Regression $\longrightarrow$ Simple Linear Regression
Polynomial linear regression
Multiple Linear Regression

Regression $\longrightarrow$ How much? --- Label.

If your label is a numeric data,
go for regression.

Regression is represented by the formula,

one (x) feature
one (y) label

$$y = mx + b$$ ------- SLR

slope of the line

intercept of the line

$$y = m_1 x^0 + m_2 x^1 + m_3 x^2 + b$$ .... Poly LR with degree = 2

$$\boxed{y = m_1(1) + m_2 x + m_3 x^2 + b}$$ .... Polynomial features

'n' features $x_1 \dots x_n$ where $n \in N$
one label (y)

$$y = m_1 x_1 + m_2 x_2 + \dots + m_n x_n + b$$

$\hookrightarrow$ Multiple Linear Regression

eg.

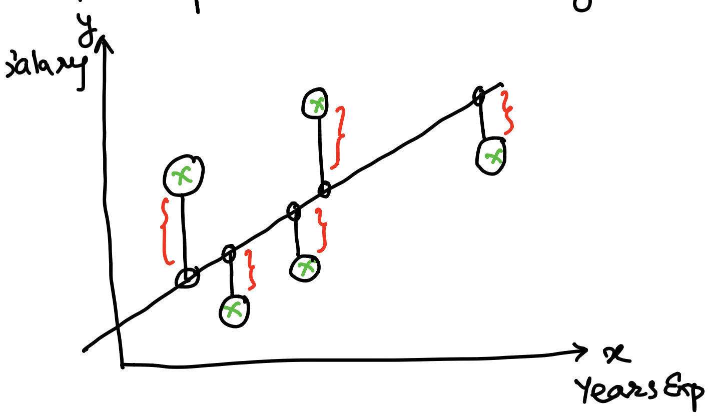
profit = m_1 (R&D) + m_2 (Adm) + m_3 (markg) + m_4 (state) + b

$\Rightarrow$ model.fit(x_train, y_train) the values/ co-efficients of each feature variable and the intercept of the line gets calculated.

In the world of ML, its all about achieving a GENERALIZED MODEL.

A given generalized model has only approximation.

YearsExperience v/s Salary



(x axis → features ---> 2d plots
y axis → label) --- 3d plots

pairplot

↑
errors must be checked w.r.t label

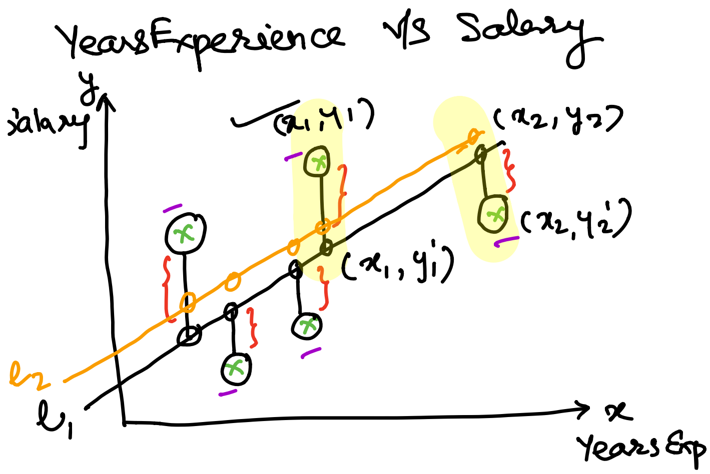
Errors $\Longrightarrow$ Residuals.

Calculation of errors

↓ Absolute value and calc. the mean (Mean Absolute Error)

↓ Square value and calc. the mean (Mean Squared Error)

Mean Absolute Error



Our motto is to place the line in such a way that mean of absolute errors must be minimum

given 5 points what is the error value;

Mean Calculating Error $\rightarrow$ $$\frac{\sum_{i=1}^{m} \left| (y_1' - y_1) \right|}{m}$$

m is number of data points

Mean absolute Error.

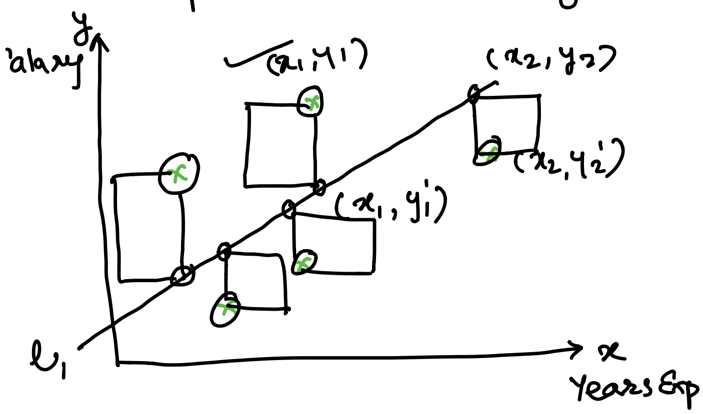
line 1 $\rightarrow$ 20
line 2 $\rightarrow$ 19
line 3 $\rightarrow$ 18
line 4 $\rightarrow$ 21
line 5 $\rightarrow$ 22

$\rightarrow$ $\boxed{18}$ $\rightarrow$ line 3 can be considered as an acceptable model.

