). COST COMPUTATION (alpha = 0.008) :

Estimated Training Cost for this model is 0.2270.

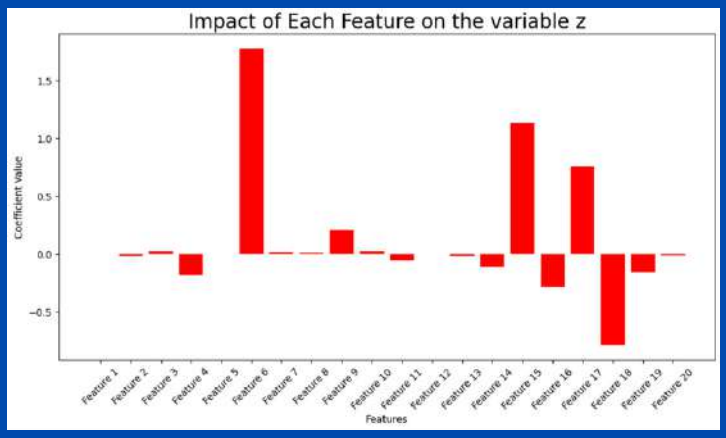
Estimated Cross Validation Cost for this model is 0.0290 .

```
Iteration      0: Cost   0.3484  dJ_db:  1.803e-01   b:-1.44229e-01
Iteration    800: Cost   0.2295  dJ_db: -7.992e-06   b:-1.14789e+00
Iteration   1600: Cost   0.2286  dJ_db:  1.110e-05   b:-1.15079e+00
Iteration   2400: Cost   0.2281  dJ_db:  1.126e-05   b:-1.15824e+00
Iteration   3200: Cost   0.2277  dJ_db:  9.592e-06   b:-1.16490e+00
Iteration   4000: Cost   0.2275  dJ_db:  8.237e-06   b:-1.17059e+00
Iteration   4800: Cost   0.2273  dJ_db:  7.148e-06   b:-1.17550e+00
Iteration   5600: Cost   0.2272  dJ_db:  6.198e-06   b:-1.17977e+00
Iteration   6400: Cost   0.2271  dJ_db:  5.333e-06   b:-1.18345e+00
Iteration   7200: Cost   0.2270  dJ_db:  4.548e-06   b:-1.18661e+00
TRAIN COST estimated is =  0.2270
Total time for gradient descent: 55.5581 seconds
```



18. Sixth Model Training and Evaluation:

Taking the previous approach further, I introduced higher order features derived from the first twenty features in the model, that still have high coefficient values.



Thus, added fifth power of Feature 6, 15, 17 and 18.

a). MODEL :

f_ab = a_1.x_1 + … + a_43.(x_17) + a_44.(x_18) + b

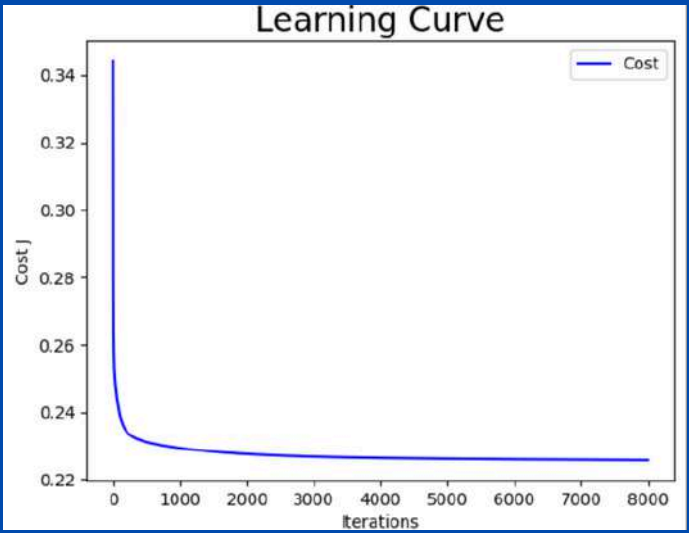
or equivalently, f_ab = a_1.x_1 + … + a_44.x_44 + b

where a_1, a_2, … , a_43, a_44 and b are parameters & x_1, x_2, … , x_43, x_44 are features including x_1, … , x_20, square terms, interacting features, cube terms, fourth power terms and fifth power terms.

b). COST COMPUTATION (alpha = 0.08) :

Estimated Training Cost for this model is 0.2258.

Estimated Cross Validation Cost for this model is 0.0289 .




```
Iteration      0: Cost   0.3442  dJ_db:  1.803e-01   b:-1.44229e-01
Iteration    800: Cost   0.2298  dJ_db:  3.340e-05   b:-1.14190e+00
Iteration   1600: Cost   0.2281  dJ_db:  2.252e-05   b:-1.15929e+00
Iteration   2400: Cost   0.2272  dJ_db:  1.517e-05   b:-1.17131e+00
Iteration   3200: Cost   0.2267  dJ_db:  9.952e-06   b:-1.17919e+00
Iteration   4000: Cost   0.2264  dJ_db:  6.974e-06   b:-1.18452e+00
Iteration   4800: Cost   0.2262  dJ_db:  5.083e-06   b:-1.18834e+00
Iteration   5600: Cost   0.2261  dJ_db:  3.748e-06   b:-1.19114e+00
Iteration   6400: Cost   0.2260  dJ_db:  2.763e-06   b:-1.19321e+00
Iteration   7200: Cost   0.2259  dJ_db:  2.025e-06   b:-1.19473e+00
TRAIN COST estimated is =  0.2258
Total time for gradient descent: 59.2867 seconds
```

After all these observations,

I chose the model with the lowest train and cross-validation cost for better generalization.

That is the sixth model,

f_ab = a_1.x_1 + … + a_43.(x_17) + a_44.(x_18) + b

or equivalently, f_ab = a_1.x_1 + … + a_44.x_44 + b

where a_1, a_2, … , a_43, a_44 and b are parameters & x_1, x_2, … , x_43, x_44 are features including x_1, … , x_20, square terms, interacting features, cube terms, fourth power terms and fifth power terms.

19. Computing Test Cost: The Test Cost for the selected model is 0.0277 . This comparable to CV cost which shows good generalization for unseen data.

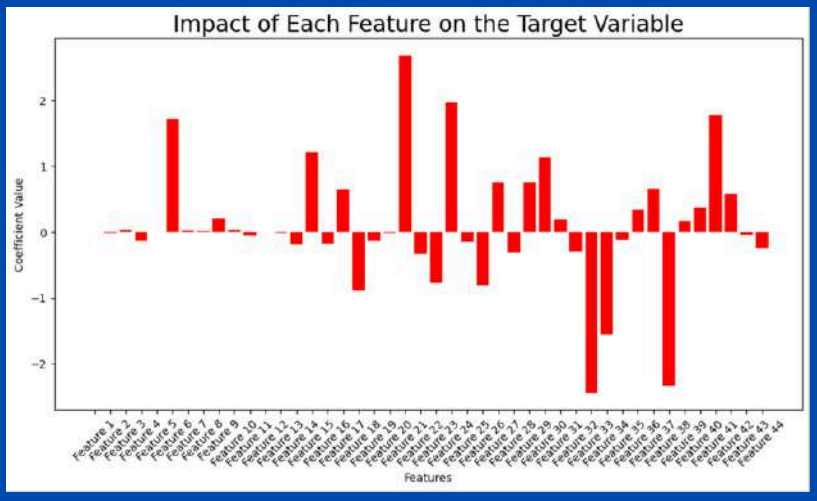
20. Computing Accuracy:
Used CV set to determine what should be the threshold above which Prediction should be 1.
Using threshold 0.5 gave Accuracy = 58.45%
decreasing threshold was decreasing accuracy, so now increased threshold till accuracy starts decreasing again.
Balanced Accuracy obtained was 62.97% at threshold = 0.96 , for CV set.
Accuracy for Train Set = 80.84% .

Accuracy for Test Set = 63.01% .

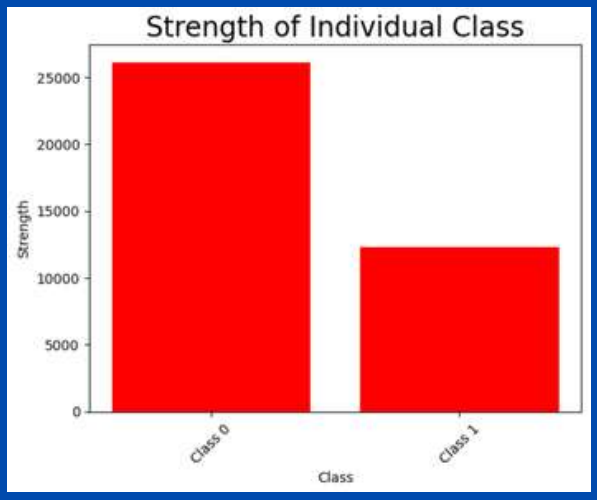
21. Conclusion: The selected model, showed low test cost, demonstrating generalization and minimal overfitting. By focusing on key features and avoiding unnecessary complexity, it balances predictive accuracy with simplicity. This approach ensures reliable performance on unseen data, making it the most suitable choice for the current problem.

Thus, training process is complete.

22. Visualizing Feature Impacts: A bar graph was plotted to visualize the impact of each individual feature on the target variable.

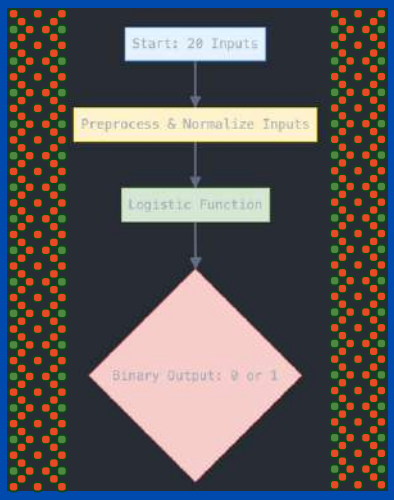


23. Diagnostic: When I computed the F1 Score of the final model using Final Model it came out nan, because precision and recall both were zero. This was because of the unbalanced dataset,



That's why keeping the threshold high was giving higher accuracy.

24. Changing Algorithm: After the results of the previous model,



>>>Flow Chart showing working of Logistic Regression Model

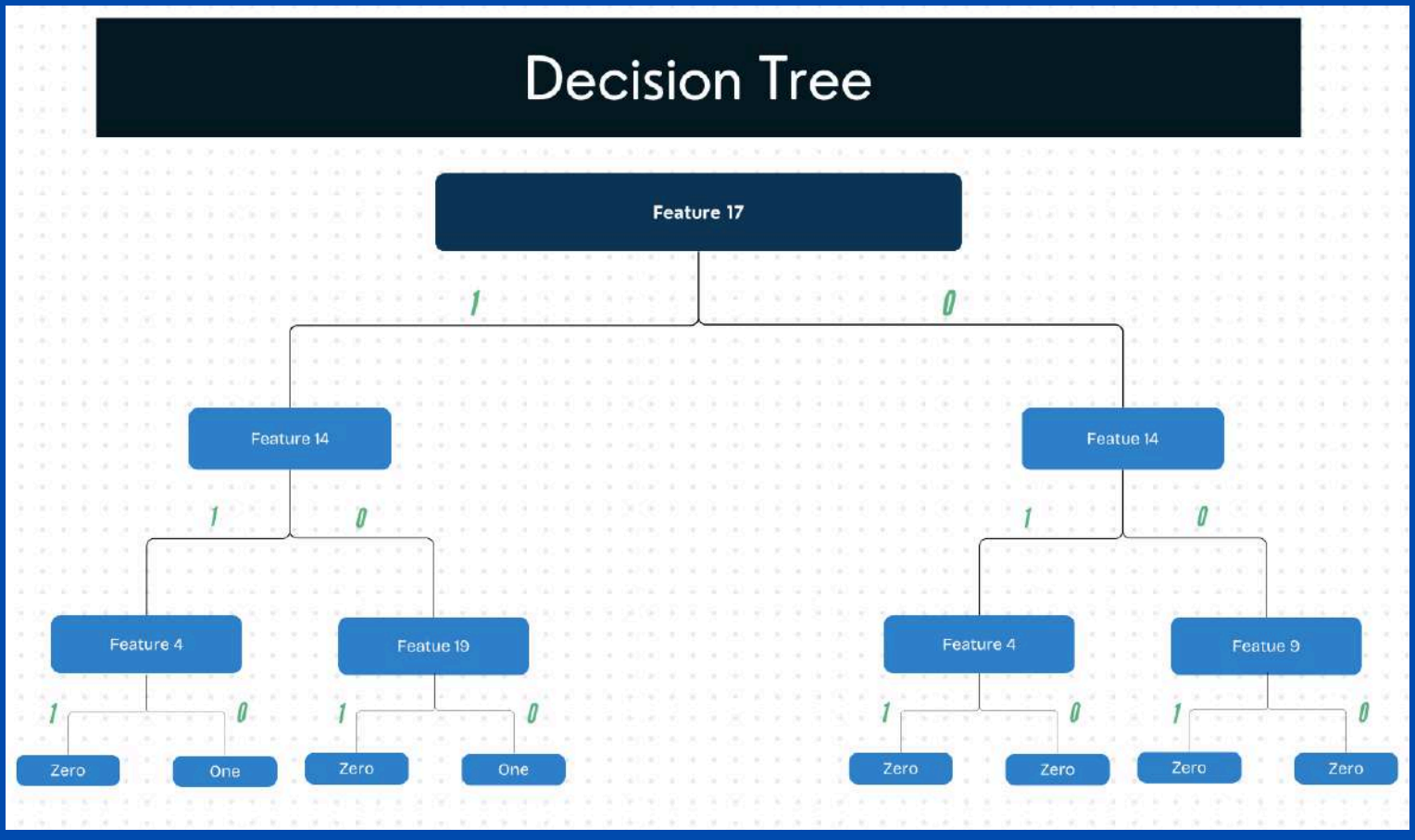
I decided to change the algorithm for better results.
I implemented KNN on the same dataset.

25. Results: The computed F1 Score was 0.89 , so I finalized this approach.

26. Decision Tree: Made a decision tree of depth 2 for this dataset to analyse its performance.

27. Results: Train Set Accuracy is 75.0% .
Cross Validation Set Accuracy is 70.0% .
Test Set Accuracy is 80.0% .

I didn't finalize this algorithm, even though its performance was decent, because a single decision tree proved less reliable due to the numerous assumptions made during the tree-building process. A tree ensemble would likely have been a better choice for this dataset, delivering significantly improved results.



>>> Resultant Decision Tree after Training

28. Predictions: The predictions for the official binary test csv file was computed and saved in Google Drive.

CYBER LABS

DECEMBER 2024

PROJECT REPORT

MULTICLASSIFICATION (K NEAREST NEIGHBOR)

PRESENTED TO

Machine Learning Division

PRESENTED BY

Arnav Tripathi
24JE0090

COMMENCEMENT OF THE MODEL

Problem Statement:

Develop a Multiclass Classification Model for Predicting Class Variable Using 20 Features.

