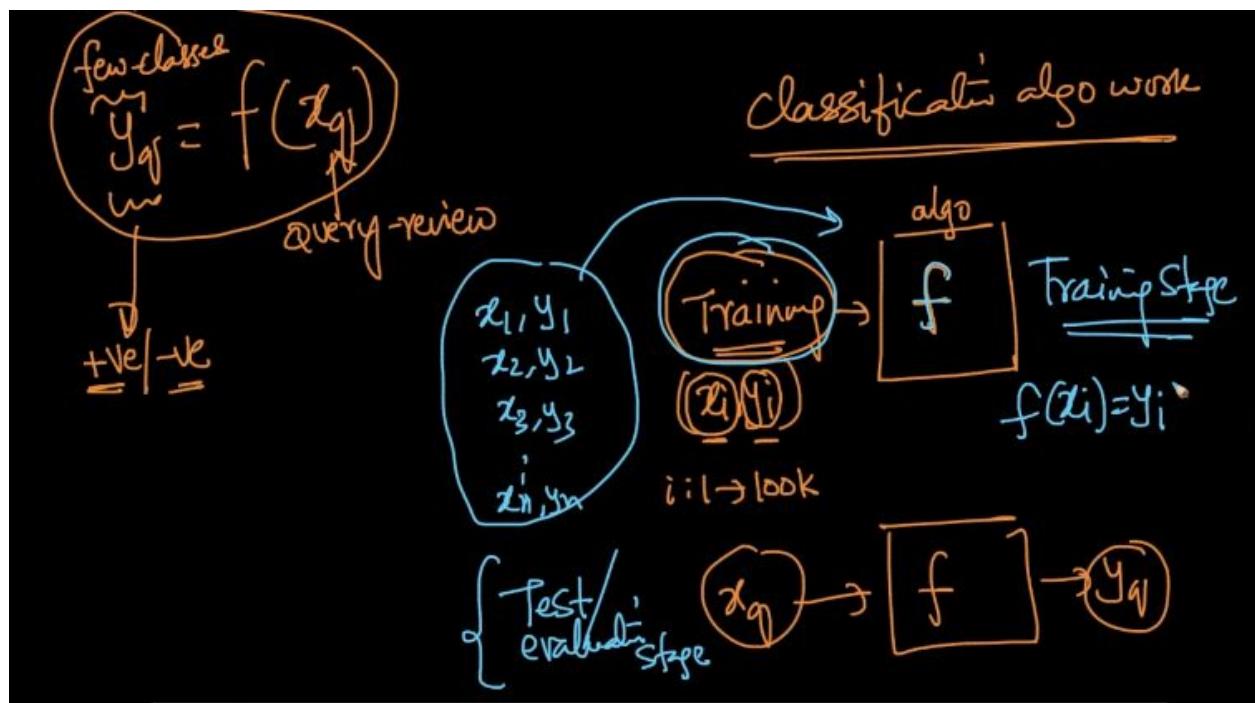
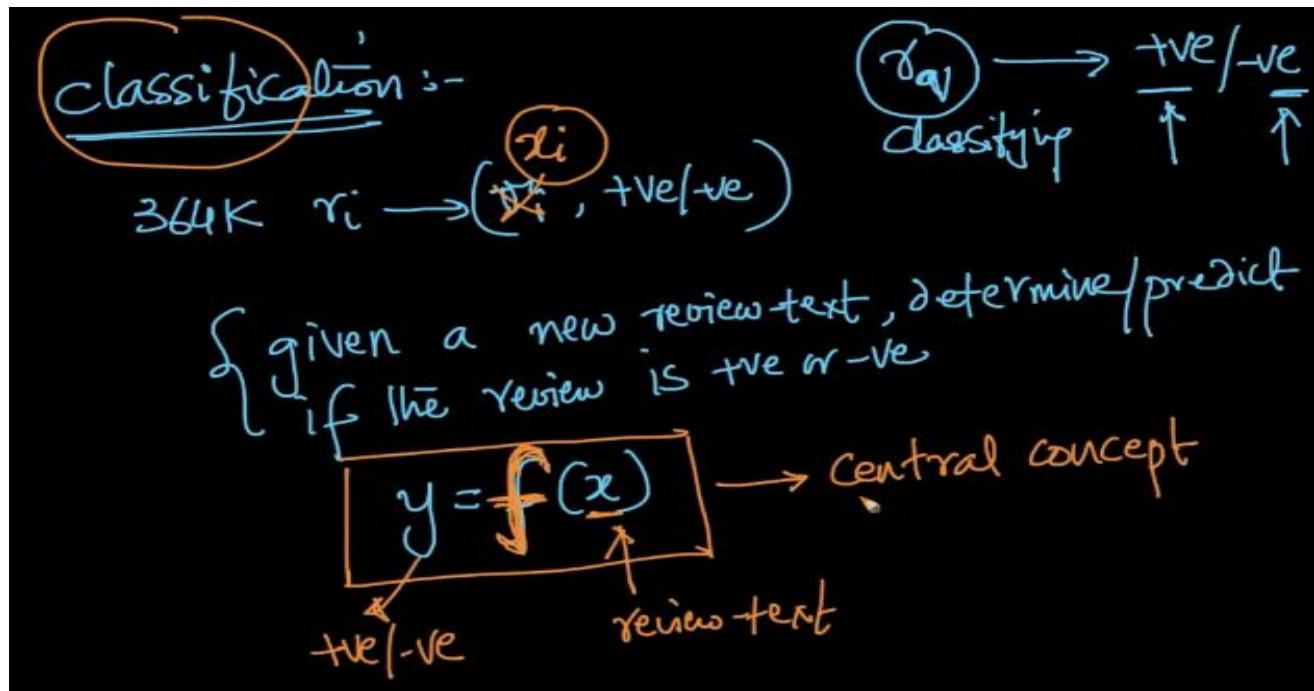


Module 3: Foundations of NLP and ML

Classification and Regression : KNN

Objective of Classification :



if we give new data x_q , the algorithms should predict y_q such that it has trained from various other datasets.

Data Matrix Notation

2.2 Data Matrix Notation

$x_i \rightarrow$ Text \rightarrow vector
 $y_i \rightarrow$ Class (true/false)
 $y = f(x)$

x_i^T $n \times d$ $n \times 1$

y_i

$D = \{(x_i, y_i)\}_{i=1}^n$ such that
 $x_i \in \mathbb{R}^d$
 $y_i \in \{0, 1\}$
 -ve true

$D = \{(x_i, y_i)\}_{i=1}^n$ such that
 $x_i \in \mathbb{R}^d, y_i \in \{0, 1\}$
 -ve, true

English \Rightarrow D is a dataset of pairs of numbers x_i and y_i .
 They have n such no such that
 x_i has d dimensional vector
 y_i could be either 0 or 1
 $0 \rightarrow$ -ve
 $1 \rightarrow$ +ve

$x_i^{\text{vec}} \rightarrow x_i$
 $\hookrightarrow \{0, 1\}(y_i)$

Classification:
 $f(x_i) = y_i$

Learning the function

Classification VS Regression

(Q.3)

Classification VS Regression

$$D = \{ (x_i, y_i) \}_{i=1}^n \mid x_i \in \mathbb{R}^d, y_i \in \{0, 1\}^3 \}$$

$y_i \in \{0, 1\}^3$ ↗
 ↙ re true ↗ 2 class Amazon food Reviews
 ↗ 2 class classification | Binary

MNIST dataset:

$$y_i \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\} \rightarrow 10 \text{ class classification}$$

what if,

$$y_i \in \mathbb{R} \xrightarrow{\text{Real no}}$$

y_i is no more part of a small finite set of classes.

x_i : <weight, age, gender, race>
 1: 1 to 10K

y_i : height

$$\boxed{y_i = f(x_i)} \rightarrow \begin{matrix} \text{Regression} \\ \hookdownarrow \text{Real no} \end{matrix}$$

problem

real no: 182.6 cm, 153.7 cm

$y_i \in \{0, 1\}$ - 'Classification'

$y_i \in \mathbb{R}$ - 'Regression'

K-Nearest Neighbours Geometric intuition with a toy example

K-Nearest Neighbors: (Geometric)

2D-toy dataset:



$x_{qj} \rightarrow y_{qj}$ (binary classification)

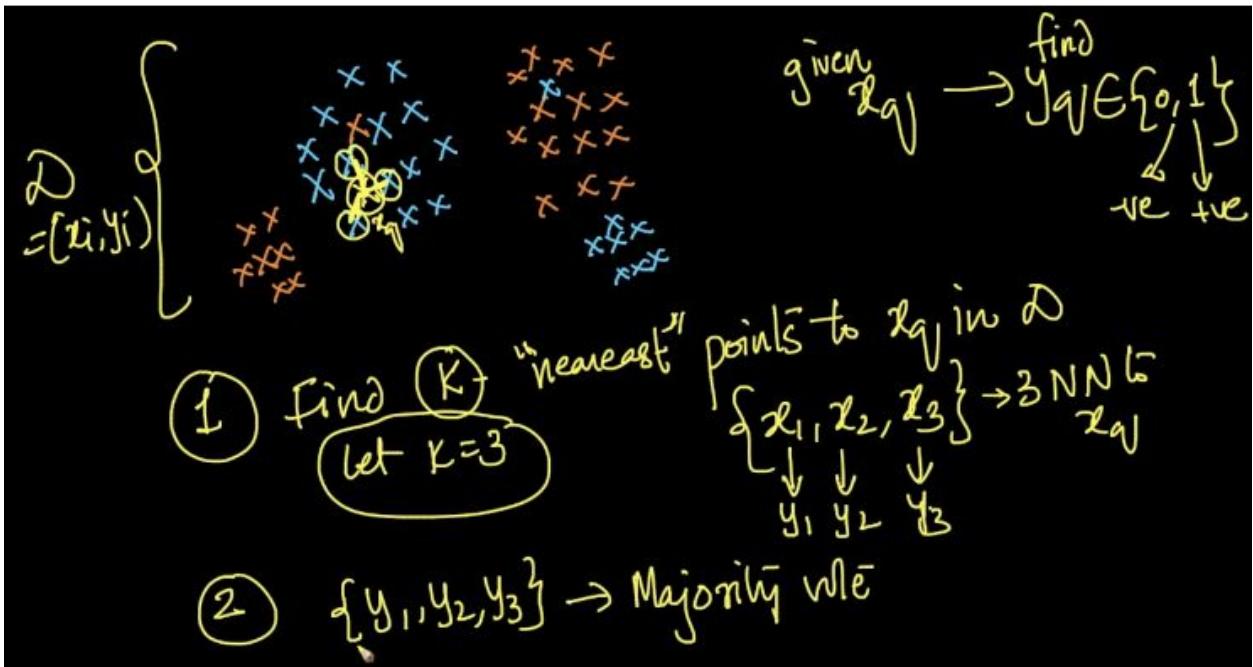
$\left\{ \begin{array}{l} x: +ve \text{ datapoint} \\ x: -ve \text{ class datapoint} \end{array} \right.$

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

geom.: x_{qj} close to x_{qj} :
 conclude $x_{qj} \rightarrow \text{blue (+ve)}$

$$D \rightarrow \boxed{ML}$$

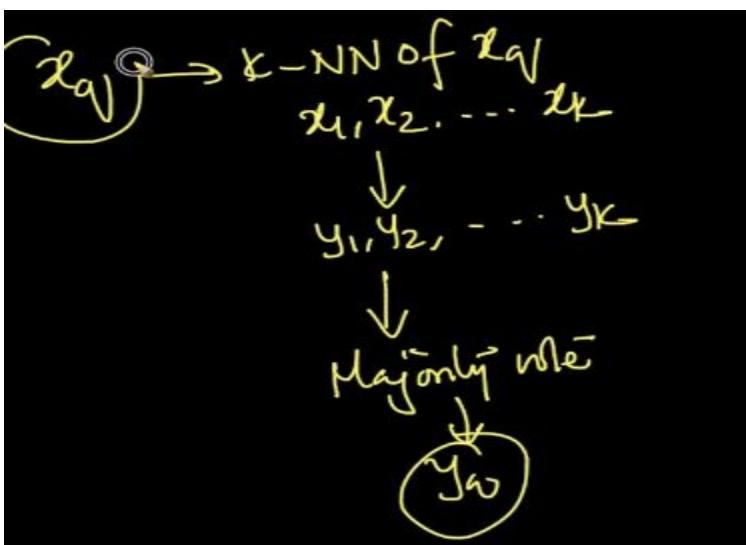
$$x_{qj} \rightarrow \boxed{f} \leftarrow \begin{matrix} +ve \\ -ve \end{matrix}$$



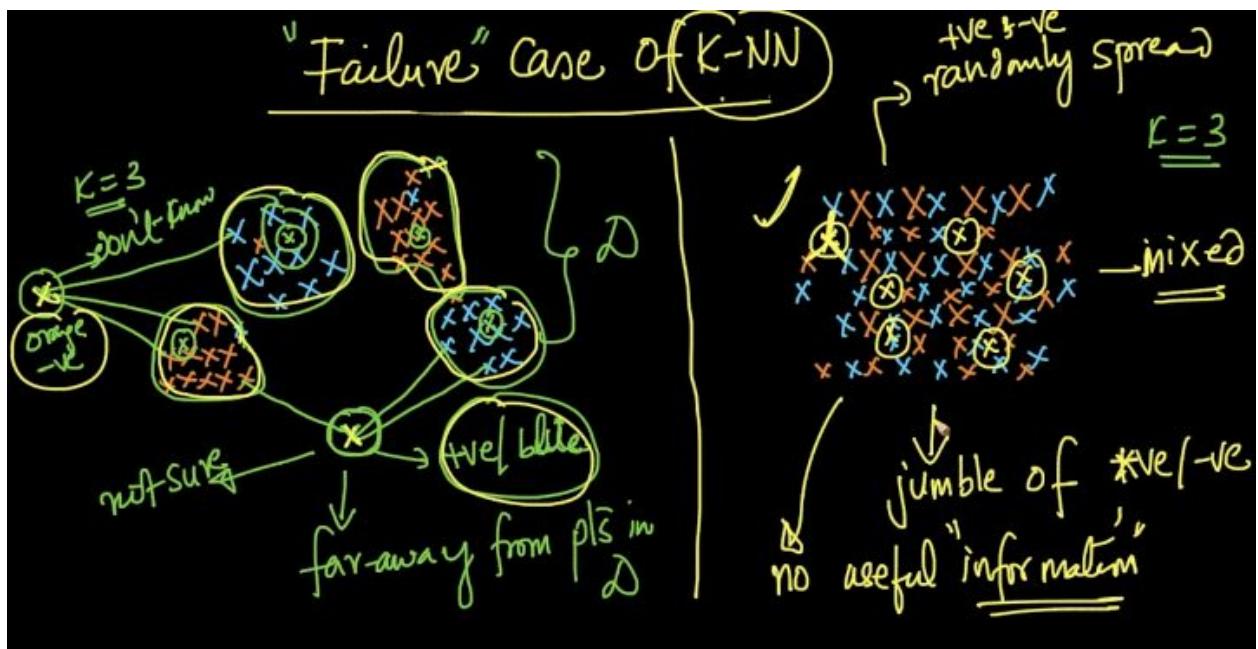
~~$K=\text{odd}$~~ $\{y_1, y_2, y_3\}$ $\rightarrow \text{Majority vote}$

$+, +, +$	$\rightarrow \oplus$	$\rightarrow y_q = \text{+ve (or) blue}$
$+, +, -$	$\rightarrow \oplus$	$y_q = \text{+ve (or) blue}$