(sklearn)
LinearRegression

Mean Squared Error

YearsExperience v/s Salary



area of square
(every time output will be positive)

$$\text{mean squared error} = \frac{\sum_{i=1}^{m} (y_i' - y_i)^2}{m}$$

m is no of data points.

line1 → 400
line 2 → 19^2
line 3 → 18^2
line 4 → 21^2
line 5 → 22^2

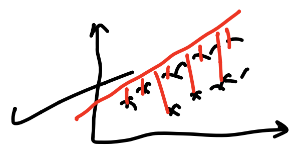
— check the magnitude (check for lower value)

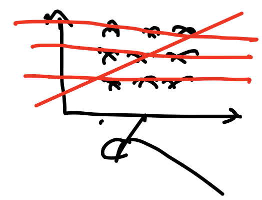
choosing that line what has less error

Ideal requirement while working with Linear Regression.

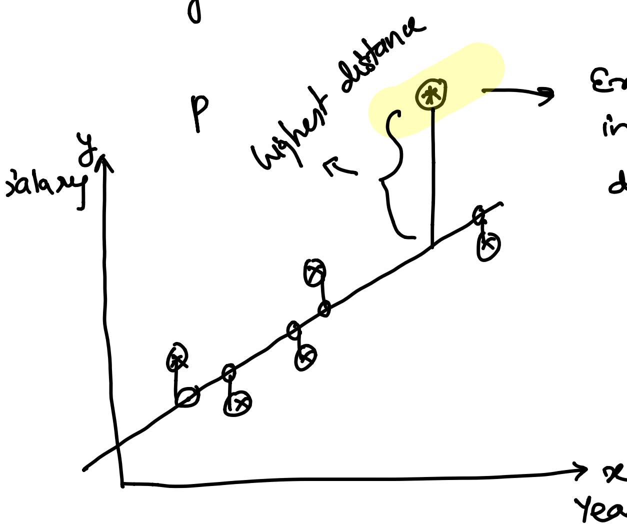
① LR works best when the data is linear.

$$\text{Error}_{\min} \propto \frac{1}{\text{Prediction}} \ \text{high}$$



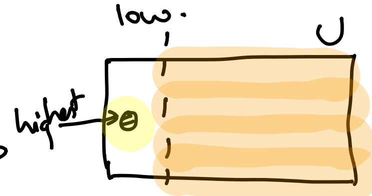


② LR is very sensitive to OUTLIERS



Error will be heavily influenced by this data point.

contribution of this set is very low.



6 pts

pt1 → 2

pt2 → 1

pt3 → 1.5

pt4 → 2

pt5 → 11 } → delete this record

pt → 2

Whenever we create a Linear Regression model, we try to minimize the error. To minimize the error we have two techniques:

ⓐ OLS ⟶ Ordinarily Least Square

ⓑ Gradient Descent ⟶ Batch Gradient Descent
Stochastic GD
mini- batch SGD.

OLS (multiple LR) . . . internally uses MSE formula

→ Introduce only those features who has higher co-relation with the label.

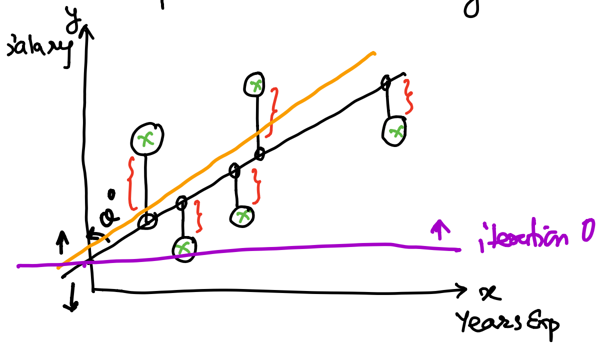
eg: f1 f2 f3 f4 f5

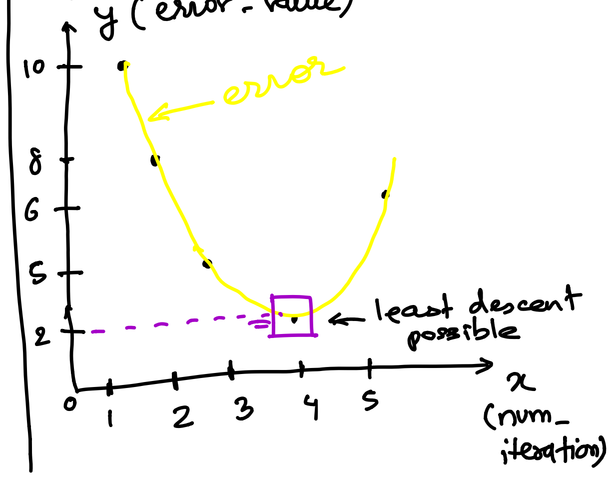
p-value ⟶ Significance level ($<= 0.05$) ⇢ proven by corollaries by statistican

using OLS, I need to discard those features whose p value > 0.05

This practical approach is called as BACKWARD ELIMINATION.

Gradient Descent → to identify the best line.
(the line which has minimum error) (MSE)





α → Learning Rate

which will affect,

① Position of y —— ①

② α° will get affect w.r.t y — ②

$y = mx + b$

salary = m (y exp) + b —①

= m (y exp) α + bα

②

assume, iteration 1 { y = 1, α = 60° } → 10

α → $\boxed{0.001}$ → 1

↳ hit and try

num_iteration → 50 to 1000

$\boxed{75}$

99% i get the best model

np explanation

dimension source depth coln wise

np.ones((50,1)).astype(int), feature, axis=1