Assignment-1 Linear Regression & Logistic Regression
Ritik Garg | 2018305

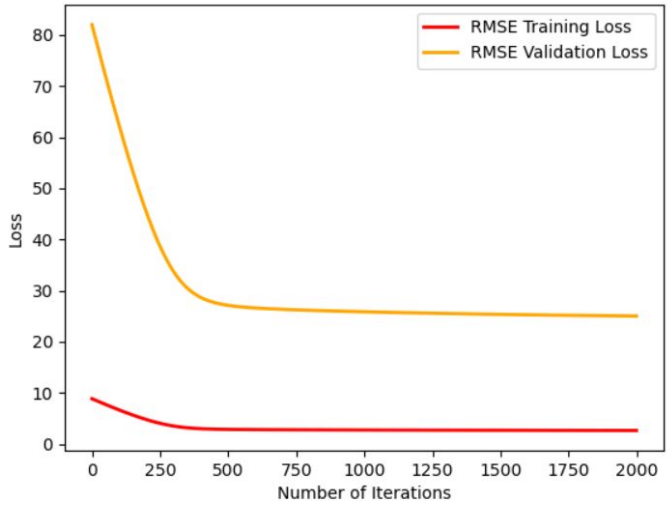
1. **Linear Regression**

K has accepted values of k = 3, 5, 10.
The dataset has length approximately equal to 3000 - 4000 after preprocessing, Hence if we take k = 10, the data in each fold will be very less and it will take more time for processing and it will not lead to better results. Similarly, for k = 3, the data in each fold will be more so, so it will not lead to better results.
I had taken k = 5 for the K fold cross validation because as the value has been shown empirically that at this value of k, test error does not have high bias and high variance.
After that, I passed the dataset for preprocessing.
Preprocessing:

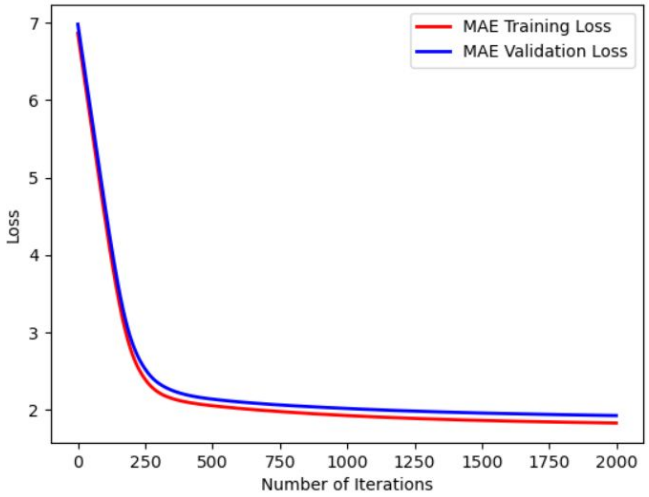
- In the dataset 0 :
    - To avoid an unfortunate split, I firstly shuffled the dataset.
    - The dataset does not have any Nan value, so no need to remove any value. So, simply read the data from the folder and pass the columns from 1 to 4 to X (features) and last column to y(output)
- In the dataset 1:
    - To avoid an unfortunate split, I firstly shuffled the dataset.
    - The dataset contains Nan values, so we need to drop them from both the features and the output. Use .dropna() function for the same.
    - After that we need to normalise the data to get better results.
    - Also the data contains outliers, so we need to remove the outliers, because here they are not needed
- In the dataset 2:
    - To avoid an unfortunate split, I firstly shuffled the dataset.
    - We don't need to use the notna() function.
    - We can normalise the data

Implementing Gradient Descent using
(1) RMSE
(2) MAE

(a). On dataset 0:
RMSE loss

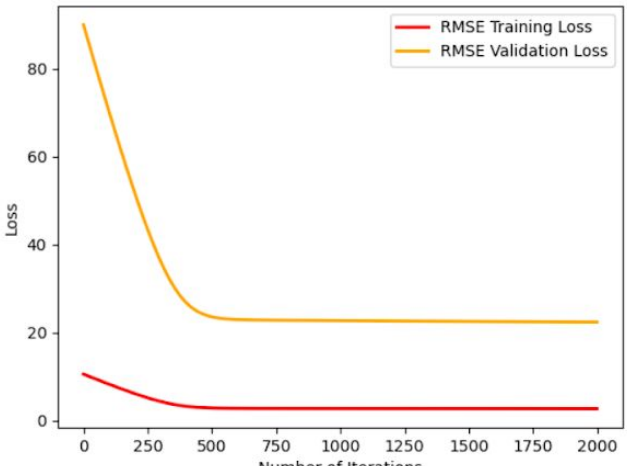


MAE loss:

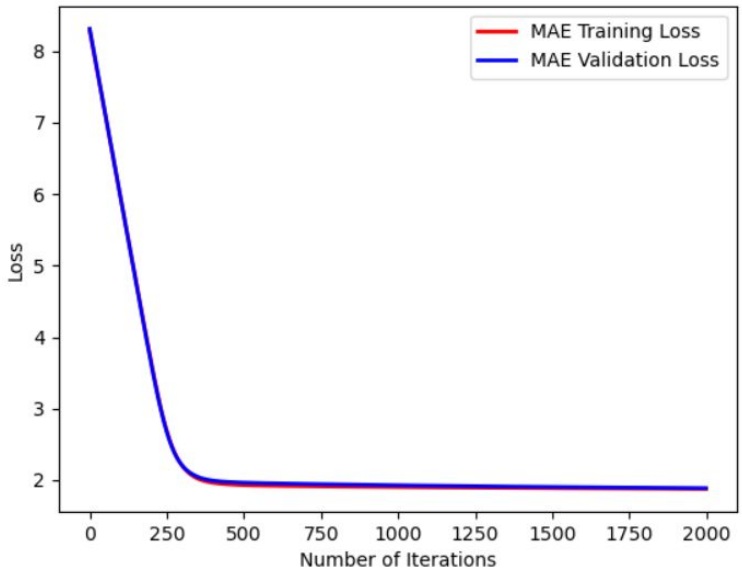


On Dataset 1:

RMSE loss:



MAE loss:



(b). On dataset 0:

Best RMSE value was achieved on **fold 4: 13.145174385123992**

Best MAE value was achieved on **fold 4**: **1.8298042549874394**

On Dataset 1:

Best RMSE value was achieved in **fold 2**: **4.849061309**

Best MAE value was achieved in **fold 3**: **0.13513076432500187**

(c). On comparing the values achieved from both RMSE and MAE loss functions , We can see that MAE is working better and leads to better performance on both the datasets.

It must be there as in RMSE, we take the sum of square of the errors and then take the square root of the value, while in MAE we take the sum of Absolute difference of the errors. As the error values are greater than 1, as result the error value in case RMSE increases very fastly as compared to that in case of MAE error.

Also RMSE penalises the outliers more as compared to the MAE loss.

(d). RMSE error is always greater than or equal to the MAE error (which we can See here as well). When the difference between them is greater then the Variance between the individual errors is greater.

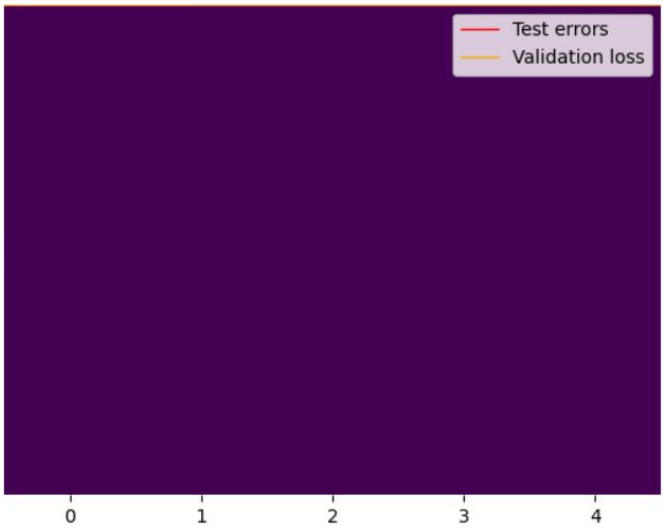
They are equal if the errors are of the same magnitude.

