Multiple Linear Regression

- We will use 3 parameters
- So, we won't draw a line in 3D plot
- We have to plot a plane
- The equation of a plane is:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2$$

x_1, x_2 → IQ, CGPA

β_1 → m (slope of x_1)

β_2 → m (slope of x_2)

$$\boxed{y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_n x_n}$$

hyperplane in n-dim coordinat

We will generate a dataset:

`from sklearn.datasets import make_regression`

```
from sklearn.datasets import make_regression
import pandas as pd
import numpy as np

import plotly.express as px
import plotly.graph_objects as go

from sklearn.metrics import mean_absolute_error,mean_squared_error,r2_scor
e
```

```
X,y = make_regression(n_samples=100, n_features=2, n_informative=2, n_targ
ets=1, noise=50)
```

1. `n_samples`: Number of samples in the dataset (default: 100).
2. `n_features`: Total number of features (default: 100).
3. `n_informative`: Number of features that are actually used to generate the target (default: 10). The rest are redundant or irrelevant.
4. `n_targets`: Number of target values (output dimensions) to generate (default: 1).
5. `noise`: Standard deviation of Gaussian noise added to the target (default: 0.0).

- Let's convert the above data into df

```
df = pd.DataFrame({'feature1':X[:,0],'feature2':X[:,1],'target':y})
```

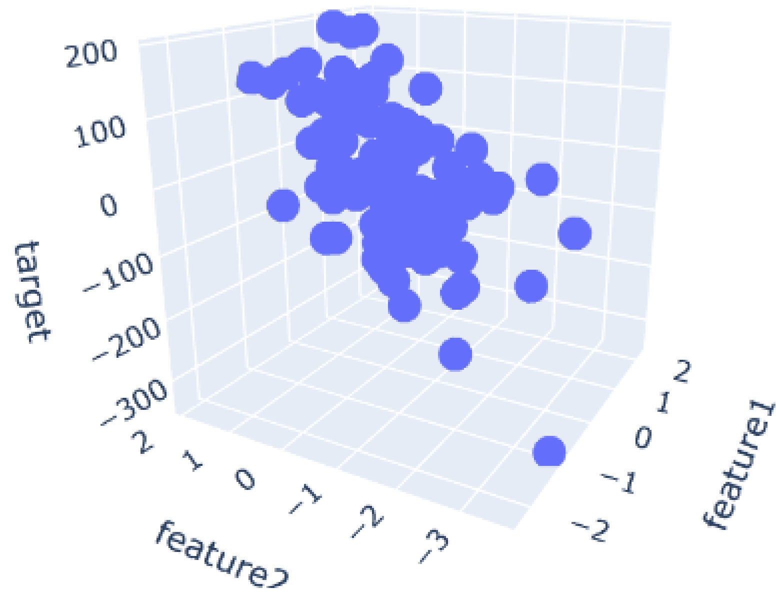
	feature1	feature2	target
0	-0.519228	0.944952	34.612196
1	-0.991890	0.140103	-90.297417
2	-0.518673	-0.839817	-112.088414
3	-0.322786	0.430100	-42.986151
4	-1.384753	-0.943044	-88.872789

```
df.shape

Output: (100,3)
```

```
fig = px.scatter_3d(df, x='feature1', y='feature2', z='target')

fig.show()
```



Split the data

```
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=
3)
```