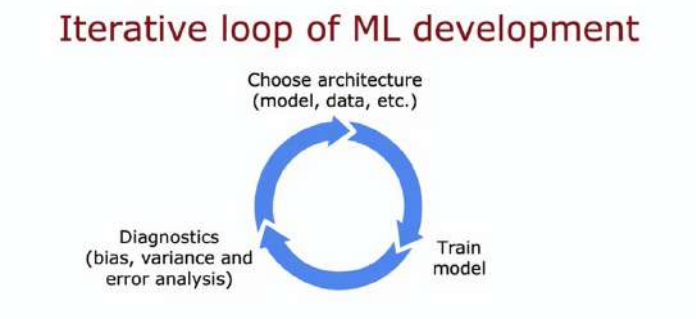
WORKING ON THE MODEL

(Utilizing the official dataset)

Neural Network (NN)

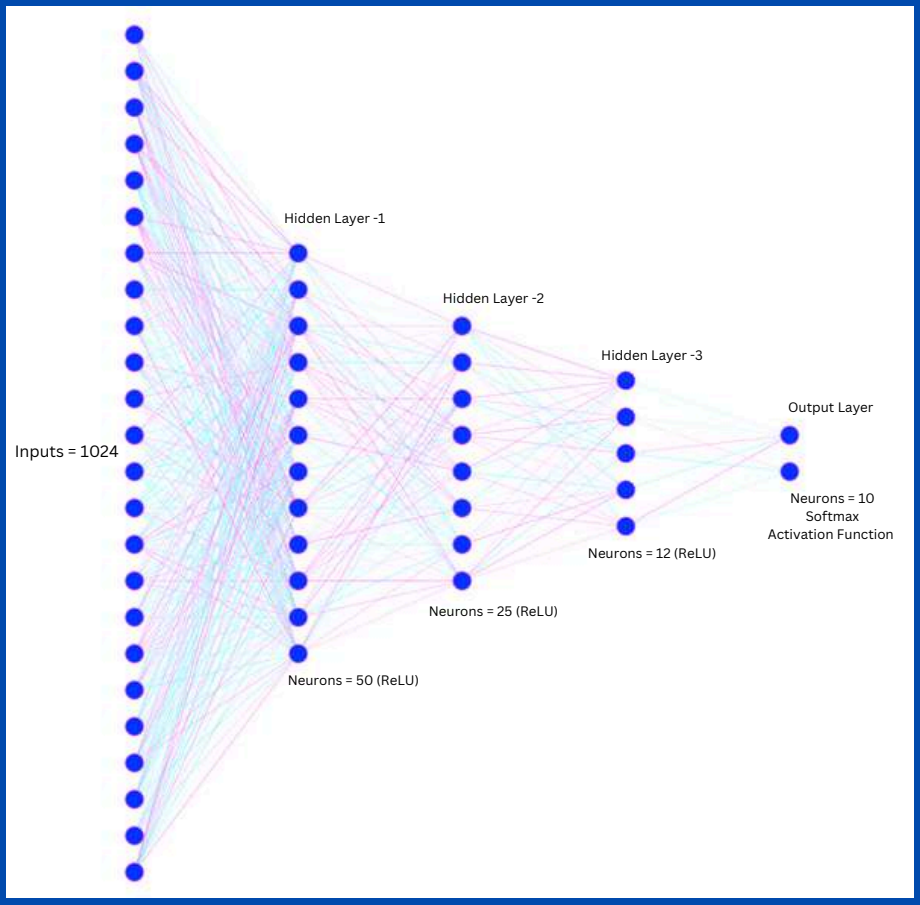
1. Data Partitioning: The dataset was divided into 80% for training, 10% for cross-validation (CV), and 10% for testing. The model is trained on the training set and evaluated using both the CV set and test set.

2. Identifying the Type of Target Variable : The task of the provided dataset is to predict class rather than continuous values.



3. Data Organization: The training set was organized into NumPy arrays to facilitate efficient data manipulation and model operations.

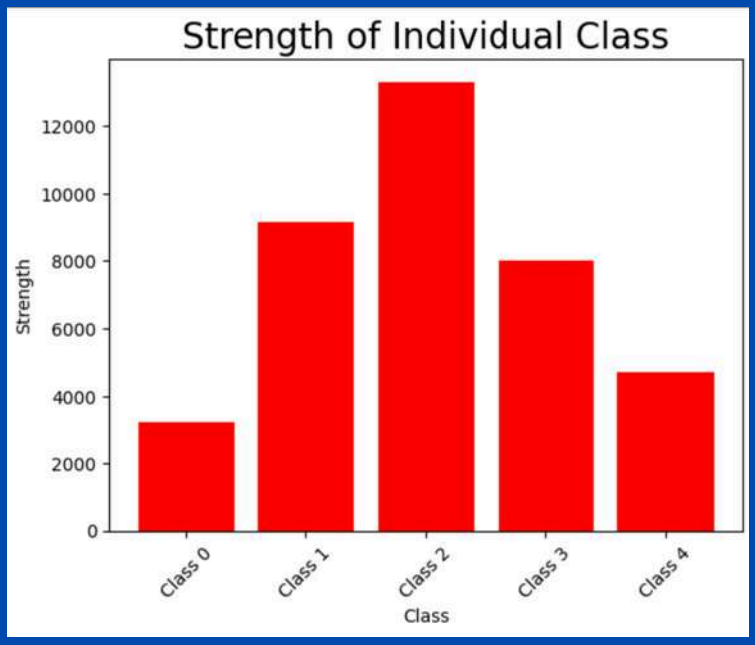
4. Neural Network Architecture: Designed a neural network with four dense layers, comprising 50 neurons in Layer 1, 25 neurons in Layer 2, 12 neurons in Layer 3, and 10 neurons in Layer 4. The hidden layers utilize the ReLU activation function, while the output layer employs a softmax activation function to predict a class label effectively.



>>>Small Representation of Neural Network Architecture

5. Feature Normalization: z-score normalization was applied to the features. This ensures all features lie in a similar scale, improving model performance. It works well with distance-based algorithms and gradient-based methods. The technique reduces the impact of outliers and ensures consistency between training and testing datasets. Overall, it leads to faster convergence and better model accuracy.

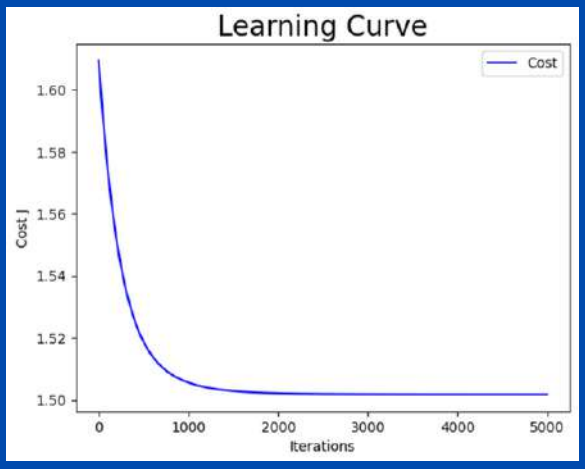
6. Data Visualisation: Plotted a bar graph to visualize the strength of each class.



7. Plotting Observations: The plots showed that Class 2 was in majority and was much more than any other class.

8. Tuning other Hyperparameters: Started with a learning rate of 0.01 and 5000 iterations. The learning curve after training for the first time was smooth but and the Cost converged quickly.

9. Training Results :
Kept the learning rate high because model was taking long time to learn even after taking a mini batch.
This can be adjusted for next batches.




```
CPU times: user 3 µs, sys: 0 ns, total: 3 µs
Wall time: 7.15 µs
Epochs      0/5000 : Cost   1.6094
Epochs    500/5000 : Cost   1.5188
Epochs   1000/5000 : Cost   1.5056
Epochs   1500/5000 : Cost   1.5029
Epochs   2000/5000 : Cost   1.5022
Epochs   2500/5000 : Cost   1.5020
Epochs   3000/5000 : Cost   1.5019
Epochs   3500/5000 : Cost   1.5019
Epochs   4000/5000 : Cost   1.5019
Epochs   4500/5000 : Cost   1.5019
TRAIN COST estimated is =  1.5019
```

The Estimated Training Cost is 1.5019 .
The Cross Validation Cost is 1.4968 .

Both were Comparable.
The Learning Curve showed converging cost, this meant that training process was complete.

10. Further Results: Estimated Test Cost was 1.5046 .

11. Accuracies :
The Estimated Test Accuracy is 34.20% .
The Estimated Cross Validation Accuracy is 34.625% .

The Estimated Train Accuracy is 34.67% .

12. Diagnosis: Even after Training the neural network the accuracy on the train set was very low.
Which definitely didn't seemed right.

13. Conclusion: This issue likely arose due to the imbalanced dataset. The high occurrence of Class 2 caused the neural network to predominantly predict Class 2, as it learned to minimize error by favoring the majority class.

14. Confirmation: I checked this conclusion by making an array full of Class 2 and same size as y_train.
Considering this generated array as the prediction for train set I estimated the accuracy of this prediction and it was 34.67% exactly same as the train set accuracy.
This further confirmed my conclusion and highlighted the need of a different approach for this task.

WORKING ON THE MODEL

(Utilizing the official dataset)

K Nearest Neighbor (KNN)

1. Data Partitioning: The dataset was divided into 80% for training, 10% for cross-validation (CV), and 10% for testing.
The train set will used to find the nearest neighbours.
The cv set will help tune the hyper parameters,
and the test set will be used for unbiased testing of algorithm.

2. Identifying the Type of Target Variable : The task of the provided dataset is to predict binary class rather than continuous values. Therefore, KNN being a good classification algorithm would be an appropriate model for this scenario (as given in the problem statement).

3. Data Organization: The training set was organized into NumPy arrays to facilitate efficient data manipulation and model operations.

4. Feature Normalization: z-score normalization was applied to the features. This ensures all features lie in a similar scale, improving model performance. It works well with distance-based algorithms and gradient-based methods. The technique reduces the impact of outliers and ensures consistency between training and testing datasets. Overall, it leads to faster convergence and better model accuracy.

5. Coding: Coded the algorithm to using just Python and NumPy to complete the task appropriately.

6. Cross Validation Set: Tested the algorithm on Cross Validation Set using the K value to be 5 .

7. Result: The Estimated accuracy for the Cross Validation Set was 95.18% .

Hence, this algorithm was working fine for the unbalanced dataset. Even when KNN is sensitive to such Data.

8. Further Results: The Estimated Test Set Accuracy was 72.5% .
The Estimated F1 Score is 0.94 .

9. Predictions: The Predictions for the official Multiclassification Test Set were estimated and saved as a CSV file.


Me
Deep learning
Simple problem

CYBER LABS

DECEMBER 2024

PROJECT REPORT

UNSUPERVISED DATA (K - means CLUSTERING)

PRESENTED TO

Machine Learning Division

PRESENTED BY

Arnav Tripathi
24JE0090

COMMENCEMENT OF THE MODEL

Problem Statement:

Given an unlabelled dataset, the objective is to implement an unsupervised learning algorithm to identify hidden patterns, clusters, or anomalies within the data. The solution should aim to uncover meaningful insights, reduce dimensionality if necessary, and provide visual representations of the discovered patterns, enabling better data understanding and decision-making.

WORKING ON THE MODEL

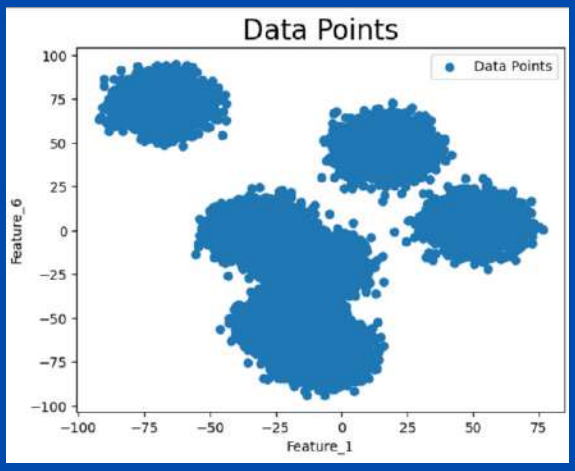
(Utilizing the official dataset)

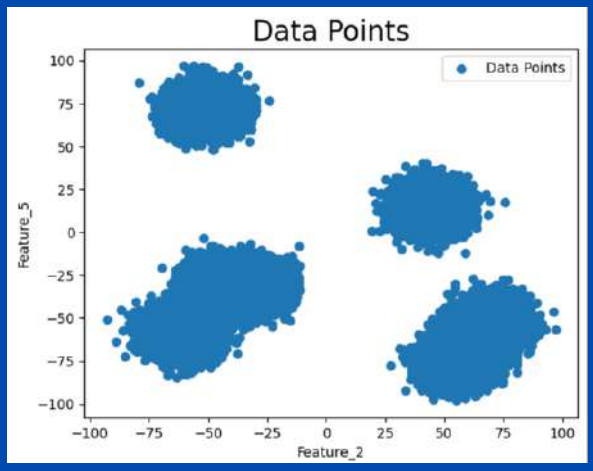
1. Data Organization: The data set was organized into NumPy arrays to facilitate efficient data manipulation and model operations.

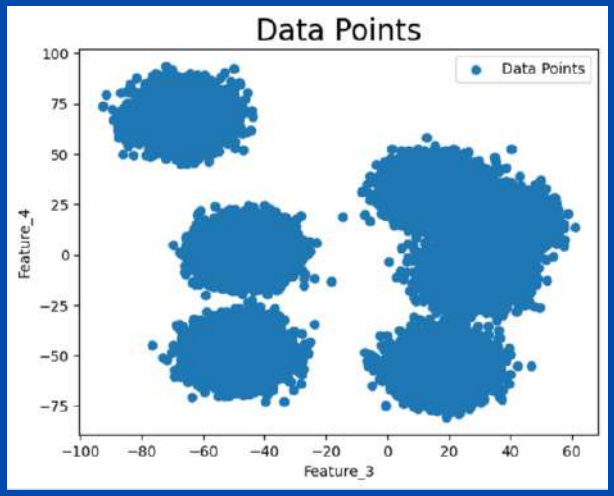
2.Data Integrity Assurance: No normalization or scaling was initially applied to the dataset to preserve its original distribution. Altering the data's distribution could have led to misinterpretation of patterns or incorrect analysis during the Task Discovery Phase.

3. Task Discovery Phase: The first goal is to analyze the unlabelled dataset to identify the underlying task or patterns, determining whether the problem aligns with clustering, anomaly detection, or dimensionality reduction, before selecting an appropriate unsupervised algorithm.

4. Feature vs Feature Visualization: The features from the train set were plotted against each other in pairs to inspect potential patterns or relationships visually. This plotting was intended to provide insights into how the features correlated with the each other, which could aid in model development and understanding of the data.



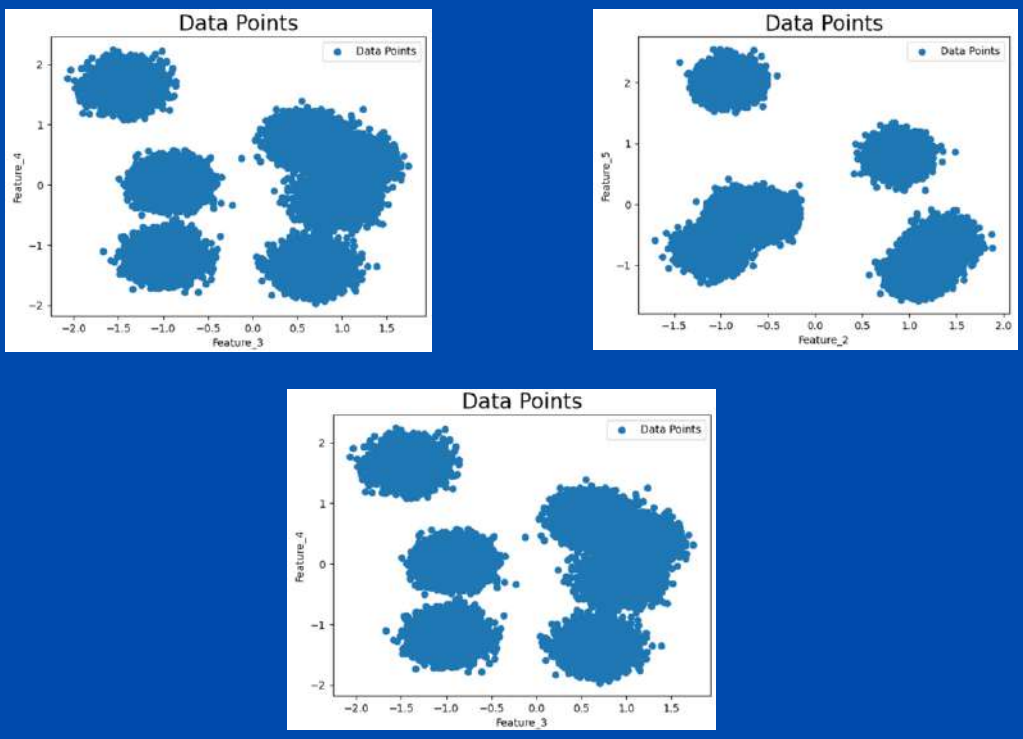




5. Plotting Observations : The scatter plot of features reveals a clear clustering pattern, with data points forming four to six distinct groups. However, the clarity of these clusters is limited by the inability to visualize all features simultaneously.

Additionally, the plots indicate that the features have varying scales, emphasizing the need for normalization to ensure that each feature contributes equally to the clustering analysis.

6. Feature Normalization: z-score normalization was applied to the features. This ensures all features lie in a similar scale, improving model performance. It works well with distance-based algorithms and gradient-based methods. The technique reduces the impact of outliers and ensures consistency between training and testing datasets. Overall, it leads to faster convergence and better model accuracy.



