Machine learning

→ Ability of algorithms to learn from data

→ logic is not provided by programmer, but the machine understand it by itself

→ We provide a ~~list~~ input and output and train it with a large amount of inputs, then deploy it

Data —(Applied for a purpose)→ Information —(build & process)→ Knowledge

input data → x-train
output data → y-train
} Training data

input → x-test
output → y-test
} testing data

Nominal Data

→ No ordering
→ Data having label but not order

eg Gender
→ Male
'm → Female

Ordinal Data

→ Data having natural order

eg How do you feel?
1- Very happy
2- happy
3 Sad
4 Very Sad

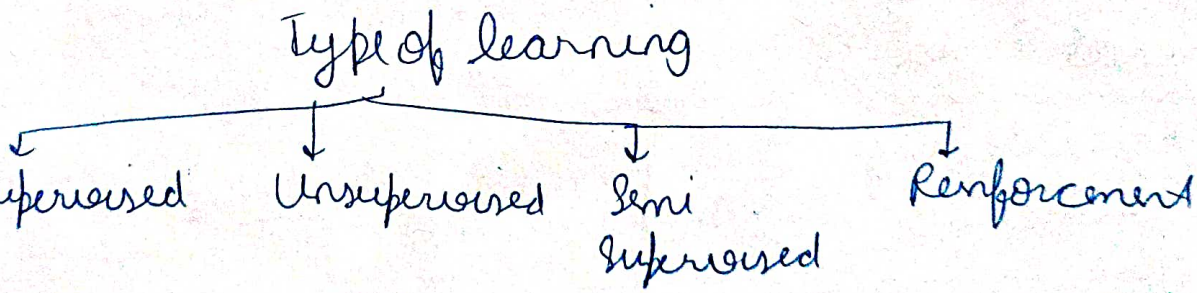
Interval data
data having intervals b/w each value

Eg Temp?
30°
40°
60°

Binary data
Only 2 labels
0, 1
Yes / No

Time data
Date, year, month, hour

LEARNING for a MACHINE
- A machine is said to be learning from past experience with respect to some class of task.



Supervised learning

when model is getting trained on a labelled dataset
labelled dataset is one having both input & output

- Classification
  - ↳ Discrete output
  - eg 0 or 1, Yes/No
- Regression
  - → Continuous output parameter
  - eg (0-5)
    - ↓
    - any value
    - 0.1, 0.2, 0.3, 1.3, 2.4

Unsupervised learning

→ No output / we don't give target to our model while training

- Clustering
  - → group the data sharing the similar properties
- Association
  - → try to look for best connection b/w parameters of large data

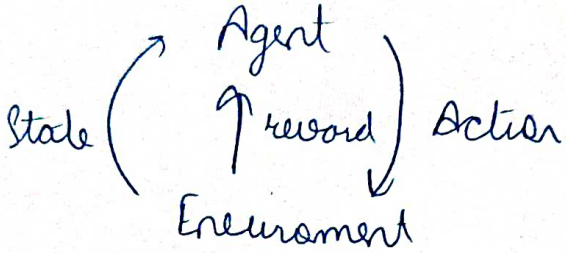
Semi supervised learning

→ some data has output labels, some have not
both labelled & unlabelled data

Reinforcement learning

In this model keeps on increasing its performance using a reward feedback to learn the behaviour/ pattern

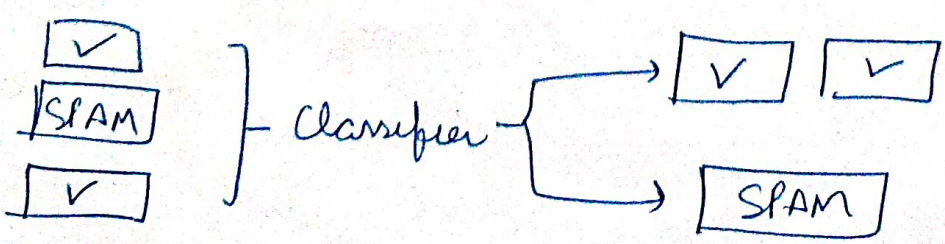
eg self driving car



* learn from mistakes

Classification	Regression
→ Email Spam detection (Yes/No)	→ House price prediction
→ Image recognition (✓/✗)	→ Health insurance assessment
→ Stock price prediction	

eg email spam detection



AI (Artificial Intelligence)

Intelligence
↳ ability of reasoning
↳ Reward based learning
↳ Generalized learning.

