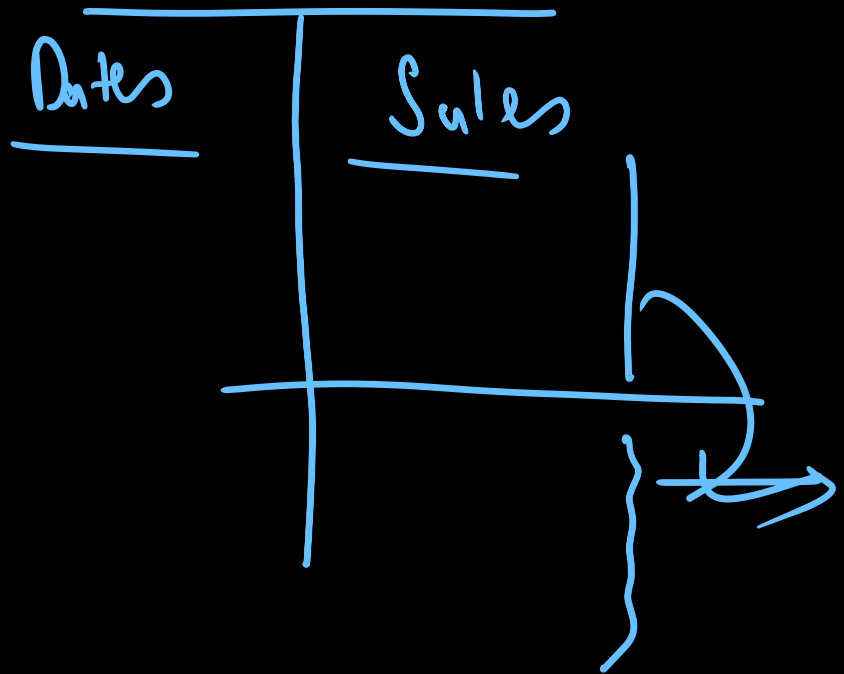
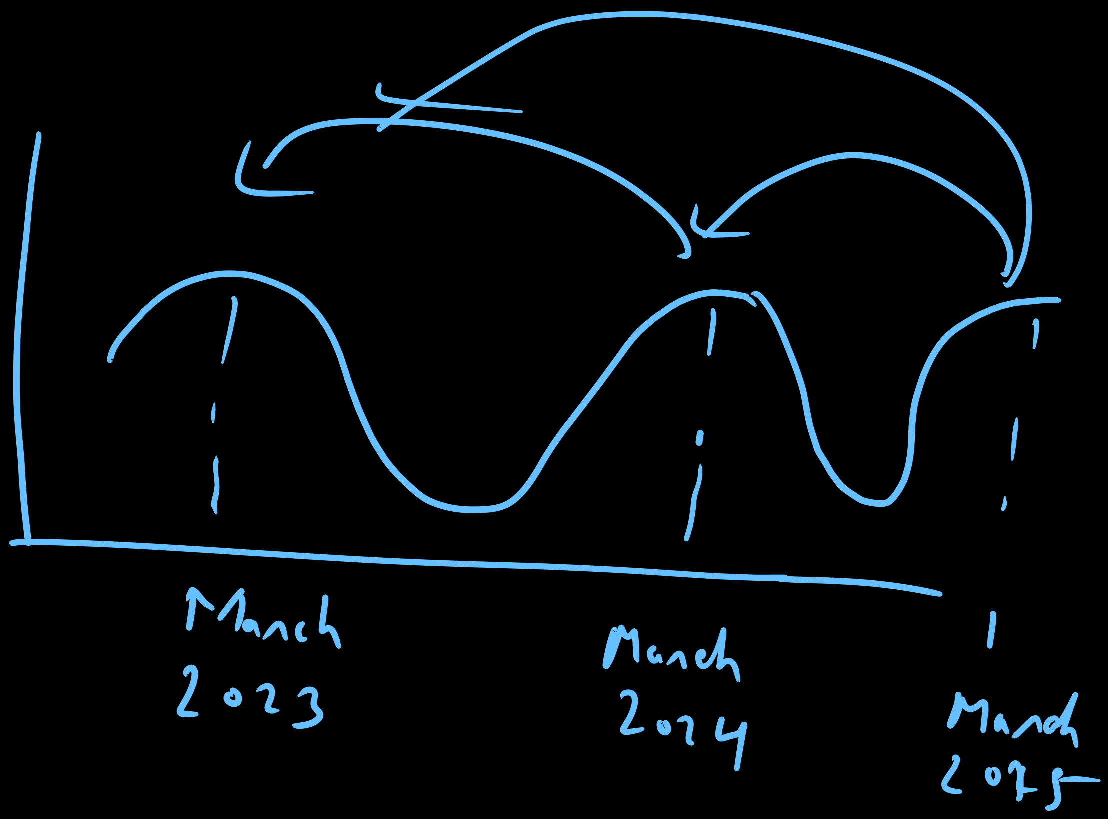
Agenda