- Same steps as simple LR

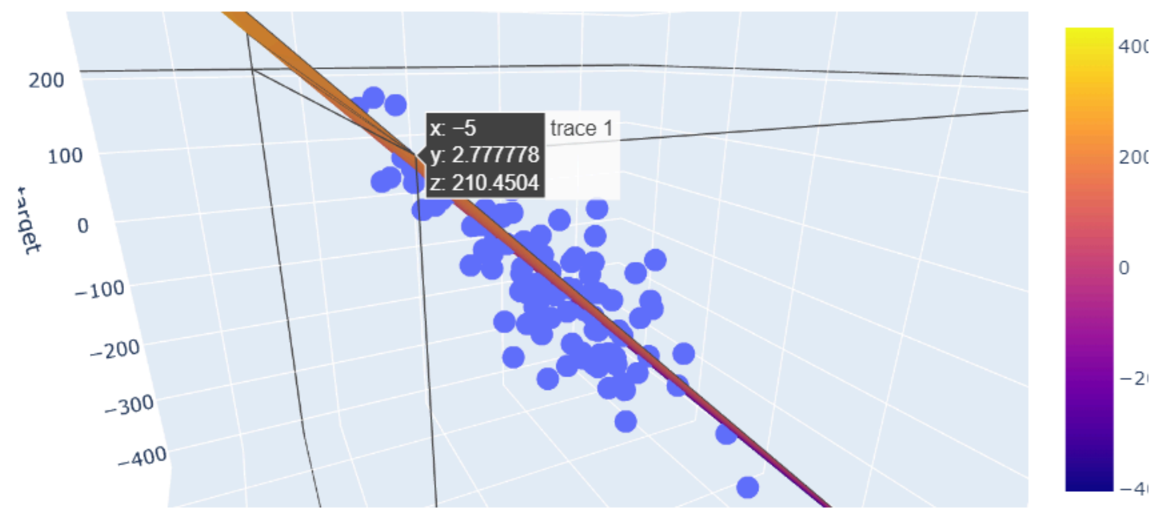
```
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(X_train,y_train)
y_pred = lr.predict(X_test)
```

```
print("MAE",mean_absolute_error(y_test,y_pred))
print("MSE",mean_squared_error(y_test,y_pred))
print("R2 score",r2_score(y_test,y_pred))
```

Output:

MAE 34.55516867926663
MSE 2084.3410603035263
R2 score 0.8458983233769555

- **LR is plotting a plane**




```
lr.coef_

Output: array([59.64518074, 13.20409431])
```

- We are getting 2 values
- These are β_1 **&** β_2

```
lr.intercept_

Output: -7.55491251398082
```

β_0👆 (Z- intercept)

Mathematical Formula

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2$$

For 2 Columns:

$$\hat{y}_1 = \beta_0 + \beta_1 x_{11} + \beta_2 x_{12}$$

$$\hat{y}_2 = \beta_0 + \beta_1 x_{21} + \beta_2 x_{22}$$

$$\hat{y}_3 = \beta_0 + \beta_1 x_{31} + \beta_2 x_{3}$$

For n columns:

$$\hat{y}_1 = \beta_0 + \beta_1 x_{11} + \beta_2 x_{12} + \beta_3 x_{13} + \beta_4 x_{14} + \dots + \beta_m x_{1m}$$

$$\hat{y}_2 = \beta_0 + \beta_1 x_{21} + \beta_2 x_{22} + \dots + \beta_m x_{2m}$$

$$\hat{y}_3 = \beta_0 + \beta_1 x_{31} + \beta_2 x_{32} + \dots + \beta_m x_{3m}$$

$$\hat{y}_4$$

$$\vdots$$

$$\hat{y}_n = \beta_0 + \beta_1 x_{n1} + \beta_2 x_{n2} + \dots + \beta_m x_{nm}$$

$$= \begin{bmatrix}\hat{y}_1\\ \hat{y}_2\\ \vdots \\ \hat{y}_n\end{bmatrix} = \begin{bmatrix} \beta_0+\beta_1 x_{11}+\beta_2 x_{12}+\beta_3 x_{13}+\beta_4 x_{14}+\dots+\beta_m x_{1m}\\ \beta_0+\beta_1 x_{21}+\beta_2 x_{22}+\dots+\beta_m x_{2m}\\ \beta_0+\beta_1 x_{31}+\beta_2 x_{32}+\dots+\beta_m x_{3m}\\ \vdots\\ \beta_0+\beta_1 x_{n1}+\beta_2 x_{n2}+\dots+\beta_m x_{nm}\end{bmatrix}$$

Shape of both is ***nx1***

- We can represent the matrix as dot product of 2 these matrices👇

$$\begin{bmatrix} 1 & x_{11} & x_{12} & \dots & x_{1m}\\ 1 & x_{21} & x_{22} & \dots & x_{2m}\\ \vdots & & & & \\ 1 & x_{n1} & x_{n2} & \dots & x_{nm}\end{bmatrix}\begin{bmatrix}\beta_0\\ \beta_1\\ \vdots\\ \beta_m\end{bmatrix}$$

