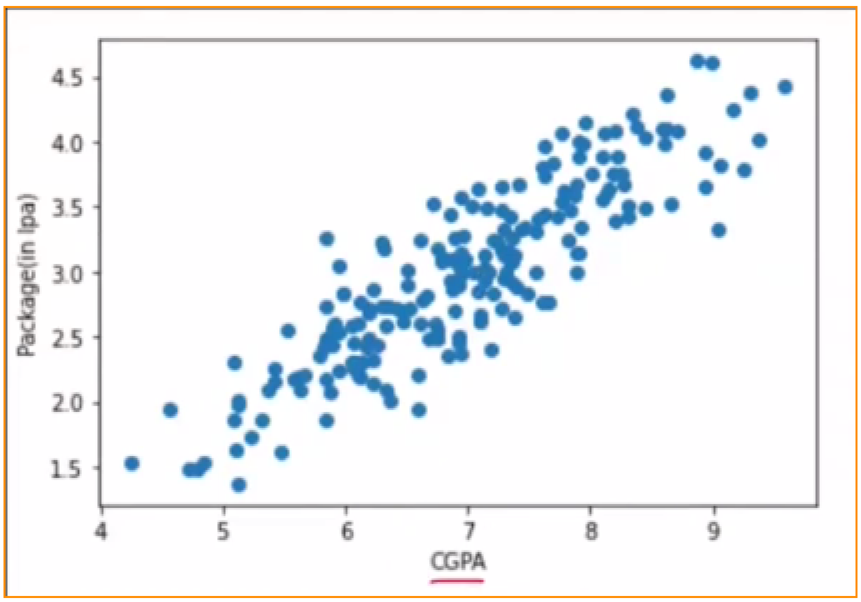
What is Linear Regression - Geometric Intuition

- SLR is a supervised machine learning algorithm
- Linear Regression generally works when data is linear means it has linear relationship
- There are 3 types:
    1. SLR - When there are only 1 input & 1 output column
    2. MLR - When there are multiple input columns
    3. Polynomial LR - When data is not linear

- suppose we were asked a question: What is the avg package in your college?
- We can simply take the average of packages & quote that amount but it will not represent the true average package bcoz student from different branch & grade will have difference in thier package.
- lets say we have a data contains CGPA & Package so we can plot that data:



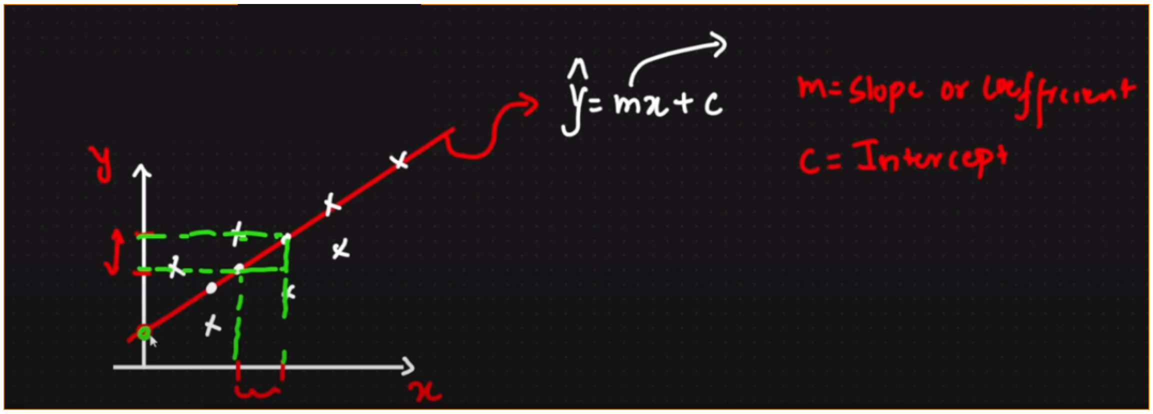
- The data shows a general trend, but it's not perfectly straight due to real-world factors we can't precisely measure, like how well someone performed in an interview or a company's urgent need to hire. These uncertainties are called stochastic errors which is why data is sort of linear.

