12-OCT-2025

Agenda:

- → ROC
- → AUC
- → Cross-validation
- → k-nearest Neighbour (ML Algo)

ROC Curve:

Classifier → Score → prob(0,1) → 1 (positive)
0 (negative)

→ To make a decision we need a threshold

Score ≥ threshold → 1 else 0

→ TPR - True positive Rate

$= \frac{TP}{TP + FN}$

spam classifier:

→ how many actual spams we caught correctly

→ FPR - False positive Rate

$= \frac{FP}{FP + TN}$

→ how many not spams were wrongly marked as spam.

ROC curve is TPR vs FPR for all threshold.
(Receiver operating characteristic)

y	$\hat{y}$	Threshold (0.2) $\geq$	Threshold (0.3) $\geq$
1	0.8	1	1
0	0.1	0	0
0	0.4	1	1
1	0.56	1	1
1	0.7	1	1
0	0.2	1	0
		↓	
		TPR & FPR	TPR & FPR

ROC Curve :—

For each threshold (0.1 → 0.9), we calculate two things:

→ TPR

→ FPR

→ we plot TPR & FPR for all threshold.

Email	True label (y)	Model-score ($\hat{y}$)
A	1	0.95
B	0	0.70
C	1	0.60
D	0	0.40
E	1	0.20

Actual spam : A, C, E

Actual negative : B, D

Email	True label (y)	Model-Score ($\hat{y}$)	Threshold (0.8)
A	1	0.95	1
B	0	0.78	0
C	1	0.68	0
D	0	0.48	0
E	1	0.28	0

Threshold : 0.8

TP: A (1) FP: 0

TN: B, D (2) FN: C, E (2)

TP :	model predict positive , actual positive
TN :	model predict negative , actual negative
FP :	model predict positive , actual negative
FN :	model predict negative , actual positive