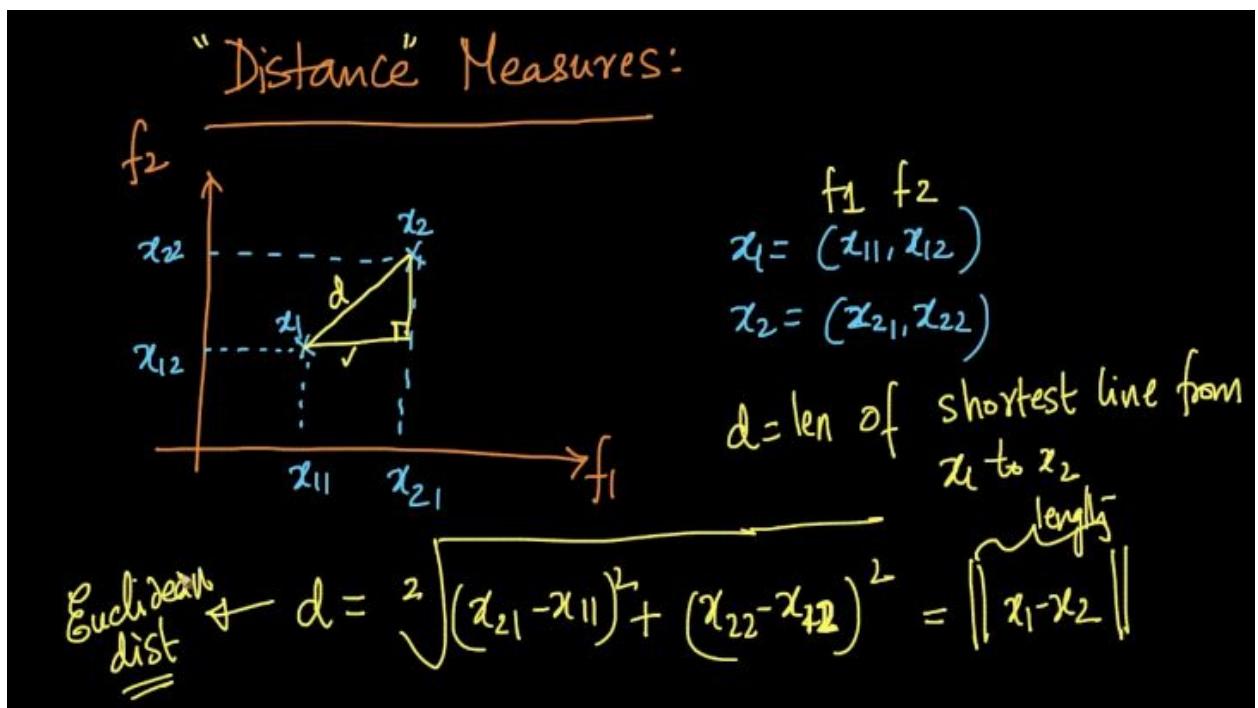
(X) if $K=4$ $+, +, -, - \rightarrow \oplus \text{ or } \ominus$



Failure Cases of KNN



Distance measures: Euclidean(L2) , Manhattan(L1), Minkowski, Hamming



$x_1 \in \mathbb{R}^d, x_2 \in \mathbb{R}^d$

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

$\|(\vec{x}_1 - \vec{x}_2)\|_2 \rightarrow L_2 \text{ norm}$

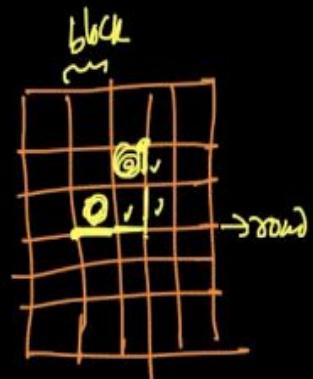
$$\|x_1\|_2 = \text{dist of } x_1 \text{ from origin} = \left(\sum_{i=1}^d x_{1i}^2 \right)^{1/2}$$

\checkmark Manhattan dist:

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

$$\text{L}_1 \text{-norm of vector } (x_1 - x_2) \quad \|x_1 - x_2\|_1$$

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



L_p -norms $\xrightarrow{\quad}$ Minkowski dist

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

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

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

$p=1 \rightarrow$ " \rightarrow Manhattan dist

Hamming dist (boolean Vectr)

$x_1, x_2 \rightarrow$ boolean Vectr \rightarrow Binary Bow

$x_1 = [0, 1, 1, 0, 1, 0, 0, 1] \rightarrow 3$

$x_2 = [1, 0, 1, 1, 0, 1, 0, 1] \rightarrow 3$

Hamming-dist $(x_1, x_2) = \#$ locations where binary vectors differ

$\hookrightarrow 3$

strings:

$x_1 = \text{apcadefghik} \leftarrow \text{Gene code / Seq}$

$x_2 = \text{acb adegjhi} \qquad \qquad \qquad \text{AGTC}$

hamming dist $(x_1, x_2) = 4$

$\left\{ \begin{array}{l} x_1 = \text{A}(\text{A})\text{GTC TCA}(\text{G}) \\ x_2 = \text{A}(\text{G})\text{A TCA}(\text{C})\text{A} \end{array} \right.$

Cosine Distance & Cosine Similarity

Cosine-Similarity vs Cosine-distance:

x_1, x_2

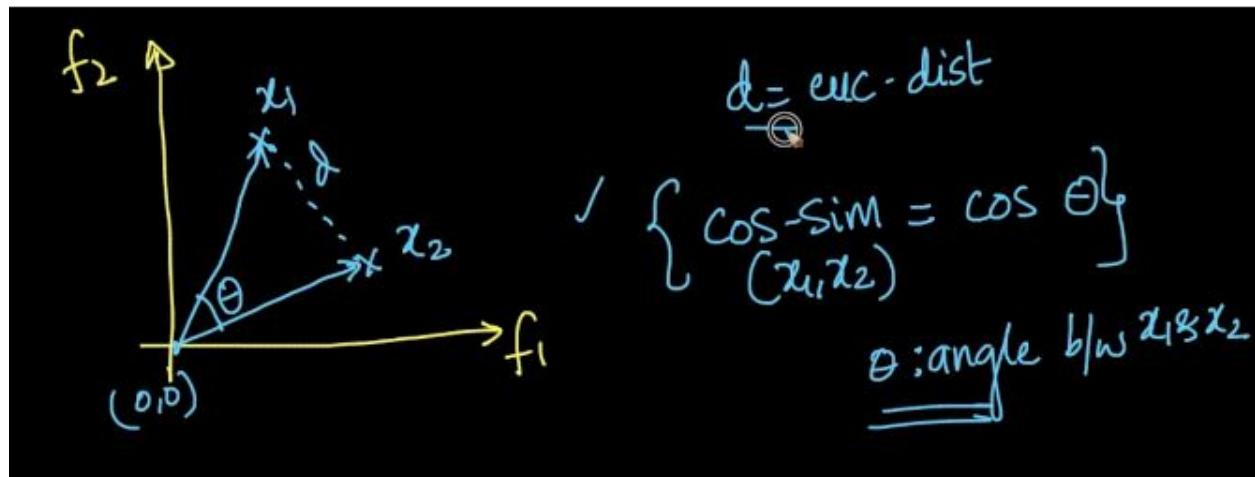
\downarrow Similarity	\uparrow distance	\uparrow inc \downarrow dec (opposite)
-----------------------------------	-------------------------------	---

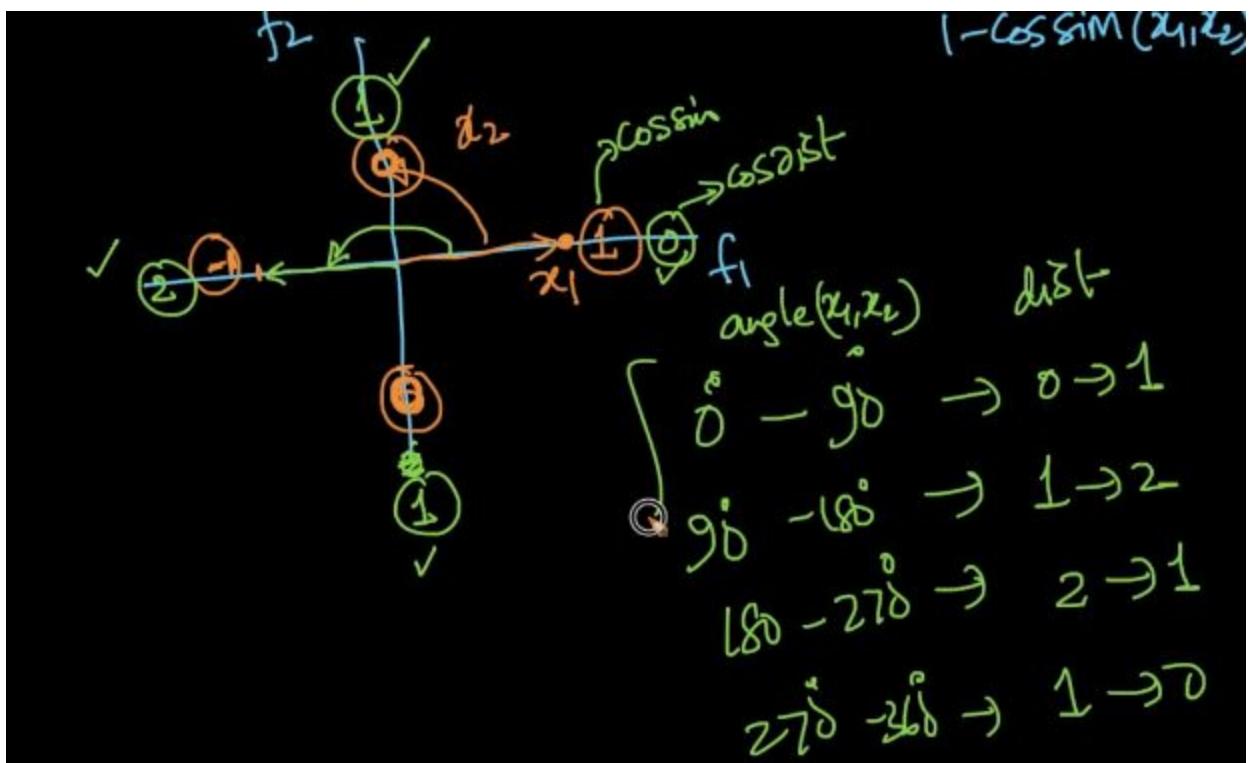
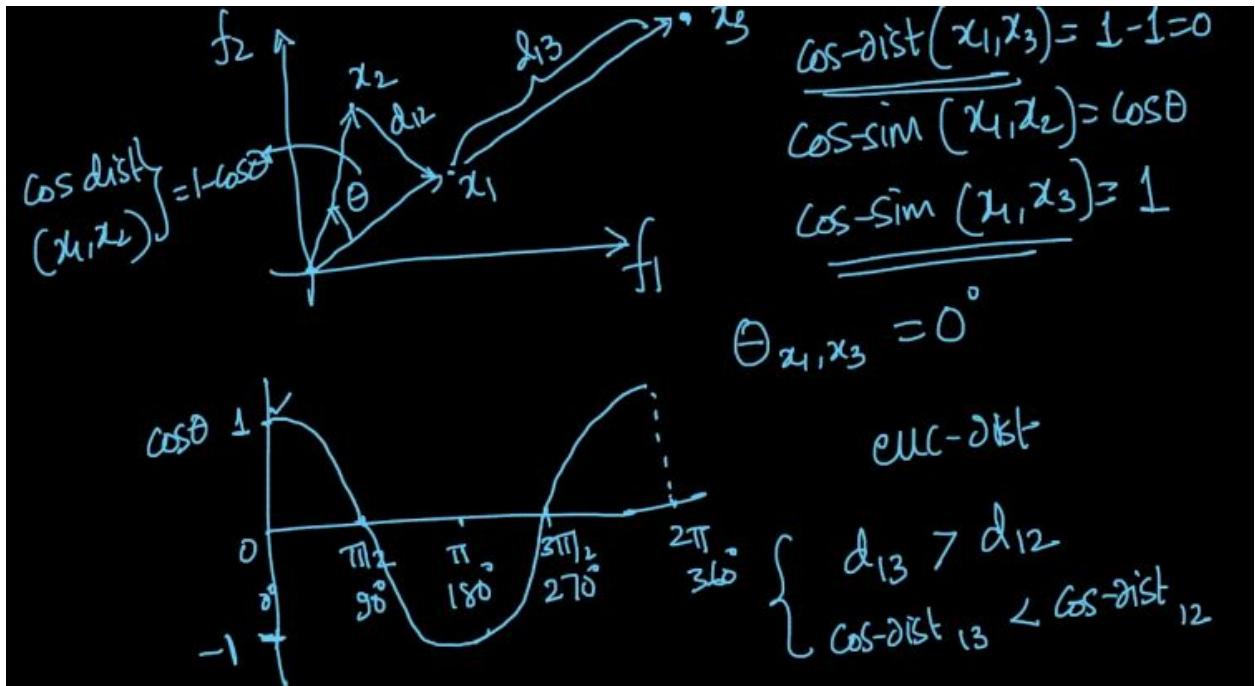
Let $\cos\text{-sim}(x_1, x_2) = +1$
 v. similar

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

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

Let $[-1, 1]$





x_1, x_2

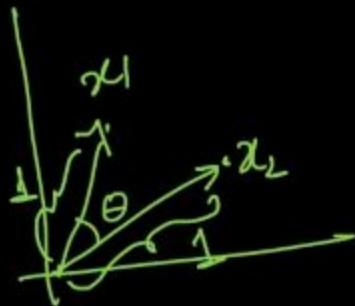
$$\cos(\theta) = \frac{x_1 \cdot x_2}{\|x_1\|_2 \|x_2\|_2}$$

$\underbrace{\quad}_{L_2 \text{ norm of } x_1}$

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)

$$\theta = \text{angle b/w}$$

 x_1 & x_2

- if x_1 & x_2 are unit vect

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

$$\Rightarrow \boxed{\sqrt{2 \cos\text{-dist}(x_1, x_2)}}$$

Link:

<https://cmry.github.io/notes/euclidean-v-cosine>

How to measure the effectiveness of k-NN?

Q. 5 How to measure the effectiveness of k-NN?

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

$D = \begin{bmatrix} 1 \\ 2 \\ 3 \\ \vdots \\ i \\ \vdots \\ n \end{bmatrix} \left[\begin{array}{c|c} x_i^T & | & y_i \end{array} \right]$
 $\underbrace{\quad}_{364K}$
 ReviewText
 \rightarrow Bag/TF-idf/w2v

Problem: Given a new food review x_q what is polarity (true/false)?
 $x_q \rightarrow \text{Text}_q \rightarrow \underbrace{y_q}_{\text{Bag/TF-idf-w2v}}$

"Measure" k-NN $\rightarrow \begin{pmatrix} \text{k-NN + Majority vote} \\ x_i^T \\ y_i^T \end{pmatrix}$

$D = \begin{bmatrix} 1 \\ 2 \\ 3 \\ \vdots \\ i \\ \vdots \\ n \end{bmatrix} \left[\begin{array}{c|c} x_i^T & | & y_i^T \end{array} \right] \left\{ \begin{array}{l} D_{\text{Train}} \\ D_{\text{Test}} \end{array} \right.$
 $\underbrace{\quad}_{364K}$

D_n $\xrightarrow{\text{Split}} D_{\text{Train}}$ $\xrightarrow{\text{Split}} D_{\text{Test}}$

$D_{\text{Train}} \cup D_{\text{Test}} = D_n$
$D_{\text{Train}} \cap D_{\text{Test}} = \emptyset$

$D_{\text{Train}} \rightarrow n_1$ $n_1 + n_2 = n$

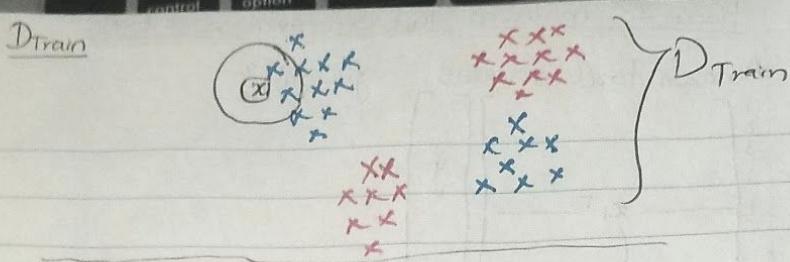
$D_{\text{Test}} \rightarrow n_2$

2 Randomly $n_1 = 0.7n$
 $n_2 = 0.3n$

D_n $\xrightarrow{\text{Split}} D_{\text{Train}}$ $\xrightarrow{\text{Split}} D_{\text{Test}}$

n_1 $D_{\text{Train}} (x_i, y_i)_{i=1}^{n_1} \rightarrow \text{k-NN}$

n_2 $D_{\text{Test}} (x_i, y_i)_{i=1}^{n_2} \rightarrow \text{k-NN}$



Each point x_i in D_{Test} $\Rightarrow \{ (x_i, y_i) \}_{i=1}^n \}$

$$x_q = (x)$$

for each pt in D_{Test}

$$y_q = \text{Blue} = (\text{blue})$$

$$\rightarrow x_q = p_t$$

\rightarrow Use D_{Train} & KNN to predict y_2

If $y_q = y_{pt}$

$$\text{Cnt} = \text{Cnt} + 1$$

$\text{Cnt} = \#$ of point for which $D_{\text{Train}} + \text{KNN}$ gave a correct level

Accuracy = $\frac{\text{Cnt}}{n_2} \rightarrow \# \text{ of point for which } D_{\text{Train}} + \text{KNN} \text{ gave correct class level}$

$$n_2 \rightarrow \# \text{ of point in } D_{\text{Test}}$$

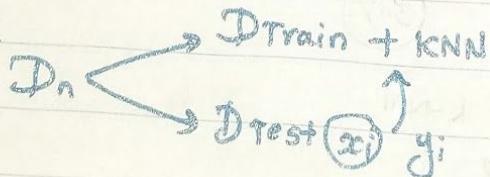
$$0 \leq \text{Acc} \leq 1$$

$\text{Acc} = 0.91 = 91\%$ of times KNN with D_{Train} correctly/predictively predicts y_q given x_q

Conclude: $\left. \begin{array}{l} \text{K-NN gives me an accuracy of } 91\% \\ \text{on Amazon food reviews using KNN} \end{array} \right\}$

91%

Summary:



Evaluation Time and Space Complexity

(2.9) Test / Evaluation time & space complexity

$x_q \rightarrow y_q$

Input: D_{Train} , K , $x_q \in \mathbb{R}^d$; output: y_q

K is small
 $\hookrightarrow 5 \text{ or } 10$

$KNN \text{ pts} = []$

for each x_i in D_{Train} : n points: d -dim $\rightarrow n \gg d$ could be large.

