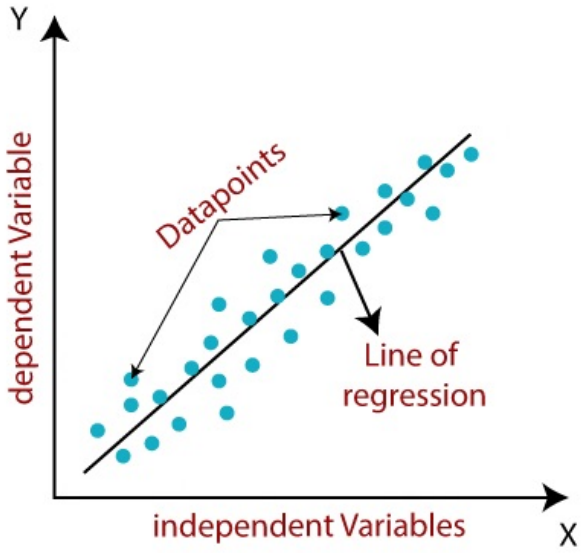
Linear Regression

Linear Regression helps us to find linear relation between dependent (Y) and independent variable(s).

⇒ Types of Linear Regression ⇒

① Simple Linear Regression
② Multiple Linear Regression
③ Polynomial Linear Regression

How it works?



$y = mx + c$

Where

y = dependent variable

x = Independ variables

m = slop (θ)

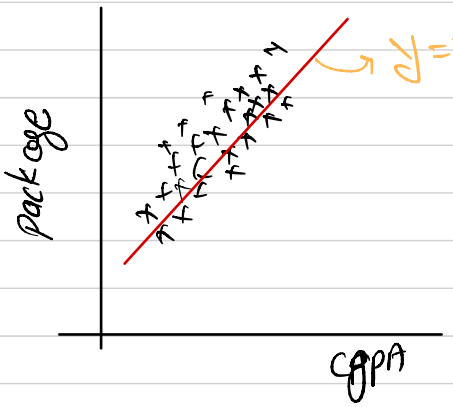
c = y-intercept

first we have to find best fit line.

⇓

line having minimum error

Mathematical Foundation



$y = mx + b$ ⇒ have to find m, b

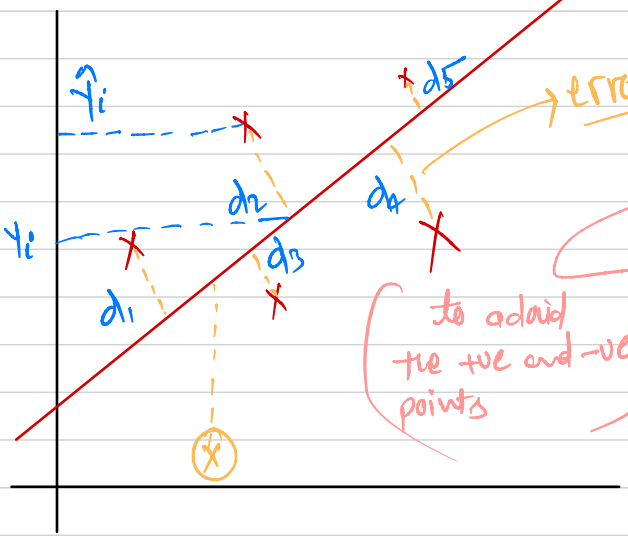
- closed form solution (formula base) ⇓ OLS [ordinary least square] ⇐ (Not good for height Dim.)
- Non-closed form solution (approximation base) [Gradient decent] ⇓ (good for height Dim)

Closed form Solution

$$b = \bar{y} - m\bar{x}$$

$$m = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sum (x_i - \bar{x})^2}$$

($\bar{x}, \bar{y}$ ⇒ mean of variables) ⇒ sklearn - Linear Regression



regression line ⇒ (find that line having minimum error to all the points)

error

$E = d_1 + d_2 + d_3 + \cdots + d_n$

$E = d_1^2 + d_2^2 + d_3^2 + \cdots + d_n^2$

(to avoid the +ve and -ve points)

$E = |d_1| + |d_2| + \cdots |d_n|$ ✗

↳ we are not using mod because of two reasons ⇒

① penalizing the outliers

② have to perform differentiation, mode graph is not diff at zero.

$$E = \sum_{i=1}^{n} d_i^2$$ → Error function (J)

Now, we need that m, b which can reduce E (error).