Subsets of A.I.

→ Machine learning → Deep learning
→ Data science
→ Natural language processing
→ Computer vision (Dealing with images/video)
→ Robotics (Hardware + Software)

* Neural network
↳ networks of neurons
single unit of an algorithm

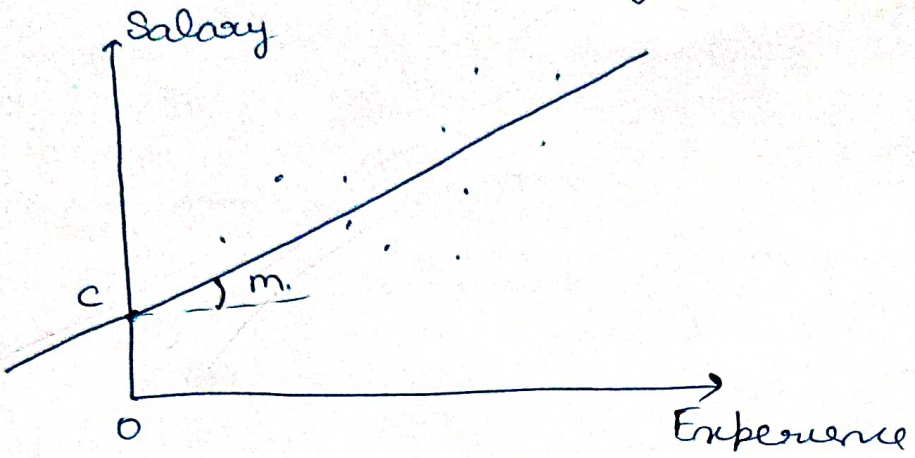
Data scientist jobs
→ understanding/problem
→

Ist ML ALGORITHM

* LINEAR REGRESSION

→ Hypothesis, $y = mx + c$
↳ straight line

eg



At time of training, known parameters
- y (output)
- x (input)

unknown parameter
- m (slope)
- c (y intercept)

Target → To have best fit line, will predict output with least error

↳ Steps

① Forward propogation (pick random value of m, c)