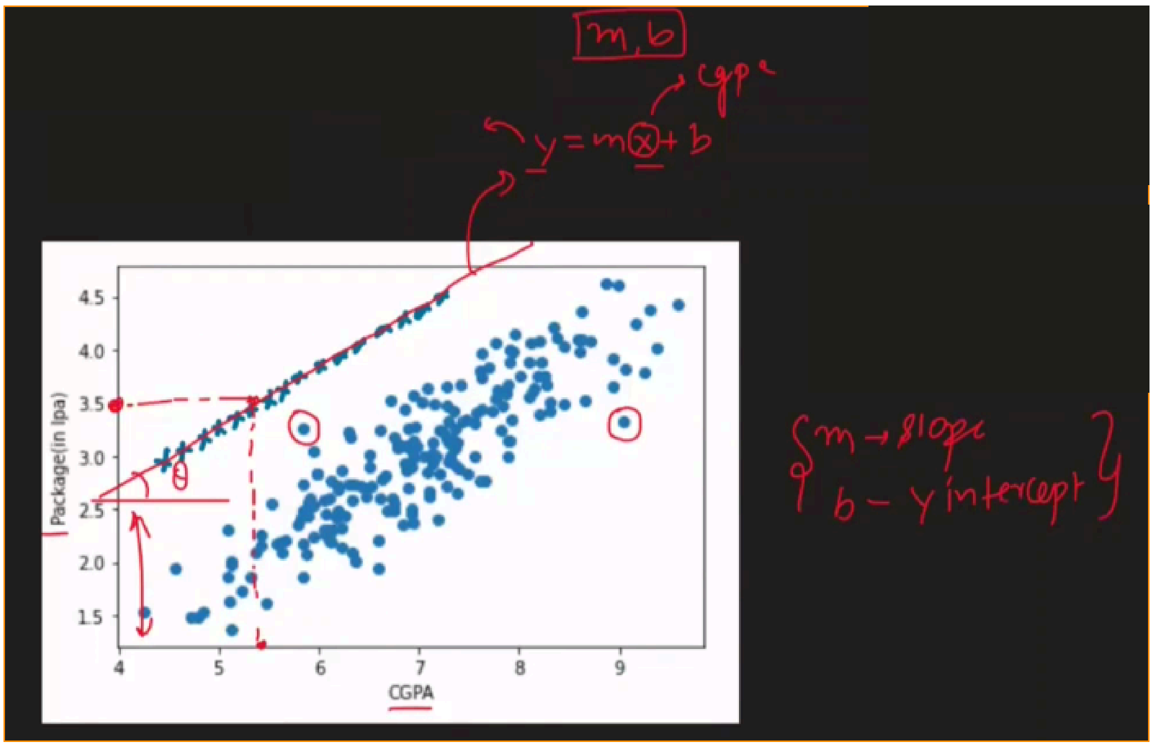
- If our data would have completly linear then we can simply plot a line using equation: y = mx + b

- where m is the θ (angle) & b is the y intercept

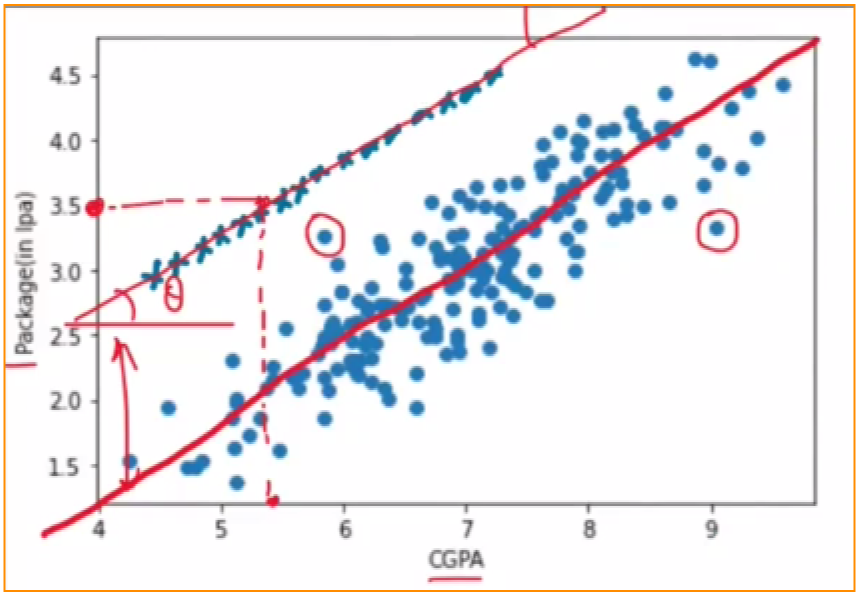
- 1 unit of change in x leads to how many units chage in y is "m" or theta

