ML Report Assignment 1

SECTION - A

(a) No, the fact that two variables exhibit a strong correlation with a third variable does not necessarily imply that they will also display a high degree of correlation with each other. Correlation is a measure of the linear relationship between two variables, and each pair of variables can have its own unique relationship, regardless of their relationship with a third variable.

Here's an example to illustrate this:

Let's say you have three variables: A, B, and C.

- Variable A and Variable B both have a strong positive correlation with Variable C. This means that as the values of A and B increase, the values of C also tend to increase.

However, this doesn't automatically mean that Variable A and Variable B will exhibit a strong correlation with each other. They might have a high degree of correlation with C individually due to their relationship with it, but their relationship with each other could be quite different. They could have a weak correlation, no correlation, or even a negative correlation with each other.

(b)

The defining criteria for a function to be classified as a logistic function include:

1. **S-Shaped Curve:** A logistic function must exhibit an S-shaped curve, which means that it starts with a gradual increase, becomes steeper as it progresses, and then levels off as it approaches the upper and lower limits (asymptotes). This characteristic S-shape is a fundamental property of logistic functions.
2. **Range:** The output of a logistic function should be constrained to a specific range, typically between 0 and 1. As the input becomes very large (positive or negative), the output approaches the asymptotes of 1 and 0.
3. **Symmetry and Center:** A logistic function is symmetric about its midpoint, which is typically located at x=0. This means that the values of the function are symmetrically distributed around this central point.

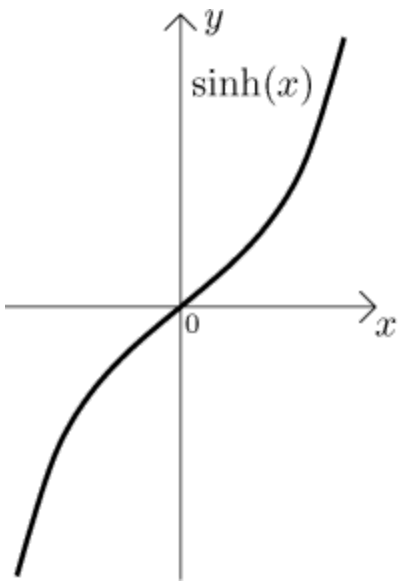
4. **Monotonicity:** A logistic function is monotonically increasing, which means that as the input increases, the output also increases. However, the rate of increase changes as input moves away from the center.
5. **Continuity and Differentiability:** Logistic functions are typically continuous and differentiable everywhere within their domain. This is important for many applications, including optimization and gradient-based algorithms.

The most common form of the logistic function is the sigmoid function, which is given by:

f(x) = 1/(1+e^-x) this function satisfies all above mentioned criterias

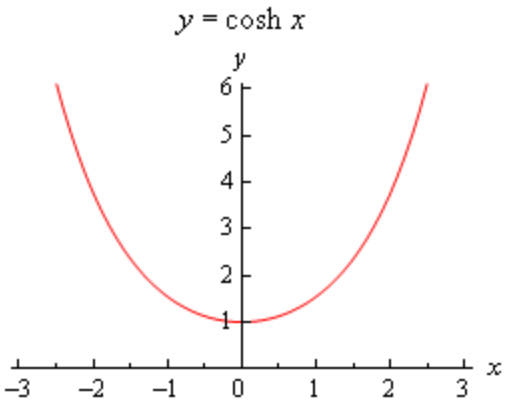
sinh(x): It is not a valid logistic function. It does not exhibit an S-shaped curve, and its output range is not constrained within 0 and 1.

sinh(x) = (e^x - e^(-x)) / 2



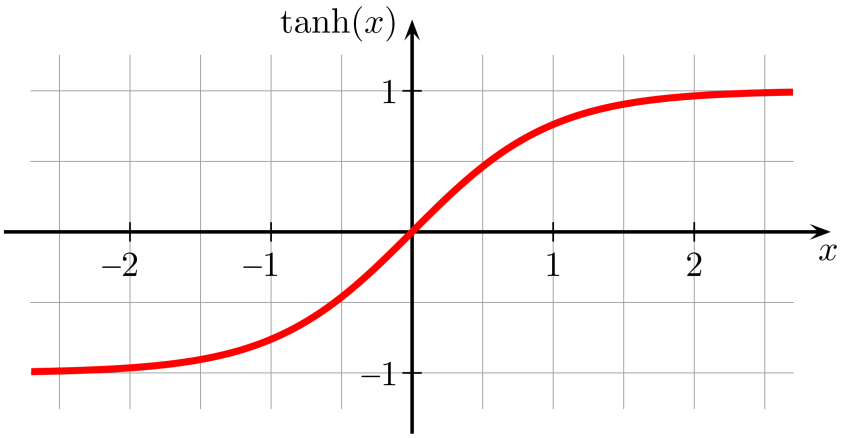
cosh(x): It is not a valid logistic function either it lacks the S-shaped curve and the constrained output range between 0 and 1.

cosh(x) = (e^x + e^(-x)) / 2

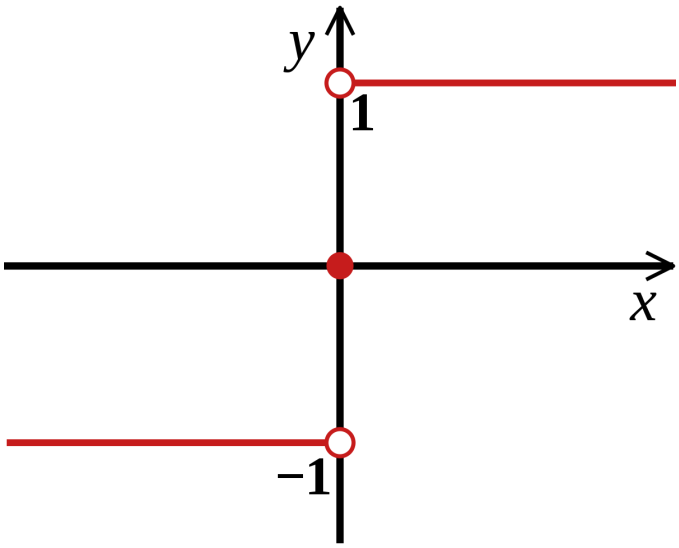


tanh(x) : It is a valid logistic function. It exhibits an S-shaped curve and its output range is between -1 and 1. While it's not constrained between 0 and 1, it still meets the criteria of an S-shaped curve and constrained range.

tanh(x) = (e^x - e^(-x)) / (e^x + e^(-x))



signum(x) : It is not a valid logistic function. It produces discrete outputs of -1, 0, and 1, and does not exhibit an S-shaped curve.



(c) For very sparse datasets, the "Leave-One-Out Cross-Validation" (LOOCV) technique can be beneficial. LOOCV is particularly advantageous when dealing with sparse data because it utilizes each data point for validation individually, which helps in making the most of the limited available data.

Leave-One-Out Cross Validation (LOOCV) is a model validation technique where a single sample from the dataset is used as the test set, and the remaining samples are used as the training set. The process is repeated n times, where n is the number of samples in the dataset. Each sample is used once as the test set and the model is trained on the remaining n-1 samples. The average performance score across all n iterations is used to validate the model.

Difference from K-Fold Cross-Validation:

In K-Fold Cross-Validation, the dataset is divided into K subsets (folds), and the model is trained K times, each time using K-1 folds for training and the remaining fold for validation.

(d)

Let $(x_1, y_1), (x_2, y_2), (x_3, y_3) \dots (x_n, y_n)$ be n data points

Let linear regression line be $\hat{y} = mx + c$

for i^{th} data point

$\hat{y}_i = mx_i + c \rightarrow$ prediction

$y_i \rightarrow$ Actual output

So, Error, $E = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$

$$\Rightarrow E = \frac{1}{n} \sum_{i=1}^{n} (y_i - mx_i - c)^2$$

$$\Rightarrow \frac{\partial E}{\partial m} = \frac{-2}{n} \sum_{i=1}^{n} (y_i - mx_i - c) x_i = 0 \quad - (1)$$

$$\Rightarrow \frac{\partial E}{\partial c} = \frac{-2}{n} \sum_{i=1}^{n} (y_i - mx_i - c) = 0 \quad - (11)$$

from (1) we get $c = y_i - mx_i$ ($\bar{y}$ = mean of y_i's, $\bar{x}$ = mean of x_i's)

(Assuming $x_i \neq 0$) or $\boxed{c = \bar{y} - m\bar{x}}$

put c in (11)

we get $\sum_{i=1}^{n} (y_i - mx_i - \bar{y} + m\bar{x}) = 0$

$$\Rightarrow \sum_{i=1}^{n} (y_i - \bar{y} - m(x_i - \bar{x})) = 0$$

$$\Rightarrow \sum_{i=1}^{n} (y_i - \bar{y}) - m \sum_{i=1}^{n} (x_i - \bar{x}) = 0$$

(Multiply $(x_i - \bar{x})$ on each term both side)

$$\Rightarrow \sum_{i=1}^{n} (y_i - \bar{y})(x_i - \bar{x}) - m \sum_{i=1}^{n} (x_i - \bar{x})^2 = 0$$

Hence, $\boxed{m = \frac{\sum_{i=1}^{n} (y_i - \bar{y})(x_i - \bar{x})}{\sum_{i=1}^{n} (x_i - \bar{x})^2}}$

(e) In the simple linear regression model:

Y = α + βx + ε

The parameters to be estimated are:

- α (intercept)
- β (slope)
- σ (standard deviation of the error term ε)

$$y = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

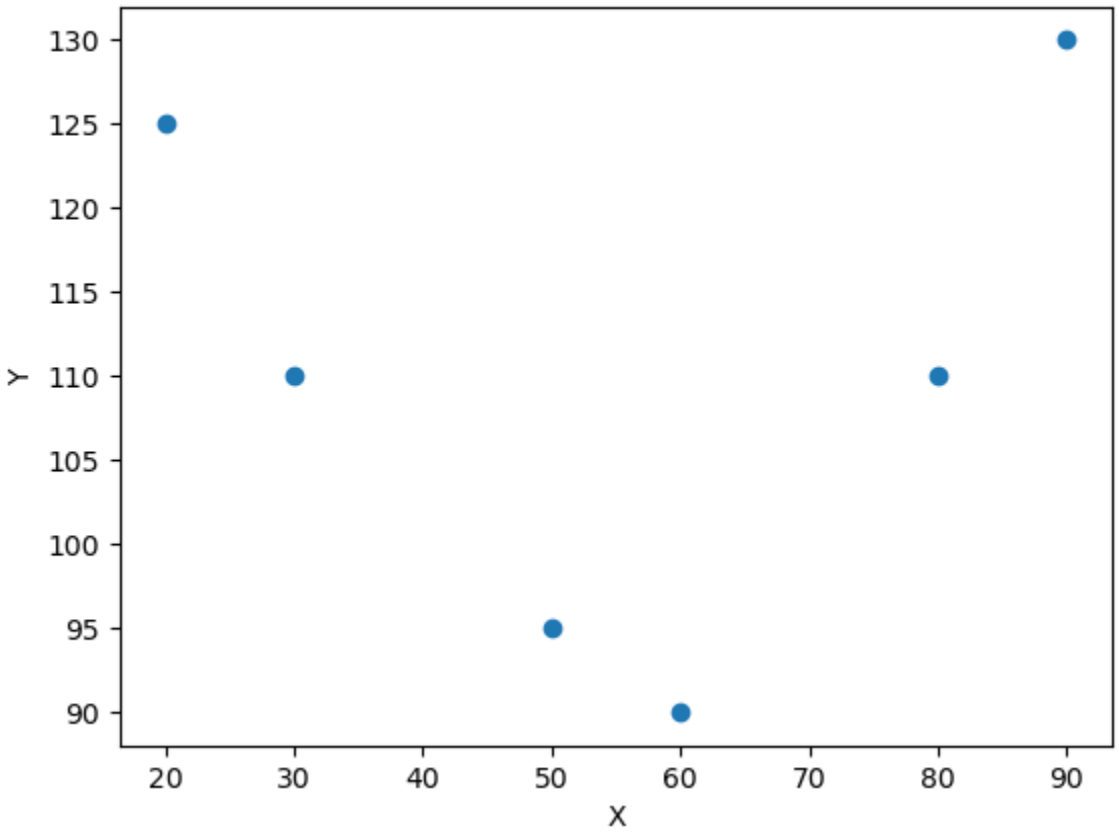
$$\mu = \text{Mean}$$
$$\sigma = \text{Standard Deviation}$$
$$\pi \approx 3.14159\cdots$$
$$e \approx 2.71828\cdots$$

So, the correct answer is:

(a) α, β, σ, because we can express ε in terms of σ hence we will directly estimate σ instead of ε (also clear from above formula)

(f) Given data:

X = [20, 30, 50, 60, 80, 90]
Y = [125, 110, 95, 90, 110, 130]



As clear from the above scatter plot we can observe an upward parabola so the coefficient of x^2 should be positive (β2 >0) hence option (d) Y= α + β1x + β2x2+ε β2 > 0 is correct.

SECTION - B

(a) Brief explanation of what the code does:

1. **LogisticRegression Class**: This class represents the logistic regression model. It has methods for sigmoid activation, cross-entropy loss calculation, training the model using SGD, and testing the model's performance.
2. **Sigmoid Function**: The `sigmoid` function calculates the sigmoid activation of a given input `z`.
3. **Cross-Entropy Loss**: The `cross_entropy_loss` function computes the binary cross-entropy loss between true labels (`y_true`) and predicted probabilities (`y_pred`). It

also includes numerical stability improvements using epsilon.

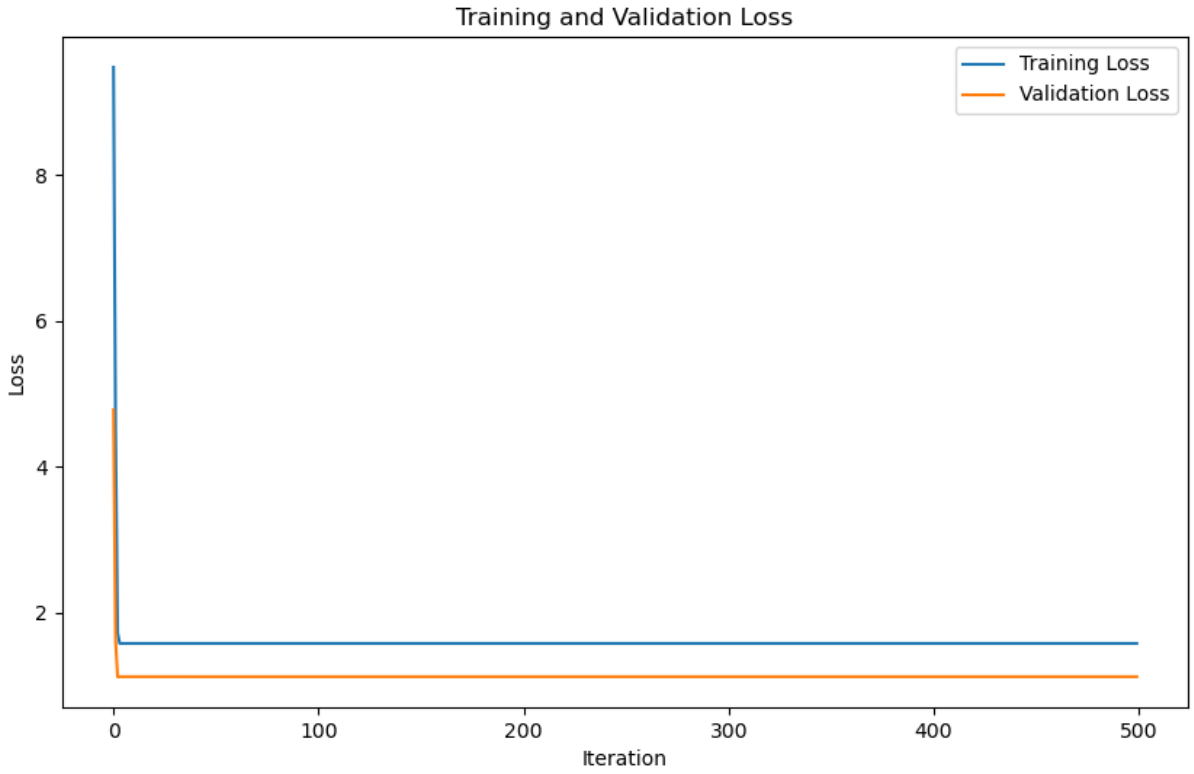
4. **Training**: The `train` method trains the logistic regression model. It iterates through the training data for a specified number of iterations using SGD. In each iteration, it calculates the gradient of the loss with respect to the model's parameters and updates the model's parameters accordingly. It also calculates and tracks training and validation losses and accuracies during training.
5. **Testing**: The `test` method evaluates the trained model on a test dataset. It computes metrics such as accuracy, precision, recall, F1 score, and a confusion matrix.
6. **Data Loading and Preprocessing**: The code loads a dataset from a CSV file, separates features and labels, standardizes the features, and adds a bias term.
7. **Hyperparameters**: It allows the user to specify the learning rate and the number of training iterations.
8. **Training and Visualization**: It initializes the logistic regression model, trains it using the training data, and visualizes the training and validation loss and accuracy over iterations using matplotlib.
9. **Testing and Metrics**: Finally, it evaluates the trained model on a test dataset and prints out the confusion matrix, accuracy, precision, recall, and F1 score to assess the model's performance.

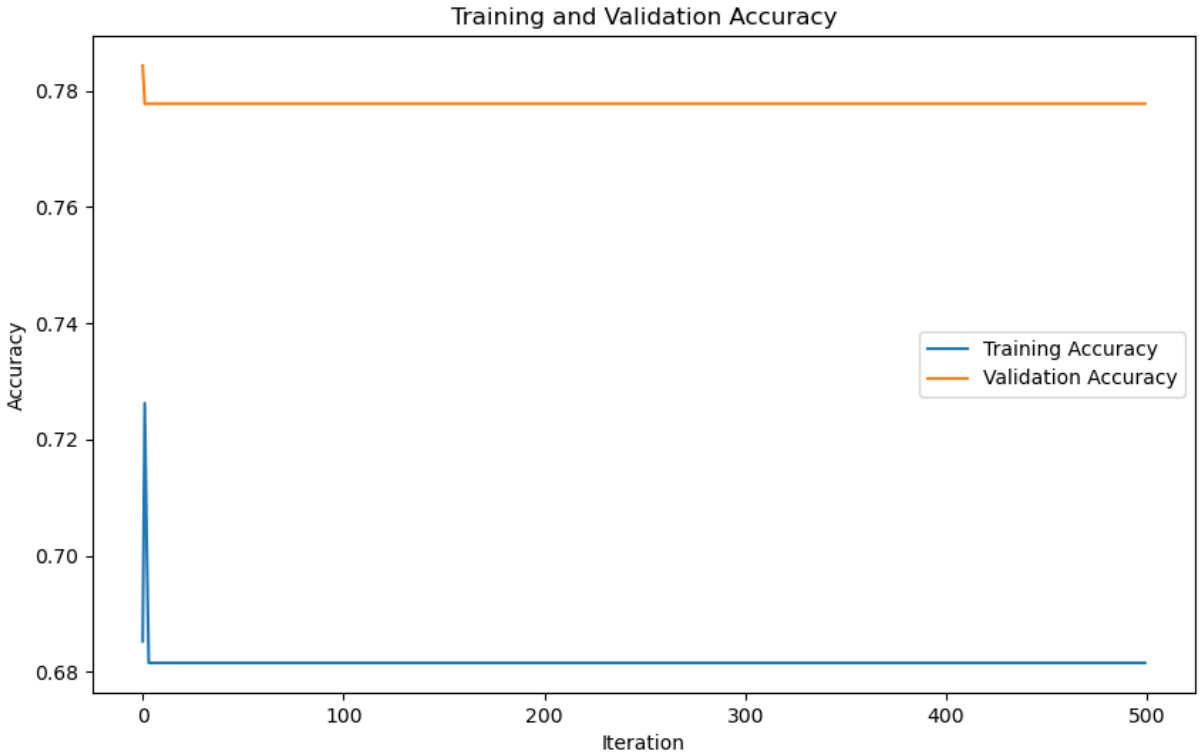
In each epoch of stochastic gradient descent I have calculated **Training loss**, **validation loss**, **Training accuracy** and **Validation accuracy.**

Convergence : Model converges early for a high learning rate and converges late for a low leaning rate

