ML ASSIGNMENT 1

PLEASE NOTE THIS: THE ANSWERS MAY DIFFER IN MY CASE BECAUSE I HAVEN'T USED THE VECTORIZED FORM AS THE REST OF THE CLASS. I HAVE SELF IMPLEMENTED EVERYTHING. THANK YOU SO MUCH!!!

I HAVE DOCUMENTED THE CODE IN THE .py FILE ITSELF.

Q1.

ANS.

FOR ABALONE DATASET

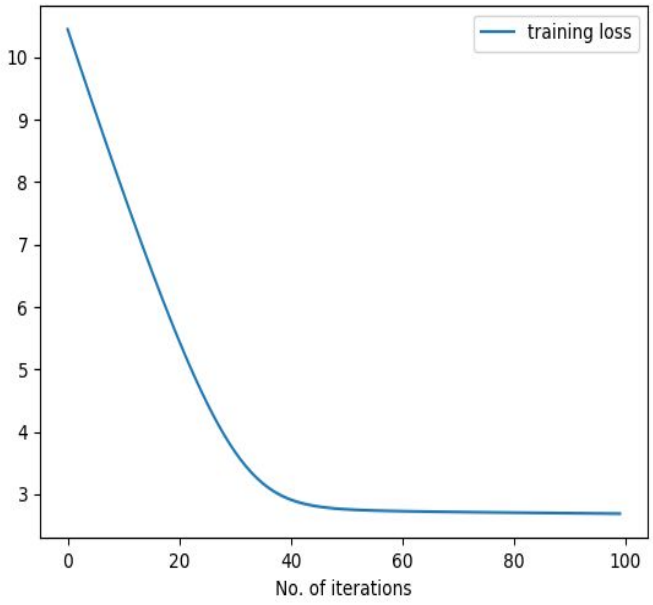
Preprocessing - All the values were already normalised according to the description given in the data set. One hot encoding (100,010,001) was used to encode the sex of the person.

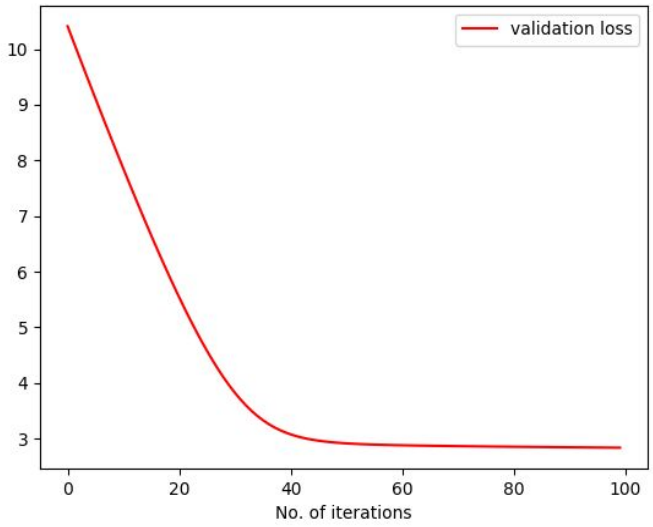
Value of K used - 5 fold classifier was used as the size of the dataset is not very large (only 4177 data points). A larger value of k was not used to prevent overfitting of data. And even the bias on the dataset was not changing much in 5 folds Vs 10 folds so to ease computational stress I choose the value of K as 5.

Linear regression models:

a) **RMSE -**

Attaching the graphs as follows for alpha = 0.1 and no. of iterations = 100

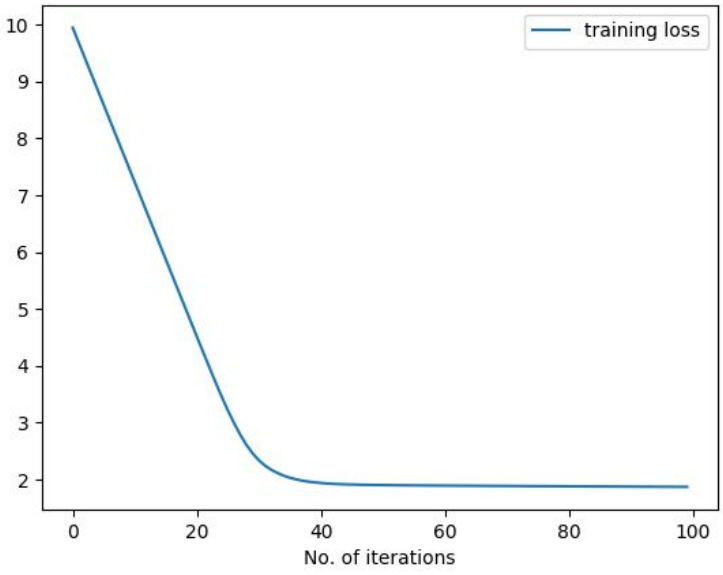


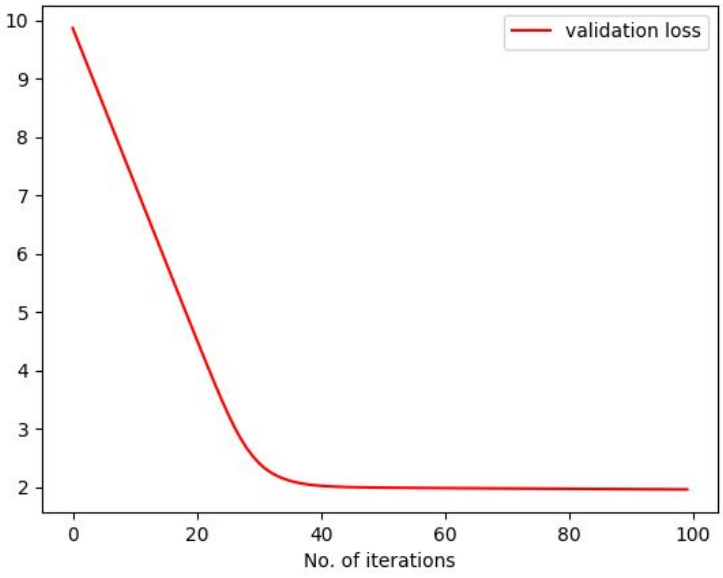


Error obtained: 2.715883271491506 on fold no 5

b) MAE -

Attaching the graphs as follows for alpha = 0.1 and no. of iterations = 100





error obtained is 1.8885489572834482 on fold no 5

Best value of RMSE error 2.215696884911102 fold no 5 alpha = 1.5 iterations = 1000
Best value achieved on MAE is error 1.637167857455581 fold no 9 alpha = 1.5 iterations = 1000
(I didn't choose this alpha for the graphs because of the large computational time.)

