Simple Linear Regression:

→ We will be having one independent & one dependent variable

→ steps to follow:

1) Collecting the Data ✓

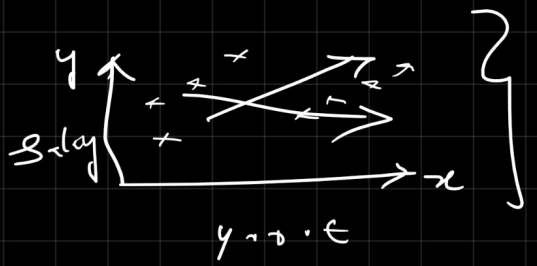
- Internal source
- External source
- Websites [Kaggle, UCI]

Math intution for Simple Linear regression:

years of experience (x)	Salary (y)
1.1	10
1.4	20
1.9	50
2.4	100
3.5	150
$\bar{x}$ = 2.0	$\bar{y}$ = 66

$\bar{x}$ → Mean(x) $\bar{y}$ = Mean(y)

→ we need to check the visualization



$$y = mx + c$$

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

Regression } → straight line → prediction

↓

$y = mx + c$ → m = Co-efficient → c = intercept

{Data} → {$y = mx + c$} Algorithm

$\bar{x} = 2.0$, $\bar{y} = 66$

$m = (1.4 - 2.0) + (1.4 - 2.0) + (1.9 - 2.0) + (2.4 - 2.0) + (3 - 2.0) \times$

$(120 - 66) + (90 - 66) + (50 - 66) + (100 - 66) + (150 - 66)$

$\boxed{m = 8}$

$y = mx + c$

↓

$y = 8x + c$ } find c ?

$\bar{x}$, $\bar{y}$

$y = 8x + c$

$c = y - 8x$

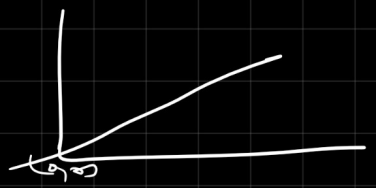
$c = 66 - 8(2.0)$

$66 - 16.0$

$\{50.0\}$

$\boxed{c = 50}$

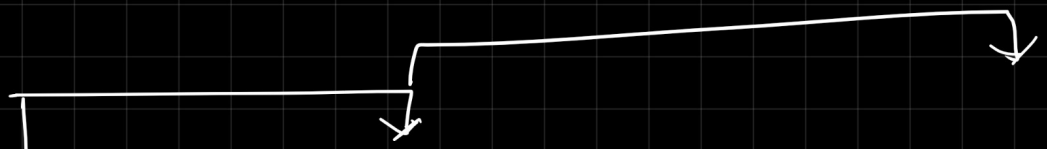
$y = mx + c$ $\overset{c=0}{=}$ $y = mx$



$y = mx + c$

↓

$\boxed{y = 8x + 50}$ } straight line



Data → { y = mx + c → Algorithm → y = 8x + 50 → Model

Algorithm : used to learn something from the data

Model : which is used for predictions.

Years of Experience (x)	Salary (y)	ŷ (predictions)	Error / Residual (y − ŷ)
1.1	10	58.8	−48
1.4	20	61.2	−41
1.9	50	65.2	−15
2.4	100	69.2	31
3.5	150	78.0	72

Training

$y = 8x + 50$ → y = mx + c

$y = 8(1.1) + 50$

$y = 8(1.4) + 50$

$y = 8(1.9) + 50$

$y = 8(3.5) + 50$

$y = 8(2.4) + 50$

Residual ↔ Error

Data : 100 observation's

{80 observation's
Training Data
$y = mx + c$

{20 observation's
Test Data

2 > year's old

Dog

Data (100 Rows)

{Training Data (80 Rows)
$y = mx + c$
$y = 8x + 50$ → model

{Test_Data (20 Rows)

Accuracy } → 96 %

Data
fit
$y = mx + c$

Co_Rows

80% → Rows → train

20% → Rows → Test

X_train	Y_train
2	25
4	40
5	60
8	1 Lpa
10	2.5 Lpa
1	10

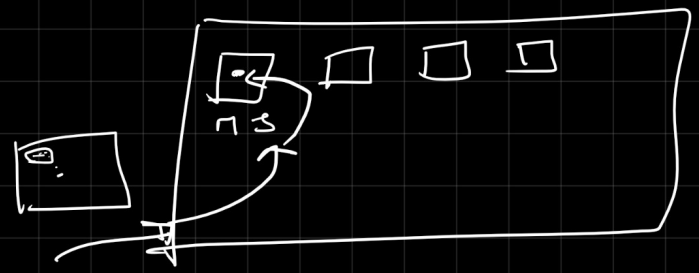
X	Y
1	10
2 →	25
3	30
4 →	40
5 →	60
6	80
7	90
8	1 Lpa
9	2 Lpa
10	2.5 Lpa