In that case, I will prefer the RMSE error because RMSe error will be penalising the errors more as compared to the MAE error

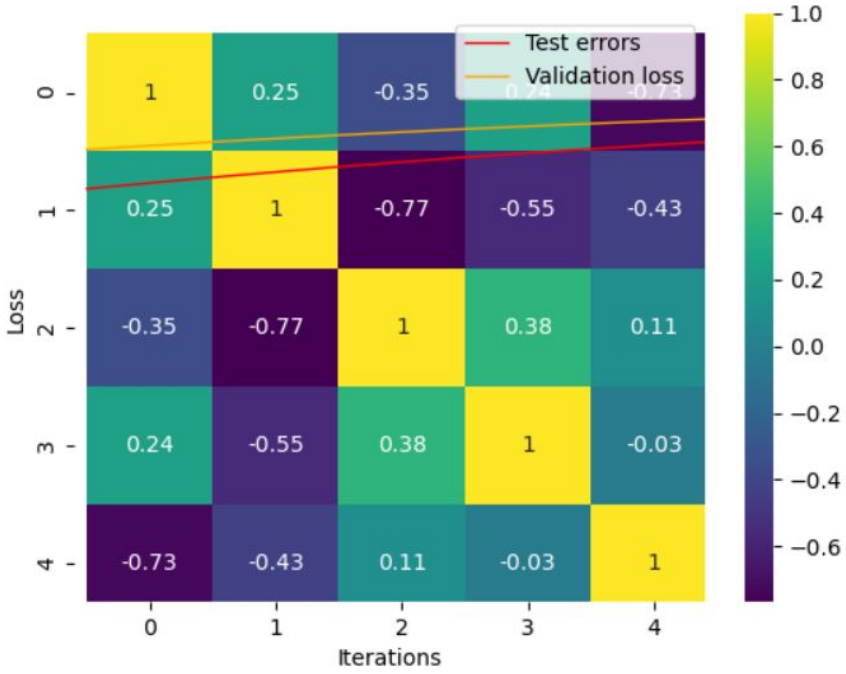
(e).

2. Logistic Regression

- **EDA Analysis:**
  - Data insights:
    - **data.shape()**
      - Tells data comprises 1336 entries and 5 characteristics. Here 4 are independent and 1 is dependent.
    - **data.info()**
      - Data has 4 classes of float values and 1 class of integer value
  - Summary Statistics:
    - **data.describe()**
      - Mean is smaller than median (which is represented by by [50%]
      - Large difference in values between [75%] and max values
      - The above two observations shows that it has extreme value outliers
  - Data Visualization:
    - **sns.heatmap(data.isnull(),cbar = False,yticklabels = False, cmap = 'viridis')**
      - We can see a clear map, which shows there are no missing values in the dataset.

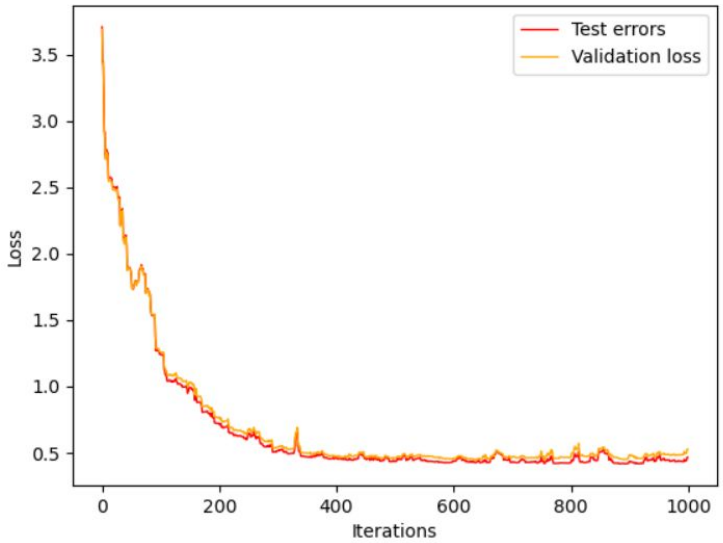


  - Checking Correlation:
    - **sns.heatmap(data.corr(),cmap = 'viridis,annot = True)**



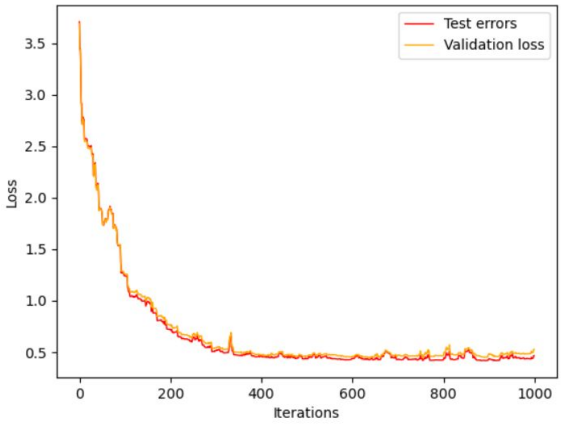
Now splitting the dataset into 7:1:2 and implementing the logistic regression