$O(d)$ - Compute $d(x_i, x_q) \rightarrow d_i$ $\rightarrow O(nd)$
 $O(1)$ - keep smallest k -dist $\rightarrow (x_i, g_i, d_i)$

$\left\{ \begin{array}{l} \text{Cnt-pos} = 0 \\ \text{Cnt-neg} = 0 \end{array} \right.$

for each x_i in KNN pts

if y_i is +ve,

$\left[\begin{array}{l} \text{Time complx: } O(nd) + O(d) + O(1) \\ = O(nd) \end{array} \right]$

$\text{Cnt-pos} += 1$

else:

$\text{cnt-neg} += 1$ if d is small

 if $\text{cnt-pos} > \text{cnt-neg}$ if $d \ll n$

 return $y_q = 1 \rightarrow$ +ve

$\boxed{O(n)}$ \rightarrow Time it takes

$O(1)$ else

$y_q = 0 \rightarrow$ -ve

Amazon food reviews

$\left\{ \begin{array}{l} n = 364K \\ d = 100K (\text{Bow}) \\ 300 (\text{w2v/tf-idf}) \end{array} \right.$

$x_q \rightarrow y_q$

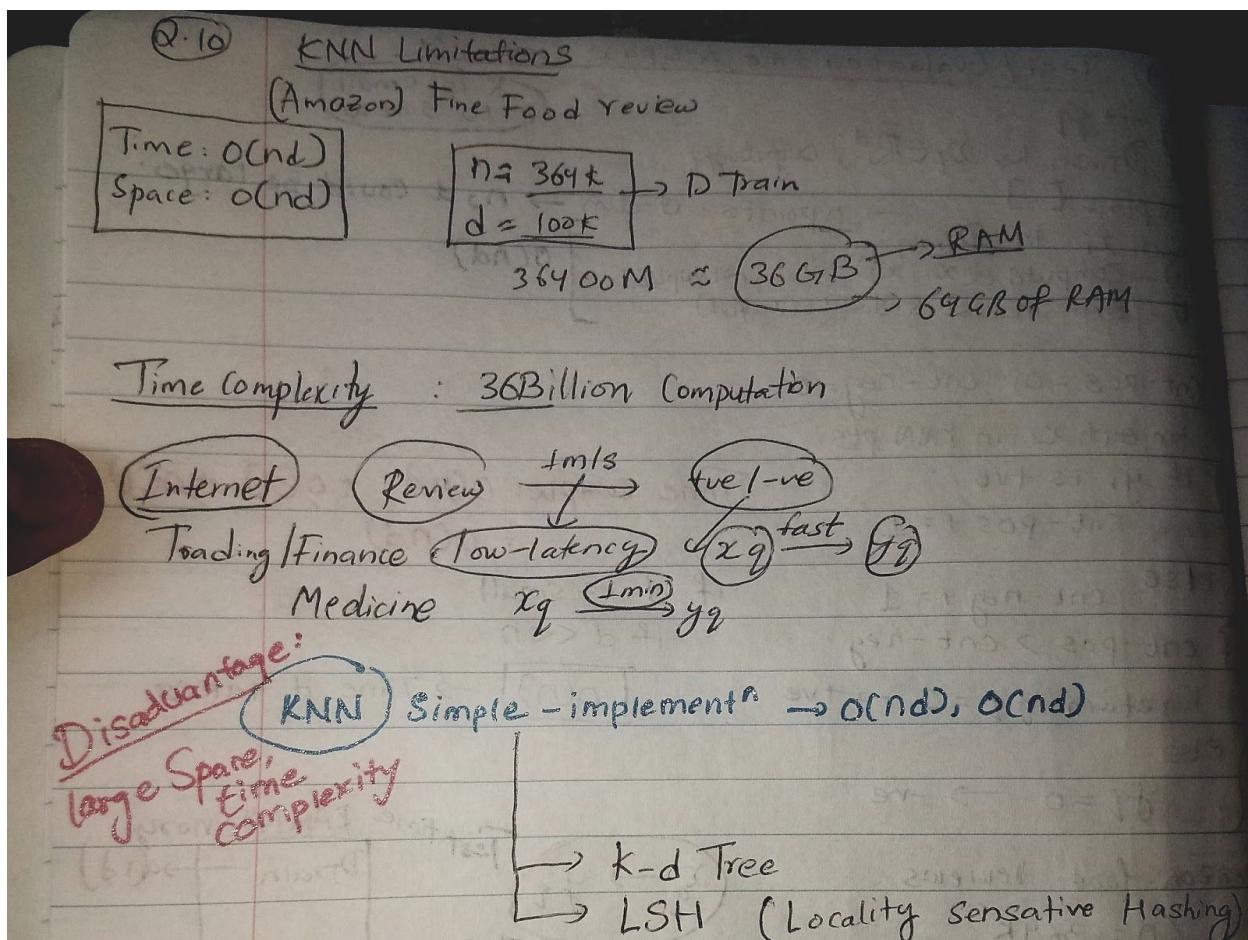
$\left[\begin{array}{l} \text{Test time RAM/Memory} \\ D_{Train} \rightarrow O(d) \end{array} \right]$

$\left[\begin{array}{l} \text{Evaluation Time} \\ x_q \rightarrow y_q \\ = O(nd) \end{array} \right]$

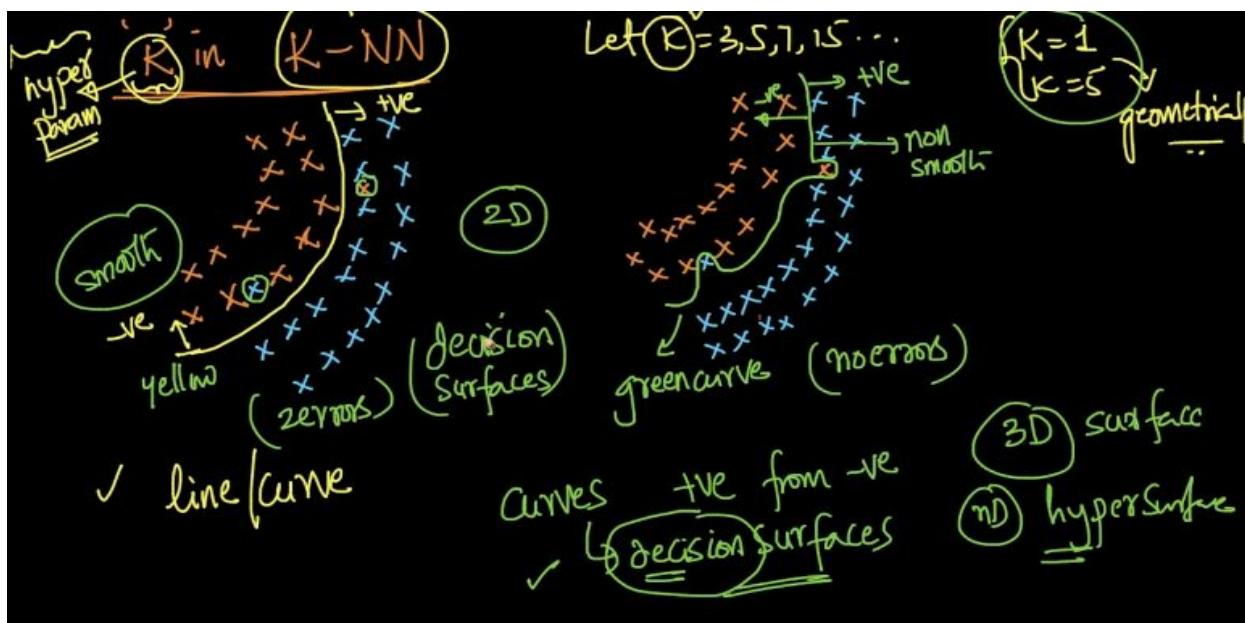
Time complexity $\div O(nd)$

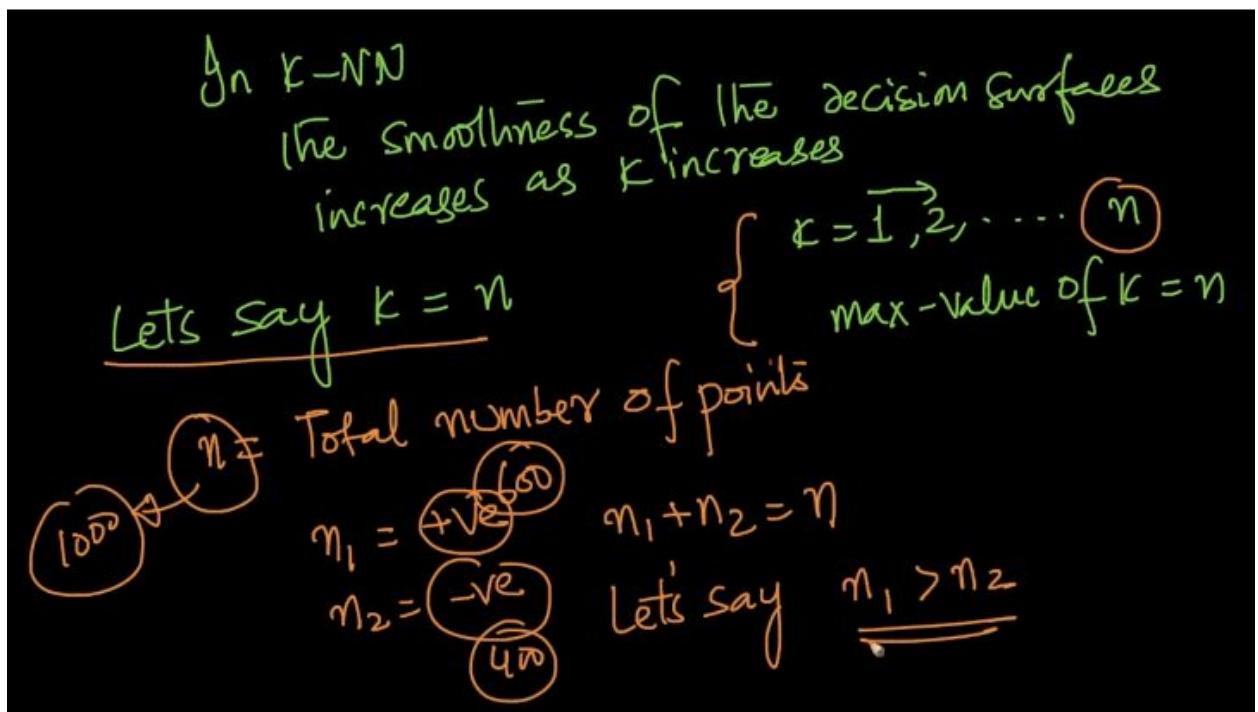
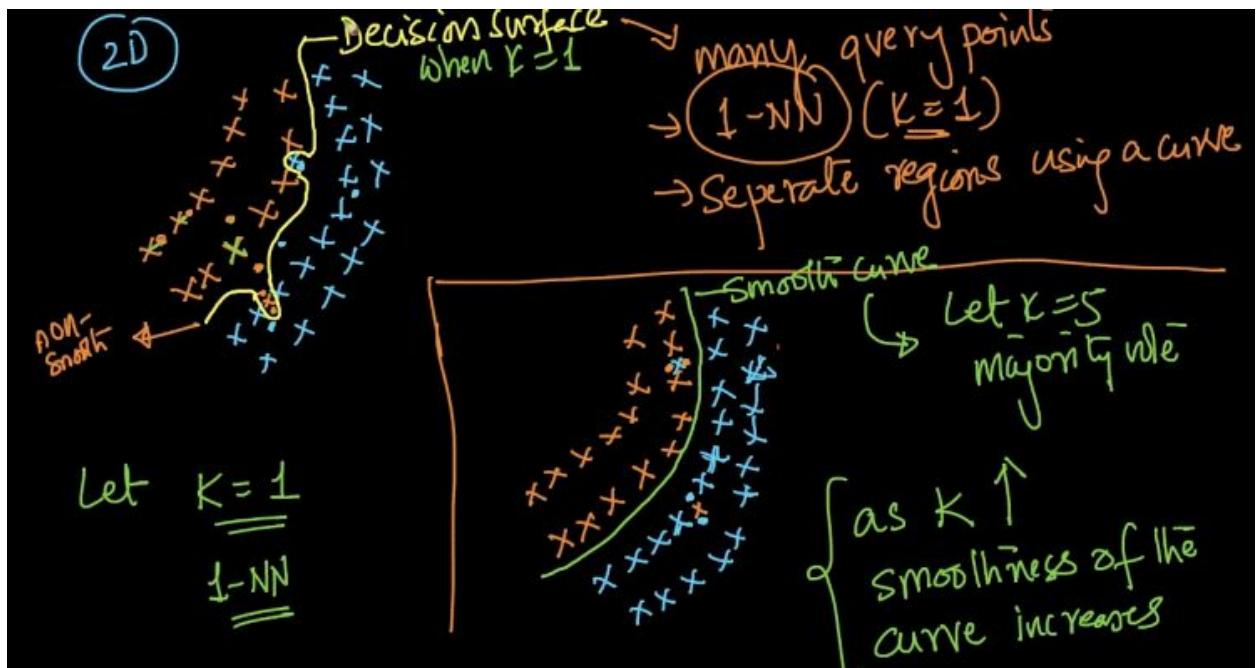
Space n = space that is needed to evaluate

