Assignment 1
Machine Learning - Monsoon 2020

Q1
1.

Out of k=3,5,10, the values of k which gave the minimum average error(both rmse and mae) was k=10.

The preprocessing strategy followed:

(Denoise-Normalize and Normalize-Denoise both were tested and the following suited the best)

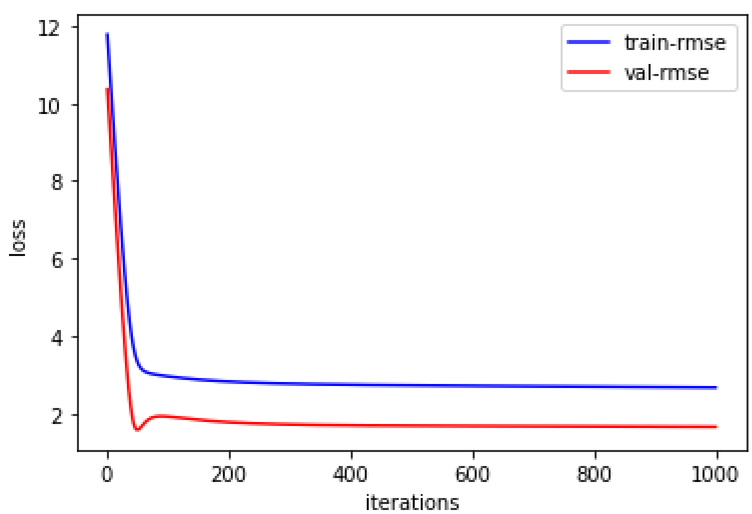
1. Remove the outliers using Z-score = (x-mean)/standard deviation: All the points with z-score>3 were removed.
2. Normalize the entire data
3. Before performing 1 & 2, NaN values were removed from the data and necessary type conversions were performed.

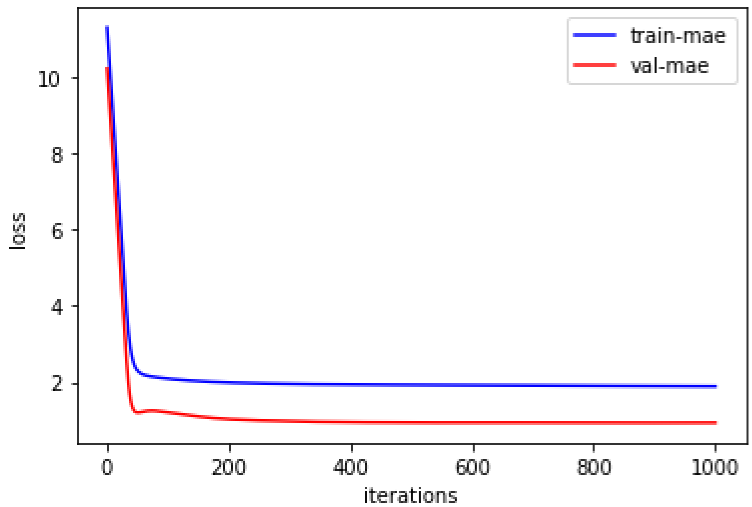
The best fold out of the k folds is plotted below.

Note: The preprocessing was only done for Dataset 2. Dataset 1 was preprocessed already.

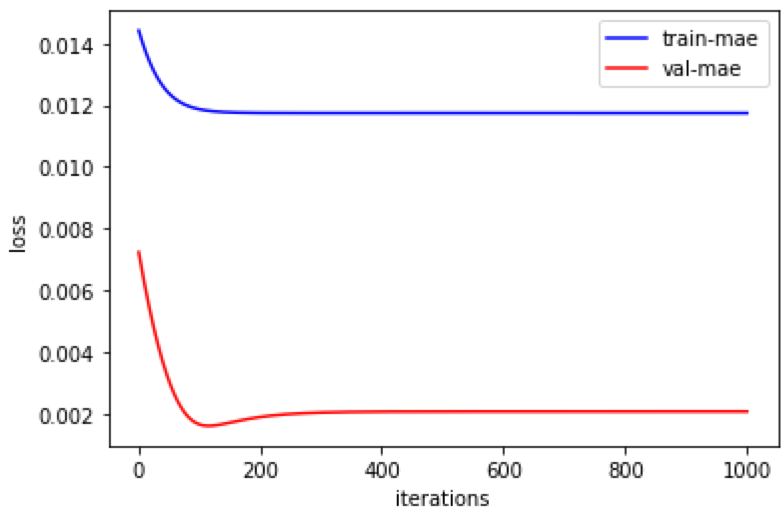
a)

