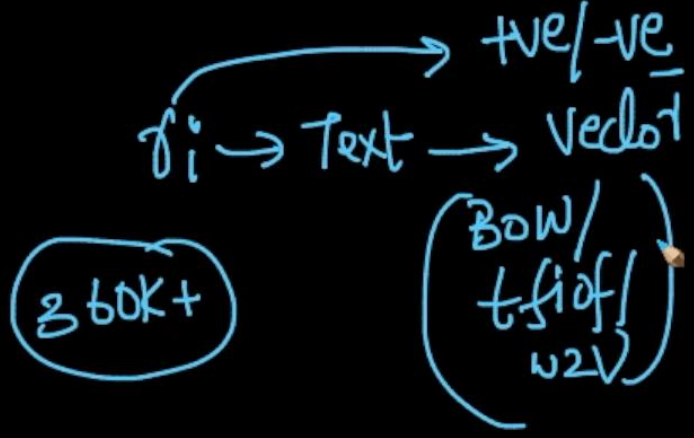
Classification & Regression : K-NN

How "classification" works



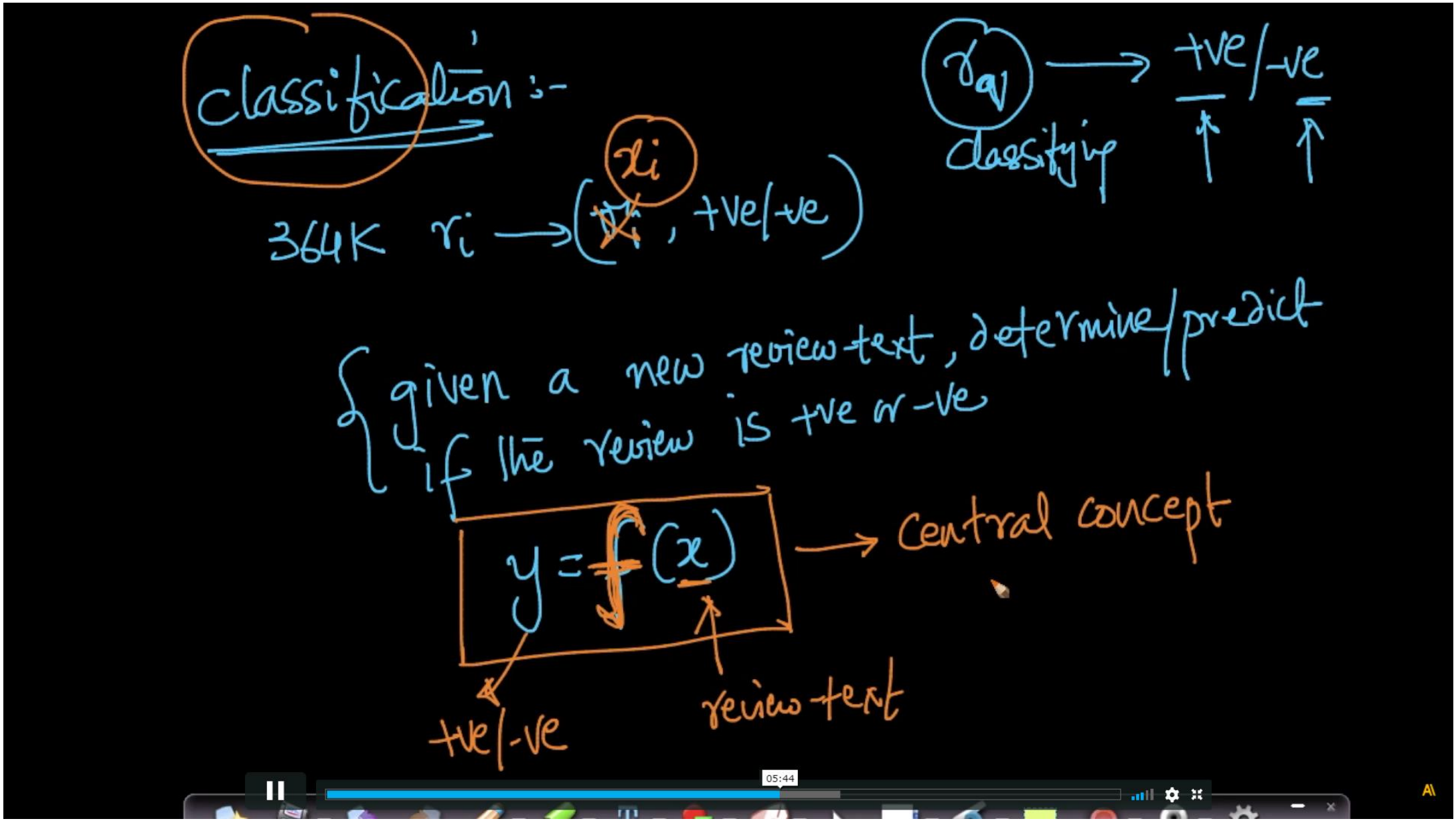
{Amazon Food reviews}

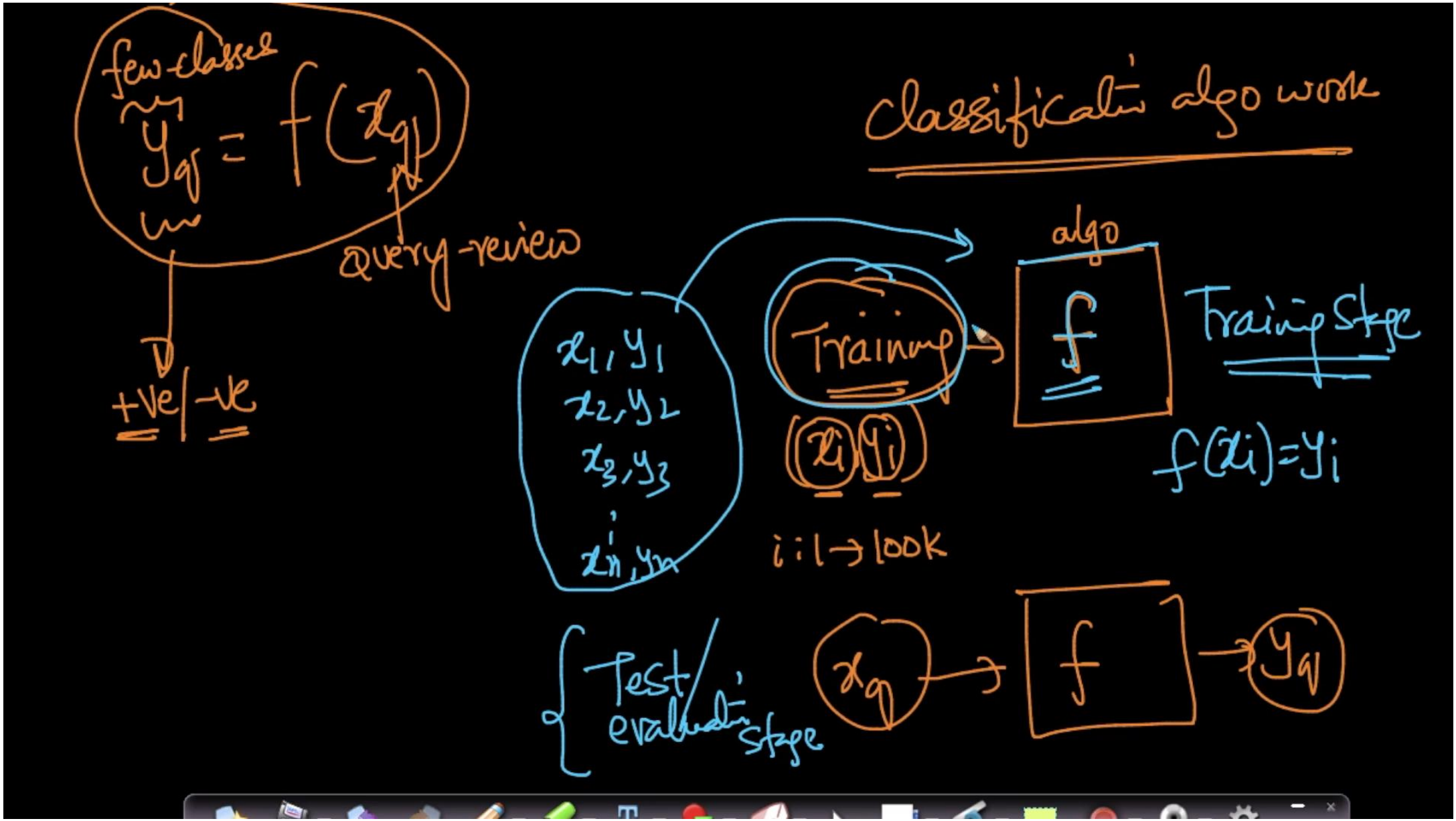
↓

real-world eg.

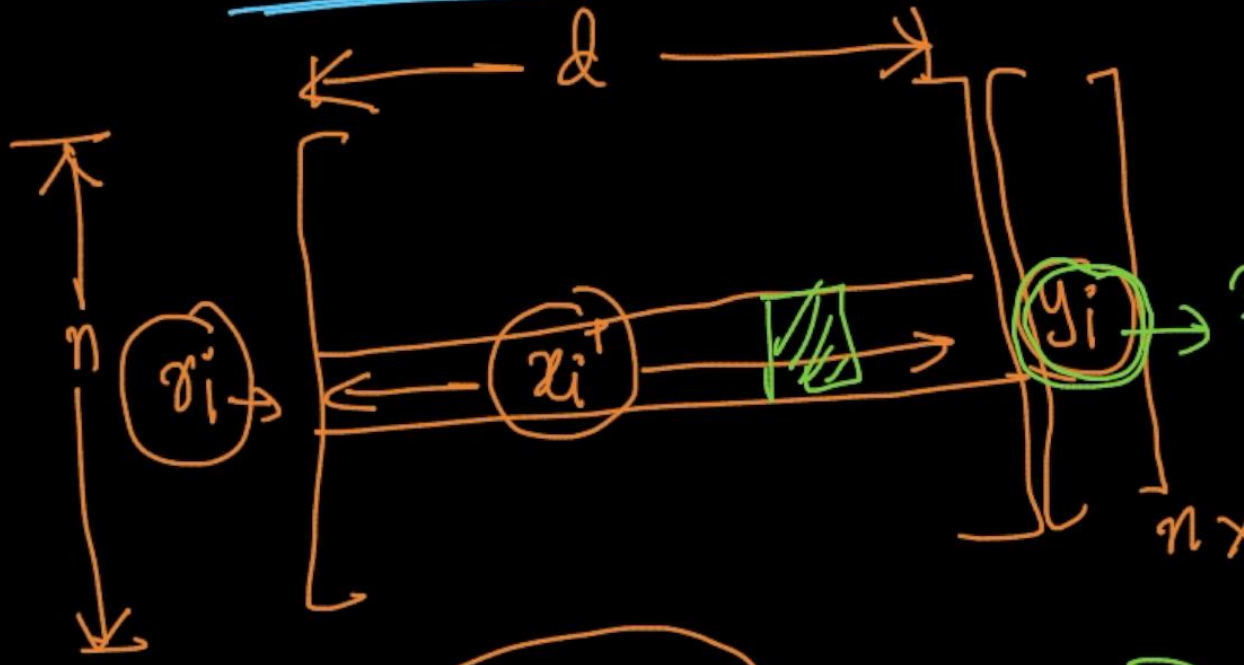
{MNIST (PCA/t-SNE)

{0,1,2, . 9}


classification :-
364K r_i → (x_i, +ve/-ve)
r_q → +ve/-ve
classifying
given a new review text, determine/predict if the review is +ve or -ve
$y = f(x)$
central concept
+ve/-ve
review text
05:44


few classes
$y_q = f(x_q)$
query review
+ve / -ve
classification algo work
algo
f
Training Stage
$f(x_i) = y_i$
Training
x_1, y_1
x_2, y_2
x_3, y_3
x_n, y_n
x_i y_i
i : 1 → look
Test / evaluation stage
x_q
f
y_q

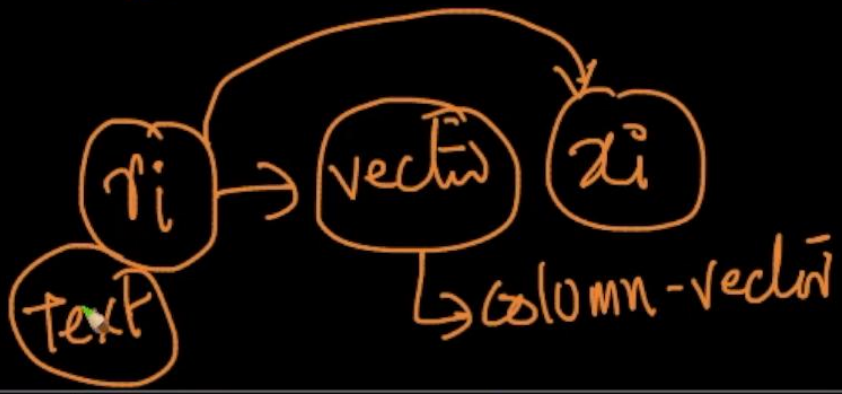
Notation:



$r_i \rightarrow$ Text $\rightarrow$ Vector $\bar{x}$

$\rightarrow$ class (+ve/-ve)

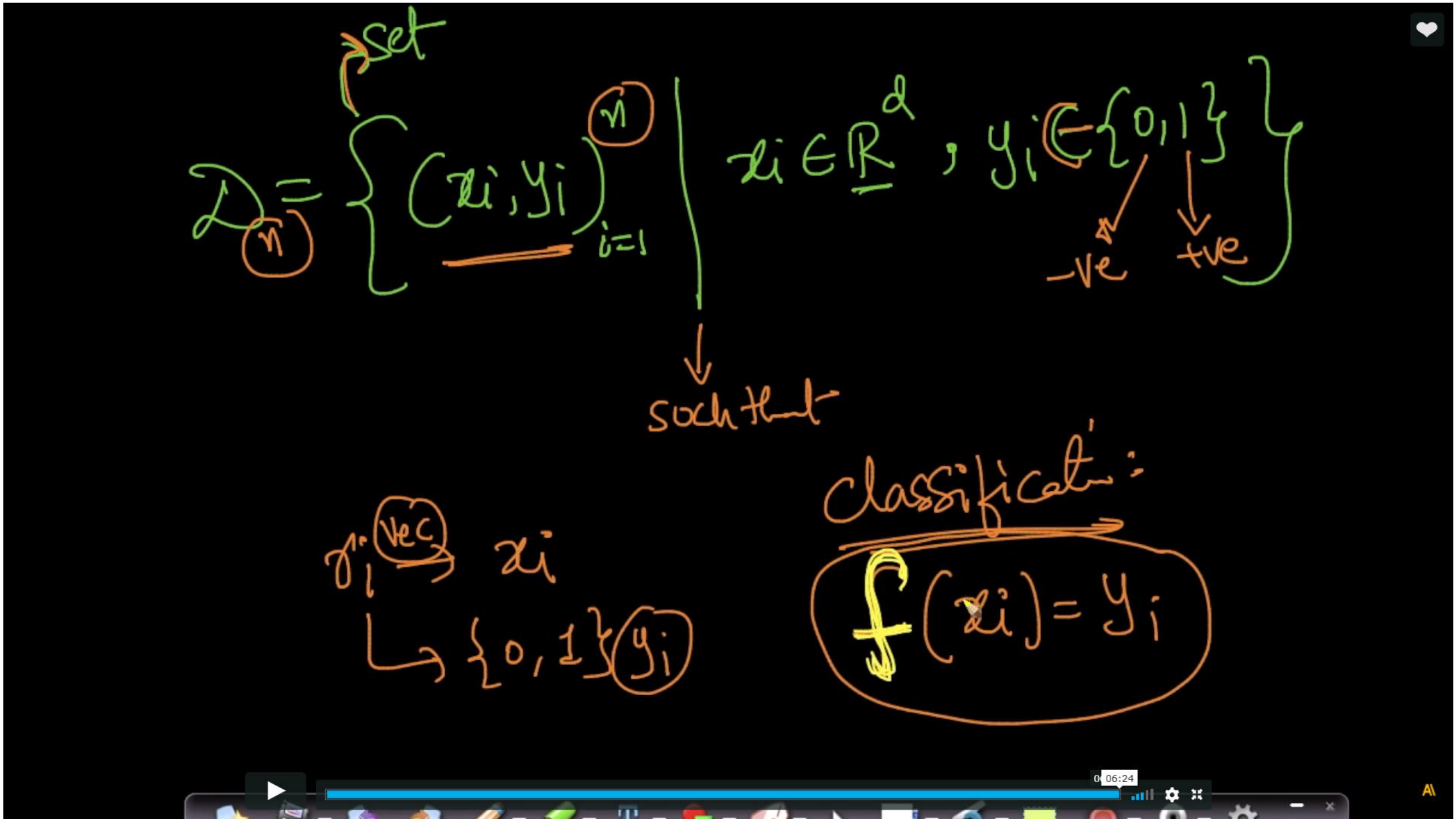
$y = f(x)$



$$\mathcal{D} = \left\{ (x_i, y_i)_{i=1}^{n} \;\middle|\; x_i \in \mathbb{R}^d, \; y_i \in \{0, 1\} \right\}$$

such that

$y_i \in \{0, 1\}$: 0 $\rightarrow$ -ve, 1 $\rightarrow$ +ve


set
$D_n = \{ (x_i, y_i)_{i=1}^{n} \mid x_i \in \mathbb{R}^d , y_i \in \{0,1\} \}$
-ve
+ve
such that
y_i (vec) x_i
$\hookrightarrow \{0,1\}$ (y_i)
classification:
$f(x_i) = y_i$
06:24

Classification vs Regression:

$$\mathcal{D} = \{(x_i, y_i)_{i=1}^{n} \mid x_i \in \mathbb{R}^d, y_i \in \{0, 1\}\}$$

$y_i \in \{0, 1\}$ ← Amazon Food reviews

y_i: 0 → -ve, 1 → +ve

→ 2 classes

→ 2 class - classification / Binary

MNIST: $y_i \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ → 10-class / Multi-class classification

K- Nearest Neighbors: (Geometric)

$x_q \rightarrow y_q$ (binary classificat

2D - toy dataset:

$\mathcal{D}$ {

x: +ve datapoint
x: -ve class datapoint

$\mathcal{D} = \{(x_i, y_i) \mid x_i \in \mathbb{R}^2, y_i \in \{0, 1\}\}$

-ve +ve

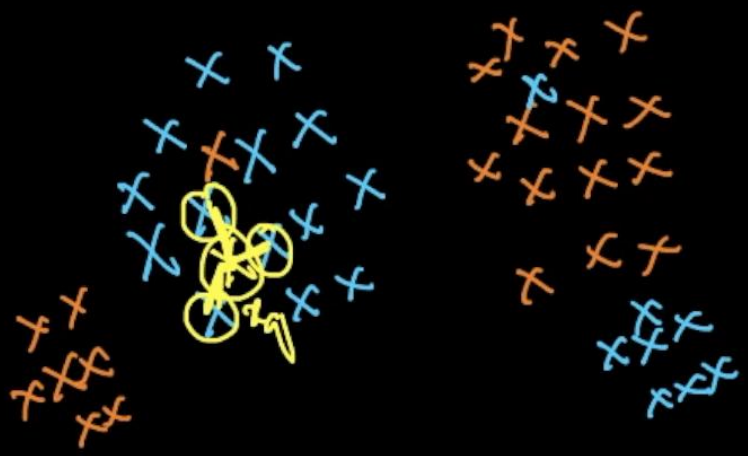
geom: close to x_q:

Conclude x_q → blue (+ve)

$\mathcal{D} \rightarrow$ [ML] $\rightarrow f$

$x_q \rightarrow [f] \rightarrow$ +ve / -ve

$D = \{(x_i, y_i)\}$



given x_q → find $y_q \in \{0, 1\}$

-ve +ve

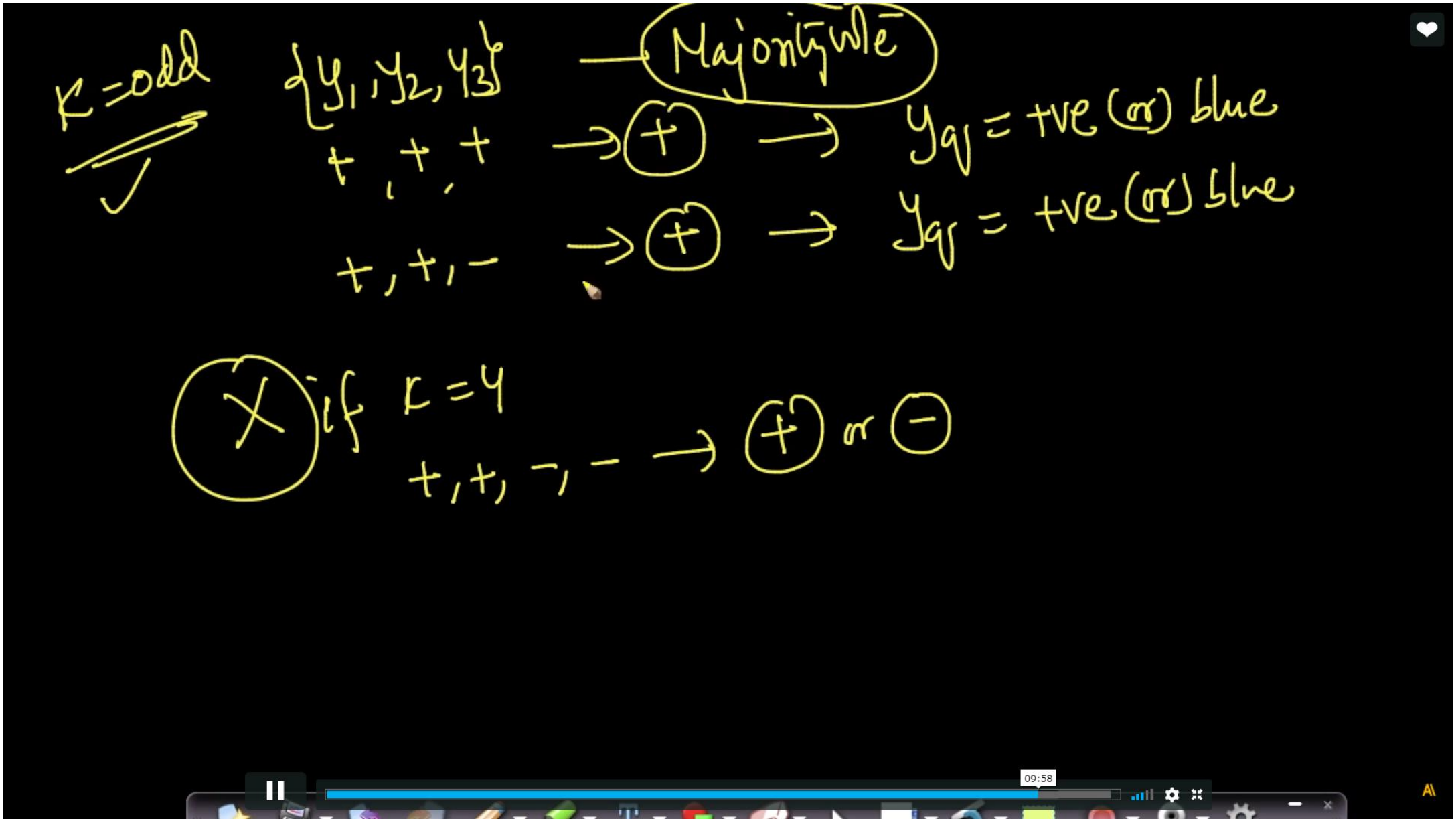
1. Find (K) "nearest" points to x_q in D