$\therefore \left(d_i = y_i - \hat{y}_i\right)$

$$E = \sum_{i=1}^{n} \left(y_i - \hat{y}_i\right)^2$$ ⇒ total error

$\therefore \left(\hat{y}_i = mx_i + b\right)$

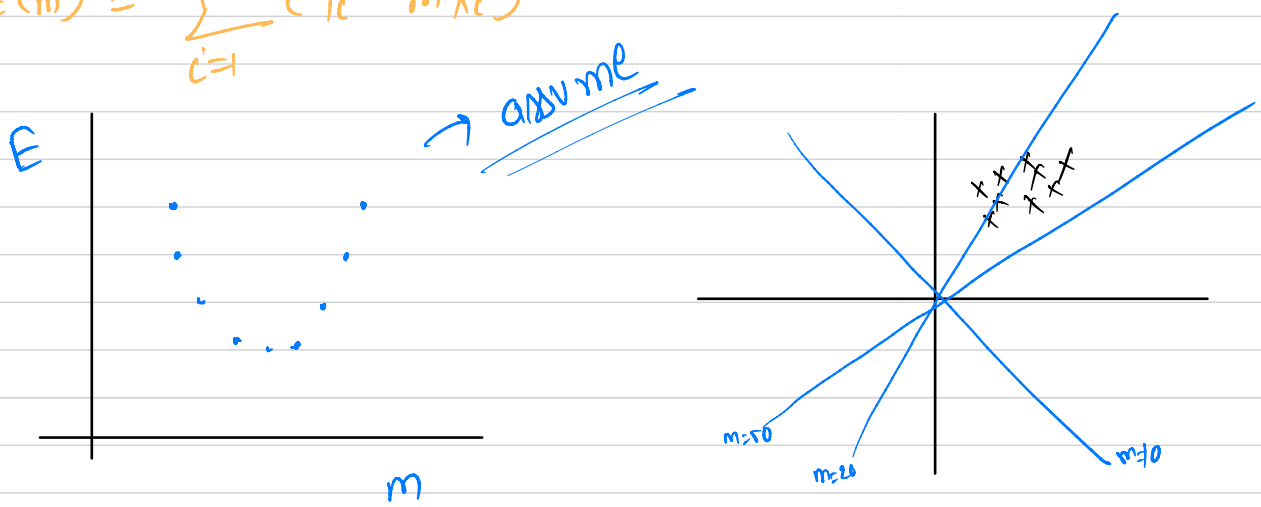
$$E(m, b) = \sum_{i=1}^{n} \left(y_i - m x_i - b\right)^2$$

[y_i, x_i will be same for a point, while we can change m, b]

Value of E can be varied if we makes change in m, b.