PLOTS POST NORMALIZATION

7. Algorithm Selection: The clustering patterns observed in the scatter plots clearly indicate the need for a clustering algorithm to group the data effectively. Given its efficiency and suitability for well-separated clusters, K-Means clustering was chosen as the appropriate algorithm for this dataset.

8.Algorithm Coding: The K-Means clustering algorithm has been coded with all necessary functions defined, including initialization, assignment, centroid update, and convergence check. The implementation is now ready for execution and testing.

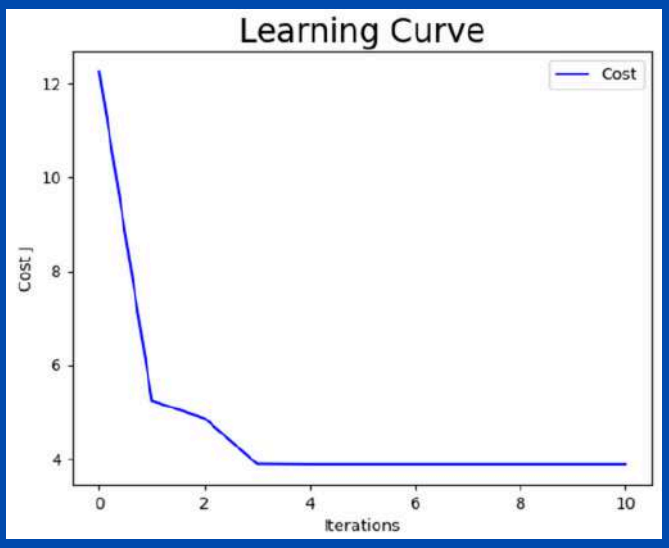
9. Centroid Initialization and Cluster Selection: To initialize the centroids, the decision was made to use random data points themselves as the initial centroids.
The initialization will be repeated multiple times to minimize the cost function for that particular number of clusters.
The final number of clusters will be determined by evaluating the maximum decrease in cost for each cluster count (Elbow Method). The number of clusters yielding the maximum reduction in cost will be chosen as the optimal number of clusters for the dataset.

10. Two Random Centroids Initialization:

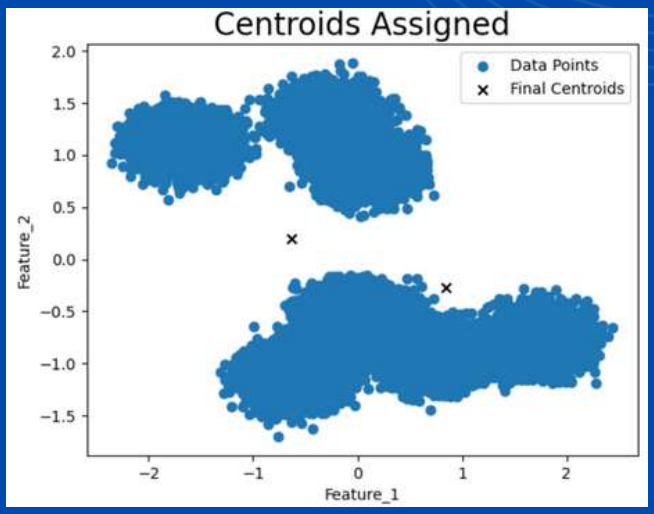
The process begins by randomly selecting two centroids from the dataset and noting the resulting readings (cost or cluster assignments).

11. Results: After multiple initialization for two centroids the converged costs observed were as follows - 4.1121 , 4.4787 , 3.8719 , 4.0115 and 4.5616.

Therefore the minima is 3.8719 .




```
CPU times: user 2 µs, sys: 0 ns, total: 2 µs
Wall time: 4.53 µs
Iteration 01/10 : Cost =   5.2310
Iteration 02/10 : Cost =   4.8575
Iteration 03/10 : Cost =   3.8810
Iteration 04/10 : Cost =   3.8719
Iteration 05/10 : Cost =   3.8719
Iteration 06/10 : Cost =   3.8719
Iteration 07/10 : Cost =   3.8719
Iteration 08/10 : Cost =   3.8719
Iteration 09/10 : Cost =   3.8719
Iteration 10/10 : Cost =   3.8719
Estimated Final Cost is =  3.8719
```

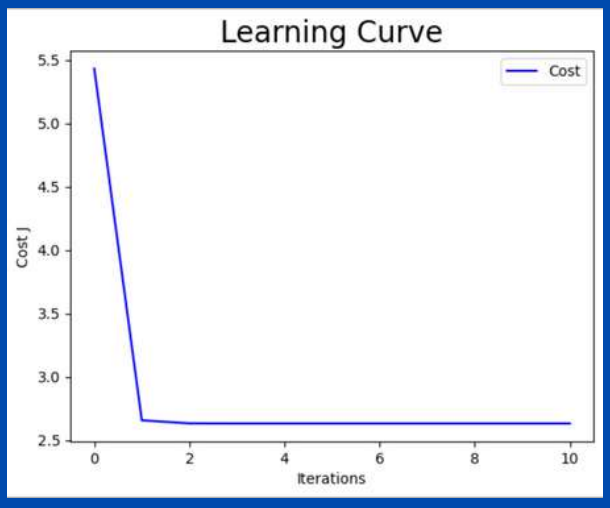


12. Three Random Centroids Initialization:

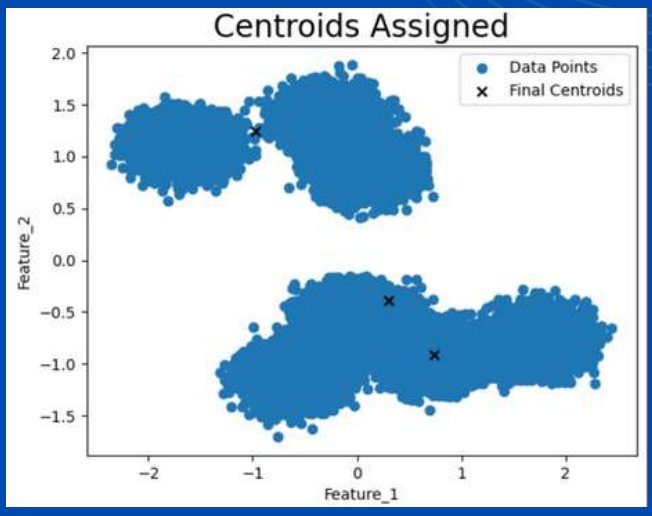
The process begins by randomly selecting three centroids from the dataset and noting the resulting readings (cost or cluster assignments).

13. Results: After multiple initialization for three centroids the converged costs observed were as follows - 2.8466, 3.3027, 2.9751, 3.0050, 2.6319 and 2.7142.

Therefore the minima is 2.6319 .



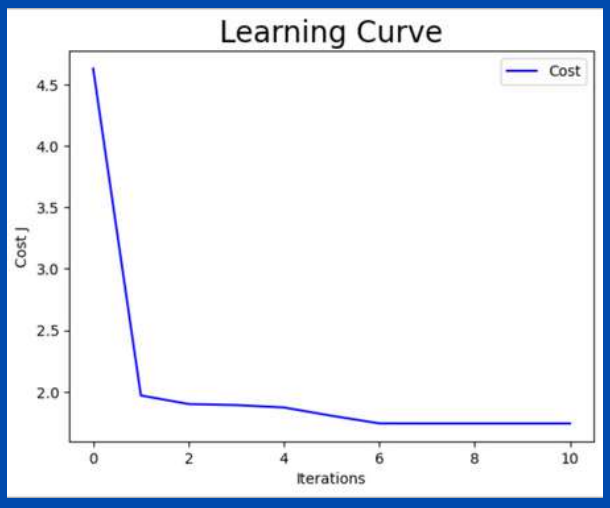

```
CPU times: user 2 µs, sys: 0 ns, total: 2 µs
Wall time: 5.48 µs
Iteration 01/10 : Cost =   2.6567
Iteration 02/10 : Cost =   2.6328
Iteration 03/10 : Cost =   2.6319
Iteration 04/10 : Cost =   2.6319
Iteration 05/10 : Cost =   2.6319
Iteration 06/10 : Cost =   2.6319
Iteration 07/10 : Cost =   2.6319
Iteration 08/10 : Cost =   2.6319
Iteration 09/10 : Cost =   2.6319
Iteration 10/10 : Cost =   2.6319
Estimated Final Cost is =  2.6319
```



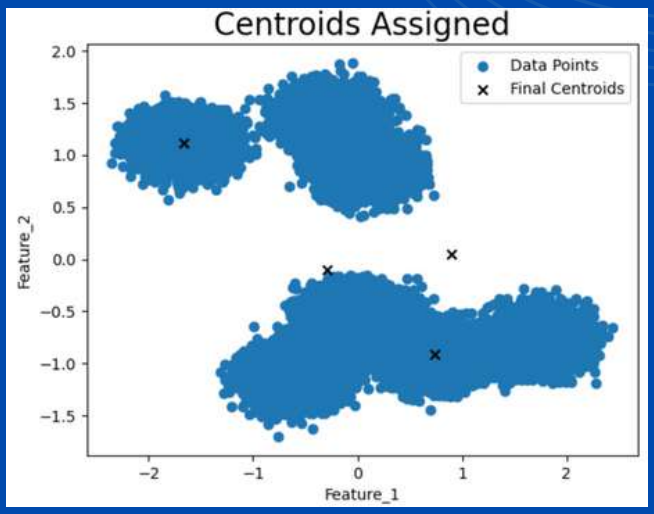
14. Four Random Centroids Initialization:

The process begins by randomly selecting four centroids from the dataset and noting the resulting readings (cost or cluster assignments).

15. Results: After multiple initialization for three centroids the converged costs observed were as follows - 2.0166, 1.8775, 2.9778 and 1.7428.
Therefore the minima is 1.7428 .



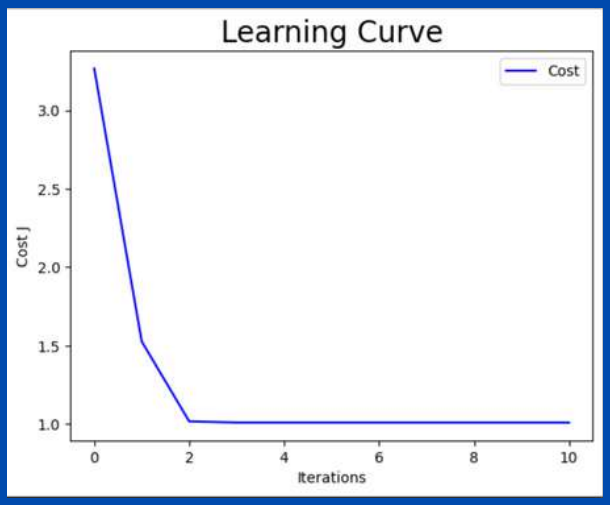

```
CPU times: user 2 µs, sys: 0 ns, total: 2 µs
Wall time: 5.72 µs
Iteration 01/10 : Cost =   1.9704
Iteration 02/10 : Cost =   1.9009
Iteration 03/10 : Cost =   1.8921
Iteration 04/10 : Cost =   1.8728
Iteration 05/10 : Cost =   1.8053
Iteration 06/10 : Cost =   1.7436
Iteration 07/10 : Cost =   1.7428
Iteration 08/10 : Cost =   1.7428
Iteration 09/10 : Cost =   1.7428
Iteration 10/10 : Cost =   1.7428
Estimated Final Cost is =  1.7428
```



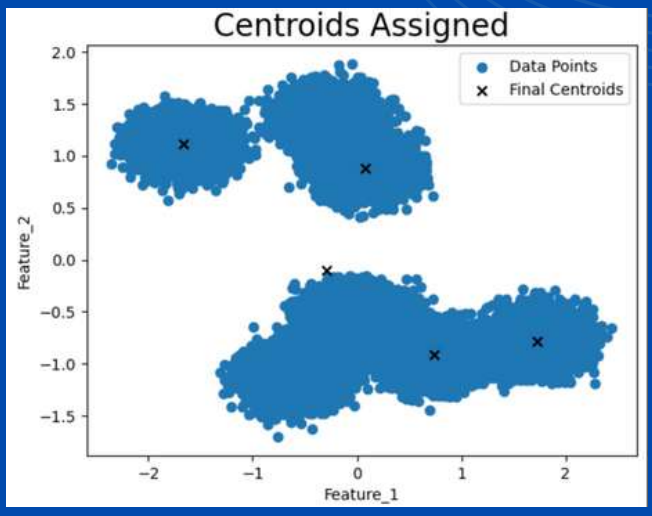
16. Five Random Centroids Initialization:

The process begins by randomly selecting five centroids from the dataset and noting the resulting readings (cost or cluster assignments).

17. Results: After multiple initialization for five centroids the converged costs observed were as follows - 1.0097, 1.0679, 2.3305 and 2.0258.
Therefore the minima is 1.0097 .



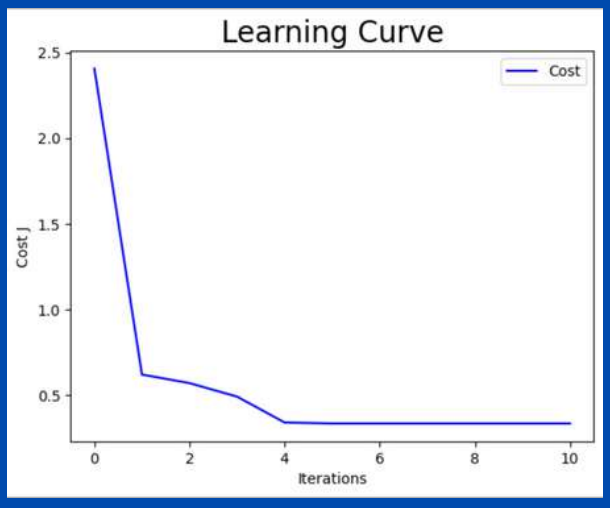

```
CPU times: user 2 µs, sys: 0 ns, total: 2 µs
Wall time: 6.2 µs
Iteration 01/10 : Cost =   1.5267
Iteration 02/10 : Cost =   1.0176
Iteration 03/10 : Cost =   1.0097
Iteration 04/10 : Cost =   1.0097
Iteration 05/10 : Cost =   1.0097
Iteration 06/10 : Cost =   1.0097
Iteration 07/10 : Cost =   1.0097
Iteration 08/10 : Cost =   1.0097
Iteration 09/10 : Cost =   1.0097
Iteration 10/10 : Cost =   1.0097
Estimated Final Cost is =  1.0097
```



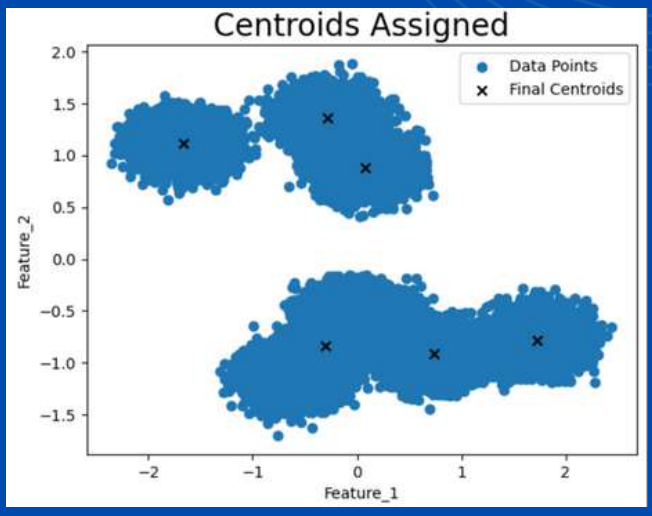
18. Six Random Centroids Initialization:

The process begins by randomly selecting Six centroids from the dataset and noting the resulting readings (cost or cluster assignments).

19. Results: After multiple initialization for six centroids the converged costs observed were as follows - 1.0116, 1.0115, 0.6412, 0.3347 and 0.8127.
Therefore the minima is 0.3347 .



