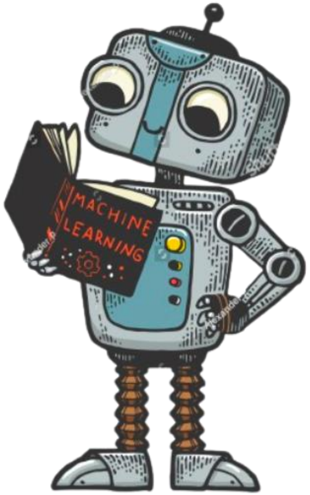
ML: K Nearest Neighbours
KNN-2

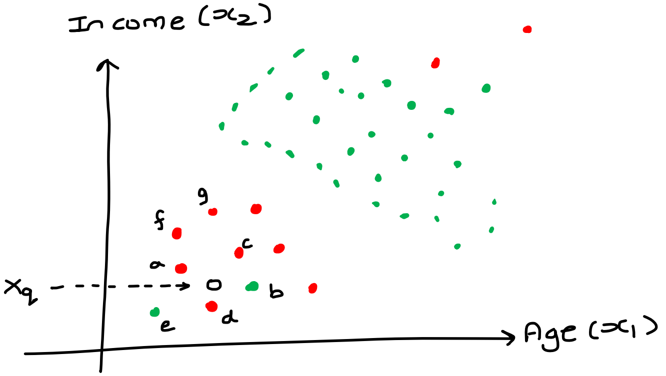
Sumit.shukla_1@scaler.com



(Supervised ML-Algo)

Summary

KNN



old customers / existing customers

	x_1 Age	x_2 Income	y Default?
A			Y
B			N
C			Y
D			N
E			Y
⋮			

y → Target variable (Dependent var)

→ Historical Data

New customer → (x_q) < Age = 19, Income = 10K

Steps of KNN

1. Compute distance from the query data point to all other points

$d(x_q, a) \rightarrow 1$

$d(x_q, b) \rightarrow 1.5$

⋮

2. Sort the data points according to the corrosponding dinstances

$x_q \rightarrow$ [a, b, d, e, c, f, g . . . . .]

3. choose k- nearst neighbour (K=3)

$\boxed{k=4}$ < Y = 2, N = 2 > split Brain situation

4. voting

a → Default = Y
b → Default = N
d → Default = Y

< y = 2, N = 1 > [x_q → default = Yes]

k = default = 5

$3 \leq k \leq 11$

k = odd value

KNN – Non-parametric algo

m. Budget	Sales

(1) Linear Regression → Train Data (x, y) → [Linear Reg. Algo] → Best weights (parameters) $\bar{W}, w_0$ → prediction

(Learning the best fit line)

line → $\bar{W}x + w_0$

Sales = w(m. Budget) + w0

estimate sales

1000

(2) Logistic Regression → Train Data (x, y) → [Logistic Reg. Algo] → Best weights (Parameter) $\bar{W}, w_0$ → predictions

(Learning the best seperating hyperplane)

$\bar{W}x + w_0$

parametric

KNN → No parameter is being learnt

non-parametric

Hyperparameter

In the case of KNN to predict the outcome of any new data point, we need to iterate through the entire training data and here the algo doesn't learn any parameters

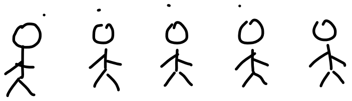
(1) parameter → this is what the algo learns from the data $(\bar{W}, w_0)$

(2) Hyperparameter → Has to be provided by the user and the algo will behave differently for different values of Hyper-parameter

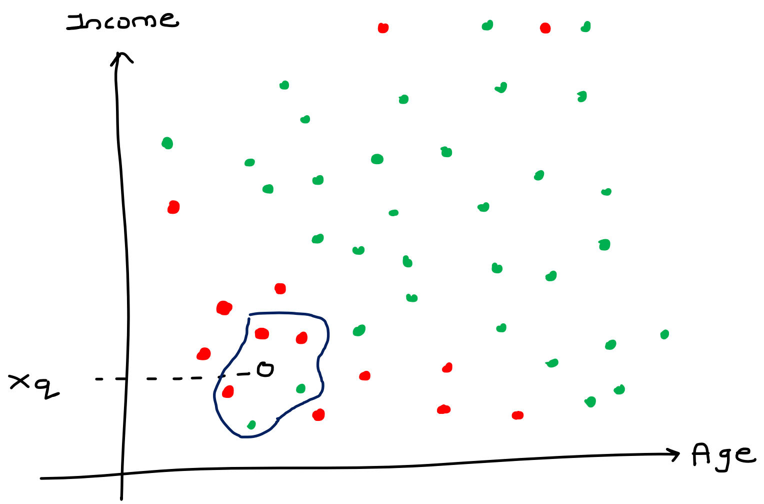
K impact on Bias & variance

K=1 (we are dependent on few neighbours)