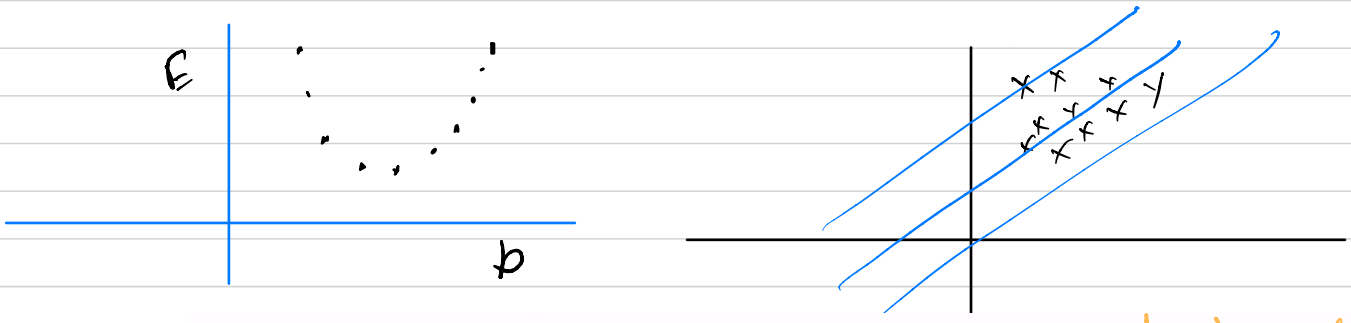
Assume $b = 0$

$$E(m) = \sum_{i=1}^{n} (y_i - m x_i)^2$$

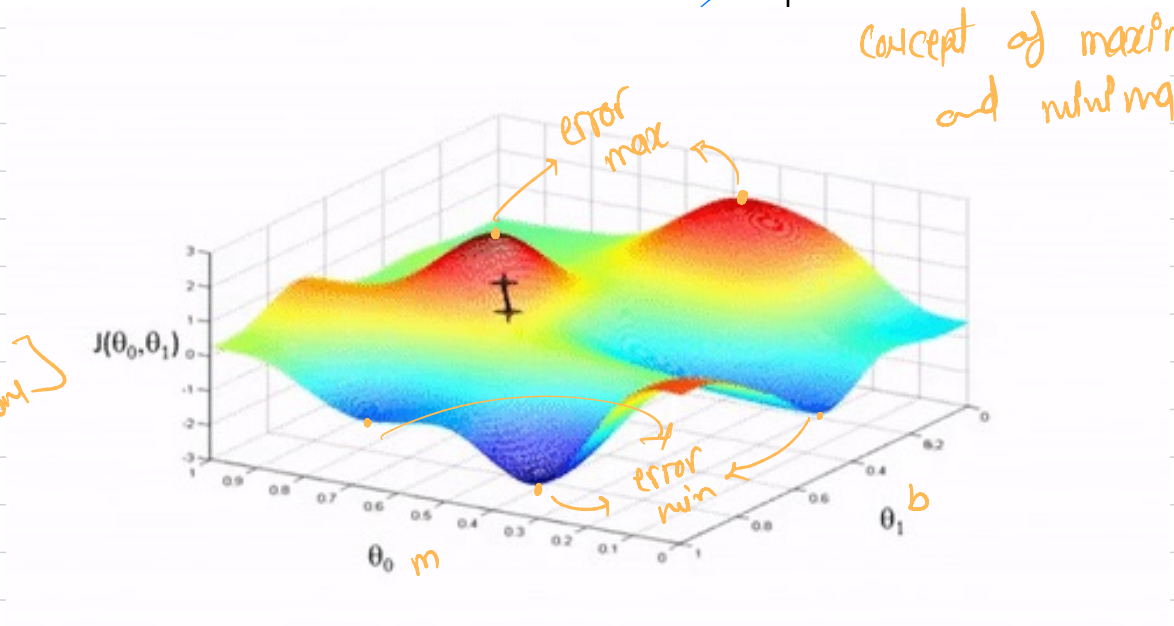


Assume $m = 0$

$$E(b) = \sum_{i=1}^{n} (y_i - x_i - b)^2$$



Concept of maxima and minima



find the derivation of error function with respect to both m, and zero

$$\frac{\partial E}{\partial B} = \frac{\partial}{\partial B}\left(\sum_{i=1}^{n}(y_i - mx_i - b)^2\right) = 0$$

Chain Rule

$$= \sum \frac{\partial}{\partial B}(y_i - mx_i - b)^2 = 0$$

$$= \sum -2(y_i - mx_i - b) = 0$$

$$\Rightarrow \sum y_i - \sum mx_i - \sum b = 0$$

$$\Rightarrow \frac{\sum y_i}{n} - m\frac{\sum x_i}{n} - \frac{\sum b}{n} = 0$$

mean of y

$\because \sum b = b+b+b+\cdots = nb$

$$\Rightarrow \bar{y} - m\bar{x} - \frac{nb}{n} = 0$$

$$\Rightarrow \bar{y} - m\bar{x} = b$$

$$\boxed{b = \bar{y} - m\bar{x}}$$

$$E = \sum (y_i - mx_i - \bar{y} + m\bar{x})^2 = 0$$

$$\frac{\partial E}{\partial m} = \sum \frac{\partial}{\partial m}(y_i - mx_i - \bar{y} + m\bar{x})^2 = 0 \quad \text{(Chain Rule)}$$

$$\Rightarrow \sum 2(y_i - mx_i - \bar{y} + m\bar{x})(-x_i + \bar{x}) = 0$$

$$\Rightarrow \sum -2\left(y_i - mx_i - \bar{y} + m\bar{x}\right)\left(x_i - \bar{x}\right) = 0$$

$$\Rightarrow \sum \left(y_i - mx_i - \bar{y} + m\bar{x}\right)\left(x_i - \bar{x}\right) = 0$$

$$\Rightarrow \sum \left(y_i - \bar{y}\right) - m\left(x_i - \bar{x}\right)\Big]\left(x_i - \bar{x}\right) = 0$$