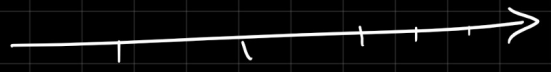
$y = mx + c$

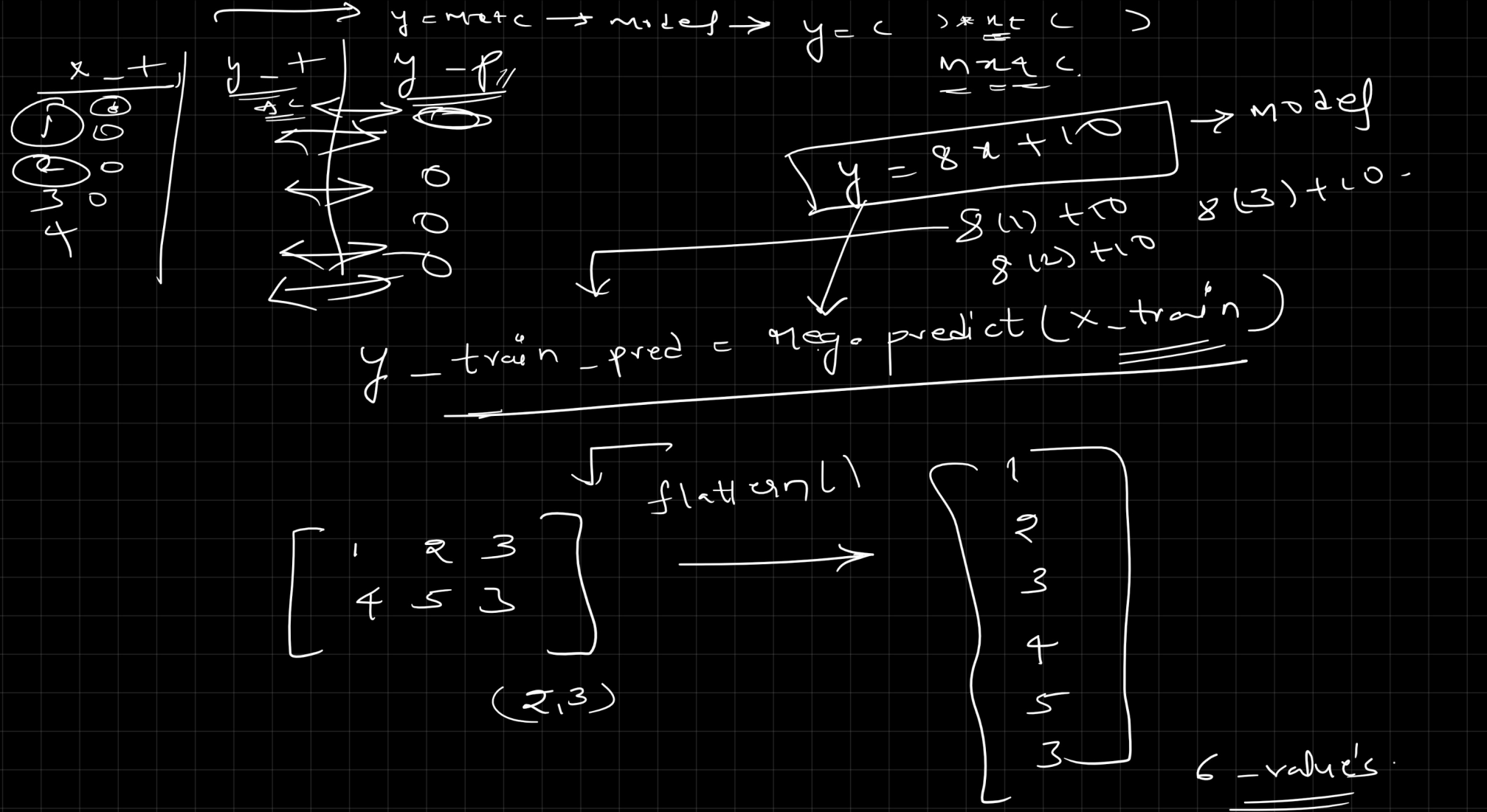
↓

model

X_test	y_test
3	30
6	80
7	90
9	2 Lpa



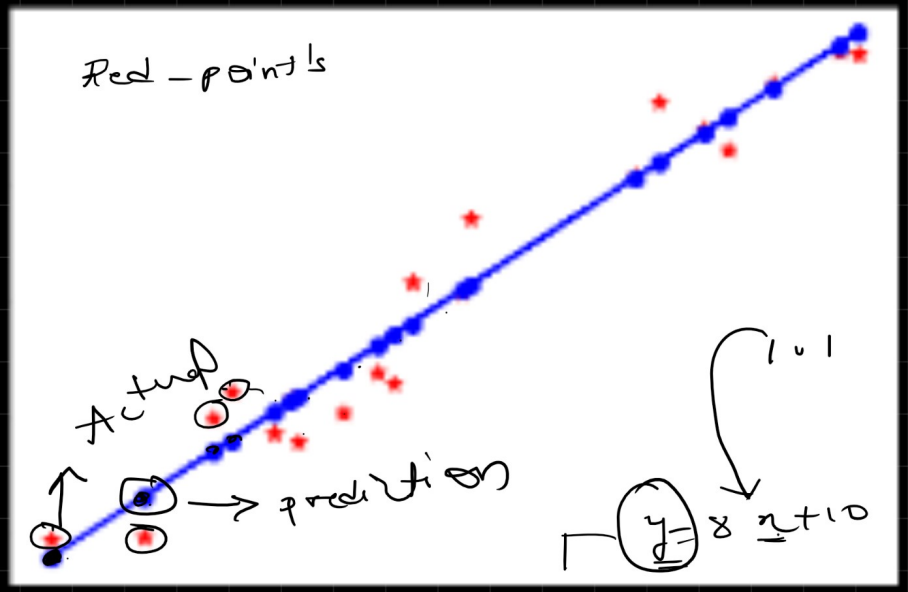



y = mx + c → model → y = ( ) * x + ( )
x_train
y_train
y_pred
y = 8x + 10 → model
8(1) + 10
8(2) + 10
8(3) + 10
y_train_pred = reg.predict(x_train)
flatten()
(2,3)
6 values

$y = mx + c$

$\downarrow$

$y = 8x + 10$



y_train

x_train.

