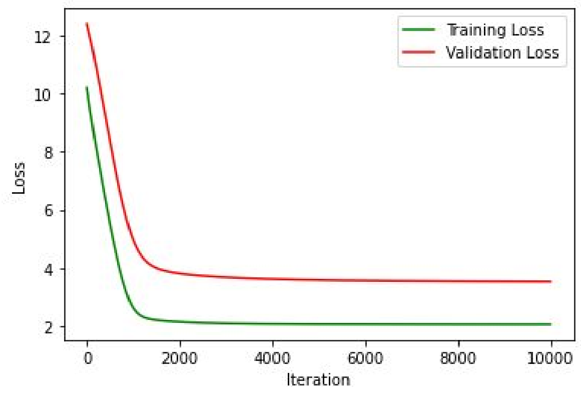
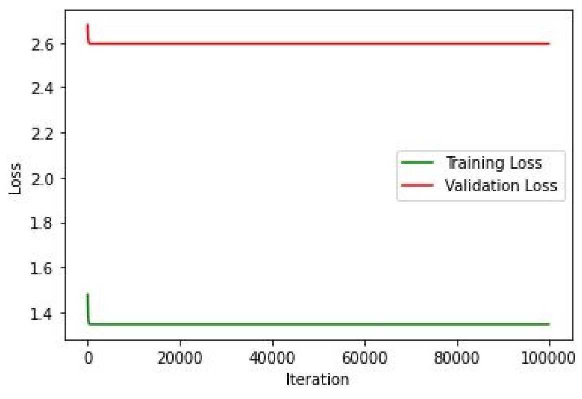
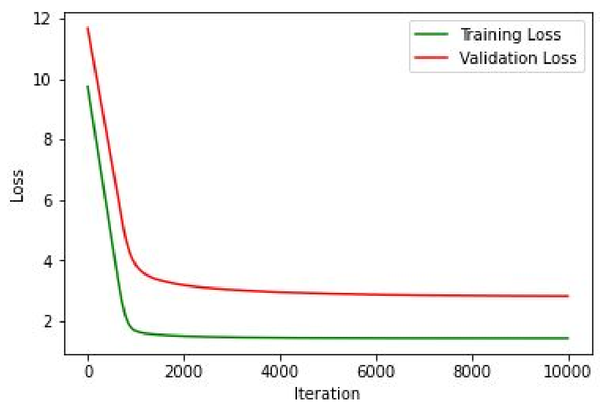
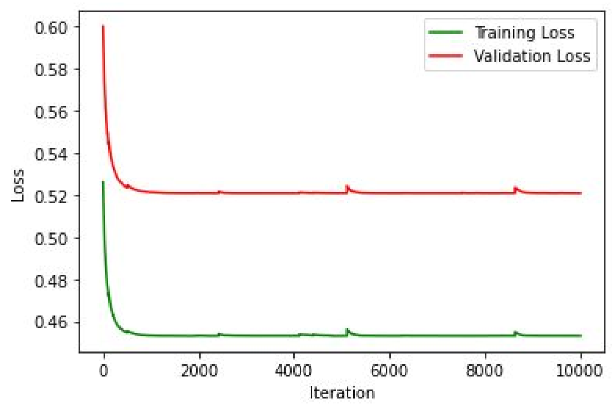
Assignment 1

1. I chose the value of *K* to be 10 in K-Fold cross-validation as this is an optimal value irrespective of the size of the dataset. A higher value of K will result in a smaller Testing set and a larger Training set which is not the desired case to train the model on.
   Pre-Processing strategy:
   - Dataset 1:
     - Rows shuffled
     - Male is mapped to 1, Female is mapped to 2, and infant is mapped to 3
     - Input Data normalized i.e mean 0 and standard deviation 1
   - Dataset 2:
     - Rows shuffled
     - Filled NA values in 'Critic_Score' with the Mean value of the column
     - Replaced 'tbd' values in the 'User_Score' columns with NA values
     - Converted all the string values to float values
     - Filled NA values in 'User_Score' with the Mean of the column
     - Input Data normalized i.e mean 0 and standard deviation 1

a.