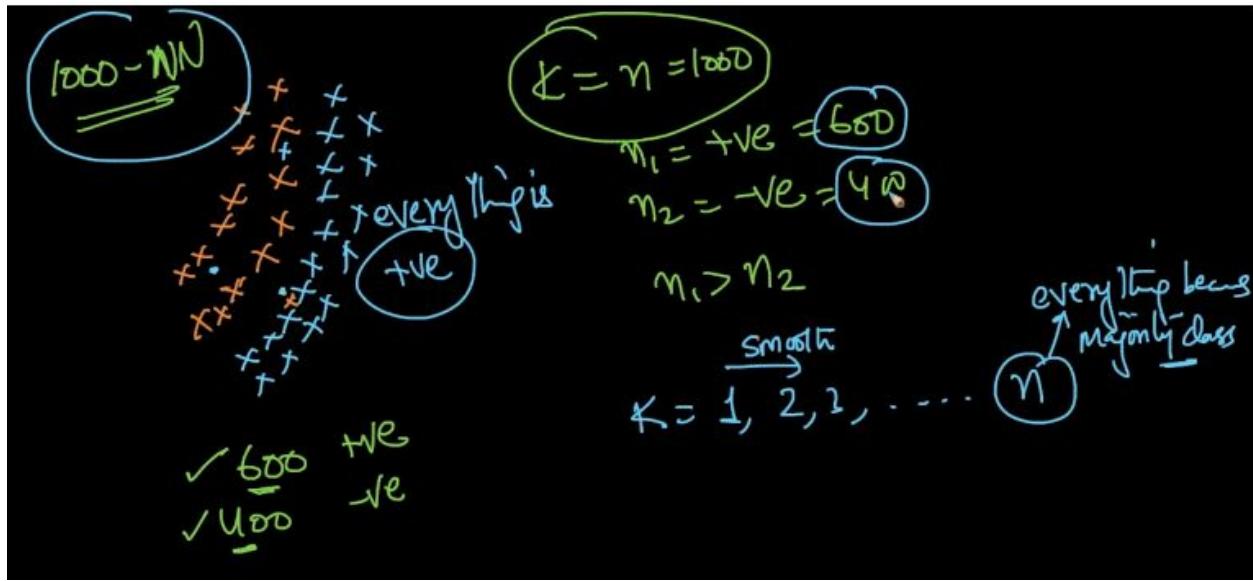
KNN Limitations



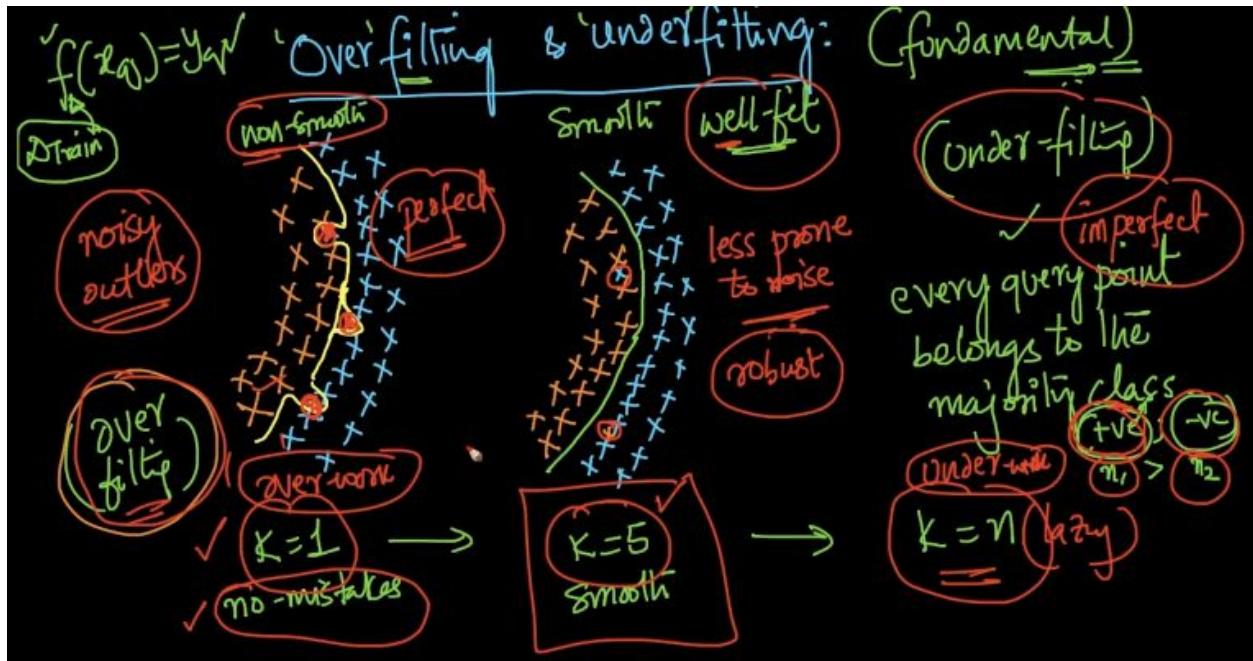
Decision surface for K-NN as K changes







Overfitting and Underfitting

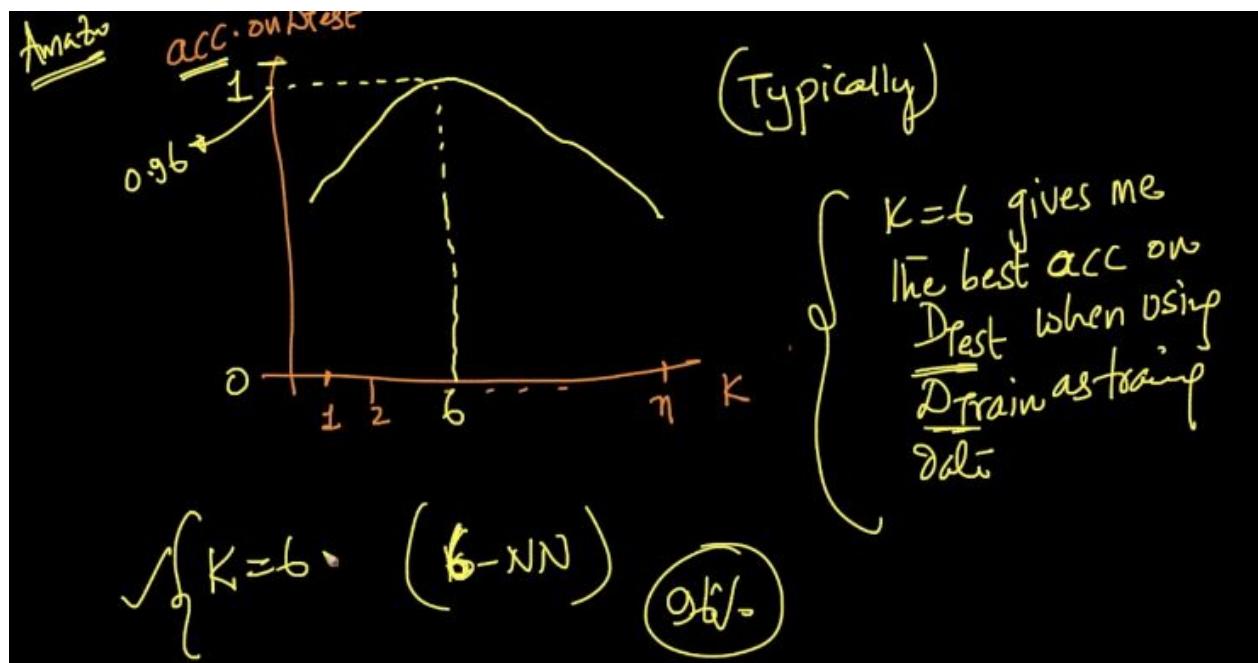


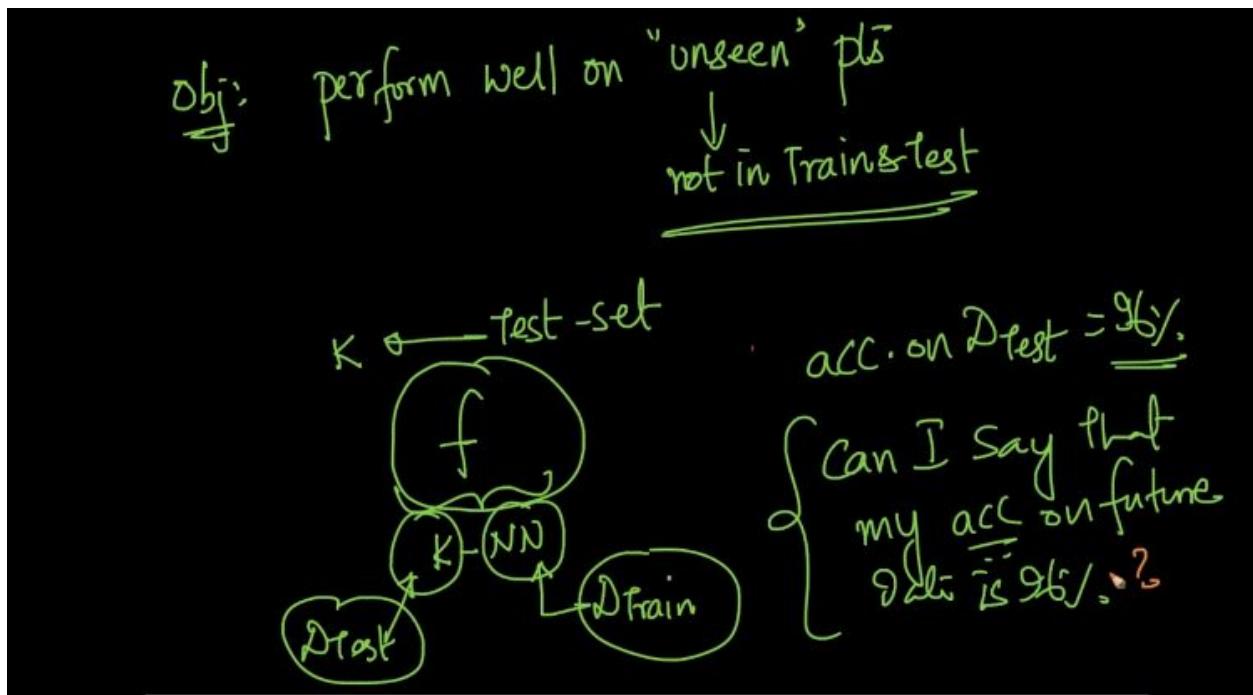
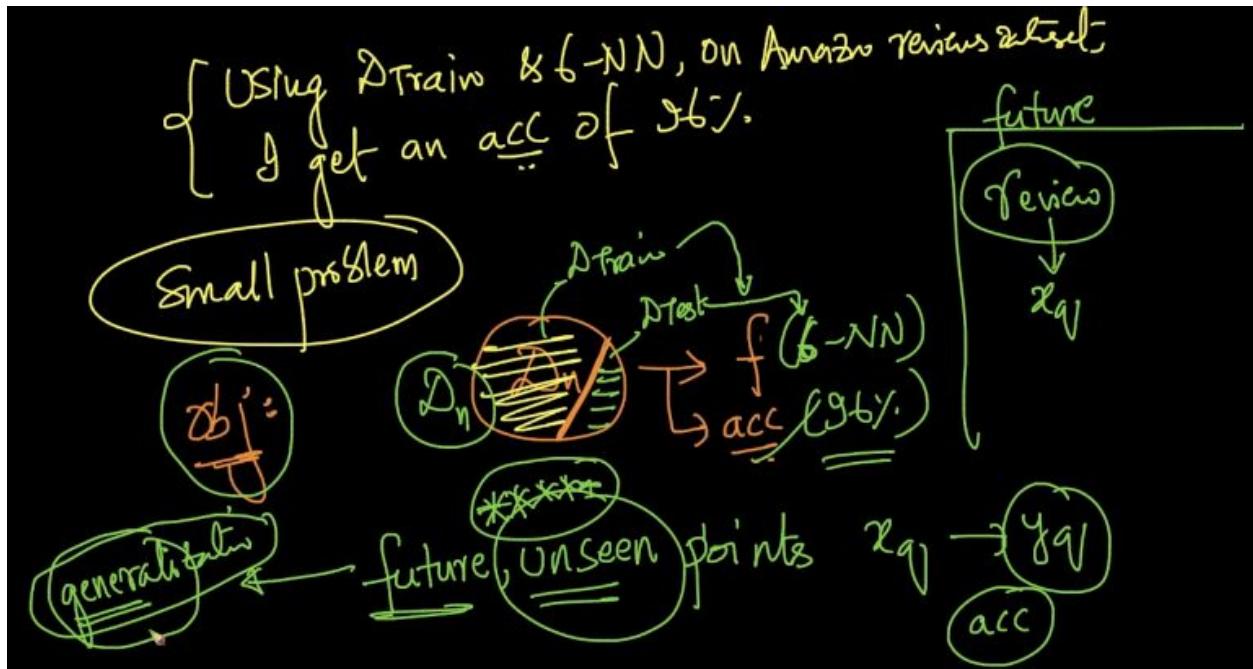
Need for Cross validation

$K = 1, 2, 3, \dots, n$ \leftarrow overfitting, underfitting
How to determine K

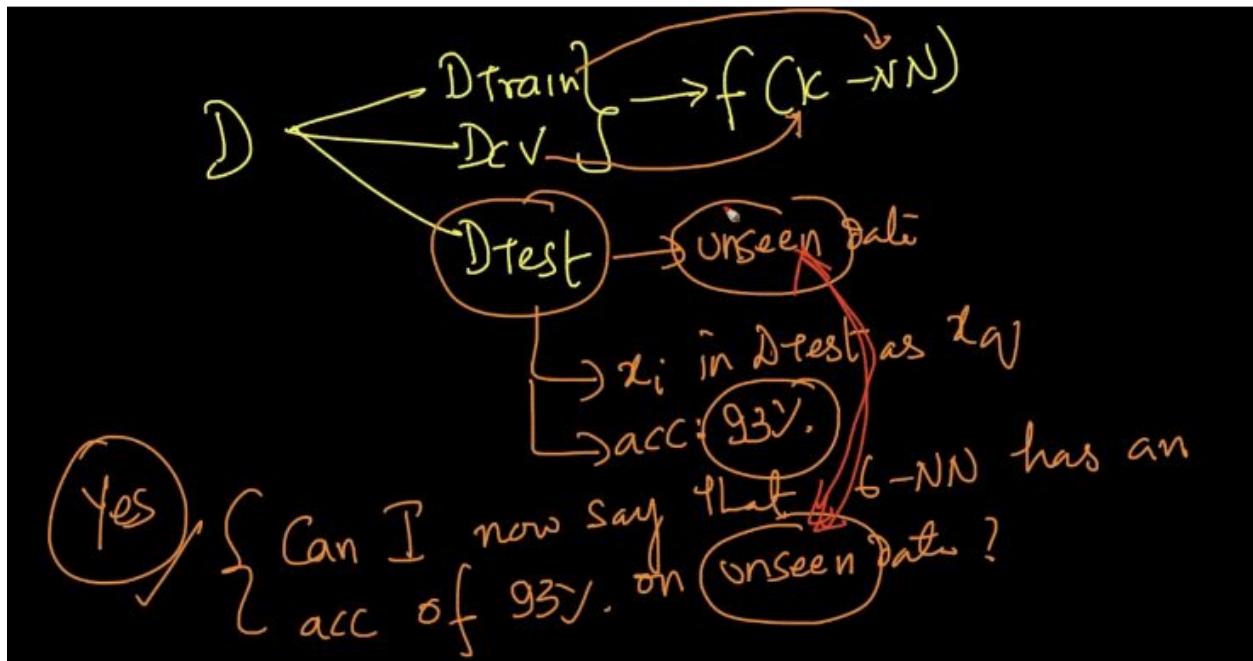
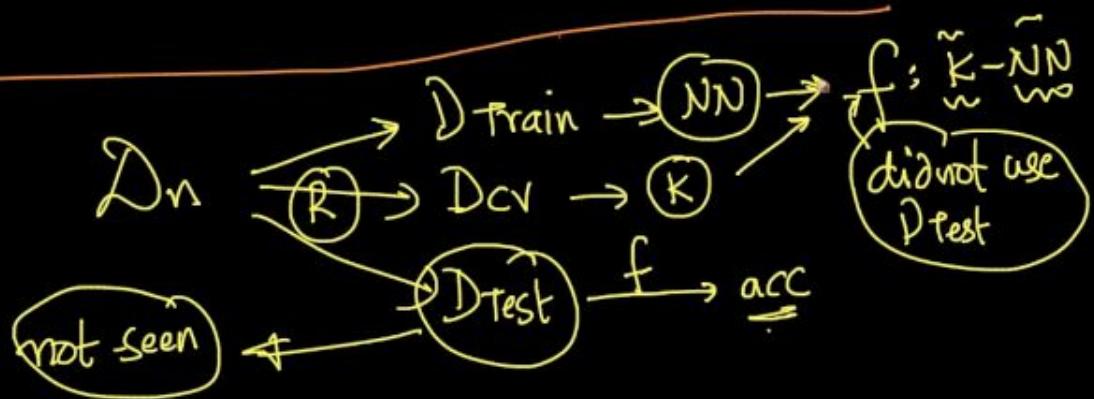
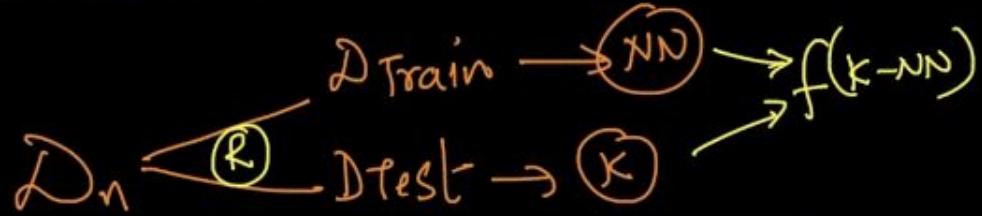
D_n	$D_{\text{train}} (70\%)$	$D_{\text{test}} (30\%)$	$\frac{\text{num. of correctly classified pls}}{\text{Total # pls}}$
	Train	(acc. on) Test	
$K=1$	D_{train}		0.78
$K=2$	"		0.82
$K=3$	"		0.85
:			