② Calculate error by cost function

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

$\hat{y}$ → predicted output
y → actual output
n → no. of instances

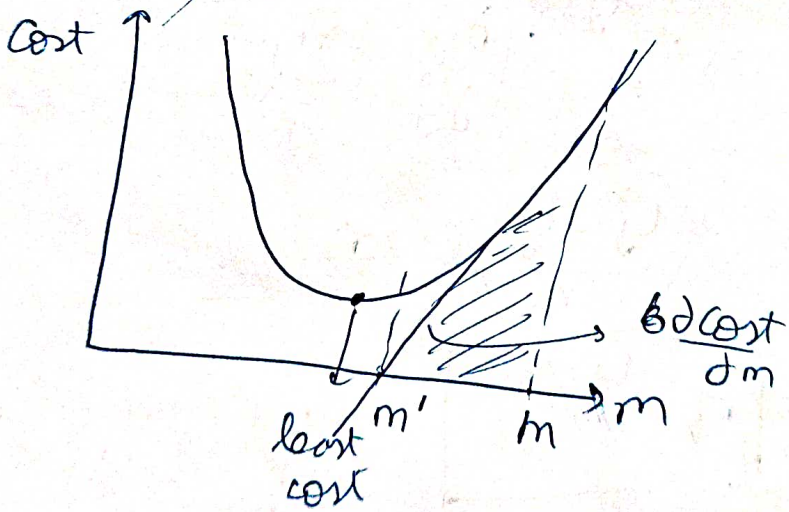
3- Gradient Descent algorithm

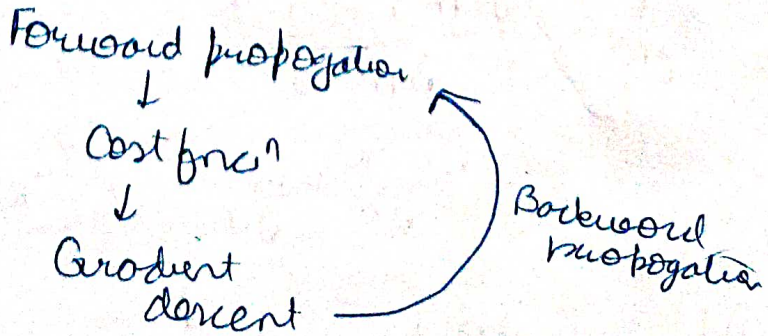
↳ used to get optimal value of m and c so that we can get least error or cost function

$$m' = m - \alpha \left(\frac{\partial Cost}{\partial m}\right) \rightarrow$$ To get minimum cost value w.r.t. m and c

$$c' = c - \alpha \left(\frac{\partial Cost}{\partial c}\right) \rightarrow$$

(learning rate) α is used to make gradient fast or slow accordingly





Maths behind linear regression

$$m' = m - \alpha \frac{\partial Cost}{\partial m}$$

$$\Rightarrow \frac{\partial Cost}{\partial m} = \frac{\partial Cost}{\partial f} \cdot \frac{\partial f}{\partial m} \quad \text{--- (1)} \qquad \{f : \hat{y} = mx + c\}$$

$$\Rightarrow \partial f = \frac{\partial Cost}{\partial f} \Rightarrow \partial \frac{(\hat{y} - y)^2}{\frac{2n}{\partial \hat{y}}} \qquad (\because \partial f = \hat{y})$$

$$\Rightarrow \frac{2}{2n}(\hat{y} - y) \cdot \frac{\partial (\hat{y} - y)}{\partial \hat{y}}$$

$$\Rightarrow \left(\frac{\hat{y} - y}{n}\right) \cdot \left(\frac{\partial \hat{y}}{\partial \hat{y}} - \frac{\partial y}{\partial \hat{y}}\right)$$

$$\Rightarrow \left(\frac{\hat{y} - y}{n}\right)(1 - 0)$$

$$\boxed{\partial f = \frac{\hat{y} - y}{n}} = \frac{\partial Cost}{\partial f} \quad \text{--- ✓}$$

Now $\frac{\partial f}{\partial m} = \frac{\partial (mx + c)}{\partial m}$

$$\Rightarrow \frac{\partial mx}{\partial m} + \frac{\partial c}{\partial m} \Rightarrow x + 0 \Rightarrow (x)$$

$$\frac{\partial f}{\partial c} \Rightarrow \frac{\partial mx + c}{\partial c} \Rightarrow (1)$$

from (1)

$$\partial m = \frac{\partial Cost}{\partial m} = \frac{\partial Cost}{\partial f} \cdot \frac{\partial f}{\partial m}$$

uses equation

$$\partial m = \left(\frac{\hat{y} - y}{n}\right) x$$

$$\partial c = \left(\frac{\hat{y} - y}{n}\right) \cdot 1$$

→ Gradient descent

$$\partial f = \frac{f(x) - y}{n}$$

$$\partial m = \partial f * x$$

$$\partial c = \partial f$$

$$m' = m - \alpha \partial m$$

$$c' = c - \alpha \partial c$$

$$m' = m - \alpha\left(\frac{y' - y}{n}\right) x$$

$$c' = c - \alpha\left(\frac{y' - y}{n}\right)$$

→ Updating parameters

Multiple linear Regression

→ Same thing as linear but with multiple inputs

→ $y = w_1 x_1 + w_2 x_2 + w_3 x_3 + b$

b - bias / constant value

$$\partial m = \frac{\partial cost}{\partial b} \times \frac{\partial b}{\partial m} \Rightarrow \frac{\hat{y} - y}{n} * x$$