Dataset-1	Dataset-2
With RMSE LOSS function:	
	
With MAE LOSS function:	
	

b. For dataset 1:

- Best RMSE value:
    - Training: 2.0486255251475742
    - Validation: 3.524723068996929

- Best MAE value:
    - Training: 1.4326572163305726
    - Validation: 2.818967895495819

For dataset 2:

- Best RMSE value:
    - Training: 1.387095520453951
    - Validation: 2.3936591149224147
- Best MAE value:
    - Training: 0.44689430360711124
    - Validation: 0.5764138583769058

Fold Number 7 gives the most optimal result in K fold cross-validation

c. Since the RMSE value lies in the range [(MSE),(MSE) $\sqrt{n}$] that's why in both the datasets MAE value is lesser than the RMSE value. Output values of the Abalone dataset range from 0 to 29 i.e the very small range to predict therefore MAE is a better choice because errors don't need to be penalized that much. Conversely, in the Video game dataset, the range of output values is quite large, and we need to avoid getting large penalties therefore MAE is more preferred than RMSE

d. MAE is mean absolute error, it measures the average magnitude of error in prediction. It sums up all the errors i.e takes error in absolute value and gives its average.

$$\mathrm{MAE} = \frac{1}{n}\sum_{j=1}^{n}|y_j - \hat{y}_j|$$

RMSE is the root mean squared error, as the name suggests it calculates the root of the mean of squared errors i.e all the errors are first squared and then averaged over data and then returns its square root value.

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)^2}$$

Both of the above loss functions are negatively-oriented i.e their low values signify a good model. Both metrics range from 0 to infinity and don't get affected by the direction of error. A greater difference in the above two signifies greater variance in the individual errors in the sample.
If errors are of the same magnitude then both the loss function give identical values and RMSE should be preferred because it is convex function which is differentiable at all points.

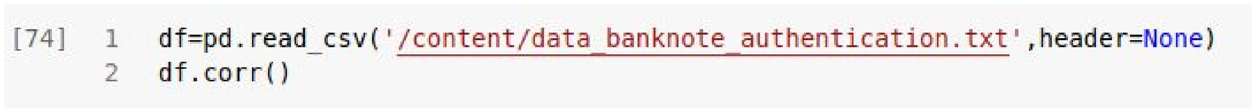
e. Optimal parameters using Normal equation is

β_0	β_1	β_2	β_3	β_4	β_5	β_6	β_7	β_9
9.9336844	-0.321553	-0.09923279	1.18716678	0.46859482	4.44739825	-4.46228469	-1.11301053	1.21107218

Training loss using the above parameters: 1.5409430714877588
Validation loss using the above parameters: 1.932138018475601

2.

Exploratory Data Analysis:

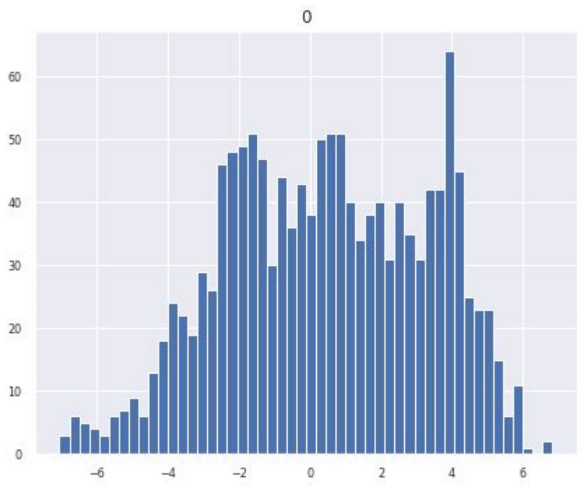
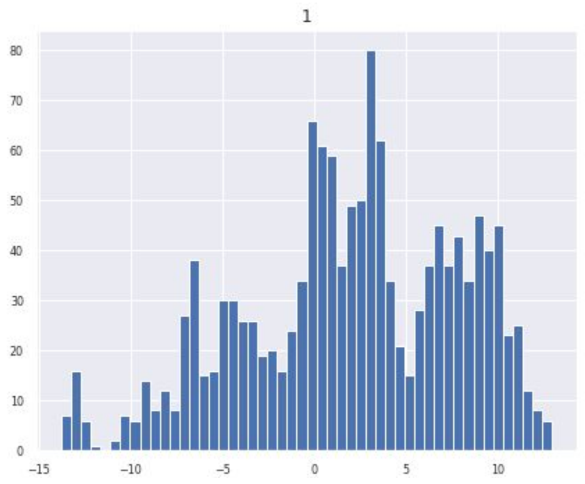
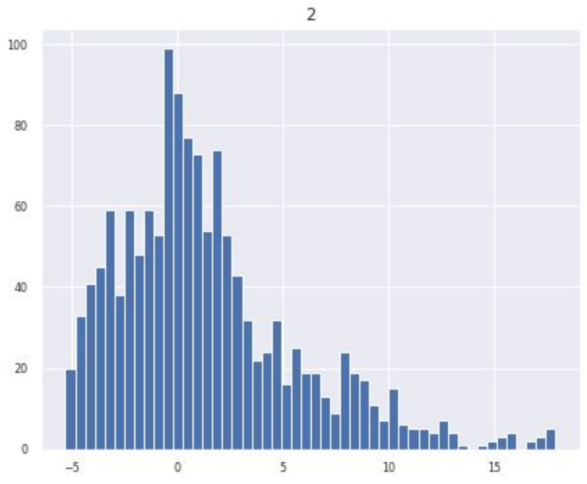
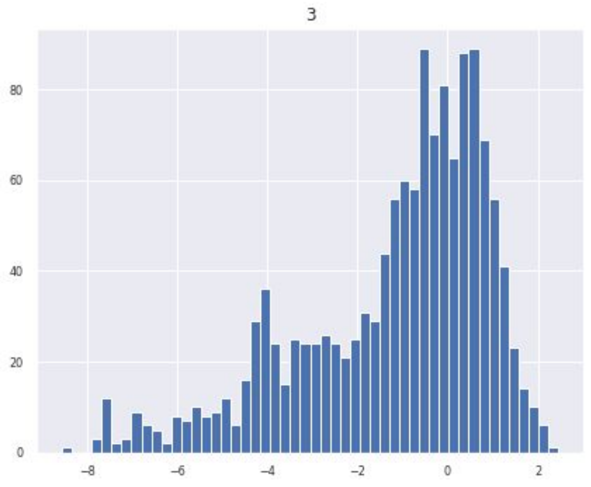


```
[74]  df=pd.read_csv('/content/data_banknote_authentication.txt',header=None)
      df.corr()
```