1) . Using Stochastic Gradient Descent:
    a) Here, Learning rate should be small to get better analysis. I had taken the learning rate = 0.01 and number of iterations = 1000. After running, I got the accuracy:
        - On Train Data : 0.8388829215896885
        - On Test Data : 0.6470588235294118
    b) Plot for Stochastic Gradient Descent:

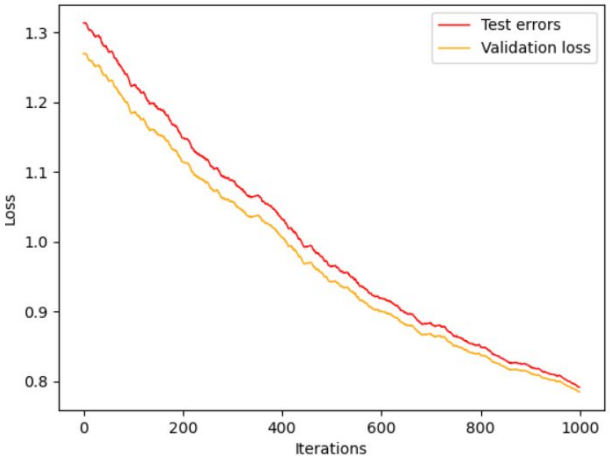


c) Re running on learning rate = 0.01: Working fine

Stochastic Gradient Descent



Then on learning rate = 0.0001 : Working fine but the graph was a bit straight downward line because it is having a very low learning rate.



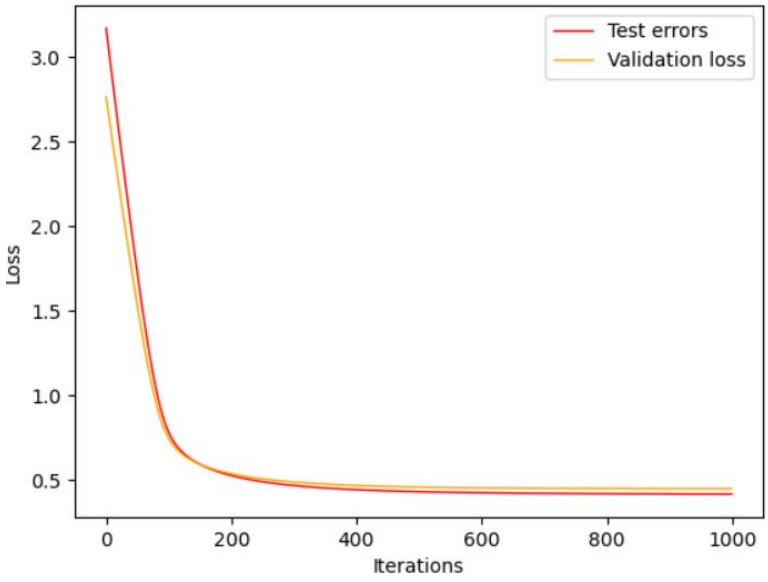
Then on learning rate = 10: Here the implementation throws error (1) Overflow error, because here the sigmoid function is taking the value to be too large and it overflows for float64.

d) Implementing Using Batch Gradient Descent:

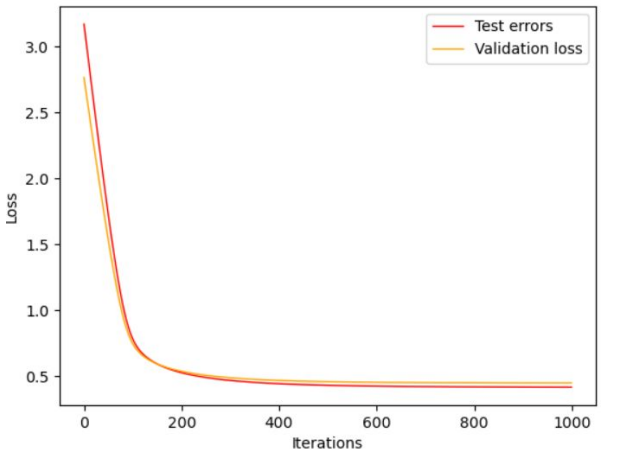
i) Here, Learning rate should be small to get better analysis. I had taken the learning rate = 0.1 and number of iterations = 1000. After running, I got the accuracy:

- On Train Data : 0.8442534908700322
- On Test Data : 0.5808823529411765

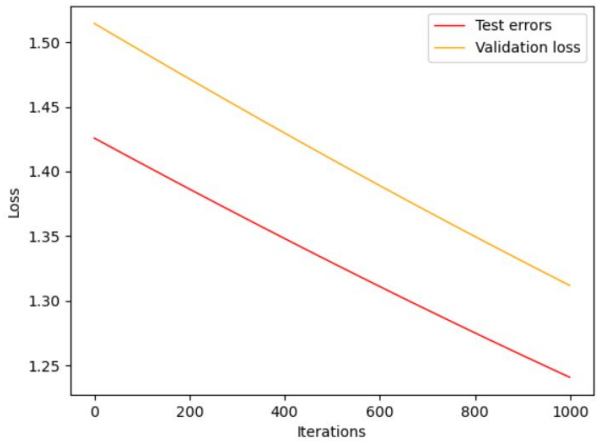
ii) Plot for Batch Gradient Descent:



iii) Re running on learning rate = 0.01: Working fine



Then on learning rate = 0.0001 : Working fine but the graph was a bit straight downward line because it is having a very low learning rate.



Then on learning rate = 10: Here the implementation throws error

(1) Overflow error, because here the sigmoid function is taking the value to be too large and it overflows for float64.

(2) Division by 0 error, because, here we get a value of y_predicted tending to 0, and this lead to create -infinity in the sigmoid function and which is taken as division by 0 error

Comparing the performance of BGD and SGD w.r.t. the following:

1. Loss Plots:
    a.
2. Number of Epochs to converge:
    a. We can see that SGD takes less number of iterations to converge than BGD. This shows that SGD is faster than the BGD.
3. Using SKlearn to Implement Logistic Regression , present in different files.
4. Keeping the learning rate = 0.01 and number of iterations = 1000

```
Accuracy on train using SDG: 80.7733619763695%
Accuracy on test data using SDG: 82.35294117647058%
```

(3).

$$\text{MSE} = \frac{\Sigma(\hat{y} - y)^2}{M}$$

Here, we have logistic regression, so $\hat{y}$ = sigmoid function

$$\hat{y} = \frac{1}{1 + e^{-\theta X}}$$

$$E = \frac{1}{M} \Sigma \left(\frac{1}{1+e^{-\theta X}} - y \right)^2$$

Taking gradient w.r.t θ

$$\frac{dE}{d\theta} = \frac{d}{d\theta} \left(\frac{1}{M} \Sigma \left(\frac{1}{1+e^{-\theta X}} - y \right)^2 \right)$$

$$\frac{dE}{d\theta} = \frac{1}{M} \Sigma \frac{d}{d\theta} \left(\frac{1}{1+e^{-\theta X}} - y \right)^2$$

$$\frac{dE}{d\theta} = \frac{1}{M} \Sigma (2) \left(\frac{1}{1+e^{-\theta X}} - y \right) \left(\frac{e^{-\theta X}}{(1+e^{-\theta X})^2} \right) (-1)(x)$$

$$\frac{dE}{d\theta} = \frac{1}{M} \Sigma (2) \left(\frac{1}{1+e^{-\theta X}} - y \right) (-1)(x)(\hat{y})(1 - \hat{y}) \quad \ldots\ldots\ldots\ldots(1)$$

It is given that if $y_{actual} = 0$, $y_{predicted} \rightarrow 1$

and $y_{actual} = 1$, $y_{predicted} \rightarrow 0$

$(\hat{y})(1 - \hat{y}) = 0$ if $y_{actual} = 0$, $(1 - \hat{y}) = 0$

$= 0$ if $y_{actual} = 1$, $(\hat{y}) = 0$

So, the gradient descent approaches $\rightarrow 0$

So, according to gradient formula:

$$\theta = \theta - \alpha \frac{dE}{d\theta}$$

Now $\frac{dE}{d\theta} \longrightarrow 0$.

So there will be no change in the value of θ. This implies that the value predicted by the model was so poor that there is no learning in the model And it will remain the same.