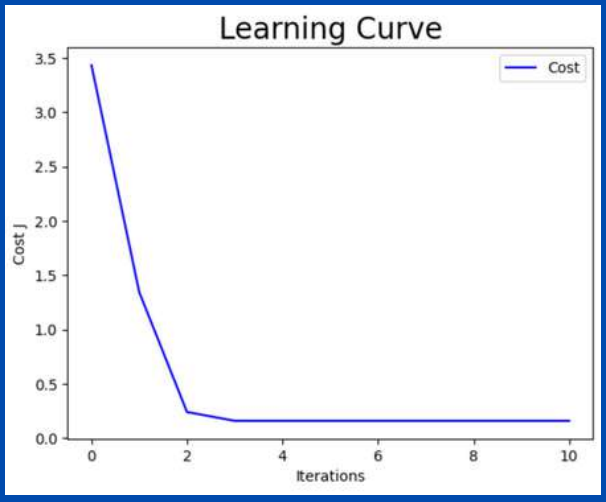

```
CPU times: user 2 µs, sys: 0 ns, total: 2 µs
Wall time: 5.48 µs
Iteration 01/10 : Cost =   0.6201
Iteration 02/10 : Cost =   0.5703
Iteration 03/10 : Cost =   0.4913
Iteration 04/10 : Cost =   0.3404
Iteration 05/10 : Cost =   0.3347
Iteration 06/10 : Cost =   0.3347
Iteration 07/10 : Cost =   0.3347
Iteration 08/10 : Cost =   0.3347
Iteration 09/10 : Cost =   0.3347
Iteration 10/10 : Cost =   0.3347
Estimated Final Cost is =  0.3347
```



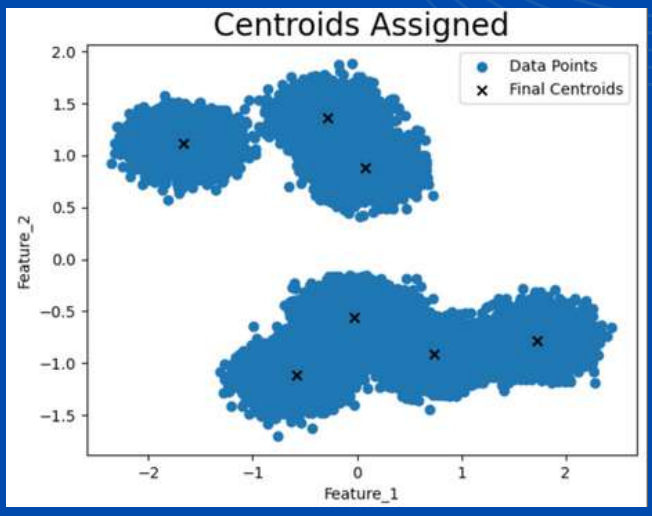
20. Seven Random Centroids Initialization:

The process begins by randomly selecting Seven centroids from the dataset and noting the resulting readings (cost or cluster assignments).

21. Results: After multiple initialization for seven centroids the converged costs observed were as follows - 0.1597, 0.8109, 0.6377, 0.3315, 0.3310 and 1.4268. Therefore the minima is 0.1597 .



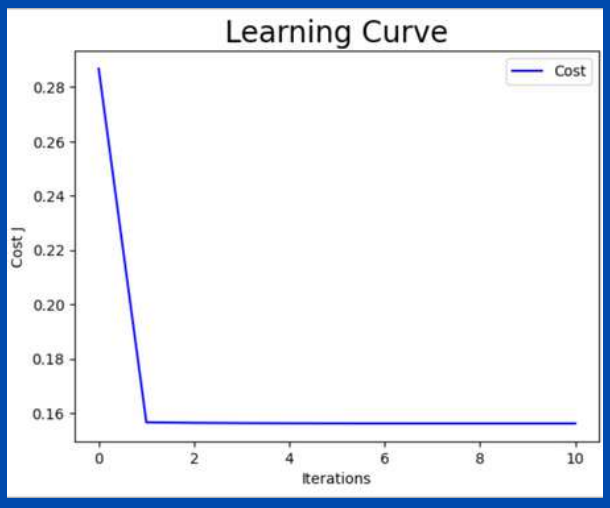

```
CPU times: user 4 µs, sys: 0 ns, total: 4 µs
Wall time: 6.68 µs
Iteration 01/10 : Cost =   1.3440
Iteration 02/10 : Cost =   0.2409
Iteration 03/10 : Cost =   0.1598
Iteration 04/10 : Cost =   0.1597
Iteration 05/10 : Cost =   0.1597
Iteration 06/10 : Cost =   0.1597
Iteration 07/10 : Cost =   0.1597
Iteration 08/10 : Cost =   0.1597
Iteration 09/10 : Cost =   0.1597
Iteration 10/10 : Cost =   0.1597
Estimated Final Cost is =  0.1597
```



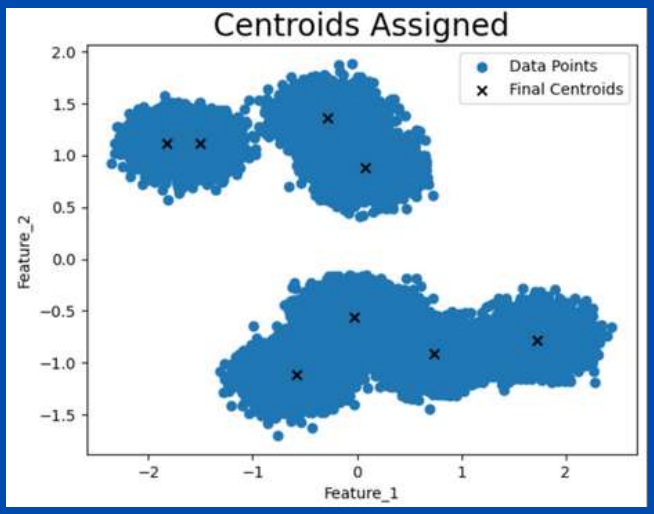
22. Eight Random Centroids Initialization:

The process begins by randomly selecting Eight centroids from the dataset and noting the resulting readings (cost or cluster assignments).

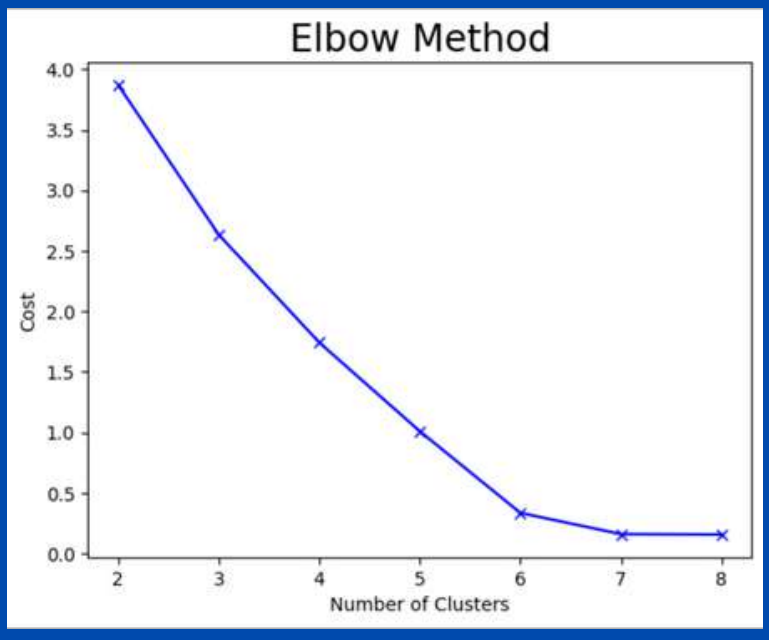
23. Results: After multiple initialization for eight centroids the converged costs observed were as follows - 0.1561 and 0.8347
Therefore the minima is 0.1561 .




```
CPU times: user 3 µs, sys: 0 ns, total: 3 µs
Wall time: 5.48 µs
Iteration 01/10 : Cost =   0.1566
Iteration 02/10 : Cost =   0.1564
Iteration 03/10 : Cost =   0.1563
Iteration 04/10 : Cost =   0.1562
Iteration 05/10 : Cost =   0.1562
Iteration 06/10 : Cost =   0.1562
Iteration 07/10 : Cost =   0.1562
Iteration 08/10 : Cost =   0.1562
Iteration 09/10 : Cost =   0.1561
Iteration 10/10 : Cost =   0.1561
Estimated Final Cost is =  0.1561
```

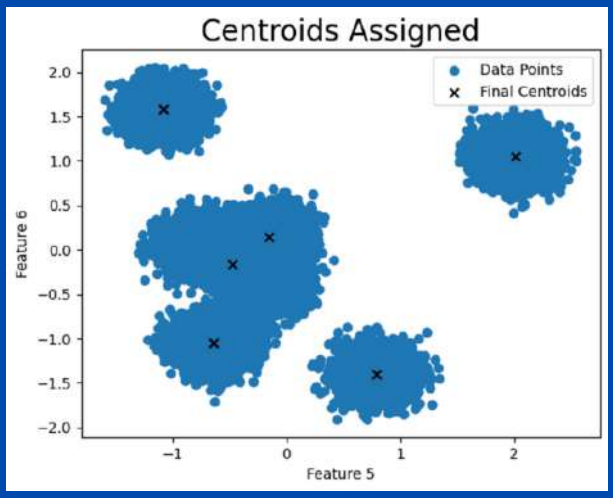


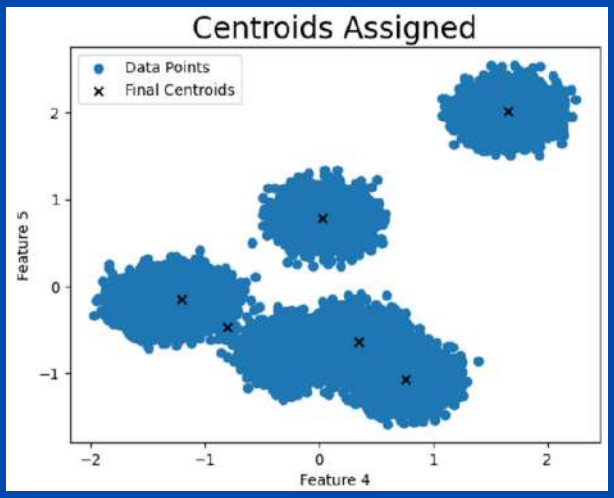
24. Elbow Method:

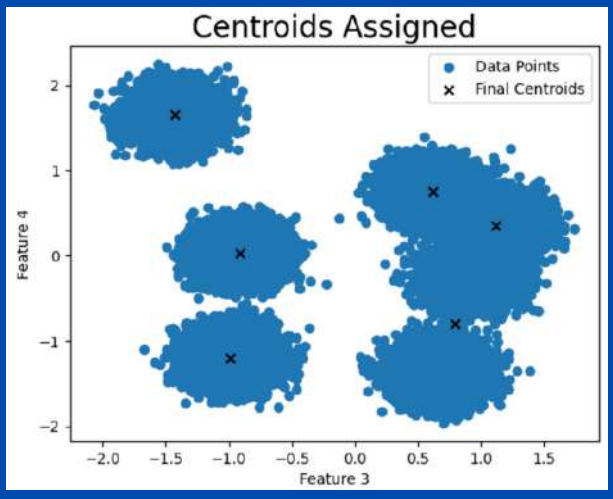


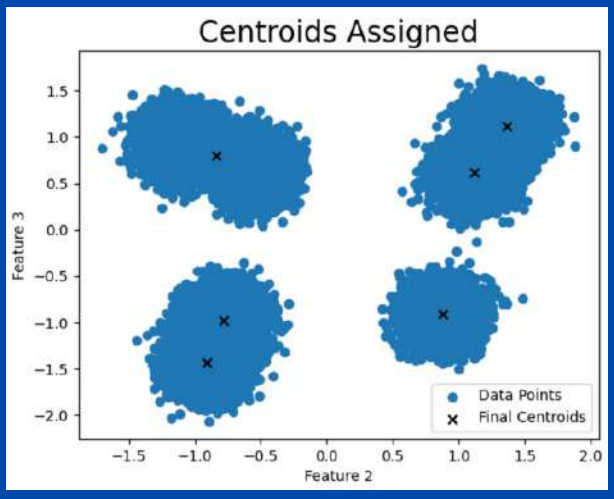
25. Conclusion from the Plot: The elbow in the graph appears to occur at the sixth or seventh cluster, indicating the optimal number of clusters for segmentation or grouping within the dataset.

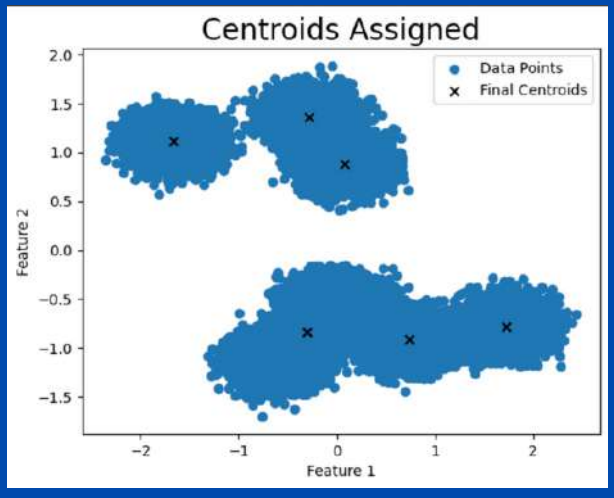
26. Six Cluster Plots:



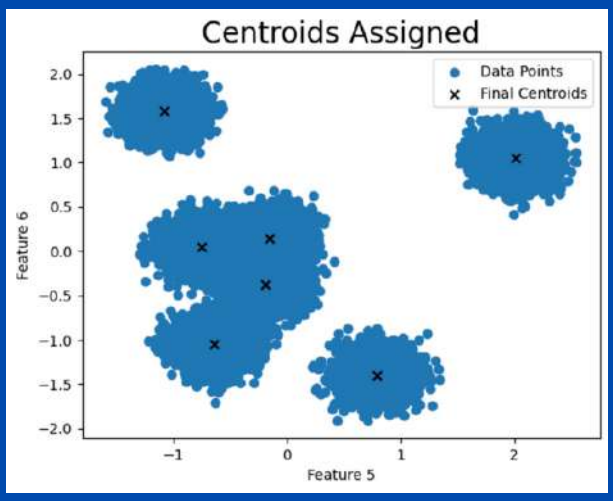


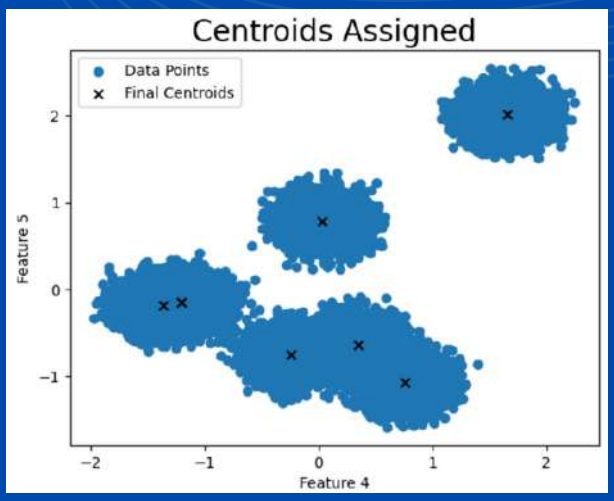


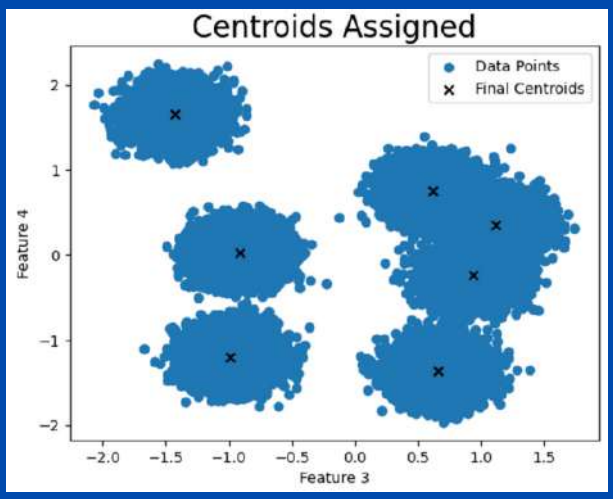


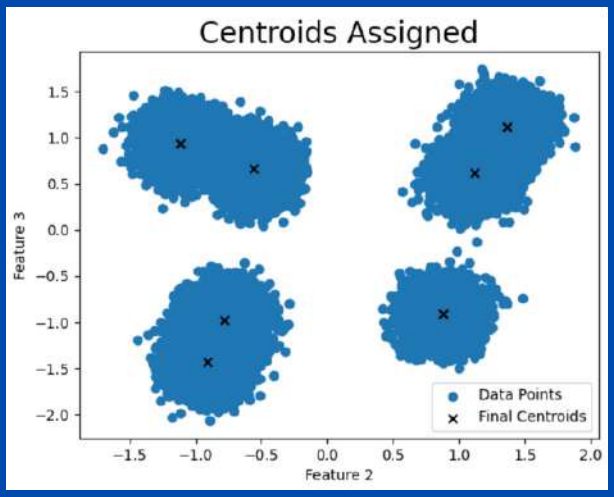


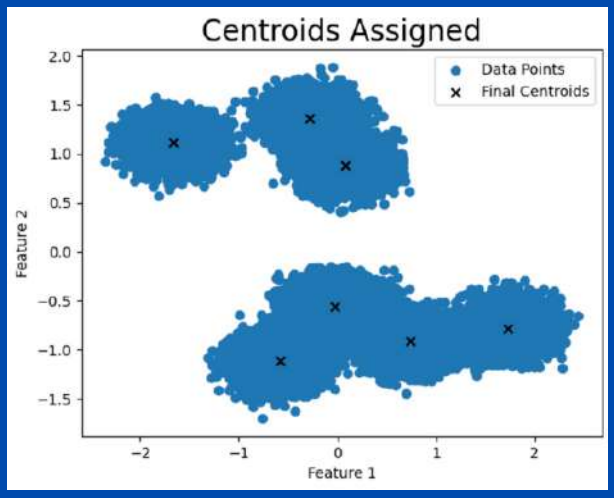
27. Seven Cluster Plots:











28. Conclusion : I chose seven clusters because the centroids in the plots were more aligned with the cluster centers, as opposed to being positioned between clusters, which was observed with six centroids

29. Saved Centroids and Clusters: The Final Centroids and the assigned custer number/index were saved in Google Drive in CSV format.