- "c" is the point where best fit line intercepting y when x == 0





- when new CGPA dta comes, we can simply join y to x axis & predict.. but we dont have completly linear data, we have sort of linear data
- we can still draw a line which is called "*Best Fit Line*"



- **Best fit** line bcoz it has minimum error or trying to pass very closley possible from every point, **Perfect line** is the one which passes through all pont.
- and Thats what **Linear Regression** does, It draws a best fit line on linear data means it calculates the value of those m & b for which line will pass very closely from all points.

Linear Regression Python Application

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

In [2]:
```
df = pd.read_csv("placement.csv")
```

In [3]:
```
df.sample(7)
```

Out[3]:

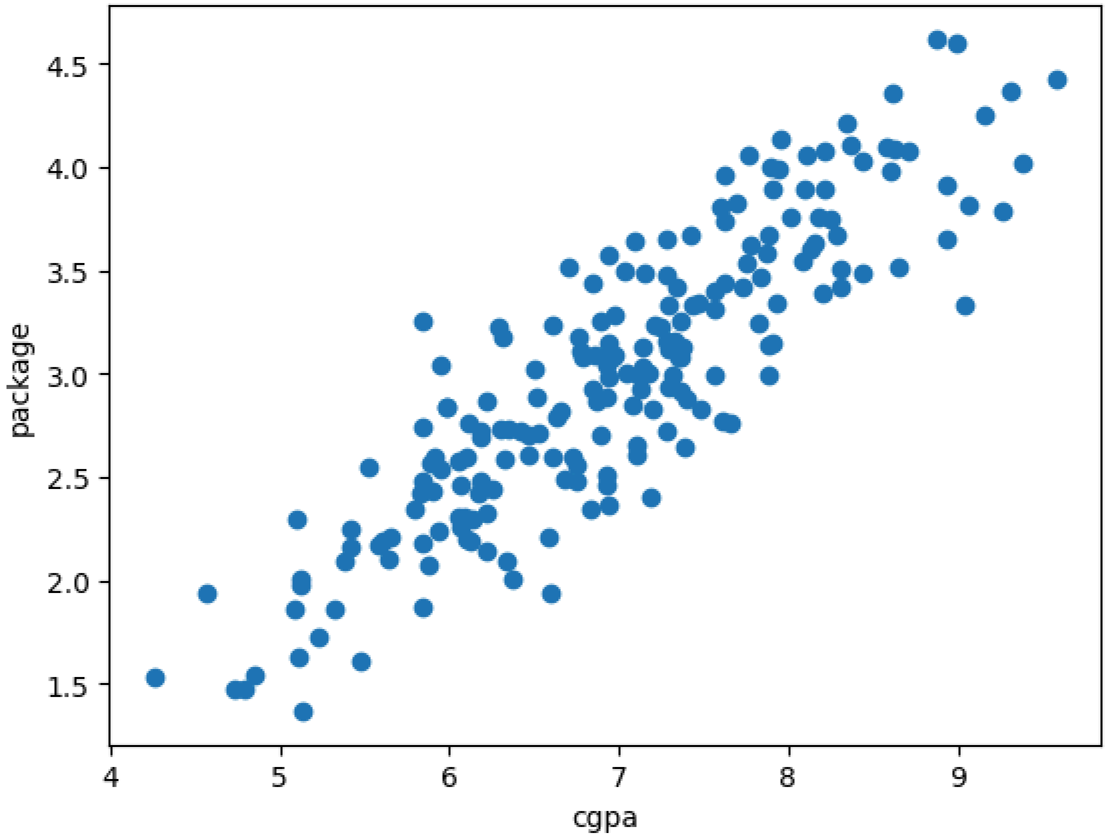
	cgpa	package
66	5.11	1.63
111	5.42	2.25
167	8.13	3.60
27	5.42	2.16
78	6.59	2.21
75	6.97	3.28
105	6.66	2.82