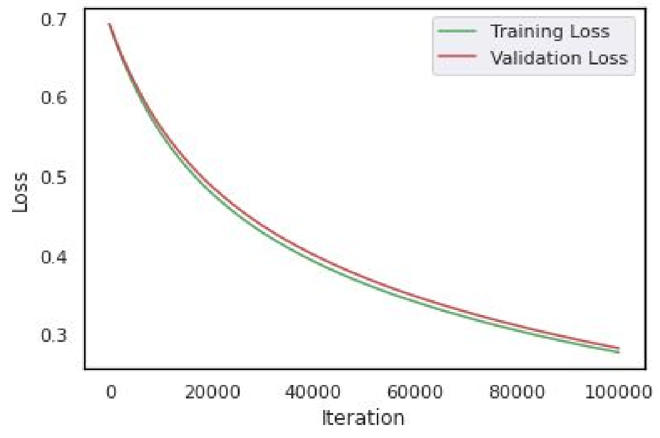
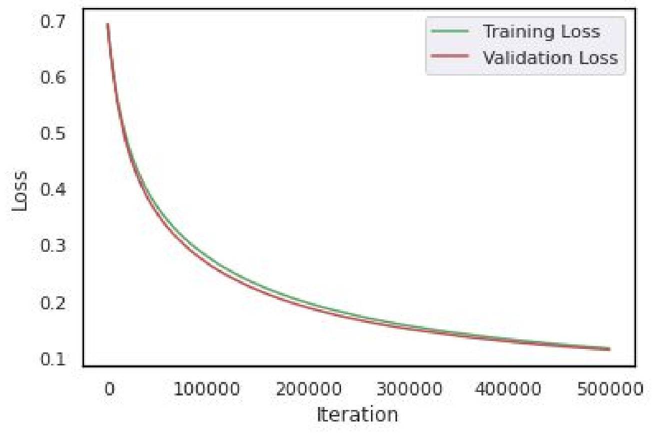
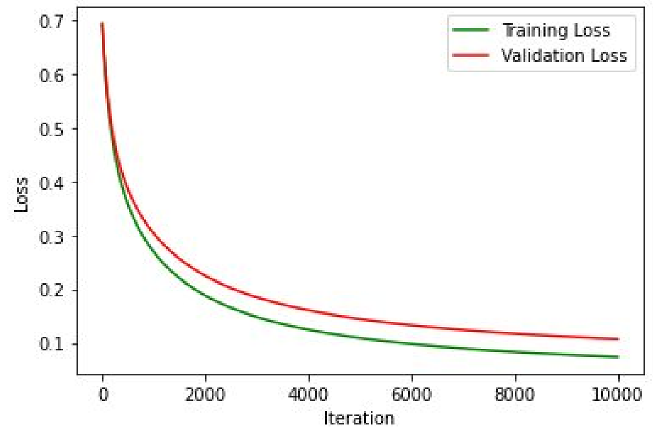
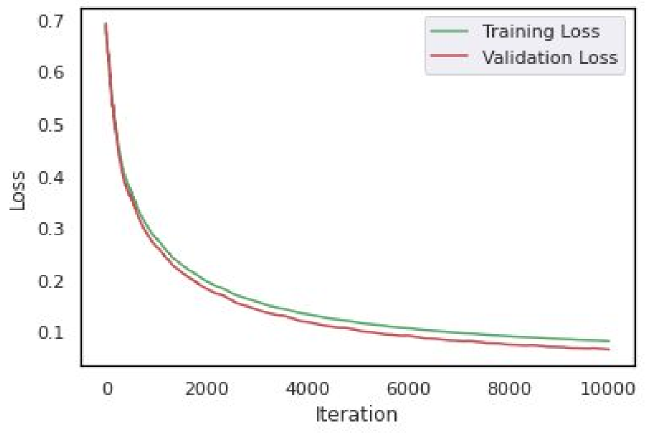
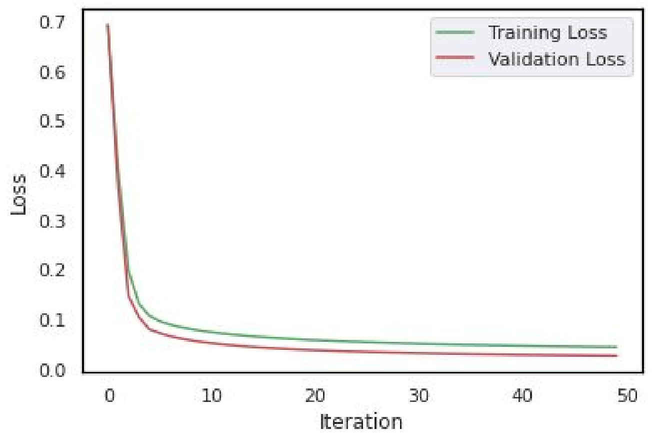
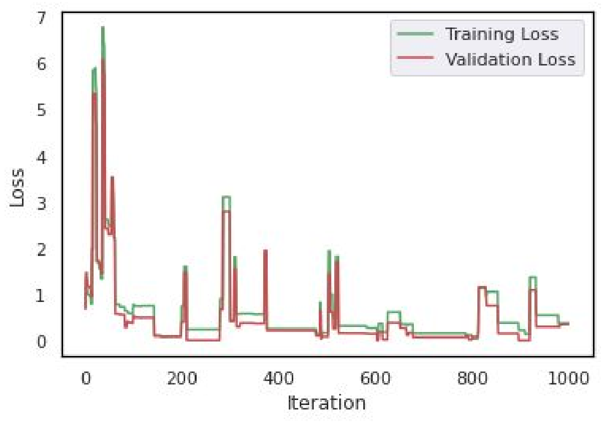
Correlation Matrix					
	0	**1**	**2**	**3**	**4**
0	1	0.264026	-0.380850	0.276817	-0.724843
1	0.264026	1	-0.786895	-0.526321	-0.444688
2	-0.380850	-0.786895	1	0.318841	0.155883
3	0.276817	-0.526321	0.318841	1	-0.023424
4	-0.724843	-0.444688	0.155883	-0.023424	1