Training Accuracy

$-1 \longleftrightarrow 0 \Rightarrow 1$

$\xi R^2 \rightarrow 1 - \frac{(y - \hat{y})^2}{(y - \bar{y})^2}$

$1 - \left[\frac{\sum_{i=1}^{n}(\cdot - P)^2}{\sum_{i=1}^{n}(\cdot - \cdot)^2}\right]$

$\Rightarrow$ Accuracy.

$\xi R^2 \Rightarrow \rightarrow 1$ ✓ Good Model

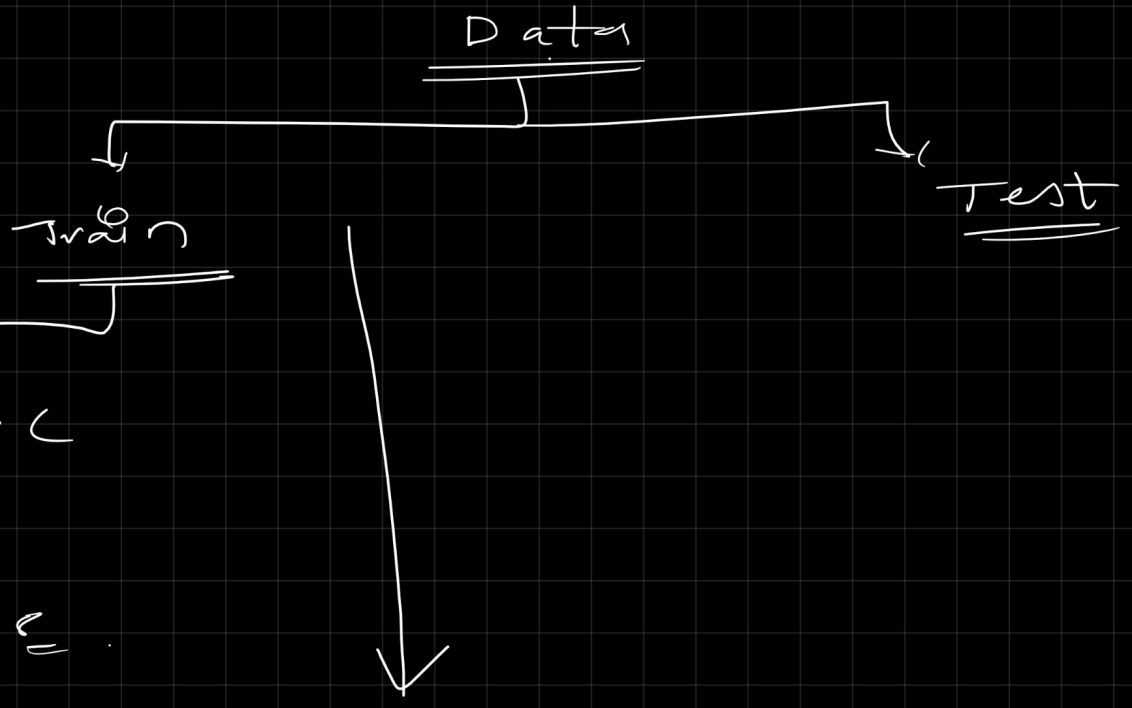
Bad Model $\leftarrow$

Test } → Model →

{ Loss } Accuracy } Loss function }

optimizer's } → Gradient Descent } Learning } step-size }

M, spoc, incept } derivative's } saving model } checking with real data.



$y = mx + c$

$y = E^{x} + c$

predicted

{A P} → P

Accuracy

↓ r²_score

$$1 - \frac{(y-\hat{y})^2}{(y-\bar{y})^2}$$

Data (Kaggle):

train →

y = ... → Alg

↓

→ y = ... → model

↓

x_train, y_train, y_trp

Error (Residual) → $(y-\hat{y})^2$

2 - 8

10K ⟷ 20K

} → seen

(Mean)

$\hat{y}$ → prediction

Test

x_test → y_test

x_t | y_test | y_p

⟷

r²_score

test_Accuracy

$R^2\_score \Rightarrow 1 - \frac{sum(y-\hat{y})^2}{sum(y-\bar{y})^2}$

$\Rightarrow$ Accuracy

---

1) Loss for model

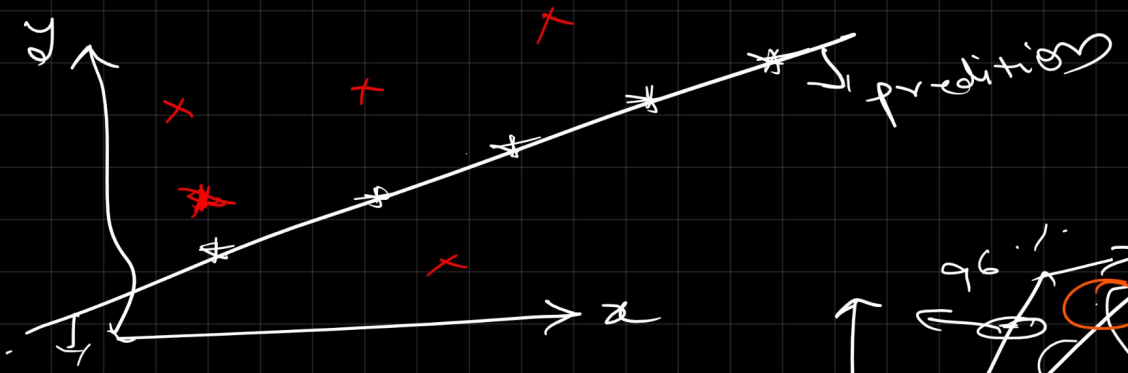
Accu = 96 %

2) optimizers

0 → 100

96 % → Acc

Loss → 4 %



100

m c :