In [4]:
```
df.shape
```

Out[4]:
```
(200, 2)
```

In [5]:
```python
#we have 200 student data of cgpa & package..
```

In [6]:
```python
plt.scatter(df['cgpa'], df['package'])
plt.xlabel('cgpa')
plt.ylabel('package')
plt.show()
```



In [7]:
```python
#we can see that data is linear so we can apply linear regression model trained on th
#which will draw a best fit line and when given new cgpa value, it will predict packa
```

In [8]:
```python
X = df.iloc[:,0:1]
y = df.iloc[:,-1]
```

In [9]:
```python
#we divide data 2 parts 1 will be used fr training & other we can use to test
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
```

In [10]:
```python
#now we can train the model
from sklearn.linear_model import LinearRegression
```

In [11]:
```python
lr = LinearRegression()
```

In [12]:
```python
lr.fit(X_train,y_train)
```

Out[12]:

▾ LinearRegression

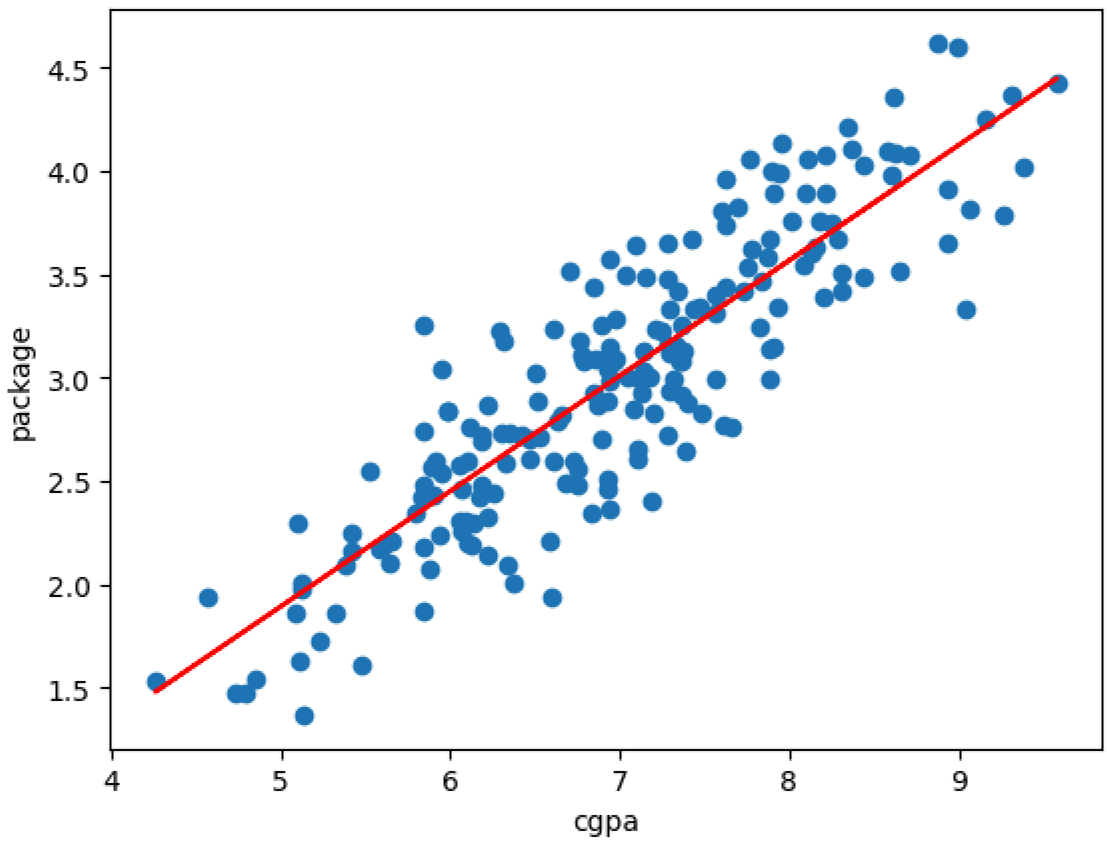
LinearRegression()

In [13]:
```
lr.predict(X_test.iloc[0].values.reshape(1,1))
```

```
C:\Users\iampr\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn\base.
py:493: UserWarning: X does not have valid feature names, but LinearRegression was fit
ted with feature names
  warnings.warn(
```