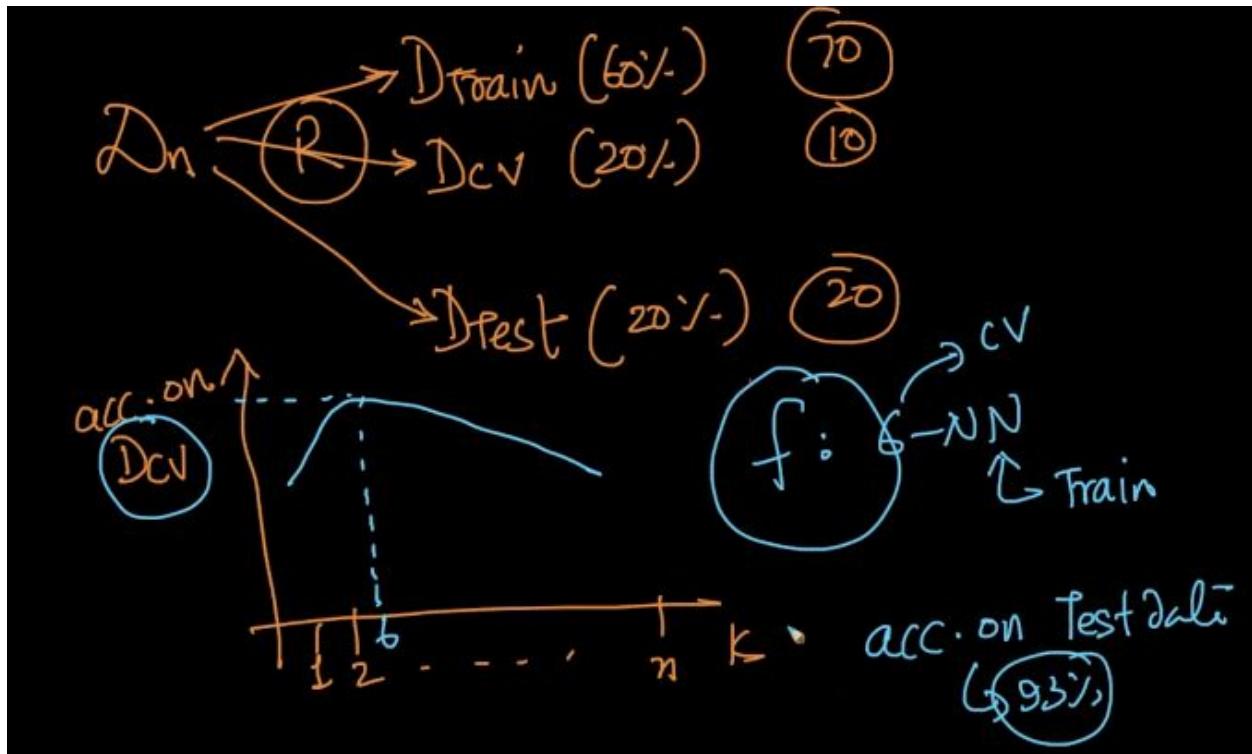
One idea:





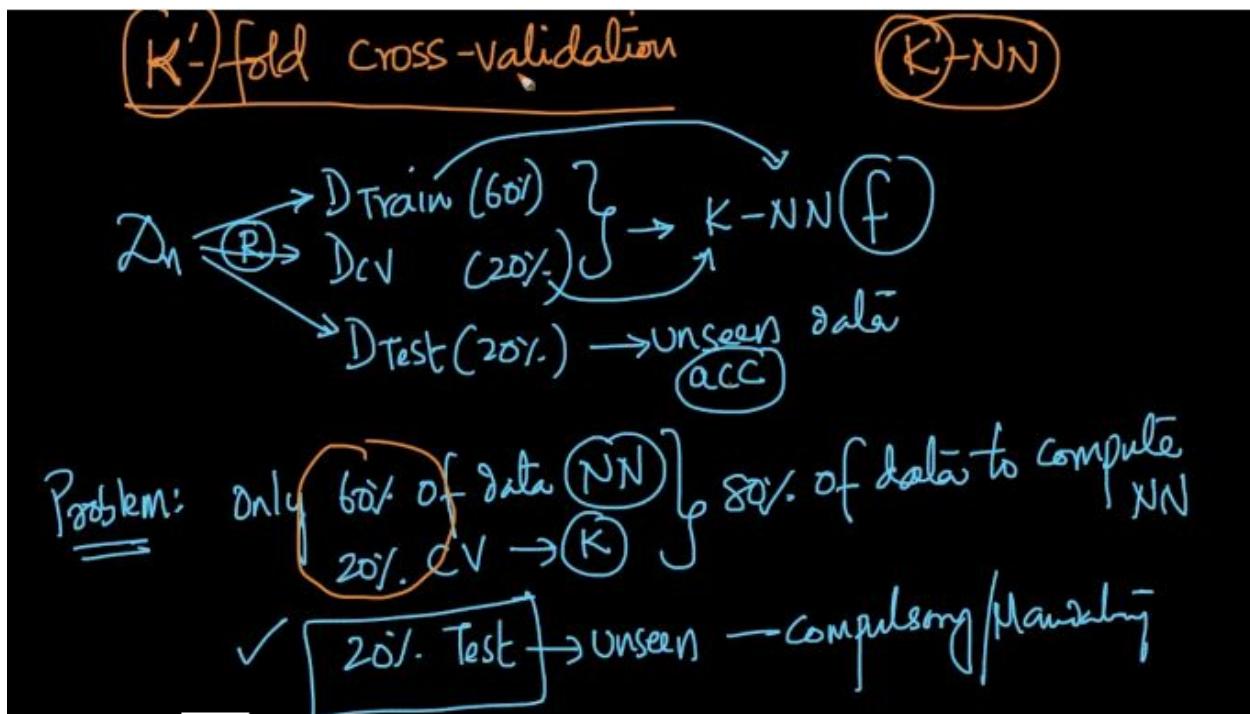
Cross-validation (CV)



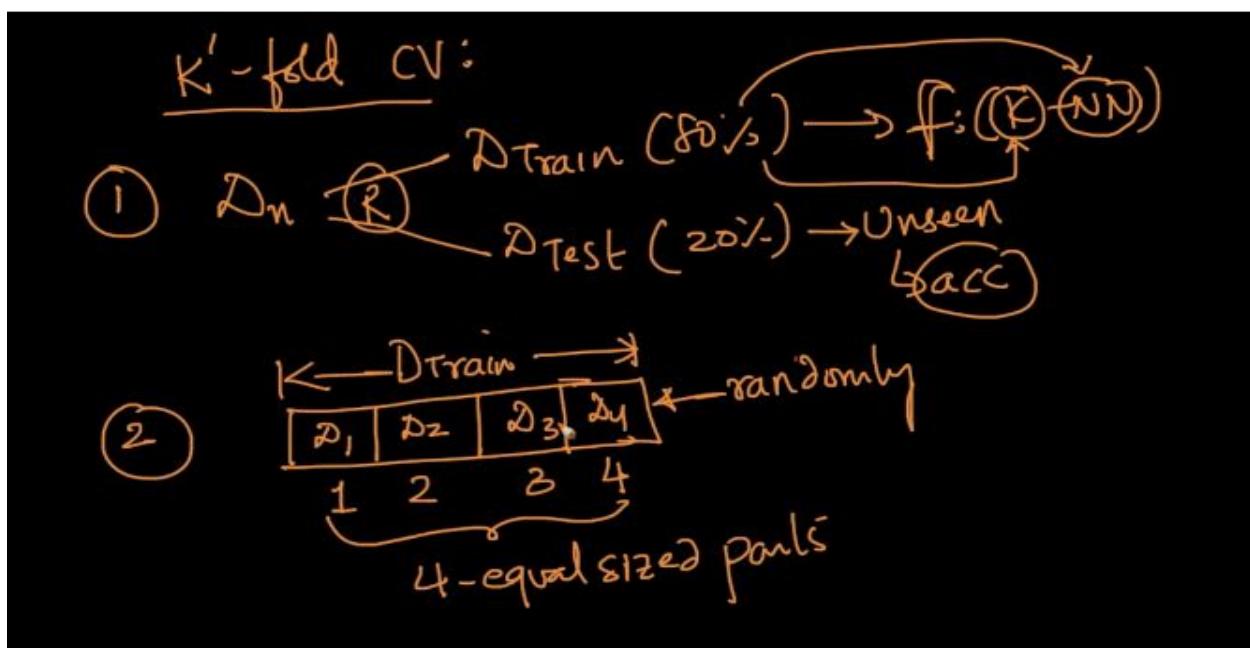


{ Using D_{train} as training set, I find 6-NN
 to have an acc of $\frac{93}{-}$ on future/unseen
 data
 { generalization accuracy
 $F.I. = \frac{1}{n} \sum \text{gen. error}$

K-fold cross-validation



More the training data, better is your algorithms.



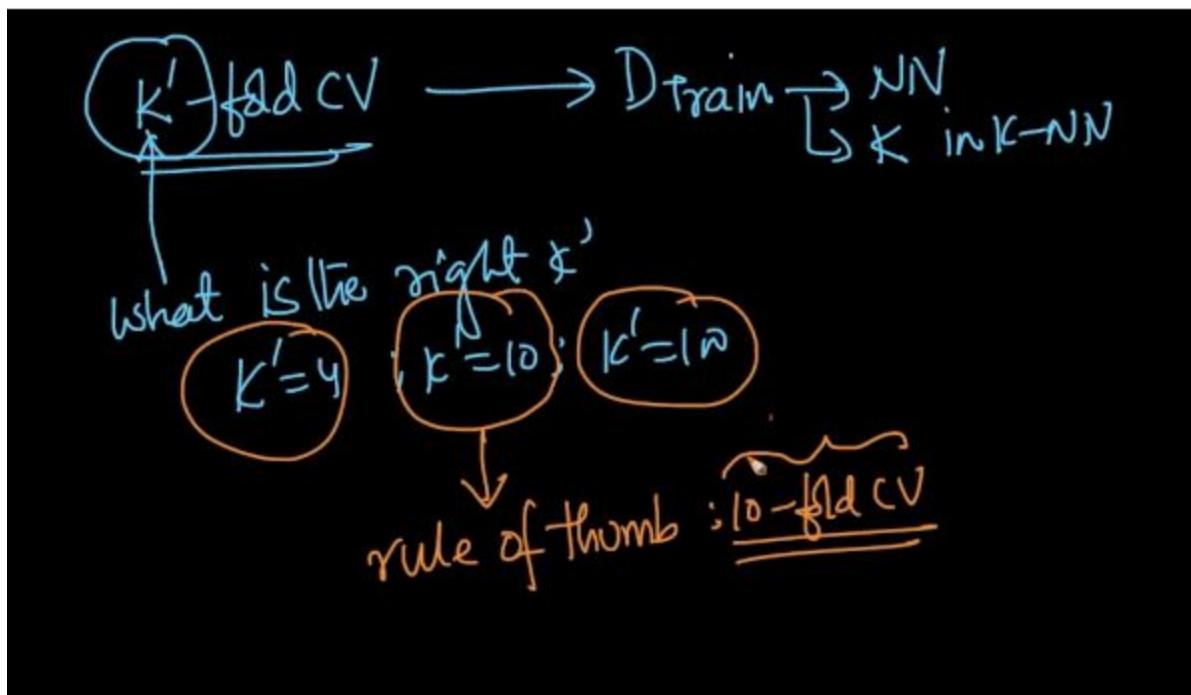
③

$K = 4\text{-fold CV}$

4-times

	Train	CV	acc on CV	acc on (CV) dataset
$K=1$	D_1, D_2, D_3	D_4	a_4	\uparrow
$K=1$	D_1, D_2, D_4	D_3	a_3	
$K=1$	D_1, D_3, D_4	D_2	a_2	
$K=1$	D_2, D_3, D_4	D_1	a_1	
			$\text{avg}(a_1, a_2, a_3, a_4)$	$\text{best } K$
			$a_{K=1}$	
$K=2$	D_1, D_2, D_3	D_4	a'_4	
$K=2$	D_1, D_2, D_4	D_3	a'_3	
$K=2$	D_1, D_3, D_4	D_2	a'_2	
$K=2$	D_2, D_3, D_4	D_1	a'_1	
			$\text{avg} \downarrow$	
			$a_{K=2}$	

acc on $K\text{-NN}$
for $K=1$ on
(CV)-dataset



Time it takes to compute the optimal /best k in K-NN increases by k' times if we use k' -fold cross validation.

K-cross validation

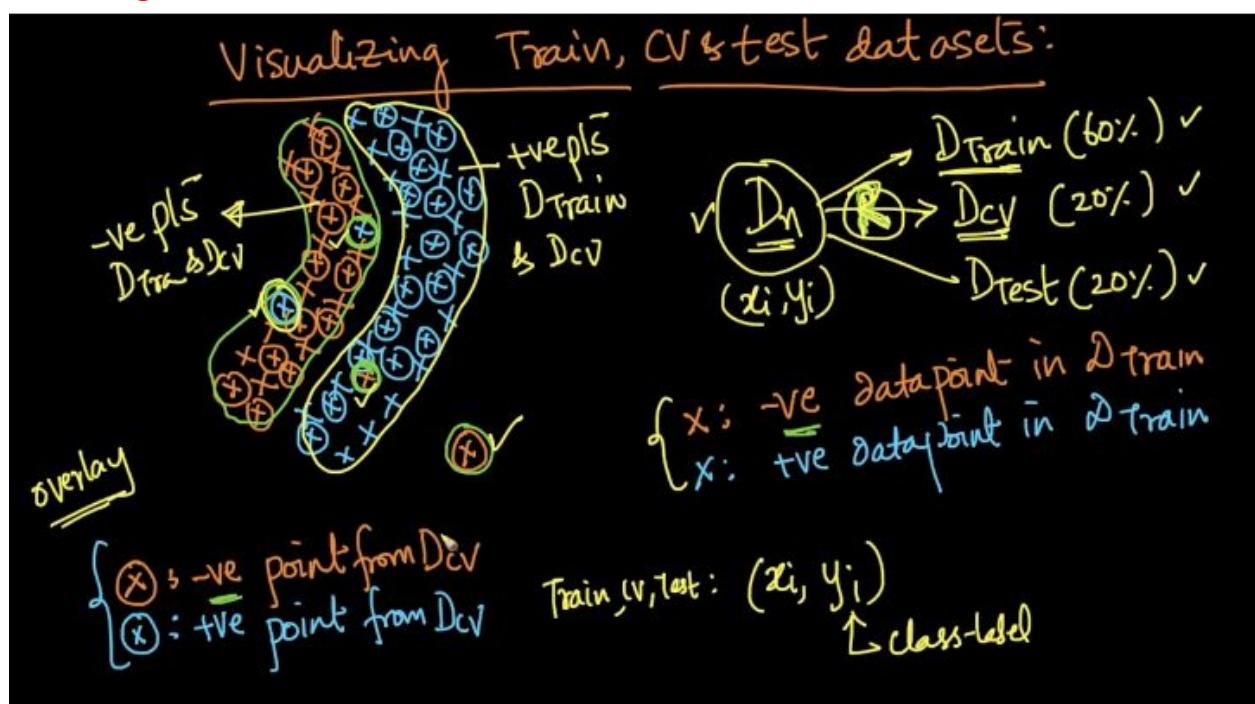
Cross validation, generally the data is splitted into $D(\text{train})$, $D(\text{CV})$ and $D(\text{test})$ with 60%, 20% and 20% of data respectively. But to achieve a good k-NN algorithm function, more data should be used to train instead of just 60%. Since we it is not right to change the content of $D(\text{test})$, therefore the datasets $D(\text{train})$ and $D(\text{CV})$ is mixed up to make it up of total 80% data.