MAE gives better results in this dataset according to my model. I guess this is happening because most of the errors are in the range of nearly the same magnitudes.

(FOR BOTH THE PARTS)
Generally MAE<=RMSE, **equality holds if all of the errors have the same magnitude**.
Or it can be MAE*sqrt(N) >= RMSE when all of the errors come from a small or a single set of data points.
I will use RMSE because of the convex nature of the function and it is differentiable in comparison to the MAE which is non-differentiable. So if there are more outliers then I will prefer to use MAE otherwise I will use RMSE.

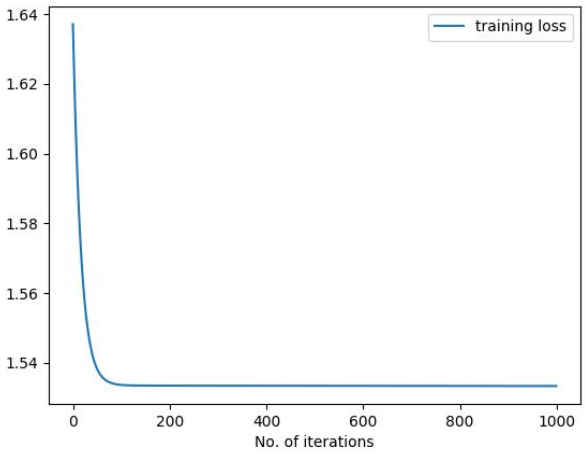
2) For Video Game Dataset

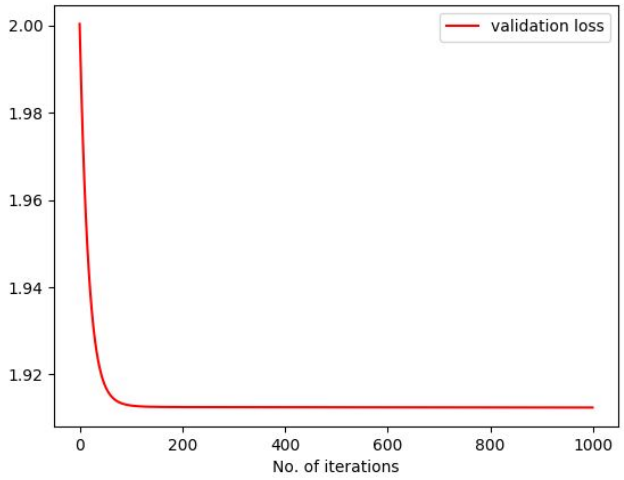
Preprocessing: NaN values were replaced by the mean value of that attribute. And only user_score, critic_score and global_sales were extracted.
In the three columns combined 17610 values were either blank or NaN.

Value of K used - 10 fold classifier was used as the size of the dataset is quite large (16719 data points). A larger value of k was used to readily shuffle the given data.

Linear Regression models :

a) **RMSE -**





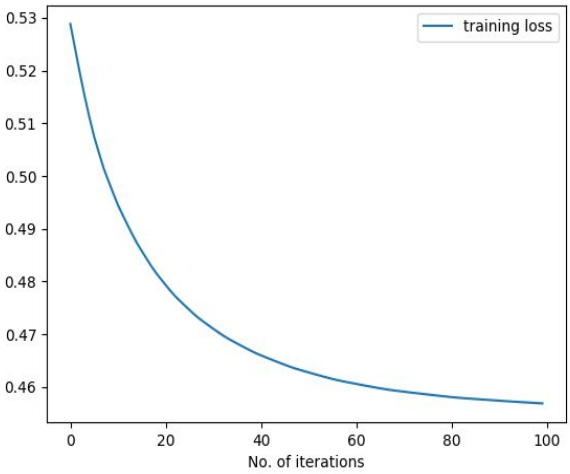
error 1.5337402131332494 fold no 5

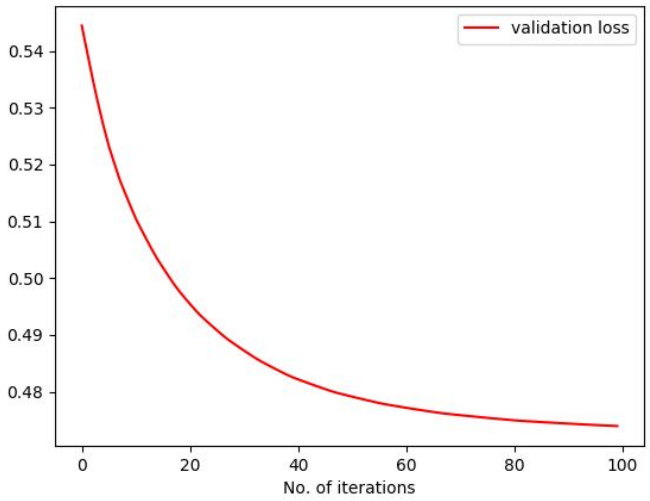
Alpha = 0.00001 iter = 1000

b) MAE-

error 0.4603382130538783 fold no 5

Alpha =0.000001 iter = 100





Best value of RMSE error1.5337402131332494 fold no 5 alpha = 0.00001 iterations = 1000
Best value achieved on MAE is error 0.4603382130538783 fold no 8 alpha = 0.000001 iterations = 100

MAE IS BETTER IN THIS CASE ALSO

e) I have implemented the normal form but it is showing very high error. I have even confirmed the formulas from the slides but still error shown is very large.
Losses for the first dataset -
Train Loss , Test Loss
(1304.12623584712, 668.0751260481914)
Losses for the second dataset -
(175.98425691465587, 87.33975298514257)

I am really unsure of this why this is happening. I have checked everything and this doesnt works for me.

Q2.