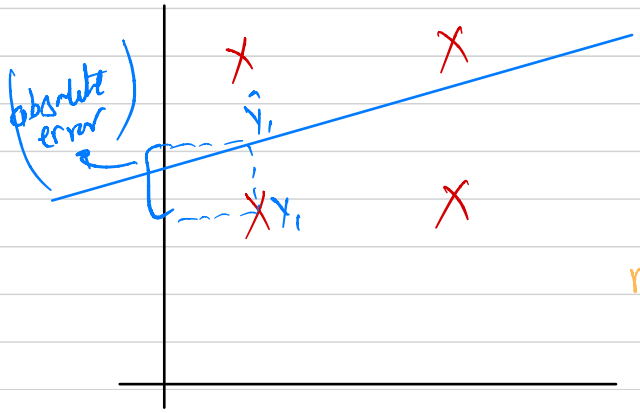
$$\Rightarrow \sum\left[\left(y_i - \bar{y}\right)\left(x_i - \bar{x}\right) - m\left(x_i - \bar{x}\right)^2\right) = 0$$

$$m = \frac{\sum_{i=1}^{n}\left(x_i - \bar{x}\right)\left(y_i - \bar{y}\right)}{\sum\left(x_i - \bar{x}\right)^2}$$

Performance of the Model

1. MAE (mean absolute error)
2. MSE (mean square error)
3. RMSE (Root ———) } loss/error functions
4. R2 score (Coefficient of determination)
5. Adjusted R2 score } performance of models

1. MAE (Mean Absolute Error)



$$= \frac{|y_1 - \hat{y}_1| + |y_2 - \hat{y}_2| + \cdots |y_n - \hat{y}_n|}{n}$$

$$mae = \frac{\sum_{i=1}^{n}(y_n - \hat{y}_n)}{n} \Rightarrow \text{loss}$$

Advantage

① Same unit
② Robust to outlier

Disadvantage

① as we are using mode, and as we know that diff. of mode is not possible at zero.

② Mean Squared error ⇒

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

Advantage

① can used as loss function because it is diff

Disadvantage

② unit will be in square
③ not good with the outlier (Not robust will outlier)

③ RMSE ⇒

$$RMSE = \sqrt{mse}$$

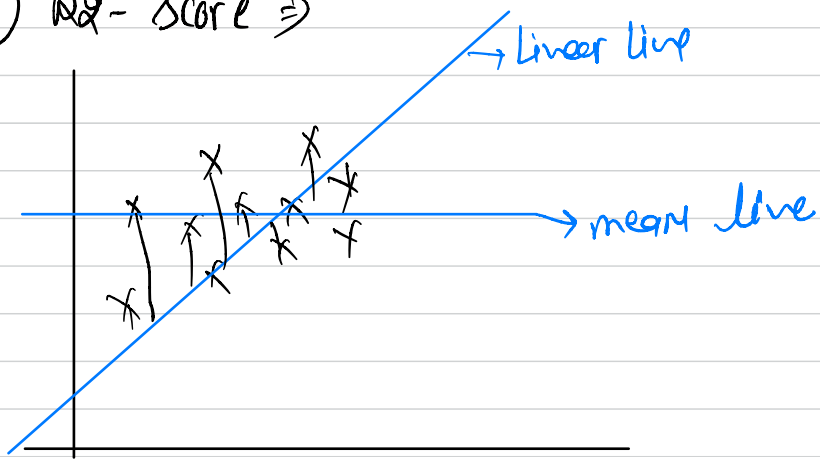
Advantage

① Same unit

Disadvantage.

② Not Robust with outliers

④ R2 - score ⇒



$$R2_{eqn} = 1 - \frac{SSR}{SSm}$$

$$= 1 - \frac{\left[\sum_{i=1}^{n} (y_i - \hat{y}_i)^2\right]_{Reg}}{\left[\sum_{i=1}^{n} (y_i - \bar{y}_i)^2\right]_m}$$

if R2 is zero ⇒ perforance of model is not Good.

it mean

$$\frac{\left[\sum_{i=1}^{n} (y_i - \hat{y}_i)^2\right]_{Reg}}{\left[\sum_{i=1}^{n} (y_i - \bar{y}_i)^2\right]_m} = 1$$

if R2 is One ⇒ best Perforance of the model

$$\frac{\left[\sum_{i=1}^{n} (y_i - \hat{y}_i)^2\right]_{Reg}}{\left[\sum_{i=1}^{n} (y_i - \bar{y}_i)^2\right]_m} = 0$$

it happen only when $\left[\sum_{i=1}^{n} (y_i - \bar{y}_i)^2\right]_m = 0$, it mean no-error on regression line which is not possible.