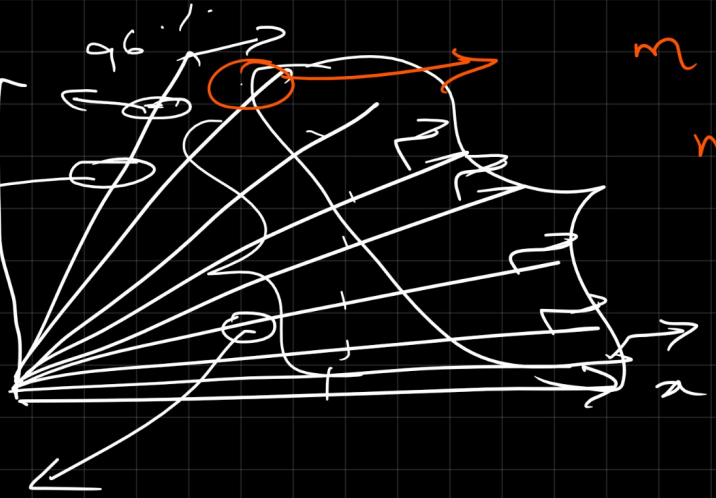
m x + c

$mx + c$

↓

input

change the predictions



$mx + c$ → change

Syntax:

optimezier Algorithms: used to Meduce the Loss of the Model:

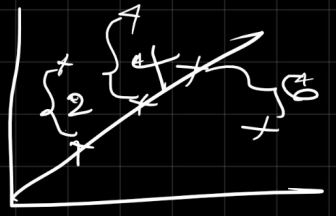
1) Gradient Descent → 21 → day's
2) Stochastic Gradient Descent
3) K_fold (Batch sizes)
4) Adam
5) Adagrad
6) Adadelta
7) Rms promp.
8) MAP.

Gradient Descent :

Loss function / error function / cost function

*) ⊛ Mean Square Error $= \frac{1}{n} * \sum_{i=1}^{n} (y - \hat{y})^2$

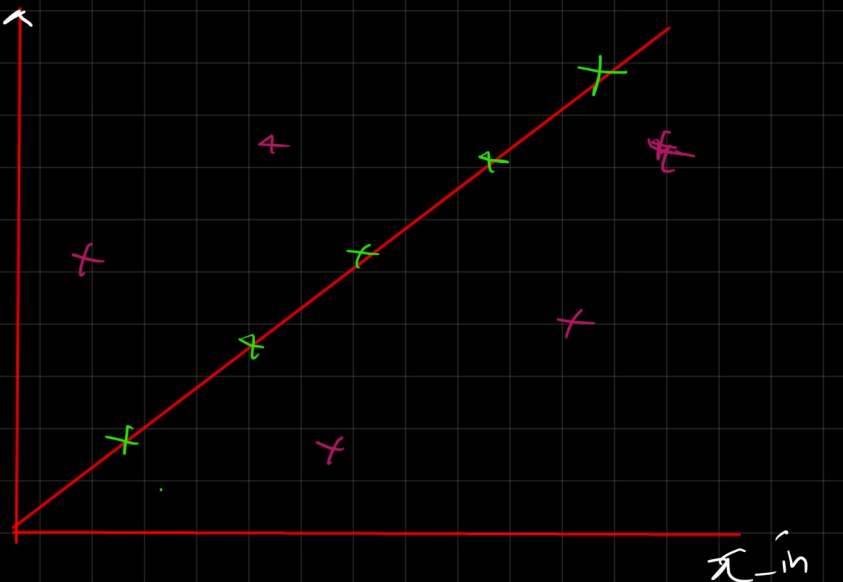
*) Absolute mean Square Error $\Rightarrow |y - \hat{y}|^2$