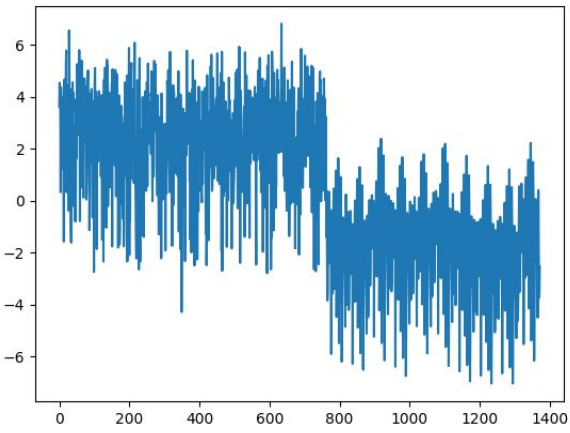
ANS.

EDA:

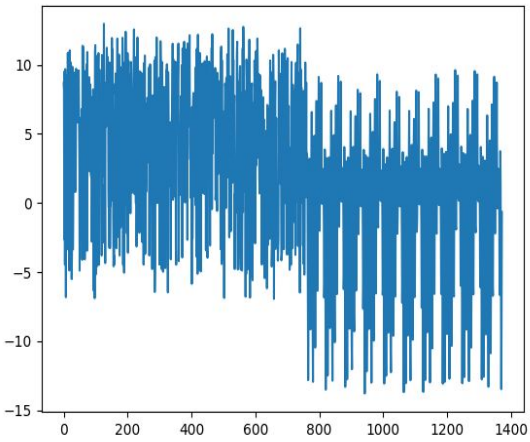
For the first value the distribution is symmetrical (and lies between 6 to -6) for the first 800 samples and last 800 samples.

Similar is the case for the 2nd (first 800 values primarily lie in range of 10 to -5 and rest 800 values in 10 to -15) and 3rd (first 800 values primarily lie in range of -5 to 10 and rest 800 values in -5 to 15) But for 4th we can determine anything and the 5th values is binary either 0 or 1.
I am attaching the photo for the distribution of the column values

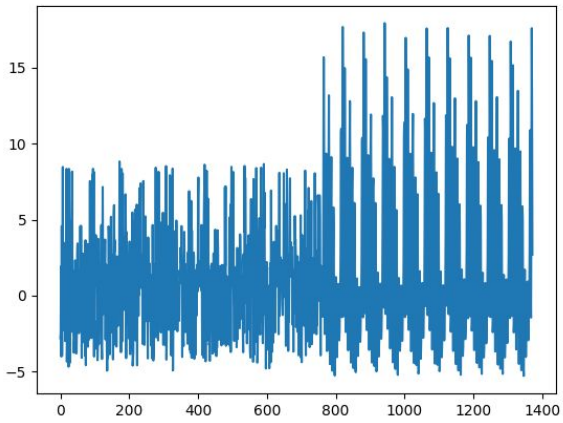
1) FIRST COLUMN VALUE



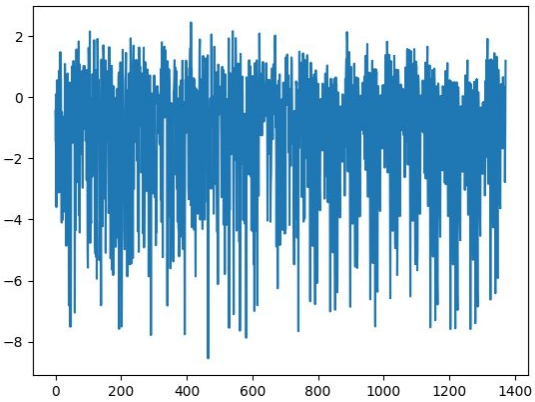
2) Second column value



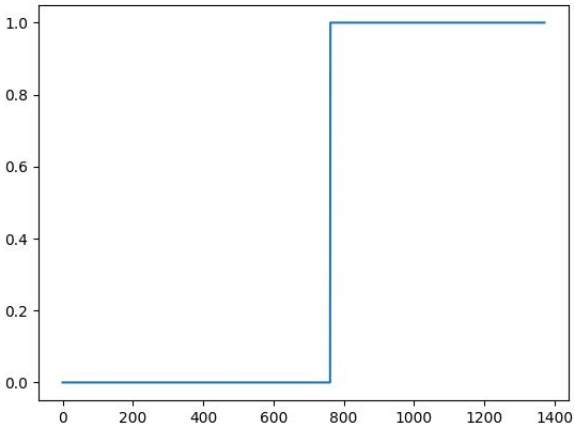
3) THIRD COLUMN VALUE



4) FOURTH COLUMN VALUE

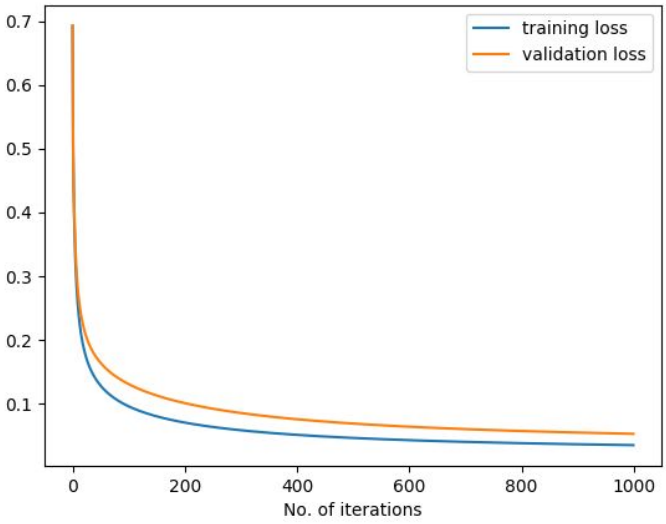


5) FIFTH COLUMN VALUE - STEP FUNCTION



BATCH GRADIENT DESCENT

Alpha = 0.1 iterations = 1000
Accuracy on training set - 98.85297184567258%
Accuracy on testing set- 98.90510948905109%