- The positive value of correlation resembles that as one value increases the other value also increases, similarly negative correlation value resembles that if one value increases then other value decreases
- Higher the value the more correlated features
- Correlation of 1 resemble that both the variables have a perfect positive relationship
- Correlation of 0 resembles that neither of the variables depends on each other.

```python
df=pd.read_csv('/content/data_banknote_authentication.txt',header=None)
df.hist(figsize=(16, 20), bins=50, xlabelsize=8, ylabelsize=8);
```

Histogram Plots	
	
Data is almost Normally distributed around the Mean	Data is almost Normally distributed around the Mean
	
Data is almost Normally distributed but skewed to Left of Mean	Data is almost Normally distributed but skewed to Right of Mean

a) For BGD at learning rate 0.01 and 10000 epoch, the Training set achieved an accuracy of 99.12 % and the test set achieved 99.74 % accuracy
For SGD at learning rate 0.01 and 10000 epoch, the Training set achieved an accuracy of 97.65 % and the test set achieved 99.27 % accuracy

BGD	***SGD***
α=0.0001	
Epochs Taken to Converge: >10^5	*Epochs Taken to Converge:* 5x10^5
	
α=0.01	
Epochs Taken to Converge: 10^4	*Epochs Taken to Converge:* 10^4
	
α=10	
Epochs Taken to Converge: 50	*Epochs Taken to Converge:* NA
	

Epochs taken to converge is relatively lower than SGD	Epochs taken to converge is relatively Higher than BGD
It eventually converges even for higher learning rate like 10	It fails to converge for high learning rates, due to overshooting of the gradient for a single sample

Using Sklearn's parameters:

Accuracy Test: 97.96 %

Accuracy Train: 98.25 %

Thetas: -4.58128443 -4.68888037 -4.27457391 0.23536925

3.

Mean squared error is defined as follows,f(x) = $\frac{-1}{m} * \sum_{i=1}^{m} (y-y_{hat})^2$

In logistic regression, y_{hat}= $1/(1+e^{-\Theta X})$

Gradient is defined as derivative of the loss function,

$$\mathbf{g(x)=}\ \frac{df}{d\Theta} \mathbf{=} \frac{df}{dy_{hat}} . \frac{dy_{hat}}{d\Theta}$$

$$\mathbf{= -2(y-y_{hat})\ .\ y_{hat}\ .\ (1-y_{hat})X} \qquad \text{-------------} \ \mathbf{(i)}$$

The above gradient function has 3 values in it, $(y-y_{hat})$, y_{hat} & $(1-y_{hat})$.

And because of these 3 values, our gradient will always tend to 0, because of the following reasons

- If its predicting opposite values than original then, $(y-y_{hat})$ -> 0
- If its predicting value close to 0 then, y_{hat} ->0
- If its predicting value close to 1 then, $(1-y_{hat})$ ->0

I.e in any case gradient will tend to 0

Conversely in the case of Cross entropy loss:

It loss function defined as follows,f(x)= $\frac{-1}{m} * \{y*Log(y_{hat})+(1-y)*Log(1-y_{hat})\}$

In logistic regression, y_{hat}= $1/(1+e^{-\Theta X})$

Gradient is defined as derivative of the loss function,

$$\mathbf{g(x)=}\ \frac{df}{d\Theta} \mathbf{=} \frac{df}{dy_{hat}} . \frac{dy_{hat}}{d\Theta}$$

$$= -\left(\frac{y}{y_{hat}} - \frac{1-y}{1-y_{hat}}\right) .(-1).(1+e^{-\Theta X})^{-2}.\ e^{-\Theta X}.X$$

$$= -\left(\frac{y}{y_{hat}} - \frac{1-y}{1-y_{hat}}\right) .y_{hat}.(1-y_{hat}).X$$