CYBER LABS

DECEMBER 2024

PROJECT REPORT

N - Layer Neural Network

PRESENTED TO

Machine Learning Division

PRESENTED BY

Arnav Tripathi
24JE0090

COMMENCEMENT OF THE MODEL

(Binary Label)

Problem Statement:

Design and implement an N-layer neural network capable of predicting binary class labels (0 or 1) given an input vector of size 1024. The goal is to achieve accurate predictions while optimizing computational efficiency and avoiding overfitting.

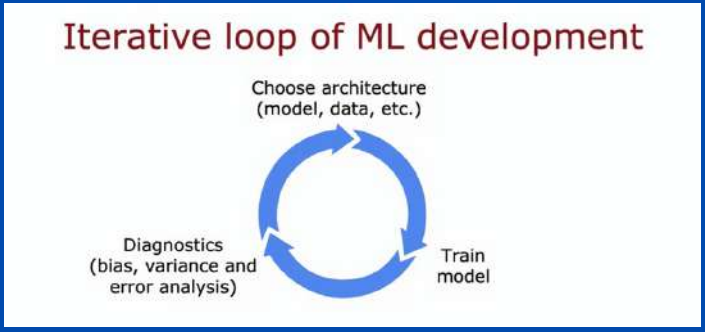
WORKING ON THE MODEL

(Utilizing the official dataset)

1.Mini-Batch Gradient Descent: Opted to implement Gradient Descent using eight mini-batches to ensure efficient model training while optimizing both time and memory usage. This approach allows the model to learn effectively with balanced computational efficiency and performance.

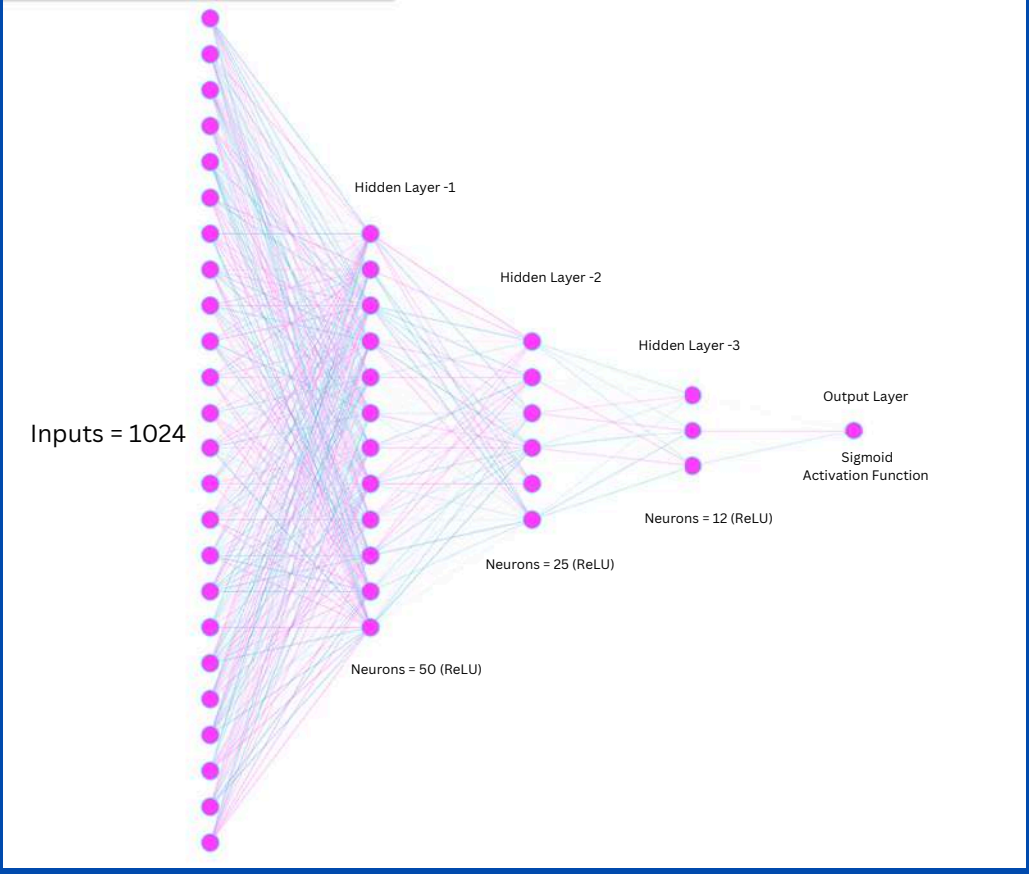
2. Data Partitioning: The dataset was divided into 80% for training, 20% for cross-validation (CV), and 20% for testing. The model is trained on the training set and evaluated using both the CV set and test set.

3. Identifying the Type of Target Variable : The task of the provided dataset is to predict binary class rather than continuous values. Therefore, Neural Network being a powerful algorithm would be an appropriate model for this scenario (as given in the problem statement).



4. Data Organization: The training set was organized into NumPy arrays to facilitate efficient data manipulation and model operations.

5. Neural Network Architecture: Designed a neural network with four dense layers, comprising 50 neurons in Layer 1, 25 neurons in Layer 2, 12 neurons in Layer 3, and 1 neuron in Layer 4. The hidden layers utilize the ReLU activation function, while the output layer employs a sigmoid activation function to predict a binary class label effectively.

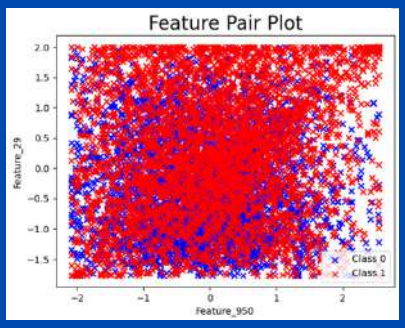


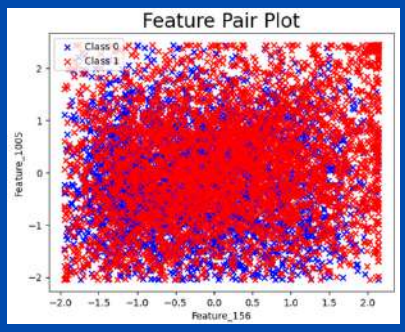
>>>Small Representation of Neural Network Architecture

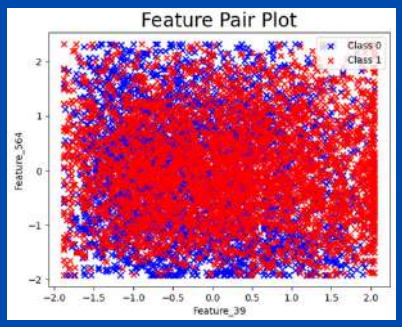
6. Checked for Skewed Dataset : I checked the dataset for skewness and found that the number of ones is exactly 50%, so the data isn't skewed. As a result, I can use accuracy as the evaluation metric. This ensures that the model's performance is assessed fairly, without the need for adjustments like F1 score.

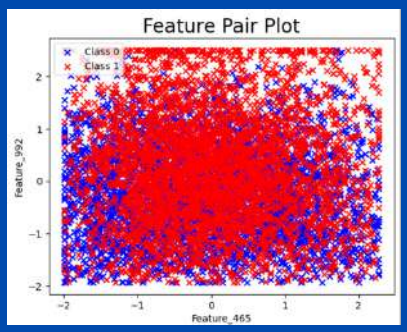
7. Feature Normalization: z-score normalization was applied to the features. This ensures all features lie in a similar scale, improving model performance. It works well with distance-based algorithms and gradient-based methods. The technique reduces the impact of outliers and ensures consistency between training and testing datasets. Overall, it leads to faster convergence and better model accuracy.

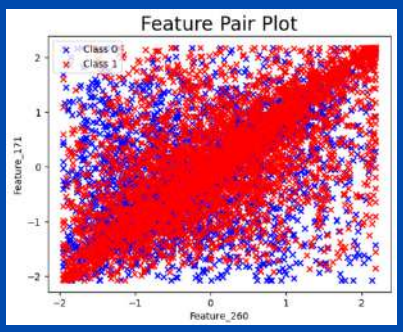
8. Data Visualisation: Plotted 100 random features in pairs for gaining insight into the patterns the data followed, and for better understanding of the data









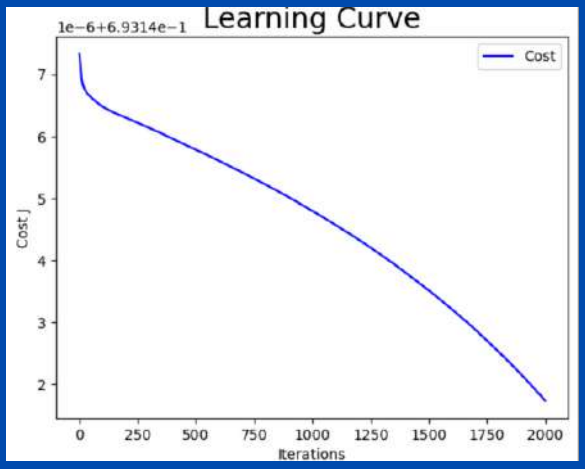


9. Plotting Limitations: The plots didn't showed any clear patterns. Thus, I moved on for further implementation.

10. Tuning other Hyperparameters: Started with a learning rate of 0.01 and 200 iterations. The learning curve after training for the first time was smooth but and decreasing but showed the Cost was still converging and further optimisation of features was required so I kept adjusting these hyperparameters to attain an acceptable learning curve.

11. Training Results : Started the training with 0.7 learning rate and the number of iterations were 2000.

Kept the learning rate high because model was taking long time to learn even after taking a mini batch.
This can be adjusted for next batches.




```
CPU times: user 3 µs, sys: 0 ns, total: 3 µs
Wall time: 8.34 µs
Epochs    0/2000 : Cost   0.6931
Epochs  200/2000 : Cost   0.6931
Epochs  400/2000 : Cost   0.6931
Epochs  600/2000 : Cost   0.6931
Epochs  800/2000 : Cost   0.6931
Epochs 1000/2000 : Cost   0.6931
Epochs 1200/2000 : Cost   0.6931
Epochs 1400/2000 : Cost   0.6931
Epochs 1600/2000 : Cost   0.6931
Epochs 1800/2000 : Cost   0.6931
TRAIN COST estimated is =  0.6931
```

The Estimated Training Cost is 0.6931 .
The Cross Validation Cost is 0.0866 .

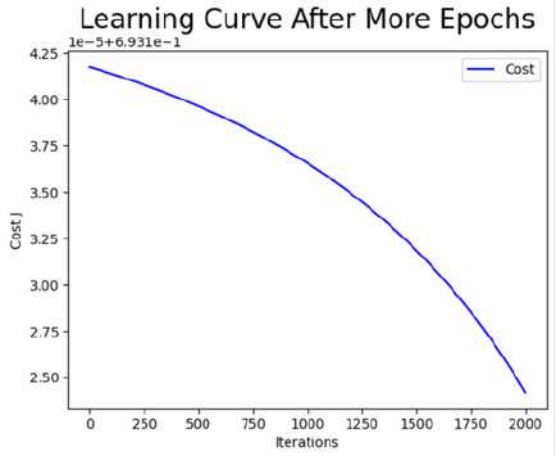
The Learning Curve showed continuous converging cost, this meant that training process was going right.

12. Continued with First Batch: Both Learning Curve and Training log insisted that Cost was still converging. So, I once again implemented gradient descent on the first batch with 0.5 alpha and 2000 additional iterations, but this time with the learned parameters. I will keep checking if the model overfits or don't generalize well.

13. Training Results :
The Estimated Training Cost is 00.6931 .
The Cross Validation Cost is 0.0866 .

The Learning Curve was smooth and decreasing, this meant that cost was trying to converge.

But since the cost wasn't showing significant improvement and enough epochs have been performed (4000) so I shifted to next batch.




```
CPU times: user 3 µs, sys: 0 ns, total: 3 µs
Wall time: 8.34 µs
Epochs    0/2000 : Cost   0.6931
Epochs  200/2000 : Cost   0.6931
Epochs  400/2000 : Cost   0.6931
Epochs  600/2000 : Cost   0.6931
Epochs  800/2000 : Cost   0.6931
Epochs 1000/2000 : Cost   0.6931
Epochs 1200/2000 : Cost   0.6931
Epochs 1400/2000 : Cost   0.6931
Epochs 1600/2000 : Cost   0.6931
Epochs 1800/2000 : Cost   0.6931
TRAIN COST estimated is =  0.6931
```

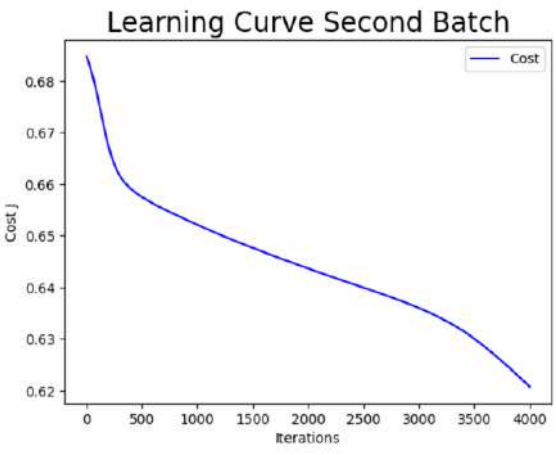
14. Second Batch: Read the second batch using pandas.
Applied the same normalization to the x_train of the second batch as I arranged them in NumPy arrays.

Started gradient descent with 4000 iterations and 0.5 learning rate. Decreased learning rate and increased iterations so model converges better for this batch than previous batch.

15. Training Results :
The Estimated Training Cost is 0.6208 .
The Cross Validation Cost is 0.0813 .

The Learning Curve was smooth and decreasing, this meant that cost was still converge.

And the cost wasn't showing significant improvements.




```
CPU times: user 3 µs, sys: 0 ns, total: 3 µs
Wall time: 6.68 µs
Epochs    0/4000 : Cost   0.6848
Epochs  400/4000 : Cost   0.6591
Epochs  800/4000 : Cost   0.6541
Epochs 1200/4000 : Cost   0.6503
Epochs 1600/4000 : Cost   0.6468
Epochs 2000/4000 : Cost   0.6437
Epochs 2400/4000 : Cost   0.6407
Epochs 2800/4000 : Cost   0.6377
Epochs 3200/4000 : Cost   0.6340
Epochs 3600/4000 : Cost   0.6284
TRAIN COST estimated of second batch is =  0.6208
```

16. Additional Iterations: Similarly, for this batch also I Implemented Gradient Descent again.

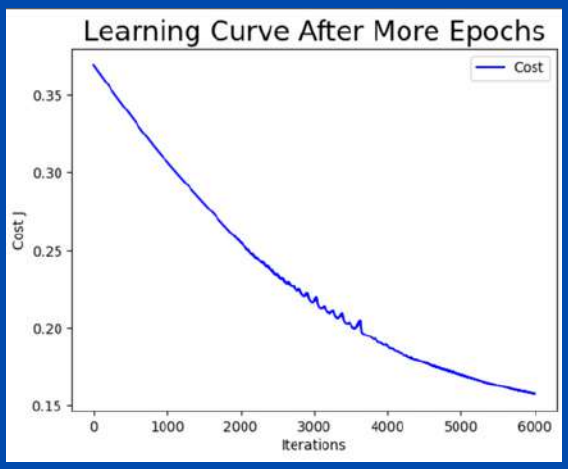
With 6000 iterations and 0.5 learning rate.