What is R2 is negative ⇒ $(SSR) > (SSm)$

rare happen

$(R^2 = .80)$ cgpa / lpa $\Rightarrow$ mean of it is $\Rightarrow$ [cgpa can explain 80% of variance in lpa.]

⑤ Adjusted R^2 score $\Rightarrow$

$$\text{Adjusted } R^2 \text{ score} = 1 - \left[\frac{(1-R^2)(n-1)}{(n-1-k)}\right]$$

n = No of row

k = No of feature

① if we add irrelevant column $\Rightarrow$ (temp)

Adj R^2 ↓

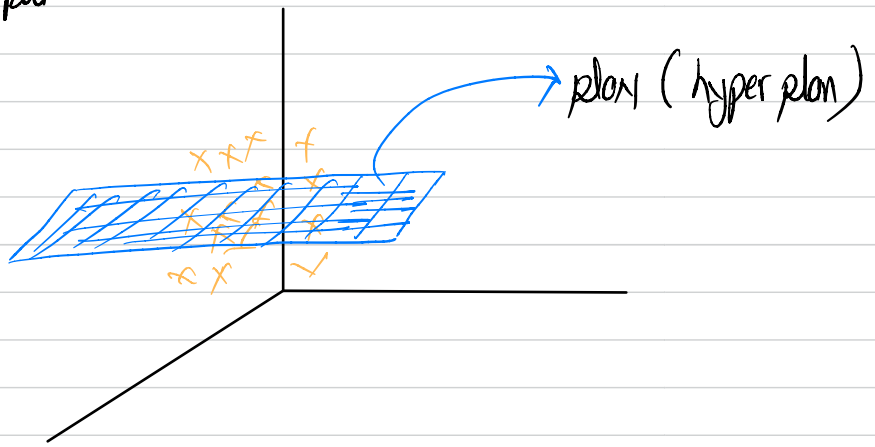
② if we add (iq)

Adj R^2 ↑

Multiple Linear Regression

When we have more than 1 input feature is there.

x_1	x_2	output



$$y = mx_1 + nx_2 + b$$

— or —

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

→ $\beta_0, \beta_1, \beta_2$] ⇒ find

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3$$

→ $\beta_0, \beta_1, \beta_2, \beta_3$

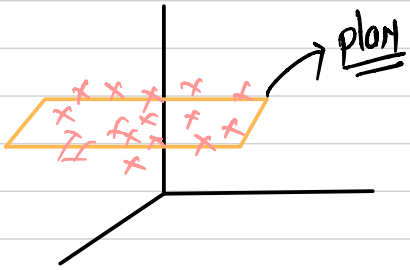
$$\boxed{y = \beta_0 + \sum_{i=1}^{n} \beta_n x_n}$$

values of β_1 and β_2 shows that which column have more weight to find the o/p.,

β_0 = offset, if $\beta_1 = \beta_2 = 0$

have to find the equation of hyper plan.

Mathematics behind MLR ⇒



as our equation $\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_N x_N$ — (1)

data :- $x_1 \mid x_2 \mid x_3 \mid y$

Suppose we have 50 row then shape ⇒ (50,4)

Now, we have to find predicted value of each raw

$$\hat{Y} = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \hat{y}_3 \\ \vdots \\ \hat{y}_{50} \end{bmatrix} = \begin{bmatrix} \beta_0 & \beta_1 x_{11} & \beta_2 x_{12} & \beta_3 x_{13} \\ \beta_0 & \beta_1 x_{21} & \beta_2 x_{22} & \beta_3 x_{23} \\ \vdots & & & \\ \beta_0 & \beta_1 x_{50,1} & \beta_2 x_{50,2} & \beta_3 x_{50,3} \end{bmatrix}$$

after converting eq (1) into matrix.