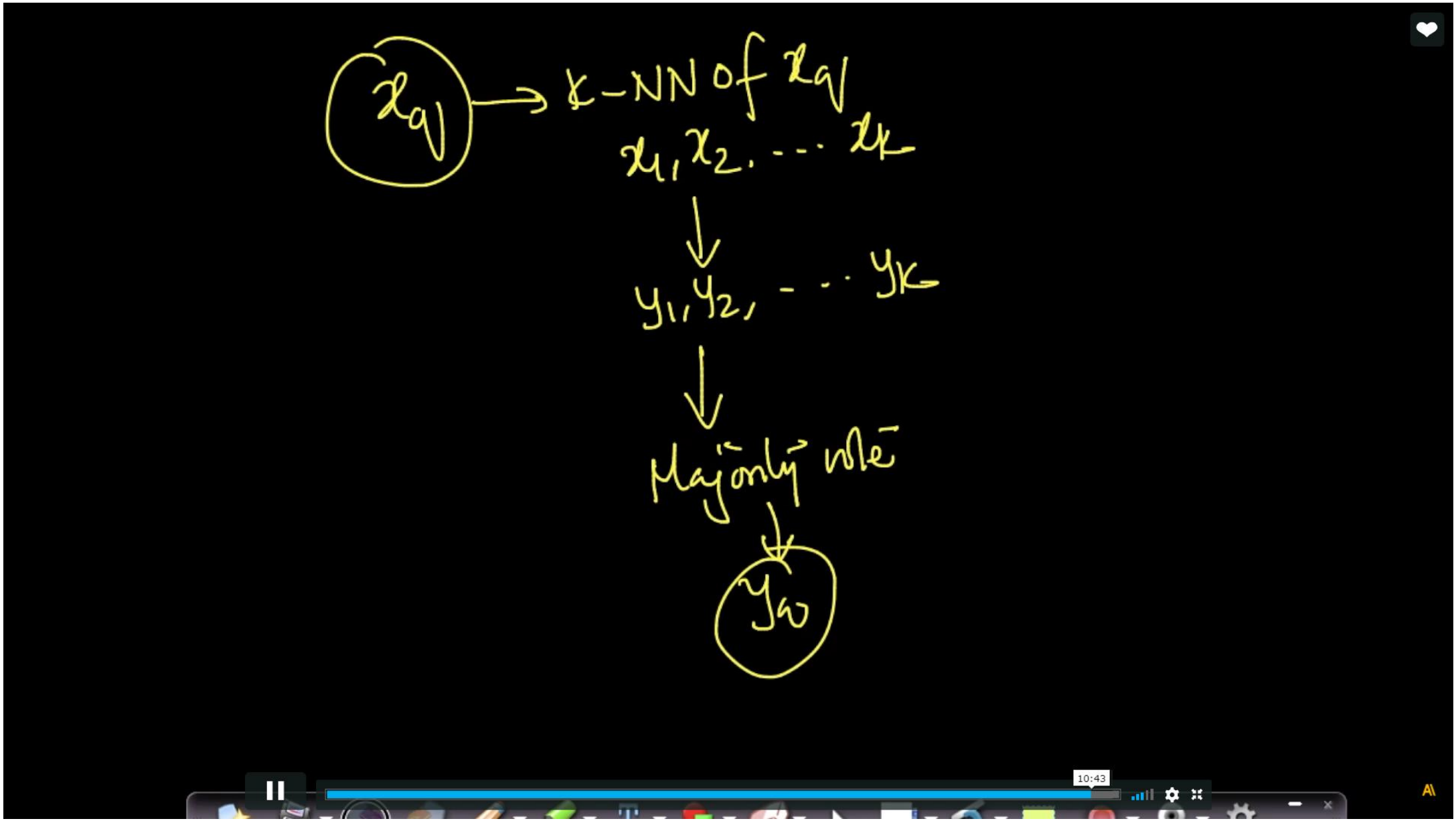
Let $K = 3$

$\{x_1, x_2, x_3\}$ → 3 NN to x_q

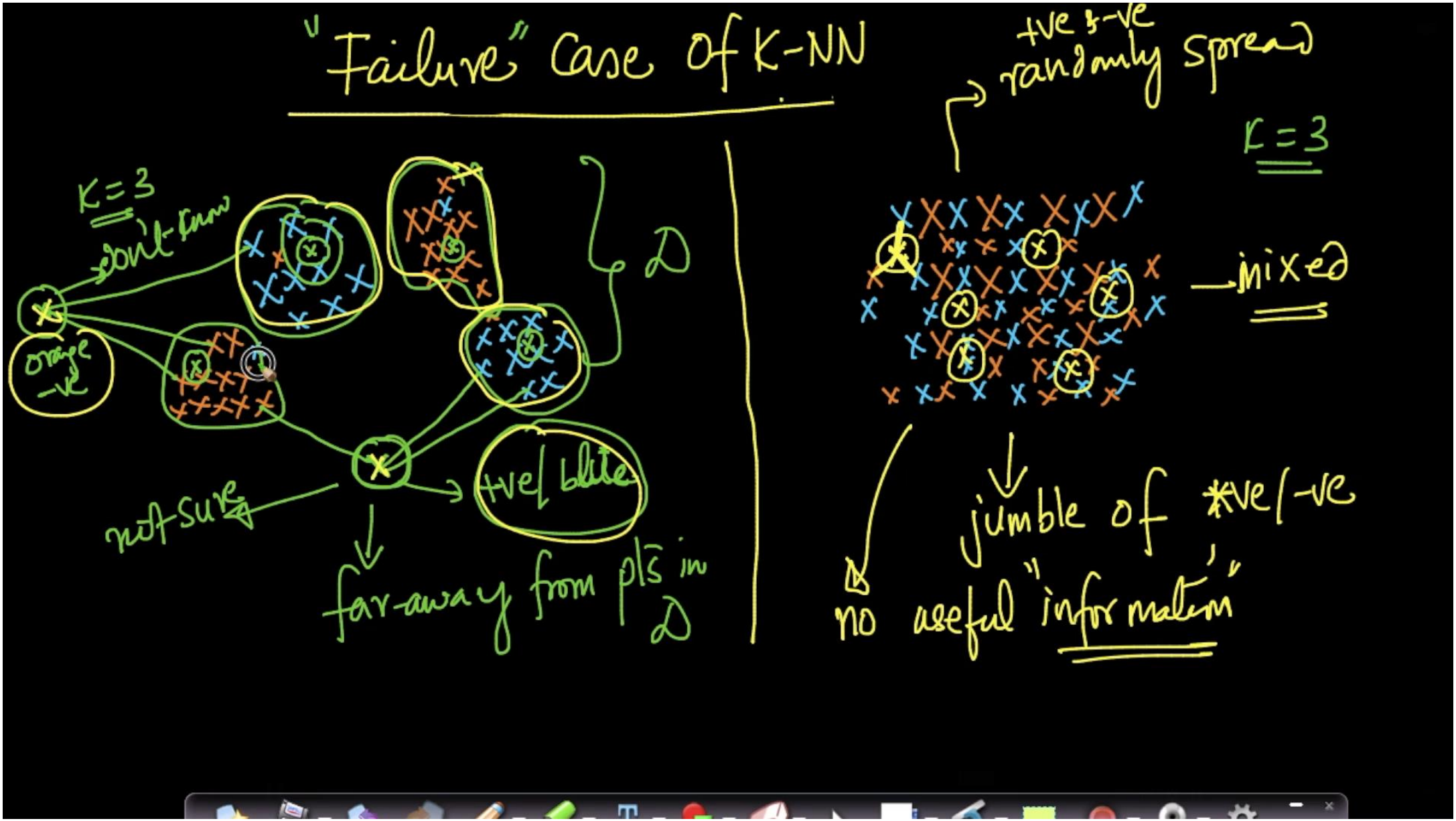
y_1 y_2 y_3

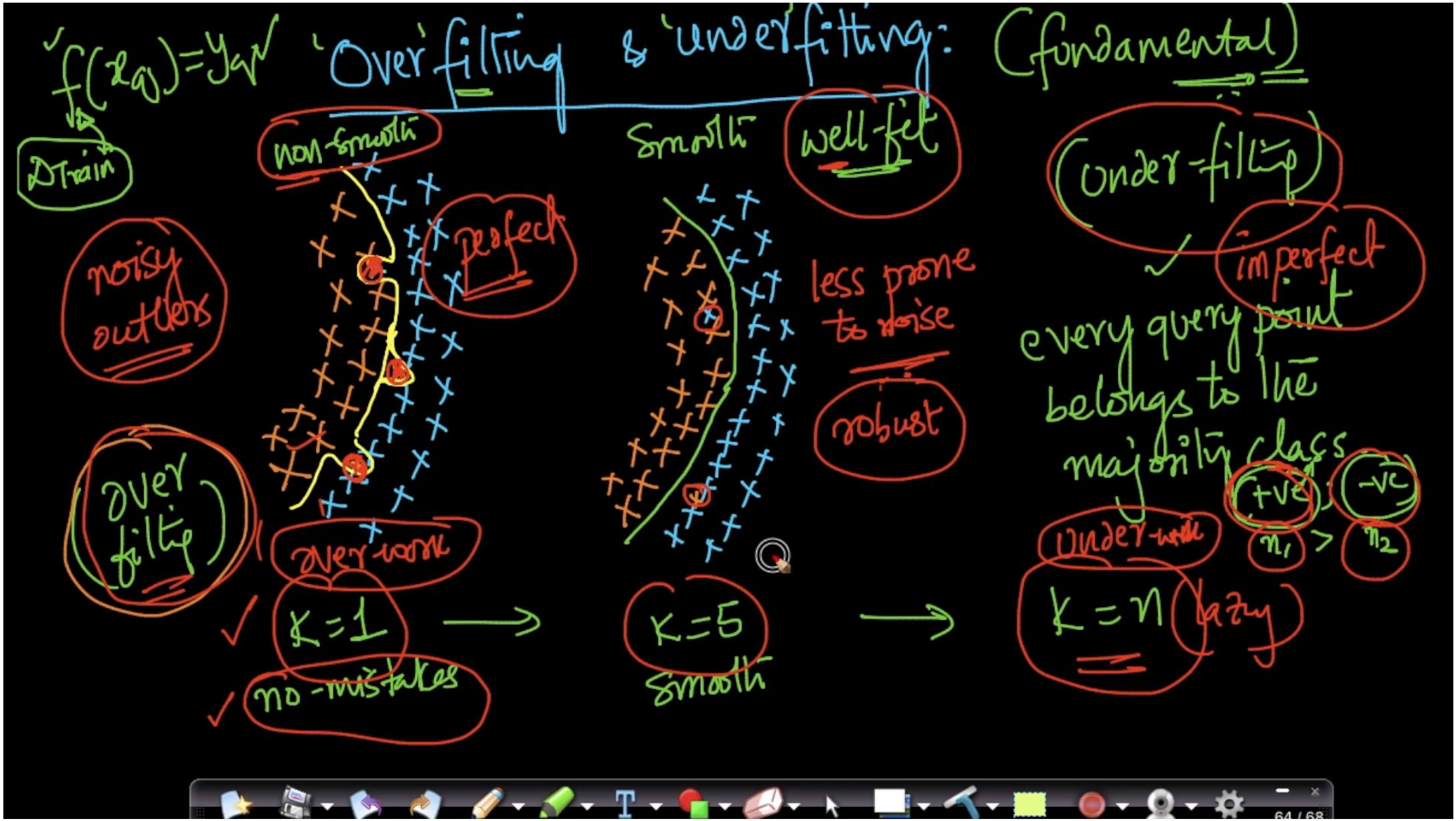
2. $\{y_1, y_2, y_3\}$ → Majority vote


K = odd
{y1, y2, y3}
Majority wle
+, +, +
+
yq = +ve (or) blue
+, +, −
+
yq = +ve (or) blue
X if K = 4
+, +, −, −
+ or −
09:58

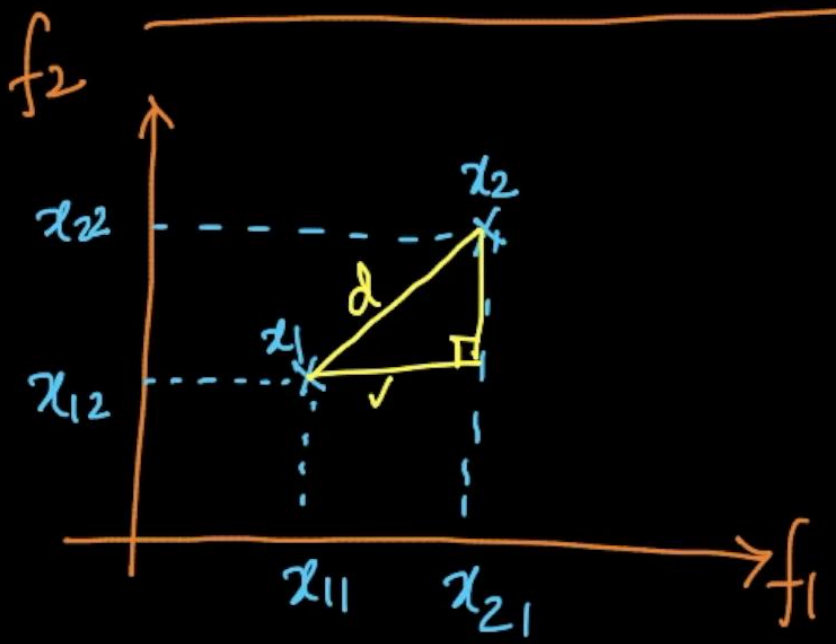

x_q
k-NN of x_q
$x_1, x_2, \ldots x_k$
$y_1, y_2, \ldots y_k$
Majority vote
y_q
10:43

"Failure" Case of K-NN




f(xq) = yq
DTrain
'Over fitting & 'under fitting: (fundamental)
non-smooth
perfect
noisy outliers
over fitting
over work
K=1
no-mistakes
smooth
well-fit
less prone to noise
robust
K=5
smooth
under-fit
imperfect
every query point belongs to the majority class
+ve
-ve
n1 > n2
under-work
K=n
lazy

"Distance" Measures:



$$x_1 = (x_{11}, x_{12}) \quad \begin{matrix} f_1 & f_2 \end{matrix}$$

$$x_2 = (x_{21}, x_{22})$$

d = len of shortest line from x_1 to x_2

Euclidean dist ← $d = \sqrt{(x_{21}-x_{11})^2 + (x_{22}-x_{12})^2} = \| x_1 - x_2 \|$ (length)

$x_1 \in R^d, x_2 \in R^d$

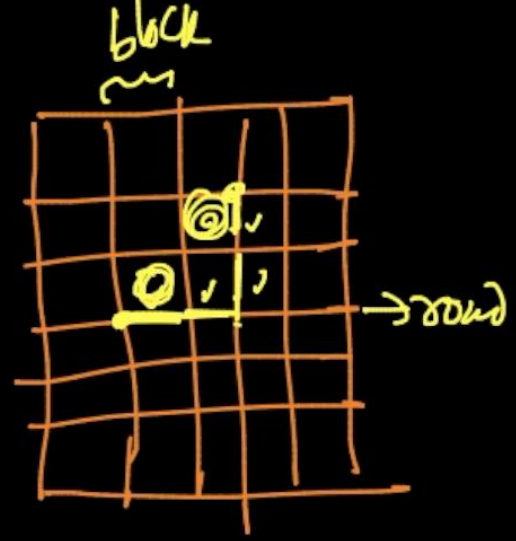
Eucl-dist: $\|\vec{x_1} - \vec{x_2}\|_2 = \left(\sum_{i=1}^{d} (x_{1i} - x_{2i})^2 \right)^{1/2}$

$\|(\vec{x_1} - x_2)\|_2 \rightarrow$ L2 norm

$\|x_1\|_2 =$ euc. dist of x_1 from origin $= \left(\sum_{i=1}^{d} x_{1i}^2 \right)^{1/2}$

"Manhattan" dist:

$$\sum_{i=1}^{d} \overbrace{|x_{1i} - x_{2i}|}^{\text{abs}}$$



L_1-norm of vector $(x_1 - x_2)$

$$\|(x_1 - x_2)\|_1$$

$$\|x_1\|_1 = \sum_{i=1}^{d} \overbrace{|x_{1i}|}^{\text{absolute}}$$

Lp-norms → Minkowski dist

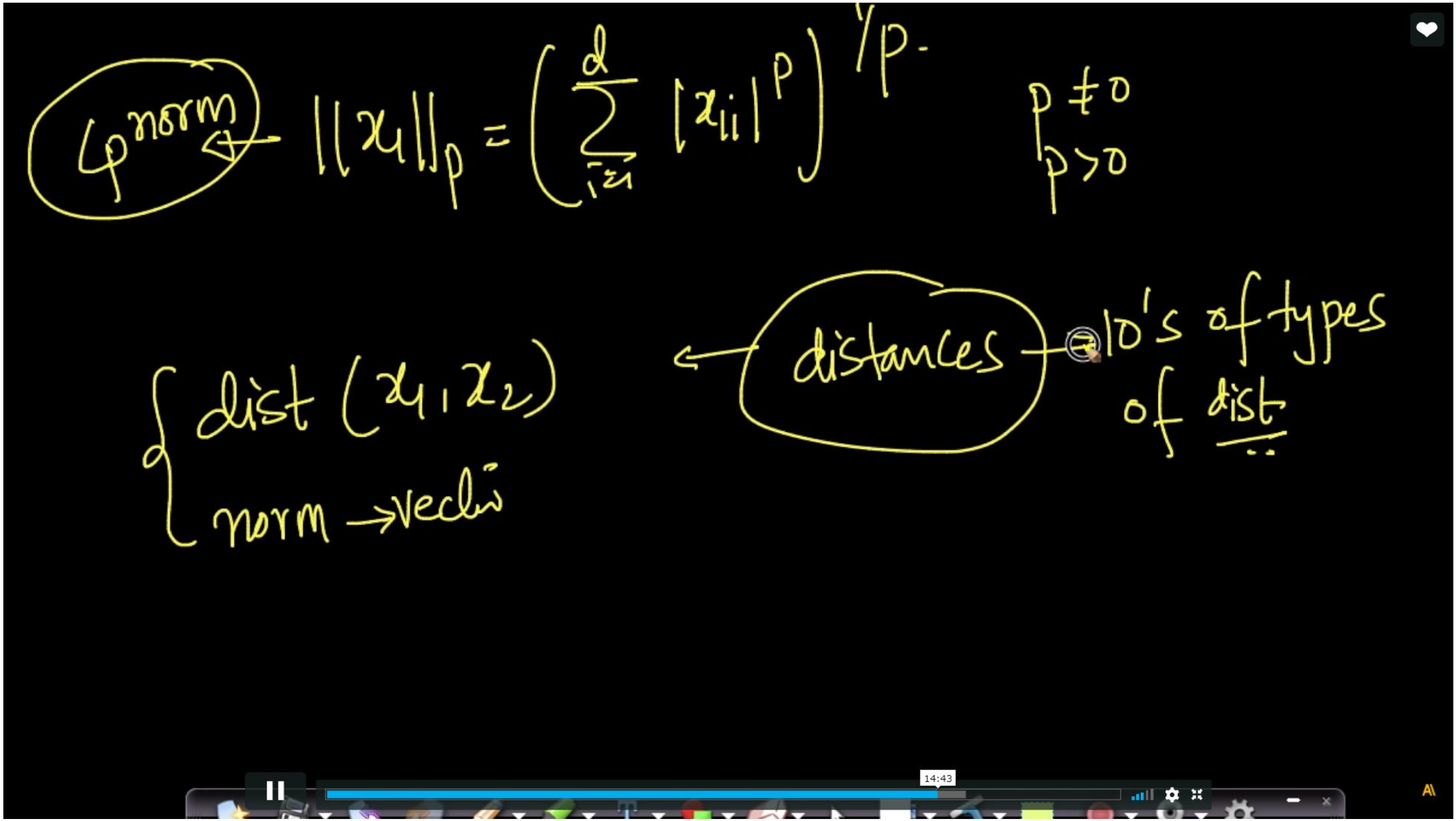
$$\|x_1 - x_2\|_p = \left(\sum_{i=1}^{d} |x_{1i} - x_{2i}|^p \right)^{1/p}$$