$$TPR = TP / (TP + FN)$$
$$= 1/3 = 0.33$$

$$FPR = FP / (FP + TN)$$
$$= 0/(0+2) = 0$$

Threshold = 0.6

TPR = 0.67

FPR = 0.5

Threshold = 0.4

TPR = 0.67

FPR = 1.0

Threshold = 0.0

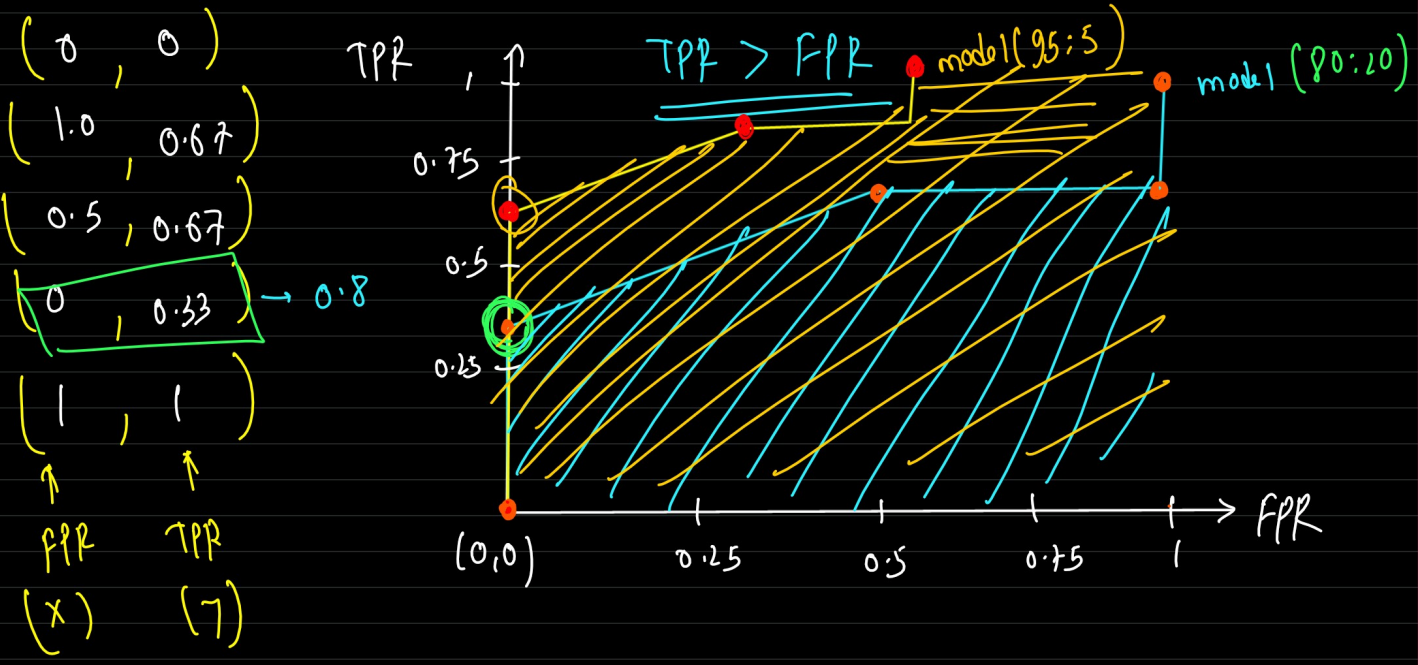
TPR = 1

FPR = 1

Threshold = 1.0

TPR = 0

FPR = 0



AOC – Area under ROC Curve

– numerical value of the area under ROC curve

: prob that the model ranks a random positive higher than random negative.

AUC → 1 : model is better at separating classes

= 0.7

= 0.5

x_1 x_2 x_3 x_4 carrier

1

0

1

0

5-10% → 1
90-95 → 0
} flipped

model → 0.9

0.8
0.5
0.4
0.3
0.45 → 2

Cross-validation :—

data → split → train → fit → evaluate the model
                test ↗

* Maybe our test set is too hard.
* Maybe our test set is too easy.

→ reduce this dependency of data split.

→ solution:

Instead of one fix split, we train multiple times, each time with a different split.

Index	Data
1	A
2	B
3	C
4	D
5	E
6	F

: 3 fold cross validation:

: Each data becomes test data once

fold	Train	Test
1	[C, D, E, F]	[A, B] → ML.fit → Metric
2	[A, B, E, F]	[C, D] → ML.fit → Metric
3	[A, B, C, D]	[E, F] → ML.fit → Metrics

1 → 96%.
2 → 84%.
3 → 90%.

} average → bias variance } not for model improvement.

K-fold → split data into k parts, each used once for testing

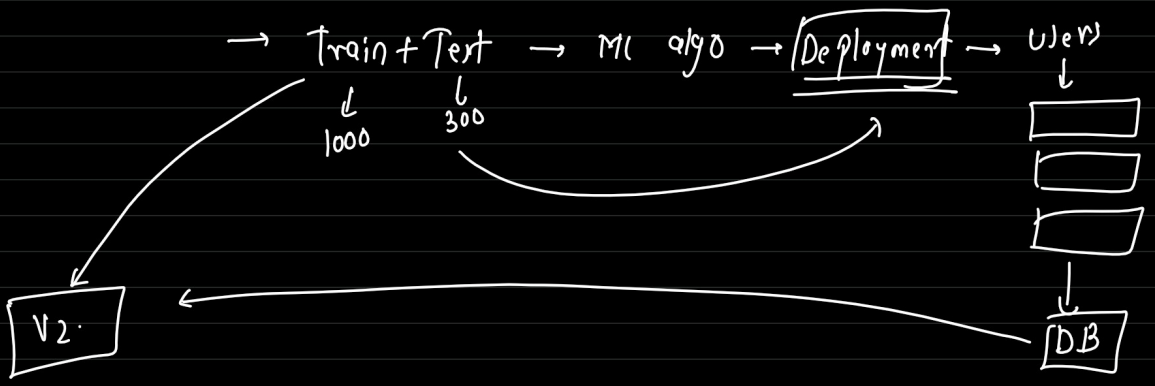
stratified k-fold → ensure class balance in each fold