assume I have n rows and m columns

$$\hat{Y} = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_n \end{bmatrix} = \begin{bmatrix} \beta_0 & \beta_1 x_{11} & \beta_2 x_{12} & \cdots & \beta_m x_{1m} \\ \beta_0 & \beta_1 x_{21} & \beta_2 x_{22} & \cdots & \beta_m x_{2m} \\ \vdots & \vdots & \vdots & & \vdots \\ \beta & \beta_1 x_{n1} & \beta_2 x_{n2} & \cdots & \beta_m x_{nm} \end{bmatrix}$$

$$\hat{Y} = \begin{bmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1m} \\ 1 & x_{21} & x_{22} & \cdots & x_{2m} \\ & \vdots & & & \\ 1 & x_{n1} & x_{n2} & & x_{nm} \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_n \end{bmatrix}$$

$= X$, $= \beta$

input

$$\hat{Y} = X\beta \quad \text{---} \quad (2)$$

predicted

co offficient

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

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

In Simple linear regression

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

loss function for multiple linear Regression

$$E = e^T e$$

$$= [(y_1 - \hat{y}_1) \quad (y_2 - \hat{y}_2) \cdots (y_n - \hat{y}_n)] \begin{bmatrix} (y_1 - \hat{y}_1) \\ (y_2 - \hat{y}_2) \\ \vdots \\ (y_n - \hat{y}_n) \end{bmatrix}$$

$(1 \times n)$ $\quad$ $(n \times 1)$

$$E = \sum_{i=1}^{n} (y_n - \hat{y}_n)^2$$

$$E = e^T \cdot e$$

$$= (Y - \hat{Y})^T (Y - \hat{Y})$$

$$= (Y^T - \hat{Y}^T)(Y - \hat{Y})$$

from eq ①

$$= [Y^T - (X\beta)^T](Y - X\beta)$$

$$= Y^TY - Y^TX\beta - (X\beta)^TY + (X\beta)^TX\beta$$

∴ as we know that

$$Y^TX\beta = (X\beta)^TY$$

$$E = Y^TY - 2Y^TX\beta + \beta^TX^TX\beta$$

$$\therefore \frac{dE}{d\beta} = 0 \quad \text{so}$$

$$\frac{d}{d\beta}[Y^TY - 2Y^TX\beta + \beta^TX^TX\beta] = 0$$

$$0 - 2Y^TX + \frac{d}{d\beta}(\beta^TX^TX\beta)$$

learn matrix differentiation

$$Y = A^TXA$$

$$\frac{dY}{dA} = 2XA^T$$

$$-2Y^TX + 2X^TX\beta^T = 0$$

$$X^TX\beta^T = Y^TX$$

$$\beta^T = \frac{Y^TX}{(X^TX)}$$

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

$$(\beta^T)^T = \left(Y^T X (X^T X)^{-1}\right)^T$$

$$\beta = \left[(X^T X)^{-1}\right]^T (Y^T X)^T$$

$$= \left[(X^T X)^{-1}\right]^T (X^T Y)$$

$\because \left((X^T X)^{-1}\right)^T = (X^T X)^{-1}$, because it is a square matrix

$$\boxed{\beta = (X^T X)^{-1} X^T Y}$$

Why Gradient Descent

⇒ OLA will find all the values from the coefficient.

⇒ because of $(X^T X)^{-1}$

read article on wiki ⇒ Computational complexity of mathematical operation

(matrix algebra)

Linear Regression

In [1]:
```
# import Libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
# read file
df = pd.read_csv('Salary_dataset.csv')
```

In [3]:
```
# drop first column
df = df[['YearsExperience', 'Salary']]
```

Basic EDA

drow a regression line that will show the relation between both columns, our 'YearsExperience' is input data and 'Salary' is output.

In [4]:
```
sns.lmplot(x='YearsExperience', y='Salary', data=df)

# Set axis labels and plot title
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.title('Relationship between Years of Experience and Salary')
```

Out[4]:
```
Text(0.5, 1.0, 'Relationship between Years of Experience and Salary')
```

In [5]:
```
X = df[['YearsExperience']]
y = df[['Salary']]
```

In [6]:
```
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)
```

In [7]:
```
# using sklearn library to apply model
from sklearn.linear_model import LinearRegression
```

In [8]:
```
lr = LinearRegression()
```

In [9]:
```
lr.fit(X_train,y_train)
```

Out[9]:
```
▾ LinearRegression
LinearRegression()
```

In [10]:
```
y_pred = lr.predict(X_test)
```

In [11]:
```
y_pred
```

Out[11]:
```
array([[36834.63210301],
       [34920.71472592],
       [67457.3101364 ],
       [59801.64062805],
       [92338.23603852],
       [81811.69046455]])
```