Comparison and analysis of plot:-

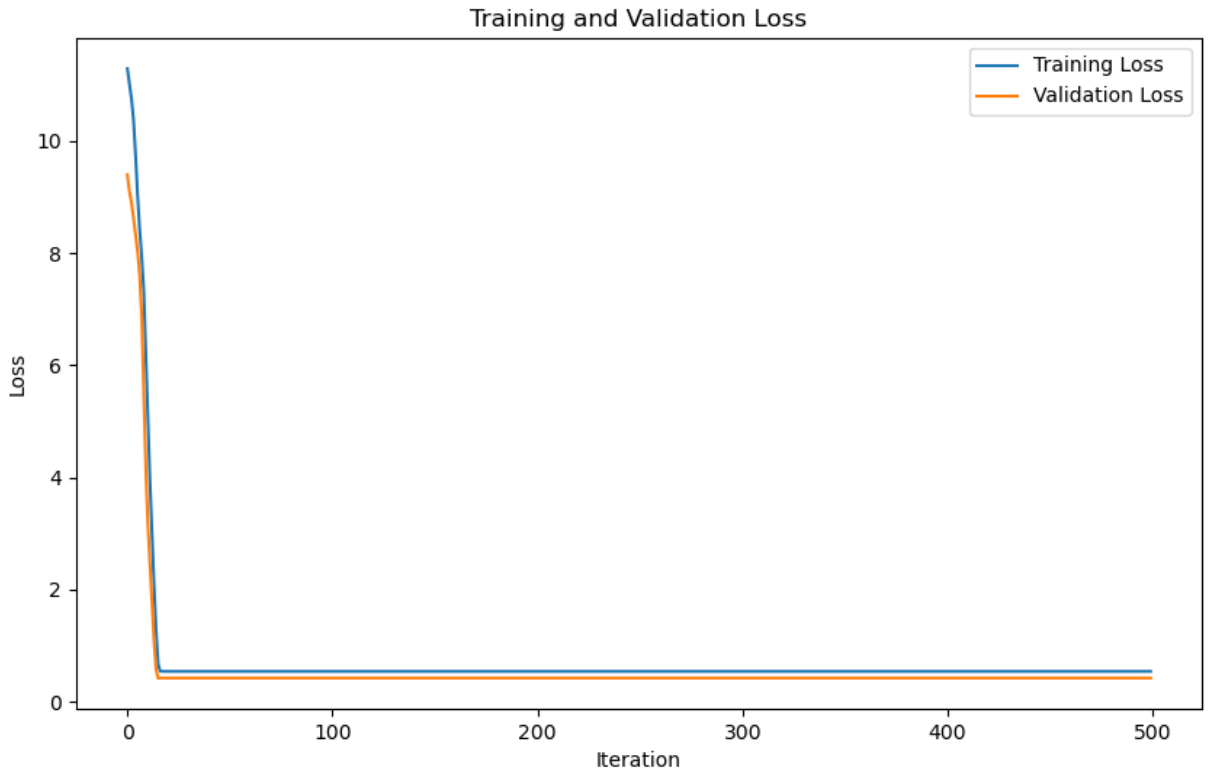
(b) & (c) Learning rate = 1

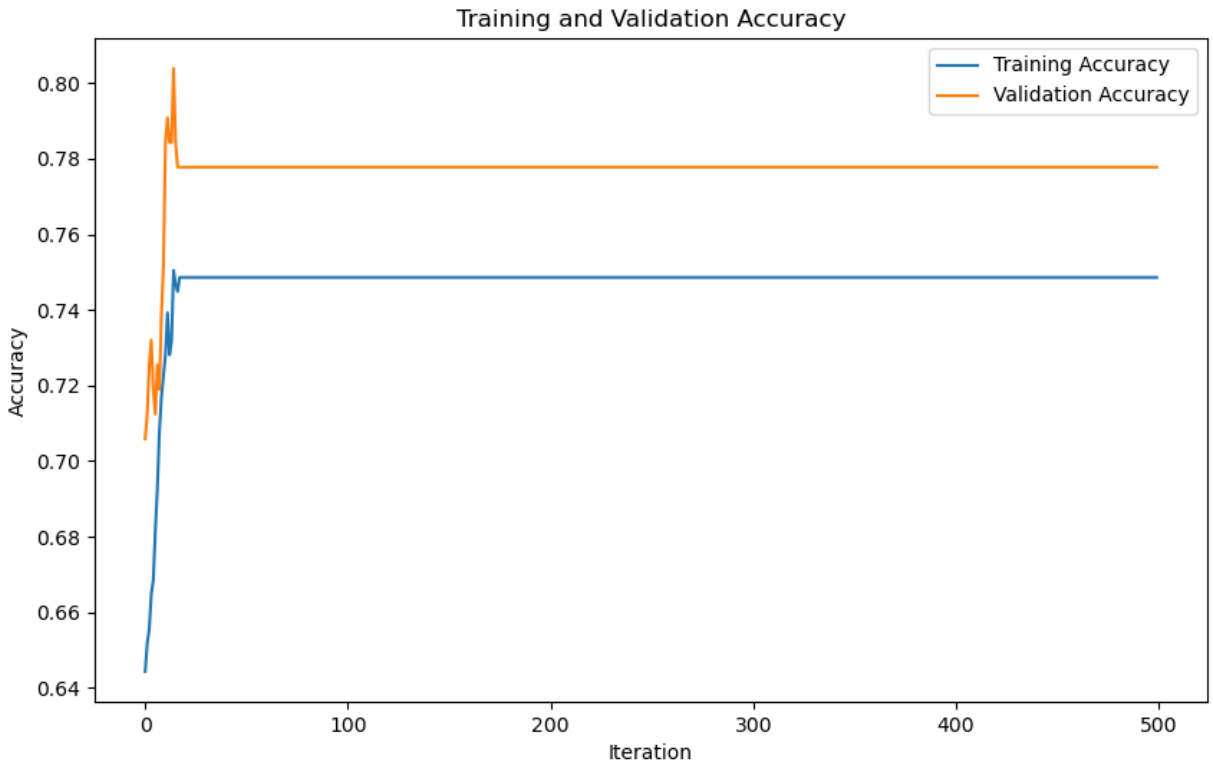

Training and Validation Loss
Training Loss
Validation Loss
Loss
8
6
4
2
0
100
200
300
400
500
Iteration


Training and Validation Accuracy
Training Accuracy
Validation Accuracy
Accuracy
0.78
0.76
0.74
0.72
0.70
0.68
0
100
200
300
400
500
Iteration

Confusion matrix: [[17, 14], [5, 42]]
Accuracy: 0.7564102564102564
Precision: 0.7727272727272727
Recall: 0.5483870967741935
F1 Score: 0.32075471698113206

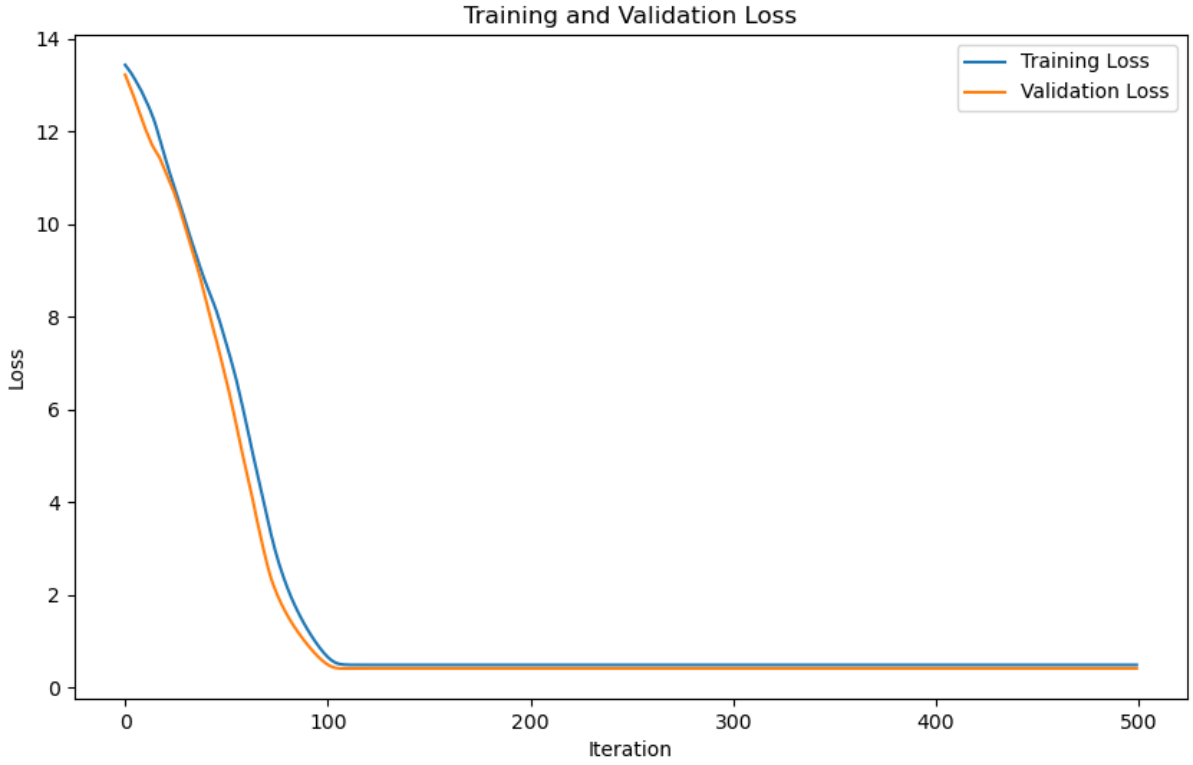
Learning rate = 0.1

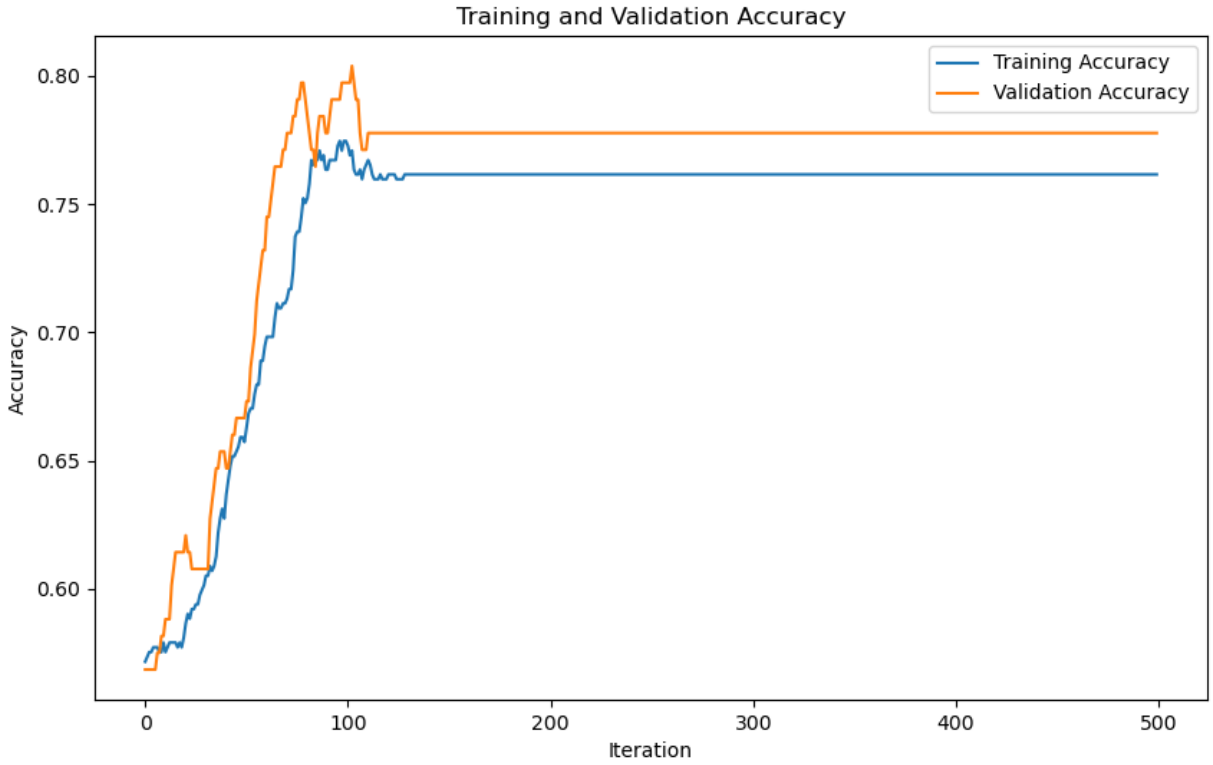




Confusion matrix: [[14, 17], [3, 44]]
Accuracy: 0.7435897435897436
Precision: 0.8235294117647058
Recall: 0.45161290322580644
F1 Score: 0.29166666666666663