Cross Entropy:

$E = (y)*\log(\hat{y}) + (1 - y)*\log(1 - \hat{y})$ ……(1)

$$E = y.\log\left(\frac{1}{1+e^{-\theta X}} \right) + (1 - y).\log\left(1 - \frac{1}{1+e^{-\theta X}}\right)$$

On Solving,

$$E = y.\log\left(1 + e^{-\theta X}\right) + (1-y).\log\left(\frac{e^{-\theta X}}{1+e^{-\theta X}} \right) \quad \text{(taking -ve inside)}$$

$$E = y.\log\left(1 + e^{-\theta X}\right) + (1-y).[-\theta X - \log\left(1 + e^{-\theta X}\right)]$$

$$E = y.\log\left(1 + e^{-\theta X}\right) + (-\theta X) + \log\left(1 + e^{-\theta X}\right) - yX\theta - y.\log\left(1 + e^{-\theta X}\right)$$

$$E = (-\theta X) + \log\left(1 + e^{-\theta X}\right) + yX\theta$$

On differentiating w.r.t to θ

$$\frac{dE}{d\theta} = -X + yX + (X)\left(\frac{e^{-\theta X}}{1+e^{-\theta X}} \right)$$

$\frac{dE}{d\theta}$ = -X $\left[1 - y - \left(\frac{e^{-\theta X}}{1+e^{-\theta X}}\right)\right]$

$\frac{dE}{d\theta}$ = -X $\left[\left(\frac{1+e^{-\theta X}-e^{-\theta X}}{1+e^{-\theta X}}\right) - y\right]$

$\frac{dE}{d\theta}$ = -X $\left[\left(\frac{1}{1+e^{-\theta X}}\right) - y\right]$

$\frac{dE}{d\theta}$ = -X[$\hat{y}$ - y]

$\frac{dE}{d\theta}$ = X[y - $\hat{y}$]

Now, if y = 0, $\hat{y} \rightarrow 1$ so, $\frac{dE}{d\theta} \neq 0$ [$\frac{dE}{d\theta}$ = X]

If y = 1, $\hat{y} \rightarrow 0$ so, $\frac{dE}{d\theta} \neq 0$ [$\frac{dE}{d\theta}$ = -X]

So, according to gradient formula:

$$\theta = \theta - \alpha \frac{dE}{d\theta}$$

So, In both the cases, learning would occur in the model and .

(4). Using MLE, We can use the given dataset and perform the analysis.
(The code is present in the zip file and the dataset used is number 3)

- So, β_0 , [β_1 , β_2] will be the intercept and the coefficient of the model used.
  β_0 = -0.401
  β_1 = 0.141
  β_2 = -0.851
- Best Fitted response:
  y = $\beta_0 + \beta_1 X_1 + \beta_2 X_2$
  h(y) = $\frac{1}{1+e^{-(\beta_0+\beta_1 X1+\beta_2 X2)}}$

- On calculating the values (using numpy)
  e^{β_1} = 1.1517
  e^{β_2} = 0.426

  As, we know that
  log(odds) = $y_{predicted}$
  log(odds) = $\beta_0 + \beta_1 X_1 + \beta_2 X_2$

  Taking antilog both sides
  Odds = $e^{(\beta_0+\beta_1 X1+\beta_2 X2)}$
  Odds = $e^{(\beta_0).(\beta_1 X1).(\beta_2 X2)}$

Odds = $e^{(\beta_0)} . e^{(\beta_1 X1)}\ e^{(\beta_2 X2)}$

Odds = $e^{(\beta_0)} . e^{(\beta_1)^{X1}}\ e^{(\beta_2)^{X2}}$

So the odds is the product of these parameters raised to their respective values of $X_{i.}$. Or we can say that e^{β_1} and e^{β_2} are the odds of the particular point.

- Here we are given $X_1 = 75$ and $X_2 = 2$, using the above values of coefficient and intercept

  Y = (-0.401) + (0.141)*(75) + (-0.851)*(2)

  Y = 8.474

  $\frac{p}{1-p} = e^{(\beta_0 + \beta_1 X1 + \beta_2 X2)}$

  So, the p = $\frac{e^{(\beta_0 + \beta_1 X1 + \beta_2 X2)}}{1 + e^{(\beta_0 + \beta_1 X1 + \beta_2 X2)}}$

  = $\frac{e^{(8.474)}}{1 + e^{(8.474)}}$

  = 0.9997