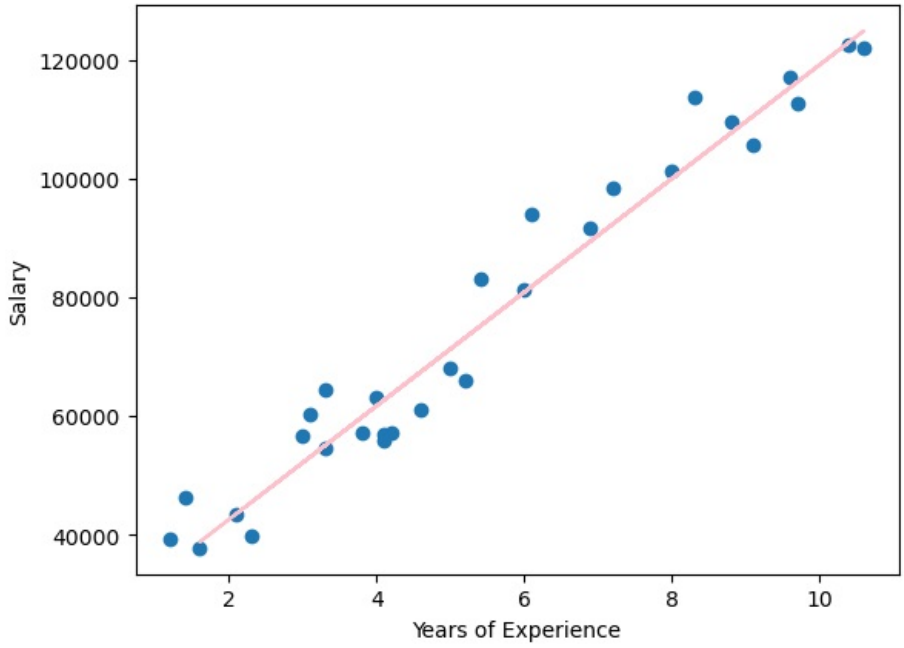
In [12]:
```
y_test
```

Out[12]:

	Salary
1	46206.0
0	39344.0
14	61112.0
9	57190.0
21	98274.0
19	93941.0

In [13]:
```
plt.scatter(X,y)
plt.plot(X_train,lr.predict(X_train),color='pink')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
```

Out[13]:
```
Text(0, 0.5, 'Salary')
```



In [14]:
```
b1 = lr.coef_
print('Slop :- ', b1)
```

```
Slop :-  [[9569.58688543]]
```

In [15]:
```
b0 = lr.intercept_
print('Intercept :- ', b0)
```

```
Intercept :-  [23437.21046341]
```

In [16]:
```
# our equaltion is y_pred = b0 + (b1*x)
```

In [17]:
```
# for x = 2.1 we have predict y
x = 2.1
y_x_pred = b0 + (b1*x)
y_x_pred
```

Out[17]:
```
array([[43533.34292281]])
```

Sketch code of Simple Linear Regression

Now, we have to find the best value for b0 and b1, for that we have the formula for both b0 and b1

In [18]:
```
class Linearsketch:
    def __init__(self):
        self.b0 = None
```

```
        self.b1 = None

    def fit(self,X_train, y_train):

        num = 0
        den = 0
        for i in range(X_train.shape[0]):
            num = num + ((X_train.values[i] - X_train.values.mean())*(y_train.values[i] - y_train.values.mean()
            den = den + ((X_train.values[i] - X_train.values.mean())*(X_train.values[i] - X_train.values.mean()

        self.b1 = num/den
        self.b0 = y_train.values.mean() - (self.b1 * X_train.values.mean())

        print(self.b0,self.b1)

    def predict(self, X_test):
        return self.b0 * X_test.values + self.b1
```

In [19]:
```
lr1 = Linearsketch()
lr1.fit(X_train,y_train)
```

```
[23437.21046341] [9569.58688543]
```

In [20]:
```
lr1.predict(X_test)
```

Out[20]:
```
array([[ 42381.6815342 ],
       [ 37694.23944152],
       [117380.7550171 ],
       [ 98630.98664637],
       [178317.50222195],
       [152536.5707122 ]])
```

