M T W T F S S

DATE

Machine Learning

- Supervised
  - Labelled data
    - Advantage
      - easy to understand
      - easy to train model
      - easy to predict on new samples
    - Dis Adv.
      - Labelling a data is challenging
      - finding relevant dataset (labelled) is tough.
- unsupervised
  - unlabeled data

Linear Regression

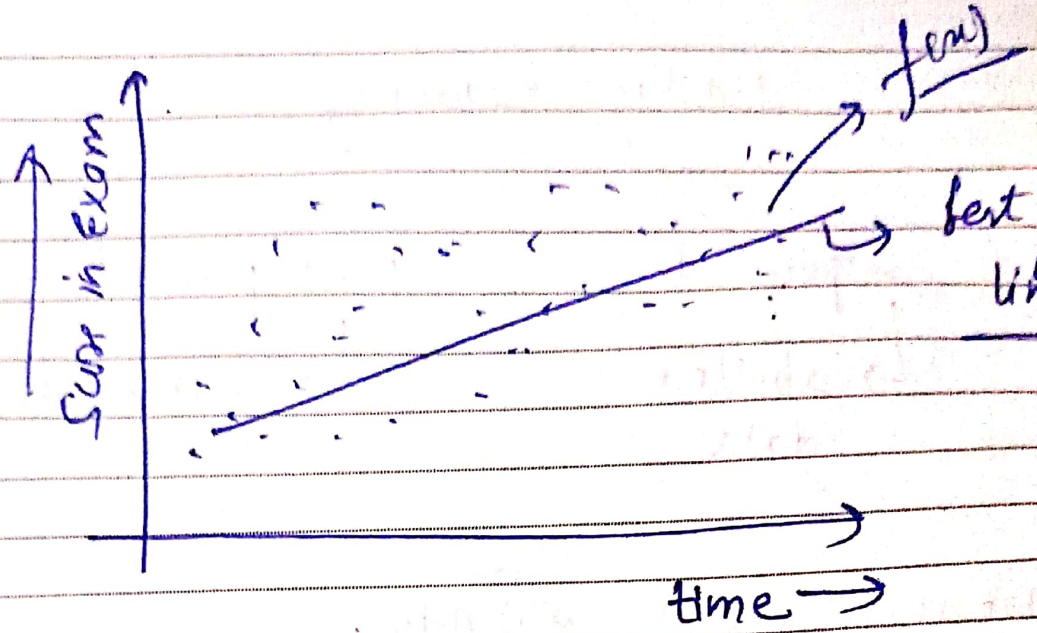
* Supervised ML model
* Basic ML model

eqn $y = mx + c$

example house price prediction

size of house (x)	price (y)
210m	1.5cr
150m	90 lakhs
-	- - . so on.

M T W T F S S DATE



Task is to minimize the error.

What is error function? → f(x) Cost function

MSE → Mean square error.

$$E = \frac{1}{n}\sum_{i=0}^{i=n}(y_i - \hat{y}_i)^2$$

y_i ↙ actual

$\hat{y}_i$ → predicted $(mx_i + c)$

Calculate gradient → differentiate

$$\frac{dE}{\partial m} = \frac{-2x_i}{n}\left(y_i - (mx_i + c)\right)$$

M T W T F S S

DATE

$$\frac{\partial E}{\partial c} = \frac{-2}{n}\left(y_i - (mx_i + c)\right)$$

Calculate gradient descent

$$m = m - L\frac{dE}{dm}$$

Learning rate → 0.0001 to 0.1

$$c = c - L\frac{dE}{dc}$$

Python implementation

```
def cost( Y, m, c, x) :
    n = len(x)
    cost = 0
    for i in range(n):
        cost += (Y[i] - (m x[i] + c))**2
    return cost/n
```

M T W T F S S

DATE

```
def gradient(m, c, x, Y):
    n = len(x)
    dc = 0
    dm = 0
    for i in range(n)
        dm += -2 * X[i] * (Y[i] - (m[xi] + c
        dc += -2 * (Y[i] - (m(xi] + c))

    return dm/n, dc/n

def gradient descent(m, x, Y, c, lr, epoch)
                                  learning rate
    cost_list = []
    for i in range(epoch):
        dm, dc = gradient(m, c, x, Y)
        m = m - lr dm
        c = c - lr dc
        lost = cost(m, x, c, Y)
        lost_list.append(cost)
    return m, c, costlist
```

M T W T F S S DATE

```
m=0
c=0
lr = 0.0001
epoch = 100
m, c, list = gradient_descent(m, x, y, c, lr, epoch)
y_pred = X * m + C
plt.scatter(X, Y)
plt.plot(X, Y_pridict, color='red')
plt.show()
```