$$\partial w_1 = \frac{\hat{y} - y}{n}(x_1)$$

$$\partial w_2 = \frac{\hat{y} - y}{n}(x_2)$$

⋮ ⋮ ⋮

* Linear Regression Assumptions

→ linear relation b/w input & output

$y = mx + c$

→ Homoscedasticity

As, prediction $= \hat{y} = mx + c$ ①

residuals, error $= (\hat{y} - y)$ ②

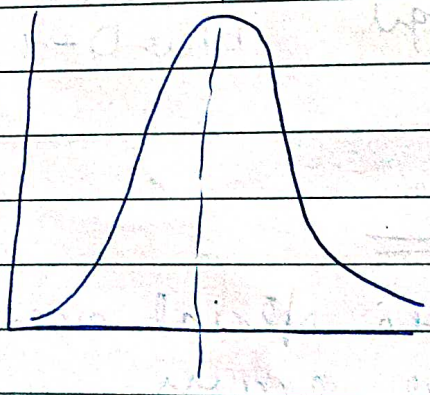
So graph b/w ①, ② will be random

no relation b/w ①, ②

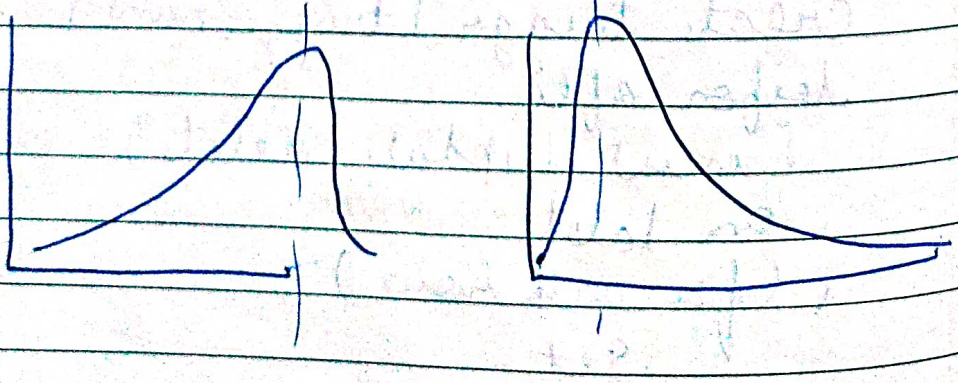
→ Normality in residuals

↓

should have normal (gaussian) distribution (symmetric)



skewed distribution (high error)



DATE : __/__/__
PAGE : _____

* OLS method

→ Ordinary least square ←

R-squared value

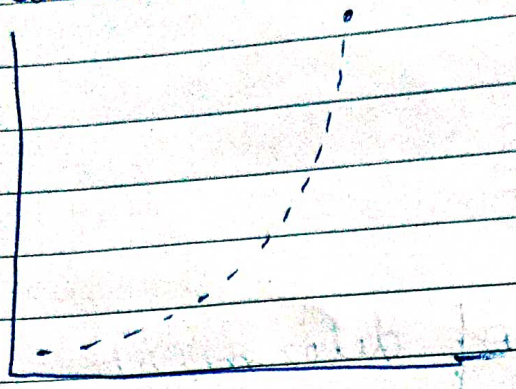
$$R^2 = 1 - \frac{RSS}{TSS}$$

R^2 → coeff^n of determination
RSS → sum of squares of residuals
TSS → Total sum of square

Polynomial linear Regression

→ If data is not linear

eg →



As in linear

$y = mx + c$

$x \quad x^2 \quad x^3$

So eqn can be

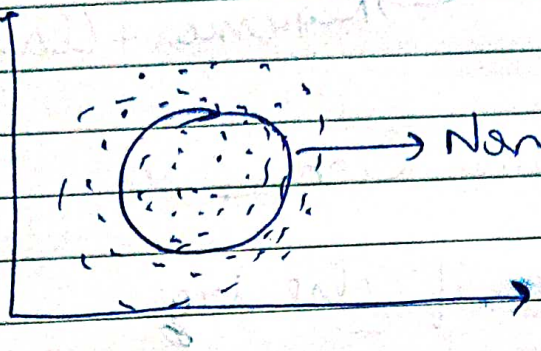
$$\hat{y} = \underbrace{m_0 x_1^0}_{c} + m_1 x_1 + m_2 x_1^2 + m_3 x_1^3$$