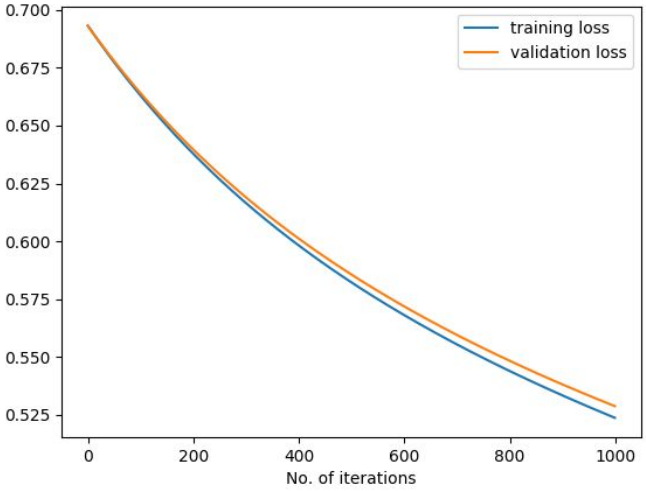


I have included both the plots in one plot for better understanding

a) Alpha = 0.0001 iterations = 1000

Accuracy on training set - 0.7643378519290928
Accuracy on testing set- 0.7262773722627737

Plot

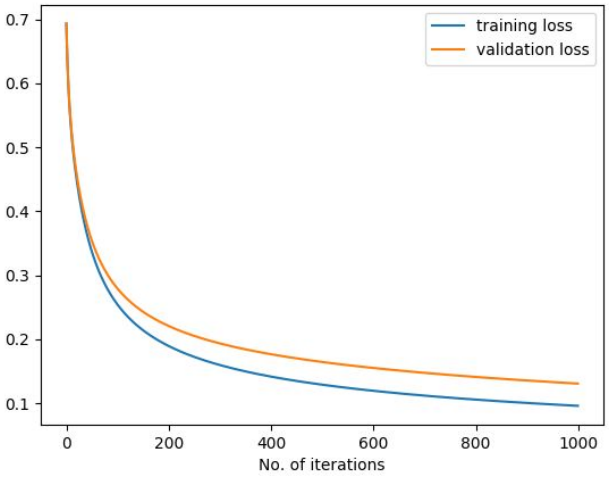


b) Alpha = 0.01 iterations = 1000

Accuracy on training set - 0.9718456725755996

Accuracy on testing set- 0.9744525547445255

Plot

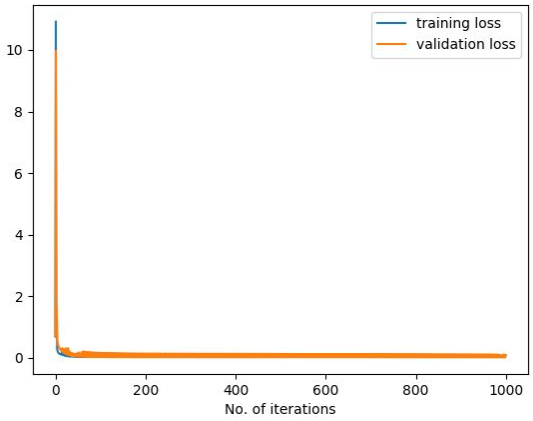


c) Alpha = 10 iterations = 1000

Accuracy on training set - 0.9885297184567258

Accuracy on testing set- 0.9890510948905109

Plot



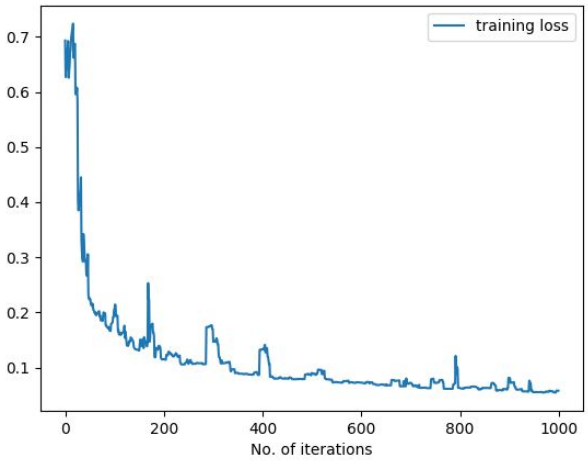
STOCHASTIC GRADIENT DESCENT

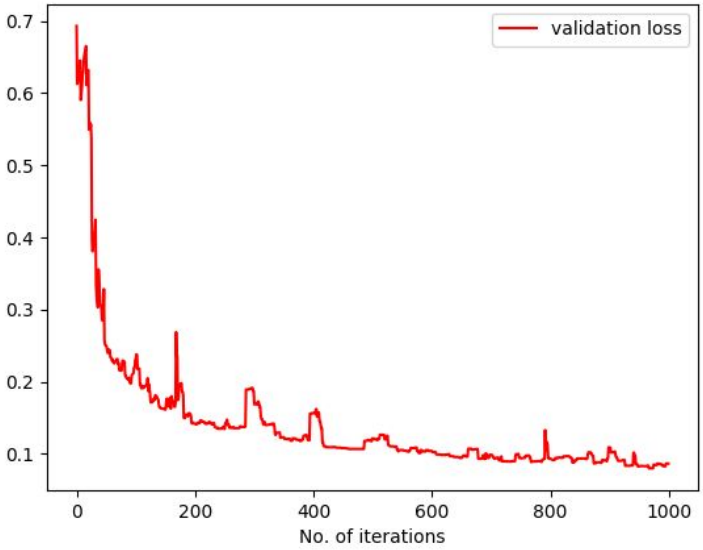
(I have taken only sample not a group samples while doing gradient descent)