overfitting



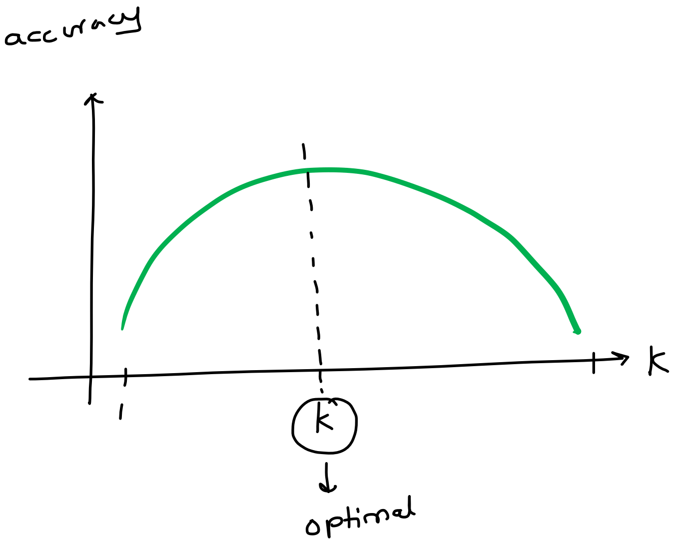
2 B = modi
3 = R.G



In case k is very small, we are being dependent on few neighbours and if any of the neighbour change, the outcome may get impacted ⟶ overfitting [High variance, Low Bias]

K= 100 In this case when k is very high, we are being dependent on so many neighbours and the change in one or two neighbours will not impact my final outcome but the results are over generalised and hence very high error is expected

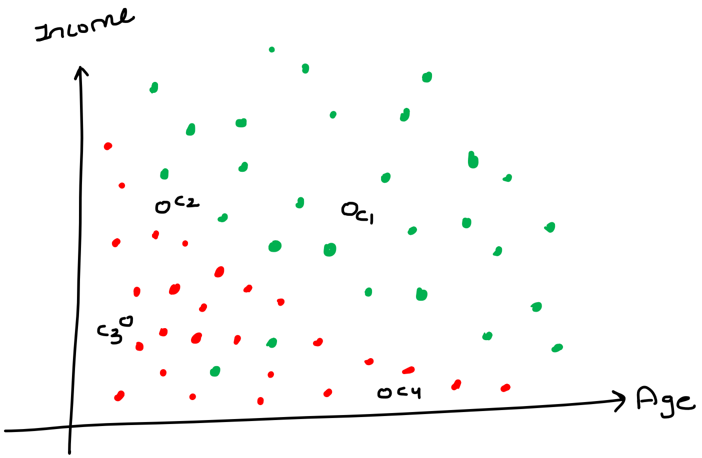
underfitting [Low variance, High Bias]



Hyperparameter Tuning

	Age	Income	y
c1			
c2			
c3			
⋮			
c100			

Training Data



	Age	Income	y
c1			
c2			
c3			
⋮			
c10			

Testing

for each data point in test data we will run the entire KNN steps

$c_1 \longrightarrow$ KNN (Training data)

$c_2 \longrightarrow$ KNN (" ")

$c_3 \longrightarrow$

⋮

Time complexity of KNN

x_1	x_2	x_3	$\cdots$	x_d

$\vdots$

n

$x_q \rightarrow [x_{1q}, x_{2q} \cdots x_{dq}]$

$$D = |x_1 - x_{1q}| + |x_2 - x_{2q}| + \cdots + |x_d - x_{dq}|$$

$\vdots$

n

1. Distance calculation $\longrightarrow O(nd)$
2. Sort distance ascending $\longrightarrow O(n \log n)$
3. pick k-values $\longrightarrow O(k)$
4. voting $\longrightarrow$ very less

$\left.\begin{matrix} O(k) \\ \text{very less} \end{matrix}\right\}$ very low

Time complexity $= O(nd + n\log n)$

problem with KNN

1. As n increase KNN will go slow
2. As d increase KNN will go slow

Advantages

1. KNN can generate non-linear boundary
2. Simple to understand
3. very powerful on smaller data
4. very easily applied in multi-class classification.

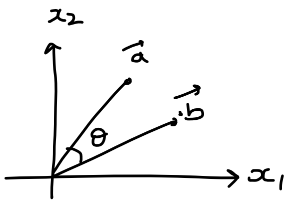
Various Distance consideration
KNN

1. minkowski Distance $d = \left[\sum_{d} |x_{ad} - x_{bd}|^p \right]^{1/p}$ → for custom distance metric

2. $P=1$ [L1] $d = \sum_{d} |x_{ad} - x_{bd}|$

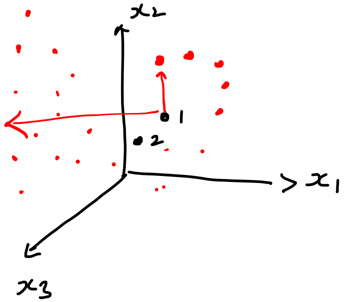
3. $P=2$ [L2] $d = \left(\sum_{d} |x_{ad} - x_{bd}|^2 \right)^{1/2}$ → Data is less