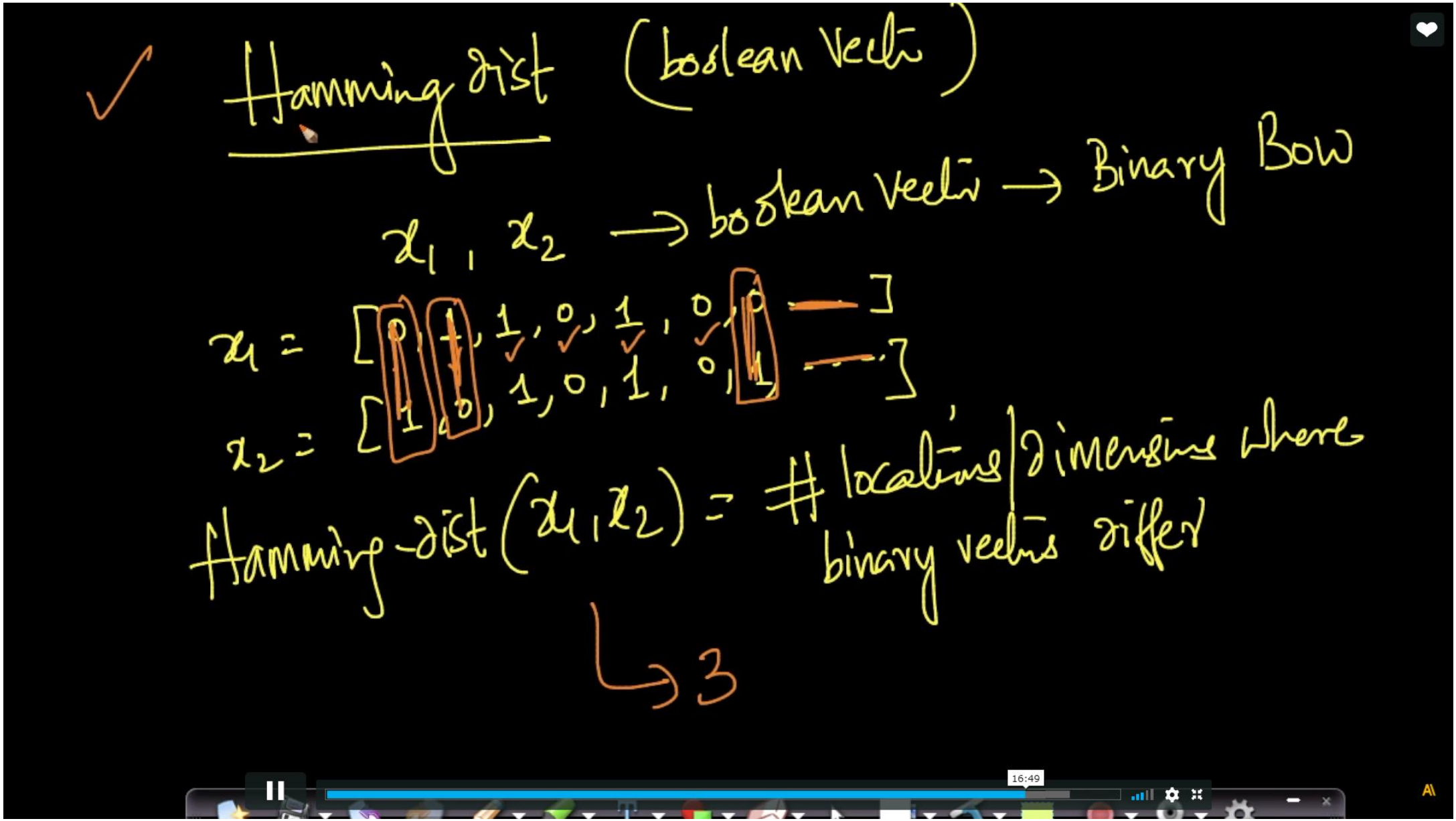
$\{ (-3)^2 = 9 \quad (|-3|)^2 = 9$

= Lp-norm of $(x_1 - x_2)$

$p = 2 \rightarrow$ Minkowski dist → Eucl. dist

$p = 1 \rightarrow$ " → Manhattan dist


L_p norm
$\|x_i\|_p = \left(\sum_{i=1}^{d} |x_{ii}|^p\right)^{1/p}$
$p \neq 0$
$p > 0$
distances
10's of types of dist...
dist (x_1, x_2)
norm → vector
14:43


Hamming dist (boolean vectors)
x1, x2 → boolean vectors → Binary BoW
x1 = [0, 1, 1, 0, 1, 0, 0 ...]
x2 = [1, 0, 1, 0, 1, 0, 1 ...]
Hamming-dist(x1, x2) = # locations/dimensions where binary vectors differ
↳ 3
16:49

Cosine-Similarity & Cosine-distance:

x_1, x_2

Similarity ↑ inc ↓ dec

distance ↑ inc ↓ dec (opposite)

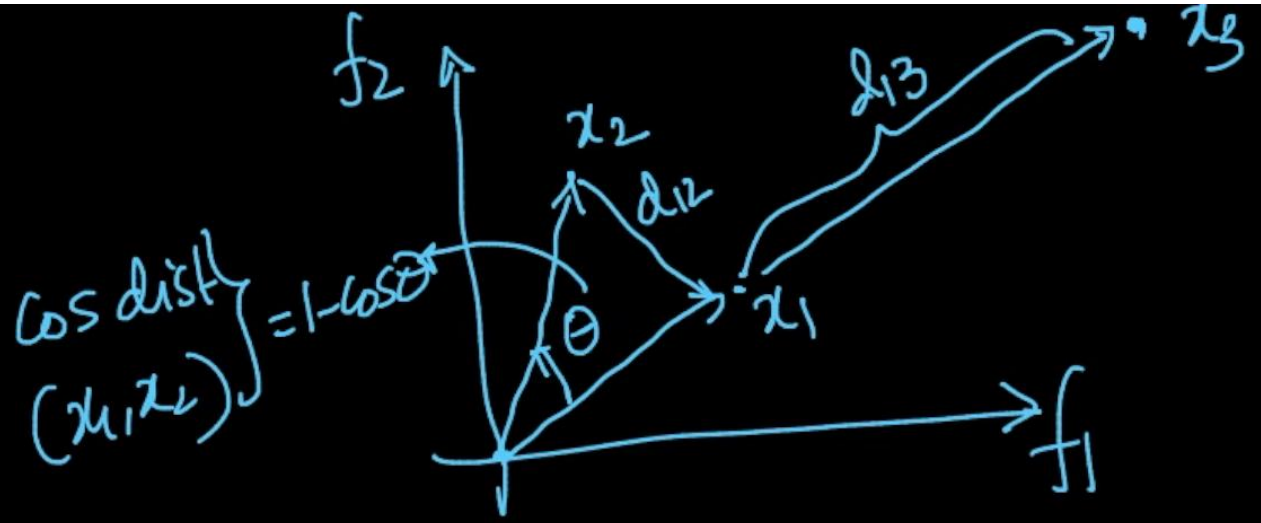
Let

V. similar $\cos\text{-sim}(x_1, x_2) = +1$

V. dissimilar $\cos\text{-sim}(x_1, x_2) = -1$

$$1 - \cos\text{-sim}(x_1, x_2) = \cos\text{-dist}(x_1, x_2)$$

Let $[-1, 1]$



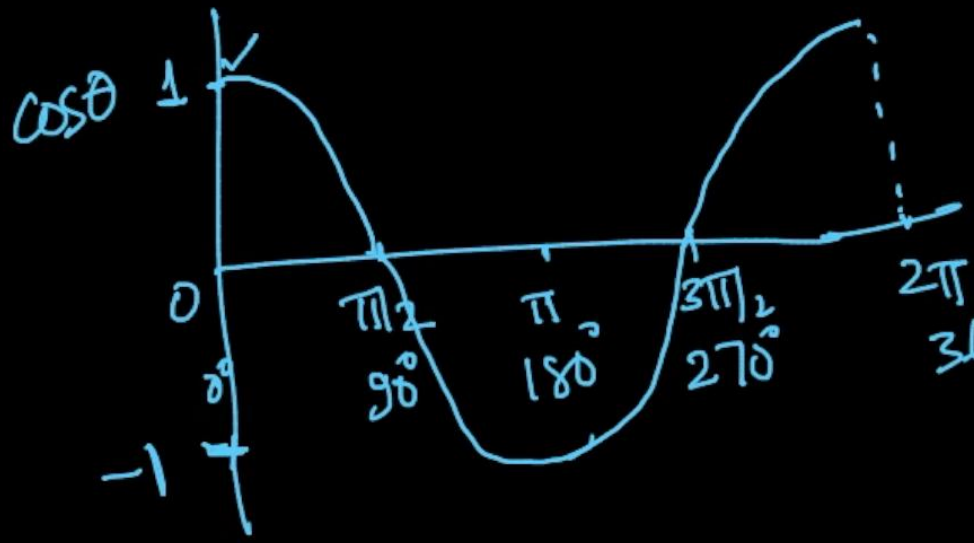
$\text{Cos dist}\ (x_1, x_2) = 1 - \cos\theta$

$\text{Cos-dist}(x_1, x_3) = 1 - 1 = 0$

$\text{Cos-sim}\ (x_1, x_2) = \cos\theta$

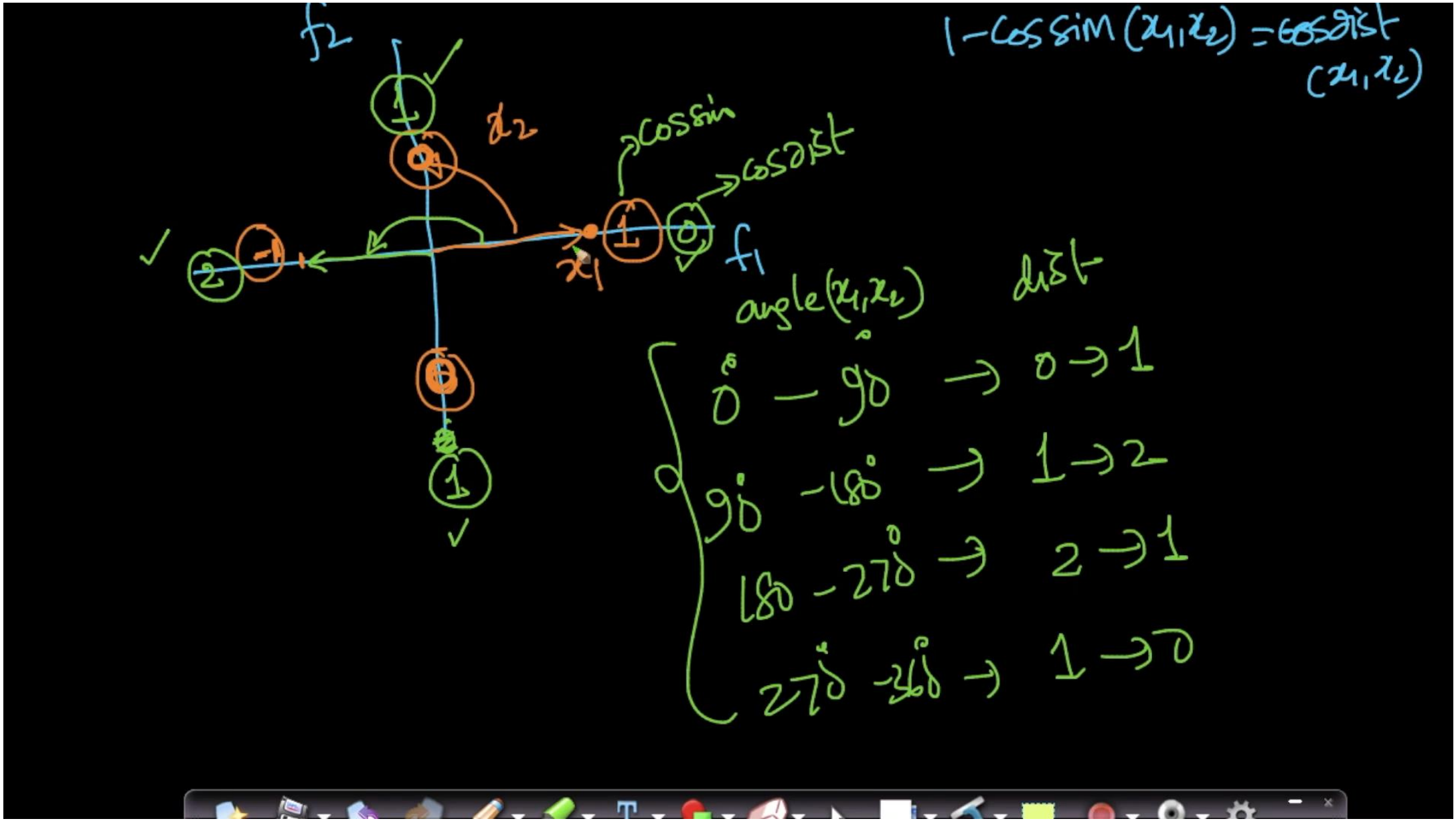
$\text{Cos-sim}\ (x_1, x_3) = 1$

$\theta_{x_1, x_3} = 0^\circ$



euc-dist

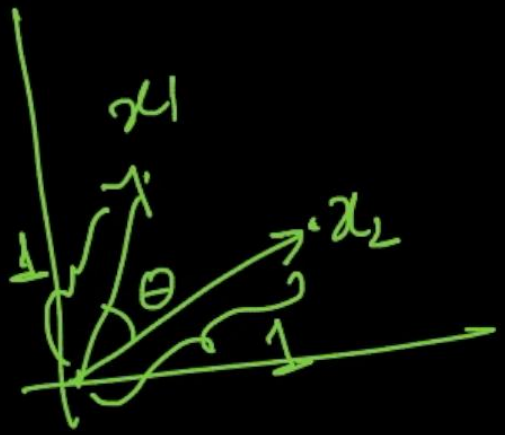
$$\begin{cases} d_{13} > d_{12} \\ \text{Cos-dist}_{13} < \text{Cos-dist}_{12} \end{cases}$$


f_2
x_2
cossim
cosdist
f_1
x_1
$1 - \text{cossim}(x_1, x_2) = \text{cosdist}(x_1, x_2)$
angle(x_1, x_2)
dist
$0° - 90° \rightarrow 0 \rightarrow 1$
$90° - 180° \rightarrow 1 \rightarrow 2$
$180 - 270° \rightarrow 2 \rightarrow 1$
$270° - 360° \rightarrow 1 \rightarrow 0$

x_1, x_2

$$\cos(\theta) = \frac{x_1 \cdot x_2}{\sqrt{\|x_1\|_2^2 \, \|x_2\|_2^2}}$$

$\|x_1\|_2^2$: L_2 norm of x_1



① If x_1 & x_2 are unit vec

$$\|x_1\|_2 = \|x_2\|_2 = 1$$

$$\boxed{\cos\theta = x_1 \cdot x_2}$$

relationship b/w euc-dist & cos-sim (cos-dist)

- if x_1 & x_2 are unit vect

θ = angle b/w x_1 & x_2

$$[\text{euc-dist}(x_1, x_2)]^2 = 2\left(1 - \underbrace{\cos(\theta)}_{\text{cos-sim}}\right)$$

$$= \underbrace{2\ \text{cos-dist}(x_1, x_2)}_{\text{cos-dist}}$$

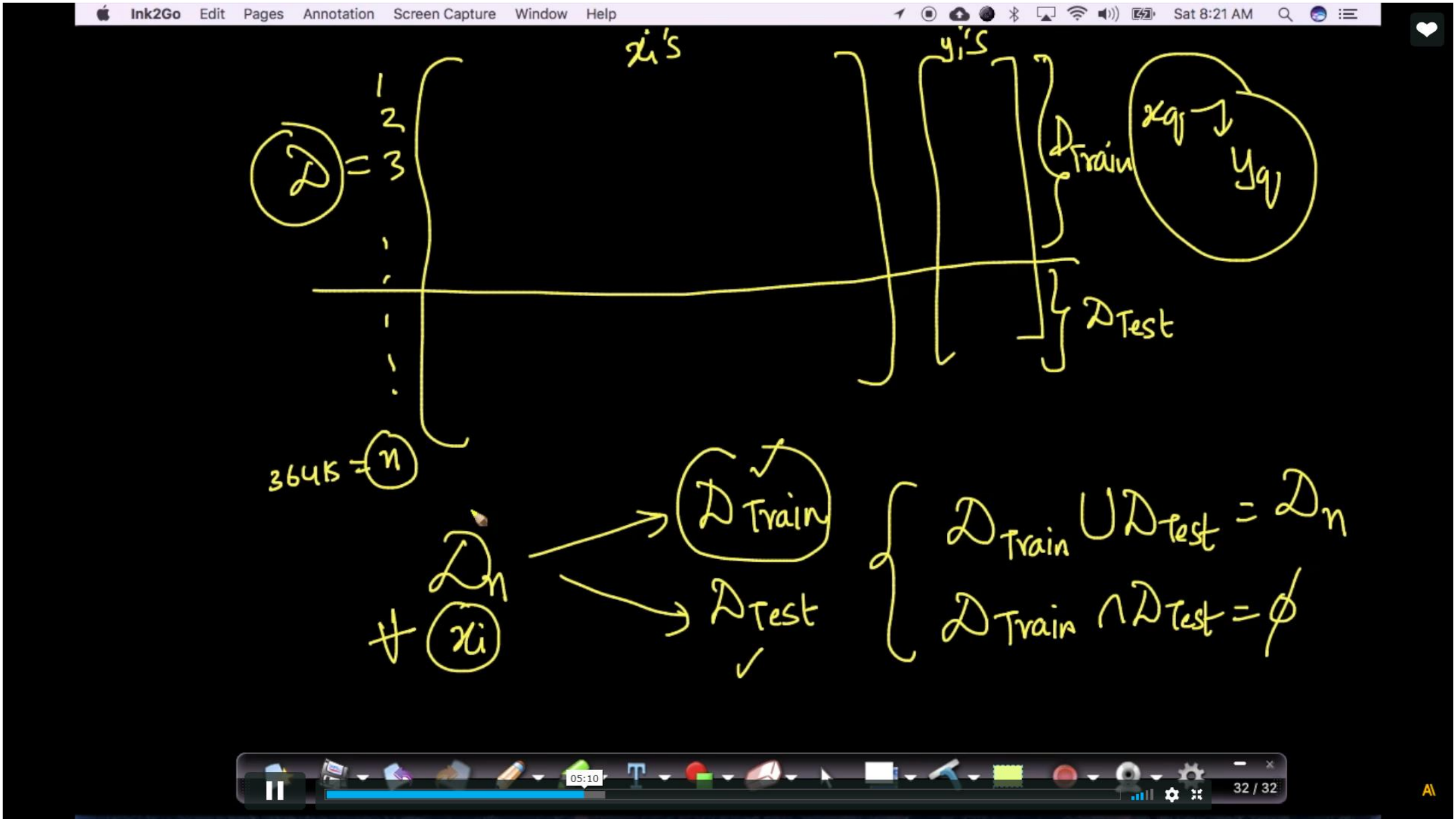
Problem: Given a new review (r_q), what is its polarity (+ve/-ve)

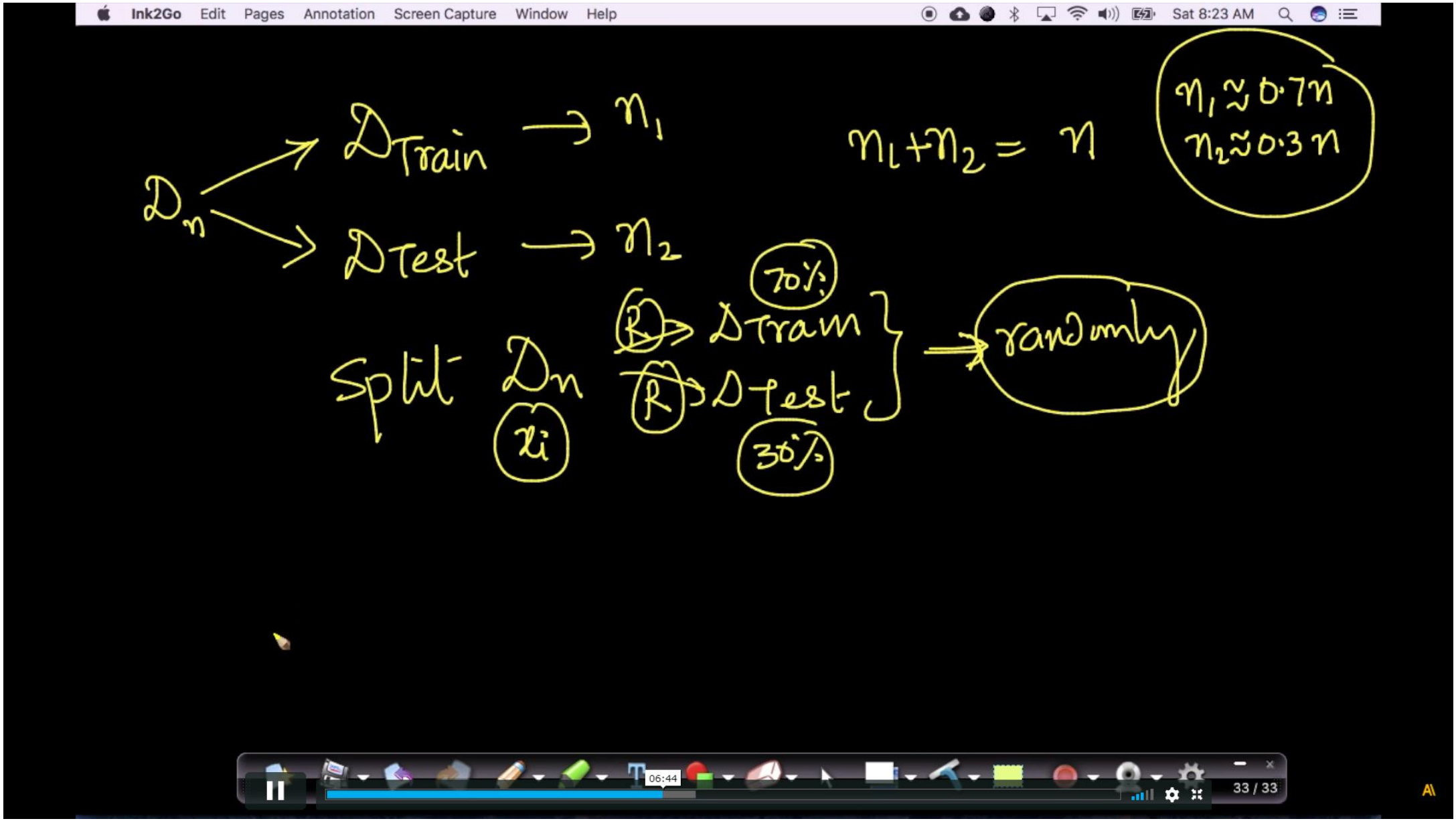
$r_q \rightarrow Text_q \xrightarrow{BOW/\ldots} x_q$