$$= - \{y(1-y_{hat}) -(1-y)y_{hat}\}.X$$

$$\mathbf{= - (y-y_{hat}).X} \qquad \text{-------------} \ \mathbf{(ii)}$$

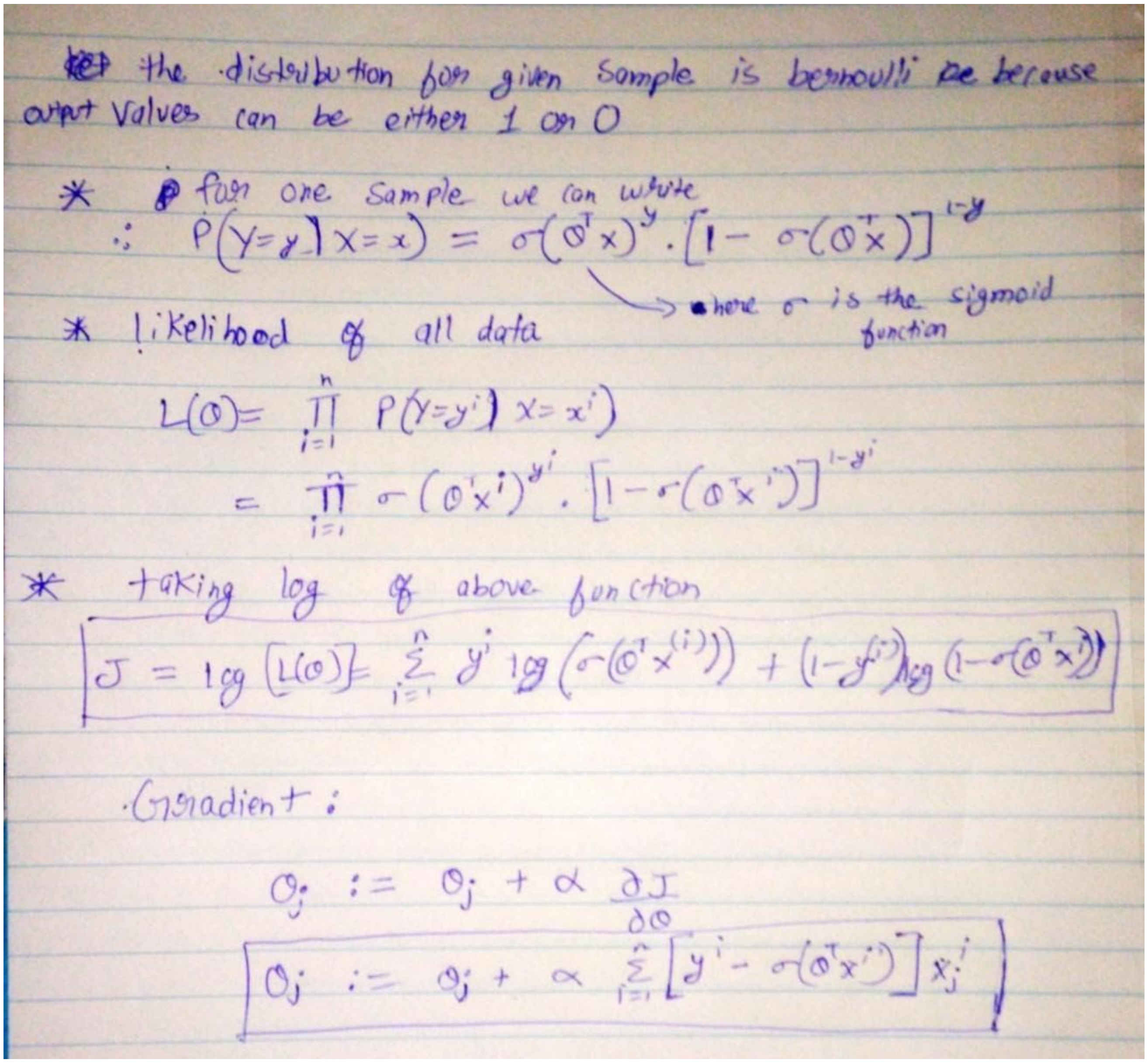
This gradient function is optimal because if the predicted value is opposite to it then it would move in different directions and if both the values are equal then that would result in a 0 gradient and hence the optimal solution is reached only in this case.