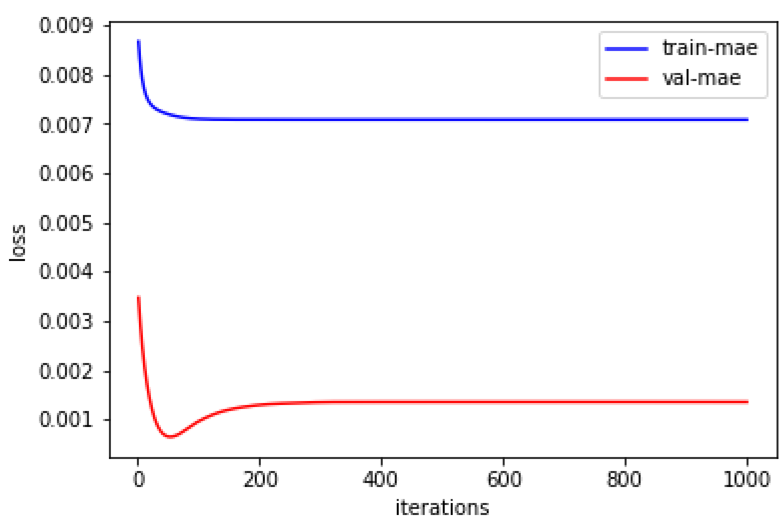
DataSet1: learning rate: 0.1, iterations:1000





DataSet2: iteration: 1000, Learning rate=0.0001





b)

DataSet 1:
k=10
RMSE: 1.6728355422753942 Best fold: 3
MAE : 1.191074079145462 Best fold: 3
DataSet 2:
k=10
MAE = 0.0015076035660499336 Best fold: 5
RMSE = 0.0020882292269219006 Best fold: 3

c)
To compare the value of RMSE and MAE through a common metric, I used MSE on both the training and testing set using the parameters obtained from running Gradient descent using RMSE and MAE.
Then, I took the average of the MSE value obtained for both testing and training set on RMSE and MAE.
The average was less for MAE. Even in the individual cases, the value was less for MAE.
Dataset 1:
Average MSE(MAE): 0.2040085792541504
Average MSE(RMSE): 0.24970269203186035
Dataset 2:
Average MSE(MAE): 0.17169952392578125
Average MSE(RMSE):0.2195751667022705

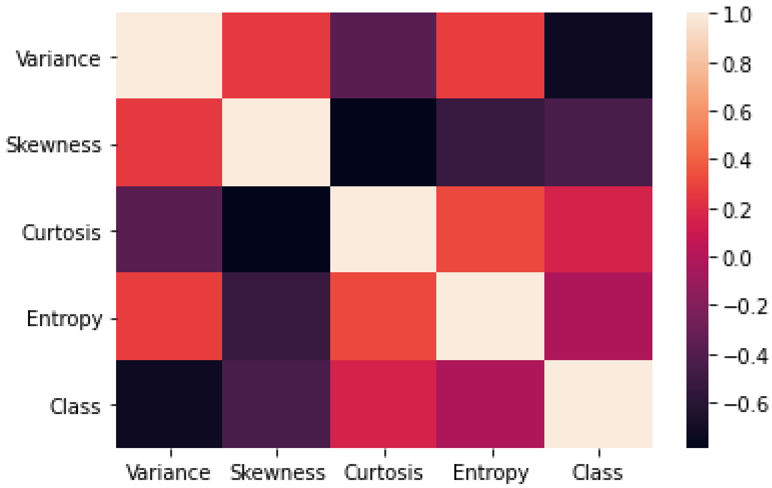
The number of iterations required for MAE to converge are less compared to RMSE.
Considering the time taken to execute the same number of iterations was less in MAE.
The same pattern was seen in both the datasets