Now from this 80% data we split it into 4 equal parts (D_1, D_2, D_3, D_4) and it is done randomly. And now from these parts we try to find best NN and k by taking three parts as training the data and 1 part as CV data and it is done for each and every combination of the 4 parts and also for every value of k . For example let say D_1, D_2 and D_3 will act as training data and D_4 is used as to find accuracy from the training data (D_1, D_2, D_3) by taking value of $k = 1$, and let say we get accuracy from each iteration as a_1, a_2, a_3 and a_4 . Now to find final accuracy for the value of $k = 1$, we take average of all accuracies.

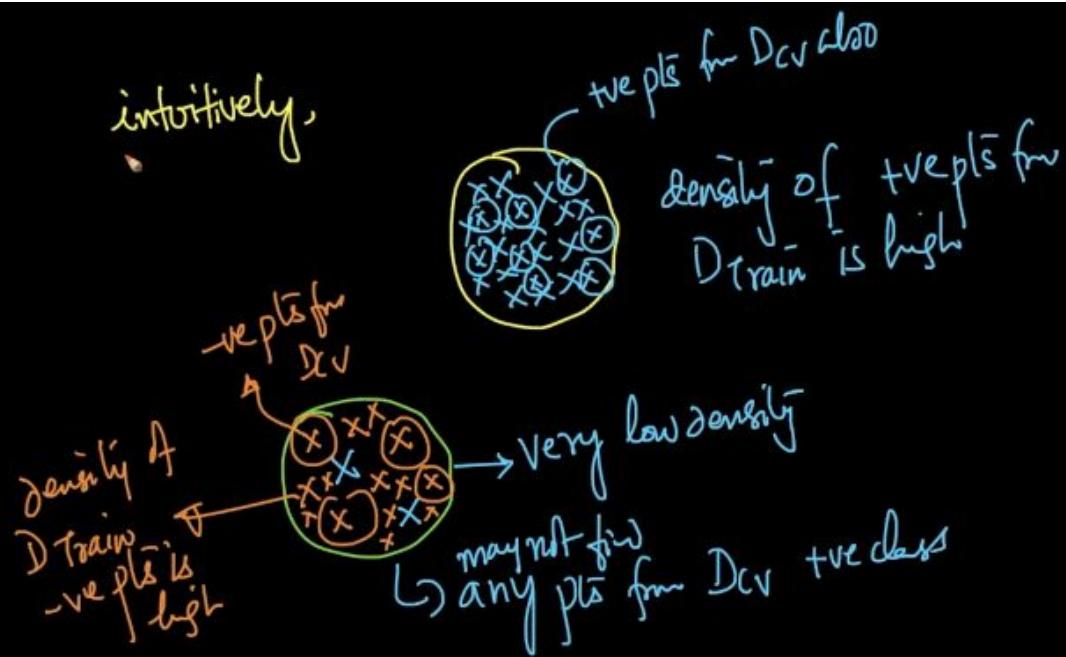
And this same process is done for all k values like 2, 3, 4, ..., n . From this a plot is plotted and best k can be observed.

So in this way all parts are being used to get the NN and k . And this whole process is known as K-fold cross validation. And since in above example 4 parts has been used therefore it will be called as 4-cross validation.

Visualizing train, validation and test datasets



- ✓ ① D_{train} & D_{CV} don't overlap perfectly when randomly sampled
- ② If there are many +ve pls from D_{train} in a region, then it is highly likely to find many +ve pls from D_{CV} in that region
- ③ If there are very few +ve pls in a region from D_{train} , then it is very unlikely to find any +ve from D_{CV} in that region
- noise error outlier*



How to determine overfitting and underfitting?

(Q16) How to determine overfitting and underfitting
Overfitting vs Underfitting

By using K-fold CV or DCV \rightarrow we get best $K \rightarrow$ not overfit or underfit

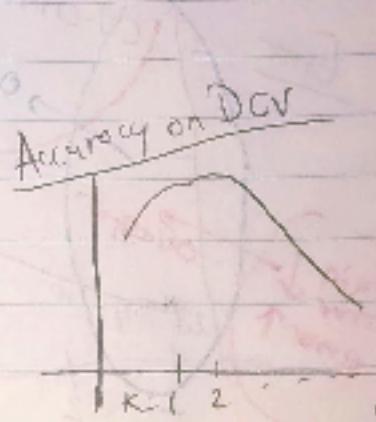
(Q2) How can we be sure it is neither Underfit or overfit?

(Ans) Accuracy = $\frac{\# \text{ correctly classified point}}{\# \text{ total points}} = 0.93 = 93\%$

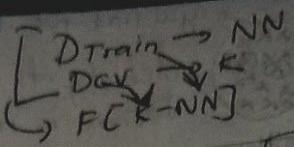
Reduce Error: $1 - \text{acc} = 0.07 = 7\%$

$D_n \rightarrow D_{\text{train}}$ $D_{\text{train}} \xrightarrow{\text{K-NN}}$ Accuracy on DCV
 $D_n \rightarrow D_{\text{CV}}$
 $D_n \rightarrow D_{\text{test}}$

Accuracy on DCV
for each point in DCV
 $x_q = p_t$
 $y_q \rightarrow y_i$



Training error:



for each x_i in D_{Train} :

- Find 2 nearest neighbors to x_i from D_{Train}

- Majority vote to get class label
- if $y_i = \text{class label}$ accurate

else error

what is error (train error)
if z -NN?

$D_{\text{Train}} \rightarrow \text{Training}$
 $D_{\text{Train}} \rightarrow \text{accuracy/error}$

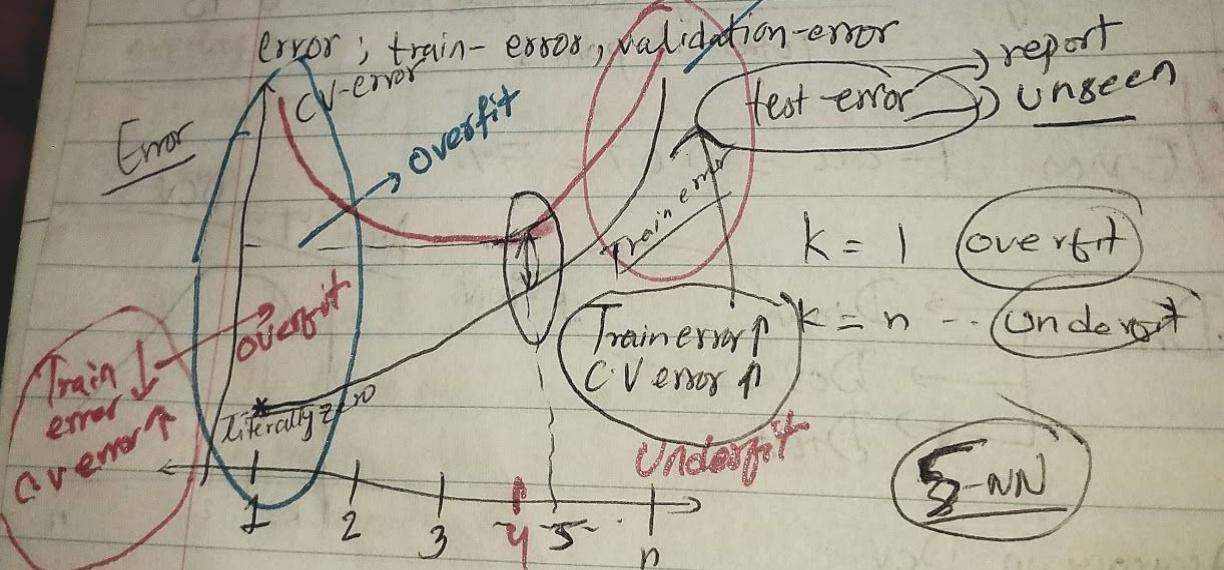
what is error of 2NN on D_{CV}

validation
error

$\left\{ \begin{array}{l} \text{Train: } D_{\text{Train}} \\ \text{Dcv: error on } D_{\text{CV}} \end{array} \right.$

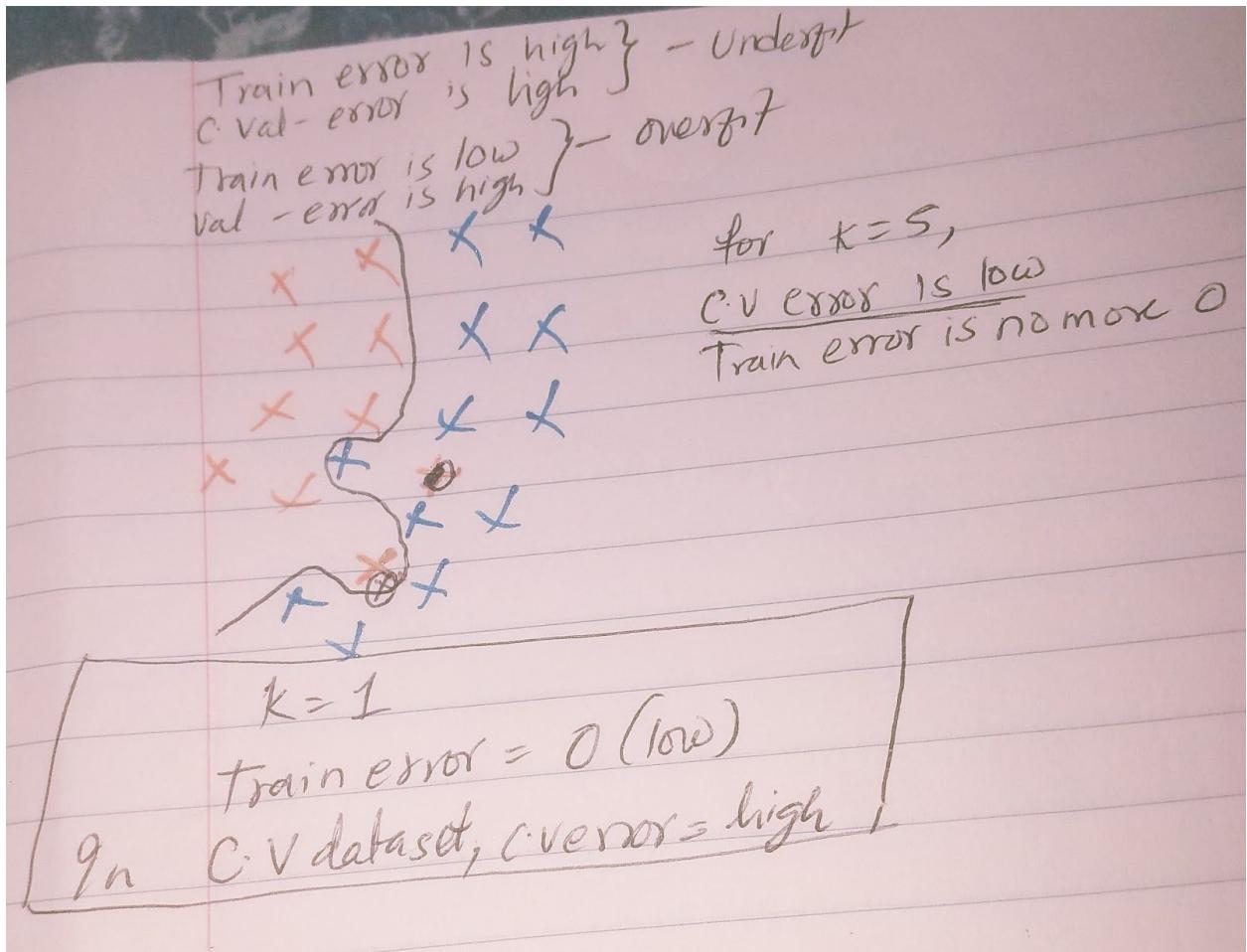
$\left\{ \begin{array}{l} \text{Train: } D_{\text{Train}} \\ \text{Dcv: error on } D_{\text{CV}} \end{array} \right.$

underfit

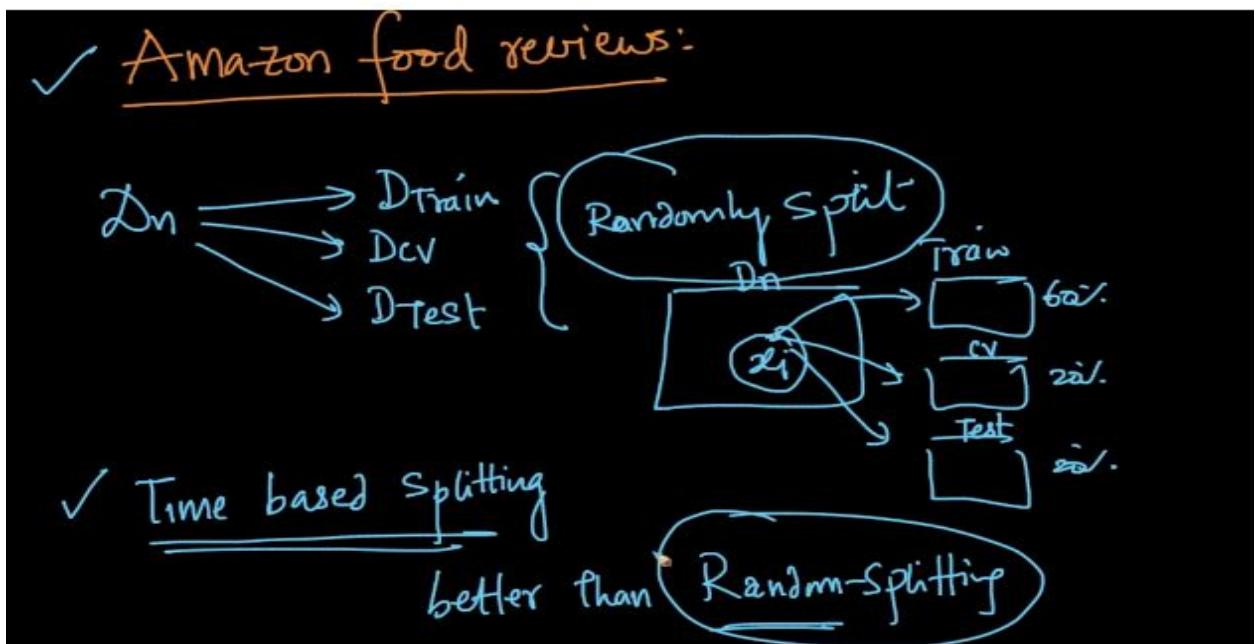


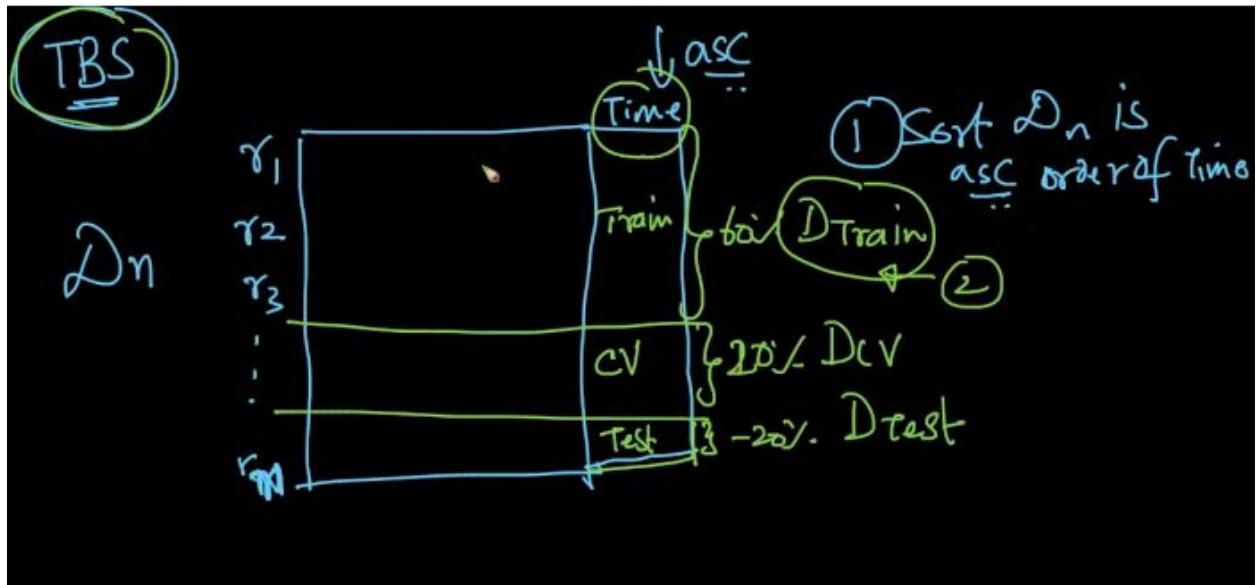
$D_{\text{Train}}, D_{\text{CV}}, D_{\text{Test}}$