1. Feature Scaling helps training
2. Types of GD
3. Polynomial Regression
4. Underfitting & Overfitting

→ 5 No Autocorrelation





→ Autocorrelated : Sales on this year depends on last year sale

→ Stock

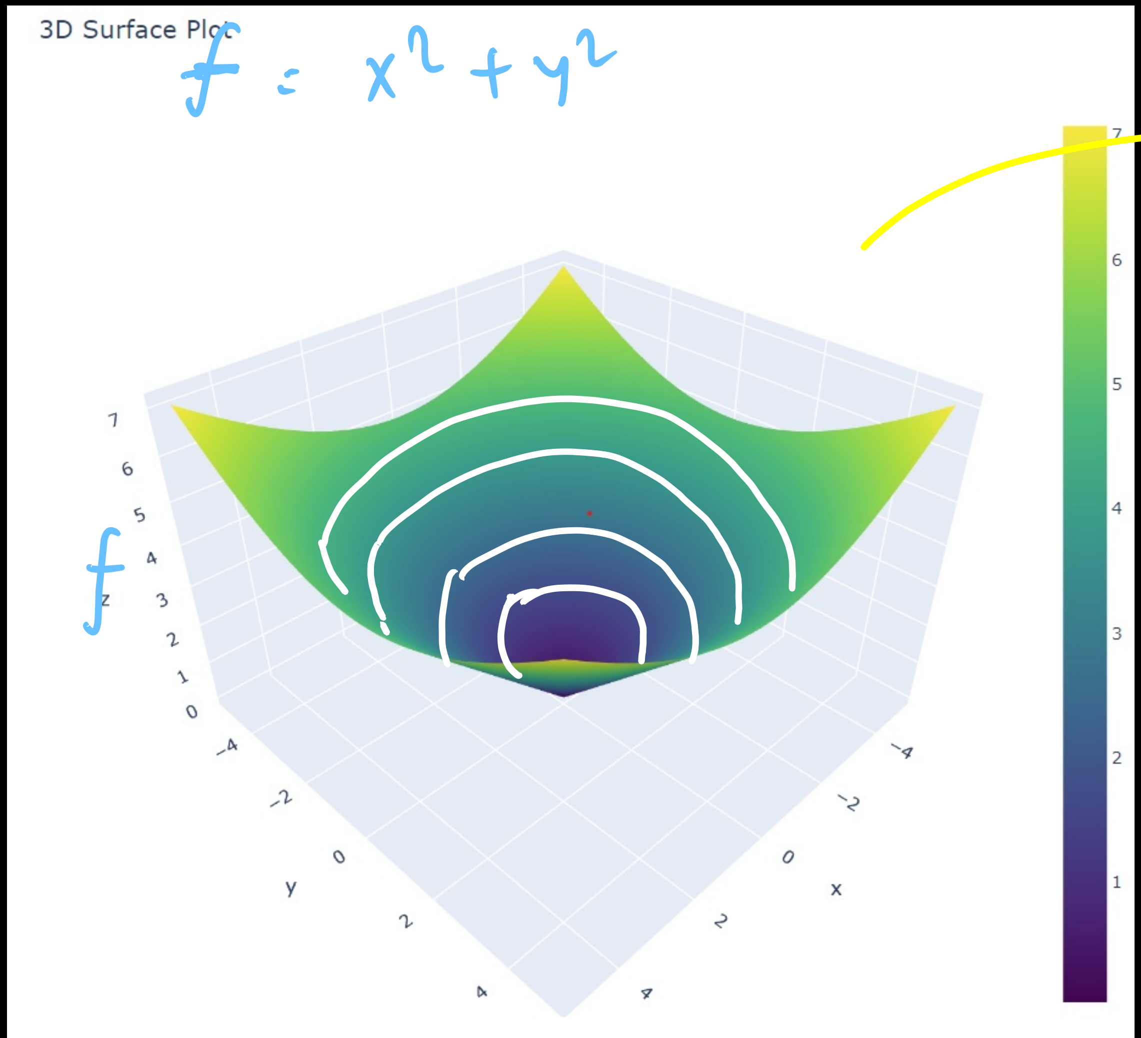
→ <u>Feature Scaling</u>

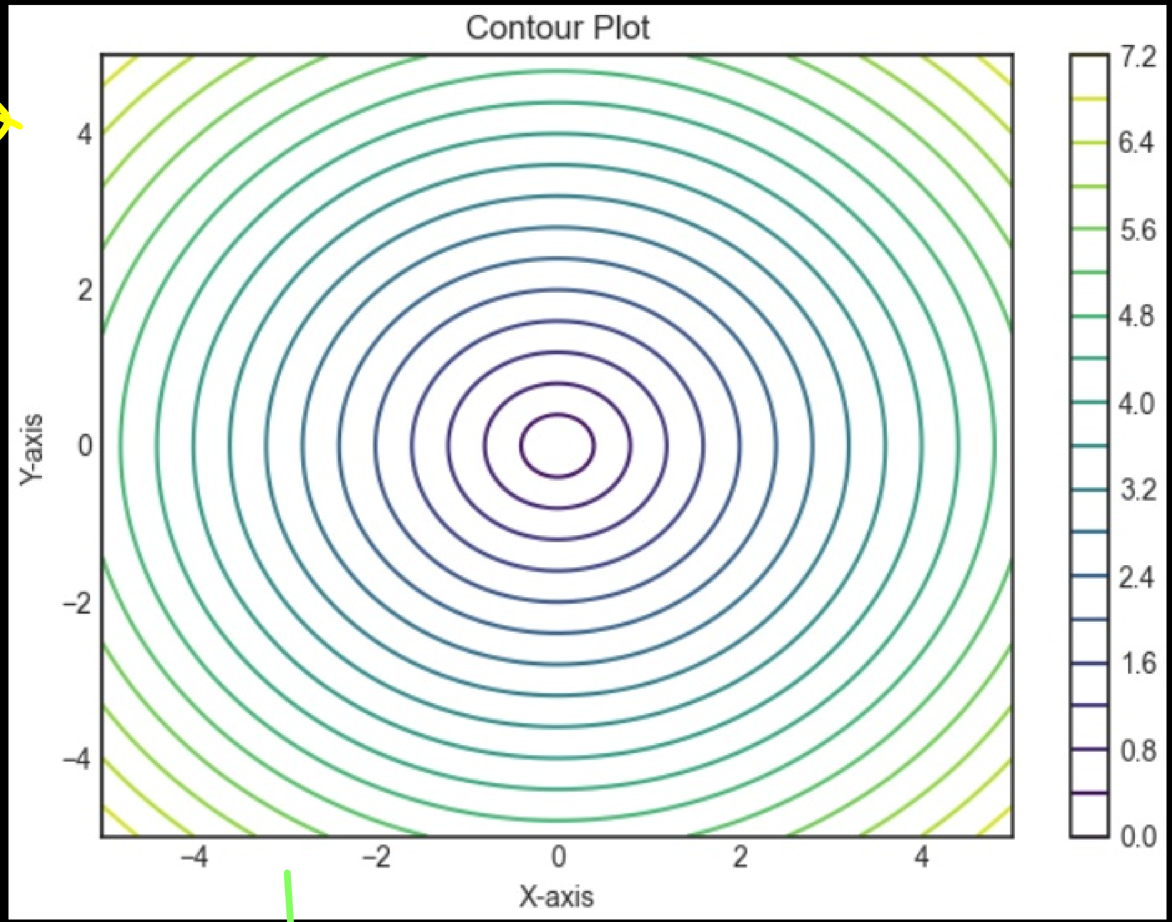
$$\hat{y} = w_1 \cdot \text{Age of car} + w_2 \cdot \text{Km\_driven} + w_0$$

Age of car: [1 - 15] → [0 - 1]

Km_driven: [10 - 100000] → [0 - 1]