d)
The value of RMSE will be greater than MAE in general.
Their values will be equal when the sum of squared errors for each data point will be equal to the sum of absolute errors divided by the square root of the number of samples.
sigma($(y_i - y)^{2)}$ = sigma($|y_i - y|$)/sqrt(n)
One possible condition could be when all the errors are exactly +1/-1.
When the two are equal, RMSE is preferred since RMSE penalizes the outliers more compared to MAE and hence produces better results.

e)
Optimal Parameters: [b0, b1,....,b7] = array([[3.72102929],[3.18268296],[11.17392824],[-1.23912465],[2.07195585],[-0.56813076],[-6.39990391],[0.23325777]])
MAE
Loss_train using optimum parameters:
1.0344207216173091

Loss_validation using optimum parameters
0.7980300454757314

Q2.

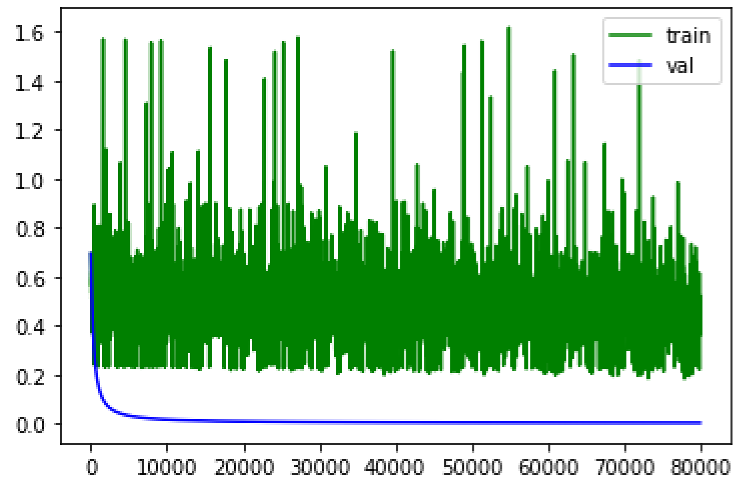


	Variance	Skewness	Curtosis	Entropy	Class
Variance	1.000000	0.264026	-0.380850	0.276817	-0.724843
Skewness	0.264026	1.000000	-0.786895	-0.526321	-0.444688
Curtosis	-0.380850	-0.786895	1.000000	0.318841	0.155883
Entropy	0.276817	-0.526321	0.318841	1.000000	-0.023424
Class	-0.724843	-0.444688	0.155883	-0.023424	1.000000

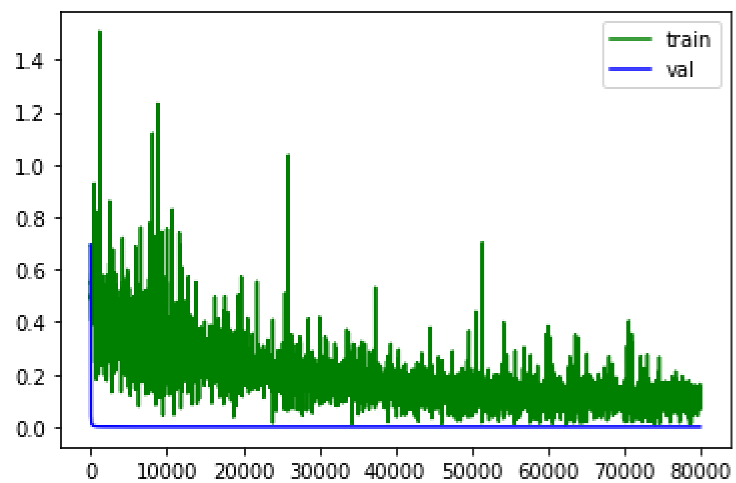
	Variance	Skewness	Curtosis	Entropy	Class
count	1372.000000	1372.000000	1372.000000	1372.000000	1372.000000
mean	0.433735	1.922353	1.397627	-1.191657	0.444606
std	2.842763	5.869047	4.310030	2.101013	0.497103
min	-7.042100	-13.773100	-5.286100	-8.548200	0.000000
25%	-1.773000	-1.708200	-1.574975	-2.413450	0.000000
50%	0.496180	2.319650	0.616630	-0.586650	0.000000
75%	2.821475	6.814625	3.179250	0.394810	1.000000
max	6.824800	12.951600	17.927400	2.449500	1.000000