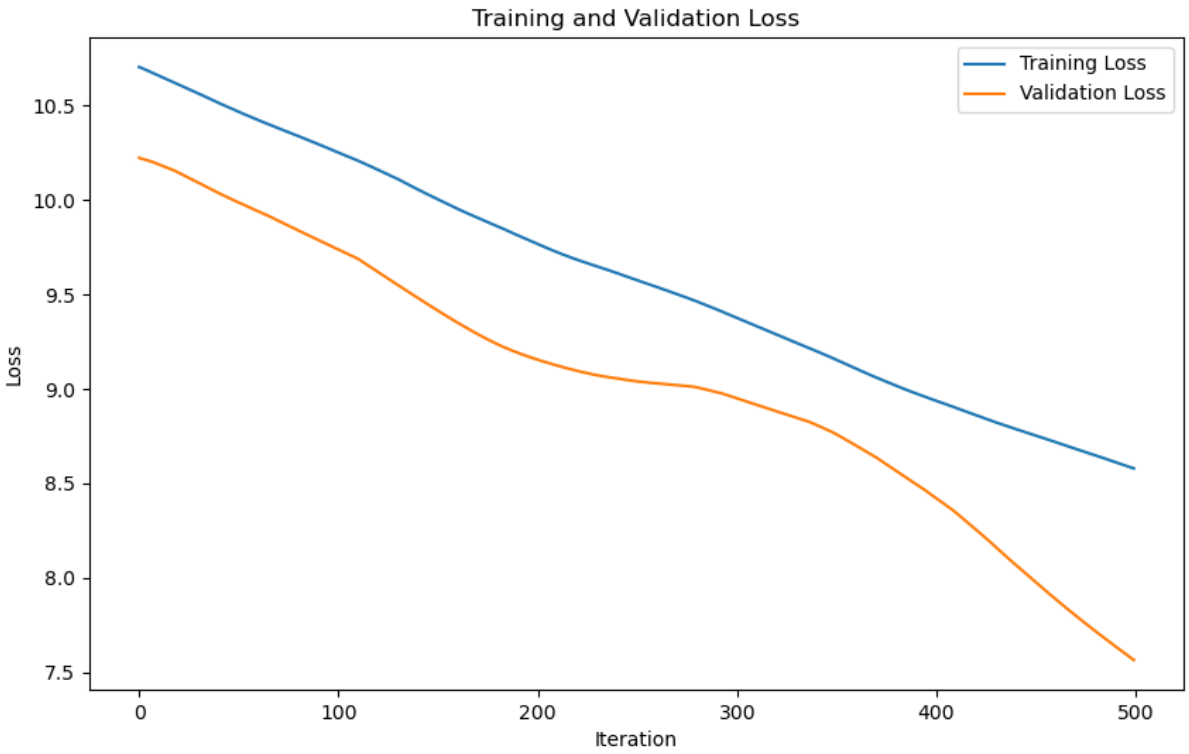
Learning rate = 0.01

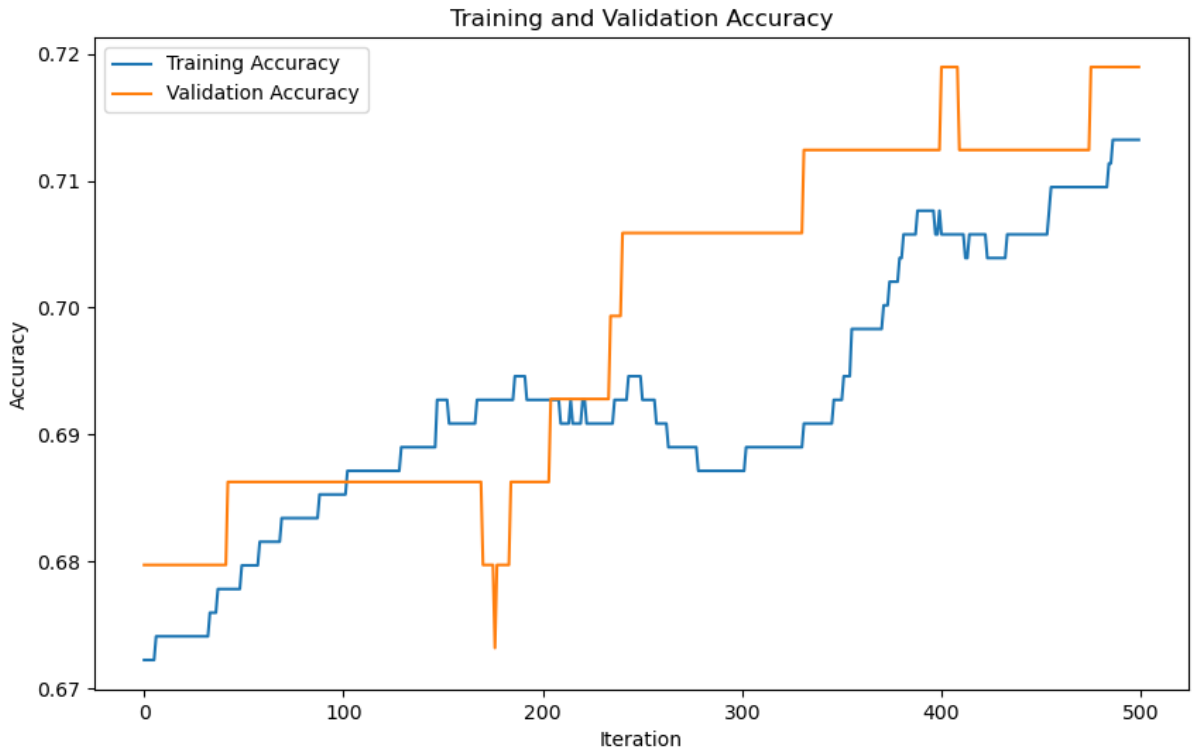

Training and Validation Loss
Training Loss
Validation Loss
Loss
Iteration
0
2
4
6
8
10
12
14
0
100
200
300
400
500


Training and Validation Accuracy
Training Accuracy
Validation Accuracy
Accuracy
Iteration
0.60
0.65
0.70
0.75
0.80
0
100
200
300
400
500

Confusion matrix: [[18, 13], [3, 44]]
Accuracy: 0.7948717948717948
Precision: 0.8571428571428571
Recall: 0.5806451612903226
F1 Score: 0.34615384615384615

Learning rate = 0.001





Confusion matrix: [[24, 7], [18, 29]]
Accuracy: 0.6794871794871795
Precision: 0.5714285714285714
Recall: 0.7741935483870968
F1 Score: 0.3287671232876712

Analysis : As we decrease the learning rate steepness of loss and training graph decreases (graph becomes more smooth for decreasing learning rate) and we attain maximum accuracy and precision at 0.01 learning rate.

(d) Just the modified the loss function by adding a penalty term

New loss functions are:-

Lasso regression:

y_true*log(y_pred) + (1-y_true)*log((1-y_pred)) + λ* ∑ | wj | where λ is L1 regularization parameter.

This is the additional part specific to **Lasso regression**. It adds the absolute values of the coefficients (weights) of the features, multiplied by a regularization parameter (λ), to the loss function.

gradient of this new loss function is (pedictions - y_i)*x_i + λ*(sign(weight))

Ridge regression:

y_true*log(y_pred) + (1-y_true)*log((1-y_pred)) + λ* ∑(wj^2)where λ is L2 regularization parameter.

This is the additional part specific to **Ridge regression**. It adds the sqaured values of the coefficients (weights) of the features, multiplied by a regularization parameter (λ), to the loss function.

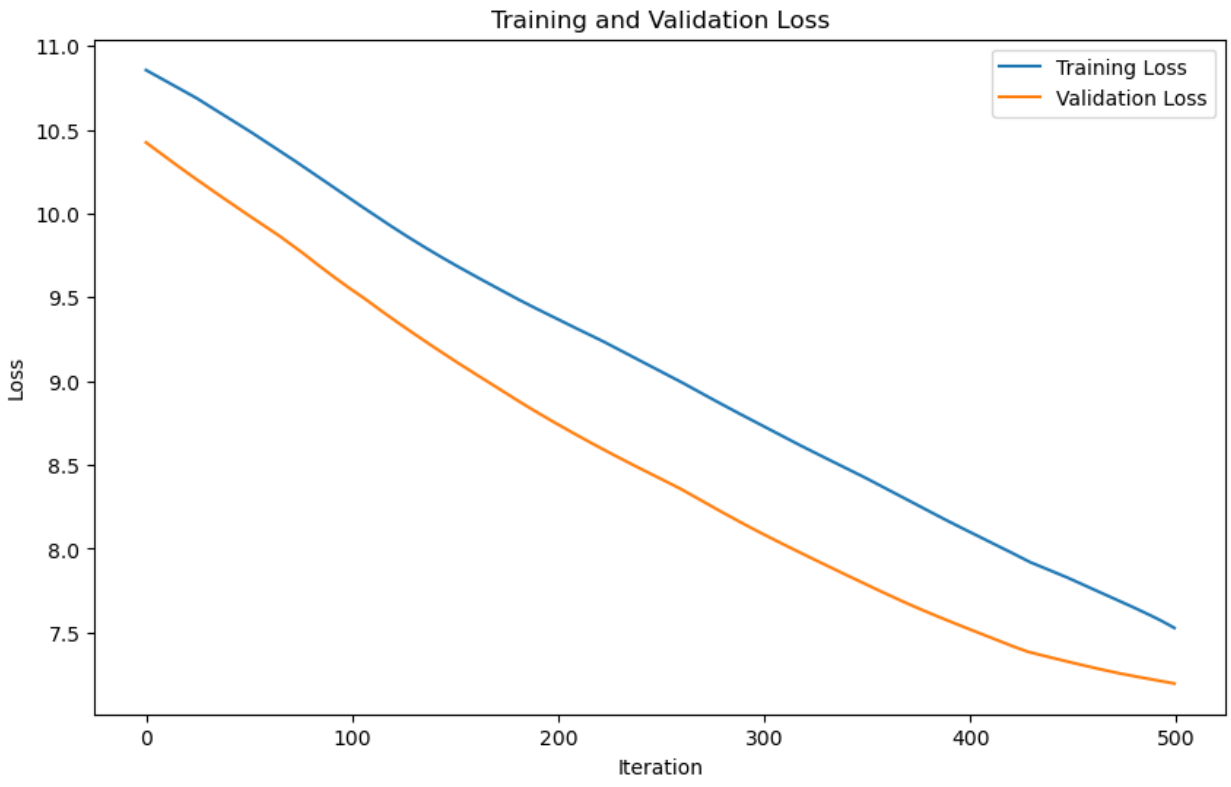
gradient of this new loss function is (pedictions - y_i)*x_i + 2λ*weight