1 → Model should NOT be biased toward feature larger in value

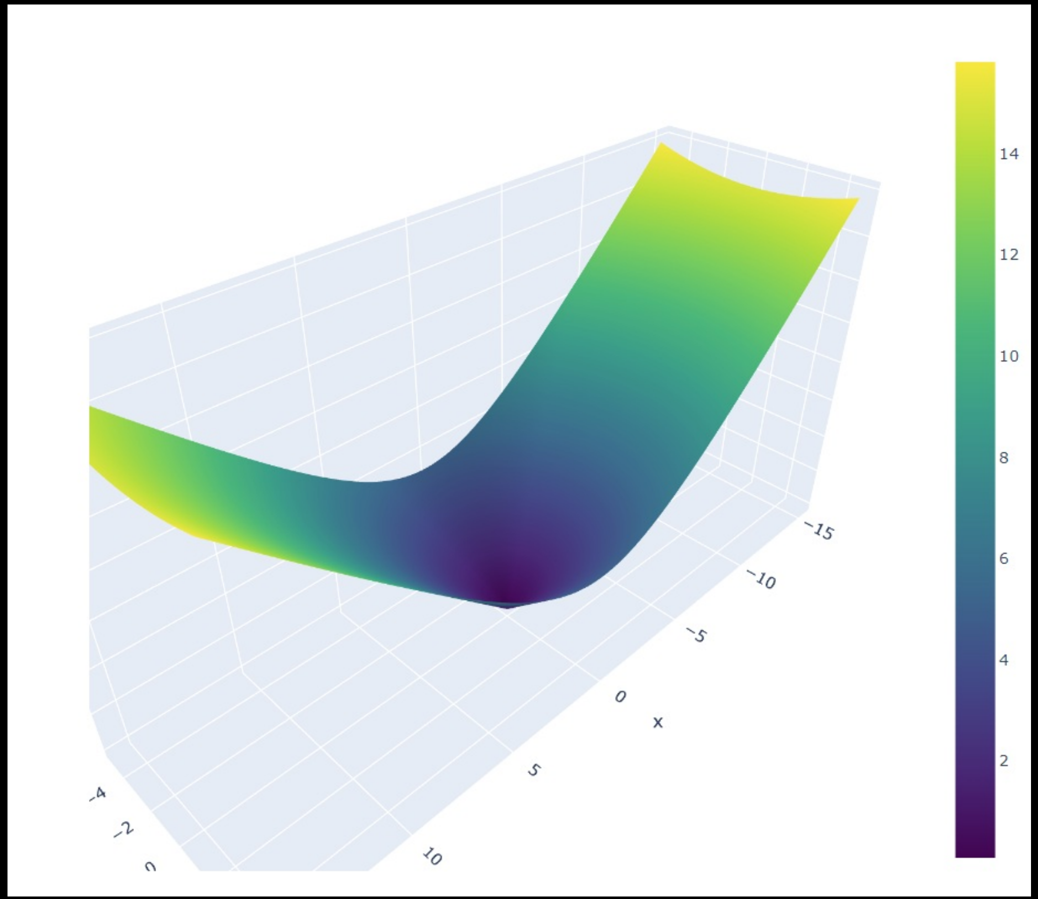


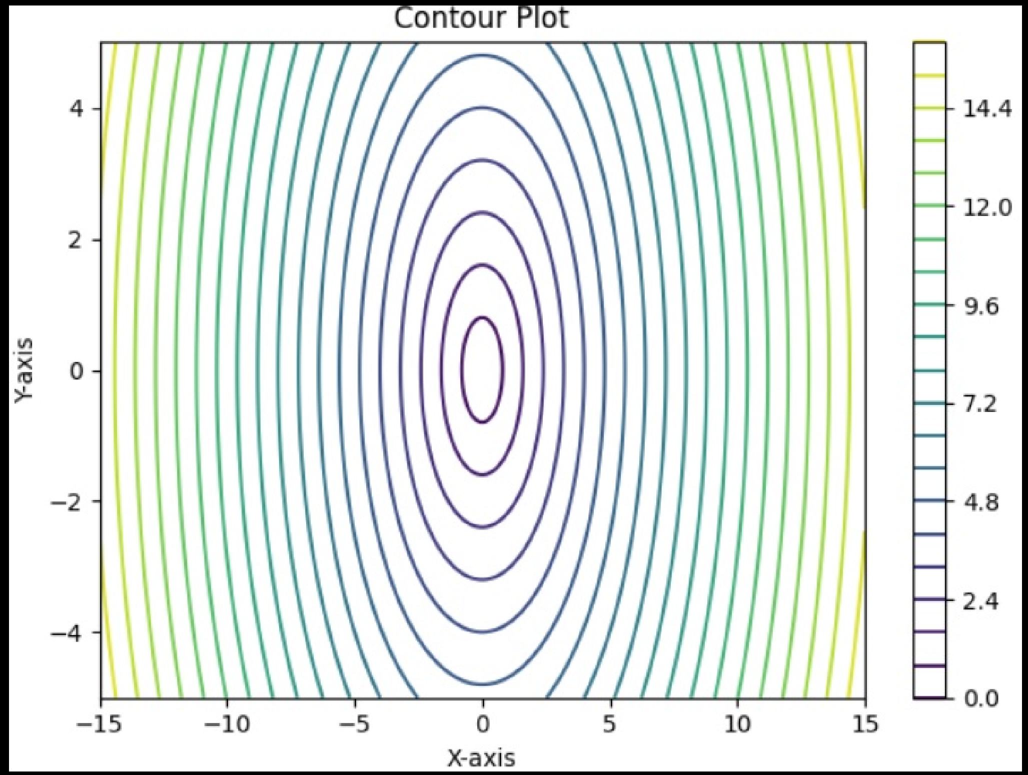


$-5 \quad 5$

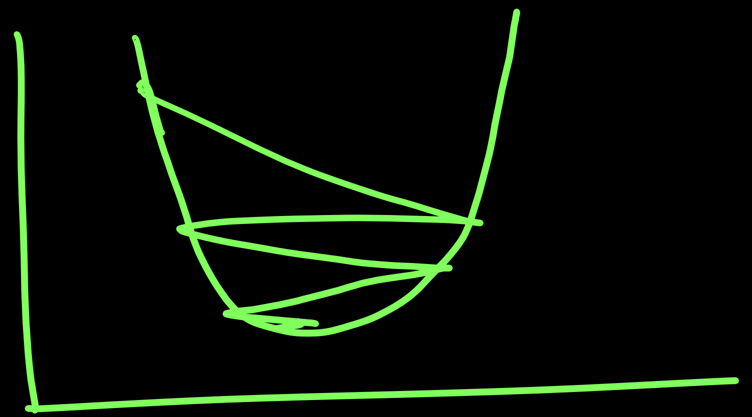
$-5 \quad 5$

Circles





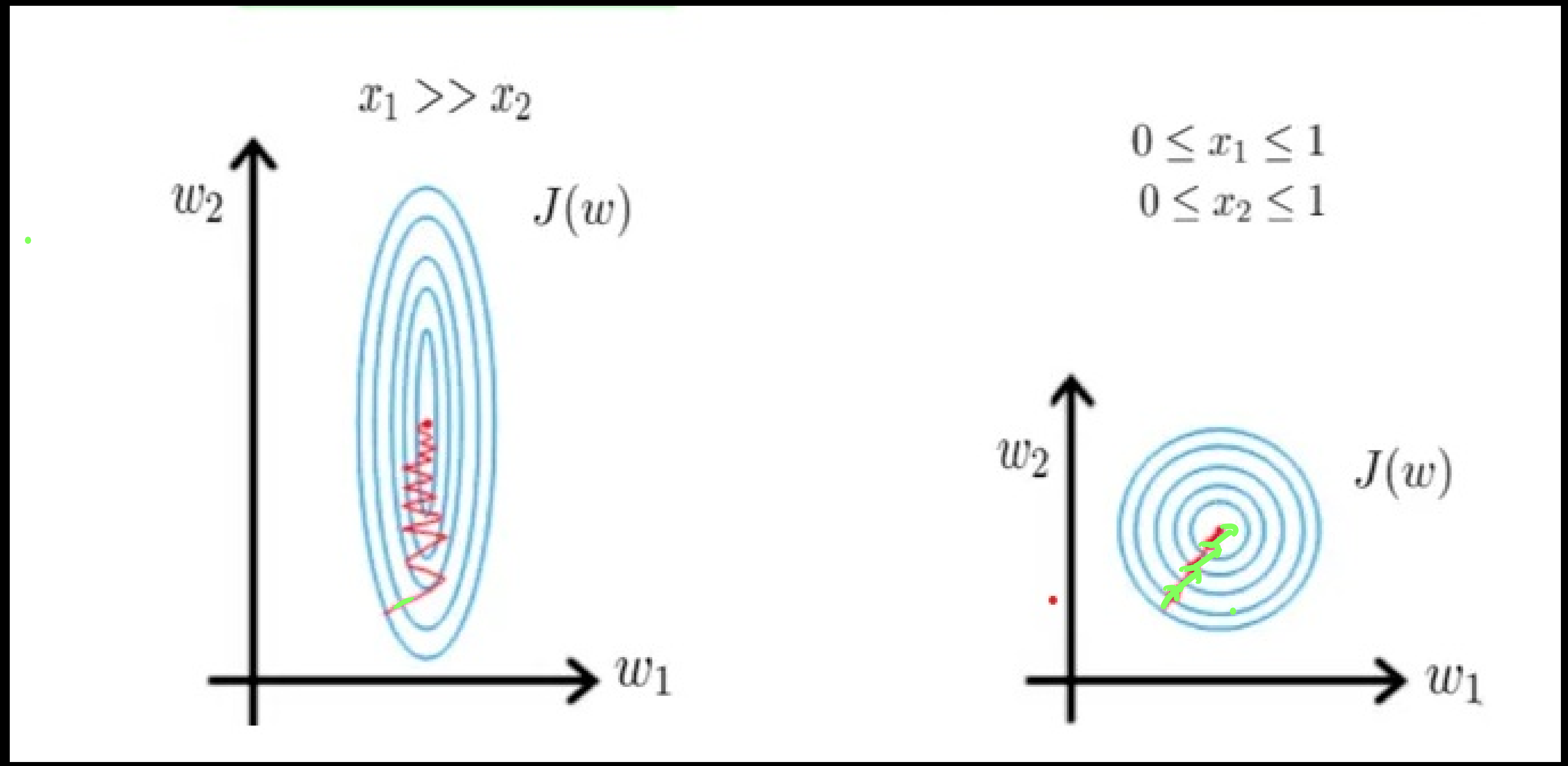
−15 to 15 → No feature scaling → Elliptical
−5 to 5



Repeat until convergence:

$$w_2 = w_2 - \eta \frac{\partial L}{\partial w_2}$$

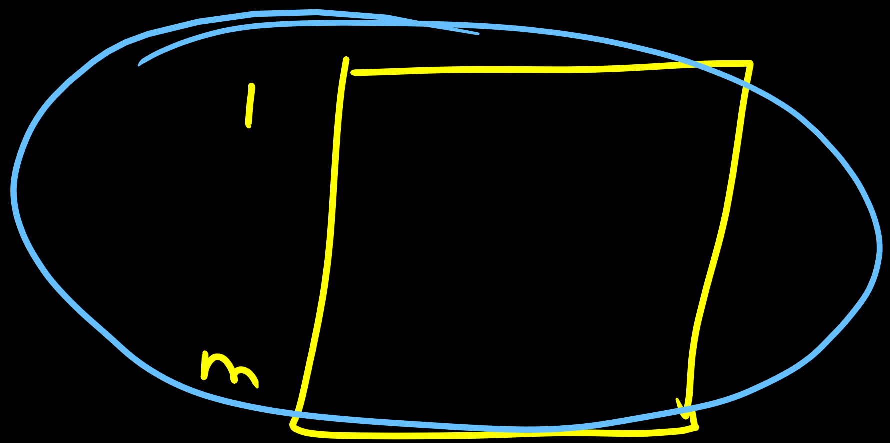
$$w_1 = w_1 - \eta \frac{\partial L}{\partial w_1}$$