This will also gives us idea about the number of iterations required by a batch to converge.

17. Training Results: Estimated Train Cost is 0.1573 .
Estimated Cross Validation Cost is 0.1152 .

High Cross Validation even with Same amount of data showed neural network was learning data wasn't generalising well on unseen data.

The learning curve was however decreasing but less smooth.
Now, I shifted to next batch.

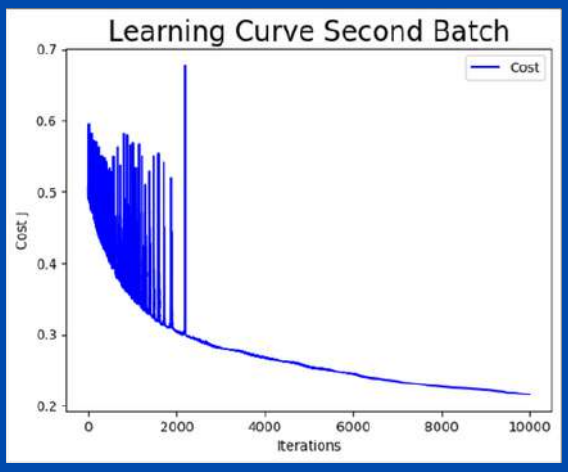



```
CPU times: user 3 µs, sys: 0 ns, total: 3 µs
Wall time: 7.63 µs
Epochs    0/6000 : Cost   0.3690
Epochs  600/6000 : Cost   0.3306
Epochs 1200/6000 : Cost   0.2958
Epochs 1800/6000 : Cost   0.2644
Epochs 3000/6000 : Cost   0.2174
Epochs 3600/6000 : Cost   0.2020
Epochs 4200/6000 : Cost   0.1835
Epochs 4800/6000 : Cost   0.1726
Epochs 5400/6000 : Cost   0.1641
TRAIN COST estimated is =  0.1573
```

18. Third Batch: Read the third batch using pandas.
Applied the same normalization to the x_train of the third batch as I arranged them in NumPy arrays.

Started gradient descent with 10000 iterations and 0.5 learning rate. This batch will further throw light on correct number of epochs.

19. Training Results: Estimated Train Cost is 0.2156 .
Estimated Cross Validation Cost is 0.1405 .




```
CPU times: user 4 µs, sys: 0 ns, total: 4 µs
Wall time: 9.54 µs
Epochs      0/10000 : Cost   0.5058
Epochs   1000/10000 : Cost   0.3519
Epochs   2000/10000 : Cost   0.3042
Epochs   3000/10000 : Cost   0.2812
Epochs   4000/10000 : Cost   0.2678
Epochs   5000/10000 : Cost   0.2542
Epochs   6000/10000 : Cost   0.2441
Epochs   7000/10000 : Cost   0.2341
Epochs   8000/10000 : Cost   0.2267
Epochs   9000/10000 : Cost   0.2223
TRAIN COST estimated of second batch is =  0.2156
```

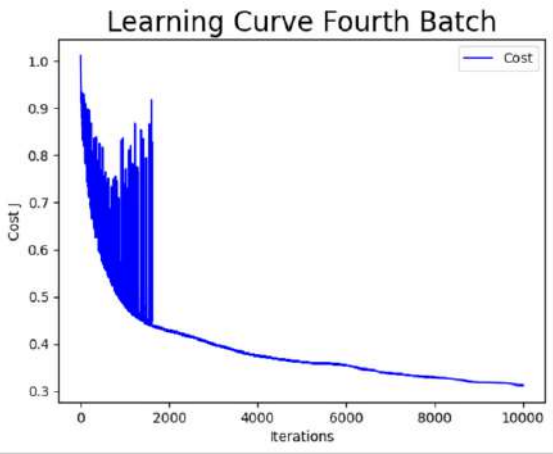
The oscillatory cost in starting is explanatory as it completely depends on the batch.

20. Fourth Batch: Read the fourth batch using pandas.
Applied the same normalization to the x_train of the fourth batch as I arranged them in NumPy arrays.

Started gradient descent with 10000 iterations and 0.5 learning rate.

21. Training Results: Estimated Train Cost is 0.3120 .
Estimated Cross Validation Cost is 0.1318 .

Estimated Test Cost is 0.1476 .




```
CPU times: user 2 µs, sys: 1e+03 ns, total: 3 µs
Wall time: 6.44 µs
Epochs      0/10000 : Cost   1.0113
Epochs   1000/10000 : Cost   0.4860
Epochs   2000/10000 : Cost   0.4278
Epochs   3000/10000 : Cost   0.3982
Epochs   4000/10000 : Cost   0.3746
Epochs   5000/10000 : Cost   0.3611
Epochs   6000/10000 : Cost   0.3538
Epochs   7000/10000 : Cost   0.3373
Epochs   8000/10000 : Cost   0.3288
Epochs   9000/10000 : Cost   0.3181
TRAIN COST estimated of second batch is =  0.3120
```

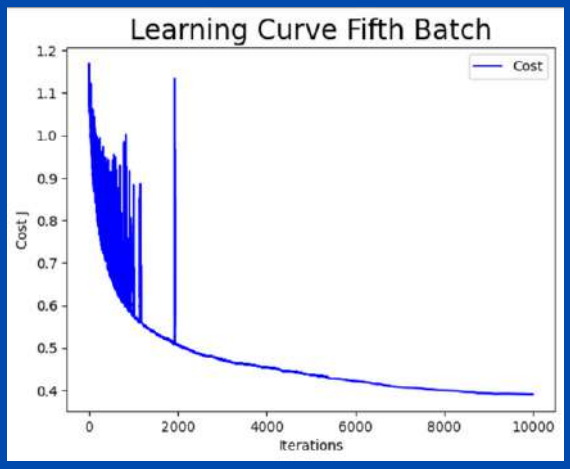
The oscillatory behaviour in starting is explainable and can happen when neural network is learning.It completely depends on the batch.
And, eventually the learning curve is decreasing so the neural network is ultimately learning.

22. Fifth Batch: Read the fifth batch using pandas.
Applied the same normalization to the x_train of the fifth batch as I arranged them in NumPy arrays.

Started gradient descent with 10000 iterations and 0.5 learning rate.

23. Training Results :
The Estimated Training Cost is 0.3910 .
The Cross Validation Cost is 0.1734 .

Estimated Test Cost is 0.1685 .




```
CPU times: user 4 µs, sys: 0 ns, total: 4 µs
Wall time: 11.2 µs
Epochs      0/10000 : Cost   1.1680
Epochs   1000/10000 : Cost   0.5837
Epochs   2000/10000 : Cost   0.5074
Epochs   3000/10000 : Cost   0.4722
Epochs   4000/10000 : Cost   0.4541
Epochs   5000/10000 : Cost   0.4359
Epochs   6000/10000 : Cost   0.4220
Epochs   7000/10000 : Cost   0.4074
Epochs   8000/10000 : Cost   0.3999
Epochs   9000/10000 : Cost   0.3930
TRAIN COST estimated of fifth batch is =  0.3910
```

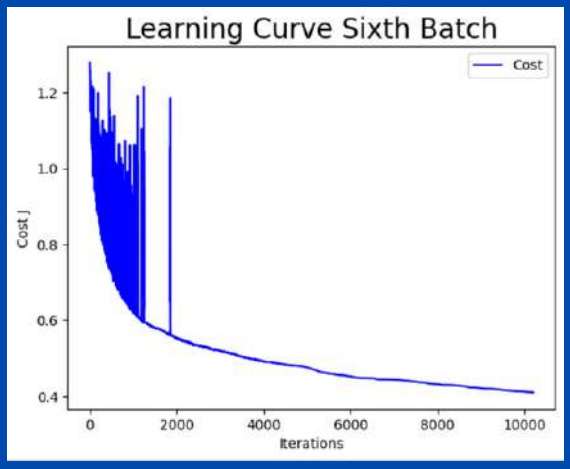
At a threshold of 0.5 ,
Estimated Test Accuracy is 72.40% .
Estimated Cross Validation Accuracy is 71.00% .
Estimated Train Accuracy is 96.17% .

24. Sixth Batch: Read the sixth batch using pandas.
Applied the same normalization to the x_train of the sixth batch as I arranged them in NumPy arrays.

Started gradient descent with 10200 iterations and 0.5 learning rate.
Increased the number of of iterations because I felt that few more iterations will approximately converge the cost completely.

25. Training Results :
The Estimated Training Cost is 0.4104 .
The Cross Validation Cost is 0.1458 .

Estimated Test Cost is 0.1703 .




```
CPU times: user 5 µs, sys: 0 ns, total: 5 µs
Wall time: 9.78 µs
Epochs      0/10200 : Cost   1.2782
Epochs   1020/10200 : Cost   0.6202
Epochs   2040/10200 : Cost   0.5512
Epochs   3060/10200 : Cost   0.5179
Epochs   4080/10200 : Cost   0.4903
Epochs   5100/10200 : Cost   0.4726
Epochs   6120/10200 : Cost   0.4498
Epochs   7140/10200 : Cost   0.4427
Epochs   8160/10200 : Cost   0.4298
Epochs   9180/10200 : Cost   0.4195
TRAIN COST estimated of sixth batch is =  0.4104
```

At a threshold of 0.5 ,
Estimated Test Accuracy is 74.30% .
Estimated Cross Validation Accuracy is 73.30% .
Estimated Train Accuracy is 96.12% .

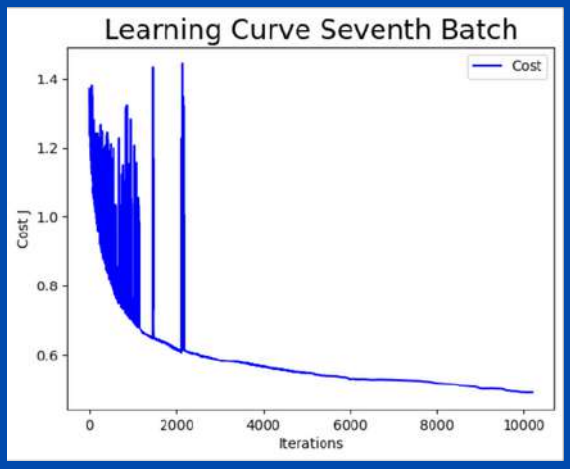
With increasing number of examples the accuracy of test set is increasing, which is a very positive sign that model is learning more complex patterns of data and is generalizing well.

26. Seventh Batch: Read the seventh batch using pandas.
Applied the same normalization to the x_train of the seventh batch as I arranged them in NumPy arrays.

Started gradient descent with 10200 iterations and 0.5 learning rate.

27. Training Results :
The Estimated Training Cost is 0.4909 .
The Cross Validation Cost is 0.1704 .

Estimated Test Cost is 0.1993 .




```
CPU times: user 3 µs, sys: 0 ns, total: 3 µs
Wall time: 6.68 µs
Epochs      0/10200 : Cost   1.3694
Epochs   1020/10200 : Cost   0.7103
Epochs   2040/10200 : Cost   0.6138
Epochs   3060/10200 : Cost   0.5830
Epochs   4080/10200 : Cost   0.5640
Epochs   5100/10200 : Cost   0.5467
Epochs   6120/10200 : Cost   0.5305
Epochs   7140/10200 : Cost   0.5272
Epochs   8160/10200 : Cost   0.5169
Epochs   9180/10200 : Cost   0.5013
TRAIN COST estimated of seventh batch is =  0.4909
```

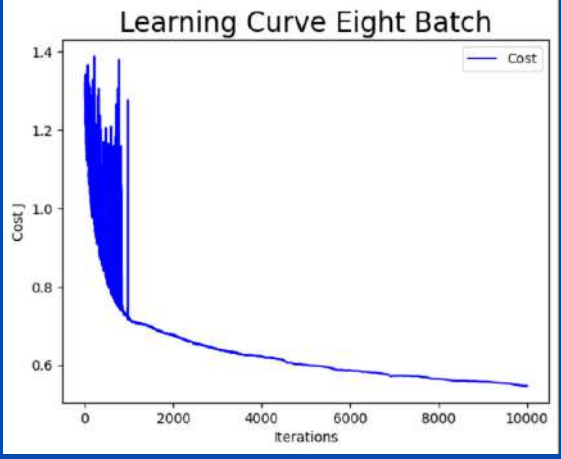
At a threshold of 0.5 ,
Estimated Test Accuracy is 71.480% .
Estimated Cross Validation Accuracy is 72.80% .
Estimated Train Accuracy is 95.69% .

28. Eight Batch: Read the eight batch using pandas.
Applied the same normalization to the x_train of the eight batch as I arranged them in NumPy arrays.

Started gradient descent with 10000 iterations and 0.5 learning rate.
Deccreased the number of of iterations because I felt that more iterations was making the model less generalised because the performance in the previous batch decreased a bit.

29. Training Results :
The Estimated Training Cost is 0.5469 .
The Cross Validation Cost is 0.1853 .

Estimated Test Cost is 0.1708 .



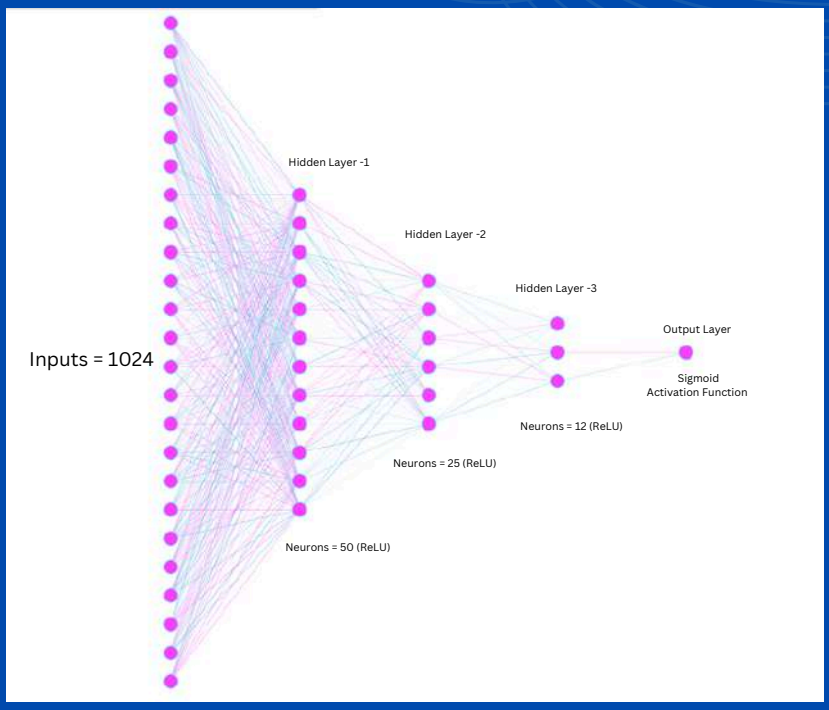

```
CPU times: user 3 µs, sys: 0 ns, total: 3 µs
Wall time: 6.44 µs
Epochs      0/10000 : Cost   1.3203
Epochs   1000/10000 : Cost   0.7181
Epochs   2000/10000 : Cost   0.6763
Epochs   3000/10000 : Cost   0.6397
Epochs   4000/10000 : Cost   0.6220
Epochs   5000/10000 : Cost   0.6000
Epochs   6000/10000 : Cost   0.5864
Epochs   7000/10000 : Cost   0.5724
Epochs   8000/10000 : Cost   0.5643
Epochs   9000/10000 : Cost   0.5573
TRAIN COST estimated of eight batch is =  0.5469
```

At a threshold of 0.5 ,
Estimated Test Accuracy is 73.40% .
Estimated Cross Validation Accuracy is 75.20% .
Estimated Train Accuracy is 95.51% .