2.
Using SGD:
Learning Rate=0.1
Iterations=8000
    Train Accuracy = 99.5(+0.5/-0,5)%
    Test Accuracy = 99.5(+0.5/-0.5)%
Using BGD:
Learning Rate=0.1
Iterations=80000
    Train Accuracy = 99(+0.5/-0.5)%
    Test Accuracy = 99.5(+0.5/-0.5)%%

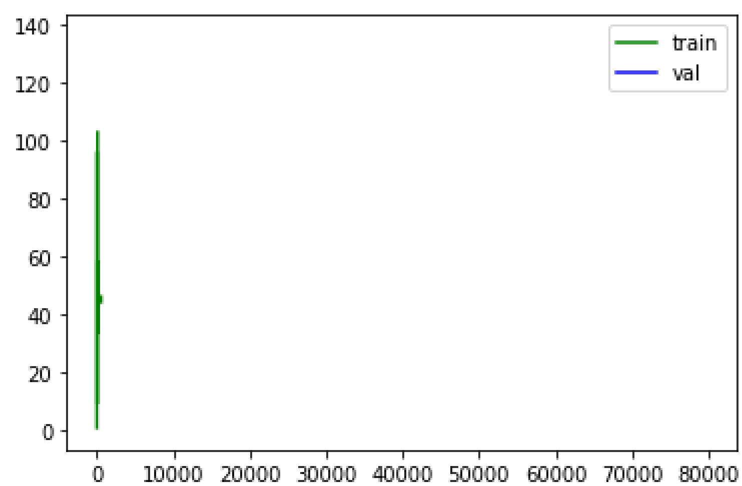
3.
SGD:



(0.0001)

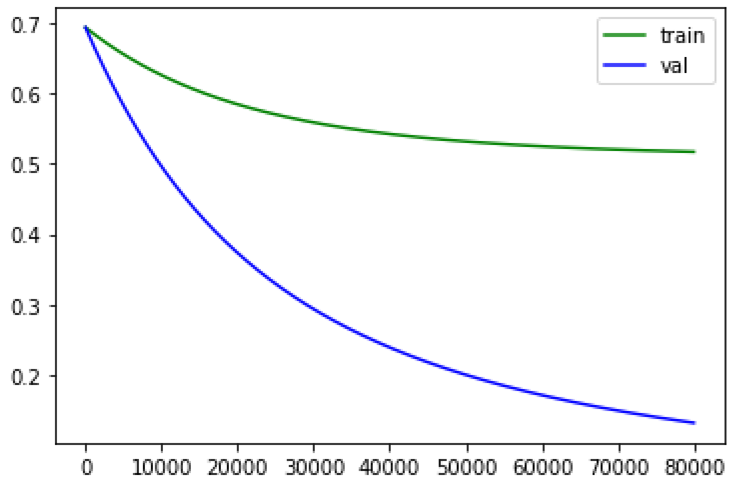


(0.01)