We can get curve using degrees, x_1^2, x_1^3

* LR → $\hat{y} = mx + c$
* MLR → $\hat{y} = m_1 x_1 + m_2 x_2 + m_3 x_3 + c$
* PLR → $\hat{y} = m_1 x_1 + m_2 x_1^2 + m_3 x_1^3 + c$

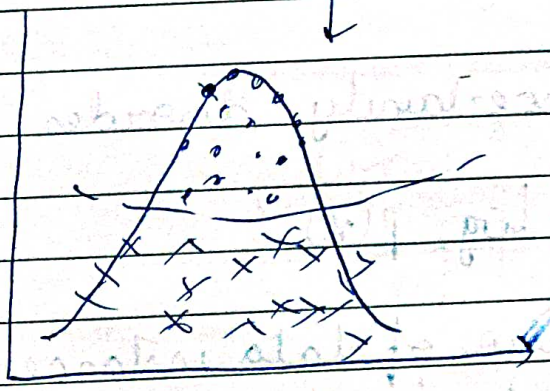
DATE :__/__/____
PAGE :_____

Support vector machine Algo



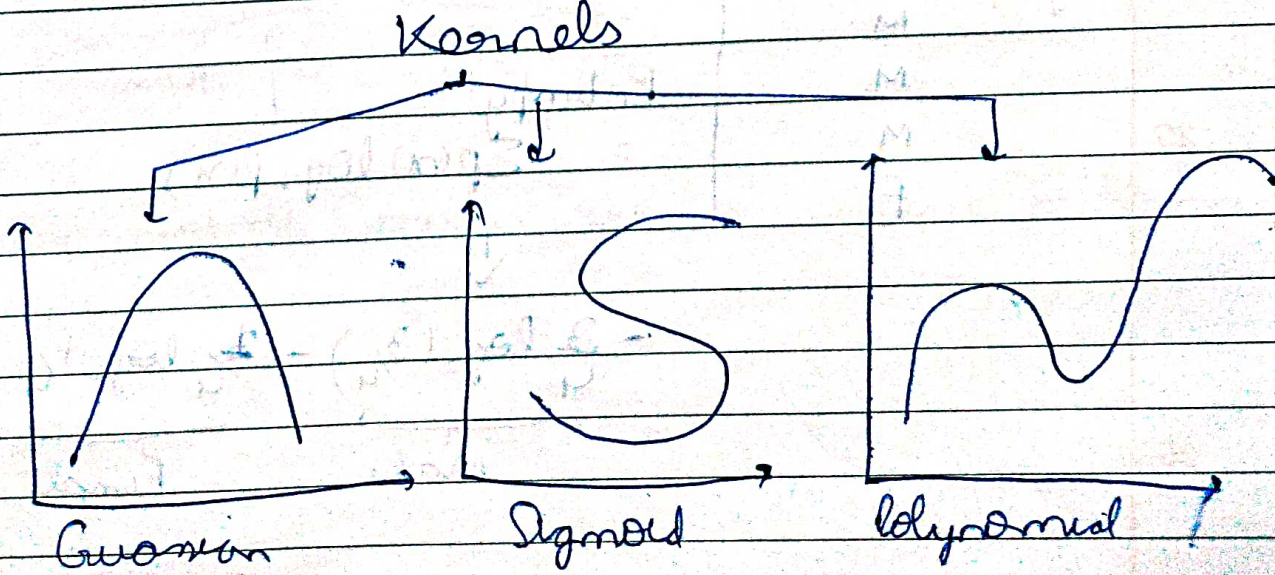
→ Non linear Seperable data
$y = x_1^2 + x_2^2$ (circle)

→ We mapour data to kernel



* SVM takes data and add dimensionality to make data linearly seperable using kernels

* Then you use hyperplane



DATE :__/__/__
PAGE :