→ Avrage

10 &

*) Root mean Square error $= \sum_{i=1}^{n} \sqrt{\frac{(y - \hat{y})^2}{n}}$



Data → $y = mx + c$

↓

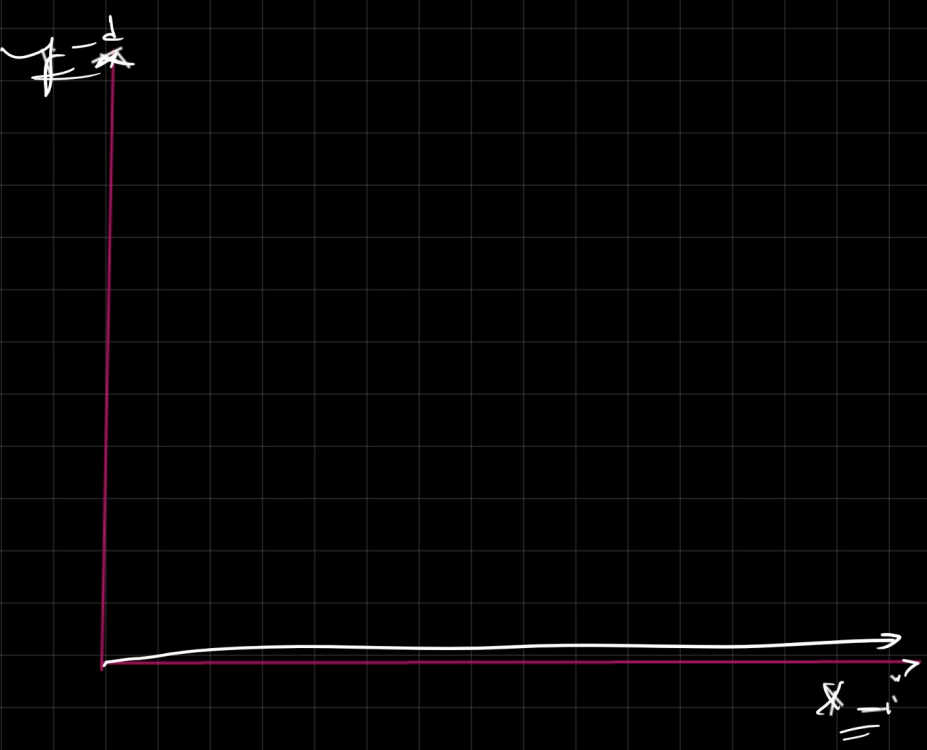
$m = 4, \quad c = 0$

$y = 4x + 0$

Data → $y = mx + c \rightarrow m = 0, \; c = 0$

$y = 0x + 0$

$y = 0$



Cost function $\Rightarrow \frac{1}{n} \sum_{i=1}^{n} (y - \hat{y})^2$

$\hat{y} = \hat{m}x + c$

Gradient_descent:

Apply derivative's for slope & intercept

$c.f \Rightarrow \frac{1}{n} \sum_{i=1}^{n} (y - (\hat{m}x + c))^2$

$$ds \Rightarrow -\frac{2}{n} \sum_{i=1}^{n} x(y - mx + c)$$

$x^2 = 2x$

$n \cdot x^{n-1}$

$2 \cdot x^{2-1} = 2x$

Derivate's

$$d \cdot c \Rightarrow \frac{2}{n}\sum_{i=1}^{n}\left(y-(mx+c)\right)^2$$

$$\frac{d}{dx}\left(\frac{-4x}{x}\right)$$

$$\frac{d}{dx}k = 0$$

$$\left\{\frac{d}{dx}\quad x^n = n \cdot x^{n-1}\right.$$

$$x^2 = 2 \cdot x^{2-1}$$

$$= 2x$$

$$\Rightarrow 2 \cdot (-4x)(-4x)$$

$$\Rightarrow$$

Step_size_s = Learning rate. * ds

Step_size_c =

Data → Algor → Model

↓ prediction

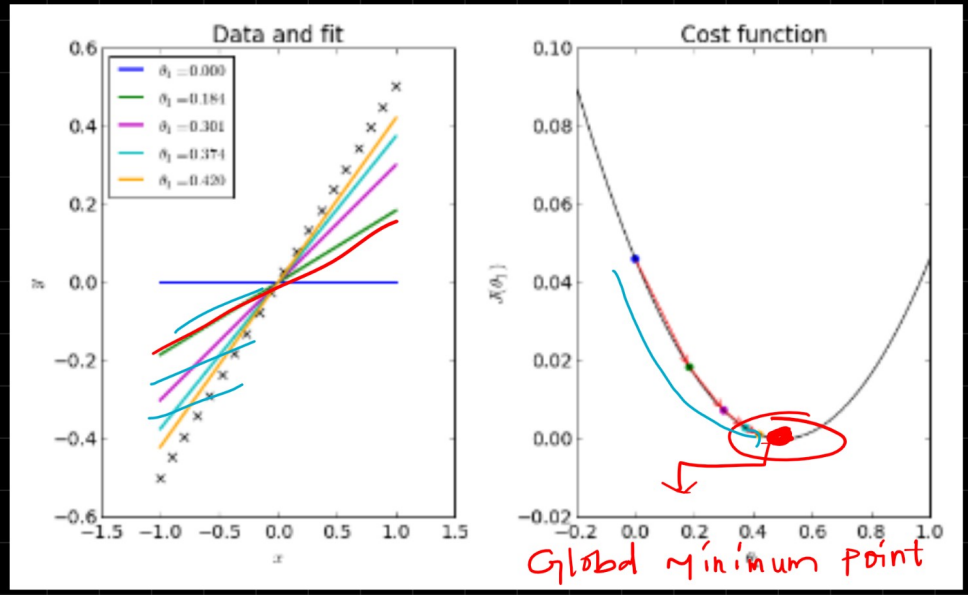
↓ Accuracy

↓ Loss

← Loss

Loss function

Mean_squared_error:



Data → $y = mx + c$ → Model ($y = 0 \cdot x + 0$)