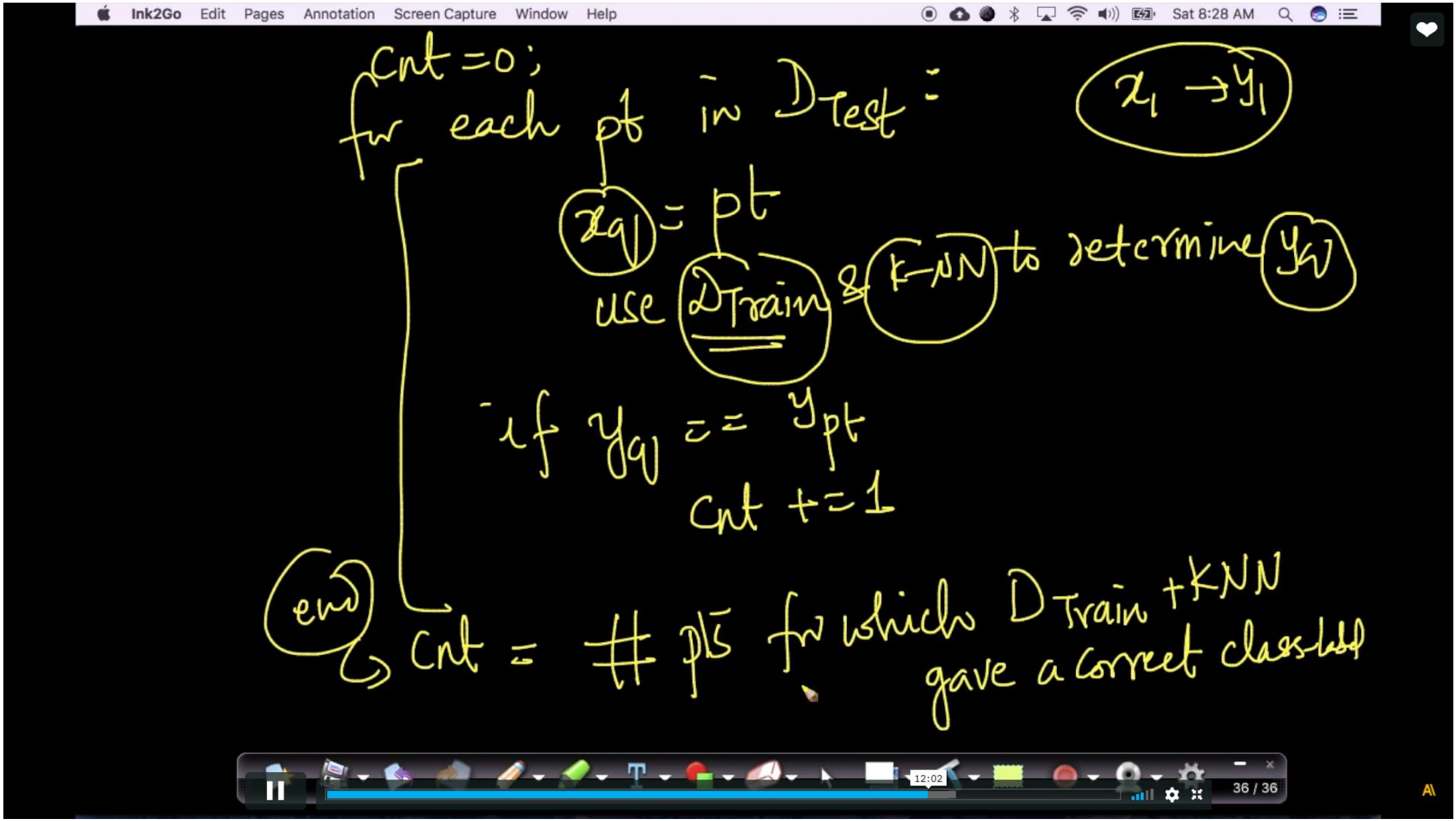
$x_q \rightarrow y_q$

"Measure"

X-NN $\rightarrow$ (K-NN + Majority vote)


Ink2Go Edit Pages Annotation Screen Capture Window Help
Sat 8:21 AM
x_i's
y_i's
$\mathcal{D}$ = 1 2 3 ⋮ 3648 = n
$\mathcal{D}_{Train}$
x_q → ? y_q
$\mathcal{D}_{Test}$
$\mathcal{D}_n$ ∀ x_i
$\mathcal{D}_{Train}$
$\mathcal{D}_{Test}$
$\mathcal{D}_{Train} \cup \mathcal{D}_{Test} = \mathcal{D}_n$
$\mathcal{D}_{Train} \cap \mathcal{D}_{Test} = \phi$
05:10
32 / 32


Ink2Go Edit Pages Annotation Screen Capture Window Help
Sat 8:23 AM
D_n
$D_{Train} \rightarrow n_1$
$D_{Test} \rightarrow n_2$
$n_1 + n_2 = n$
$n_1 \approx 0.7n$
$n_2 \approx 0.3n$
Split D_n
x_i
R
D_{Train}
D_{Test}
70%
30%
randomly
06:44
33 / 33


Ink2Go Edit Pages Annotation Screen Capture Window Help
Sat 8:28 AM
cnt = 0;
for each pt in D_test :
x_1 → y_1
x_q = pt
use D_Train & K-NN to determine y_q
if y_q == y_pt
cnt += 1
end
cnt = # pts for which D_Train + KNN gave a correct classification
12:02
36 / 36

$$\text{Accuracy} = \frac{\text{Cnt}}{n_2}$$

Cnt $\longrightarrow$ # pts for which D_{train} + KNN gave a correct class label

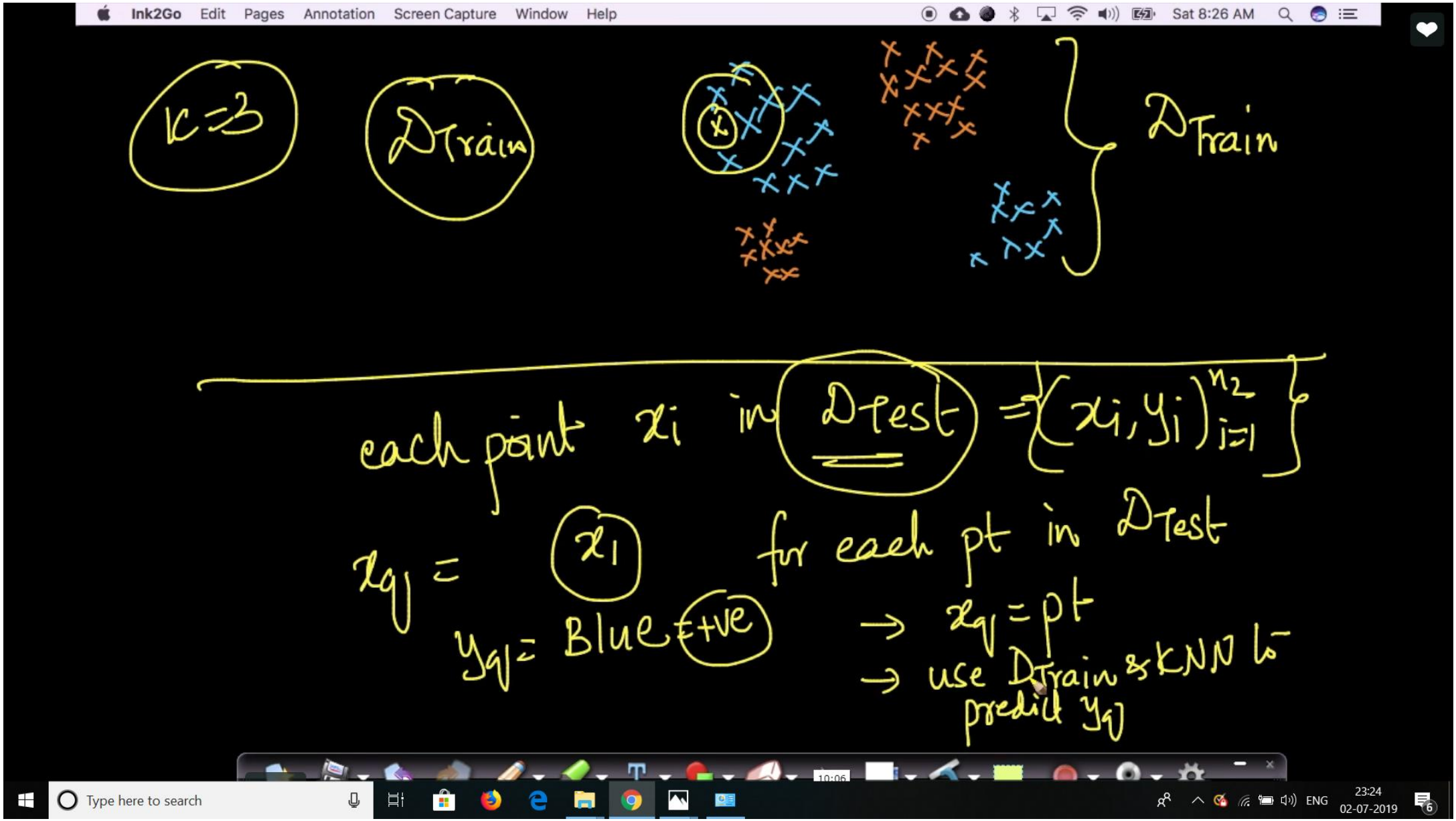
n_2 $\longrightarrow$ # pts in D_{test}

$$0 \leq Acc \leq 1$$

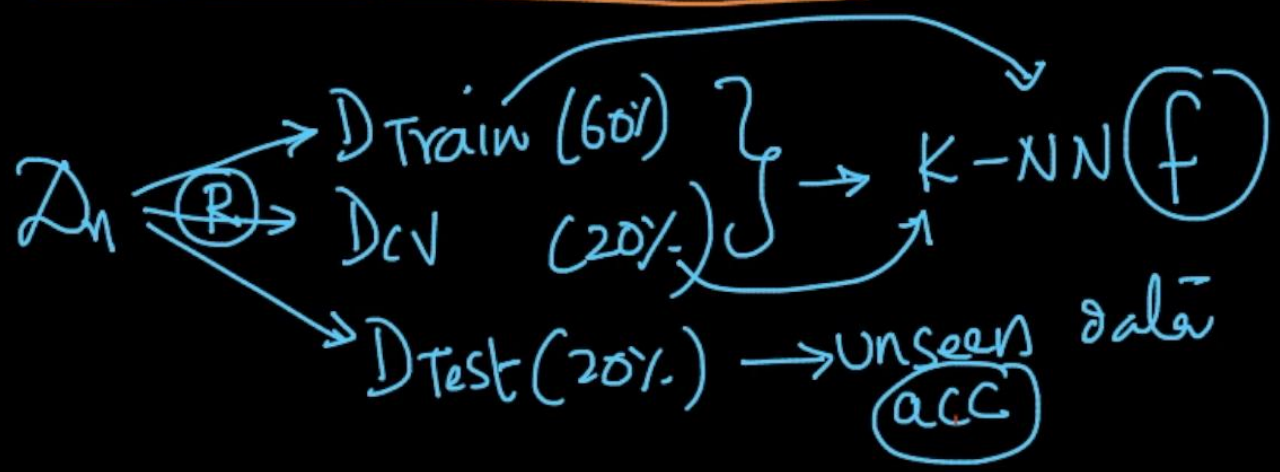
D_{test}

$Acc = 0.91 \Rightarrow$ 91% of times

$(x_q) \longrightarrow (y_q)$

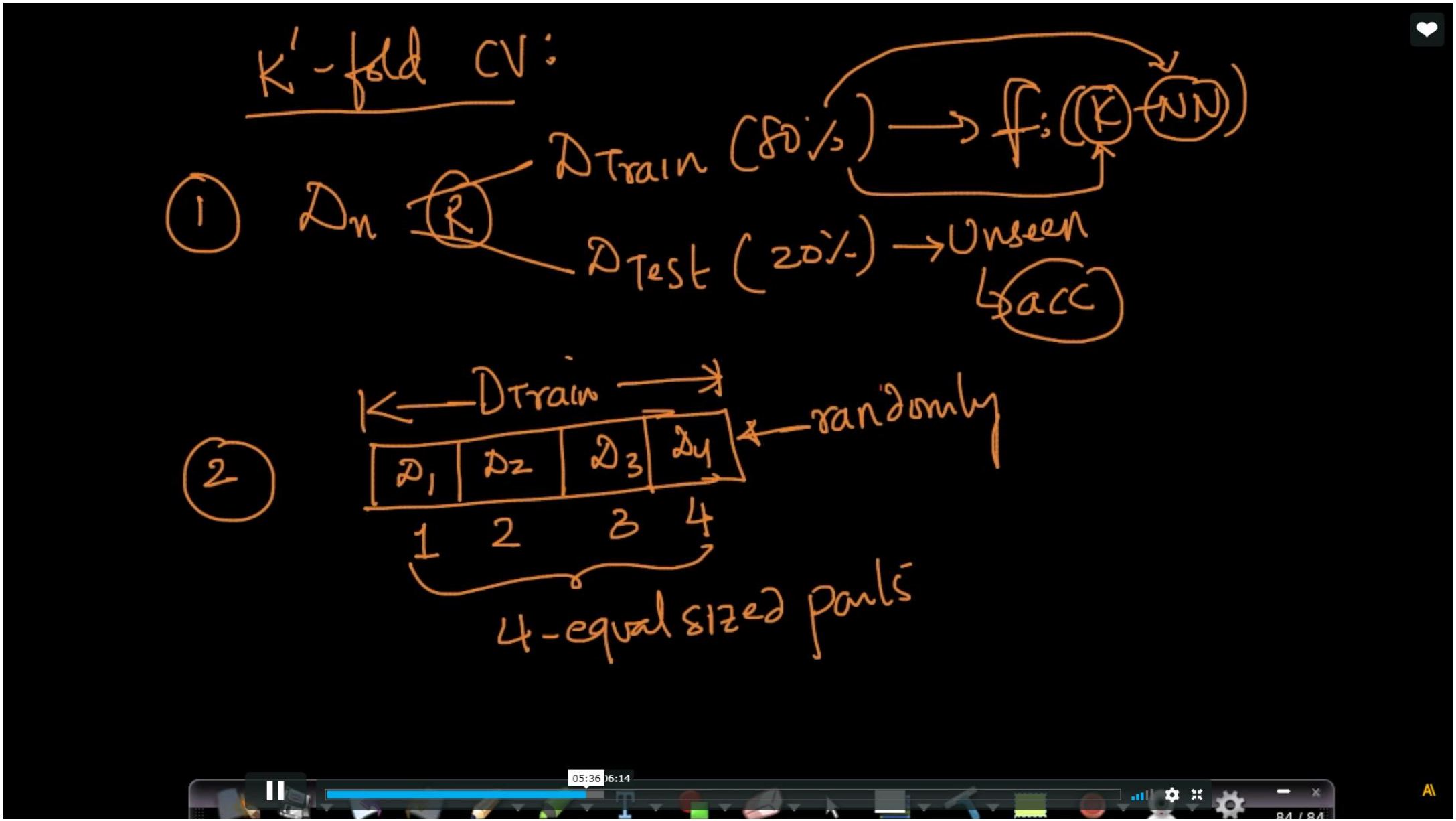

Ink2Go Edit Pages Annotation Screen Capture Window Help
Sat 8:26 AM
K=3
D Train
x
D Train
each point x_i in D test = $\{(x_i, y_i)_{i=1}^{n_2}\}$
x_q = x_1
y_q = Blue +ve
for each pt in D test
→ x_q = pt
→ use D Train & KNN to predict y_q
Type here to search
ENG
23:24
02-07-2019