Decision Tree Algo

↳ Regression + Classifier

called as ID3 → Iterative Dichotomiser

* conditional algo like if-else type

↳ Info gain
↳ Entropy

Randomness, uncertainty, disorder

* Entropy :- $-\Sigma p(x) \log_2 p(x)$

$p(x)$ → fraction of data instances of particular type

Probability of a particular data x

eg

Gender
M
M
M
F

- Entropy = $-\Sigma p(x) \log_2 p(x)$

$$= -\frac{3}{4}\log_2(3/4) - \frac{1}{4}\log_2 1/4$$

Male Female

DATE : __/__/__
PAGE : ____

* Information Gain
  ↳ how much info a feature / input parameter gives us about output / target parameter

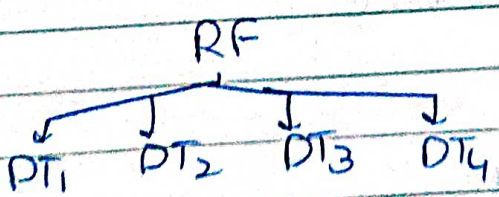
$$\text{Info Gain}(P_1) = \text{Entropy}_{(\text{Target})} - \text{Entropy}_{(P_1, \text{Target})}$$

P_1 - Parameter

Random forest Algo

↳ work as both regressor & classifier

→ Ensemble model → Bagging model
↓
use can haven no. of decision trees



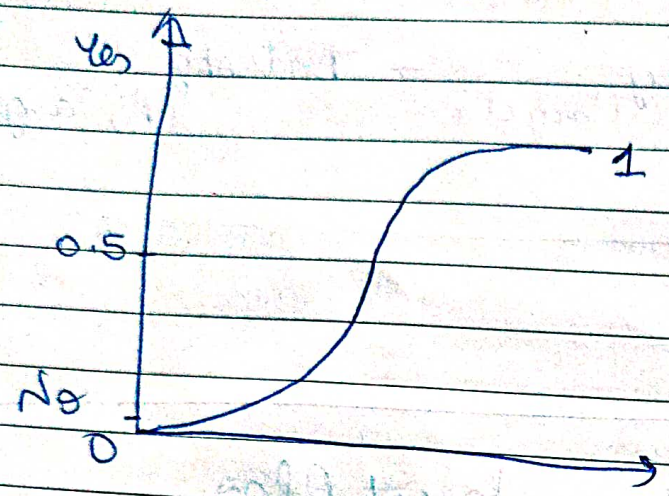
→ Enhancement of Decision Tree to increase performance

CLASSIFICATION ALGORITHMS

DATE : __/__/__
PAGE :

→ Logistic Regression algo

Uses a sigmoid function



$$g(z) = \frac{1}{1+e^{-z}}$$

$$z = \hat{y} = mx+c \quad (y \text{ predicted})$$

$$g(z) = \frac{1}{1+e^{-(mx+c)}}$$

* Case-1

y is very large

$$g(z) = \frac{1}{1+e^{-z}}$$

$$\hat{y} \uparrow \rightarrow -\hat{y} \downarrow \rightarrow e^{-\hat{y}} \downarrow$$

$$g(z) \approx 1$$

Case ②

if $\hat{y} \downarrow \rightarrow -\hat{y} \uparrow \rightarrow e^{-\hat{y}} \uparrow$

$\therefore g(z) \approx 0$

Cost function

$$J = \text{Cost} = \frac{1}{n} \sum_{i=1}^{n} [L(y_i - \hat{y}_i)]$$

$\rightarrow L$ = loss

$$\text{loss} = -[y \log \hat{y} + (1-y) \log (1-\hat{y})]$$

* If $y = 1$

$$L = -y \log \hat{y} \Rightarrow -\log \hat{y}$$

If we want loss to minimum,

$\log \hat{y} \uparrow$

$\hat{y} \uparrow \Rightarrow 1$ (max prob)