(2) Converge is faster in Feature Scaled

Variants of GD

Batch GD



for iter in range(iterations): (d, m)

$m = 10^6$

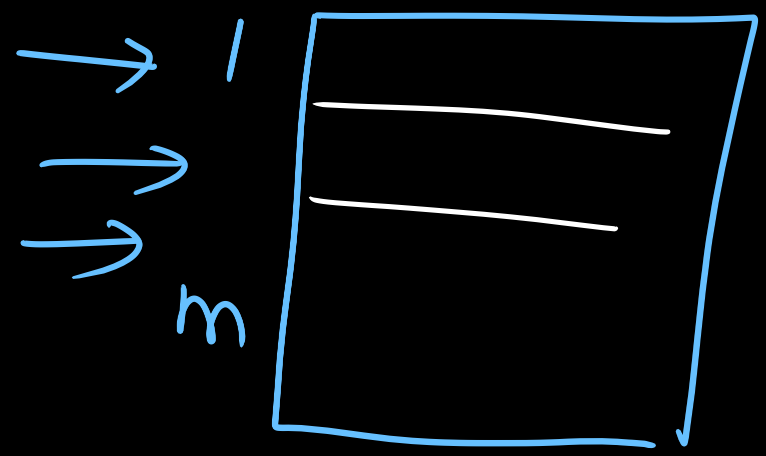
$$w_j = w_j - \eta \frac{\partial L}{\partial w}$$

$$= w_j - \eta \frac{1}{m} \sum_{i=1}^{m} \left(y^{(i)} - \hat{y}^{(i)}\right) x_j^{(i)} \rightarrow \boxed{X^T (y - \hat{y})}$$

→ Update of w happens on once in each iteration

→ Huge memory (RAM) is required to store

Stochastic GD




```
for iter in range(iterations):
    for i in range(m):
```

$$w_j = w_j - \eta \frac{\partial L}{\partial w}$$

$$= w_j - \eta \left(y^{(i)} - \hat{y}^{(i)} \right) x_j^{(i)}$$

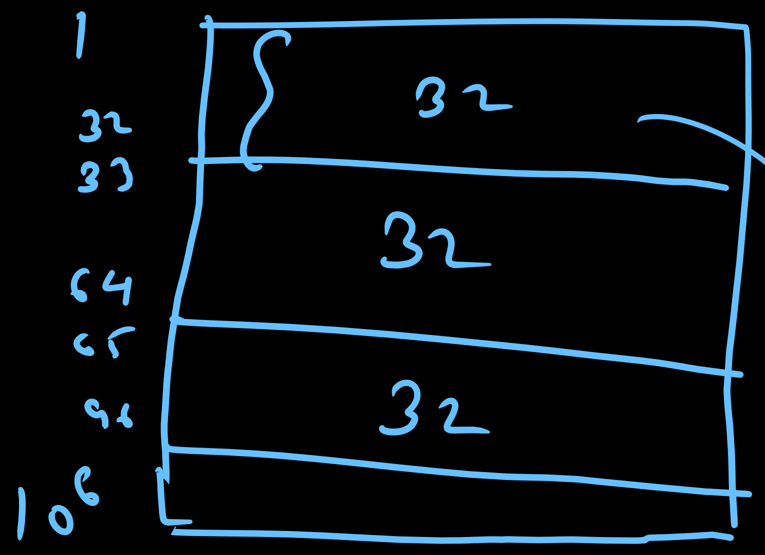
→ No vectorization

→ Update of w happens for each datapoint in each iteration

→ More updates, so it is noisy and oscillates and difficult to convergence.

Minibatch

$$\text{No of batches} = \frac{m}{\text{Batch size}}$$

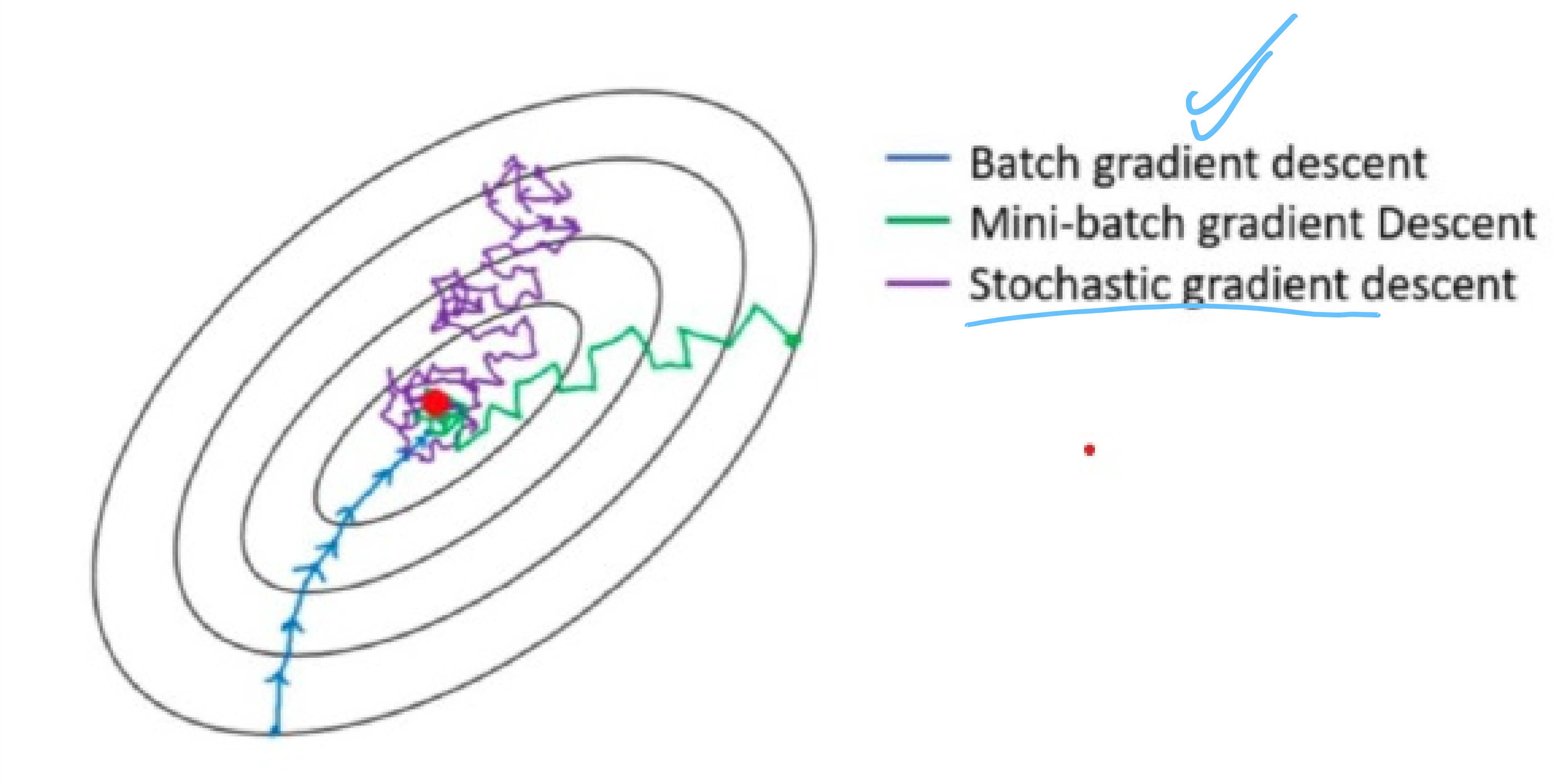


for iter in range(iterations):

for batch iter in range(No of batches):

$$w_j = w_j - \eta \frac{\partial L}{\partial w}$$

$$= w_j - \eta \frac{1}{bs} \sum_{i=1}^{bs} \left(y^{(i)} - \hat{y}^{(i)}\right) x_j^{(i)}$$


Batch gradient descent
Mini-batch gradient Descent
Stochastic gradient descent

- 1 (Stochastic GD)
- m Batch GD
- B/w [1, m) Minibatch
  - ↳ Batchsize = 32, 64, 128