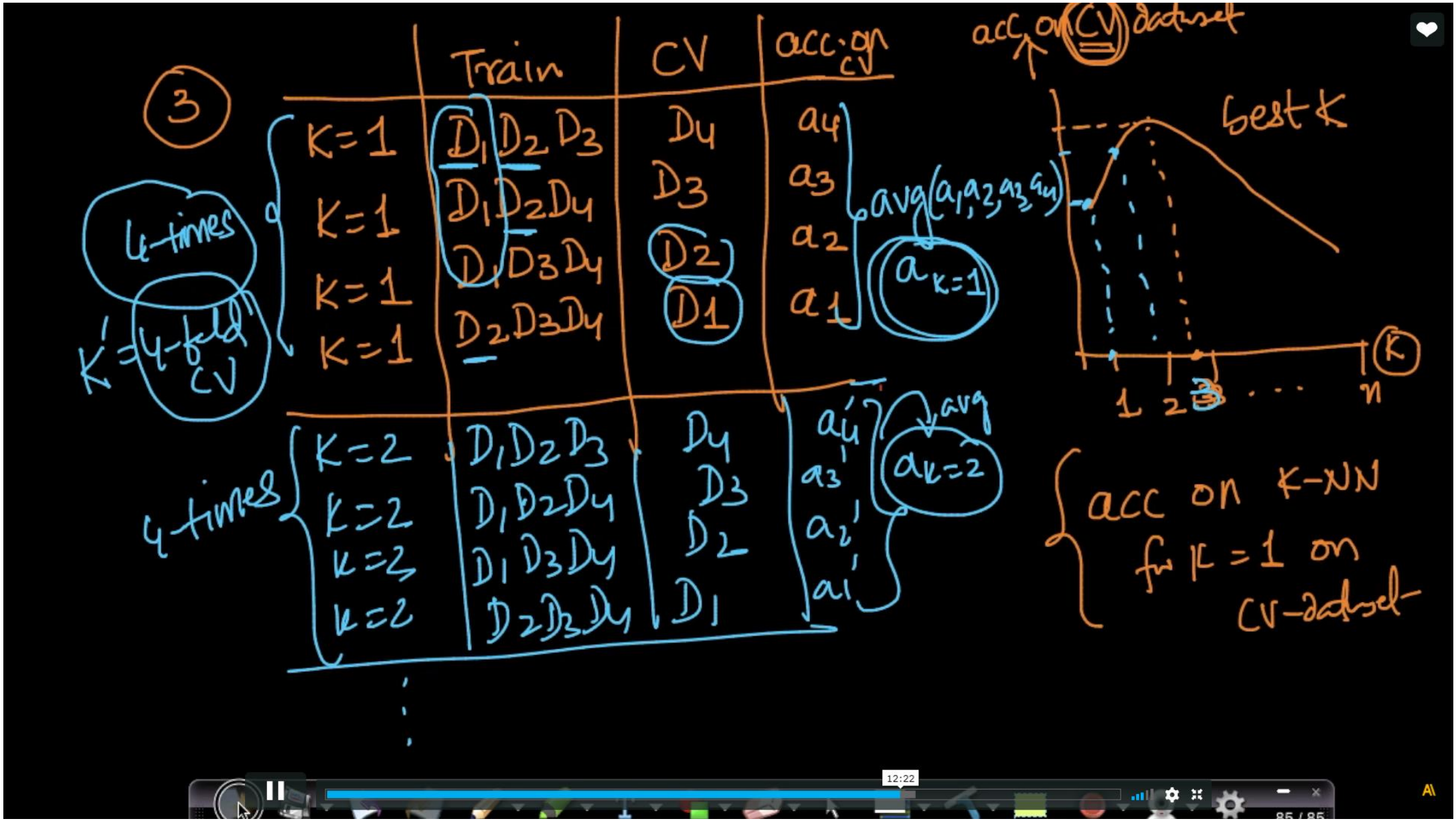
K-fold Cross-validation

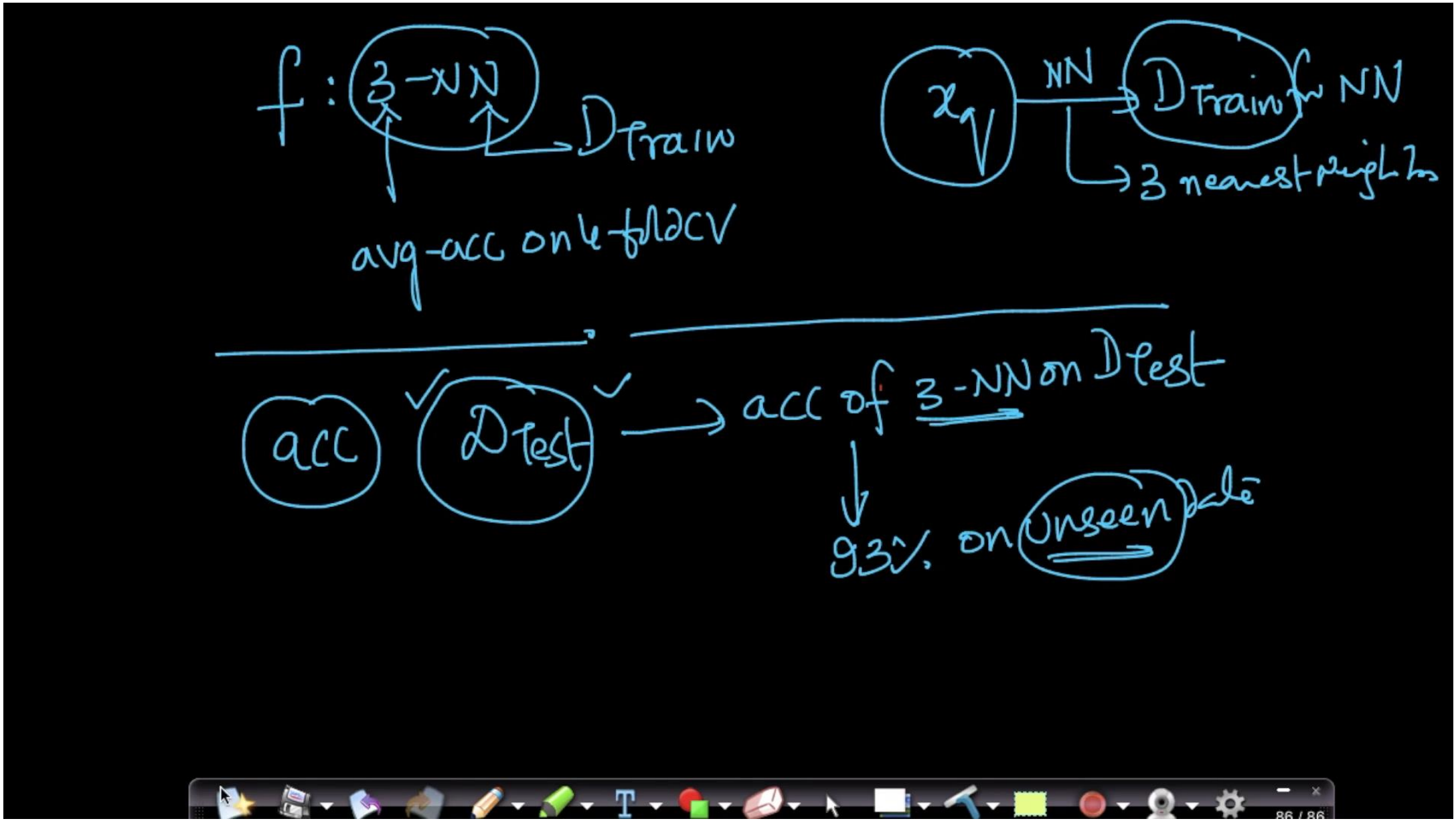


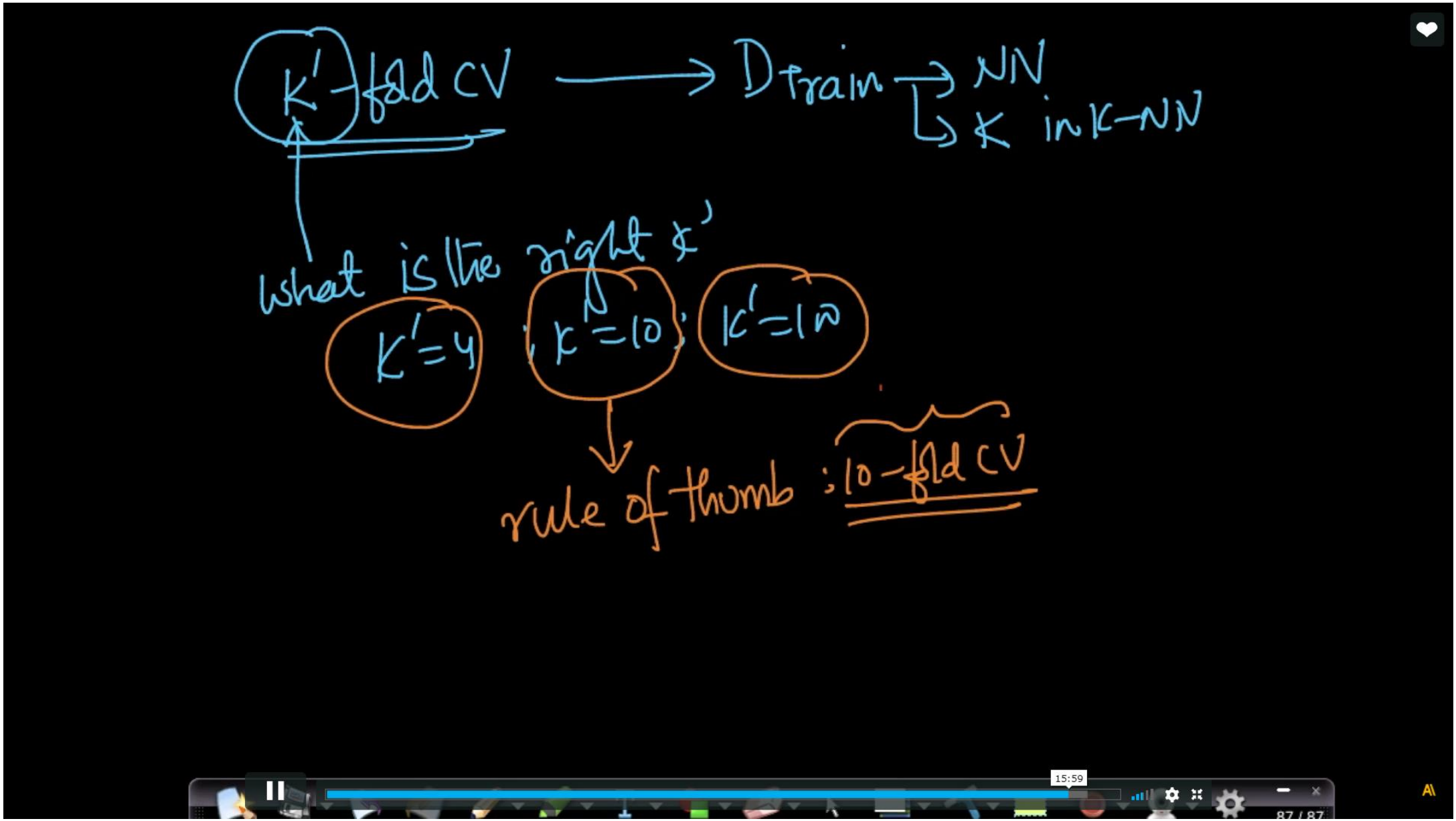
Problem: Only 60% of data (NN) } 80% of data to compute NN
20% CV → (K) }

20% Test → unseen

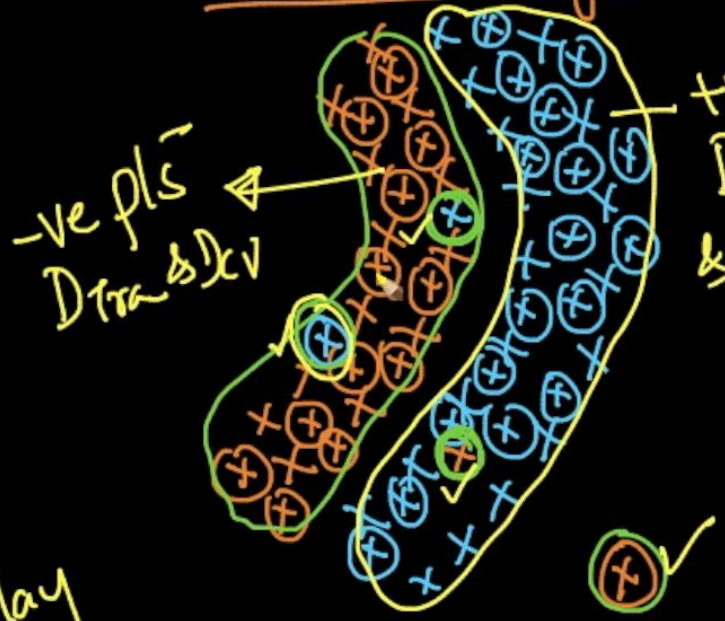

K'-fold CV:
1
D_n
R
D_{Train} (80%)
D_{Test} (20%)
→ Unseen
acc
f: (K) (NN)
2
D_{Train}
randomly
D_1
D_2
D_3
D_4
1 2 3 4
4-equal sized parts


3
4-times
K' = 4-fold CV
Train
CV
acc on CV
K=1
K=2
D1 D2 D3
D1 D2 D4
D1 D3 D4
D2 D3 D4
D4
D3
D2
D1
a4
a3
a2
a1
avg(a1,a2,a3,a4)
a K=1
4-times
a'4
a'3
a'2
a'1
avg
a K=2
acc on CV dataset
best K
1
2
3
n
K
acc on K-NN for K=1 on CV-dataset


f : 3-NN
D train
avg-acc on k-fold CV
x_q
NN
D Train for NN
3 nearest neighbors
acc
D test
acc of 3-NN on D test
93% on Unseen data

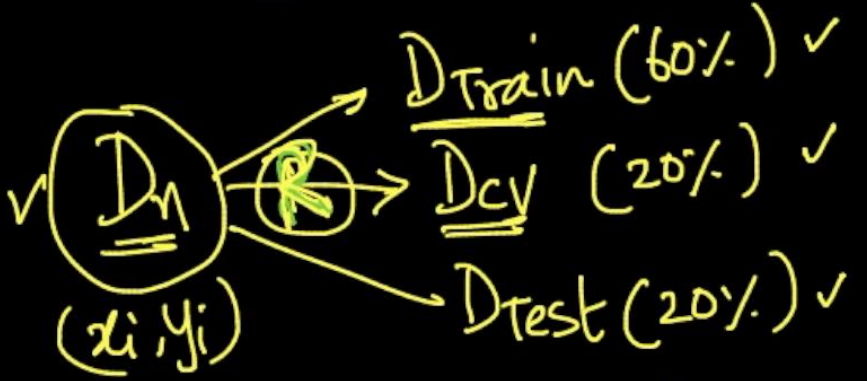

K-fold CV
D train
NN
K in K-NN
what is the right K?
K=4
K=10
K=100
rule of thumb : 10-fold CV
15:59

Visualizing Train, CV & test datasets:



overlay

{ (x) : -ve point from DCV
  (x) : +ve point from DCV



{ x : -ve datapoint in DTrain
  x : +ve datapoint in DTrain

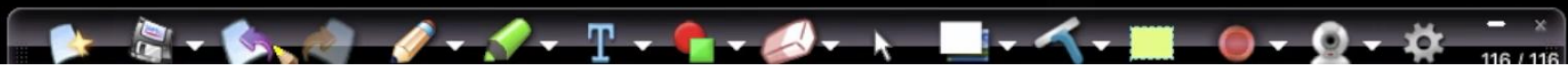
Train_CV_Test : (xi, yi)
↑ class-label

✓ (1) D_{train} & D_{cv} (and test) do not overlap perfectly when randomly sampled

(2) If there are many +ve/-ve pts from D_{train} in a region, then it is highly likely to find many +ve/-ve pts from D_{cv} in that region

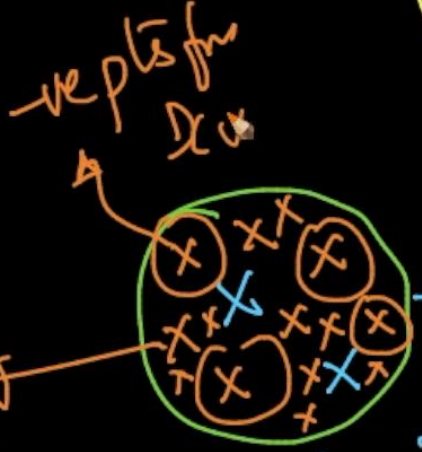
(3) If there are very few +ve/-ve pts in a region from D_{train}, then it is very unlikely to find from +ve/-ve from D_{cv} in that region

noise, error, outlier



intuitively,

density of +ve pts from Dtrain is high

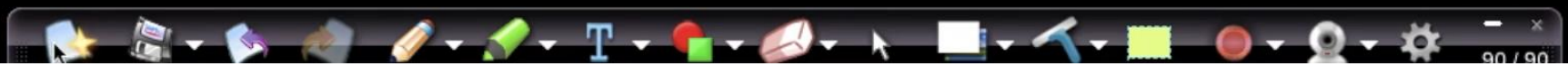


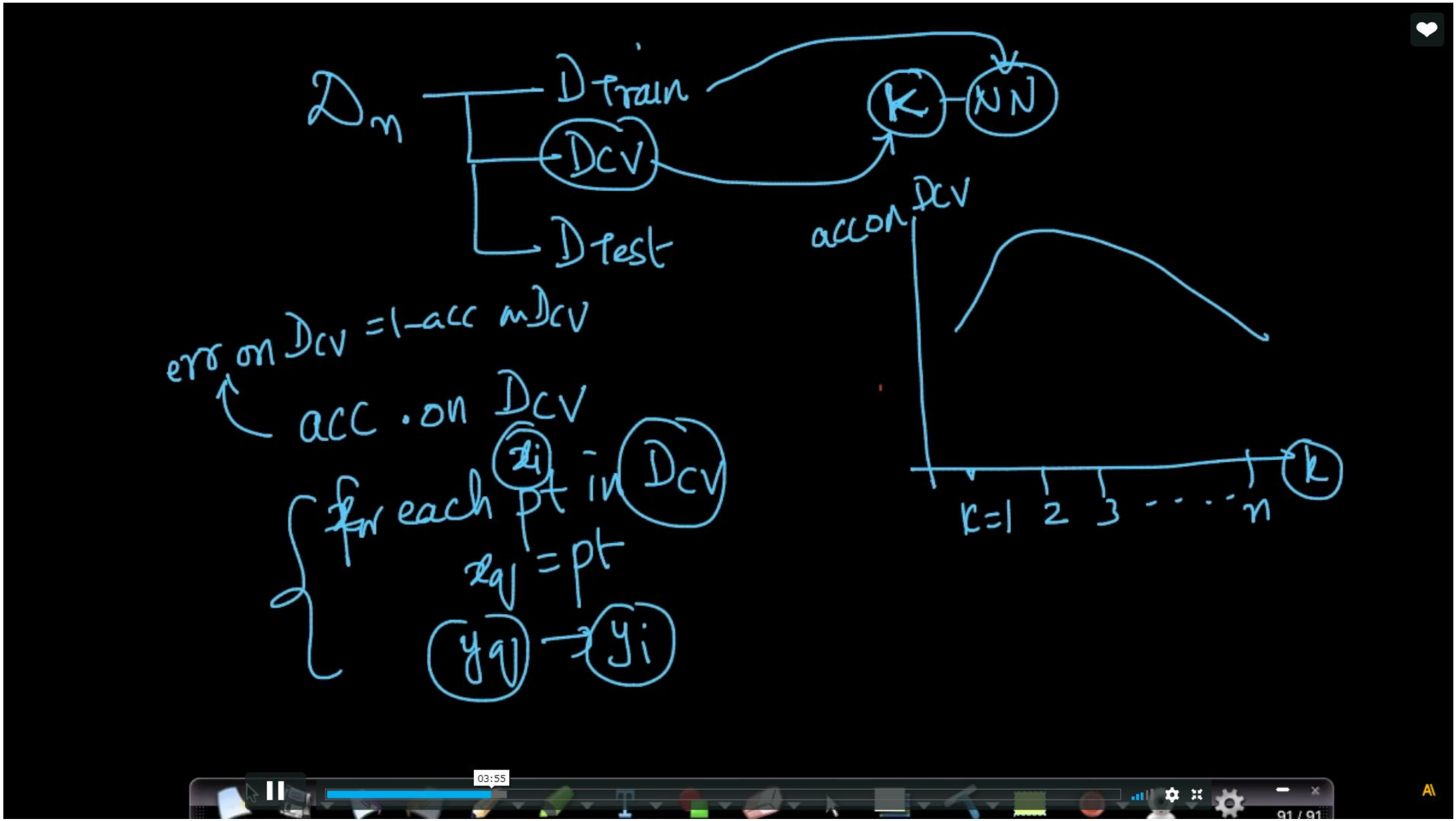
→ very low density

↳ may not find any pts from Dcv +ve class

$$(\max)\uparrow \; acc = \frac{\#\text{ correctly class pts}}{\text{Total }\#\text{ pts}} = 0.93$$

$$(\min)\downarrow \; \text{error} = 1 - acc = 0.07 \rightarrow 7\%$$




D_n
D train
Dcv
D test
K
NN
err on Dcv = 1 - acc on Dcv
acc . on Dcv
for each pt in Dcv
x_i
x_q = pt
y_q
y_i
acc on Dcv
k
K=1 2 3 . . . . n
03:55
91 / 91

Training error:

for each x_i in D_{train}
- find 2 nearest neigh to x_i from D train
- majority rule to get the class label
- if y_i == class label accurate
  else error

$D_{train} \rightarrow NN$

$D_{CV} \rightarrow K$

$f[K\text{-}NN]$

What is the train error if 2-NN

$D_{train} \rightarrow$ Training

$D_{train} \rightarrow$ acc/error

Overfitting vs Underfitting:

{ K-fold CV (or) Dcv → best K → neither overfit nor underfit

✓(Q) How can we be sure that we are not underfitting (or) overfitting? → plot

Training error:

for each x_i in D_{train}

- find 2 nearest neigh to x_i from D train
- majority rule to get the class label
- if $y_1 == $ class label
  - accurate
  - else error

$D_{train} \rightarrow NN$

$D_{cv} \rightarrow K$

$f[K\text{-}NN]$

what is the train error if 2-NN

Training error:

for each x_i in D_{train}
- find 2 nearest neigh to x_i from D train
- majority rule to get the class label
- if y_i == class label
  - accurate
  - else error

$D_{train} \rightarrow NN$

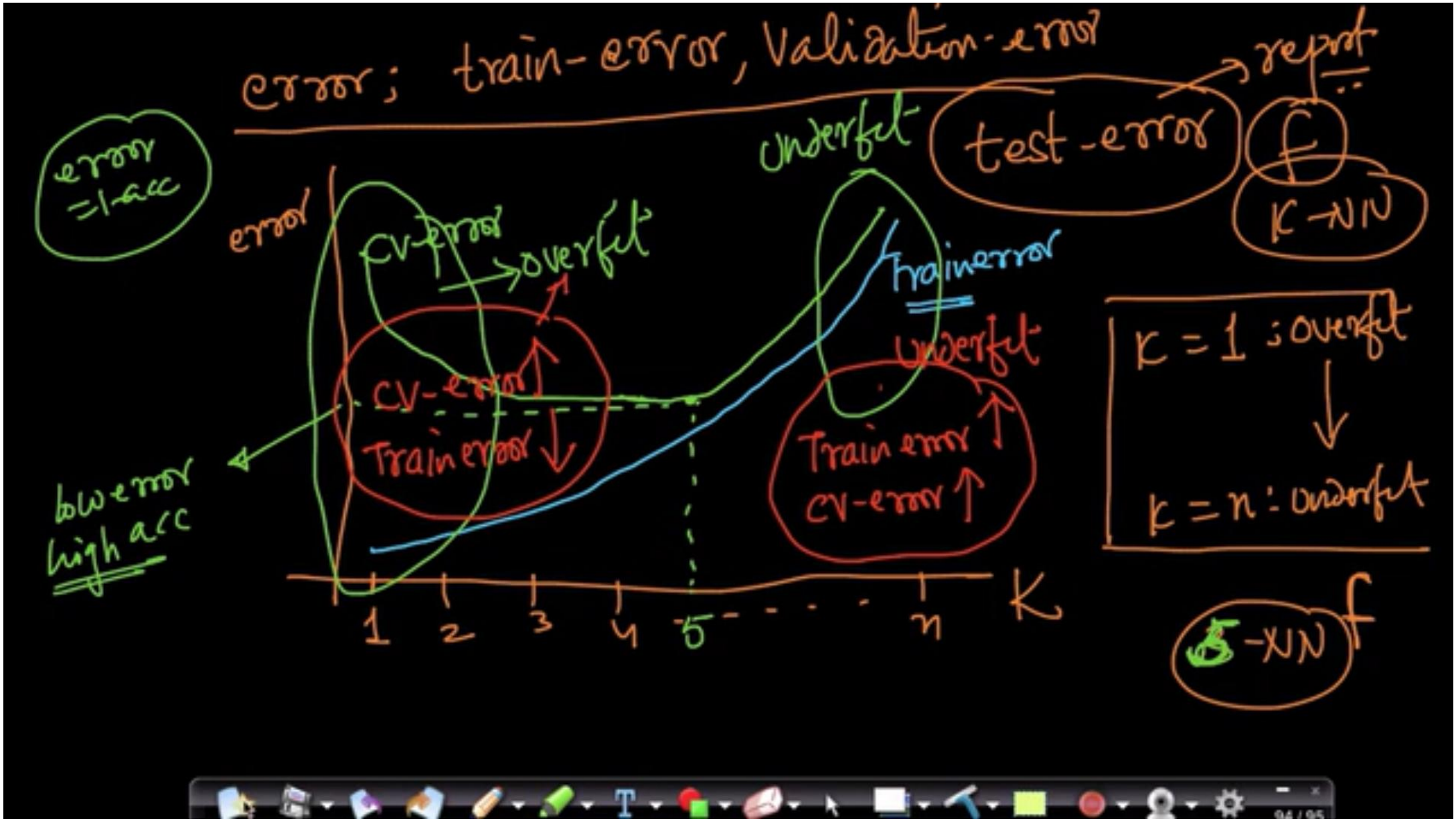
$D_{CV} \rightarrow K$

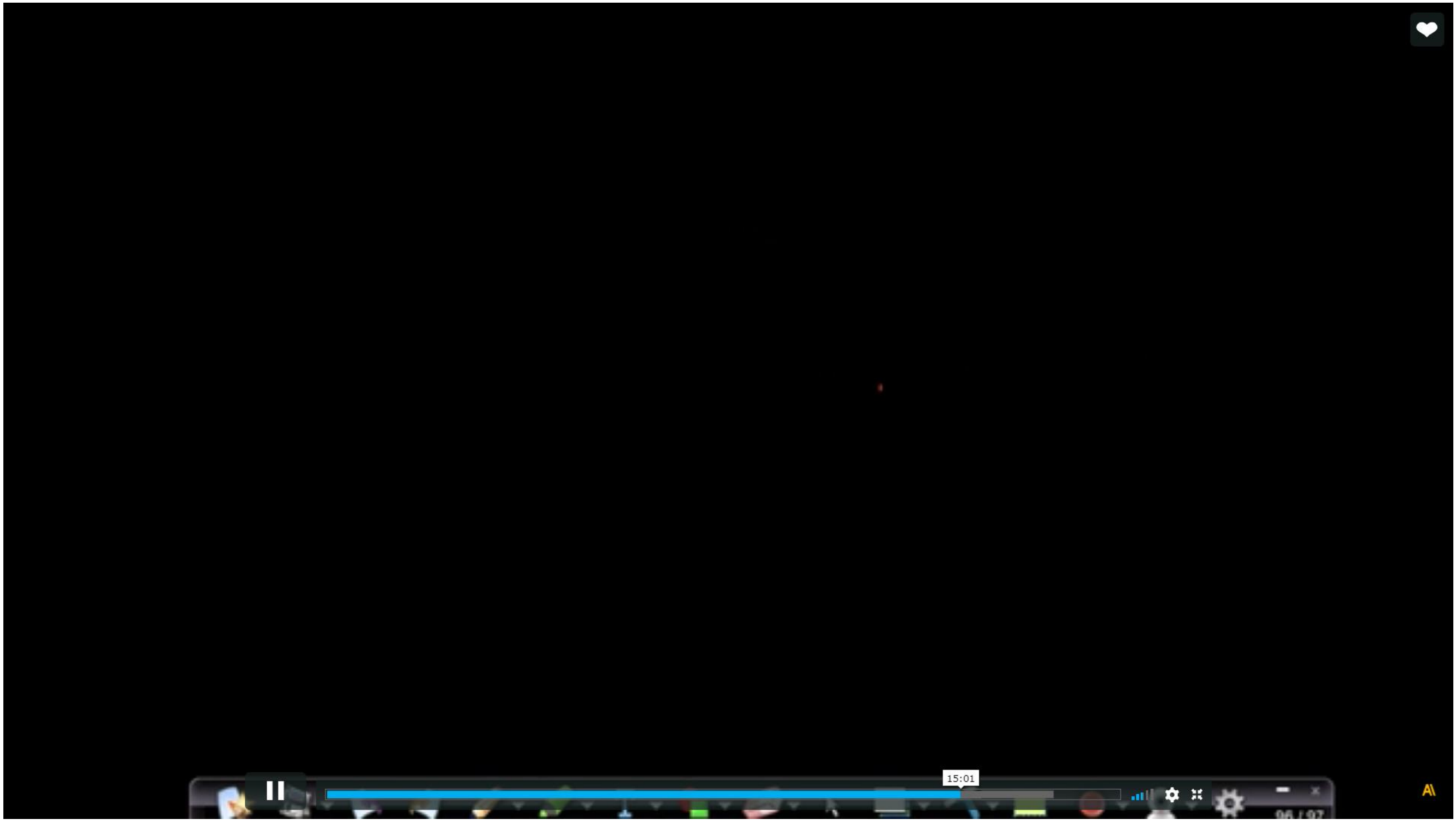
for [K-NN]