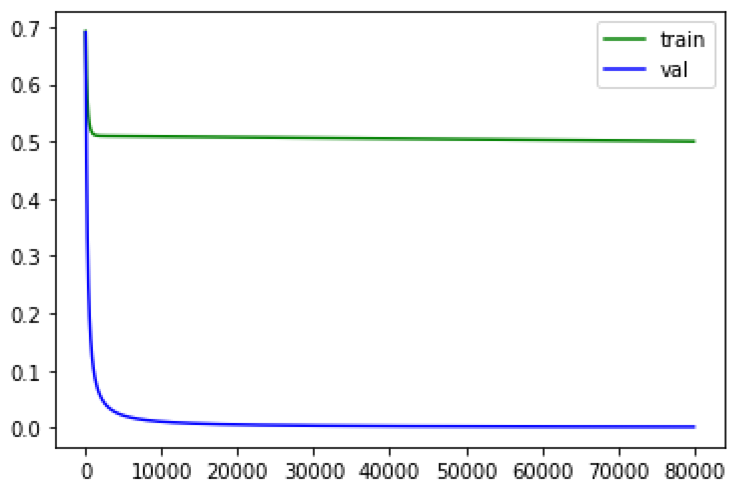


(10)

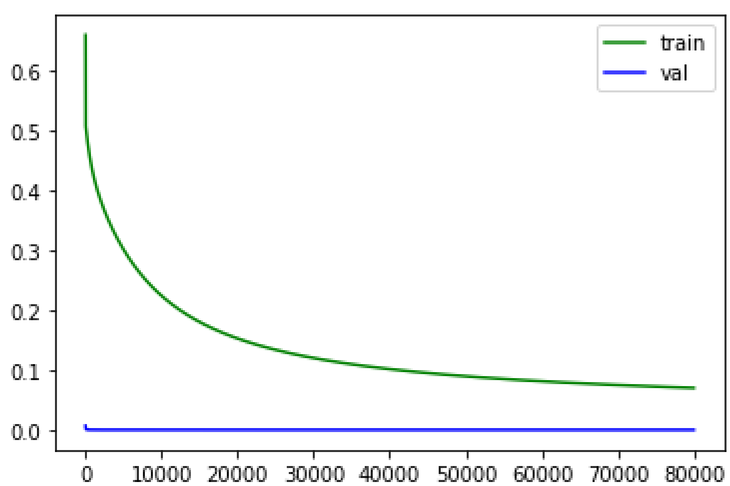
BGD:



(0.0001)



(0.01)



(10)

Comparison of SGD and BGD:

Loss Plots: In SGD, since the data points are randomly chosen, parameters would descend towards minima in a fluctuating manner, but it is for sure that we'll head towards the minima(local/global). Therefore, we see that the loss plots fluctuate initially, but they settle later. In BGD, parameters move towards the minima always via a very smooth curve(smooth decrease in the loss/cost). Therefore, we see a smooth download sloping curve for the loss function.

Number of Epochs to converge:
SGD takes a lower number of epochs to converge compared to BGD with the same learning rate, pretty much visible from the plots itself. This indicates that SGD is a lot faster than BGD. Though initially it looks like that SGD might not yield good results, but later the curve moves towards the minima.

Using sklearn logistic regression, test accuracy=99.27%, train accuracy= 99.06%

Using sklearn SGDClassifier, test accuracy = train accuracy=98.12%, test accuracy=92.7%

Time taken to run self implemented SGD = 1.61sec
Time taken to run sklearn SGD = 0.002 sec

Q3

Q3: Let x be our ~~vector~~ training set as a vector (n-samples × n-features)
and Let y be our training set as a vector (nsamples×1)

Let θ be a vector (1 × n-features)

Let consider for a data point k

$$MSE = J(\theta) = (h(\theta . x^T[k]) - y^T[k])^2$$

$$\frac{dJ}{d\theta} = 2(h(\theta . x^T[k]) - y^T[k]) \left(\frac{dh(\theta . x^T[k])}{d\theta}\right)$$

~~Now h(θ(~~ $h(\theta . x^T[k]) = \frac{1}{1+e^{-\theta . x^T[k]}}$.

$$= 2\left[\frac{1}{1+e^{-\theta . x^T[k]}} - y^T[k]\right]\left[\frac{-x^T[k]\, e^{-\theta x^T[k]}}{(1+e^{-\theta . x^T[k]})^2}\right]$$