1 Million [10^6]

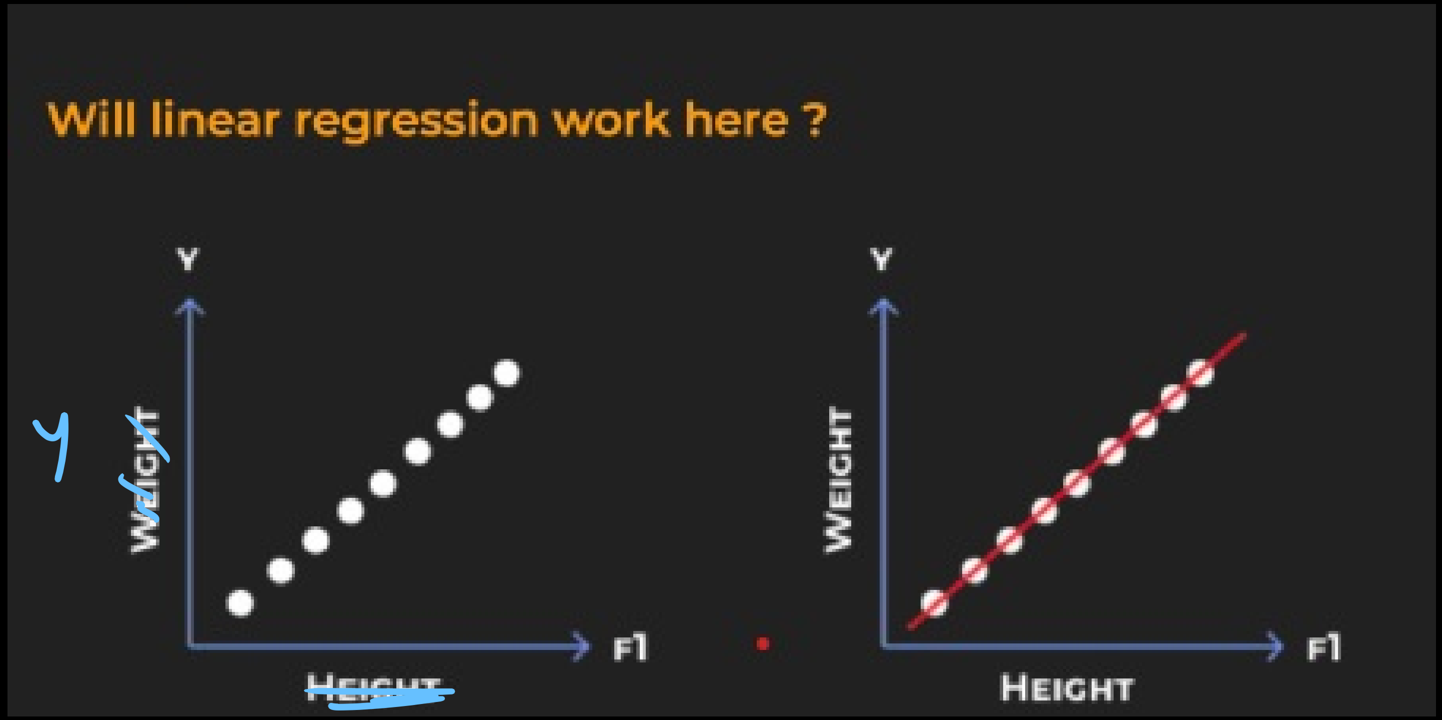
Memory

→ Vectorization

→ Updates will be < SGD

> BGD

→ Break until 22:19



y

x

$\hat{y} = \omega_1 x + \omega_0$

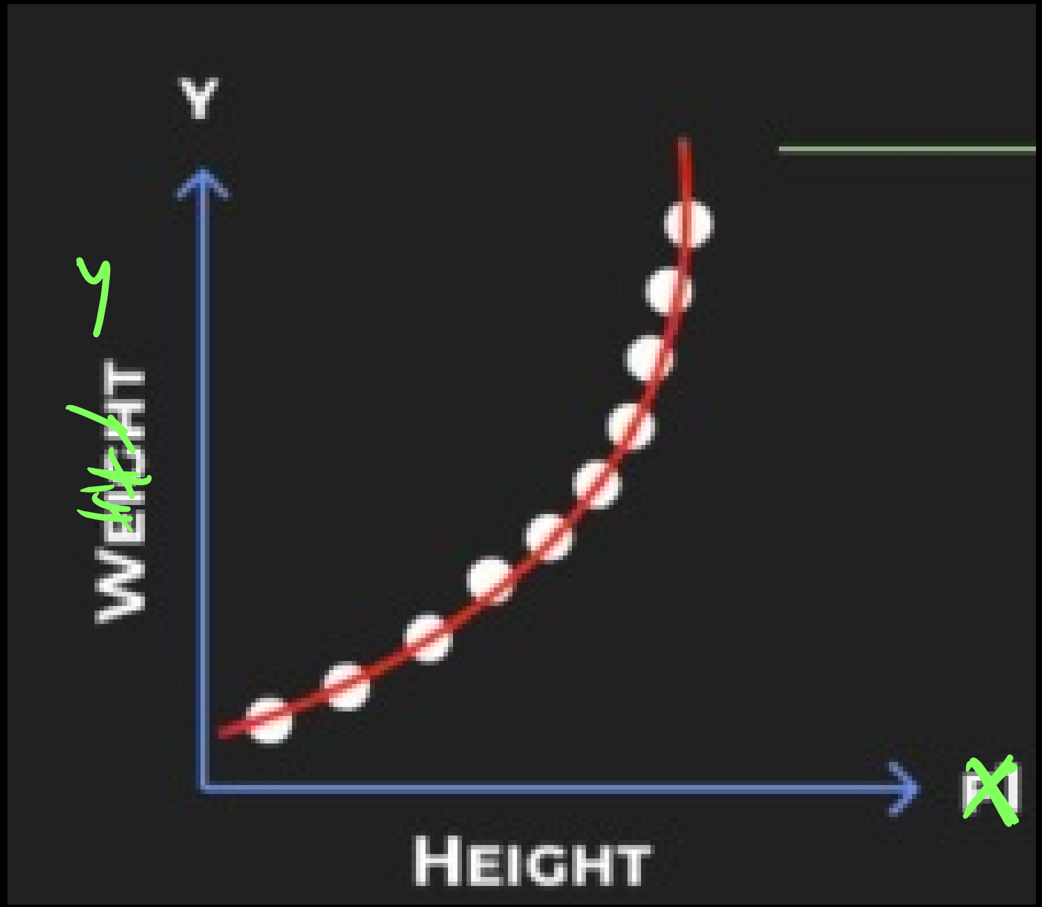
Polynomial Regression

$$\hat{y} = \overbrace{w_0 + w_1 x_1 + w_{12} x_1^2}^{f_1 \quad\quad f_2}$$

Plane

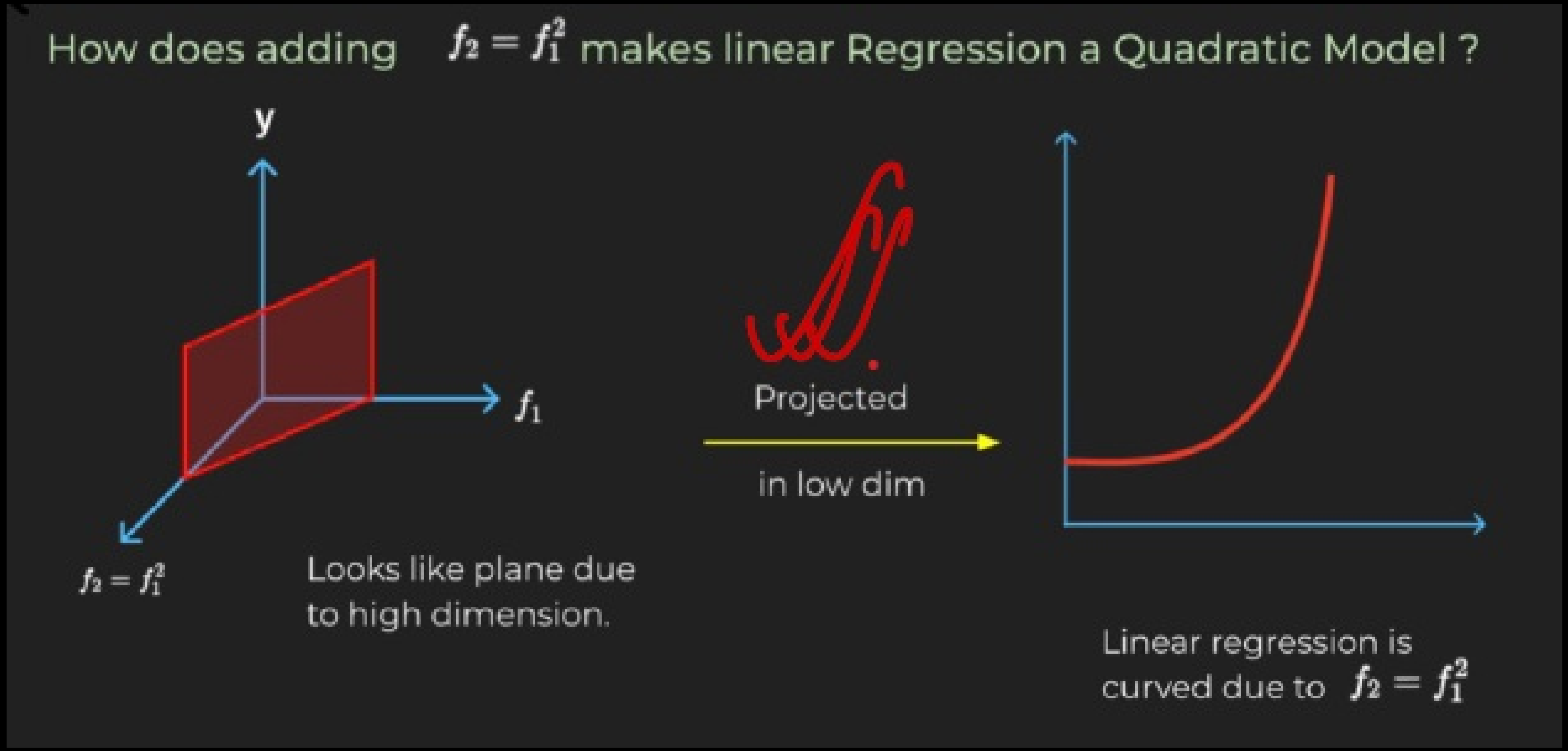