Alpha = 30 iterations = 1000

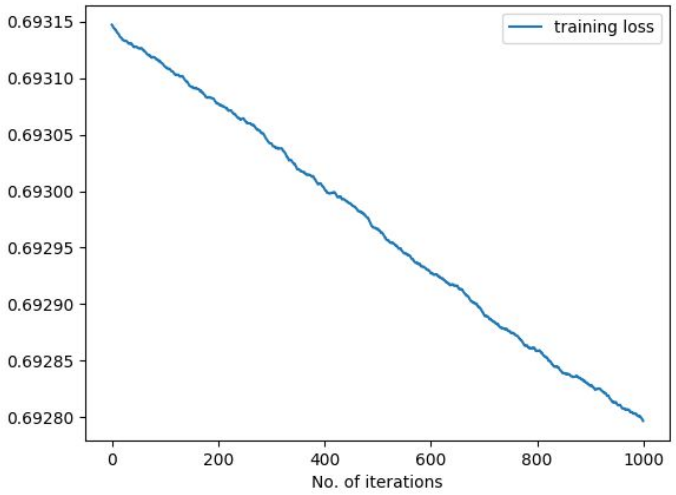
Accuracy on training set - 0.9812304483837331

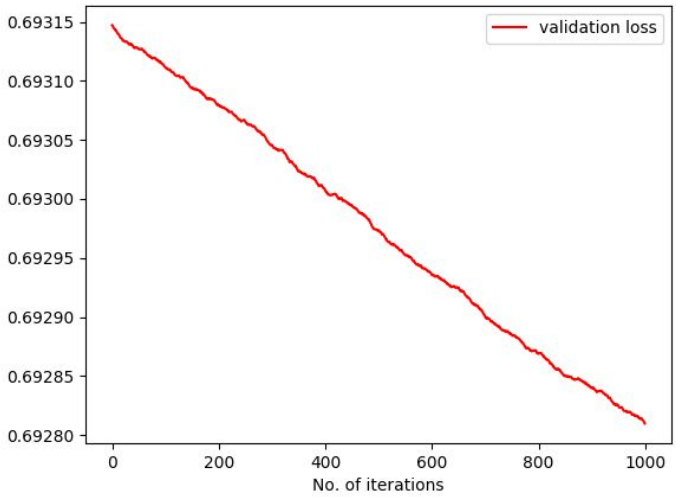
Accuracy on testing set - 0.9854014598540146



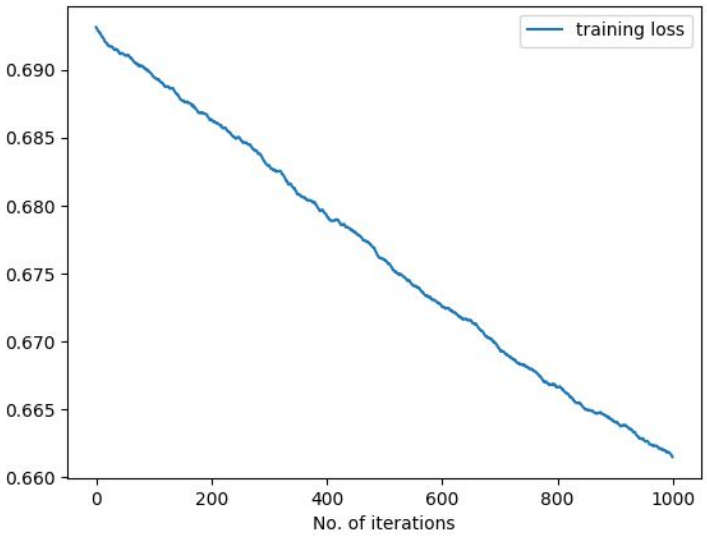


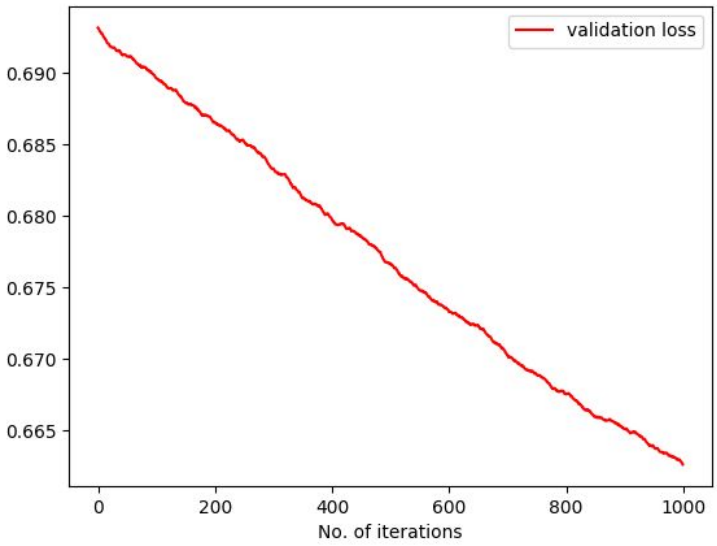
a) Alpha = 0.0001 iterations = 1000
Accuracy on training set - 0.6976016684045881
Accuracy on testing set - 0.6715328467153284