Out[13]:
```
array([3.89111601])
```

In [14]:
```
plt.scatter(df['cgpa'], df['package'])
plt.plot(X_train, lr.predict(X_train), color='red')
plt.xlabel('cgpa')
plt.ylabel('package')
plt.show()
```



In [15]:
```
#above is the best fit line which our lr model finds which also means it has find the
#best fit line is making least mistakes or passing very vlose to each point
```

In [16]:
```
m = lr.coef_
```

In [17]:
```
b = lr.intercept_
```

In [18]:
```
x = 8.58
y = (m * x) + b
```

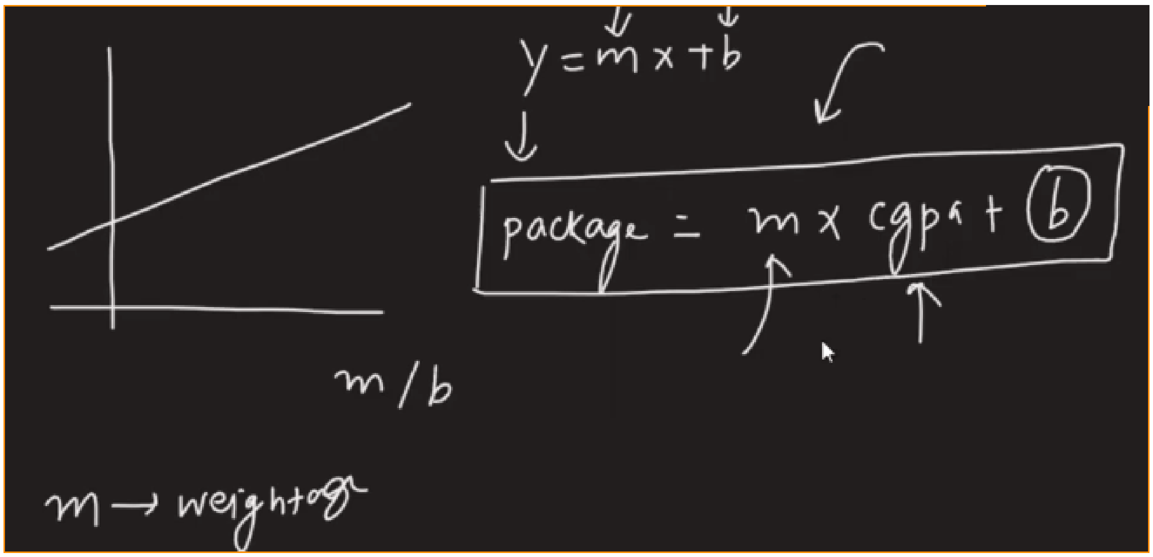
```
print(y)
```

[3.89111601]

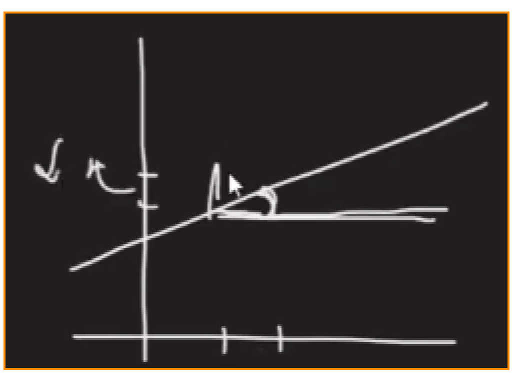
In [19]:
```
#even if we dont have that value of cgpa still our model will be able to predict
x = 100 #imaginery number
y = (m * x) + b
print(y)
```

[54.89908542]

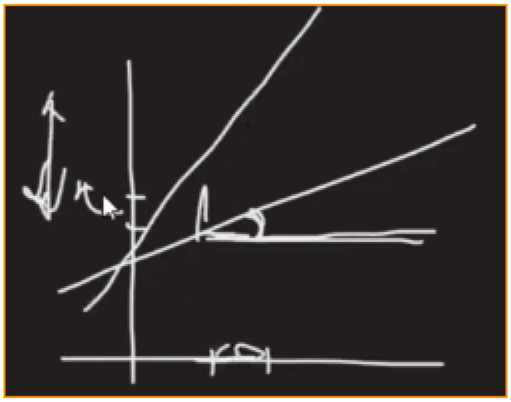
Further Understanding



- m is the weightage, how much cgpa depends on package, if value of m will be very less then package will be very less dependant on cgpa & vice versa
- means if value of m changes, change is package will be less since the theta value is very less



- but if slope is very high then less change in cgpa will have more change in package



- if we initialize the value of b as 0.. and suppose we get a student of experience = 0 (instead of cgpa for understanding) then package should be 0 however freshers also get some salary which is b. Its called Offset.