What is the train error if 2-NN

$D_{train} \rightarrow$ Training

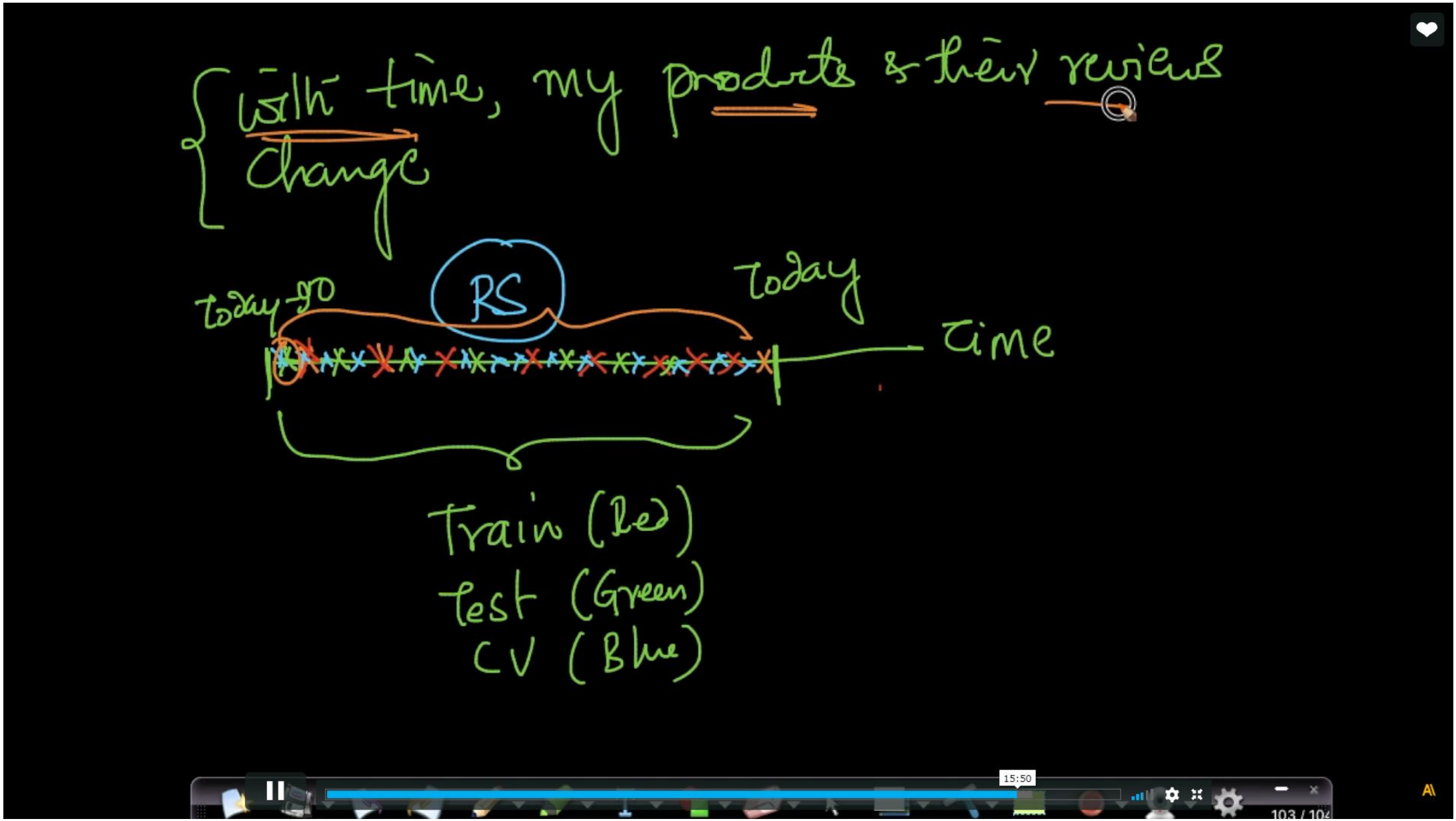
$D_{train} \rightarrow$ acc/error

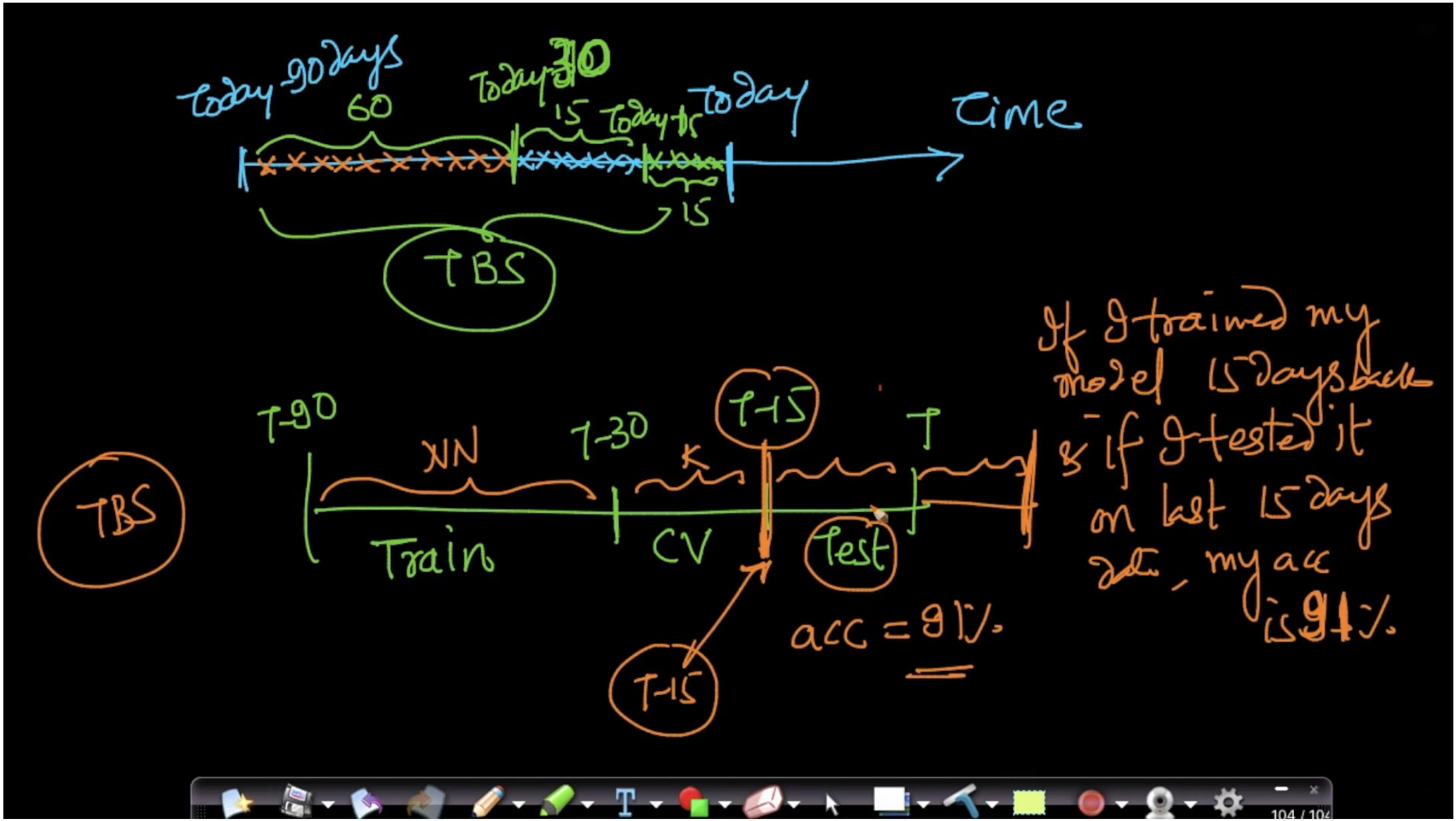

error; train-error, Validation-error
error =1-acc
error
cv-error
overfit
underfit
test-error
report
f
K-NN
Train error
low error
high acc
CV-error
Train error
Train error
CV-error
K = 1 : overfit
K = n : underfit
1
2
3
4
5
n
K
5-NN
f


15:01

Train error is high
Val-error is high
} — Underfit

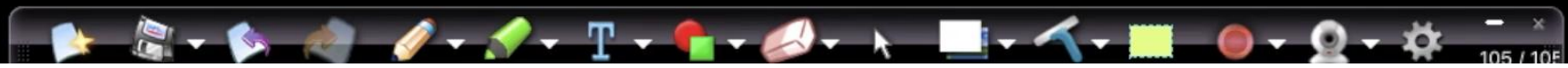
Train error is low
Val-error is high
} — overfit


With time, my products & their reviews change
today-go
RS
today
time
Train (Red)
Test (Green)
CV (Blue)
15:50
103 / 104


today 90 days
60
today 30
15
today 15
today
time
15
TBS
TBS
T-90
NN
T-30
K
T-15
T
Train
CV
Test
T-15
acc = 91%
If I trained my model 15 days back & if I tested it on last 15 days data, my acc is 91%.

(*) Whenever time is available & if things / behaviour or data changes over time

then time based splitting is preferable



K-NN for regression:

2-class classfn

$$D = \left\{ (x_i, y_i)_{i=1}^{n} \;\middle|\; x_i \in \mathbb{R}^d, \; y_i \in \{0, 1\} \right\}$$

$f(x_q) \rightarrow y_q \rightarrow$ class-label

✓ find → K-NN → Majority vote

Regfn

$$D = \left\{ (x_i, y_i)_{i=1}^{n} \;\middle|\; x_i \in \mathbb{R}^d; \; y_i \in \mathbb{R} \right\}$$

$x_q \rightarrow y_q \rightarrow$ number

(1) given x_q, find K-nearest neighbours

$(x_1, y_1), (x_2, y_2) \ldots (x_k, y_k)$

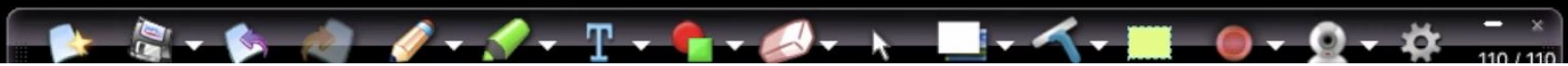
Classification → Majority vote

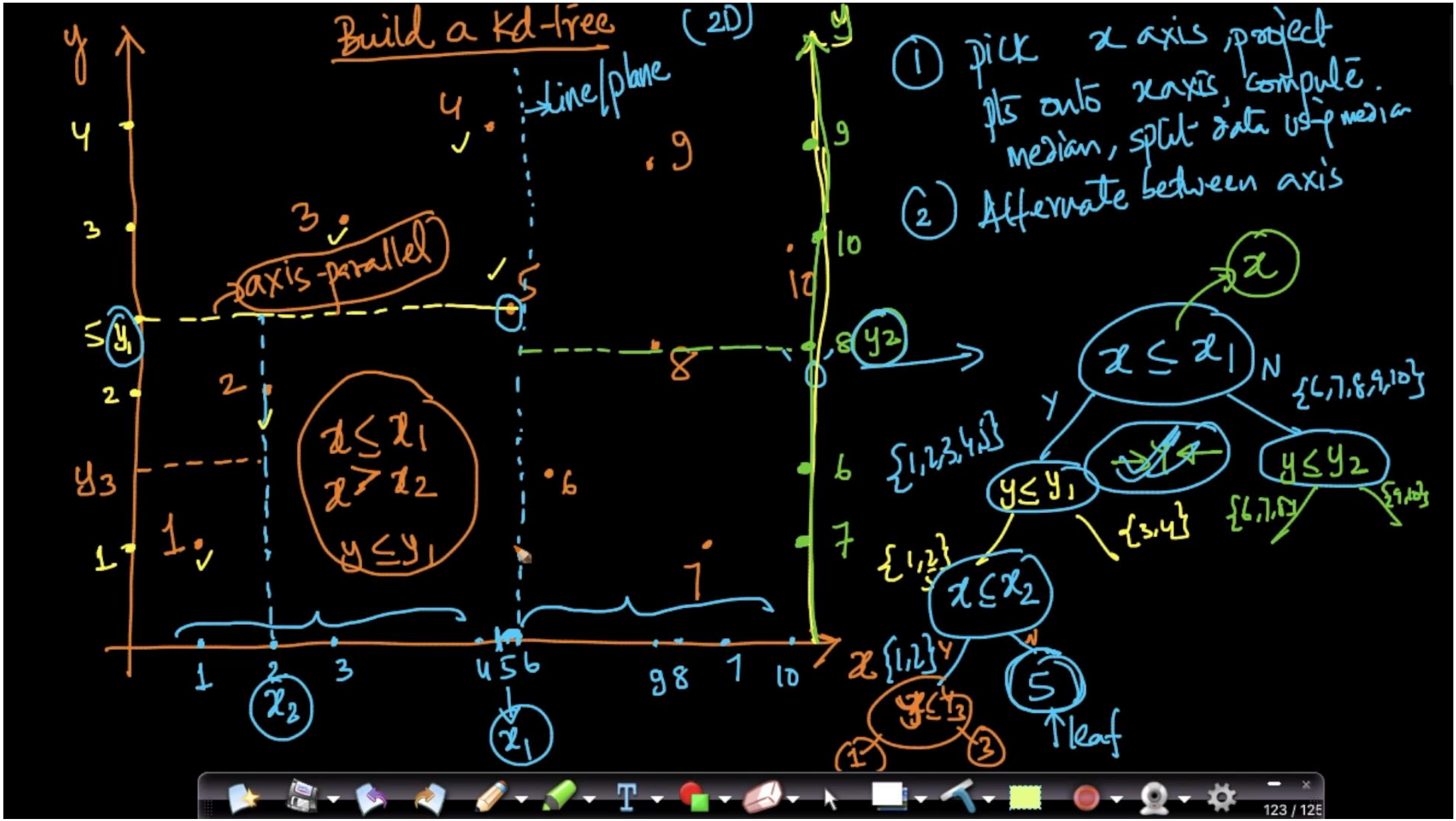
Regression

(2) $y_q \leftarrow y_1, y_2, y_3 \ldots y_k$ → not 0 or 1 (R)

$$y_q = \text{mean}(y_i)_{i=1}^{k}$$

$$y_q = \text{median}(y_i)_{i=1}^{k} \rightarrow \text{less prone to outliers}$$

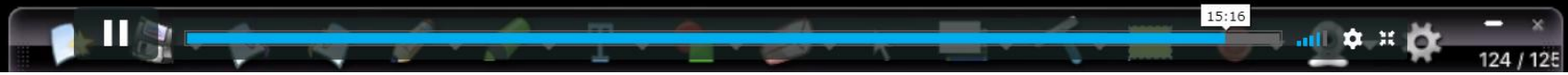


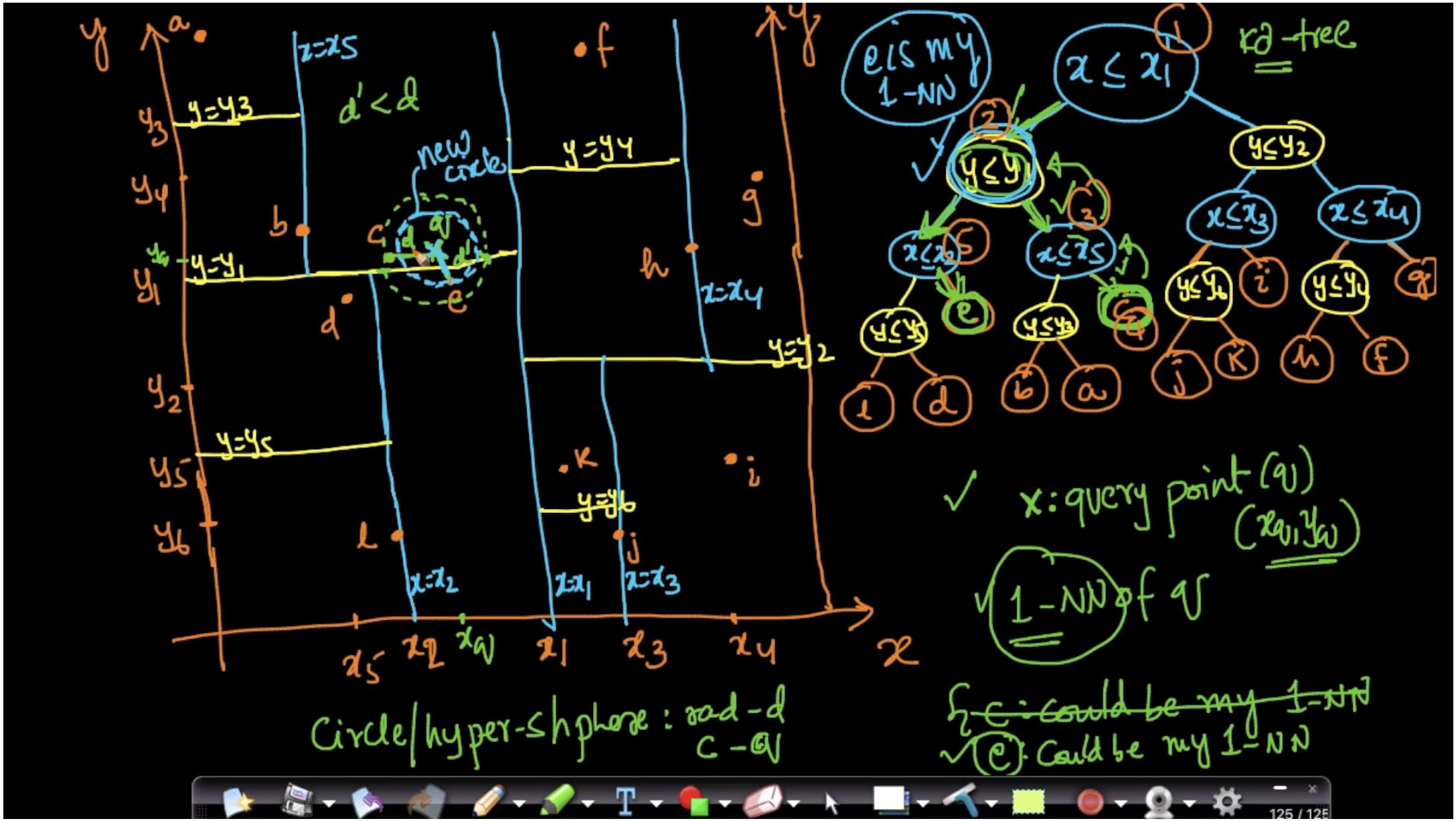

Build a Kd-tree (2D)
line/plane
axis-parallel
① pick x axis, project pts onto x axis, compute median, split data using median
② Alternate between axis
x ≤ x1
y ≤ y1
x ≤ x2
y ≤ y2
leaf
{1,2,3,4,5}
{6,7,8,9,10}
{3,4}
{1,2}
{6,7,8}
{9,10}

Kd-tree → go through (alternate) each axis one after the other till leaf node

↳ breaking up my space using axis-parallel lines/planes into rectangles/cuboids/hyper-cuboids

- 2D → axis parallel lines → rectangles
- 3D → " " planes → cuboids
- nD → " " hyperplanes → hypercuboids




kd-tree
Circle/hyper-sphere : rad-d
c-q
x: query point (q)
(xq, yq)
1-NN of q
{c: could be my 1-NN
e: Could be my 1-NN