K



Time based splitting





Amazon food TBS → (Time) is a necessary field

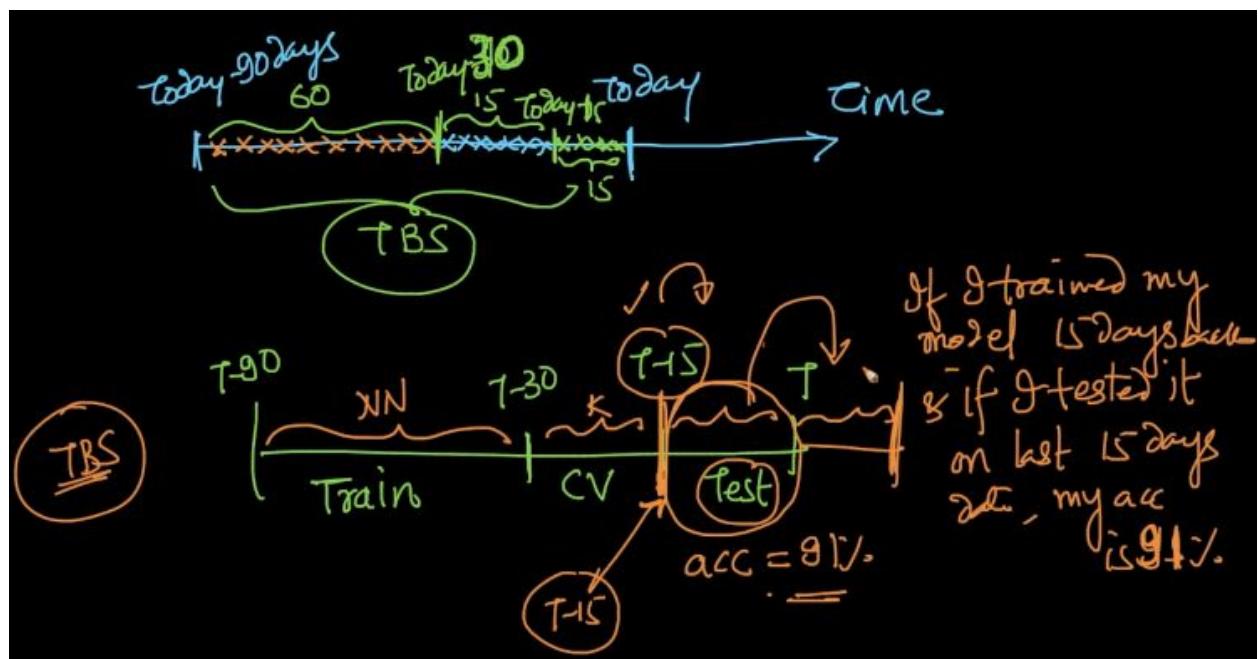
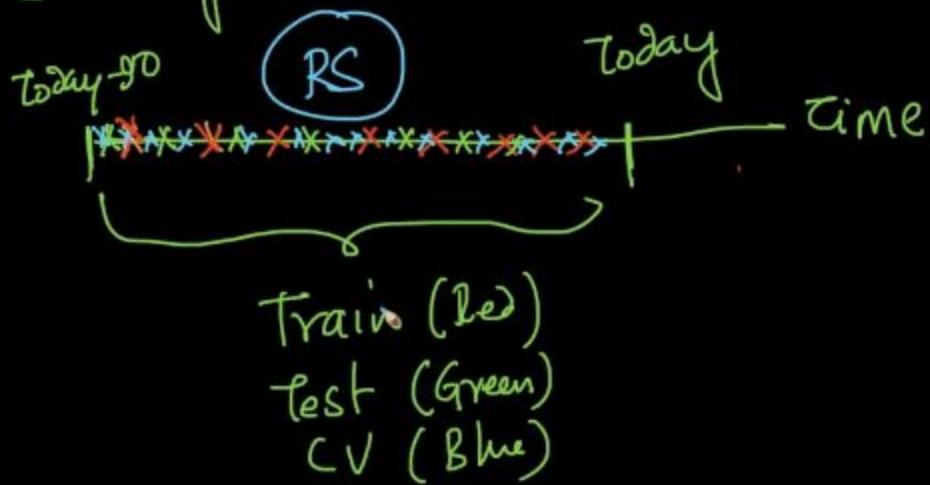
RS

Let's say I used random-splitting

5-NN acc is

$\frac{60\%}{20\%}$ D_train	\rightarrow	XN
$\frac{20\%}{20\%}$ D_cv	\rightarrow	k in k-NN $(k=5)$
$\frac{20\%}{20\%}$ D_test	\rightarrow	acc (93%)

{ With time, my products & their reviews
change }



Whenever time is available and things/behavior or data changes over times then doing time based splitting is preferable instead of random splitting.

KNN For Regression

(K-NN) for regression:

2-class classfn $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n \mid x_i \in \mathbb{R}^d, y_i \in \{0, 1\}\}$ ✓ find K-NN by majority vote

Regresⁿ $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n \mid x_i \in \mathbb{R}^d; y_i \in \mathbb{R}\}$ $x_q \rightarrow y_q \rightarrow \text{number}$

① given (x_q) , find k -nearest neighbors $(x_1, y_1), (x_2, y_2), \dots, (x_k, y_k)$

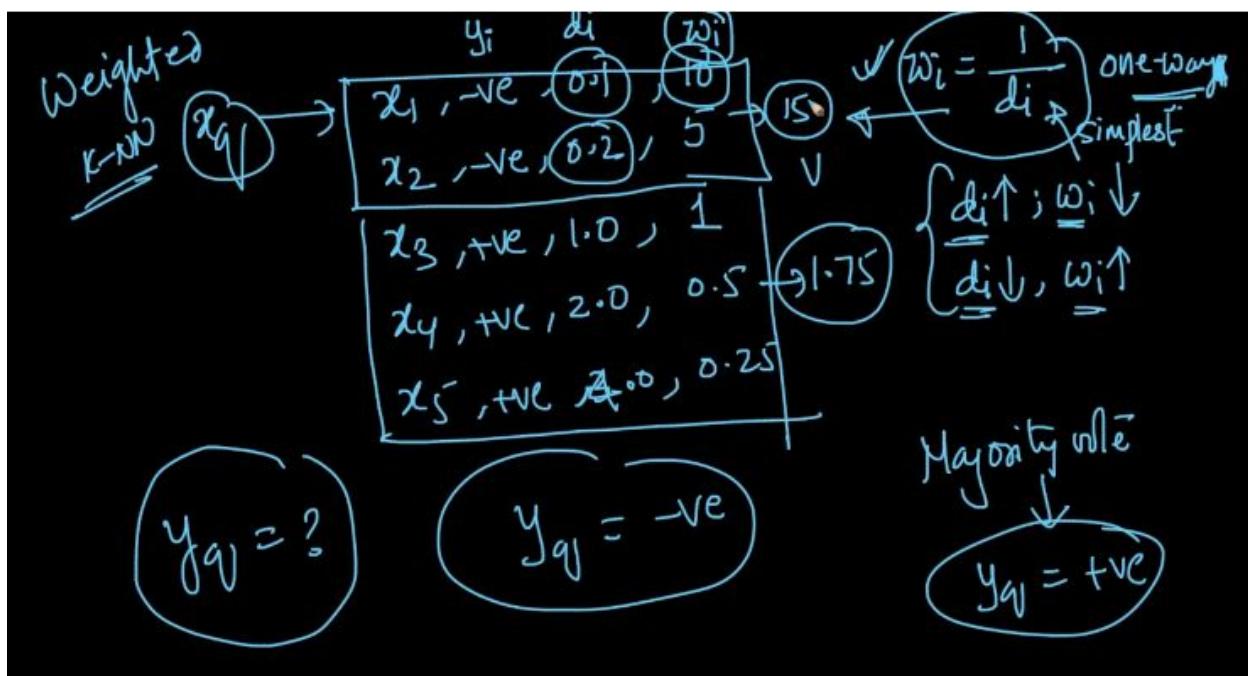
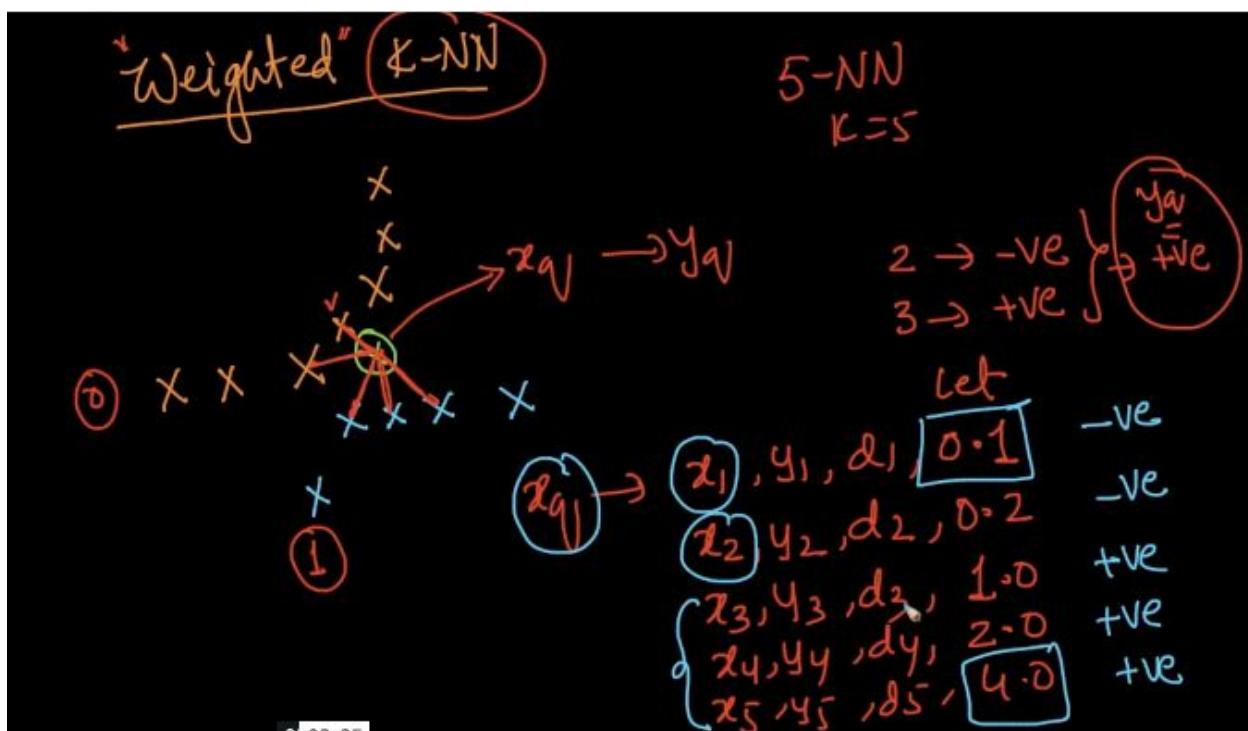
~~Class \rightarrow Majority vote~~

② $y_q \leftarrow$ ~~mean~~ $y_1, y_2, y_3, \dots, y_k$ \rightarrow not \mathbb{R}

regresⁿ: Simple extension/modificⁿ of classfn \rightarrow less prone to outliers

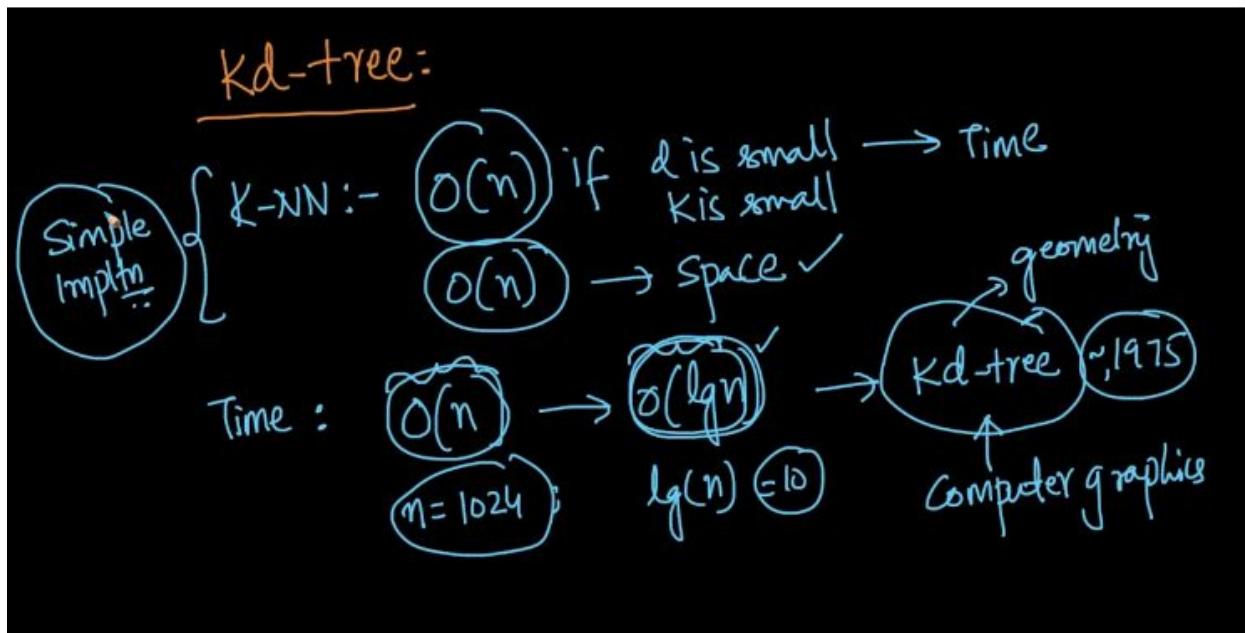
$$\begin{cases} y_q = \underline{\text{mean}}(y_i)_{i=1}^k \\ y_q = \underline{\text{median}}(y_i)_{i=1}^k \end{cases}$$