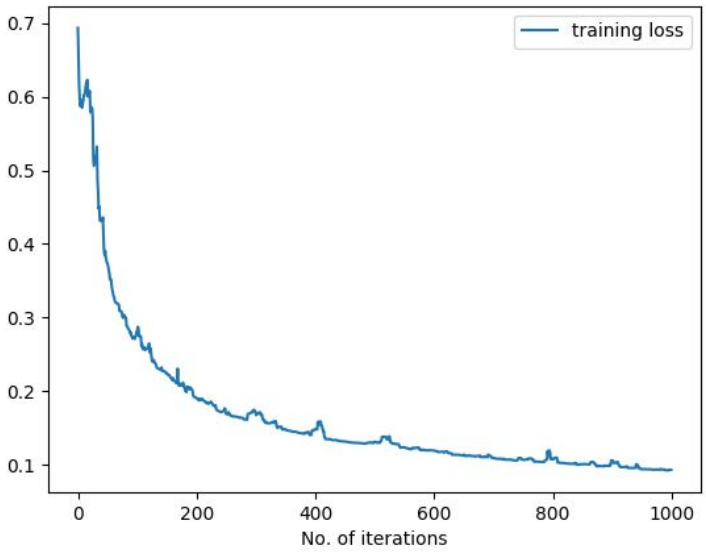


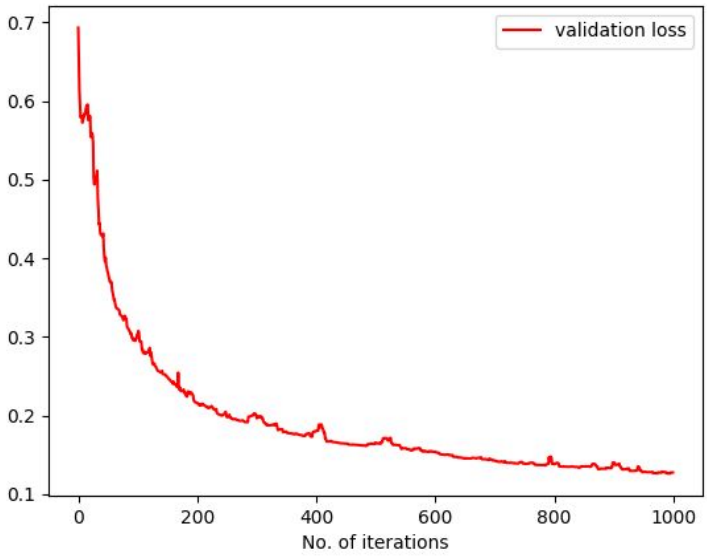
b) Alpha = 0.01 iterations = 1000
Accuracy on training set - 0.7028154327424401
Accuracy on testing set - 0.6861313868613139





c) Alpha = 10 iterations = 1000
Accuracy on training set - 0.9760166840458812
Accuracy on testing set - 0.9671532846715328





BGD VS SGD

I think that result of SGD are justifiable because since only one sample is used so it's I like a random chance so there are peaks and drops.

I think that result of BGD are justifiable because of its batch nature it is slow but is more "good" or provide better results than SGD.

a) Loss Plots -

BGD - plots are monotonically decreasing and are smooth as we have iterated over whole range of datapoints so error is minimized. Generally accuracy is better here.
SGD - plot have bends and peaks and are not monotonically decreasing as we have choosed a random sample. Generally accuracy is low here.

b) Number of epochs taken to converge -
Using the plots we can see that in BGD Plots converge faster than in SGD. In SGD they take very large amount of time even for very large alphas to converge.

c) Code named as logistic_scikit.py
d) For alpha = 0.0001 iter = 1000
Training set accuracy - 0.9885297184567258
Testing set accuracy - 0.9781021897810219

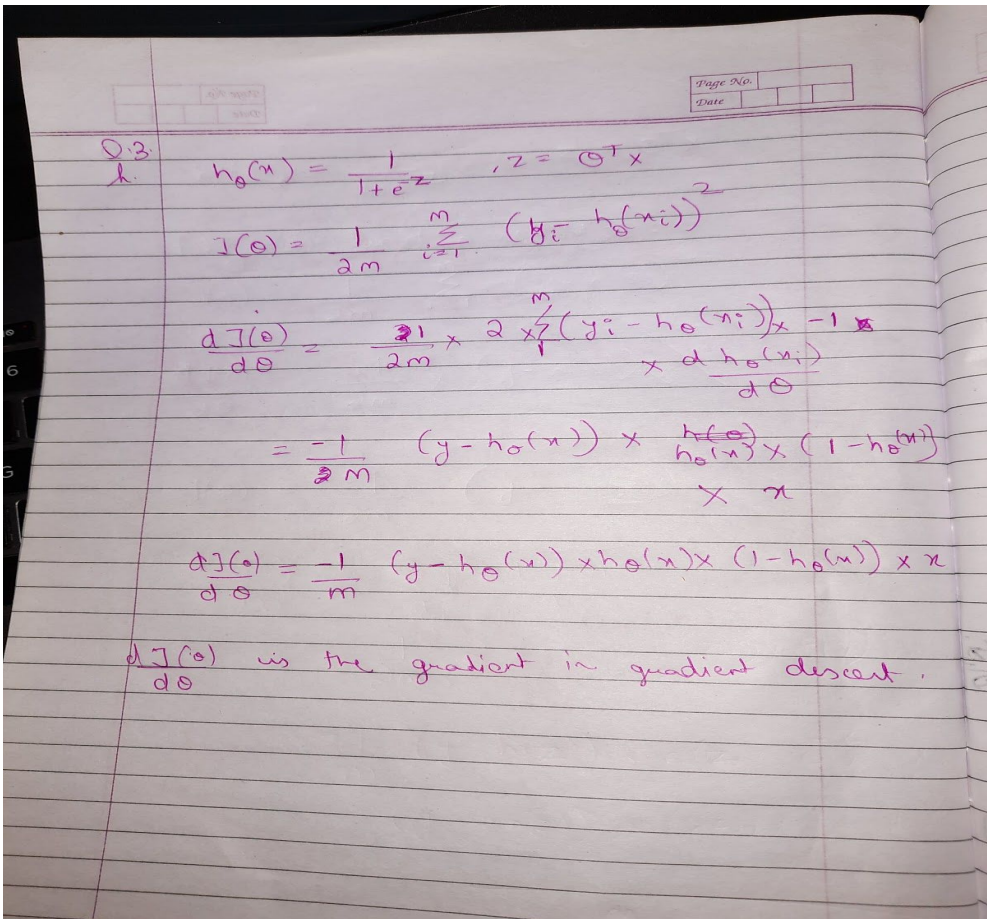
For alpha = 0.01 and iter = 1000
Training set accuracy - 0.9874869655891554
Testing set accuracy - 0.9708029197080292

For alpha = 10 iter = 1000
Training set accuracy - 0.690302398331595
Testing set accuracy - 0.6934306569343066

THESE RESULTS MATCH SOMEWHAT BGD SO I THINK THAT SCIKIT LEARN MAYBE USE MIX OF BGD AND SGD.

Q3.
ANS.
So we want to check the properties of the derivative and double derivative of cost function to check for convexity and for proving gradient in gradient descent will tend to zero.

I am attaching some photos to prove the same.