Time-complx

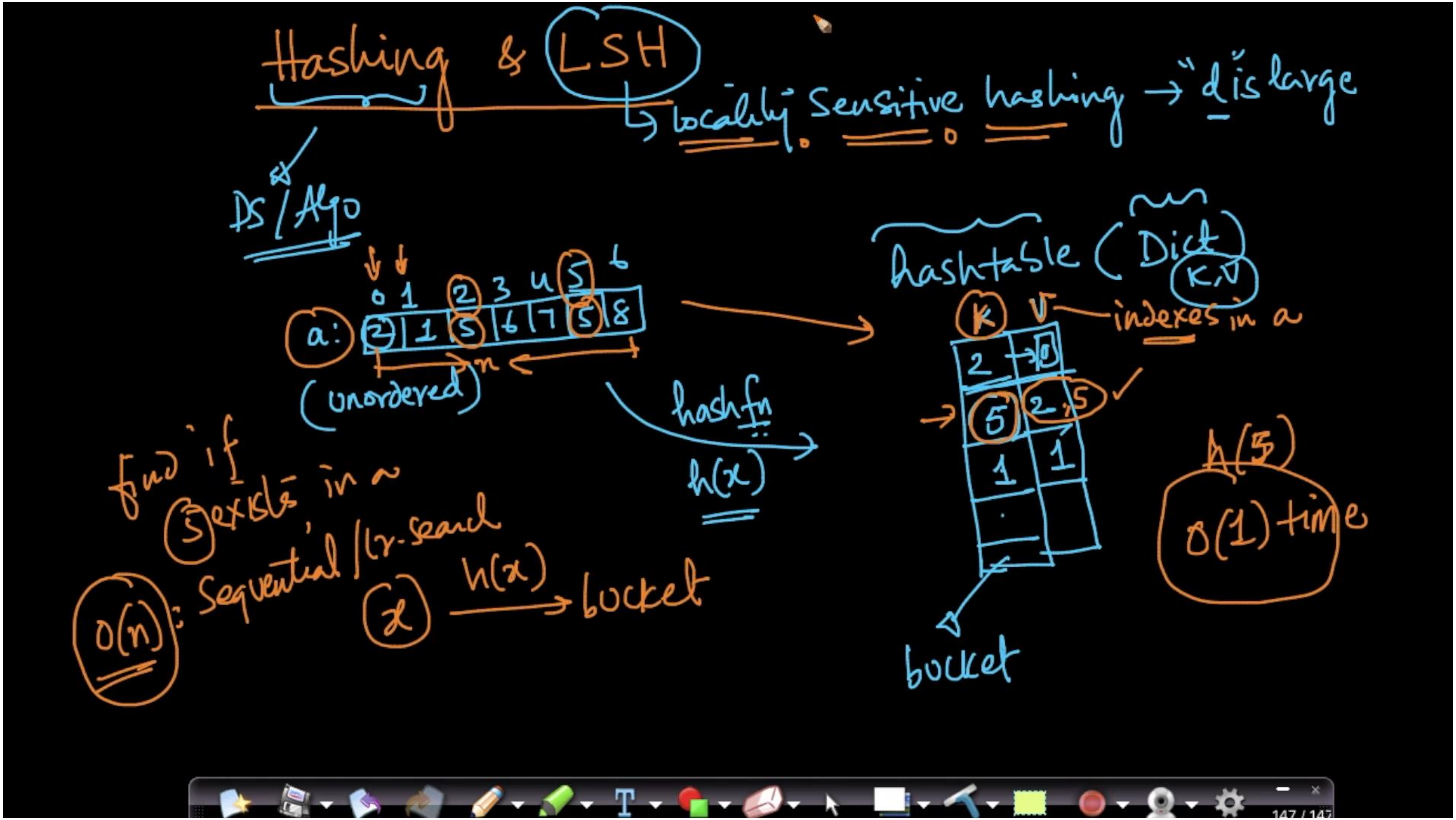
comparisions to find 1-NN

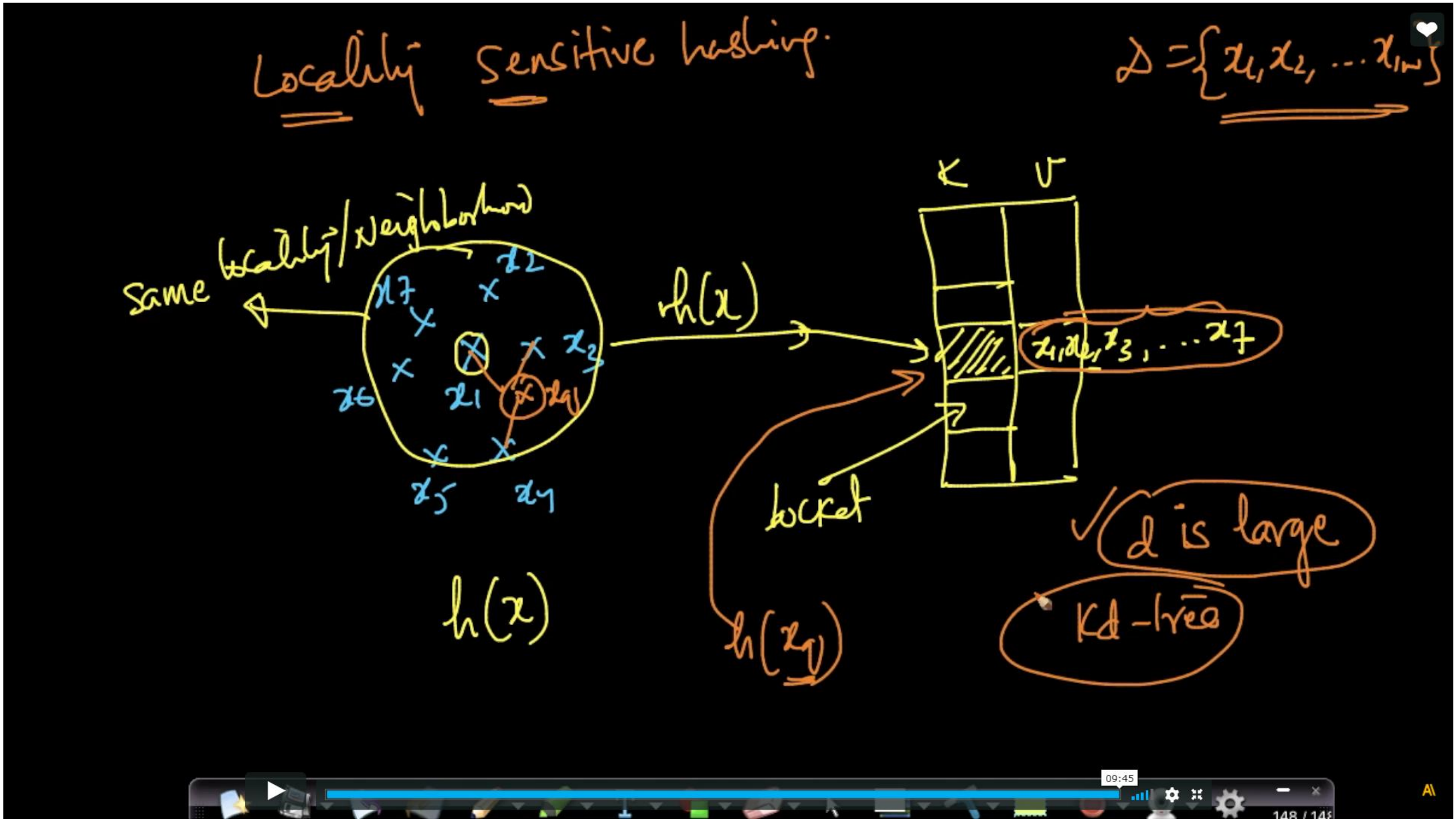
- ✓ best-case: $O(\lg(n))$
- ✓ Worst-case: $O(n)$

n : # pts

$O(\lg n)$

perfect analysis :- very complex

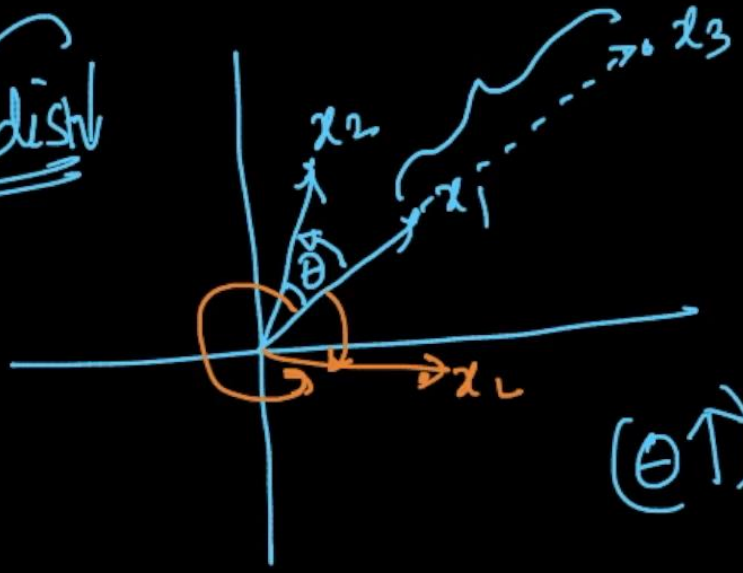

Hashing & LSH
locality sensitive hashing → "d" is large
DS / Algo
0 1 2 3 4 5 6
a: 2 1 5 6 7 5 8
n
(unordered)
find if 5 exists in a
O(n): sequential / lr-search
x h(x) → bucket
hash fn
h(x)
hashtable (Dict)
K,V
K V ← indexes in a
2 0
5 2, 5
1 1
bucket
h(5)
O(1) time
147 / 147


Locality Sensitive hashing.
D = {x1, x2, ... x1m}
Same locality/neighborhood
x7
x2
x3
x6
x1
x4
x5
h(x)
K
V
x1, x2, x3, ... x7
bucket
h(x)
h(x4)
d is large
Kd-tree

Locality Sensitive hashing for Cosine Similarity

L_2, L_1, Hamming

sim ↑, dist ↓



$\cos\text{-sim}(x_1, x_2) = \cos(\theta)$

$\cos\text{-sim}(x_1, x_3) = \cos(\theta) = 1$

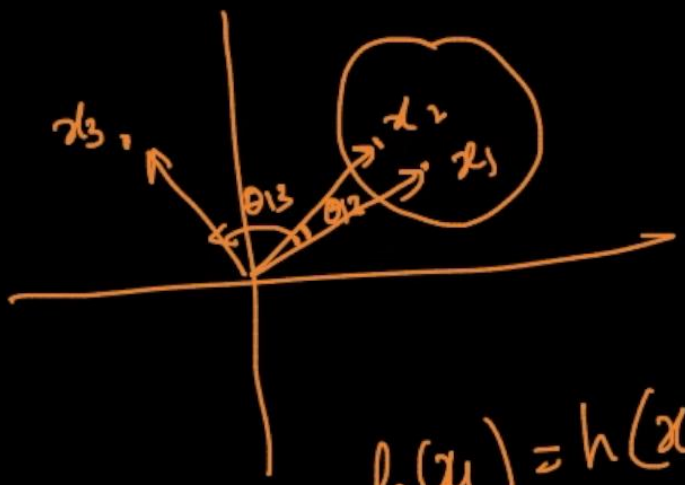
$(\theta \uparrow)$; → (x_1 & x_2 are "angularly" seperated ↑) → (dist ↑)

(Cosine ↓) (Sim ↓)

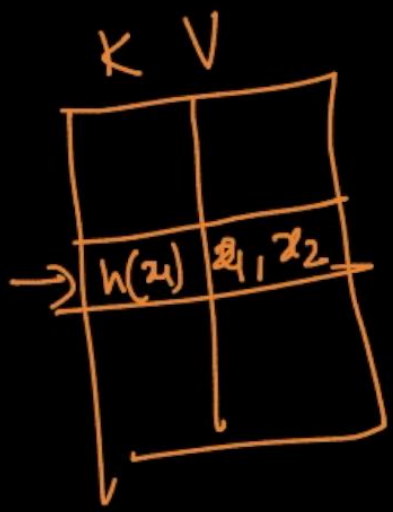
Cosine-Sim ≈ angular-similarity

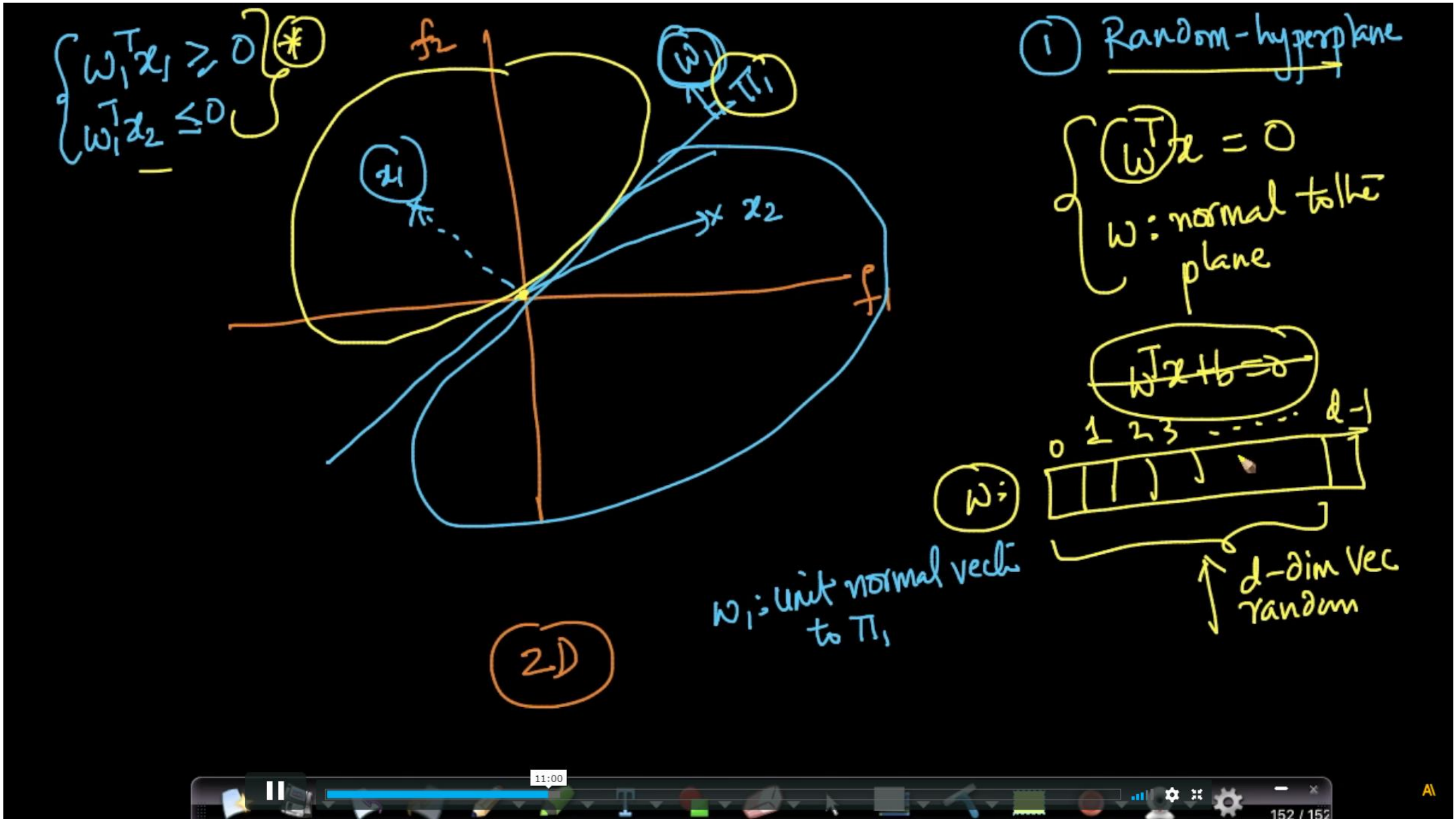
LSH fn cos-Sim

$\theta_{12} \sim 0$ (small) $\Rightarrow$ x_1 & x_2 are similar/close

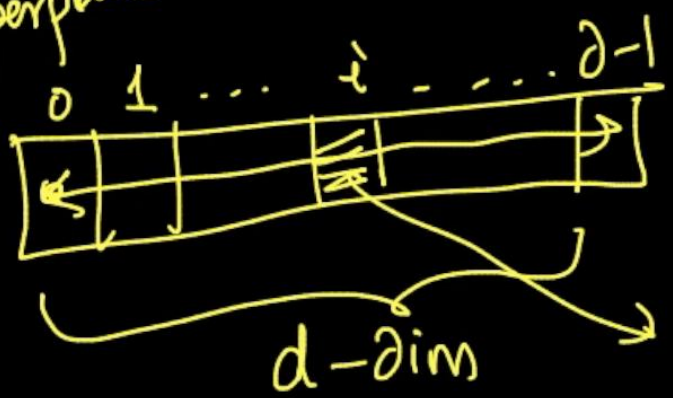


$h(x_1) = h(x_2) = key$



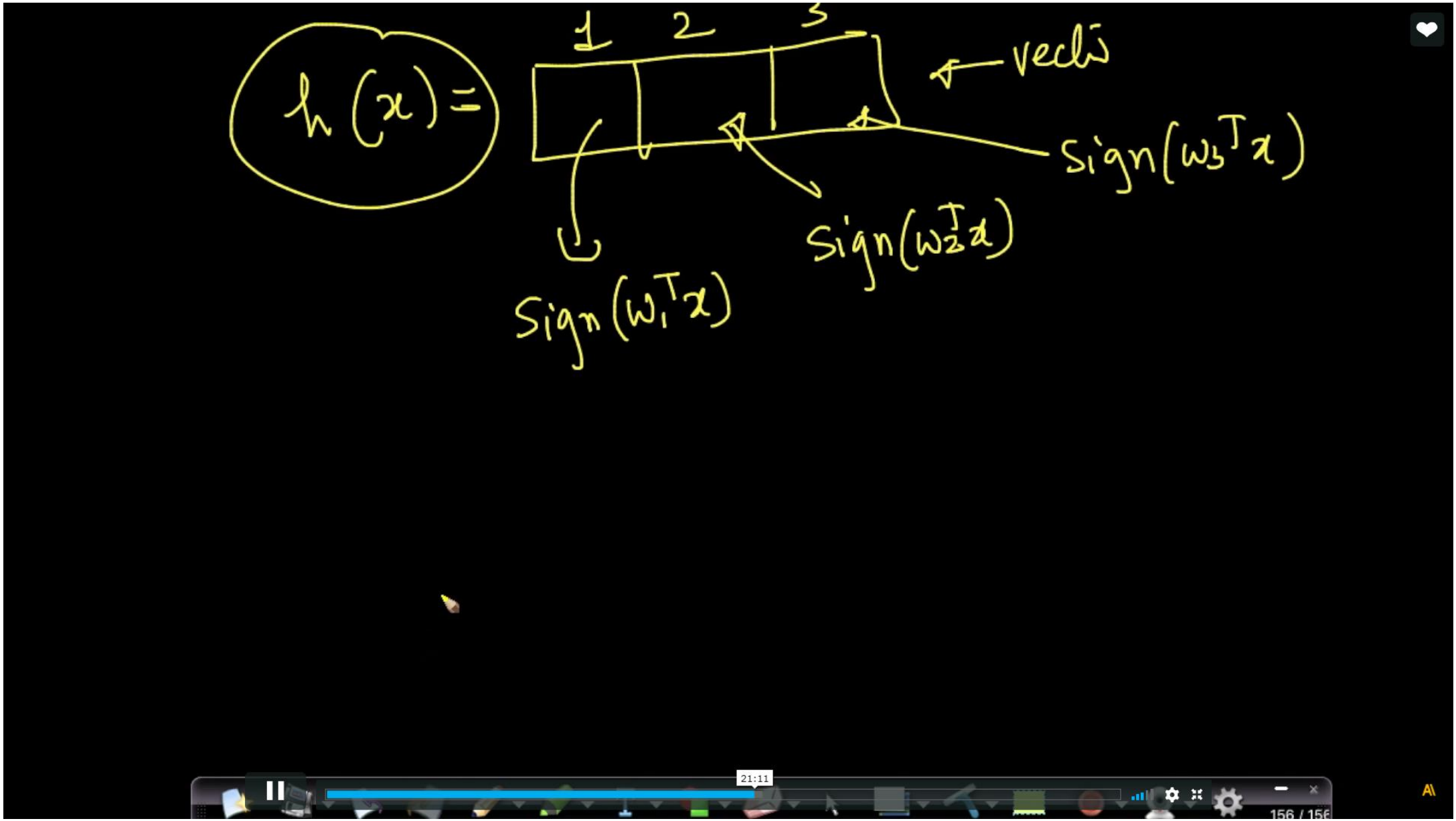

$\begin{cases} w_1^T x_1 \geq 0 \\ w_1^T x_2 \leq 0 \end{cases}$ (*)
f_2
f_1
x_1
x_2
w_1
π_1
2D
w_1: Unit normal vector to π_1
1 Random-hyperplane
$w^T x = 0$
w: normal to the plane
$w^T x + b = 0$
0 1 2 3 ..... d-1
w_i
d-dim Vec random
11:00
152 / 152