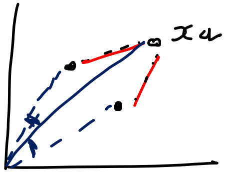
4. Cosine Similarity $\cos(\theta) = \dfrac{\vec{a}^T \cdot \vec{b}}{||\bar{a}|| \, ||\bar{b}||}$ → In the case of highly dimensional data



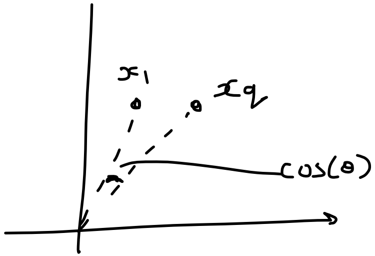
$$E.d = \left[(x_q - x_1)^2 + (x_q - x_2)^2 + \cdots\cdots + (x_q - x_{100})^2 \right]^{1/2}$$

$Ed(1) \simeq Ed(2)$ Because of higher dimensions





cosine similarity in higher dimension the angles $\cos(\theta)$ appears very different and easy for algo to differentiate.



Weighted KNN

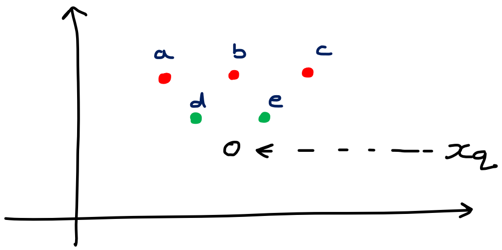
$d(x_q, a) = 0.5$

$d(x_q, b) = 0.5$

$d(x_q, c) = 0.5$

$d(x_q, d) = 0.2$

$d(x_q, e) = 0.2$



$K = 5$

$x_q \rightarrow$ Red

Idea → Data point close to the query data point should be given more weight in the voting

$$\text{weight} \propto \frac{1}{\text{distance}}$$

$$w(i) = \frac{1}{d(i)}$$

$w_1 = \frac{1}{0.5} = 2$

$w_2 = \frac{1}{0.5} = 2$] Red

$w_3 = \frac{1}{0.5} = 2$

$w_4 = \frac{1}{0.2} = 5$] Green

$w_5 = \frac{1}{0.2} = 5$

x_q: Red = $[2+2+2] = 6$

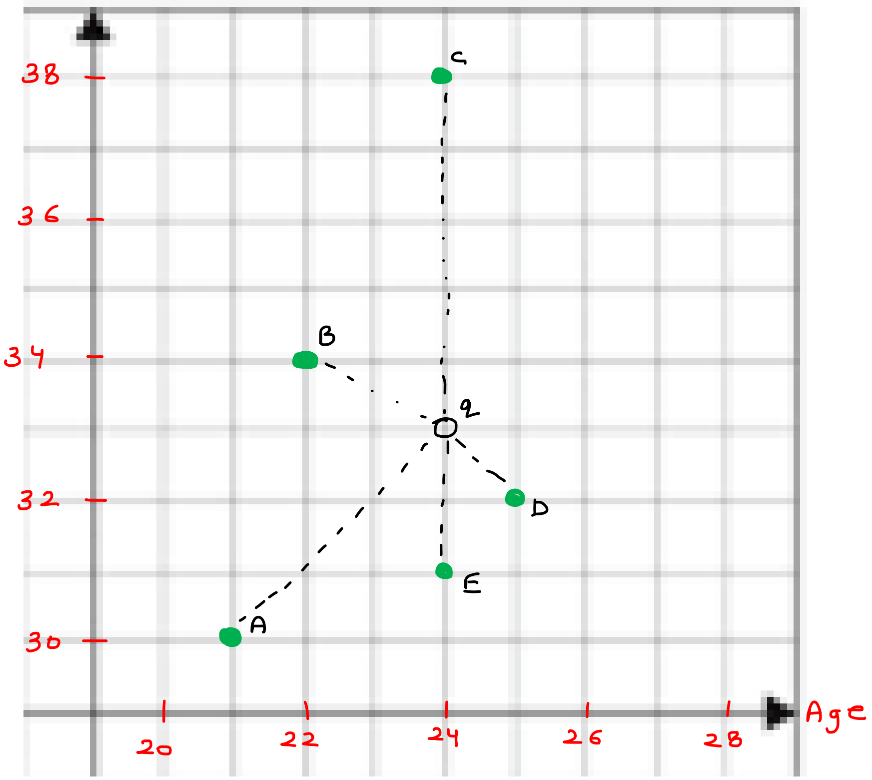
Green $[5+5] = 10$

↓

green

KNN- Imputation (for missing value imputation)

	Age	Income	Tenure	Distance
A	21	30	2	4.24
B	22	34	5	2.24
C	24	38	1	5.00
D	25	32	2	1.41
E	24	31	4	2.00
q	24	33	NA	



The missing data point can be predicted by taking avg of closest neigbours

$$\boxed{K=3}$$

$$\text{Tenure}(q) = \frac{\text{Tenure}(B) + \text{Tenure}(E) + \text{Tenure}(D)}{3}$$

$$= \frac{5+2+4}{3} = 3.$$