In [21]:
```
print('Value of b0, b1 from sklearn.linear_model :- ', b0,b1)
print('Value of b0, b1 from our linear mode  :- [23437.21046341] [9569.58688543]]')
```

```
Value of b0, b1 from sklearn.linear_model :-  [23437.21046341] [[9569.58688543]]
Value of b0, b1 from our linear mode  :- [23437.21046341] [9569.58688543]]
```

In [22]:
```
from sklearn.metrics import mean_absolute_error,mean_squared_error,r2_score
```

In [23]:
```
print("R2 score",r2_score(y_test,y_pred))
r2 = r2_score(y_test,y_pred)
```

```
R2 score 0.8886956733784561
```

In [24]:
```
print("MAE",mean_absolute_error(y_test,y_pred))
```

```
MAE 6802.779572073905
```

In [25]:
```
print("MSE",mean_squared_error(y_test,y_pred))
```

```
MSE 56137509.99782566
```

In [26]:
```
print("RMSE",np.sqrt(mean_squared_error(y_test,y_pred)))
```

```
RMSE 7492.49691343451
```

In [ ]:
```

```

MultiLinear Regression

In [27]:
```
df = pd.read_csv('advertising.csv.xls')
```

In [28]:
```
df
```

Out[28]:

	TV	Radio	Newspaper	Sales
0	230.1	37.8	69.2	22.1
1	44.5	39.3	45.1	10.4
2	17.2	45.9	69.3	12.0
3	151.5	41.3	58.5	16.5
4	180.8	10.8	58.4	17.9
...	...	...	...	...
195	38.2	3.7	13.8	7.6
196	94.2	4.9	8.1	14.0
197	177.0	9.3	6.4	14.8
198	283.6	42.0	66.2	25.5
199	232.1	8.6	8.7	18.4

200 rows × 4 columns

In [29]:
```
X = df[['TV','Radio','Newspaper']]
y = df[['Sales']]
```

In [30]:
```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=2)
```

In [31]:
```
lr1 = LinearRegression()
lr1.fit(X_train,y_train)
y_predt1 = lr1.predict(X_test)
```

In [32]:
```
r2_score(y_test,y_predt1)
```

Out[32]:
```
0.8407131803267819
```

In [33]:
```
lr1.coef_
```

Out[33]:
```
array([[0.05536768, 0.102883  , 0.00233839]])
```

In [34]:
```
lr1.intercept_
```

Out[34]:
```
array([4.49983137])
```

Sketch code of Multiple Linear Regression

In [35]:
```
class MultipleLinearRegression:

    def __init__(self):
        self.coef_ = None
        self.intercept_ = None

    def fit(self,X_train,y_train):
        X_train = X_train.to_numpy()
        y_train = y_train.to_numpy()

        X_train = np.insert(X_train,0,1,axis=1)

        # calcuate the coeffs
        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 = MultipleLinearRegression()
lr.fit(X_train,y_train)
```

In [36]:
```
y_pred = lr.predict(X_test)
y_pred
```

Out[36]:
```
array([[15.81794247],
       [10.15032354],
       [ 8.26737799],
       [18.25580684],
       [18.06466407],
       [17.24354571],
       [ 8.81294515],
       [22.04513076],
       [12.38208193],
       [21.03691149],
       [ 9.58287515],
       [20.13791992],
       [10.69692702],
       [ 9.05217041],
       [17.19403767],
       [10.33653542],
       [ 8.63490536],
       [17.27389797],
       [18.24930692],
       [19.29279069],
       [18.94026036],
       [19.46168254],
       [11.11298733],
       [10.6675807 ],
       [18.0316489 ],
       [14.58414874],
       [16.58323824],
       [ 9.27911751],
       [18.82160438],
       [17.18558892],
       [20.11211809],
       [16.22950562],
       [15.87936272],
       [14.05507975],
       [ 7.89699942],
       [12.12455603],
       [21.74608041],
       [21.4076981 ],
       [19.58225049],
       [19.50042709]])
```

In [37]:
```
r2_score(y_test,y_pred)
```

Out[37]:
```
0.8407131803267822
```

In [38]:
```
lr.coef_
```

Out[38]:
```
array([[0.05536768],
       [0.102883  ],
       [0.00233839]])
```

In [39]:
```
lr.intercept_
```

Out[39]:
```
array([4.49983137])
```