Q.3

h. $h_\theta(x) = \frac{1}{1+e^{-z}}$, $z = \theta^T x$

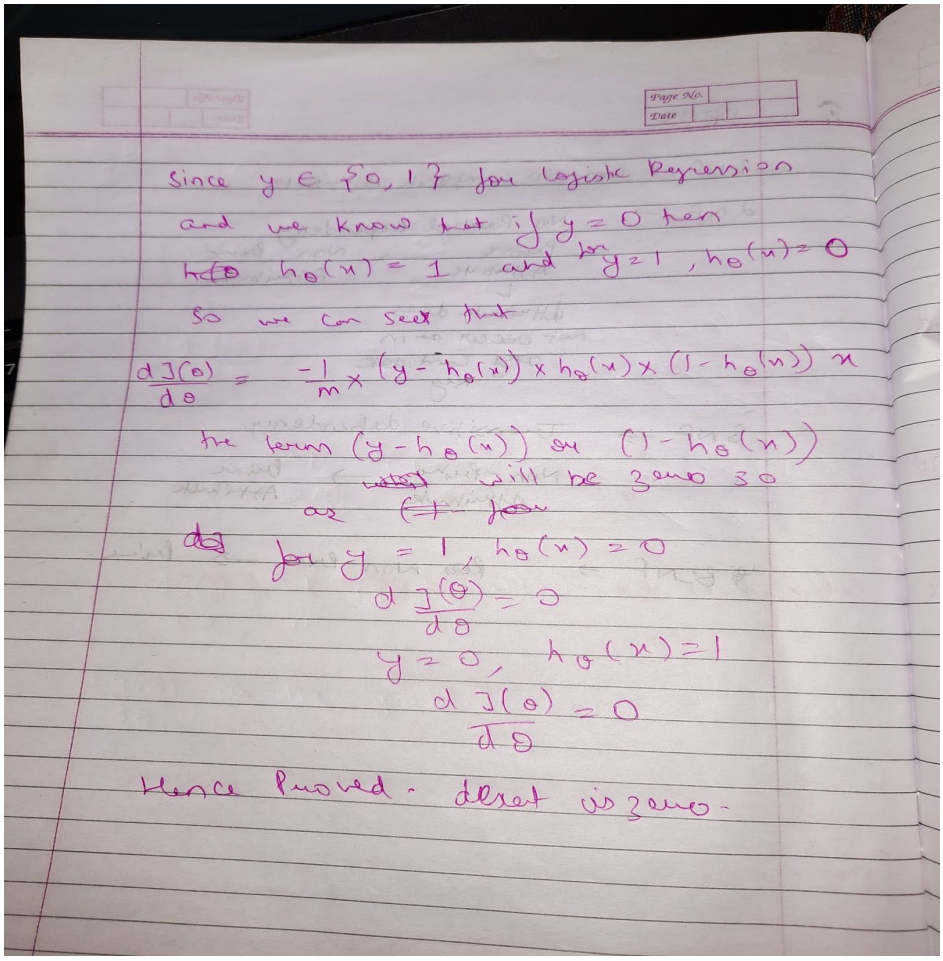
$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (y_i - h_\theta(x_i))^2$

$\frac{dJ(\theta)}{d\theta} = \frac{1}{2m} \times 2 \times \sum_{i}^{m} (y_i - h_\theta(x_i)) \times -1 \times \frac{d\, h_\theta(x_i)}{d\theta}$

$= \frac{-1}{2m} (y - h_\theta(x)) \times \frac{h(\theta)}{h_\theta(x)} \times (1 - h_\theta(x)) \times x$

$\frac{dJ(\theta)}{d\theta} = \frac{-1}{m} (y - h_\theta(x)) \times h_\theta(x) \times (1 - h_\theta(x)) \times x$

$\frac{dJ(\theta)}{d\theta}$ is the gradient in gradient descent.



The above photo proves the first part of the question.
Its implications are that the model would never be able to learn properly as

1) Gradient is zero
2) The function is not convex

To prove 2nd part (MSE is not convex) I am attaching some photos of the derivative and double derivative of the cost function(MSE) .

Page No.
Date

Lets check convexity of a func which can be tested by
double derivative is positive

$$\frac{d^2 J(\theta)}{d\theta} = -2\left[y - 2y \times h_\theta(x) - 2h_\theta(x) + 3h_\theta(x)^2\right] \times x \cdot x \times h_\theta(x) \times (1 - h_\theta(x))$$