* If $y = 0$

$$L = -(1-y) \log (1-\hat{y})$$

$$= -\log (1-\hat{y})$$

loss minimum $\rightarrow \log (1-\hat{y}) \uparrow$

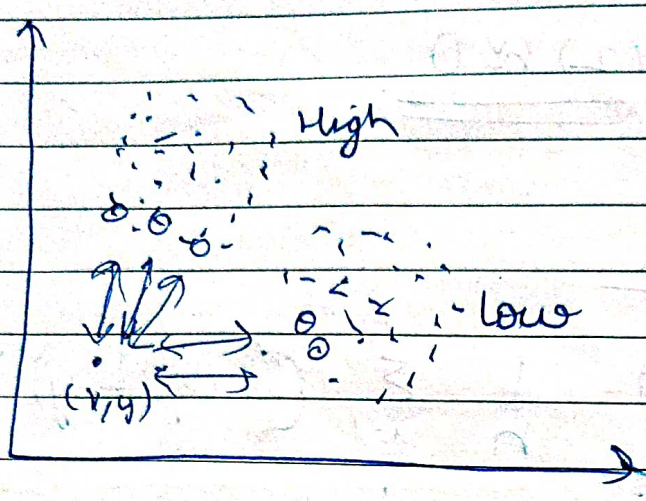
$(1-\hat{y}) \uparrow$ $\quad \hat{y} = 0$

(max prob)

DATE :__/__/
PAGE :

K-NN algorithm

K-nearest neighbours



→ uses euclidean distance

$$\sqrt{(x_{test} - x_{train})^2 + (y_{test} - y_{train})^2}$$

It will find k nearest neighbour and determine whether pt. lies where in decision

~~Naive Bias~~ Naive Bayes Classifier

Coin → (event)

T H → Sample space)

{T, H}

2 toss

↳ {HH, HT, TH, TT}

P(at least one head) = 3/4

DATE : __/__/__
PAGE : _____

P(2head) = 1/4

P(1head, 1Tail) = 1/4

* Bayes Theorem

$$P(A/B) = \frac{P(B/A) * P(A)}{P(B)}$$

Event
↙ ↘
E_1 E_2

P(both E_1, E_2 happening) = $P(E_1) * P(E_2)$

P(either E_1 or E_2 happened) = $P(E_1) + P(E_2)$

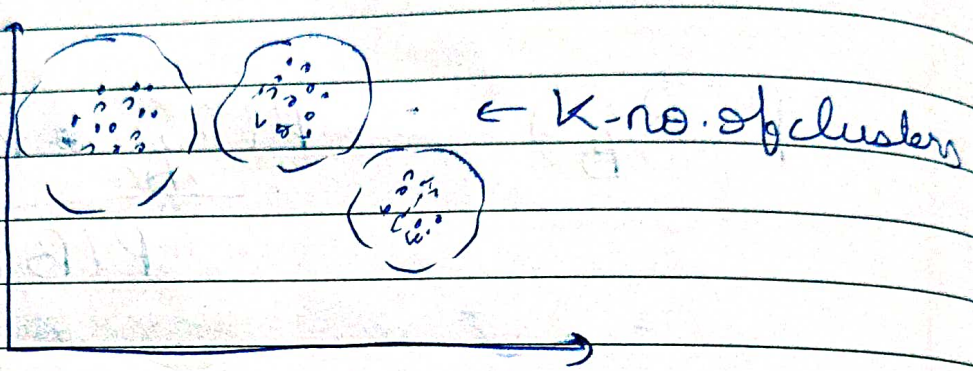
DATE : __/__/__
PAGE :

* Unsupervised learning

→ Clustering Algorithms

① K means Clustering

↳ also called hard clustering



Steps

→ Random Initialization
→ Assign pt. to nearest neighbour
→ Updating mean coordinate
→ Repeating process until cluster is found

* → This algo. will categorize the items into k groups or clusters of similarity
To calculate similarity, we use euclidean distance.

DATE : __/__/____
PAGE : _____

* Feature engineering

→ data cleaning
→ Scaling of data
    ↳ from sklearn.preprocessing import MinMaxScaler
        ↳ x = pd.DataFrame(MinMaxScaler().fit_transform(x))
→ Correlation
    ↳ heatmap or
    ↳ Extra Tree classifier
        (from sklearn.ensemble imp ExtraTreeClass
        (model = ETC())
        (model.fit(x, y))
        (features_imp = pd.Series(model.feat_imp)
        (feature_imp.nlargest(1)

    ↳ Select K best method

* PCA (Principal component Analysis)

→ To reduce data parameters to efficient amount
* suppose we have a data of 100 dimensions, we can not plot it as it is, we use PCA