Mathematical Formulation :

ie

	x_1 cgpa	x_2 iq	y placement
β_0 offset	β_1 8	80	8
	7	β_2 70	7
	5	120	15
	β_1	β_2	

let, given data of 3 students

$y_1 = 8,\ y_2 = 7,\ y_3 = 15$

$\hat{y_1} = ?,\ \hat{y_2} = ?,\ \hat{y_3} = ?$

$$\hat{y_1} = \beta_0 + \beta_1(8) + \beta_2(80)$$

$$\hat{y_2} = \beta_0 + \beta_1(7) + \beta_2(70)$$

$$\hat{y_3} = \beta_0 + \beta_1(5) + \beta_2(120)$$

notation,

cgpa	iq	placement
x_{11}	x_{12}	y_1
x_{21}	x_{22}	y_2
x_{31}	x_{32}	y_3

$$\hat{y_1} = \beta_0 + \beta_1(x_{11}) + \beta_2(x_{12})$$

$$\hat{y_2} = \beta_0 + \beta_1(x_{21}) + \beta_2(x_{22})$$

$$\hat{y_3} = \beta_0 + \beta_1(x_{31}) + \beta_2(x_{32})$$

Similarly, for 'm' columns & 'n' students (rows)

$$\begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \hat{y}_3 \\ \vdots \\ \hat{y}_n \end{bmatrix} = \begin{bmatrix} \beta_0 + \beta_1 X_{11} + \beta_2 X_{12} + \beta_3 X_{13} + \cdots + \beta_m X_{1m} \\ \beta_0 + \beta_1 X_{21} + \beta_2 X_{22} + \beta_3 X_{23} + \cdots + \beta_m X_{2m} \\ \beta_0 + \beta_1 X_{31} + \beta_2 X_{32} + \beta_3 X_{33} + \cdots + \beta_m X_{3m} \\ \vdots \\ \beta_0 + \beta_1 X_{n1} + \beta_2 X_{n2} + \beta_3 X_{n3} + \cdots + \beta_m X_{nm} \end{bmatrix}$$

$n \times 1$
rows cols

$n \times 1$
rows cols

dot-product

$$\underbrace{\begin{bmatrix} 1 + X_{11} + X_{12} + \cdots + X_{1m} \\ 1 + X_{21} + X_{22} + \cdots + X_{2m} \\ \vdots \\ 1 + X_{n1} + X_{n2} + \cdots + X_{nm} \end{bmatrix}}_{n \times (m+1)} \underbrace{\begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_m \end{bmatrix}}_{(m+1) \times 1}$$

$$\underbrace{\hat{y}}_{\substack{n \times 1 \\ \text{(verified)}}} = \underbrace{X}_{n \times (m+1)} \underbrace{\beta}_{(m+1) \times 1}$$

———— equation (1)

★ Error function of MLR :

$$\boxed{E = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2}$$

for simple L.R

but here we are dealing with multiple-values Hence, we convert it in form of matrix.

We have, $\hat{y} = X\beta$

predicted value
~~(on hyperplane)~~

$$y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \qquad \hat{y} = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_n \end{bmatrix}$$

$$e = y - \hat{y} = \begin{bmatrix} y_1 - \hat{y}_1 \\ y_2 - \hat{y}_2 \\ \vdots \\ y_n - \hat{y}_n \end{bmatrix}$$

(n×1) shape

$$e^T e = \underbrace{[\, y_1 - \hat{y}_1 \quad y_2 - \hat{y}_2 \quad \cdots\cdots \quad y_n - \hat{y}_n \,]}_{(1\times n)\text{ shape}} \underbrace{\begin{bmatrix} y_1 - \hat{y}_1 \\ y_2 - \hat{y}_2 \\ \vdots \\ y_n - \hat{y}_n \end{bmatrix}}_{(n\times 1)\text{ shape}}$$

$$e^T e = (y_1 - \hat{y}_1)^2 + (y_2 - \hat{y}_2)^2 + \cdots\cdots + (y_n - \hat{y}_n)^2$$

$= \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$ (verified)

Hence, $\boxed{E = e^T e}$ for MLR

equn (2)

→ **Minimizing Error:**

$$E \Rightarrow (y - \hat{y})^T (y - \hat{y})$$

$$\Rightarrow (y^T - \hat{y}^T)(y - \hat{y})$$

$$\Rightarrow y^T y \boxed{- y^T \hat{y} - \hat{y}^T y} + \hat{y}^T \hat{y}$$

$y^T \hat{y} = \hat{y}^T y$

Let $A = y$; $B = \hat{y}$

to prove: $A^T B = B^T A$

$(A^T B)^T = B^T A$

propo used

(i) $(AB)^T = B^T A^T$

(ii) $(A^T)^T = A$

$y^T \hat{y} \leftarrow$ to prove: $A^T B$ is symm.

$$\boxed{\underbrace{A^T B}_{C} = (A^T B)^T \qquad C = C^T}$$

$C = C^T$
symmetric matrix

e.g. $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \xrightarrow{T} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

$\Rightarrow y^T X \beta$

y^T: $(1 \times n)$; X: $n \times (m+1)$; β: $(m+1) \times 1$

$X\beta$: $(n \times 1)$

$(1 \times 1) \rightarrow$ scalar matrix

Hence, $C = C^T$ (symmetric matrix)