Now we know $y_{pred}^T[k] = \frac{1}{1+e^{-\theta . x^T[k]}}$

$$1 - y_{pred}^T[k] = \frac{\cancel{1} - \cancel{1} - e^{-\theta x^T[k]}}{1+e^{-\theta x^T[k]}}$$

$$\Rightarrow \frac{dJ}{d\theta} = 2x^T[k] \underbrace{\left[y_{pred}^T(k) - y^T(k)\right]}_{\text{will be } \approx \pm 1} \underbrace{(1 - y_{pred}^T[k])(y_{pred}^T[k])}_{\text{will be } \approx 0}$$

∴ when y_{pred} is approaching zero and y_{True} is 1 or vice verce,

$$\boxed{\frac{dJ}{d\theta} \approx 0}$$

This means the gradient is ~~exactly~~ close to zero, therefore for the model to run effectively, the learning should be high or number of iterations have to be increased since the gradient is ~~exactly~~ close to zero. Other implication could be the model is not running perfectly

Cross entropy loss $J(\theta) = -\left[y\log\left(\frac{1}{1+e^{-x\theta}}\right) + (1-y)\log\left(1-\frac{1}{1+e^{-x\theta}}\right)\right]$

$$J(\theta) = y\log(1+e^{-x\theta}) - (1-y)\log\left(\frac{e^{-x\theta}}{1+e^{-x\theta}}\right)$$

$$J(\theta) = y\log(1+e^{-x\theta}) - (1-y)\left[-x\theta - \log(1+e^{-x\theta})\right]$$

$$J(\theta) = \cancel{y\log(1+e^{-x\theta})} + x\theta + \log(1+e^{-x\theta}) - xy\theta - \cancel{y\log(1+e^{-x\theta})}$$

$$J(\theta) = x\theta + \log(1+e^{-x\theta}) - xy\theta$$

$$\frac{dJ(\theta)}{d\theta} = x + \frac{1}{1+e^{-x\theta}}(e^{-x\theta})(-x) - xy$$

$$\frac{dJ(\theta)}{d\theta} = x\left(1 - \frac{e^{-x\theta}}{1+e^{-x\theta}}\right) - xy$$

$$\frac{dJ(\theta)}{d\theta} = x\left(\frac{1}{1+e^{-x\theta}}\right) - xy$$

$$\Rightarrow x\left(\frac{1}{1+e^{-x\theta}} - y\right)$$

$$\frac{dJ(\theta)}{d\theta} = x\underbrace{(y_{pred} - y)} \quad \text{where } y_{pred} = \frac{1}{1+e^{-x\theta}}$$

$$\frac{dJ(\theta)}{d\theta} = x\underbrace{(y_{pred} - y)}_{\text{close to } 1/-1}$$

Therefore, here we see that $\frac{dJ(\theta)}{d\theta}$ at a particular datapoint depends on the value of the data at that point. This is definitely better since, this function will have only one global minima while MSE loss might have a local minima and gradient might be zero at that point and may give an illusion that we have a minimum value despite we have not yet reached the global minima.

Q4 :
(i) $\beta_0 = -4.36$ $\beta_1 = 0.064$ $\beta_2 = 0.52$

maximizing log (likelihood) also maximizes the likelihood.

$\therefore$

log (likelihood is given by the function)

$$l(\omega) = \sum_{i=1}^{m} y_i \log\left(\frac{1}{1+e^{-(\beta_0 + \beta_1 x_1^i + \beta_2 x_2^i)}}\right) + (1-y_i)\log\left(1 - \frac{1}{1+e^{-(\beta_0+\beta_1 x_1^i+\beta_2 x_2^i)}}\right)$$

$\therefore$ Response function becomes:

(ii) $$\underline{l(\omega)} = \sum_{i=1}^{m} y_i \left(\frac{1}{1+e^{T}}\right) + (1-y_i) \log\left(1 - \frac{1}{1+e^{T}}\right)$$

Here, $T = -(\beta_0 + \beta_1 x_1^i + \beta_2 x_2^i)$

Substituting the values

$T = -(-4.36 + 0.064 x_1^i + 0.52 x_2^i)$

(iii) $e^{\beta_1} = 1.066$ $e^{\beta_2} = 1.682$

$\log(\text{odds}) = \beta_0 + \beta_1 x_1 + \beta_2 x_2$

$\text{odds} = e^{\beta_0 + \beta_1 x_1 + \beta_2 x_2}$

$\text{odds} = e^{\beta_0} \cdot e^{\beta_1 x_1} \cdot e^{\beta_2 x_2}$

$\text{odds} = e^{\beta_0} \cdot (e^{\beta_1})^{x_1} \cdot (e^{\beta_2})^{x_2}$

These numbers are already calculated above and therefore can be used to find the odds of a point in the data set w.r.t our fitting line.