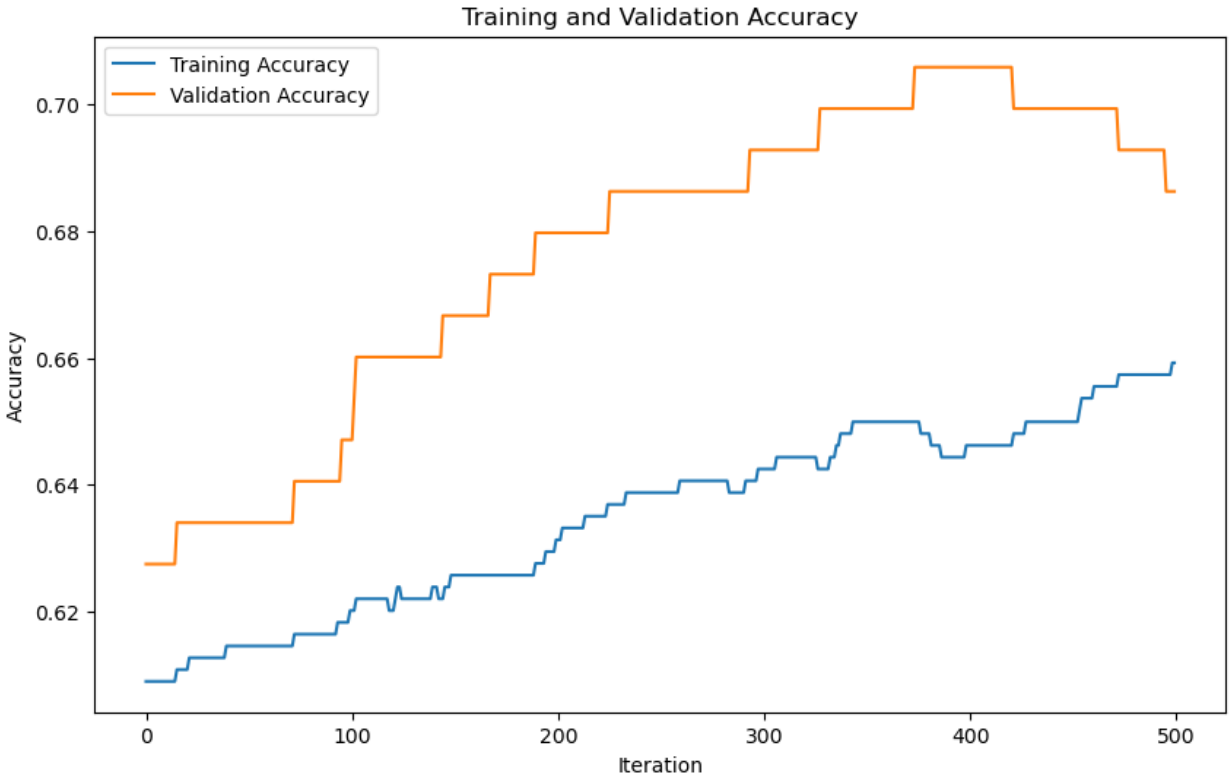
Used (0.001, 0.01, 0.1, 1, 10) values for penalty term in both Lasso and Ridge regression and got

Best l1_penalty: 0.1

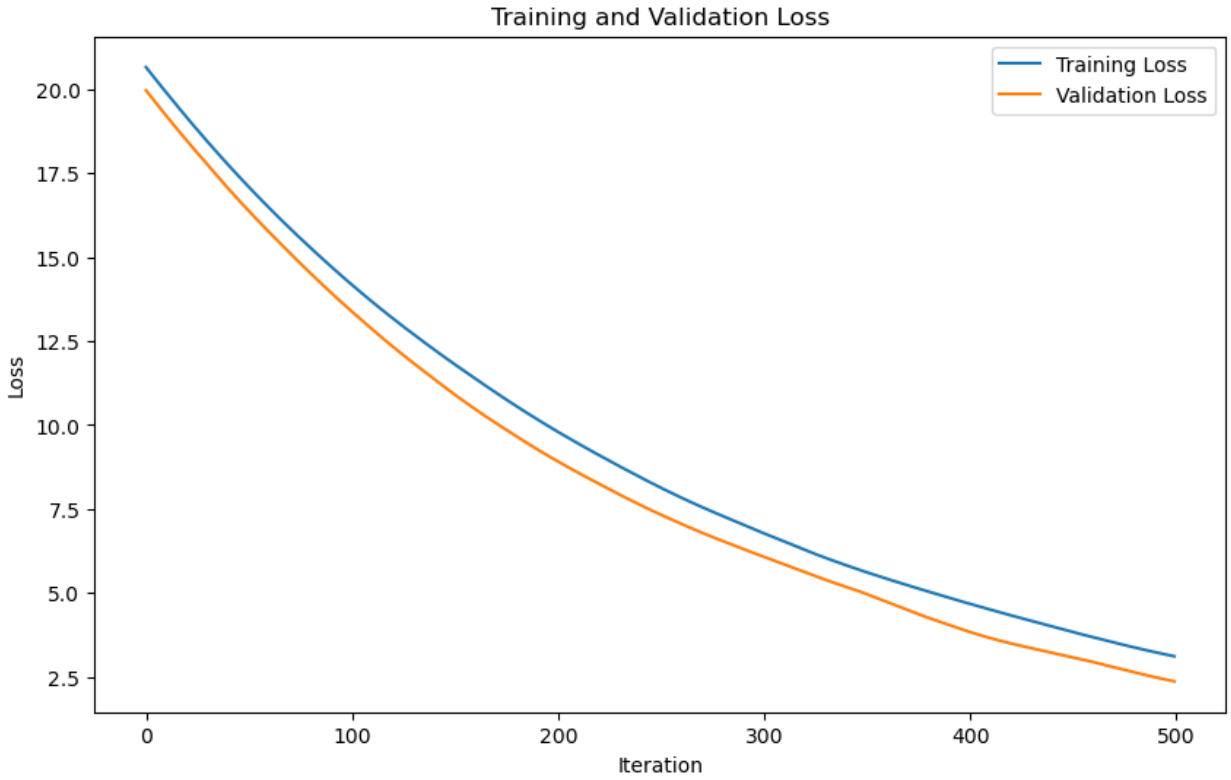
Best l2_penalty: 0.001

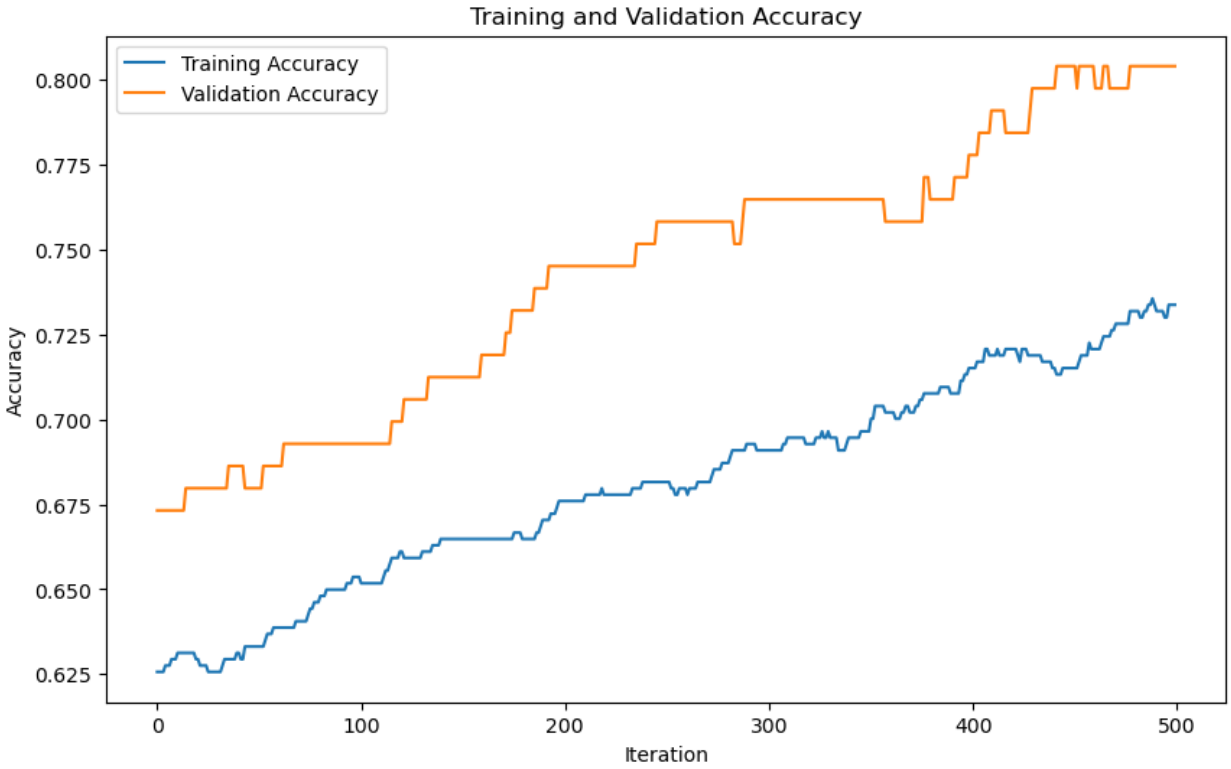
(Lasso regression for best L1 penalty):





(Ridge regression for Best L2 penalty):





(e) tanh(x) = (e^x - e^(-x)) / (e^x + e^(-x))

range of tanh(x) is [-1,1] so we adjust the range with a new function f(x) = (1+tanh(x))/2 to [0,1] and use the same loss function