2 → 3 Hyperplane

n → n+1



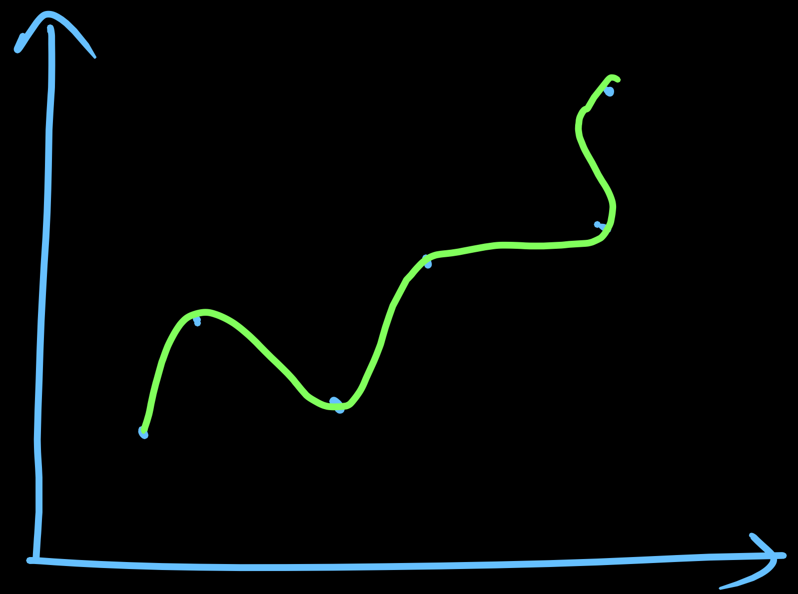
X

$$y = w_1 \overset{f_1}{x_1} + w_{12} \overset{f_2}{x_1^2} + w_0$$



$$x^2 \simeq \alpha x + \beta$$

Multicolinearity exist

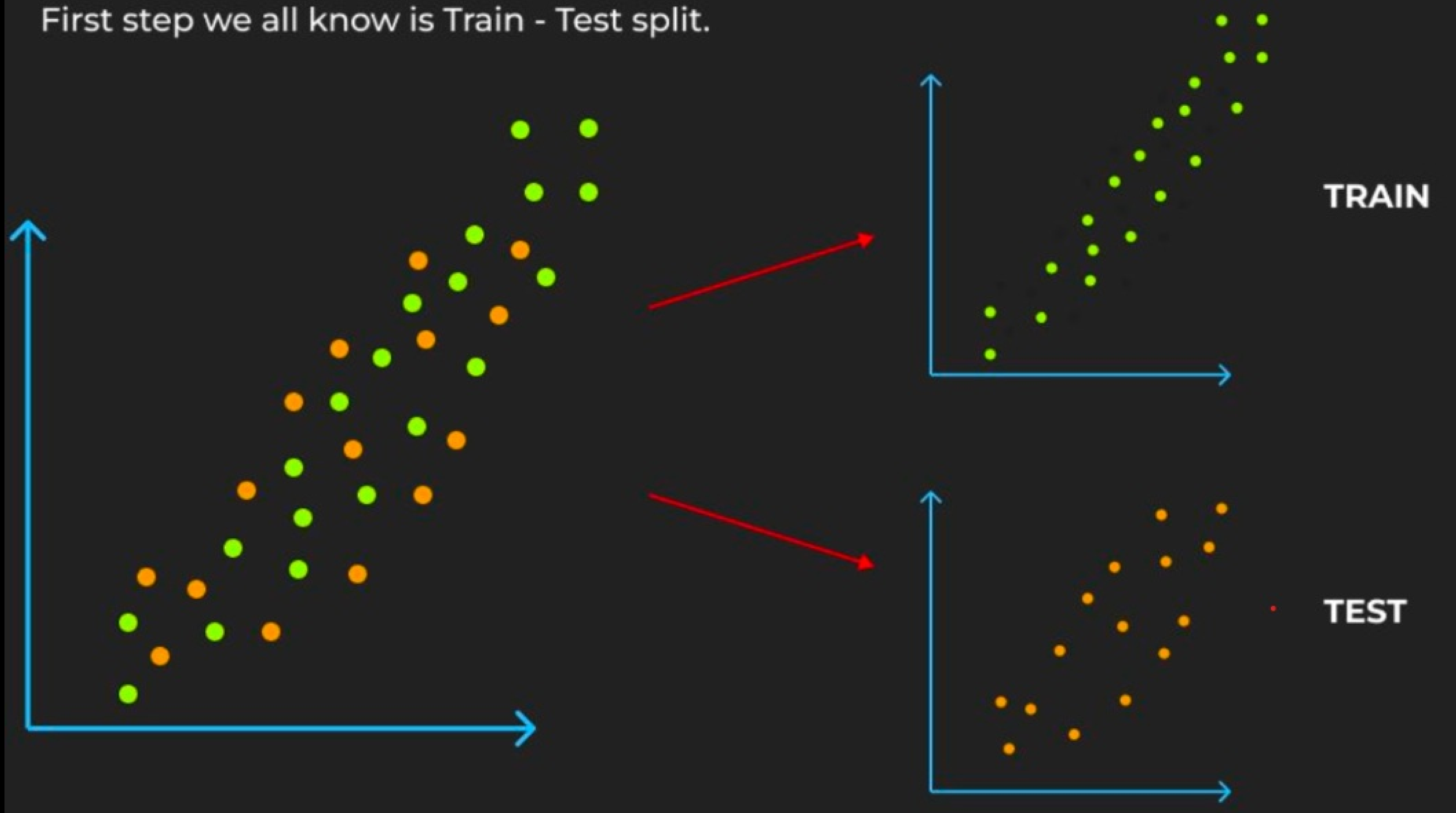


$$w_1 x_1 + w_2 x_2$$
$$+ w_{12} x_1^2 + w_{22} x_2^2$$
$$+ w_{13} x_1^3 + w_{23} x_2^3$$