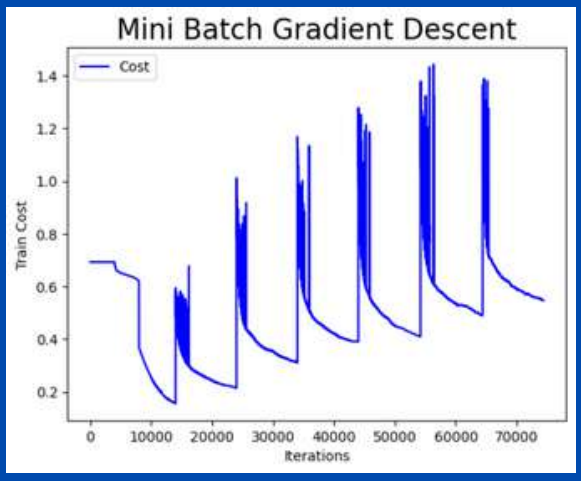
30. Training Successful: The Train, Cross Validation and Test Cost were quite less and the accuracy of all three were quite acceptable.

Hence, Training process was complete.

31. Predictions for Test Set: Predictions for the official nn_test.csv were computed using the trained model and saved as a CSV file in Google Drive.



>>>Neural Network architecture showing working of Final Model



>>>The graph of combined iterations versus combined training cost reveals a slight oscillatory behavior, characteristic of Mini-Batch Gradient Descent, where the model adapts to varying patterns across different mini-batches. Additionally, a minor increase in training error is observed as the number of training examples grows, making it more challenging for the model to fit all the data effectively.

Michael Snowdon Get verified
@Blogsbloke

It's vectors. All the way down.

COMMENCEMENT OF THE MODEL

(Class Label)

Problem Statement:

Design and implement an N-layer neural network capable of predicting class labels (1 to 10) given an input vector of size 1024 and binary labels. The goal is to achieve accurate predictions while optimizing computational efficiency and avoiding overfitting.

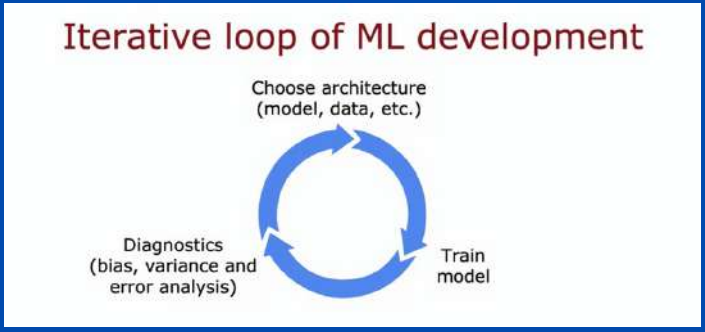
WORKING ON THE MODEL

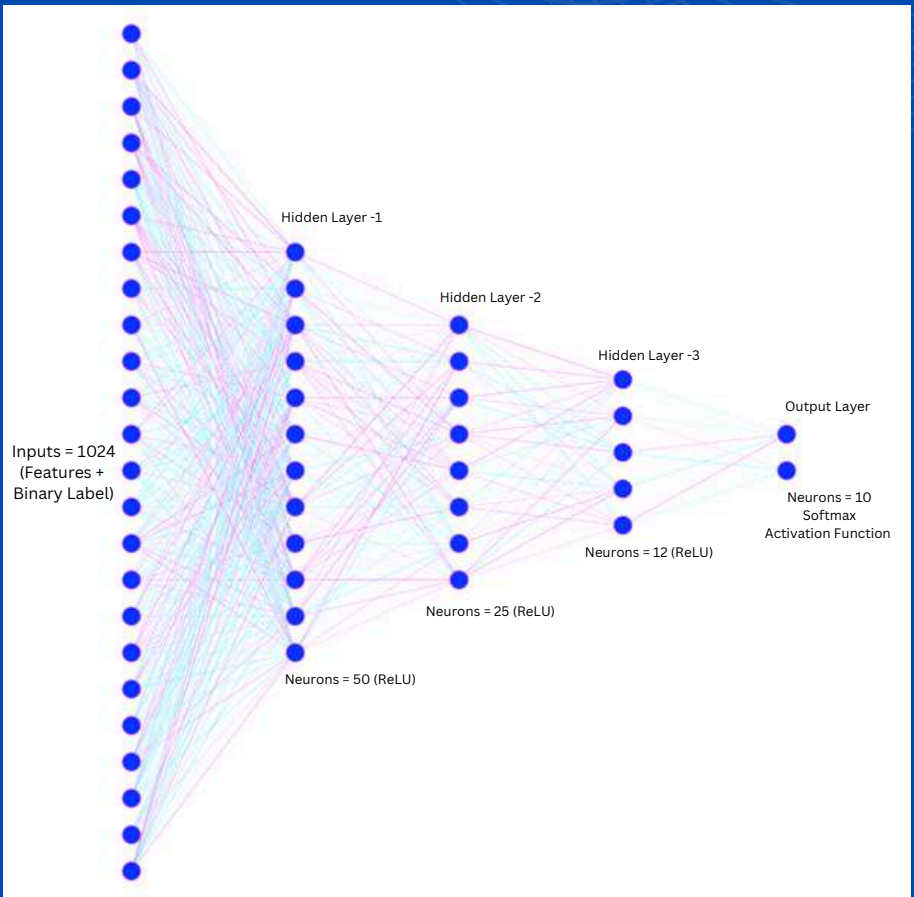
(Utilizing the official dataset)

1.Mini-Batch Gradient Descent: Opted to implement Gradient Descent using two mini-batches to ensure efficient model training while optimizing both time and memory usage. This approach allows the model to learn effectively with balanced computational efficiency and performance.

2. Data Partitioning: The dataset was divided into 80% for training, 20% for cross-validation (CV), and 20% for testing. The model is trained on the training set and evaluated using both the CV set and test set.

3. Identifying the Type of Target Variable : The task of the provided dataset is to predict class label rather than continuous values. Therefore, Neural Network being a powerful algorithm would be an appropriate model for this scenario (as given in the problem statement).





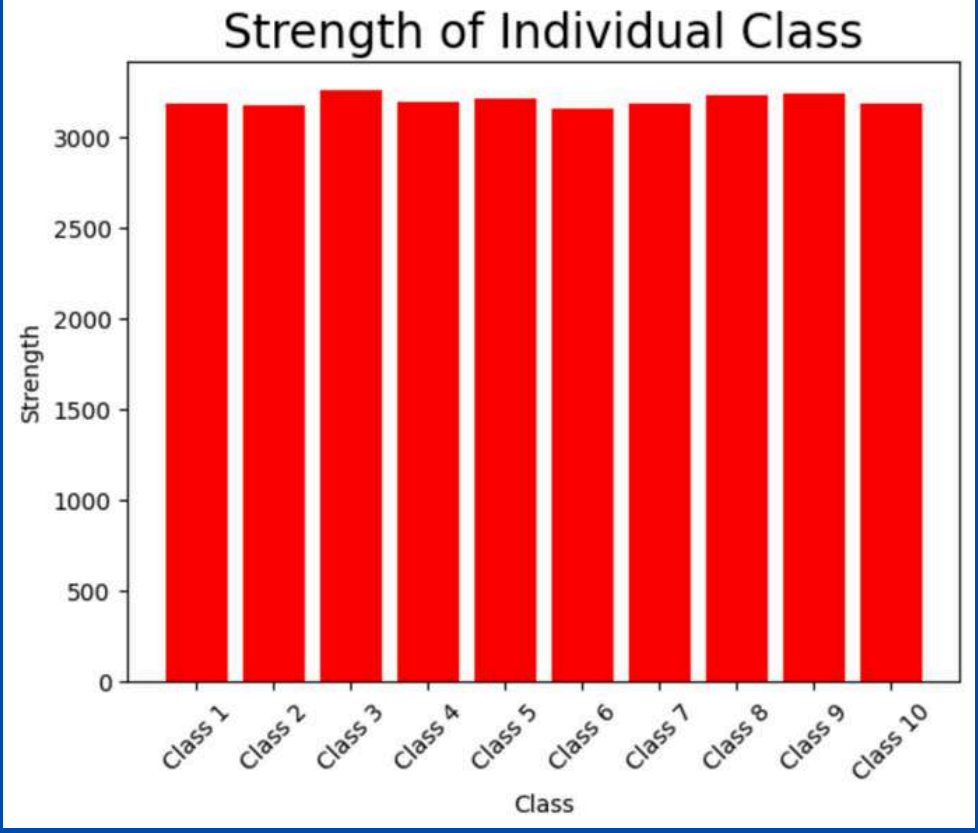
>>>Small Representation of Neural Network Architecture

4. Data Organization: The training set was organized into NumPy arrays to facilitate efficient data manipulation and model operations.

5. Neural Network Architecture: Designed a neural network with four dense layers, comprising 50 neurons in Layer 1, 25 neurons in Layer 2, 12 neurons in Layer 3, and 10 neurons in Layer 4. The hidden layers utilize the ReLU activation function, while the output layer employs a softmax activation function to predict a ten class label effectively.

6. Feature Normalization: z-score normalization was applied to the features. This ensures all features lie in a similar scale, improving model performance. It works well with distance-based algorithms and gradient-based methods. The technique reduces the impact of outliers and ensures consistency between training and testing datasets. Overall, it leads to faster convergence and better model accuracy.

7. Data Visualisation: Plotted a bar graph to visualise the strength of each label in the train data.

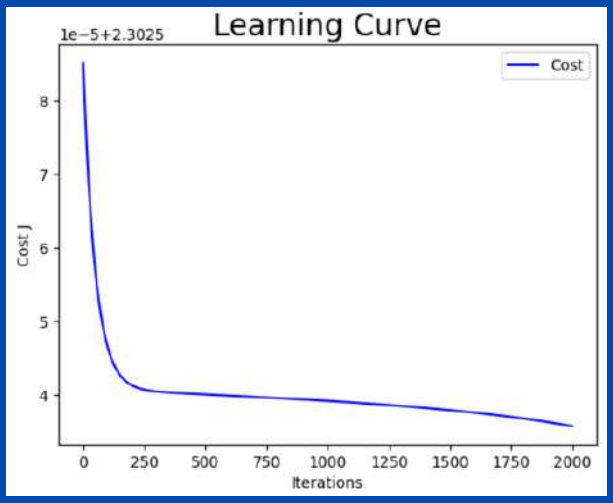


8. Plotting Observations: The plots told that each class was approximately equally present in train data.

9. Incremental Training Approach: Decided to train the model in stages, running it for a few iterations, observing the results, and then determining the next steps accordingly. This approach ensures the model trains effectively while minimizing the risk of overfitting on the training data.

10. Training Results : Started the training with 0.1 learning rate and the number of iterations were 2000.

These can be adjusted for next batches.

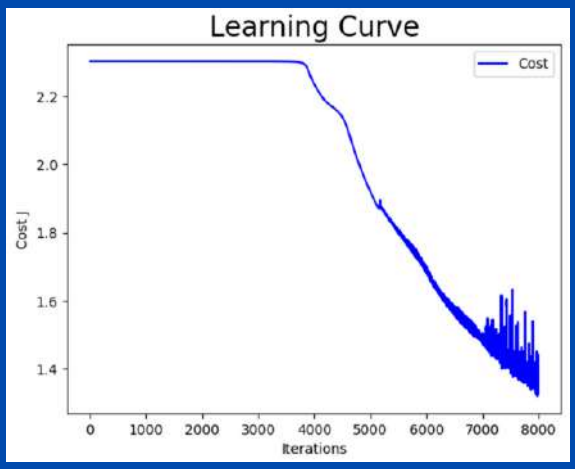



```
CPU times: user 3 µs, sys: 0 ns, total: 3 µs
Wall time: 7.87 µs
Epochs      0/2000 : Cost   2.3026
Epochs    200/2000 : Cost   2.3025
Epochs    400/2000 : Cost   2.3025
Epochs    600/2000 : Cost   2.3025
Epochs    800/2000 : Cost   2.3025
Epochs   1000/2000 : Cost   2.3025
Epochs   1200/2000 : Cost   2.3025
Epochs   1400/2000 : Cost   2.3025
Epochs   1600/2000 : Cost   2.3025
Epochs   1800/2000 : Cost   2.3025
TRAIN COST estimated is =  2.3025
```

The Estimated Training Cost is 2.3025 .

11. Continued with First Batch: Both Learning Curve and Training log insisted that Cost was still converging. So, I once again implemented gradient descent on the first batch with 0.06 alpha and 8000 , but this time with the learned parameters.
I will keep checking if the model overfits or don't generalize well.

12. Training Results :
The Estimated Training Cost is 1.3279 .
The Cross Validation Cost is 1.6874 .



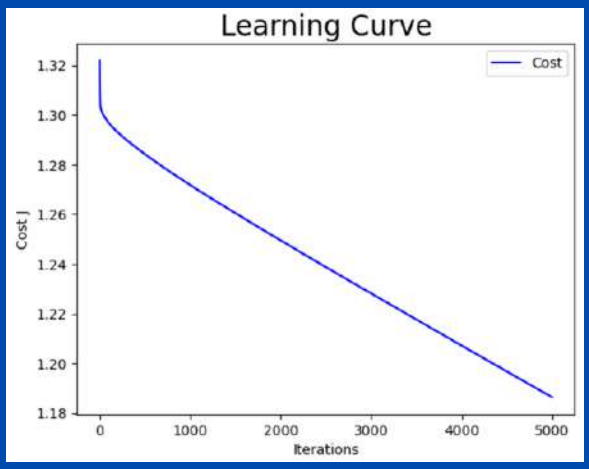

```
CPU times: user 4 µs, sys: 1e+03 ns, total: 5 µs
Wall time: 9.3 µs
Epochs      0/8000 : Cost   2.3025
Epochs    800/8000 : Cost   2.3025
Epochs   1600/8000 : Cost   2.3025
Epochs   2400/8000 : Cost   2.3025
Epochs   3200/8000 : Cost   2.3023
Epochs   4000/8000 : Cost   2.2351
Epochs   4800/8000 : Cost   2.0003
Epochs   5600/8000 : Cost   1.7804
Epochs   6400/8000 : Cost   1.5840
Epochs   7200/8000 : Cost   1.4623
TRAIN COST estimated is =  1.3279
```

Accuracy:
Estimated Train Accuracy is 50.49% .
Estimated Cross Validation Accuracy is 40.5% .

13. Continued with First Batch: Both Learning Curve and Training log insisted that Cost was still converging. So, I once again implemented gradient descent on the first batch with 0.006 alpha and 5000 additional iterations. Reduced the learning rate because learning curve was little oscillating.

I will keep checking if the model overfits or don't generalize well.

14. Training Results :
The Estimated Training Cost is 1.1864 .
The Cross Validation Cost is 1.7866 .



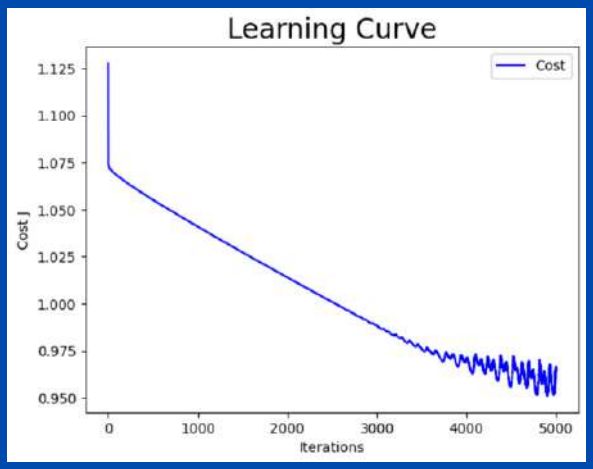

```
CPU times: user 4 µs, sys: 0 ns, total: 4 µs
Wall time: 12.4 µs
Epochs      0/5000 : Cost   1.3219
Epochs    500/5000 : Cost   1.2842
Epochs   1000/5000 : Cost   1.2718
Epochs   1500/5000 : Cost   1.2605
Epochs   2000/5000 : Cost   1.2495
Epochs   2500/5000 : Cost   1.2387
Epochs   3000/5000 : Cost   1.2281
Epochs   3500/5000 : Cost   1.2176
Epochs   4000/5000 : Cost   1.2072
Epochs   4500/5000 : Cost   1.1967
TRAIN COST estimated is =  1.1864
```

Accuracy
Estimated Train Accuracy is 55.00% .
Estimated Cross Validation Accuracy is 40.45% .