$m = 0, \; c = 0$

1) $y = 0 \cdot x + 0$

Gradient Descent :

$$Cost = \frac{1}{n} \sum_{i=1}^{n} (y - (mx + c))^2$$

$$d \cdot s = \frac{d}{dx} \frac{1}{n} \sum_{i=1}^{n} (y - (mx + c))^2$$

$$d \cdot s = -\frac{2}{n} \sum_{i=1}^{n} x(y - (mx + c)) \rightarrow \text{for value}$$

$$d \cdot c = \frac{2}{n} \sum_{i=1}^{n} (y - (mx + c))$$

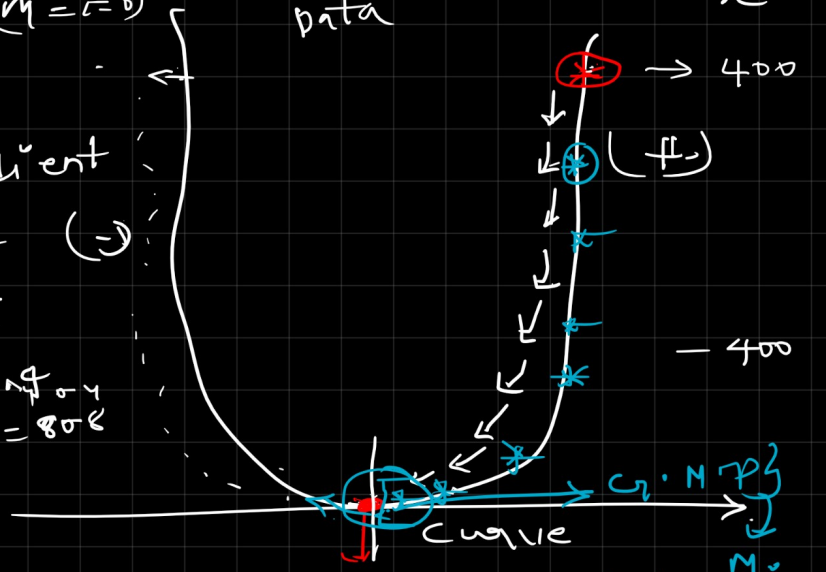
$d \cdot c$ = values

Step_size_S = Learning rate * $\frac{d}{dx}(s)$

Step_size_c = Learning rate * $\frac{d}{dx}(c)$

New_slope = old slope − step size_S

N − m = 0 − 0.4



Global_Minimum_point [Sum of Errors will be zero]

Learning rate range [0, 1] 0.01 0.1 0.2 0.7

New_intercept = Old int − step size − c

N − 9 = 0 − 0.8

$y = 0.4x + 0.8$

$y = 0 \cdot x + 0$

$\frac{d}{dx} x^2 \Rightarrow 2x$

$\frac{d}{dx}(x^2 + y^2)$

$\frac{d}{dx}(2x + c)$

$\frac{\partial}{\partial M} \frac{1}{n} \sum_{i=1}^{n} (y_i - (mx_i + b))^2$

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

$\Rightarrow$

Multiple Linear Regression: } → you can have any number of independt

S.L.R → $y = mx + c$ ← $\frac{x}{x_1} \mid \frac{x_1}{x_2} \mid \frac{x_2}{y}$

M.L.R } → $y = m_1x_1 + m_2x_2 + m_3x_3 + c$

x_1	x_2	y

x	y
(m)	

→ $y = m \cdot x + c$

$m_1 = \frac{(x_1 - \bar{x}_1)(y - \bar{y})}{(x_1 - \bar{x}_1)^2}$

$y = m_1x_1 + m_2x_2 + c$

$m_1 = 4$

$m_2 = 8$

$c = 12$

$y = 4x_1 + 8x_2 + 12$

$m_2 = (x_2 - \bar{x}_2)(y - \bar{y}) / (x_2 - \bar{x}_2)^2$

$x_1 = 10$

$x_2 = 20$

$y \to ?$

$y = 4x_1 + 8x_2 + 12$

$y = 4*10 + 8*20 + 12$

$= 40 + 160 + 12 \Rightarrow 212$

Number

a	b	c	d	e
1	2	3	{M}	10
2	10	14	{F}	20

1) code ? → dummies

2) Map :

2, 1, 3

2 | 3 | f_c

a	b	c		e
1	2	3		10
2	10	14		20

2, 3

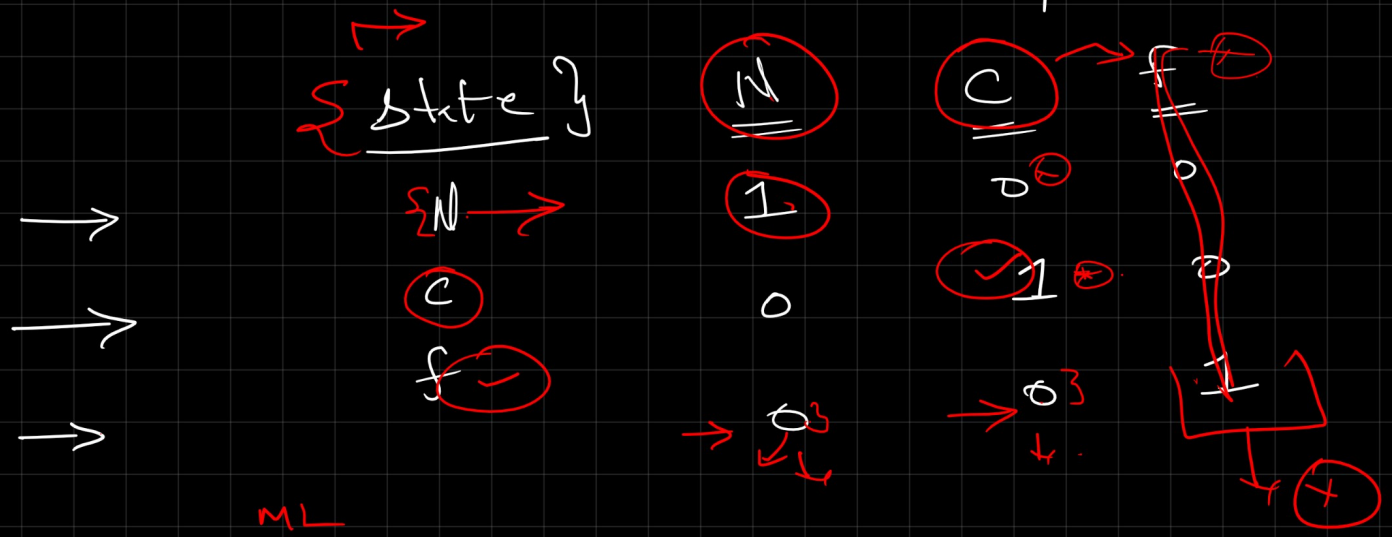
{ 100-labels

$3 \Rightarrow$ indep 1 - d.