$$\hat{y} = w_0 + w_1 x$$
$$+ w_{12} x^2$$
$$+ w_{13} x^3$$

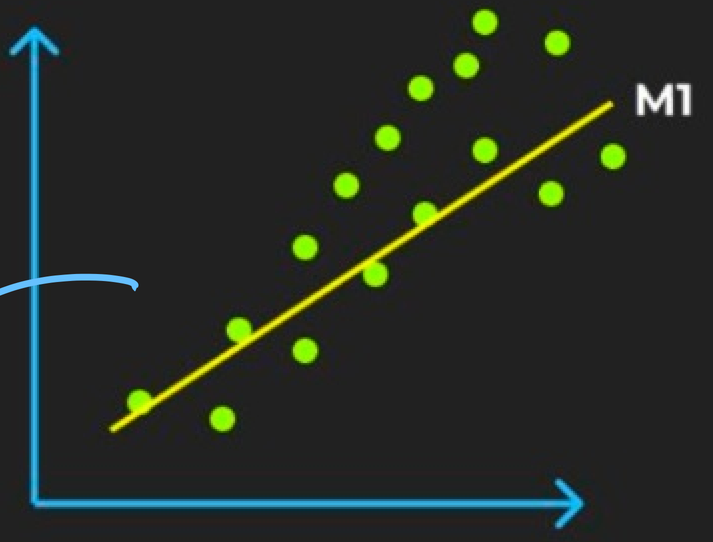
$$w_1 x_1$$
$$w_1 x_1 + w_{12} x_1^2$$
$$w_1 x_1 + w_{12} x_1^2 + w_{13} x_1^3 \; . \; -$$

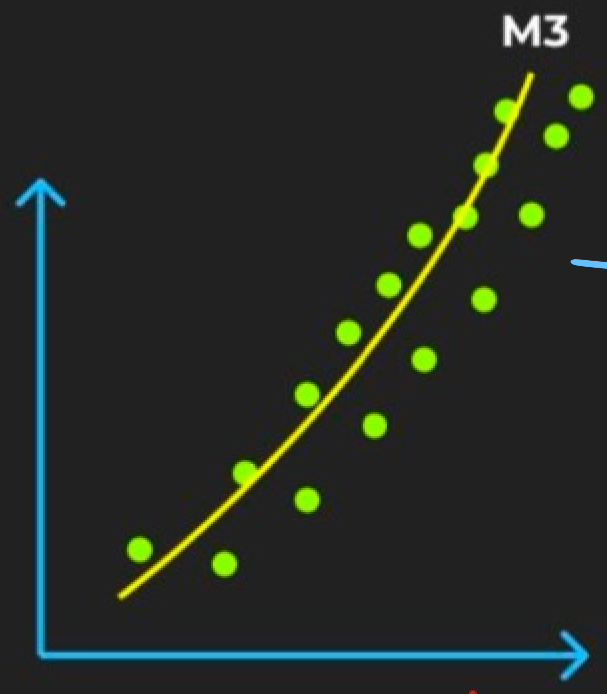
First step we all know is Train - Test split.



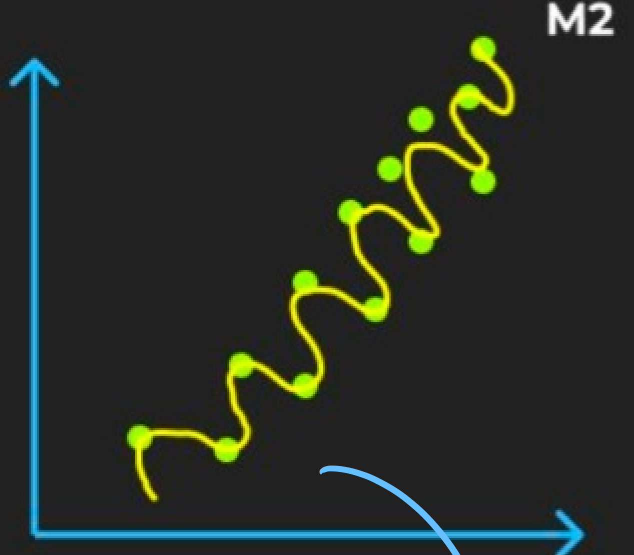
Training

$R^2 = 0.7$





$R^2 = 0.93$



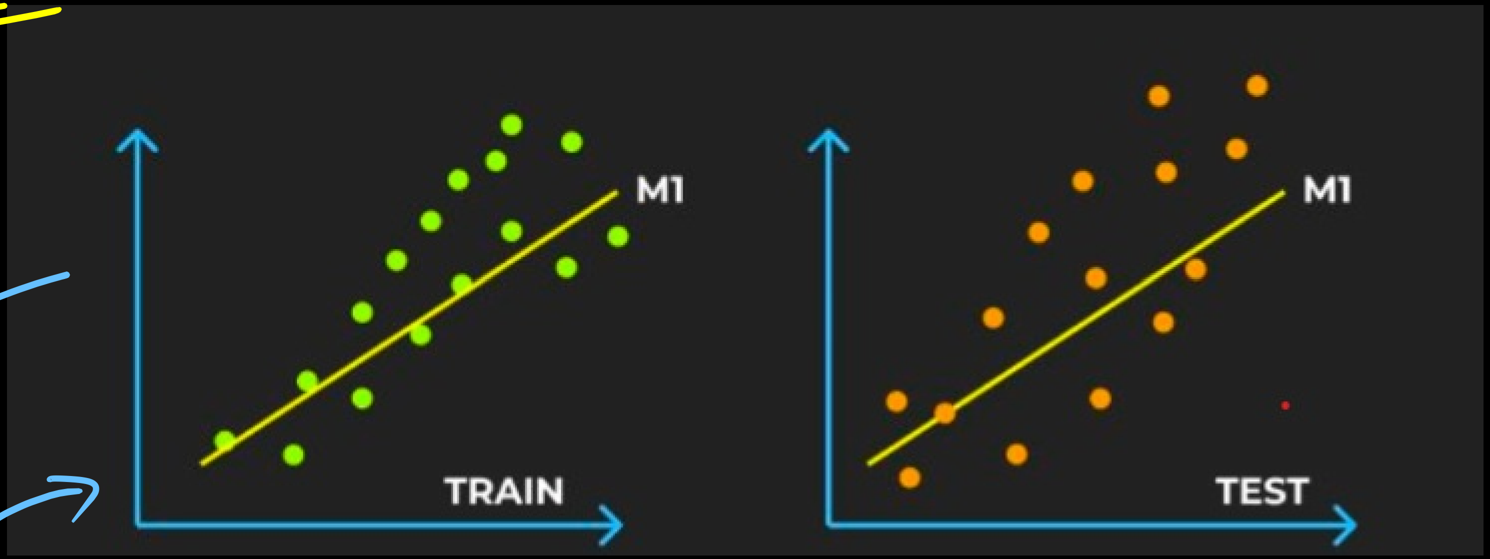
Which three models out of M1, M2, M3 do you think is best ?