$h_\theta(x) = [0, 1]$

$h_\theta(x)(1 - h_\theta(x)) \in [0, \frac{1}{4}]$

$$= -2x^2\left[3h_\theta(x)^2 - 2h_\theta(x)(y+1) + y\right]$$

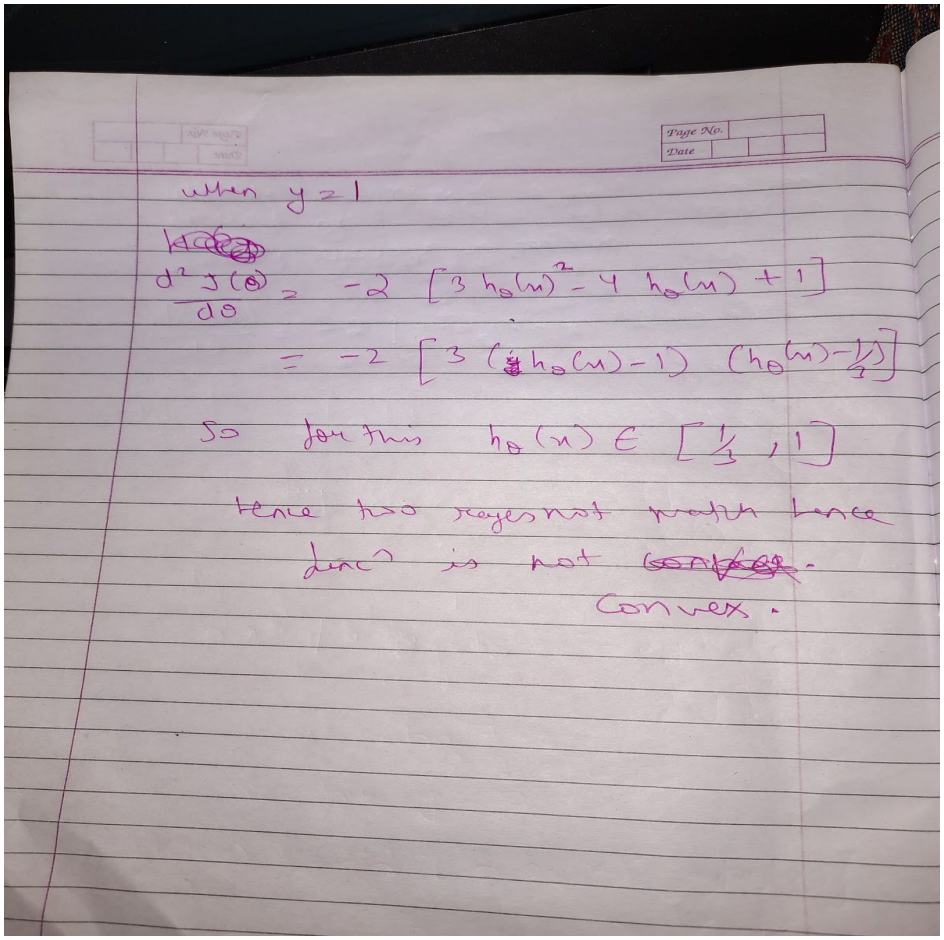
x^2 also +ve

So for $y = 0$

$$\frac{d^2 J(\theta)}{d\theta} = -2\left[3h_\theta(x)^2 - 2h_\theta(x)\right]$$

$$= -2\left[3h_\theta(x)\left(h_\theta(x) - \frac{2}{3}\right)\right]$$

if $h_\theta(x) \in [0, \frac{2}{3}]$ then +ve



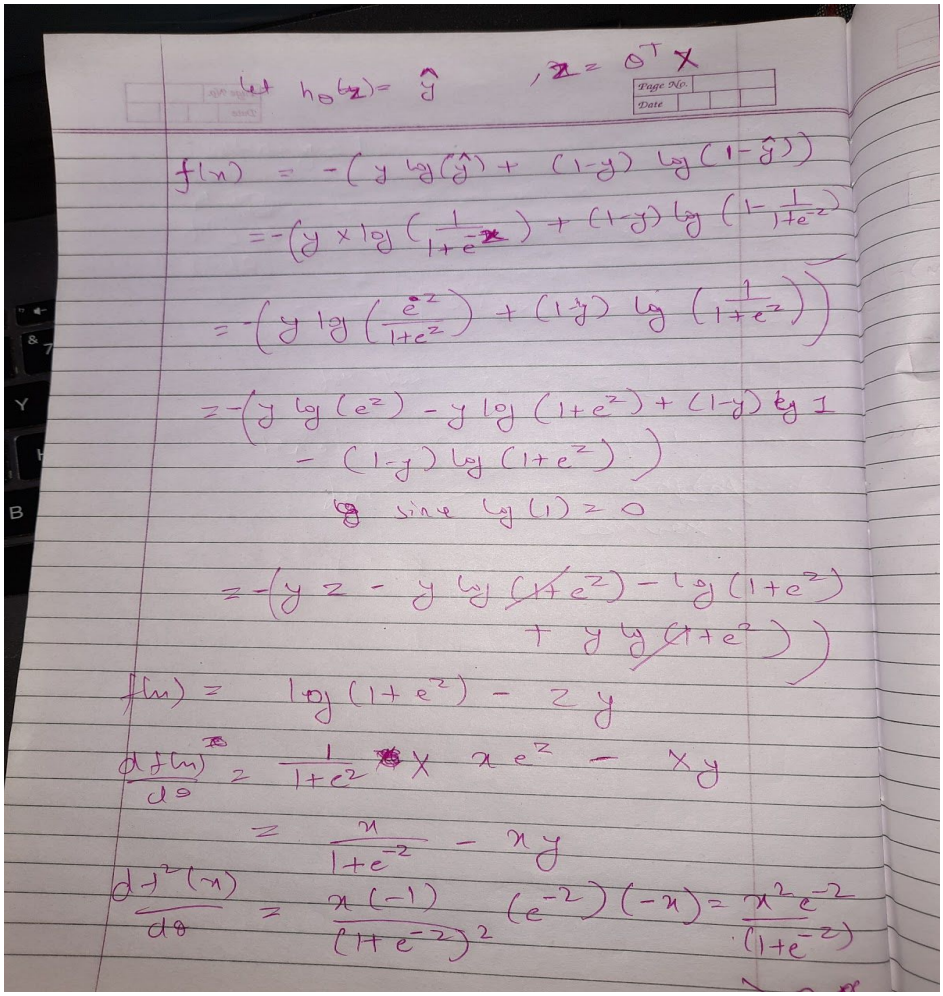
We can see that for y = 0 for the derivative to be always positive H(x) should lie between [0,⅔] But for y = 1, H(x) should be between [⅓,1]. So for H(x) in [⅔,1] and y = 0 the derivative will be negative which proves that the function is not convex.

Cross - Entropy Loss

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

So we had a convexity problem in the MSE so to counter this we use Cross Entropy Loss function.

And proof of the same is attached below



Since the double derivative is always positive the function is always convex for all x.

Q5.
Ans

Q5.

Page No.
Date

h

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

The predicted output will be of form

~~$\hat{y}_i = \beta_1 + \beta$~~

$$\hat{y}_i = b_1 + b_2 X_{2i} + b_3 X_{3i} + \dots + b_k X_{ki}$$

So the residue would be

$$e_i = y_i - \hat{y} = y_i - b_1 - b_2 X_{2i} - b_3 X_{3i} - b_k X_{ki}$$

So we have to minimize this

$$f(\beta) = \min_{b_1, b_2, b_3, \dots b_k} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

~~$\frac{d}{d\beta}$ $f(\beta) = (y_i - y)$~~

$$= (y_i - X\beta)'(y_i - X\beta)$$

$$= y'y + \beta' X' X \beta - 2\beta' X' y$$

(Assuming $f(\beta)$ is convex valued differentiable func^n)

$$\frac{d f(\beta)}{d \beta} = 2X'X\beta - 2X'y$$

$$\frac{d^2 f(\beta)}{d\beta^2} = 2X'X$$

So normal eq^n is

$$\frac{df(\beta)}{d\beta} = 0$$

$$X'X\beta = X'y$$

$$b = (X'X)^{-1}X'y$$

(b' is set of vectors from all possible vectors of β that minimizes residual sum)

Assumption

rank $(X) = k$

for $X'X$ to give positive and definite solution of eq^n