Model seemed to over fit a little so I added regularization for further optimization of parameters.

15. Continued with First Batch: Both Learning Curve and Training log insisted that Cost was still converging. So, I once again implemented gradient descent on the first batch with 0.008 alpha and 5000 additional iterations and regularization parameter of 0.01 .

16. Training Results :
The Estimated Training Cost is 0.9643 .
The Cross Validation Cost is 2.246 .



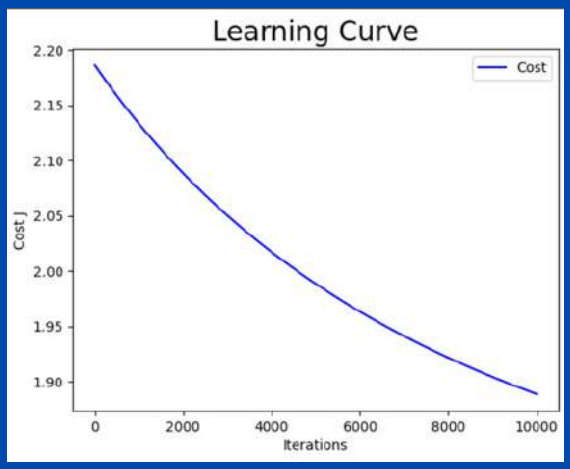

```
CPU times: user 3 µs, sys: 0 ns, total: 3 µs
Wall time: 8.34 µs
Epoch      0/5000 : Cost   1.1277
Epoch    500/5000 : Cost   1.0552
Epoch   1000/5000 : Cost   1.0409
Epoch   1500/5000 : Cost   1.0273
Epoch   2000/5000 : Cost   1.0140
Epoch   2500/5000 : Cost   1.0010
Epoch   3000/5000 : Cost   0.9885
Epoch   3500/5000 : Cost   0.9761
Epoch   4000/5000 : Cost   0.9710
Epoch   4500/5000 : Cost   0.9569
TRAIN COST estimated is =  0.9643
```

Accuracy
Estimated Train Accuracy is 63.490625% .
Estimated Cross Validation Accuracy is 38.49% .

17. Continued with First Batch: Testing different values of lambda because the model still seemed to overfit.

For lambda = 640 the train and test cost were comparable so I did last iterations with this value of regularization parameter.

18. Training Results :
The Estimated Training Cost is 1.8888 .
The Cross Validation Cost is 1.7540 .

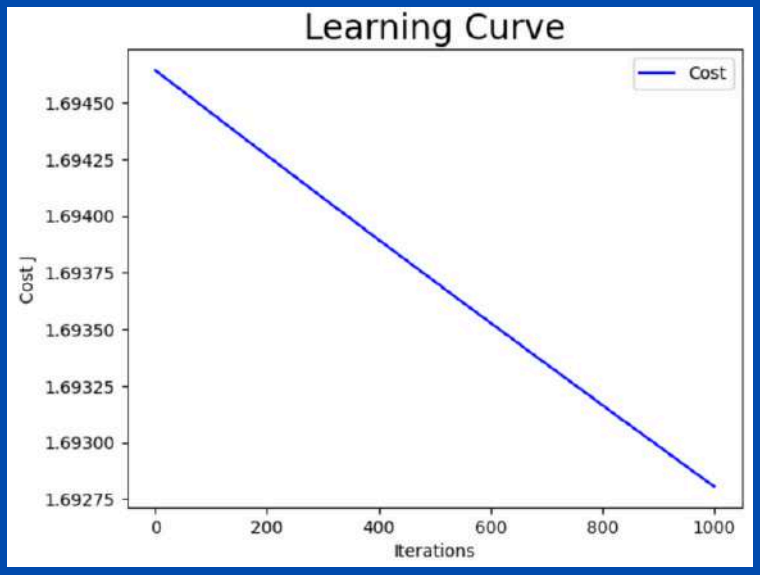



```
CPU times: user 2 µs, sys: 0 ns, total: 2 µs
Wall time: 6.2 µs
Epoch      0/10000 : Cost   2.1861
Epoch   1000/10000 : Cost   2.1330
Epoch   2000/10000 : Cost   2.0884
Epoch   3000/10000 : Cost   2.0502
Epoch   4000/10000 : Cost   2.0172
Epoch   5000/10000 : Cost   1.9885
Epoch   6000/10000 : Cost   1.9633
Epoch   7000/10000 : Cost   1.9412
Epoch   8000/10000 : Cost   1.9217
Epoch   9000/10000 : Cost   1.9043
TRAIN COST estimated is =  1.8888
```

Accuracy
Estimated Train Accuracy is 63.815% .
Estimated Cross Validation Accuracy is 40.325% .

19. Continued with First Batch: Both Learning Curve and Training log insisted that Cost was still converging.
So, I once again implemented gradient descent on the first batch with 0.02 alpha and 1000 additional iterations .

20. Training Results :
The Estimated Training Cost is 1.6928 .
The Cross Validation Cost is 1.6313 .
The Test Cost is 1.6005 .




```
CPU times: user 4 µs, sys: 0 ns, total: 4 µs
Wall time: 8.11 µs
Epoch      0/1000 : Cost   1.6946
Epoch    100/1000 : Cost   1.6945
Epoch    200/1000 : Cost   1.6943
Epoch    300/1000 : Cost   1.6941
Epoch    400/1000 : Cost   1.6939
Epoch    500/1000 : Cost   1.6937
Epoch    600/1000 : Cost   1.6935
Epoch    700/1000 : Cost   1.6933
Epoch    800/1000 : Cost   1.6932
Epoch    900/1000 : Cost   1.6930
TRAIN COST estimated is =  1.6928
```

Accuracy
Estimated Train Accuracy is 59.4125% .
Estimated Cross Validation Accuracy is 42.025% .
Estimated Test Accuracy is 42.9% .

Estimated F1 Score is 0.4195 .

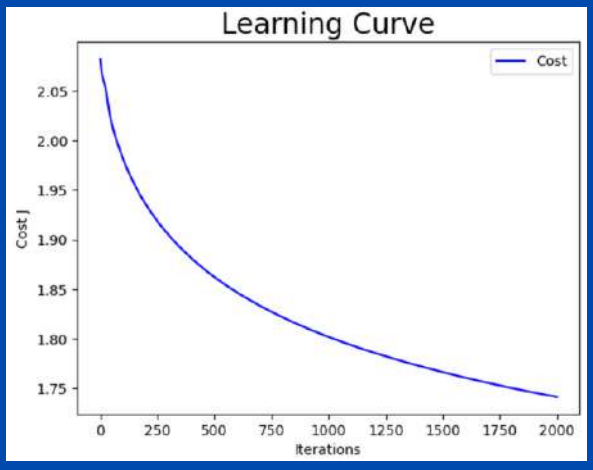
As the training log insists, the improvement in train cost are minimal so training for first batch was complete.

21. Second Batch: Read the second batch using pandas.
Applied the same normalization to the x_train of the eight batch as I arranged them in NumPy arrays.

Started Gradient Descent with 0.01 learning rate, and 2000 iterations and regularising parameter 540 .

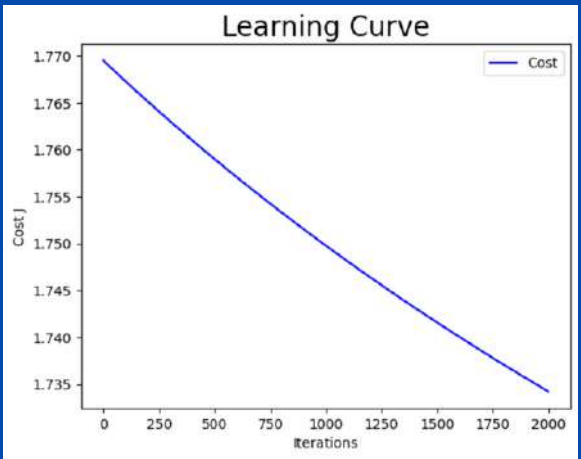
22. Diagnostic : After running 2000 epochs for batch 2
The Estimated Training Cost is 1.7412 .
The Cross Validation Cost is 1.5392 .

Accuracy
Estimated Train Accuracy is 52.190% .
Estimated Cross Validation Accuracy is 45.225% .




```
CPU times: user 3 µs, sys: 0 ns, total: 3 µs
Wall time: 5.48 µs
Epoch      0/2000 : Cost   2.0826
Epoch    200/2000 : Cost   1.9354
Epoch    400/2000 : Cost   1.8810
Epoch    600/2000 : Cost   1.8465
Epoch    800/2000 : Cost   1.8214
Epoch   1000/2000 : Cost   1.8018
Epoch   1200/2000 : Cost   1.7858
Epoch   1400/2000 : Cost   1.7723
Epoch   1600/2000 : Cost   1.7606
Epoch   1800/2000 : Cost   1.7503
TRAIN COST estimated is =  1.7412
```

23. Training Results: After another 2000 epochs on batch 2.
The Estimated Training Cost is 1.7342 .
The Cross Validation Cost is 1.5365 .

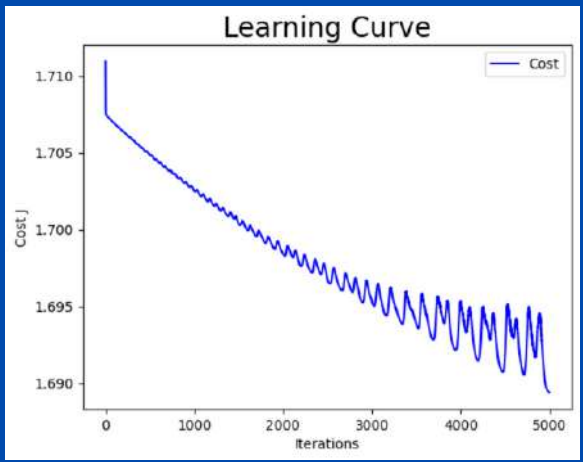



```
CPU times: user 3 µs, sys: 1e+03 ns, total: 4 µs
Wall time: 6.91 µs
Epoch      0/2000 : Cost   1.7695
Epoch    200/2000 : Cost   1.7651
Epoch    400/2000 : Cost   1.7610
Epoch    600/2000 : Cost   1.7570
Epoch    800/2000 : Cost   1.7533
Epoch   1000/2000 : Cost   1.7498
Epoch   1200/2000 : Cost   1.7464
Epoch   1400/2000 : Cost   1.7431
Epoch   1600/2000 : Cost   1.7400
Epoch   1800/2000 : Cost   1.7371
TRAIN COST estimated is =  1.7342
```

Accuracy
Estimated Train Accuracy is 53.306% .
Estimated Cross Validation Accuracy is 45.2% .

24. Continuing Gradient Descent:
Started Gradient Descent with 0.008 learning rate and 5000 iterations.

25. Training Results: After another 5000 epochs on batch 2.
The Estimated Training Cost is 1.6894 .
The Cross Validation Cost is 1.5191 .



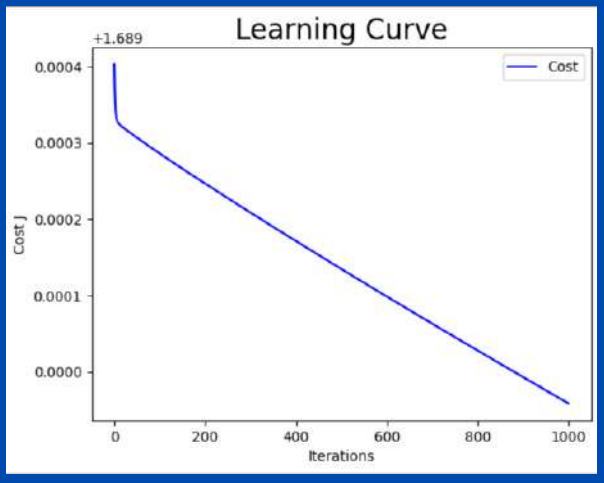
```
CPU times: user 3 µs, sys: 0 ns, total: 3 µs
Wall time: 6.44 µs
Epoch      0/5000 : Cost   1.7110
Epoch    500/5000 : Cost   1.7049
Epoch   1000/5000 : Cost   1.7025
Epoch   1500/5000 : Cost   1.7004
Epoch   2000/5000 : Cost   1.6984
Epoch   2500/5000 : Cost   1.6968
Epoch   3000/5000 : Cost   1.6953
Epoch   3500/5000 : Cost   1.6938
Epoch   4000/5000 : Cost   1.6953
Epoch   4500/5000 : Cost   1.6919
TRAIN COST estimated is =  1.6894
```

26. Continuing Second Batch:Started Gradient Descent with 0.001 learning rate and last 1000 iterations.

27. Training Results :
The Estimated Training Cost is 1.6890 .
The Cross Validation Cost is 1.5192 .
The Test Cost is 1.4889 .

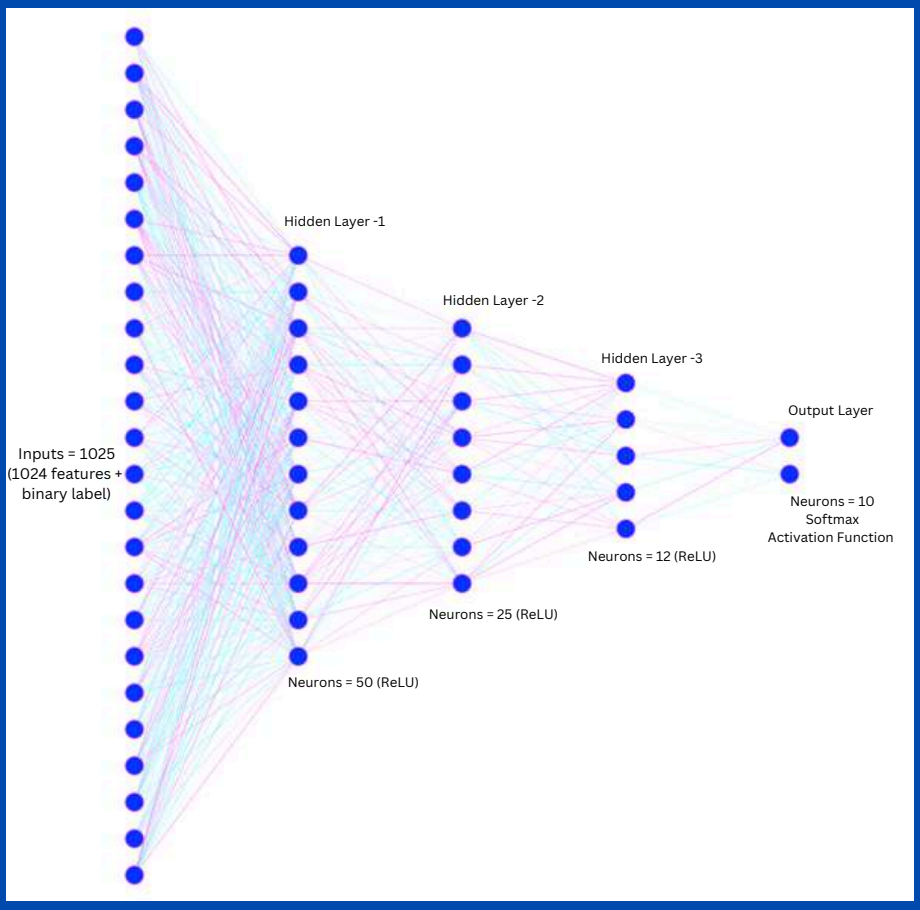
Accuracy
Estimated Train Accuracy is 57.10% .
Estimated Cross Validation Accuracy is 46.80% .
Estimated Test Accuracy is 47.075% .

F1 Score for cross validation set is 0.4691 .




```
CPU times: user 4 µs, sys: 0 ns, total: 4 µs
Wall time: 12.4 µs
Epoch      0/1000 : Cost   1.6894
Epoch    100/1000 : Cost   1.6893
Epoch    200/1000 : Cost   1.6892
Epoch    300/1000 : Cost   1.6892
Epoch    400/1000 : Cost   1.6892
Epoch    500/1000 : Cost   1.6891
Epoch    600/1000 : Cost   1.6891
Epoch    700/1000 : Cost   1.6891
Epoch    800/1000 : Cost   1.6890
Epoch    900/1000 : Cost   1.6890
TRAIN COST estimated is =  1.6890
```

28. Predictions for Test Set: Predictions for the official nn_test.csv were computed using the trained model and saved as a CSV file in Google Drive.



>>>Neural Network architecture showing working of Final Model