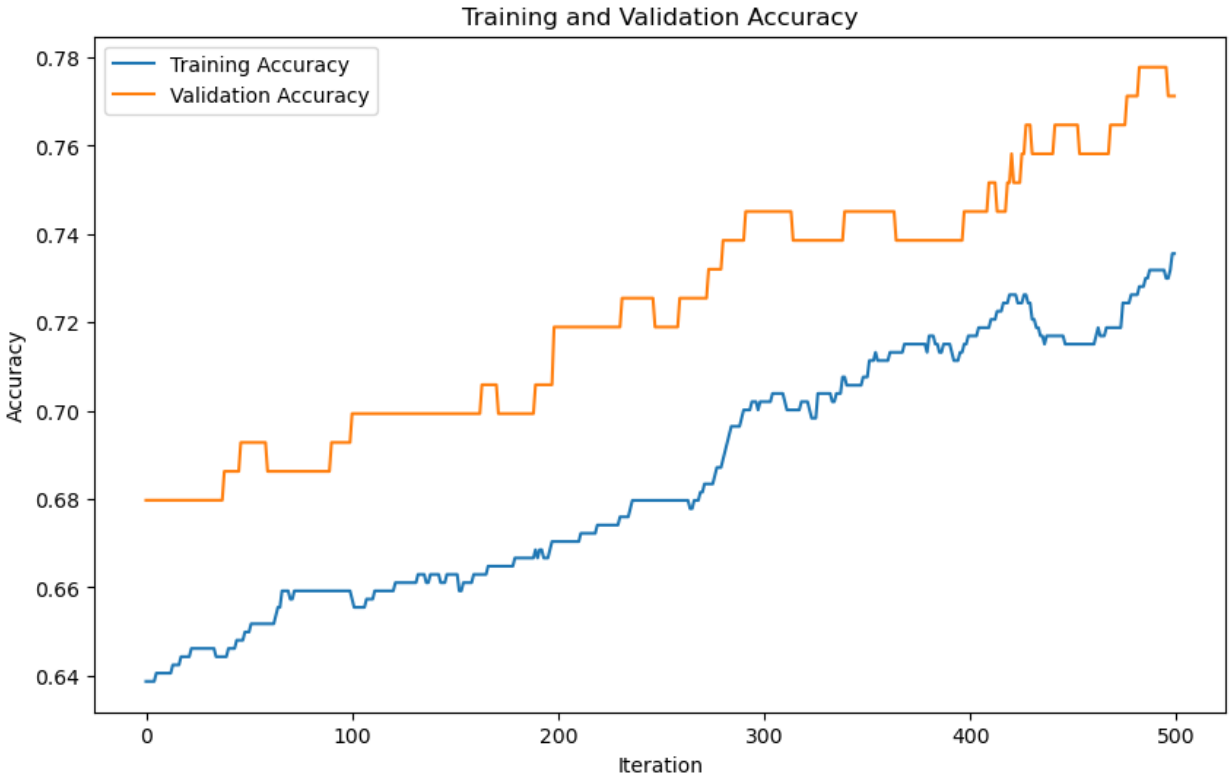
f(x) is simplified to be 1/(1+e^-2x)

loss = y*log(f(x))+(1-y)*log(1-f(x))

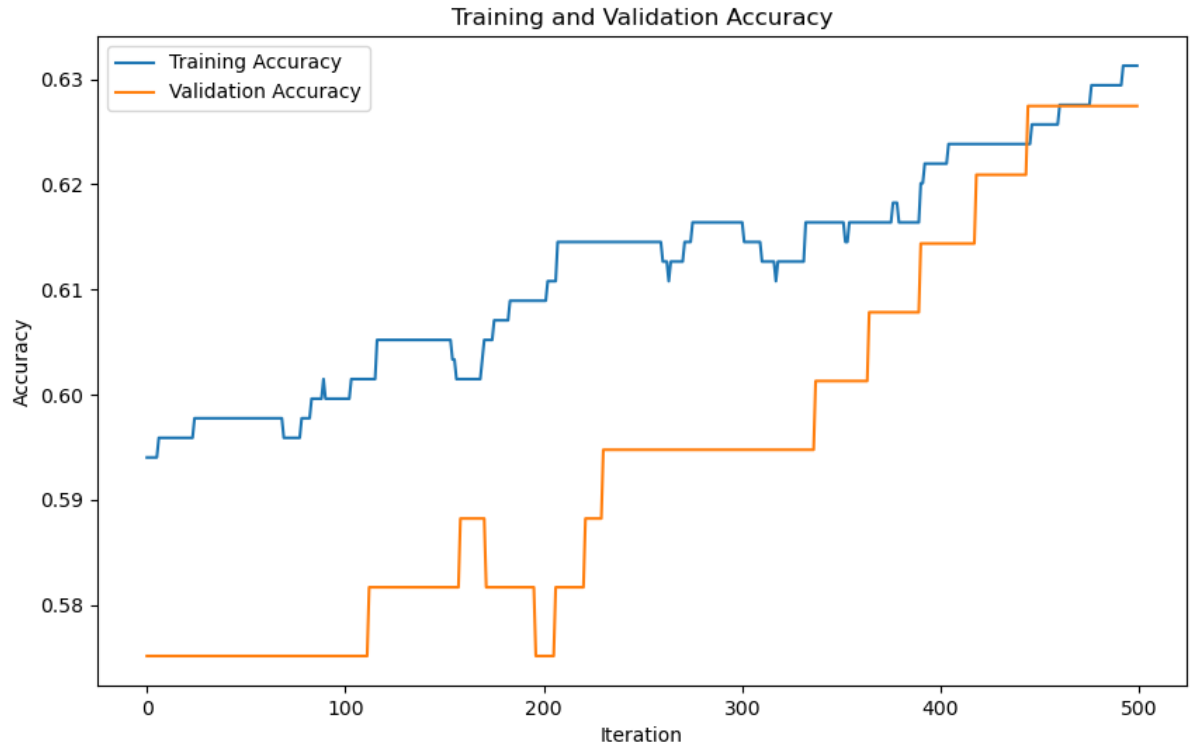
gradient = 2*(f(x) - yi) * xi

Smoothest graph for learning rate = 0.001

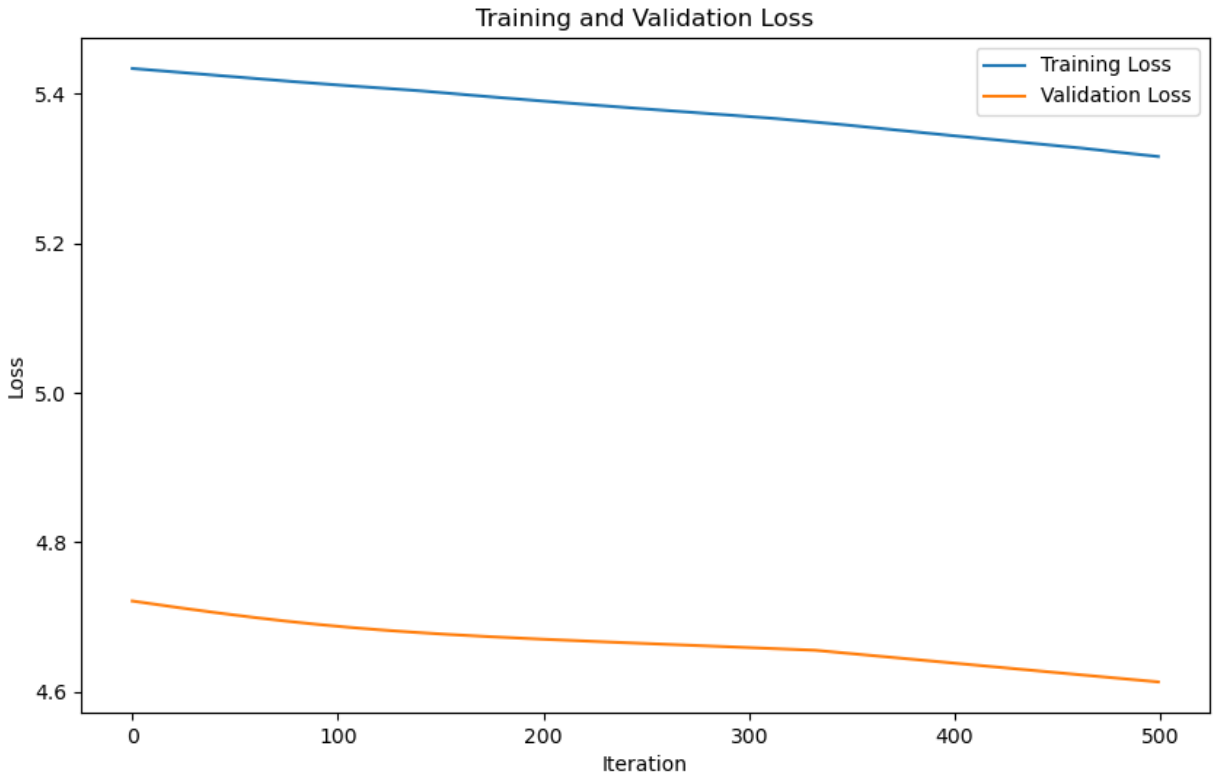
There is not much difference between the performance of model for using tan hyperbolic instead of sigmoid

