



Date.....

# Writing my own Ridge Regression Gradient Descent class:-

class myRidge\_GD:

def \_\_init\_\_(self, learning\_rate, epochs, alpha):

self.learning\_rate = learning\_rate

self.epochs = epochs

self.alpha = alpha

self.coef\_ = None

self.intercept\_ = None

$$W_{\text{new}} = W_{\text{old}} - \eta \text{ Derivative}$$

$$\text{Derivative} = \frac{\Delta L}{\Delta W} = (X^T \cdot XW - X^T \cdot Y + \lambda W)$$

def fit(self, X\_train, Y\_train):

self.coef\_ = np.ones(X\_train.shape[1]) # Setting the values of

self.intercept\_ = 0 # Initial value of intercept

W = np.insert(self.coef\_, 0, self.intercept\_)

X\_train = np.insert(X\_train, 0, 1, axis=1)

for a in range(self.epochs):

$$\text{Derivative} = (\text{np.dot}(X_{\text{train}}^T, X_{\text{train}}) \cdot \text{dot}(W) - \text{np.dot}(X_{\text{train}}^T, Y_{\text{train}}) + \text{self.alpha} * W)$$

$$W = (W - (\text{self.learning\_rate} * \text{Derivative}))$$

self.coef\_ = W[1:]

self.intercept\_ = W[0]

def predict(self, X\_test):

return (np.dot(self.coef\_, X\_test) + self.intercept\_)