- There are 2 ways to find the value of m & b
    1. Closed form solution: It refers to an explicit and direct formula that provides the exact solution to a problem.
    2. Non closed form solution: In non closed form solution we use approximation techniques (calculas, diffrentiation) to reach the solution.
- **OLS method** is used to find the value of m & b using Closed form techniques.
- **Gradient Descent** is used to find the value of m & b using non closed form techniques like diffrentiation.
- Reason why we have 2 techniques is because of complexity.. we use OLS when we have less data & gradient descent when our data is huge.
- Linear Regression class of Scikit learn has OLS technique implemented & SGD regressor class has gradient descent implemented.

Regression Metrics

- We have multiple metrics bcoz each one have some advantages & disadvantages. Eac will be good on certain type of data

1. MAE
2. MSE
3. RMSE
4. r2 score
5. Adjusted r2 score

MAE

- MAE : The average absolute difference between predicted and actual values.
- It is a non-negative value, where lower MAE indicates better accuracy of predictions.

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

$$mae = \frac{\sum_{i=1}^{n} |y_i - \hat{y}_i|}{n}$$

Suppose we have actual values $y = [10, 20, 30, 40]$ and predicted values $\hat{y} = [12, 18, 28, 38]$.

1. Calculate the absolute differences:
   - $|10 - 12| = 2$
   - $|20 - 18| = 2$
   - $|30 - 28| = 2$
   - $|40 - 38| = 2$
2. Sum these absolute differences:
   $2 + 2 + 2 + 2 = 8$
3. Calculate the MAE:
   $\text{MAE} = \frac{1}{4} \times 8 = 2$

Therefore, the MAE for this example is 2.

In [20]:
```
#advantage
```

```
# MAE is the loss.. we'll get a number & our goal is to minimize the number
# MAE has the same unit as output column
# Robust to outliers
```

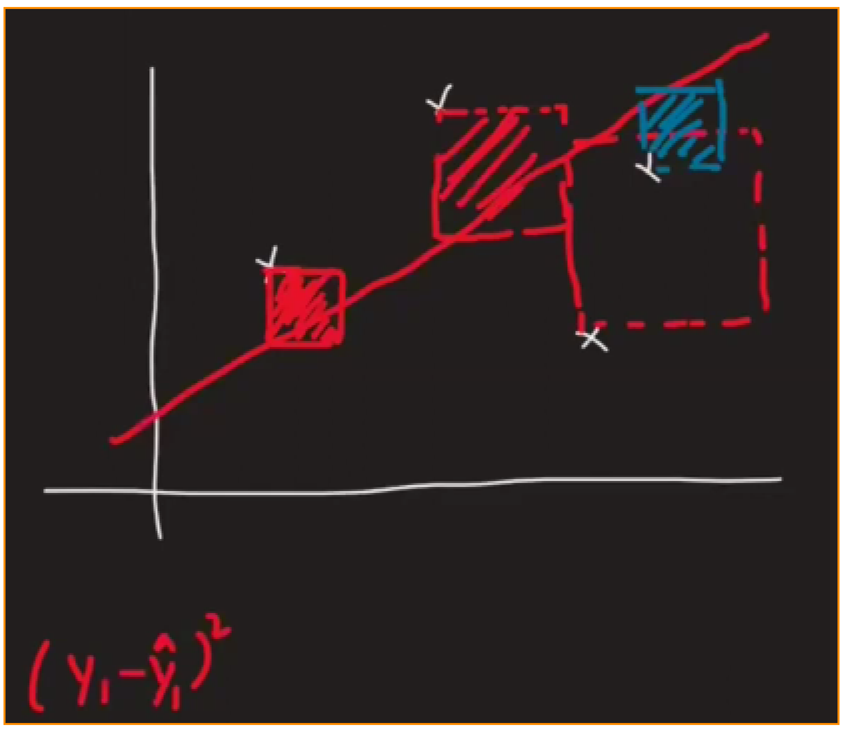
In [21]:
```
#dis-advantage

# MOD is not diffrentiable at origin & we cant apply optimization techniques like gra
# MSE(mean squared error ) solves this problem
```

MSE

- We remove mod & use square instead..