From above, we get the equation:

$$\boxed{\hat{y} = X\beta}$$

Now let's find out the E:

$$\underline{y} = \underset{n \times 1}{\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}} \qquad \underline{\hat{y}} = \underset{n \times 1}{\begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_n \end{bmatrix}}$$

Difference between the actual value and predicted value:

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

Now, multiply this with its transpose:

$$e^T e = \begin{bmatrix} y_1 - \hat{y}_1 & y_2 - \hat{y}_2 & \cdots & y_n - \hat{y}_n \end{bmatrix}_{1 \times n} \begin{bmatrix} y_1 - \hat{y}_1 \\ y_2 - \hat{y}_2 \\ \vdots \\ y_n - \hat{y}_n \end{bmatrix}$$

$$1 \times 1$$

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

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

$$\boxed{E = e^T e}$$

We have to minimize this👆

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

$$E = y^Ty - y^T\hat{y} - \hat{y}^Ty + \hat{y}^T\hat{y}$$

Both are same

$$\boxed{E = y^Ty - 2y^T\hat{y} + \hat{y}^T\hat{y}} \rightarrow \text{eq}^{n}\ 3$$

$$\hat{y} = X\beta$$

$$y = f(x) \rightarrow x$$

- Now replace $\hat{y}$ with Xß

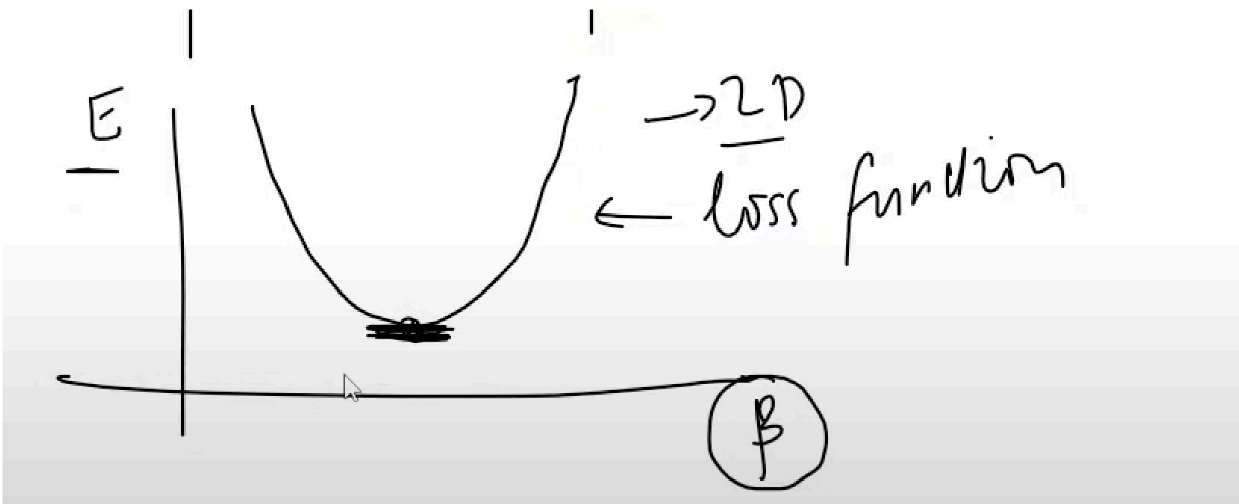
$$E = y^Ty - 2y^TX\beta + (X\beta)'(X\beta)$$

$$\boxed{E = y^Ty - 2y^TX\beta + \beta^TX^TX\beta} \quad \text{eq}^{n}\ 4$$

- E is function of ß
  - When you make any change in ß, E will change
- Why not function of x & y?
  - Because x → Input

- y → Output
- Both will not change

We need to find such value of ß matrix, that E will be minimum.



- We want slope=0
- We have to differentiate the loss function and equate it with 0

$$\frac{dE}{d\beta} = 0$$

- After complex calculations, we got

$$\boxed{\beta = (X^T X)^{-1} X^T y}$$ eq^{n} <u>5</u>

- **This is called OLS👆**