Hence from the above interpretation, we can easily say that using mean squared error in logistic regression wouldn't train the model well, and hence its predictions will have very low accuracy.

Moreover, if our model is using MSE loss in the logistic regression model, and if we predict 0 instead of the original value 1 or vice versa then MSE will interpret it as only 1 difference and hence won't penalize it as much as if the loss function was cross entropy which would have interpreted as a very large error and would have penalized the error more.

4.

a)



the distribution for given sample is bernoulli be because output values can be either 1 or 0

* for one sample we can write

$$\therefore P(Y=y \mid X=x) = \sigma(\theta^T x)^y \cdot [1 - \sigma(\theta^T x)]^{1-y}$$

→ here σ is the sigmoid function

* likelihood of all data

$$L(\theta) = \prod_{i=1}^{n} P(Y=y^i \mid X=x^i)$$

$$= \prod_{i=1}^{n} \sigma(\theta^T x^i)^{y^i} \cdot [1 - \sigma(\theta^T x^i)]^{1-y^i}$$

* taking log of above function

$$\boxed{J = \log[L(\theta)] = \sum_{i=1}^{n} y^i \log(\sigma(\theta^T x^{(i)})) + (1-y^i)\log(1-\sigma(\theta^T x^i))}$$

Gradient :

$$\theta_j := \theta_j + \alpha \frac{\partial J}{\partial \theta}$$

$$\boxed{\theta_j := \theta_j + \alpha \sum_{i=1}^{n} [y^i - \sigma(\theta^T x^i)] x_j^i}$$

Running the above gradient descent for 1000 epochs and at learning rate 0.01 we get values of theta as follows -0.35315706, 1.42809304, 0.76577219

b) Fitted response function: log[L(Θ)] <- Log likelihood function

$$= \sum_{i=1}^{m} y^i \log(\sigma(\Theta^T X^i)) + (1-y^i)\log(1-\sigma(\Theta^T X^i))$$

Where σ is the sigmoid function i.e $\sigma(x) = 1/(1+e^{-x})$

c)

$e^{\beta 1}$ =4.05519 means that with a unit increase in age of the child, the odds of the disease recurring (y=1) to the disease not recurring(y=0) increases by 4.055 times.

$e^{\beta 2}$ = 2.13827 means that with a unit increase in the percentage of spread of the disease, the odds of the disease recurring (y=1) to the disease not recurring(y=0) increases by 2.13827 times.

d) probability that a patient with 75% of disease spread and age of 2 years will have a recurrence of disease in next 5 years

$P(X_1=75 \text{ and } X_2=2) = \sigma(\Theta^T X)^y [1-\sigma(\Theta^T X)]^{1-y}$

Where X=[1,75,2] Θ =[-0.35315706, 1.42809304, 0.76577219] and y =1

$P(X_1=75 \text{ and } X_2=2) = 0.999$

5.

Hypothesis: $Y=X\beta + \epsilon$

Let the sum of squared errors be defined as

$$S = \epsilon^2 = (Y - X.\beta)^T . (Y - X.\beta) \qquad \text{-----------} \text{ (i)}$$

Where Y is (n,1) output vector, X is (n,m) feature data matrix, and β is (m,1) matrix of model parameters

Since the above function is a parabolic function in β and since we want to minimize the error in prediction, we need to find the value of β at which this function is minimum.

Using basic calculus we can easily write. $\frac{dS}{d\beta} = 0$

Solving Eqn (i) further gives,

$$S=Y^TY - 2\beta^TX^TY + \beta^TX^TX\beta$$

Therefore,

$$S^{\wedge} = \frac{dS}{d\beta} = 0$$

=> $0 - 2 X^T Y + 2 X^T X \beta = 0$

=> $X^TX\beta = X^T Y$

=> $\boldsymbol{\beta_{OLS} = (X^TX)^{-1} X^T Y}$

This Normal form is not useful in the following conditions:

- Since Matrix multiplication is a costly operation so if the number of samples or number of features is very large then it would take large space and also time to calculate the values.
- If (X^TX) is a non-invertible matrix then we can't use normal eqn to calculate optimal parameters. This mostly happens if there are more features than number of samples or there are some highly correlated features i.e redundant features.
- (X^TX) must be non singular and a positive definite matrix