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

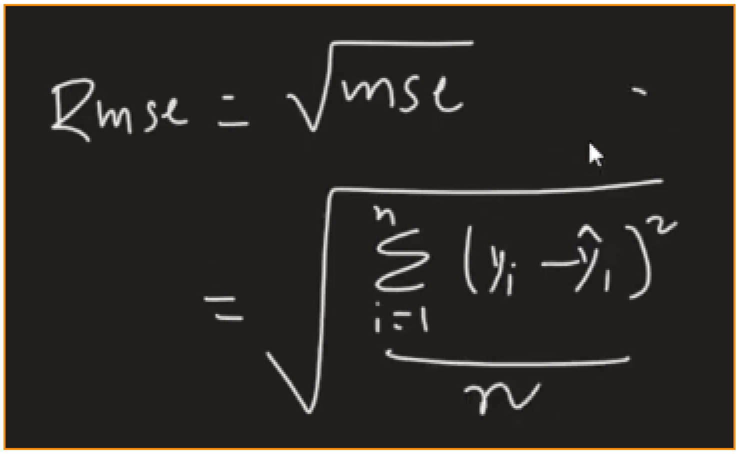


- **Disadvantage**
    - we'll get a number fron MSE but its unit is not same as output, instead it will be square of output.
    - prone to outlier since its taking a square

- **Advantage:** it can be used as a Loss Function since its diffrentiable

RMSE

- Its nothing but root of MSE

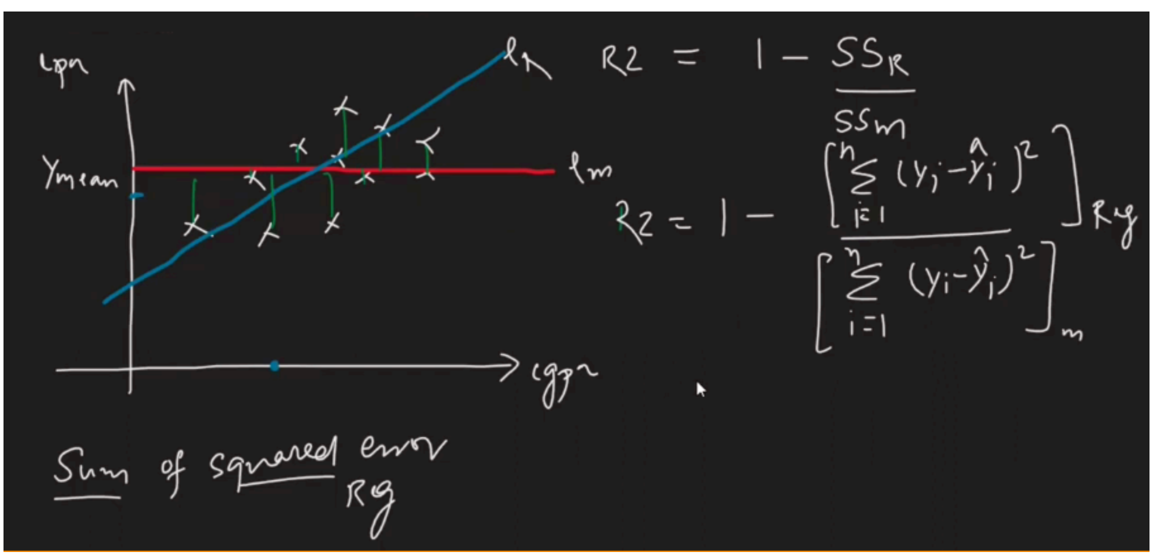


Advantage: It can be used as a Loss Function since its diffrentiable & its output is in same unit as output