(iv) probability $(x_1 = 75, x_2 = 2) = \dfrac{e^{\beta_0 + \beta_1 x_1 + \beta_2 x_2}}{1 + e^{\beta_0 + \beta_1 x_1 + \beta_2 x_2}}$

$$= \frac{e^{1.477}}{1 + e^{1.477}} = 0.814$$

How did I come up with this formula

$$\log(\text{odds}) = \beta_0 + \beta_1 x_1 + \beta_2 x_2$$

$$\text{odds} = e^{\beta_0 + \beta_1 x_1 + \beta_2 x_2}$$

$$\frac{p}{1-p} = e^{\beta_0 + \beta_1 x_1 + \beta_2 x_2}$$

$$p = (1-p)\, e^{\beta_0 + \beta_1 x_1 + \beta_2 x_2}$$

$$\boxed{p = \frac{e^{\beta_0 + \beta_1 x_1 + \beta_2 x_2}}{1 + e^{\beta_0 + \beta_1 x_1 + \beta_2 x_2}}}$$

Q5.

$$y_i = \beta_1 + \beta_2 x_{2i} + \beta_3 x_{3i} + \dots + \beta_k x_{ki} + \varepsilon_i$$

$$Y = X\beta + \varepsilon$$

$$(Y - X\beta) = \varepsilon$$

$$MSE(\beta) = \frac{1}{n}\sum_{i=1}^{n} (y_i - x_i(\beta))^2 = \frac{1}{n}\sum_{i=1}^{n} e_i^2(\beta)$$

$$MSE(\beta) = J(\beta) = \frac{1}{n}(\varepsilon^T \varepsilon) \quad \text{where } \varepsilon = \begin{bmatrix} e_1 \\ e_2 \\ e_3 \\ \vdots \\ e_n \end{bmatrix}$$

$$J(\beta) = \frac{1}{n}\varepsilon^T\varepsilon$$

$$= \frac{1}{n}(y - x\beta)^T(y - x\beta)$$

$$= \frac{1}{n}(y^T - \beta^T x^T)(y - x\beta)$$

$$= \frac{1}{n}(y^T - y^T x\beta - \beta^T x^T y + \beta^T x^T x\beta)$$

$$(y^T x\beta)^T = \beta^T x^T y.$$

$$\therefore \; J(\beta) = \frac{1}{n}(y^T y - 2\beta^T x^T y + \beta^T x^T x\beta)$$

$$\frac{\partial J(\beta)}{\partial \beta} = \frac{1}{n}\left(\frac{\partial \cdot y^T y}{\partial \beta} - 2\partial \beta^T x^T y + \partial \beta^T x^T x\beta\right)$$

$$= \frac{1}{n}(0 - 2x^T y + 2x^T x\beta)$$

$$= \frac{2}{n}(x^T x\beta - x^T y)$$

Now to find the minima

$$\frac{2}{n}(x^T x\beta - x^T y) = 0 \implies x^T x\beta_{min} = x^T y.$$

$$\beta_{min} = (x^T x)^{-1} x^T y$$

We've found out the least square solution.

$$(\hat{\beta}_1)_{min} = \frac{\overline{xy} - \bar{x}\,\bar{y}}{(\overline{x^2}) - \bar{x}^2}$$

$\bar{x}$ = mean of x

$\bar{y}$ = mean of y

and $$(\hat{\beta}_0)_{min} = \bar{y} - (\hat{\beta}_1)_{min}\,\bar{x}$$