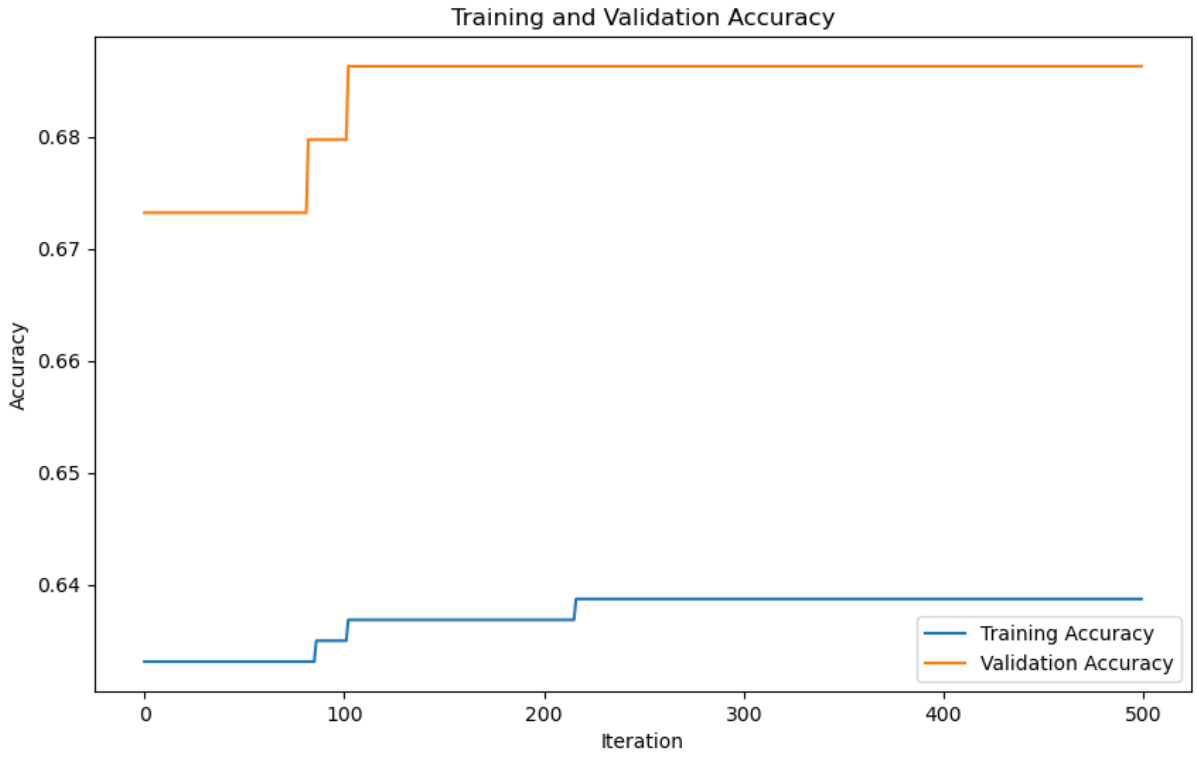


(f) Batch size = 2

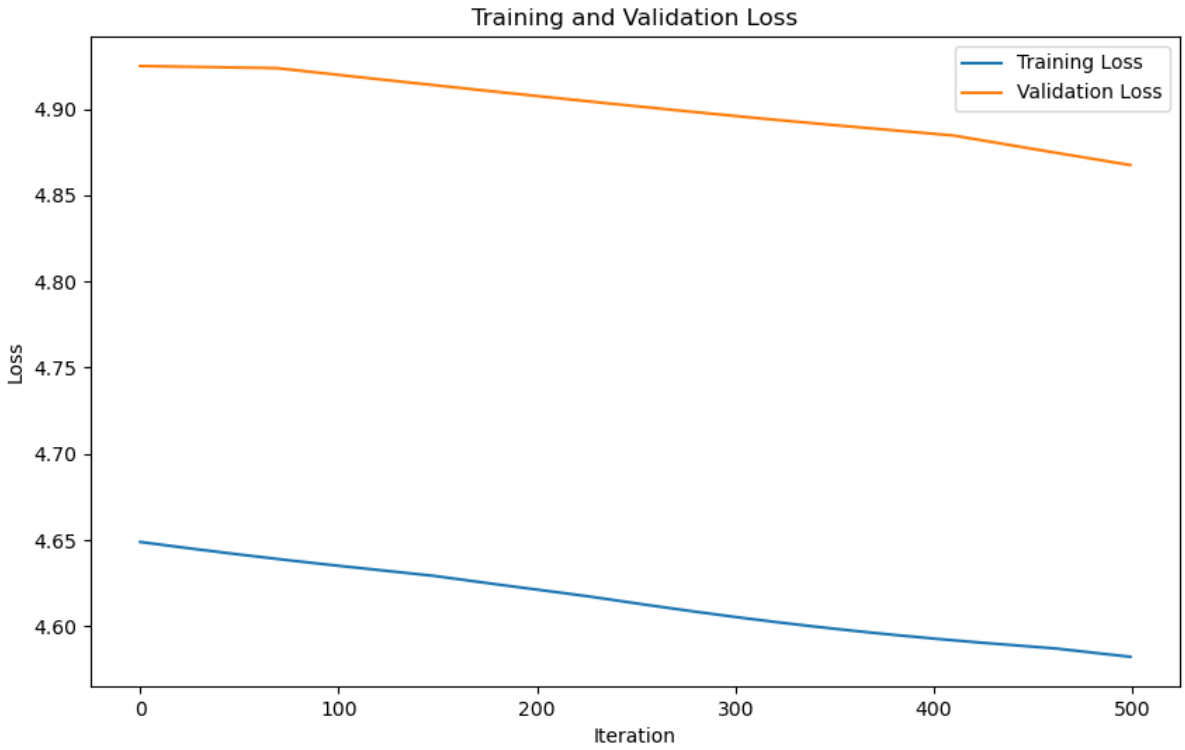

Training and Validation Accuracy
Training Accuracy
Validation Accuracy
Accuracy
0.63
0.62
0.61
0.60
0.59
0.58
0
100
200
300
400
500
Iteration

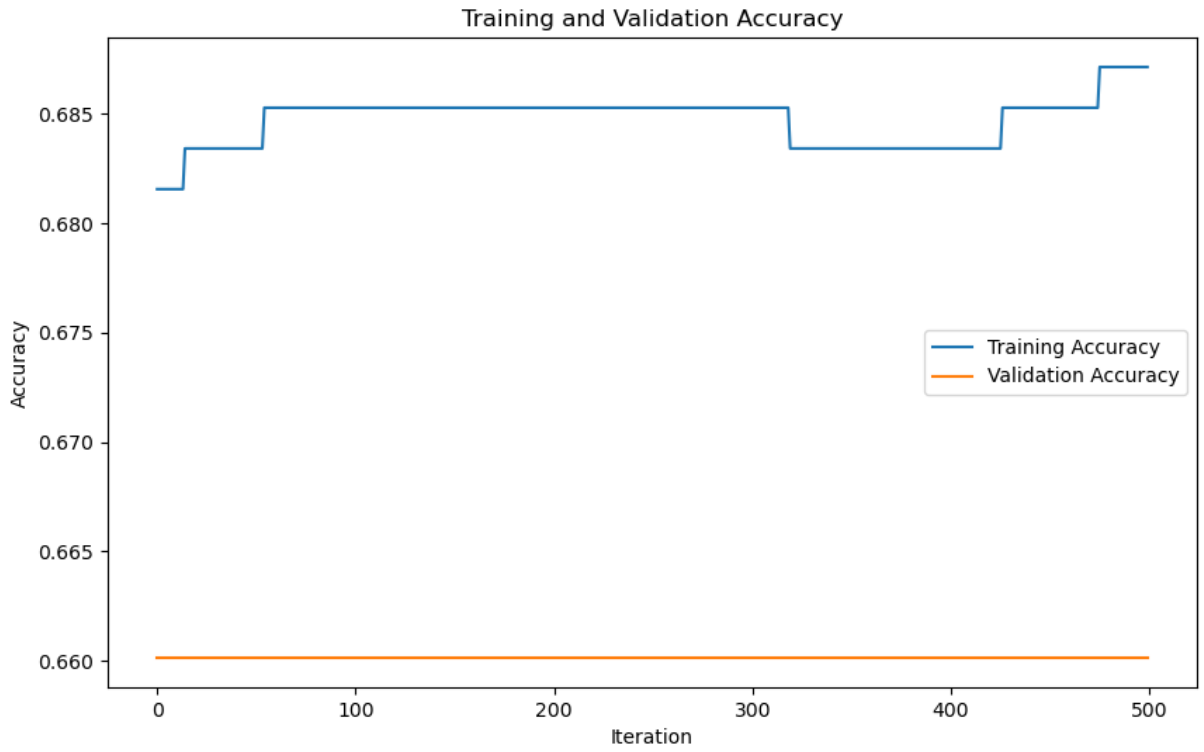
Batch size = 5


Training and Validation Loss
Training Loss
Validation Loss
Loss
5.4
5.2
5.0
4.8
4.6
0
100
200
300
400
500
Iteration

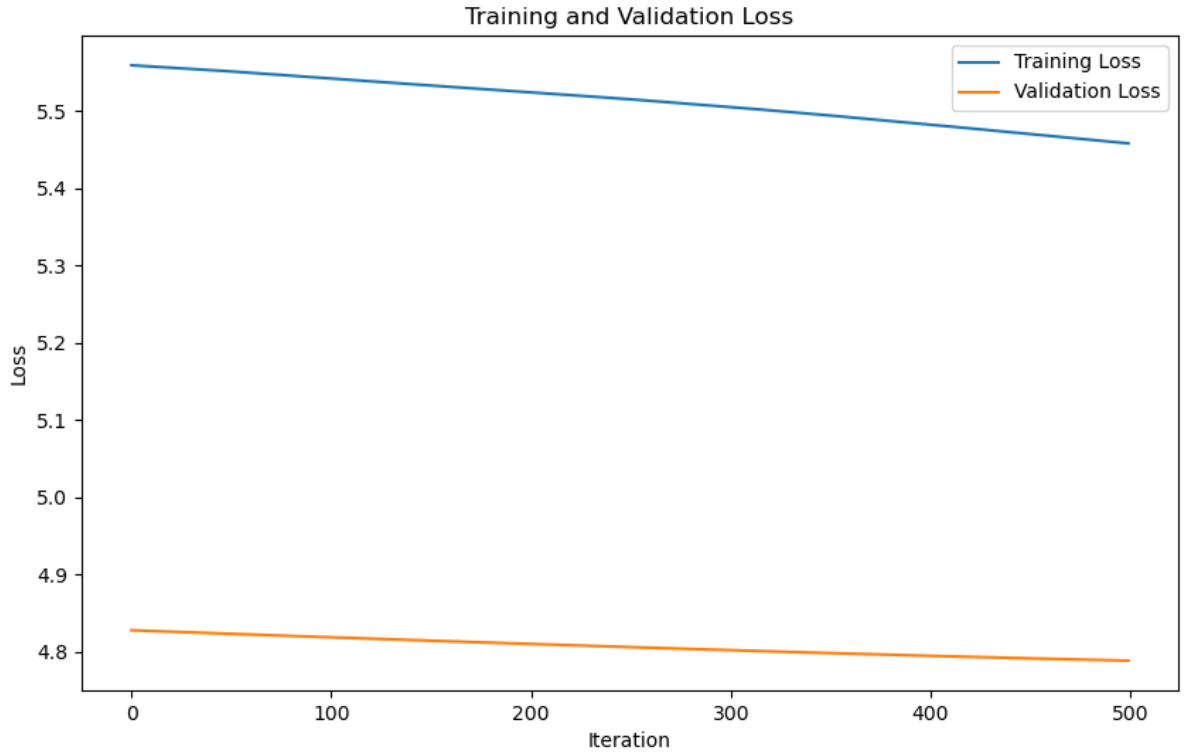

Training and Validation Accuracy
Accuracy
0.68
0.67
0.66
0.65
0.64
0
100
200
300
400
500
Iteration
Training Accuracy
Validation Accuracy

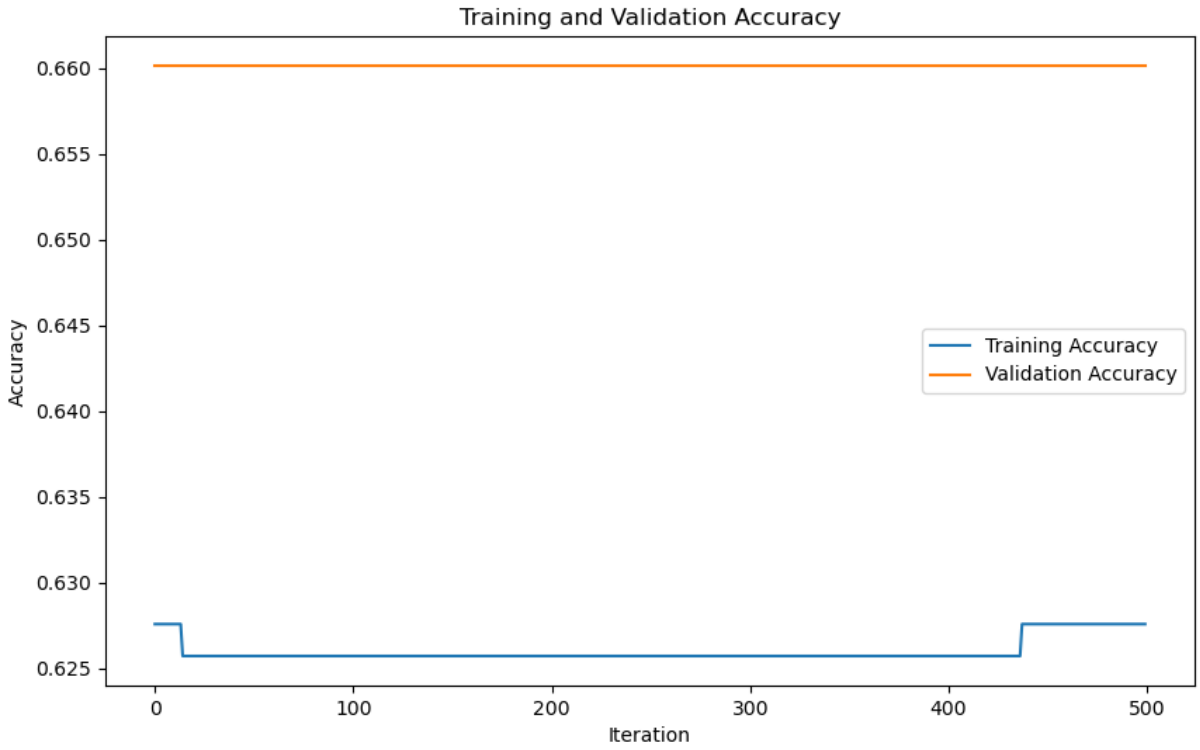
Batch size = 8


Training and Validation Loss
Loss
4.90
4.85
4.80
4.75
4.70
4.65
4.60
0
100
200
300
400
500
Iteration
Training Loss
Validation Loss