disadvantage: Not robust to outliers

r2 score

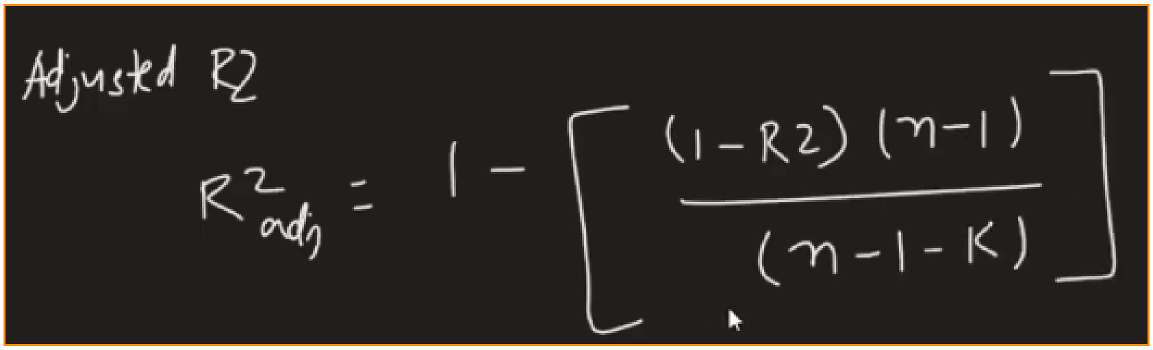
- In r2 score we find how better is our prediction then mean
- Its also coefficient of determination/ goodness of fit



- If r2 score is 0 means SSR/SSM is 1 means SSR & SSM are making same mistakes.
- If r2 score is 1 means SSR/SSM is 0, it can only be 0 when numerator(regression line) is not making any error means its passing through all data points, its a perfect line.
- So after we calculate r2 score, we have to move towards 1 rather then 0

- There are possibilities when r2 score is -ve, means SSR/SSM > 1 that can only happen when SSR is > SSM means regression is making even more mistakes then mean... thats a worst model.
- If r2 score is 0.8 --> It means cgpa is able to explain 80% variation in package
- **Problem**
- r2 score increases when we add more input columns even if its irrelevant like CGPA & Temprature where tmprature is completly out of context but r2 score may increase & to solve this problem we can use adjusted r2 score

Adjuted r2 score



where:

- r2 = r2 score
- n = no of rows
- k = total no of input cols
- lets assume we add an irrelevant col like temprature
- k (total no of cols) will increase & it will decrease the denominator (n-1-k)
- In numerator: (n-1) will remain constant since no changes done in no of rows
- r2 score when added irrelevant col:
  - it will remain unchanged: which will make numerator to be remain constant
  - since denominator is decreasing, it will increase whole term inside bracket
  - when we substract it with 1, then adjusted r2 score will decrease
- r2 score when added relevant col:
  - k (total no of cols) will increase & it will decrease the denominator (n-1-k)
  - (n-1) is constant
  - (1-r2) will decrease faster then denominator since r2 score will increase faster
  - adjusted r2 score will increase

- **when we are dealing with multiple columns then its better to count on adjusted r2 sore.**

Regression metrics Python application

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

In [23]:
```
y_pred = lr.predict(X_test)
```

In [24]:
```
y_test.values
```

Out[24]:
```
array([4.1 , 3.49, 2.08, 2.33, 1.94, 1.48, 1.86, 3.09, 4.21, 2.87, 3.65,
       4.  , 2.89, 2.6 , 2.99, 3.25, 1.86, 3.67, 2.37, 3.42, 2.48, 3.65,
       2.6 , 2.83, 4.08, 2.56, 3.58, 3.81, 4.09, 2.01, 3.63, 2.92, 3.51,
       1.94, 2.21, 3.34, 3.34, 3.23, 2.01, 2.61])
```

In [25]:
```
print("MAE: ", mean_absolute_error(y_test, y_pred))
print("MSE: ", mean_squared_error(y_test, y_pred))
print("RMSE: ", np.sqrt(mean_squared_error(y_test, y_pred)))
print("R2 Score: ", r2_score(y_test, y_pred))
```

```
MAE:  0.2884710931878175
MSE:  0.12129235313495527
RMSE:  0.34827051717731616
R2 Score:  0.780730147510384
```

In [26]:
```
r2 = r2_score(y_test, y_pred)
n = len(y_test)
k = 1
r2_adj = 1 - (1 - r2) * (n - 1) / (n - k - 1)
print("Adjusted R2 Score: ",r2_adj)
```

```
Adjusted R2 Score:  0.7749598882343415
```

In [ ]:
```

```