$R^2 \simeq 0.997$

$M_2 > M_3 > M_1$

Underfitting

$R^2 = 0.7$

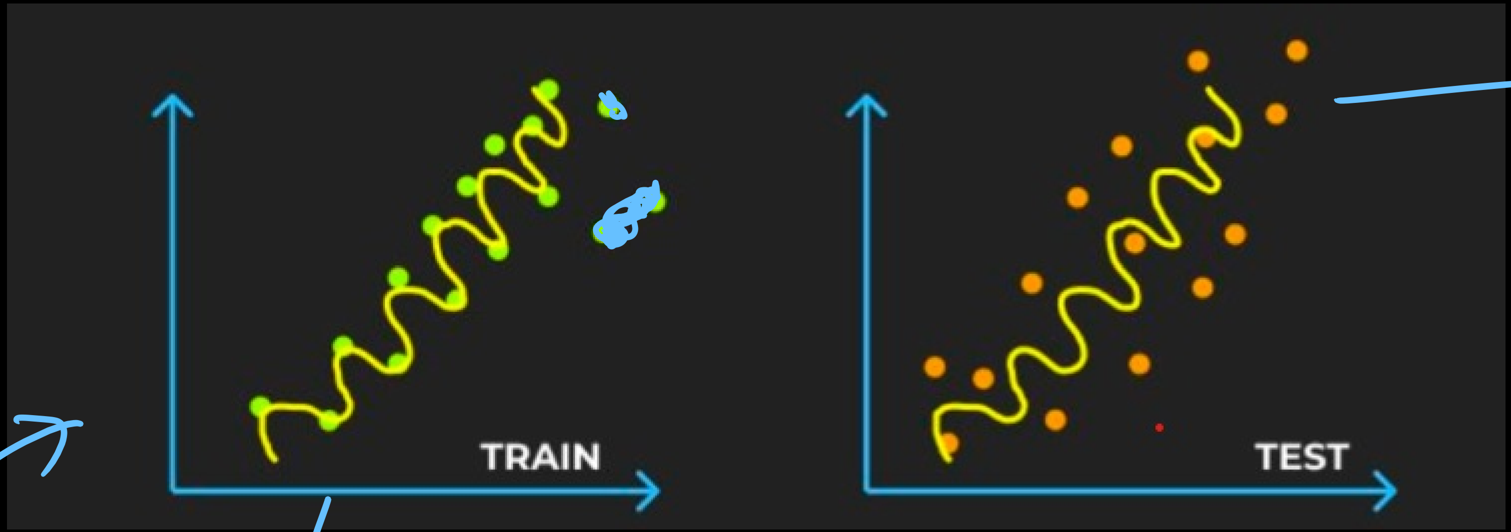


$$\boxed{\hat{y} = w_0 + w_1 x}$$

→ Does not fit well on <u>training data</u> <u>and test data</u>

Overfitting

→ Training performance is very good



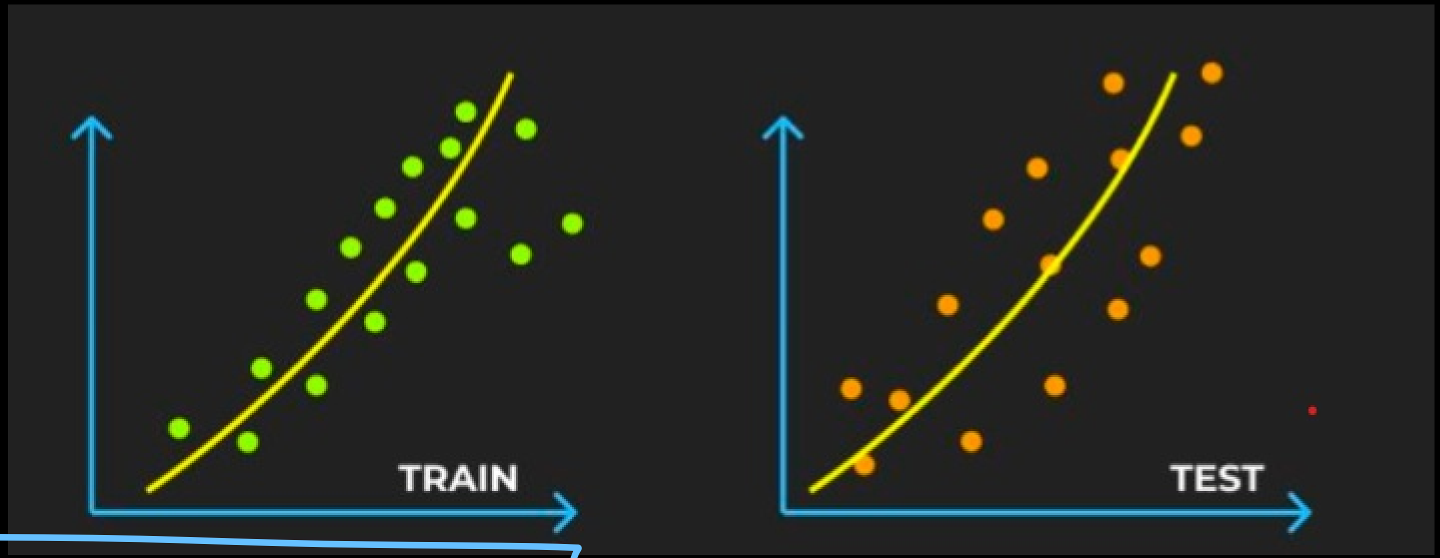
→ Test per is bad

$$\boxed{\hat{y} = w_0 + w_1 x_1 + w_{12} x_1^2 + w_{13} x_1^3 + w_{14} x_1^4}$$

→ Memorized at training time

→ Performance on test set is poor

Best fit



$$\boxed{\hat{y} = w_0 + w_1 x + w_{12} x^2}$$

→ Understood the concept

→ Perform well at test time

$w_1 x + w_0$

Simplest ↕ Complex

	Training	Testing
Underfit	Poor	Poor
Perfectly Fit	Great	Great
Overfit	Best	~~Decent~~ Poor

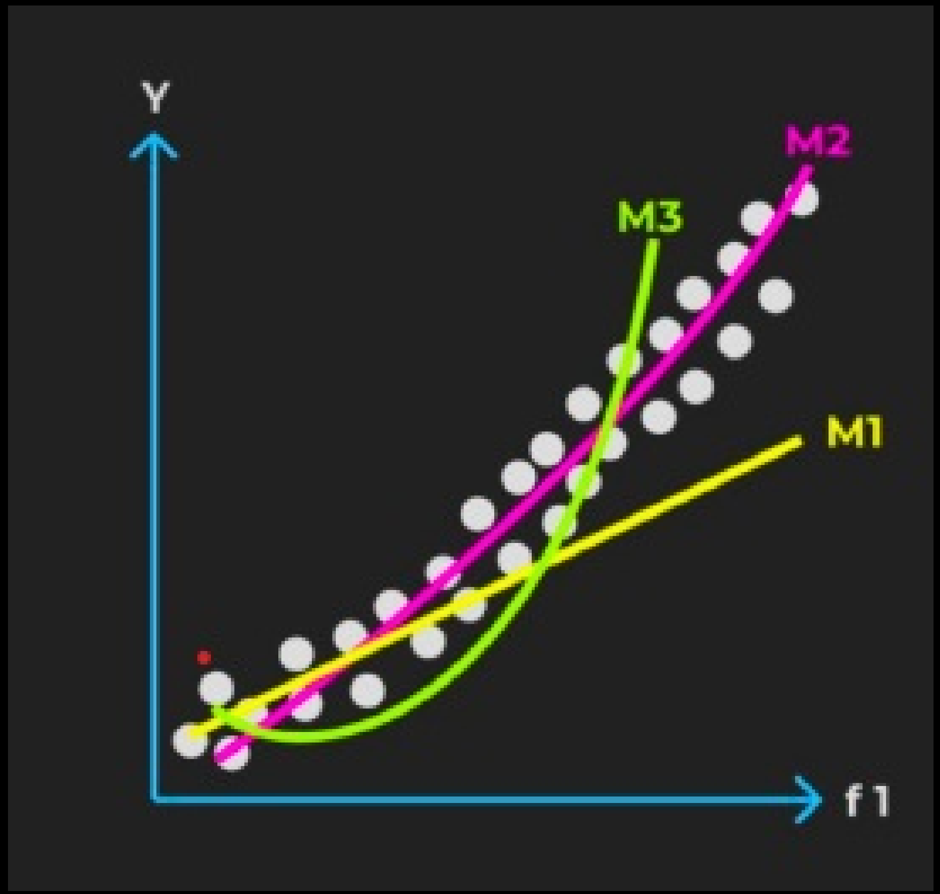
1) Underfit
2) Overfit

$$5x_1 + 10x_1^2 + 20x_1^3 + 30x_1^4$$

$$\hat{y} = 5f_1 + 0f_1^2 + 0f_1^3 + 0f_1^4 \rightarrow$$ Underfit

Ocam Razor

↳ Many solution to the problem but always choose simpler one



M1 : x_1 ← Linear

M2 : x_1, x_1^2 ← Quad

M3 : x_1, x_1^2, x_1^3 ← Cubic

Test Per: M1 → Bad

M2 → Good ✓

M3 → Bad

x_1	x_1^2	y

x_1	y

x_1^3

x_1^4