x_1	x_2	x_3	y

$\} \rightarrow y = m_1 x_1 + m_2 x_2 + m_3 x_3 + y.$

↳ dumie's created.



ML

ML → 80, 100 } → 500 → columns

↳ Neural Networks

if your column category → Nominal Data

dummie's:

Gender } state

50% ← M → ?

← f → 50%.

f

m

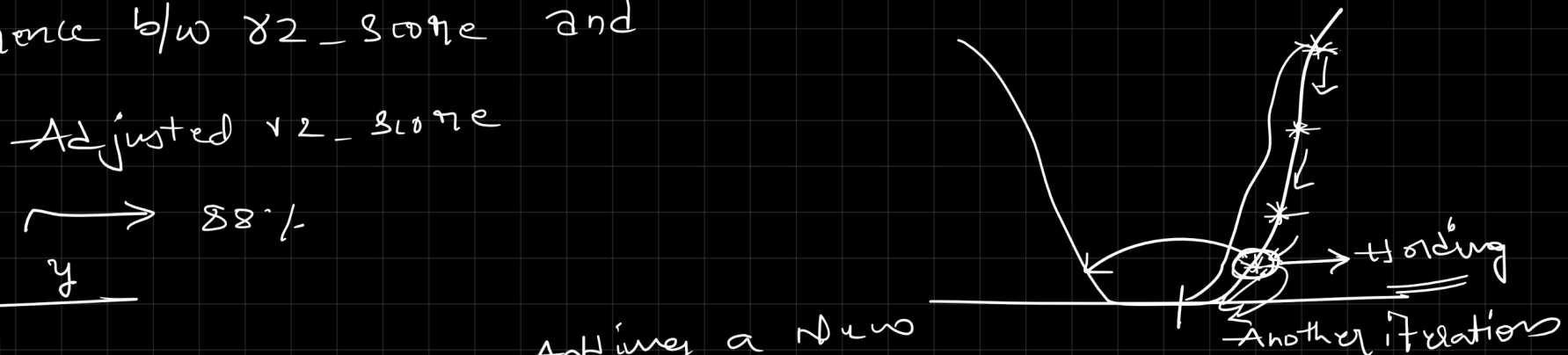
Ordinal columns

{ primary → 1
HOD → 2
Death → 3
J.L. → 4 }

Occupation

IAS → 1
IPS → 2

1) Difference b/w R2_Score and Adjusted R2_Score



→ 88%

x	x_1	y

→ 92%

x	x_1	x_2	x_3	x_4	y

Adding a New feature definitely increases the Accuracy:

Data → 88%

But how can we known the New added colum is imp or not:

x	x_1	x_2	y

→ 92%

client → x

x	x_1	x_2	x_3	y

y-o-c | is | → 96%

→ 94%

imp or not ∴ Accuracy:

Adjusted $\Rightarrow R^2$

always ↑

formula = $1 - \frac{(1-R^2)(N-1)}{(N-P-1)}$

R^2 → R2score

N → No. of observation

P → No. of columns

y = | height | s

→ 97%

↓ Not a good thing

y.o.c | y | client

Acc = 96%

Adj = 50% 50% 02% ⊗ Adj = 52%

y.o.c y

Acc = 98%

82%

Climate Good Bad

$x_1 | x_2 | \cdots | x_{10}$

y | y1

100

200

300

→ 3

125%

Reg 97%

y1 | y2

⊗

0.6 > 0.6 ✓

Jagesh → exam

40 → 3 → pass

100 → max.

* Regularization :- if there is overfitting problem:

Overfitting → if our model is working well with training data
△ Not working well with Test data.

Training_acc
96%

Test_acc
66%
} → NO:

Underfitting → Model N/w with Train & N/w Test

Train_acc
66%

Test_acc
61%

Bias & Variance :- Bias – training data, Variance → Test data
High → Bad, low → good

1) High Bias & High Var = Bad train & Bad Test → under fitting problem

2) Low Bias & Low Var = Good Train & Good Test → follow

3) Low Bias & High var ⇒ Good Train & Bad Test 2 ⇒

Overfitting problem

Regularization :- if our Model follows's overfitting problem:

Regularization

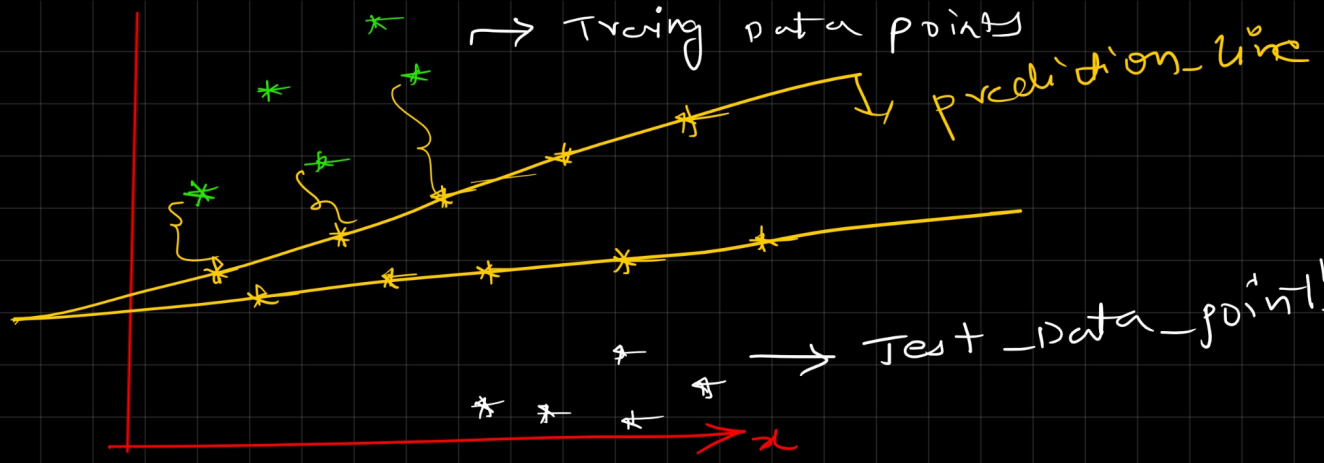
- Ridge Regression (L2)
- Lasso Regression (L1)

for ⇒ $\sum_{i=1}^{n}$ c.f

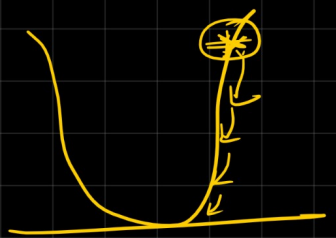
$\frac{1}{n} * \sum_{i=1}^{n} (y - \hat{y})^2$

⇒ c.f + λ *(slope)²

⇒ c.f + λ * |slope|



There is an overfitting problem in the model:



Regularization → Ridge Regression:

↳ $c.f + \lambda * (slope)^2$ updating slope

m ⇒ Tr = 96%. | Te = 80%. m → close to 0

increase small error.

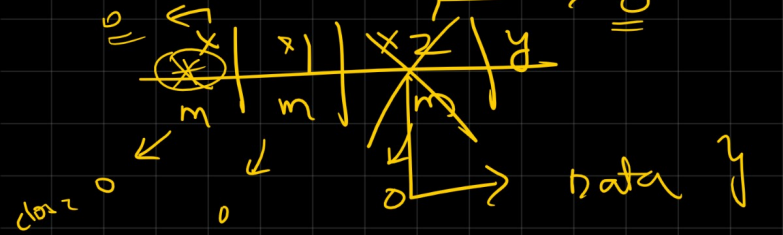
reduce the error Test part

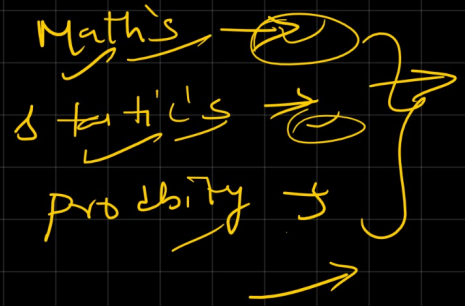
90%. 85%.

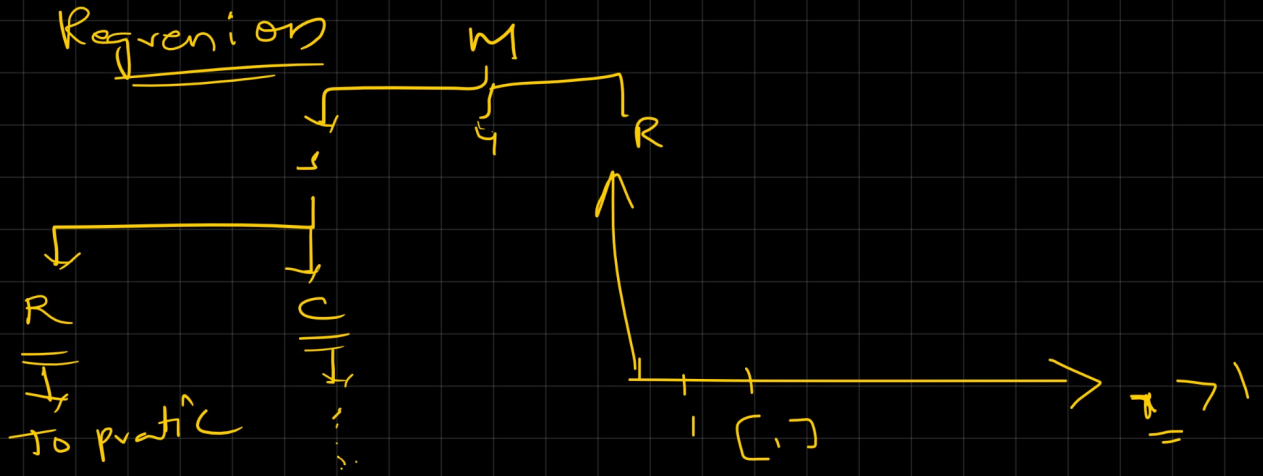
→ overfitting problem solved:

Lasso_Regression: [feature selection] powerfull technique.

m ⇒ 0




Math's
statistics
Probability


Regression
M
R
R
C
To pratic
1
[,]