Writing my own class...

$$\boxed{W=(X^T\cdot X+\lambda I)^{-1}\cdot(X^T\cdot Y)}$$

```
class Mera_Ridge_nDim:
    def_init_ (self, alpha):
        self.alpha = alpha
        self.coef_ = None
        self.intercept_ = None

  def fit (self, X_train, Y_train):
#for a in range (X_train.shape[0]): # No need for iterations required...
      X_train = np.insert (0, 1, axis=1)
       I = np.identity (X_train.shape[1])
       ~~result = np.linalg.inv (X_train.~~
       result = (np.linalg.inv (np.dot (X_train.T, X_train) + self.alpha * I
                                .dot (X_train.T, Y_train)))
    self.intercept = result[0]
    self.coef = result[1:]

 def predict (self, X_train):

    return np.dot (self.coef, X_train) + self.intercept
```