Hence, continuing the equn,

$$E = y^T y - 2 y^T \hat{y} + \hat{y}^T \hat{y} \quad \text{—— equn. (3)}$$

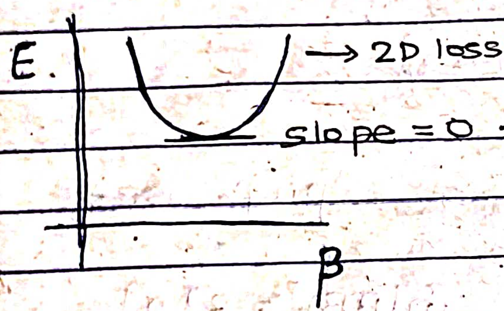
$$E = y^T y - 2 y^T X \beta + \beta^T X^T X \beta \quad \text{—— equn. (4)}$$

$\boxed{E(\beta)}$

$y \rightarrow$ data o/p

$X \rightarrow$ data i/p

Error is depending on β,
find such value of β matrix for which E is minimum $\longrightarrow$



$\longrightarrow \frac{\partial E}{\partial \beta} = 0$

$$\frac{dE}{d\beta} = 0 - 2 y^T X + 2 \beta^T X^T X = 0 \quad \text{—— eq (5)}$$

propo used:

if, $x^T A x = \alpha$

then

$\frac{d\alpha}{dx} \Rightarrow 2 x^T A$

$\Downarrow$

only true when A is symmetric

$\Rightarrow \frac{d(\beta^T X^T X \beta)}{d\beta} \Rightarrow \frac{d(x^T A x)}{d\beta}$

$\Rightarrow 2 x^T A$

$\Rightarrow 2 \beta^T X^T X$

from ⑤, $\beta^T X^T X = y^T X$

↓ we have to find "β"

$$\beta^T \underbrace{X^T X (X^T X)^{-1}} = y^T X (X^T X)^{-1}$$

$$\beta^T I = y^T X (X^T X)^{-1}$$

$$(\beta^T)^T \Rightarrow \beta = [\underbrace{y^T X}_{A} \underbrace{(X^T X)^{-1}}_{B}]^T$$

$$\Rightarrow B^T \cdot A^T$$

$$\beta = [(X^T X)^{-1}]^T (y^T X)^T$$

$$\beta = [(X^T X)^{-1}]^T (X^T y)$$

Now, to prove $[(X^T X)^{-1}]^T$ is symmetric i.e.

$$[\underbrace{(X^T X)}_{\text{Let } A}{}^{-1}]^T = X^T X^{-1}$$

$X^T X = A$

$AA^{-1} = I$

$(AA^{-1})^T = I^T$

$(A^{-1})^T (A)^T = I$

$(X^T X)^T = X^T X$

$A^T A = A$ (here)

$(A^{-1})^T A = I$

↓ mutiply both side A^{-1}

$(A^{-1})^T A A^{-1} = I A^{-1}$

↓

$(A^{-1})^T \cdot I = A^{-1}$

hence, $[(X^T X)^{-1}]^T$ is a symmetric matrix

Code implementation :

load the diabetes dataset and unpack info X and y Da[illegible]

```
import numpy as np
from sklearn.datasets import load_diabetes

data = load_diabetes()
X, y = data.data, data.target
```

X → array([[0.380 0.050 - - -
- - - - - -
]])

2D array

X.shape → (442, 10)

y → array([ ]) y.shape → (442,)

1D array

Using sklearns Linear regression :

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split
(X, y, test_size=0.2, random_state=2)
```

X_train.shape → (353, 10)
X_test.shape → (89, 10)

```
from sklearn.linear_model import LinearRegression

reg = LinearRegression()
reg.fit(X_train, y_train)

y_pred = reg.predict(X_test)

from sklearn.metrics import r2_score

r2_score(y_test, y_pred)
```

→ 0.4399

reg.coef_ → array([-9.1[illegible] -205.[illegible] 516.[illegible] [illegible]
- - -
])

reg.intercept_ → 151.8831

Mathematical implementation of MLR :

```
class myLR:
    def __init__(self):
        self.coef_ = None
        self.intercept_ = None

    def fit(self, X_train, y_train):
        X_train = np.insert(X_train, 0, 1, axis=1)
```

your original array

The index before which the values to inserted

The value to be inserted

specifies the [illegible] [illegible] which means inserting a new col

X_train.shape

$(m, n) \xrightarrow[\text{insert}]{\text{after}} (m, n+1)$

```
        betas = np.linalg.inv(np.dot(X_train.T,
            X_train)).dot(X_train.T).dot(y_train)

        self.intercept_ = betas[0]
        self.coef_ = betas[1:]
```

```
def predict(self, X_test):
    y_pred = np.dot(X_test, self.coef_)
             + self.intercept_
    return y_pred.
```

```
lr = myLR()

lr.fit(X_train, y_train)

X_train.shape -> (353, 10)

np.insert(X_train, 0, 1, axis=1).shape
  -> (353, 11) -> check for multiple matrix.

y_pred = lr.pred(X_test)

r2_score(y_test, y_pred)
  -> 0.4399

lr.coef_        lr.intercept_
  -> array([ ])     -> 151.8833
```