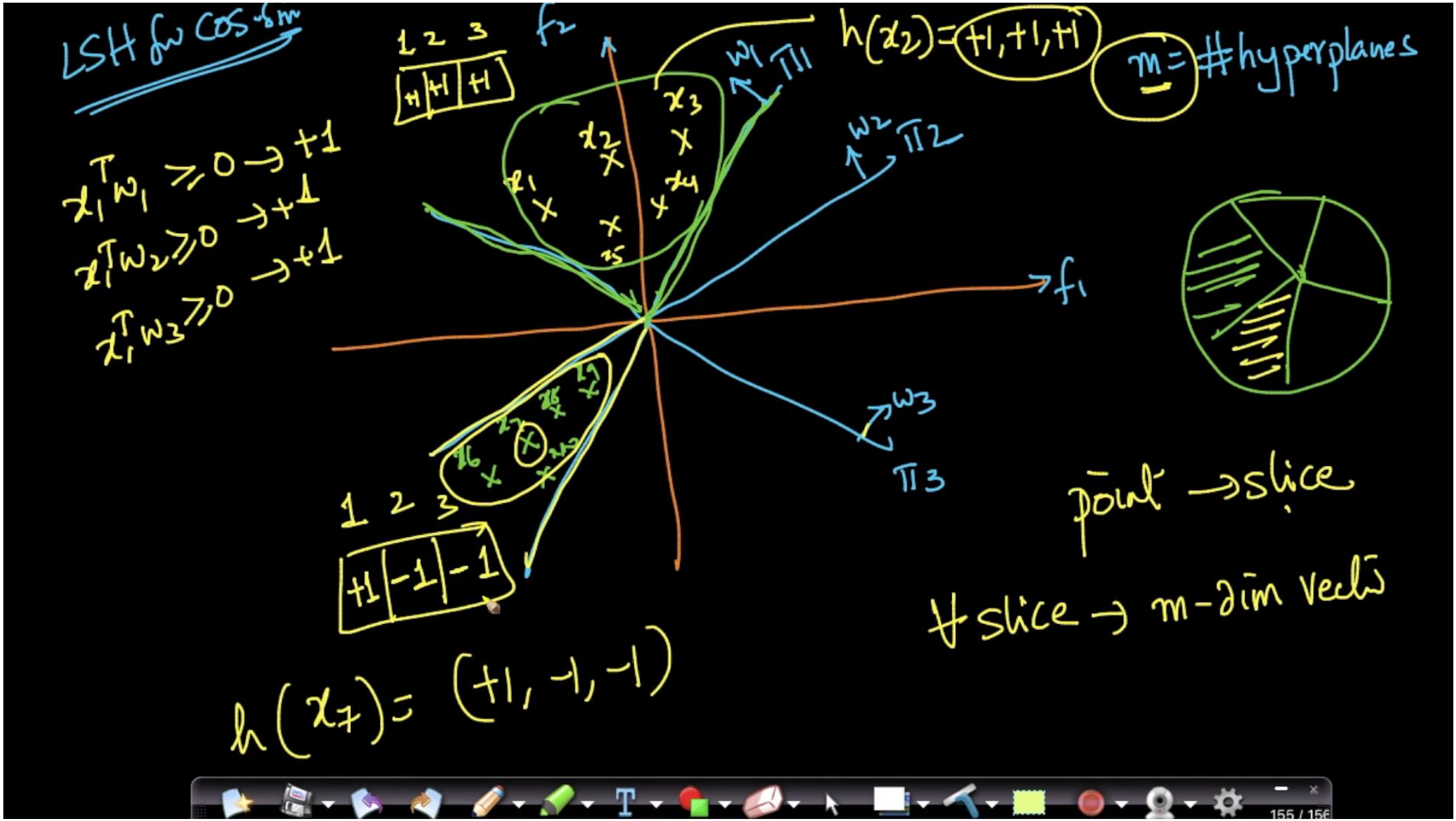
Random-hyperplane

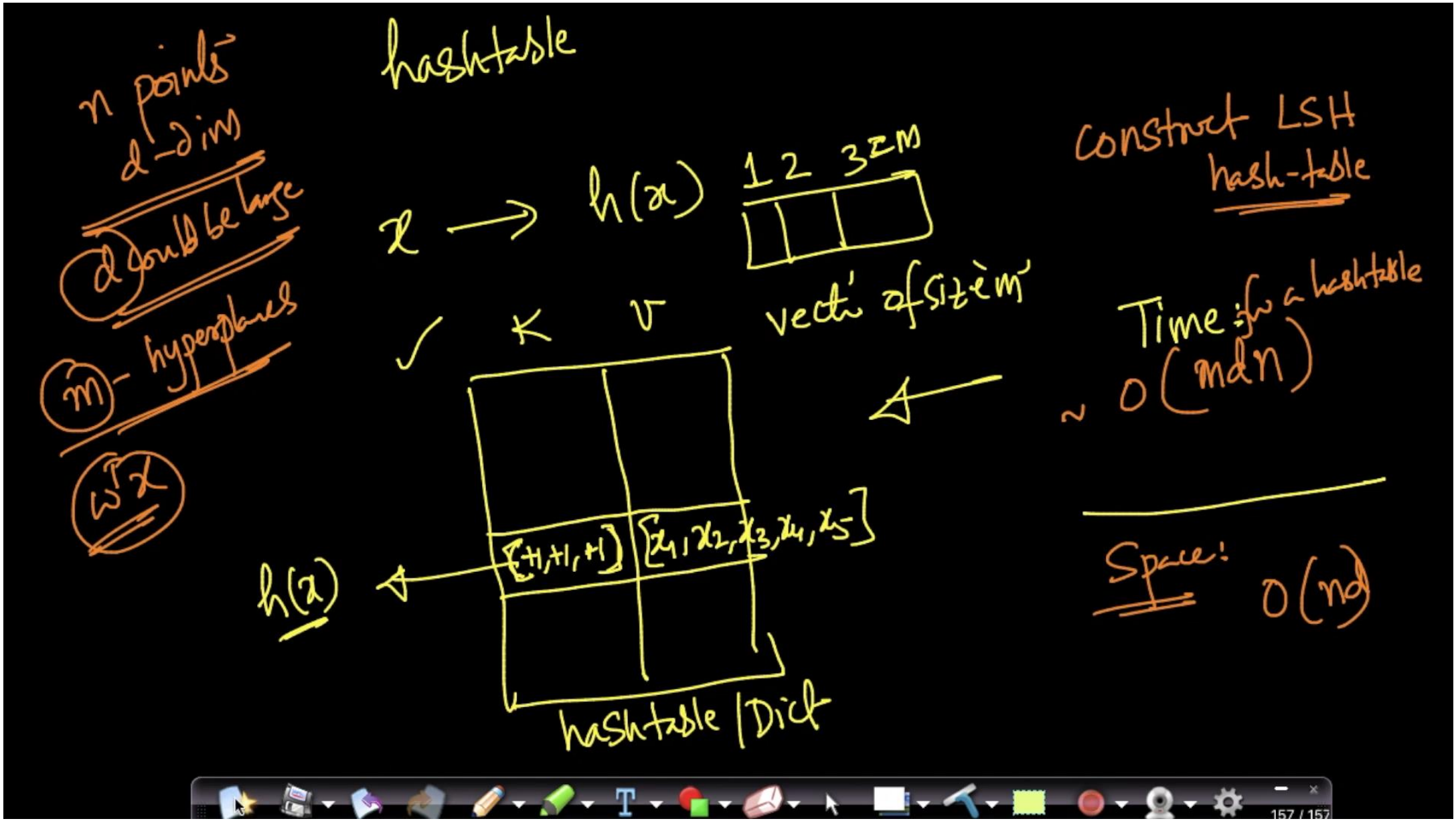


random numbers sampled from $N(0,1)$

$w = \text{numpy.random.normal}(0, 1, d)$

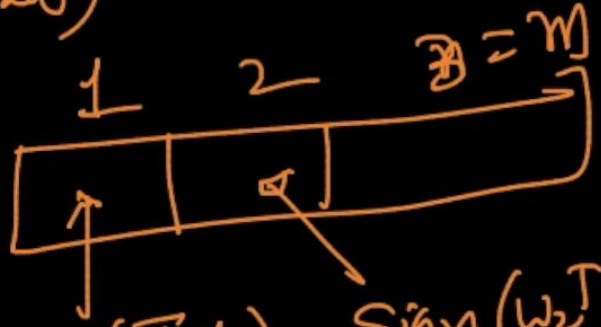

h(x) =
1
2
3
vecli
Sign(w3^T x)
Sign(w2^T x)
Sign(w1^T x)
21:11
156 / 156


LSH fw cos sim
1 2 3
+1 +1 +1
f2
h(x2) = +1,+1,+1
m = # hyperplanes
w1
π1
w2
π2
w3
π3
f1
x1
x2
x3
x4
x5
x6
x7
x1ᵀw1 ≥ 0 → +1
x1ᵀw2 ≥ 0 → +1
x1ᵀw3 ≥ 0 → +1
1 2 3
+1 -1 -1
h(x7) = (+1, -1, -1)
point → slice
∀ slice → m-dim vector


n points
d-dim
d could be large
m - hyperplanes
w.r.t
hashtable
x → h(x)
1 2 3 ... m
vector of size m
K
V
h(x)
{+1,+1,+1}
[x1, x2, x3, x4, x5]
hashtable / Dict
Construct LSH hash-table
Time: for a hashtable
~ O(mdn)
Space: O(nd)

Query point (x_q)

① $h(x_q) =$



$h(x_q)$ as the key in my hash table

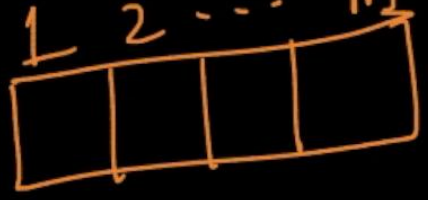
- d.get($h(x_q)$) → $\{x_1, x_2, x_3, x_4, x_5\}$

x_q

given a hash table:-

Time complx for querying:

z_q: $h(z_q)$



$O(mq)$

$O(md + n'd)$

small

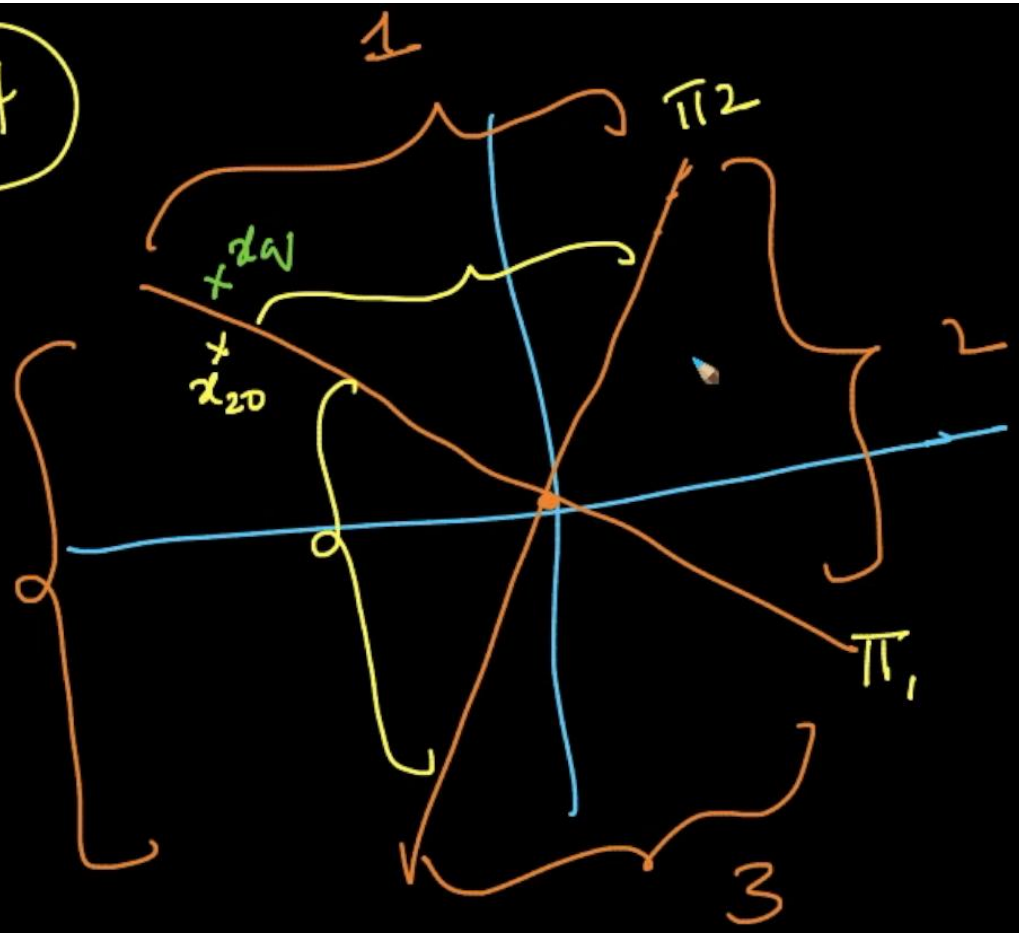
$O(md)$

$z_q \rightarrow$ n' elements in the bucket/slice

- if n' is small

$n' \approx m$

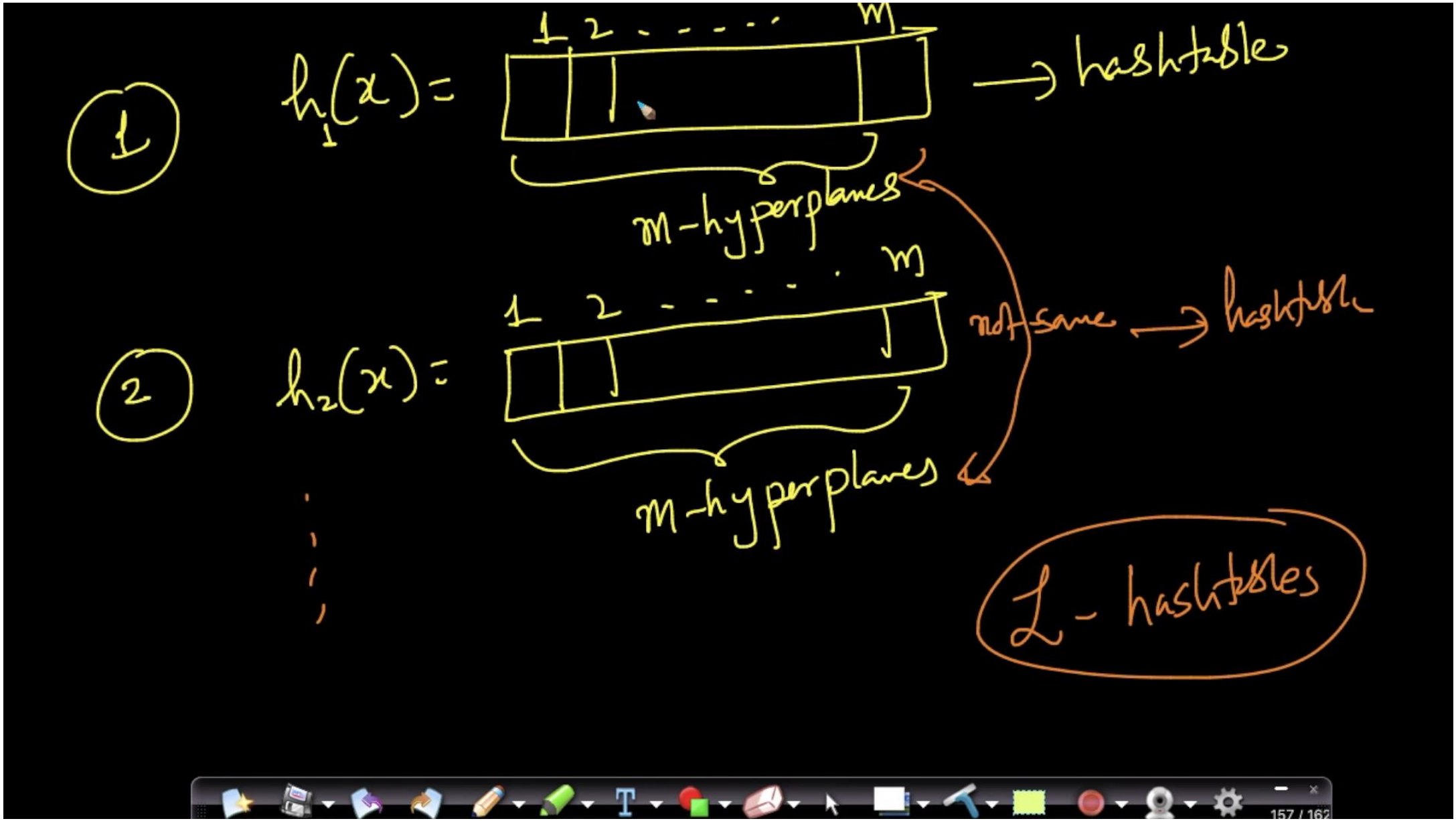
LSH

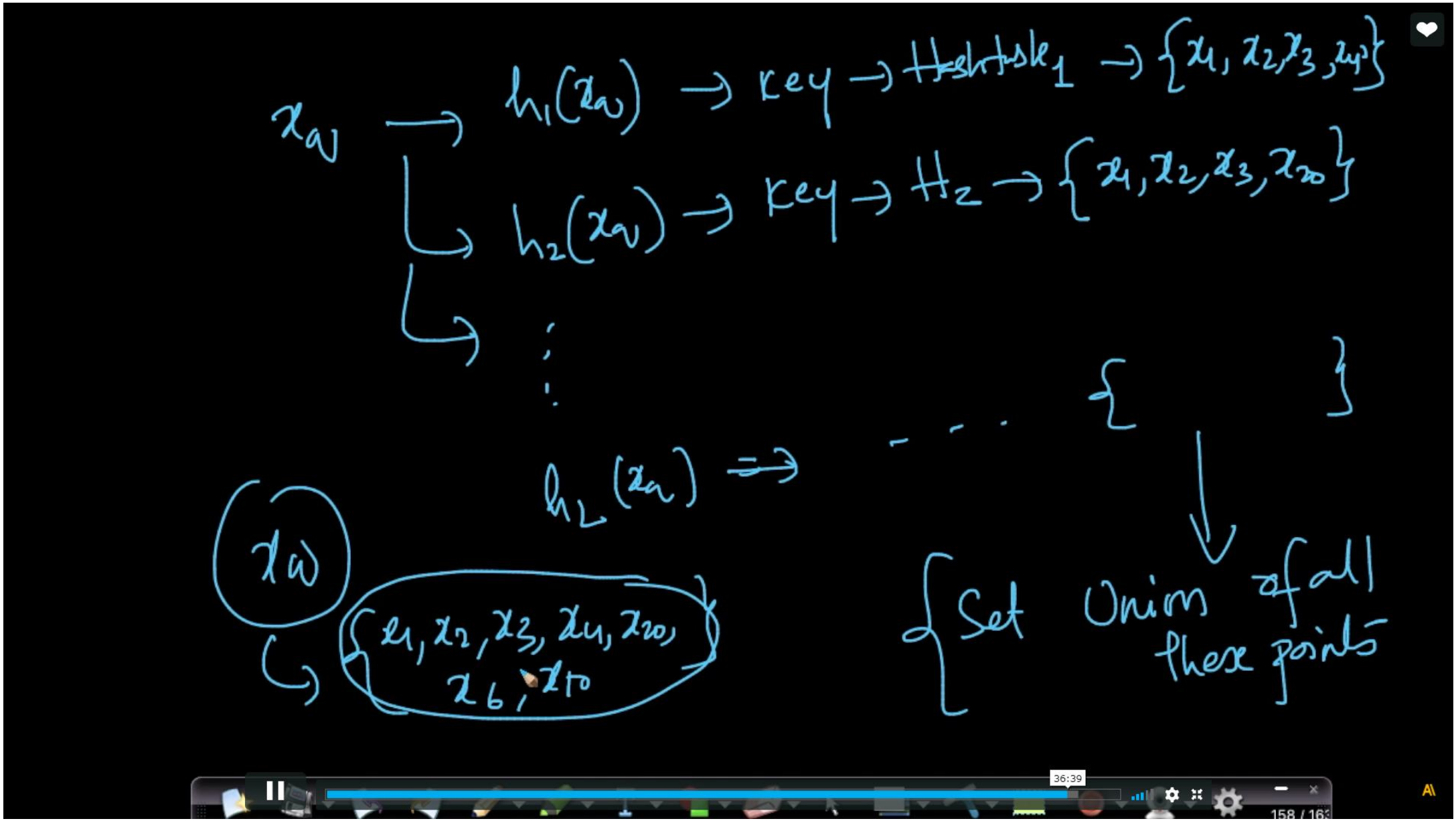


GOS-SIM

$\{x_{20}$ is very close to x_q

(*) could miss a NN in GOS-SIM as your measure


1
$h_1(x) =$
1 2 . . . . . m
hashtables
m-hyperplanes
2
$h_2(x) =$
1 2 . . . . . m
not same
hashtables
m-hyperplanes
L - hashtables


x_q
$h_1(x_q)$ → key → Hashtable$_1$ → $\{x_1, x_2, x_3, x_4\}$
$h_2(x_q)$ → key → H_2 → $\{x_1, x_2, x_3, x_{20}\}$
$h_L(x_q)$ ⇒
{ }
x_q
$\{x_1, x_2, x_3, x_4, x_{20}, x_6, x_{10}\}$
{Set Union of all these points
36:39
158 / 163

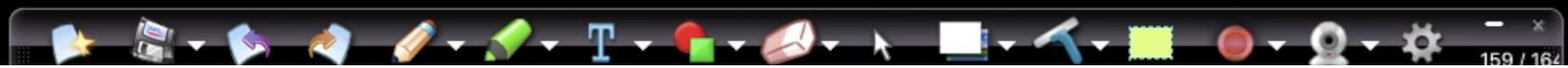
Time Complx to query given L hashtables

$m = O(\log n)$

$O(mdL)$

L is small

{ as number of hyperplanes ↑ ↓
slices ↑ ↓
pts per slice ↓ ↑ }



Riddhi Dutta

I summarised the steps in short. Pls review it

Steps for locality sensitive hashing:

1.First make 'm' hyperplanes to split into regions and create slices such that cluster of points lie in a particular slice and be called their neighbourhood.typically m = log(n)
2.Next for each point create a vector(also called hash function) by W1(transpose).point. if it is greater than 0 , it lies on the same side of that hyperplane else other side. Based on that create a vector of m size. For eg the vector can be [1,-1,-1] denoting point x lies on same side of normal to hyperplane 1, opposite side to normal of hyperplane 2 and 3. Now this vector serves as key to the hash table and all the points with the same key or vector representation will go in the same bucket as they have similar vector representation denoting they lie in the neighbourhood of each other.

3.Now it may happen that two points which are very close fall on different slice due to placing of hyperplane and hence not considered as nearest neighbour. To resolve this problem, create l hash tables. (l is typically small). In other words repeat step 2 l times thus creating l hash tables and m random planes l times. So when a query point comes compute the hash function each of the 'l' times and get its neighbours from each of bucket. Union them and find the nearest neighbours from the list. So basically in each 'l' iterations create m hyperplanes and hence region splitting will be different thus vector representation or hash function of the same query point will be different in each of the representations. Thus the hash table will be different as points which lied on the same region in previous iteration might lie in a different region in this iteration and vice versa due to different placement of hyperplanes.
Time complexity is O(m*d*l) for each query point. And for creating the hash table O(m*d*l*n) which is one time only.
Space complexity is O(n)

Reply

Jan 27, 2019 16:25 PM