Weighted k-NN

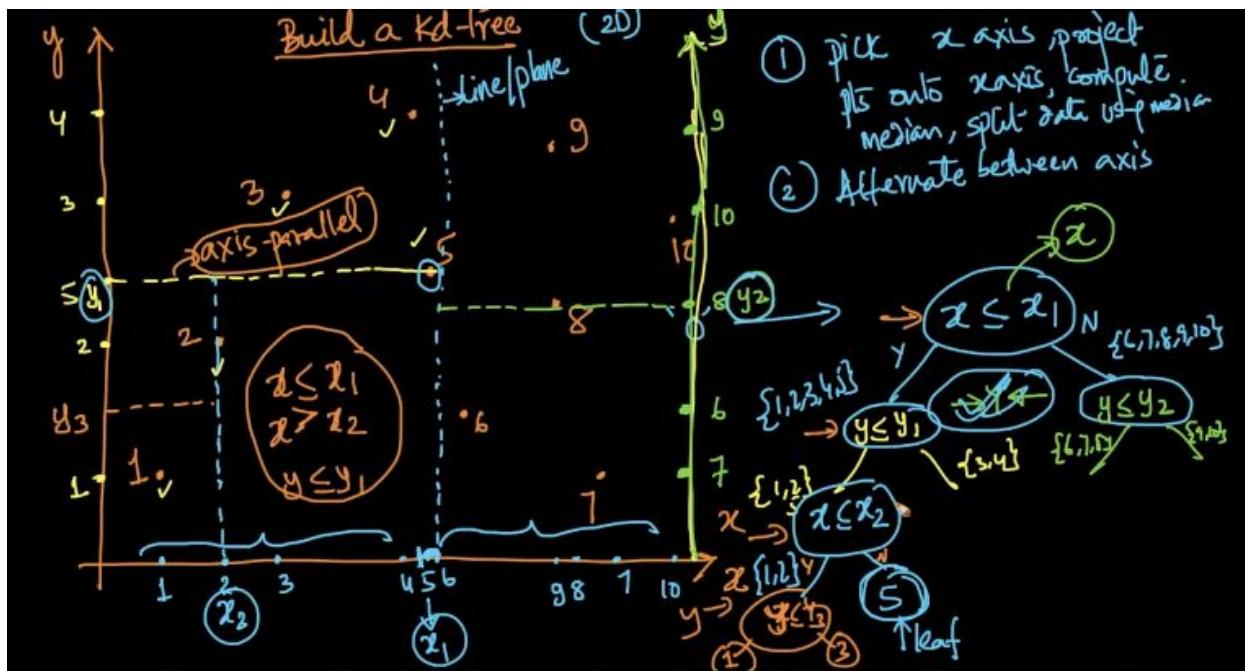


$$W_i = 1/d_i \quad (\text{Simplest Weight function})$$

Binary Search Tree



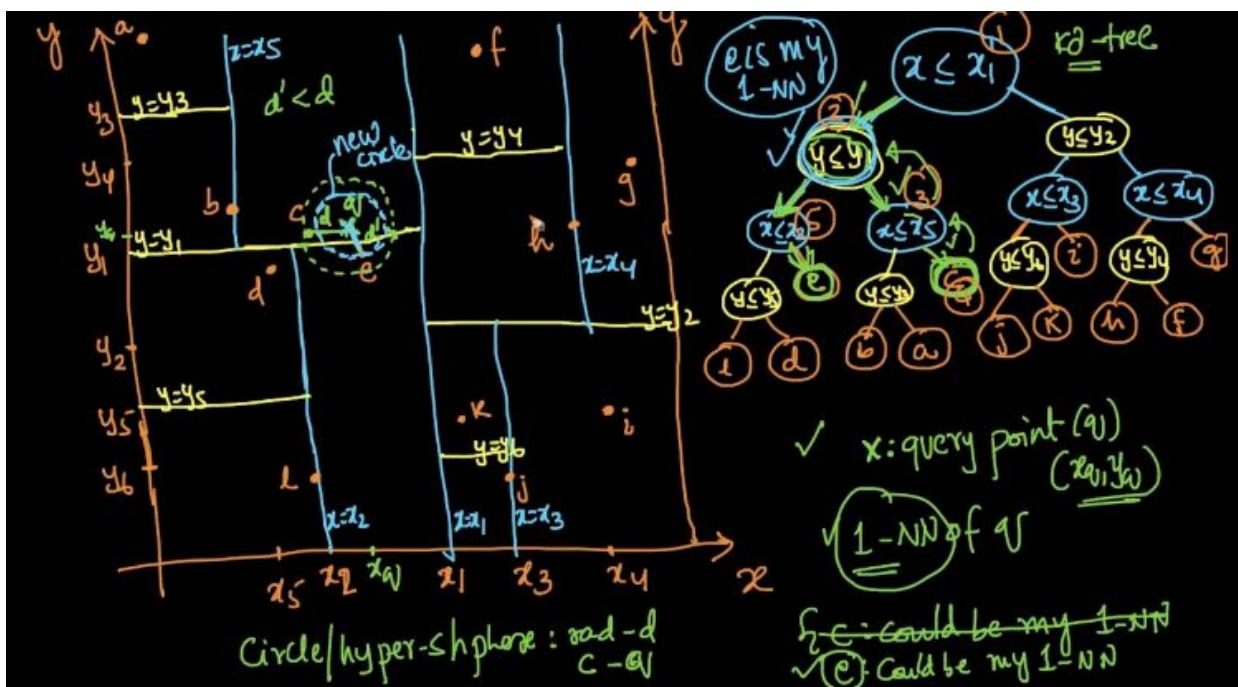
Kd-tree

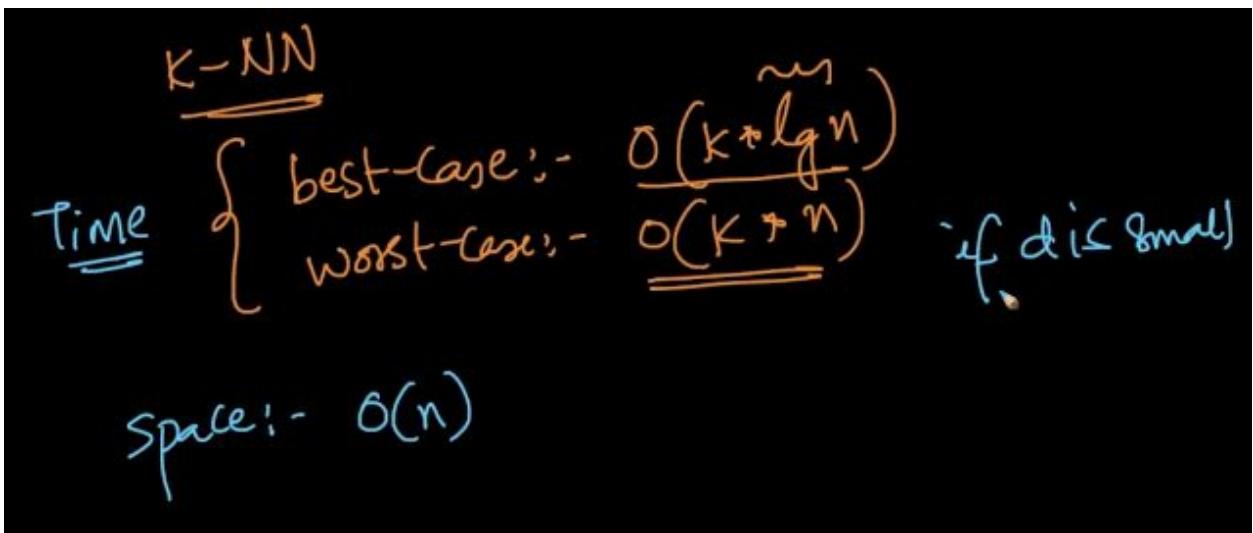
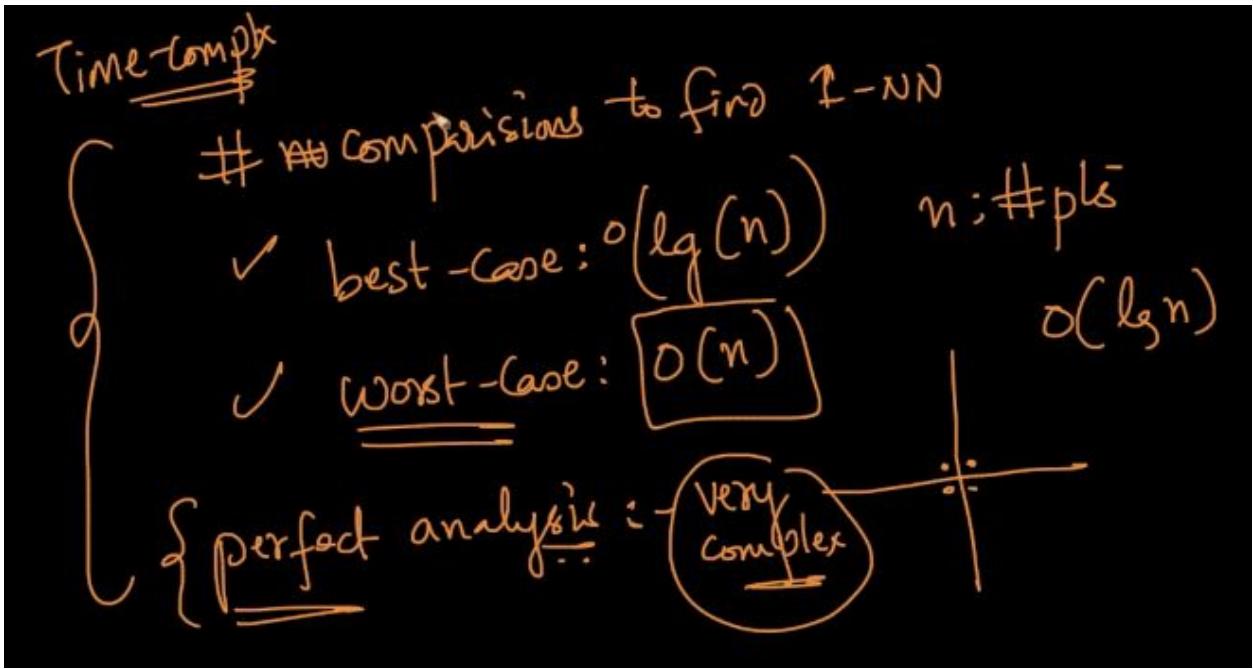


\leftarrow go through each axis one after the other till leaf node
 kd-tree \hookrightarrow breaking up space using axis-parallel lines / planes into rectangles / cuboids / hypercubes

$\left\{ \begin{array}{l} 2D \rightarrow \text{axis parallel lines} \rightarrow \text{rectangles} \\ 3D \rightarrow \text{" " planes} \rightarrow \text{cuboids} \\ nD \rightarrow \text{" " hyperplanes} \rightarrow \text{hypercubes} \end{array} \right.$

Find nearest neighbours using kd-tree





Limitations of k-d tree

Limitations of Kd-tree:

① When d is not small {2, 3, 4, 5}

\downarrow

$d = 10 ; 2^d = 1024$

$d = 20 ; 2^d \approx 1 \text{ Million}$

$d \uparrow$ worst-case #adj. cells 2^d

$n = \text{(Million)}$

$2^d \approx n$

$O(\lg n)$

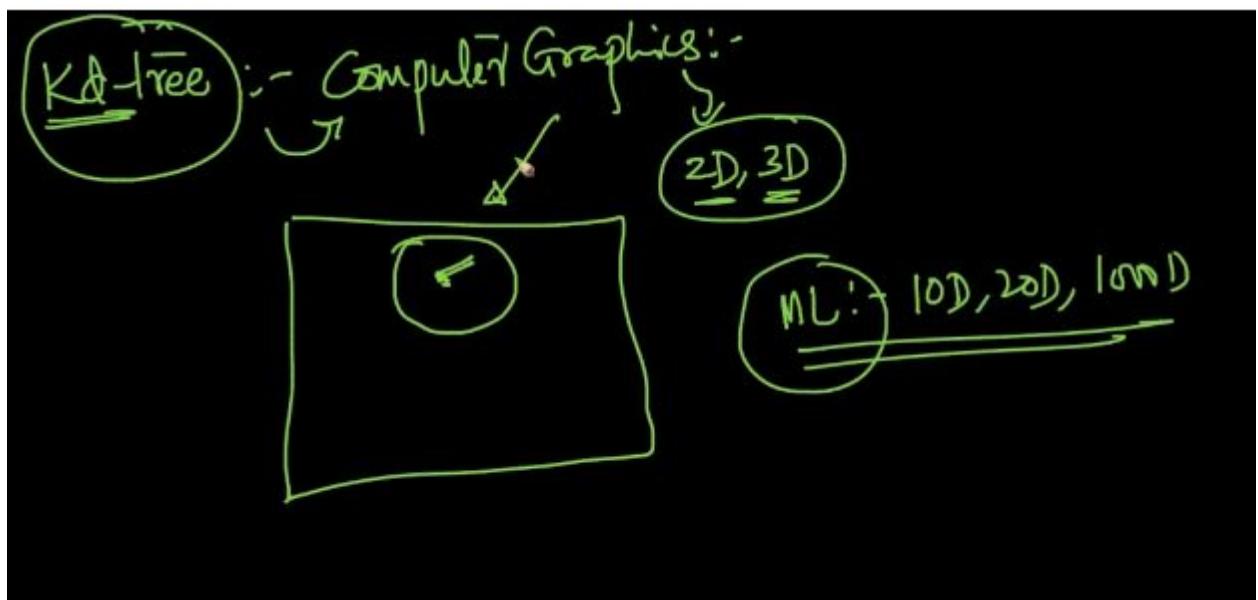
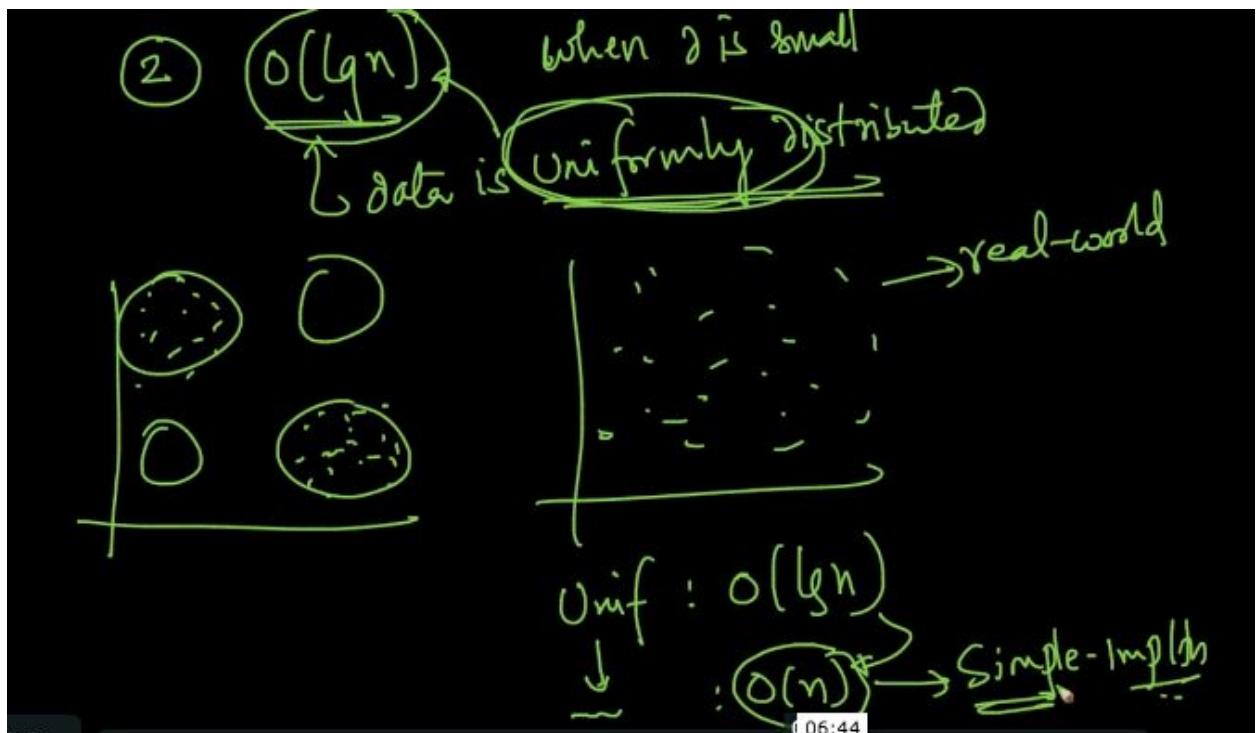
✓ When d is not small {2, 3, 4, 5} $\leftarrow 1\text{-NN}$

Time complex :- $O(\lg n)$

$O(2^d \lg n) \leftarrow \cancel{O(n \lg n)} \leftarrow O(n)$

$2^d \approx n$

{ When d is 10, 11, 12, ... Time complexity increases dramatically }



Extension : Variation in k-d tree

https://en.wikipedia.org/wiki/K-d_tree

Hashing