Batch size = 15





As we increase the Batch size loss decreases more slower and acuuracies increase more slower

In Normal SGD batch size = 1 so loss decreases more rapidly and accuracy increases more rapidly hence model converges earlier and for any other batch size as we increase the Batch size loss decreases more slower and acuuracies increase more slower so model will not converge too earlier as compared to SGD.

SECTION - C

(a) Insights:

As engine size increases CO2 emission increases

Fuel consumption comb (mpg) between 20 and 30 are in majority

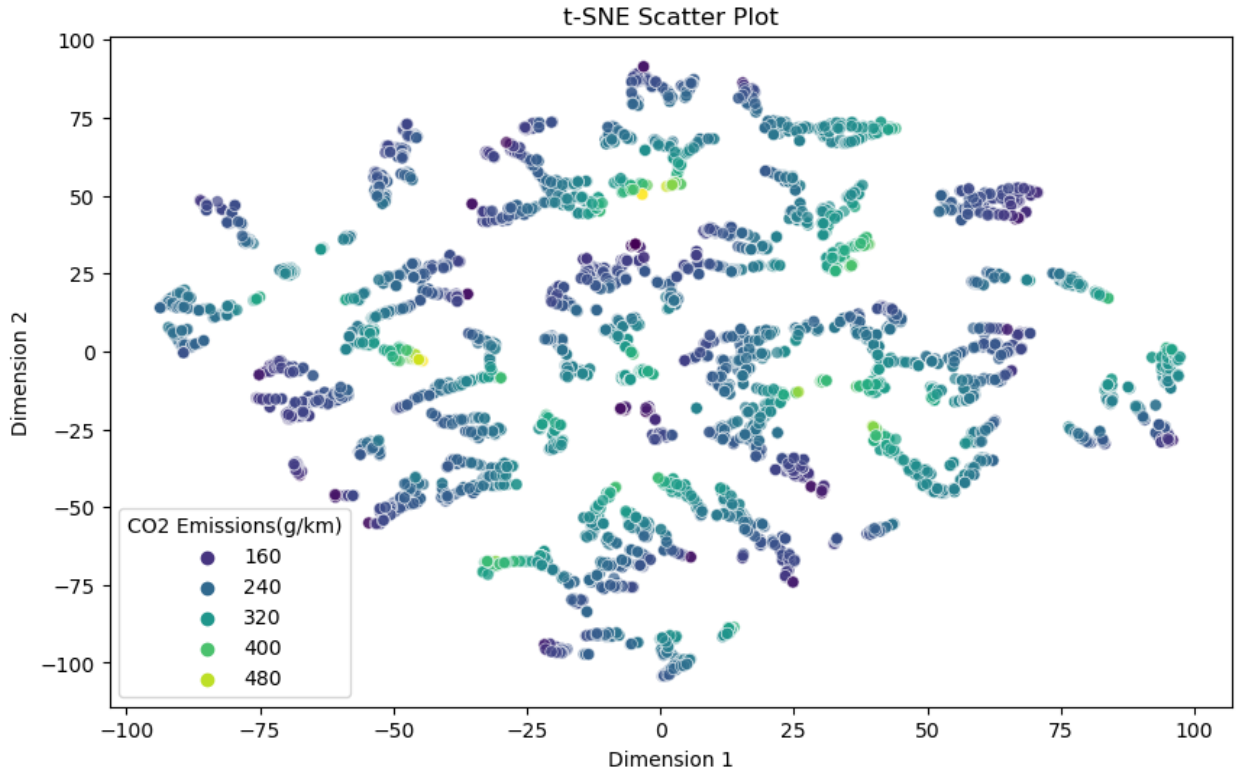
As fuel consumption comb (mpg) increases CO2 emission decreases

As the number of cylinders increase Fuel consumption comb (mpg) decreases

As Fuel Consumption comb (L/100 km) increases Fuel Consumption comb (mpg) decrease

Fuel type X are in majority and Fuel type E has highest median

(b)



From above t-SNE Scatter plot data is separable in some regions overall data is not well separable

(c)

```
Train MSE: 285.9853836055968
Test MSE: 295.30466951241624
Train RMSE: 16.911102377006557
Test RMSE: 17.18443102091007
Train R2 Score: 0.9163430991931094
Test R2 Score: 0.9141463205390631
Train Adjusted R2 Score: 0.916187022885633 9
Test Adjusted R2 Score: 0.9139861457639494
Train MAE: 10.971129887345157
Test MAE: 11.17663002420695
```

(d)

```
Number of Components: 4
Train MSE: 303.4559546146532
Test MSE: 313.064422797522
Train RMSE: 17.419987216259752
Test RMSE: 17.693626615183277
Train R2 Score: 0.9112325798808358
Test R2 Score: 0.9089830423275728
Train Adjusted R2 Score: 0.9111724291641703
Test Adjusted R2 Score: 0.9089213672757873
Train MAE: 11.722022714084135
Test MAE: 11.965185384189502


Number of Components: 6
Train MSE: 300.1203446854911
Test MSE: 306.3858495184792
Train RMSE: 17.323981779183764
Test RMSE: 17.503880984469678
Train R2 Score: 0.9122083178205035
Test R2 Score: 0.9109246983484615
Train Adjusted R2 Score: 0.912119053273295
Test Adjusted R2 Score: 0.9108341286467314
Train MAE: 11.72850858131008
Test MAE: 11.878453089957913
```

```
Number of Components: 8
Train MSE: 287.1135412586759
Test MSE: 297.34925313090844
Train RMSE: 16.944425079024544
Test RMSE: 17.243817823524708
Train R2 Score: 0.9160130887160414
Test R2 Score: 0.9135519004545349
Train Adjusted R2 Score: 0.9158991888533067
Test Adjusted R2 Score: 0.9134346628216542
Train MAE: 11.002770843690577
Test MAE: 11.21135411820483


Number of Components: 10
Train MSE: 286.01030022777957
Test MSE: 295.4046891682688
Train RMSE: 16.911839055164272
Test RMSE: 17.18734095688652
Train R2 Score: 0.9163358105430248
Test R2 Score: 0.914117241908212
Train Adjusted R2 Score: 0.9161939346918174
Test Adjusted R2 Score: 0.9139716038582004
Train MAE: 10.973075319792828
Test MAE: 11.179863441017714
```

As the number of components inceases Errors parameters decrease and R2 score increases and ence model's preformance is increasing

(e) After one hot encoding (it creates different coloumns for categorical features)

```
Train MSE: 8.634243031859752
Test MSE: 2.5344147437695 97e+26
Train RMSE: 2.938408247990696
Test RMSE: 15919845300032.273
Train R2 Score: 0.9974742974492186
Test R2 Score: -7.36828277696837 96e+22
Train Adjusted R2 Score: 0.9960352577816992
Test Adjusted R2 Score: -1.1566422100332772e+23
Train MAE: 1.9348854540268192
Test MAE: 2917398566123.729
```

As we can observe there is a large difference between train error and test error(exceptionally high) and also Test R2 score is negative which indicates **high variance** and **low bias** in the data so this implies overfitting of model due to increase number of features.

In part **(c)** difference between train and testing error is very small and R2 score is close to 1 and hence overall performance of model is much better than **(e)**

(f)

	Num Components	Train MSE	Test MSE	Train RMSE	Test RMSE	\
0	5	514.000446	538.615923	22.671578	23.208100	
1	10	465.766274	480.839380	21.581619	21.928050	
2	15	413.007743	433.462886	20.322592	20.819772	
3	20	382.671074	411.508088	19.561980	20.285662	
4	25	379.391077	404.166927	19.477964	20.103903	

	Train R2 Score	Test R2 Score	Train Adjusted R2 Score	\
0	0.849644	0.843409	0.849516	
1	0.863753	0.860206	0.863522	
2	0.879186	0.873980	0.878879	
3	0.888060	0.880363	0.887680	
4	0.889020	0.882497	0.888548	

	Test Adjusted R2 Score	Train MAE	Test MAE
0	0.843276	15.852366	16.298178
1	0.859969	14.799502	15.015568
2	0.873659	14.016162	14.292725
3	0.879956	13.689455	14.250973
4	0.881997	13.873465	14.428815

As the number of component in PCA increases Train and Test MSE,RMSE, decreases and Test R2 score increase.

(g)

```
Linear Regression:
MSE: 295.30466951241624
RMSE: 17.18443102091007
R2 Score: 0.9141463205390631
Adjusted R2 Score: 0.9139861457639494
MAE: 11.17663002420695

Lasso Regression:
MSE: 298.90385412136084
RMSE: 17.288836112398105
R2 Score: 0.9130999326094469
Adjusted R2 Score: 0.9129378056180466
MAE: 11.262986730989738

Ridge Regression:
MSE: 295.3554571783758
RMSE: 17.18590868061319
R2 Score: 0.914131555083437
Adjusted R2 Score: 0.9139713527608315
MAE: 11.177797141308643
```

MSE, RMSE and MAE for Lasso is greater than RIdge, While R2_score, Adjusted_R2_score for Ridge is greater than Lasso

(h)

```
Train MSE: 286.5833524030331
Test MSE: 296.5163484951347
Train RMSE: 16.928772914864002
Test RMSE: 17.219650068893234
Train R2 Score: 0.9161681804062047
Test R2 Score: 0.9137940501223316
Train Adjusted R2 Score: 0.9160117777577088
Test Adjusted R2 Score: 0.9136332181262912
Train MAE: 11.08228646281294
Test MAE: 11.30118540429089
```

Performance is almost same as (c)