Data → 1000 rows

EDA → FE

→ cross-validation split → general → Train

accuracy → Test → 1 → 2 → 3



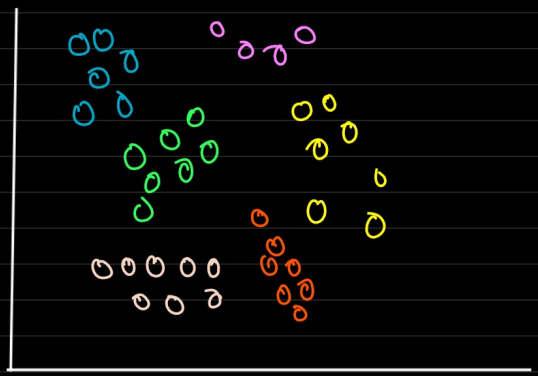
---

kNN → k - Nearest Neighbour.

→ classification

→ Regression

* mathematical equation (line or curve) ✗
* what if data doesn't form a nice line



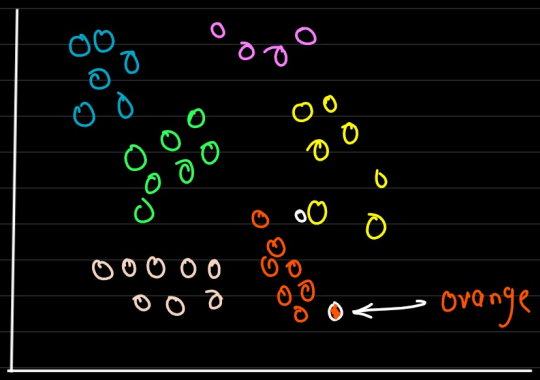
Solution: KNN

→ Instead on relying on equation, why not use the data itself to make decision.

→ we don't compute any parameters. $\boxed{m_1, m_2, \beta, M}$

→ called as Lazy learner, bcoz it doesn't learn any equation.

→ supervised algo

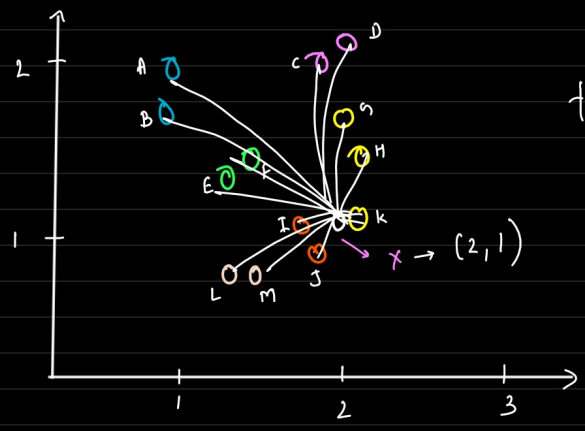


k=1 => 1 neighbour

k=3 =>

O → yellow
→ 2 orange neighbour
↓
majority
↓
orange

k => odd → majority



formula: $\sqrt{\sum (x_i - x_j)^2}$

A(1,2) ↓ x_2 x(2,1) ↓ x_1

point	coordinates	formula	Distance from (2,1)
A	(1,2)	$\sqrt{(2-1)^2+(1-2)^2}$	1.41
B	(0.9, 1.7)		1
C	(x, y)	$\sqrt{\quad}$	2
D		$\sqrt{\quad}$	4

sort the distance in ascending order

→ Data point closer to $x(2,1)$

$k = 5 \rightarrow$ 3 → orange → majority → $x(2,1)$ → orange

→ 2 → Yellow

Euclidean distance :

$$d(x_i, x_j) = \sqrt{\sum (x_i - x_j)^2}$$

Manhattan distance = $\sum |x_i - x_j|$

Optimal value of k:

Small ($k = 1, 2$) → High variance

Medium ($k = 5, 7, 9$) → Balanced

Large ($k \rightarrow N$) → High bias