24/07/25

LAB-4

Linear, Multiple, Logistic Regression.

Algorithm and Pesudocode (Linear Regression)

1. Input : A dataset with one independent variable (x) and one dependent varible (y).
2. Initialize : Set the coefficients (m) (slope) and (b) (intercept) to 0.
3. Training :
   For each iteration
   Calulate the predicted value : (y) = mx+b
   Calulate the error : {Text t{error} = y - (...)
   Update the coefficients :
   $m = m + \alpha \cdot \frac{1}{n} \sum (error \cdot x)$
   $b = b + \alpha \cdot \frac{1}{n} \sum (error)$
4. Output : the coefficients (m) and (b) function

LinearRegression (x, y, alpha, iterations)
   intialize m=0, b=0
   n = length(x)
   for i from 1 to iteration:
      y_pred = m* x + b
      error = y - y_pred
      m = m + alpha * (1/n) * sum(error * x)
      b = b + alpha * (1/n) * sum (error)
return m, b

M T W T F S S
Page No.:
Date:
YOUVA

ii) Algorithm and Pesudocode (Multiple Regression)

1. Input:- A dataset with multiple independant variables (X1, X2 , ..., Xn) and one dependent variable (y).
2. Initialize :- Set the coefficients (b0, b1 ... bn) to
3. Training:
   For each iteration:
   Calculate the predicted value = y = b0 + b1x1 + b2x2 ..... bnxn)
   Calculate the error = y - $\hat{y}$
4. Update the coefficients:
   for each j from 0 to (n):
   $bj = b_j + \alpha \cdot \frac{1}{m} \Sigma$ (Error.x).
5. Output: the coefficients (b0, b1, ... bn)

Pseudocode :- function MultipleReg (x , y, α, iter):

```
Initialize = b = [ 0, 0... 0]
m = len (x)
for i from 1 to i:
    y_pred = b[0] + sum (b[j] * x[i] for j
  in i to n):
    error = y - y_pred
  for j from 0 to n:
        b[j] = b[j] + alpha * (1/m) * sum(error * x[j])
return b.
```

iii Algorithm & Pesudocode for logistic regression

1. Initialize weights and bias
2. Compute the weighted sum:

$$z = w_1 x_1 + w_2 x_2 + \dots + w_n x_n + b$$

3. sigmoid function:

$$\sigma(z) = \frac{1}{1+e^{-z}}$$

4. Compute the loss:

$$loss = \frac{-1}{m} \sum_{i=1} [y_i \log(\hat{y}_i) + ((1-y_i)\log(1-\hat{y}_i))]$$

5. Update weights using gradient descent:

$$w = w - \alpha \frac{\partial loss}{\partial w} \quad b = b - \alpha \frac{\partial loss}{\partial b}$$

6. Repeat Steps 2 - 5.
7. Make predictions : if $\sigma(z) \geq 0.5$, perdict 1 ; otherwise, predict 0.

Code :- Initialize (w) & (b) to small random value
For iter in ran(num_epochs):
z = w * x + b ; y_pred : sigmoid(z) ;
loss : _mean (y * log * log (y_pred)
Compute gradients :
dw = mean ((y_pred - y) *
b = mean (y_pred - y)
Return final weights (w) and bias (b)