Clustering :-

ML :-

- Supervised → Ind | dep → (target) → Linreg, logreg, SVM, DT, RF, AB, GB, XB, KNN, NAIVE BAYES
- Unsupervised → X, y → x_1 x_2 x_3 x_4 ... x_n → grouping → clustering → K means, Kmeans++, Hierarchical, DBSCan.

Lazy Learner ← Supervised ML Algo (X, y)

- how this thing is working

eager learner

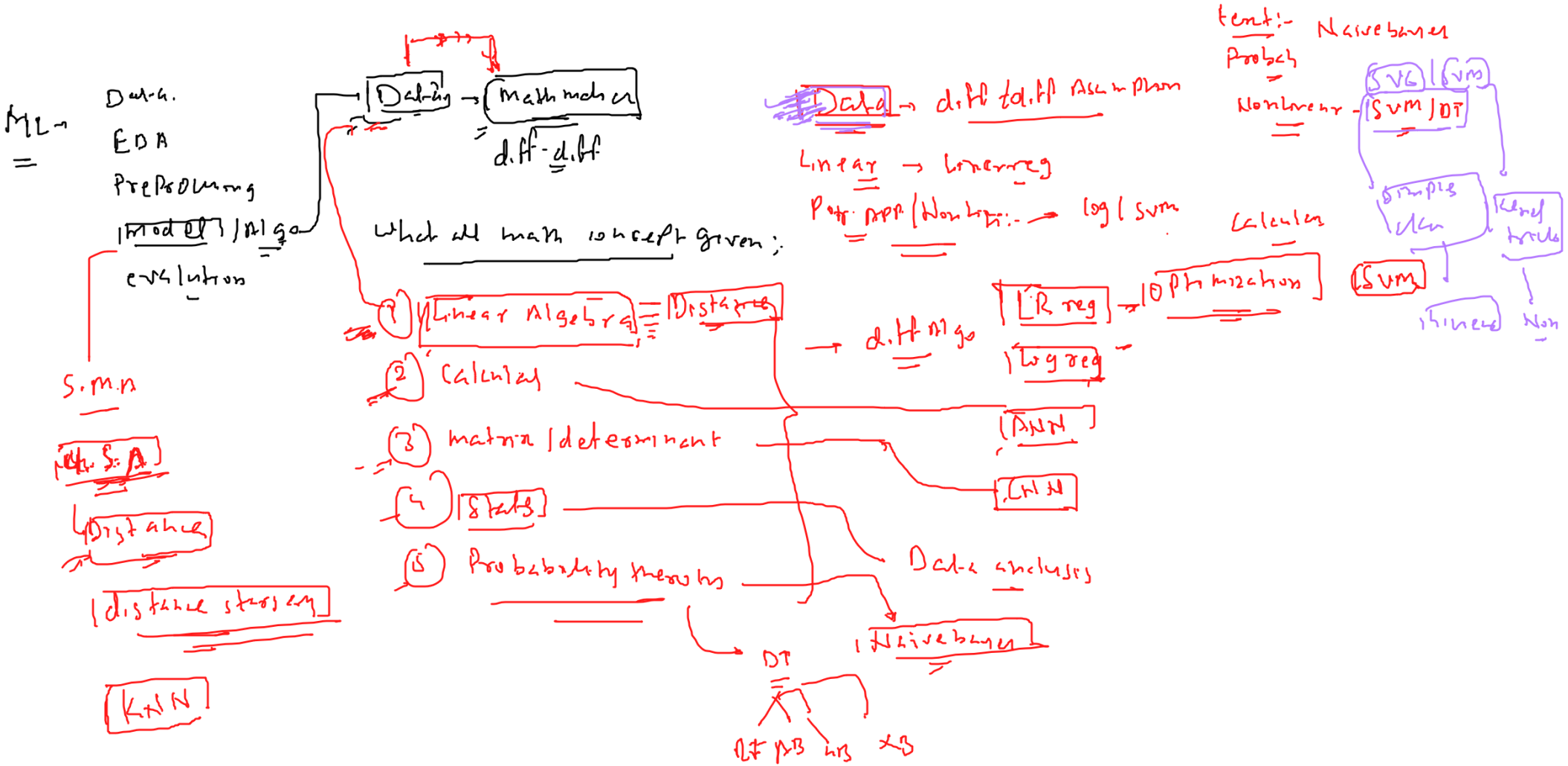
K means, Kmeans++, Minibatch. kmeans → Sklearn → automatically → Data → batches.

(m x n) → x_1 x_2 ... x_n

m x c

10000 → train → Batches → 1000, 1000, 1000

Procedure


ML
Data
EDA
Preprocessing
Model / Algo
evaluation
Data
Mathematics
diff - diff
Which all math concept given:
1 Linear Algebra = Distance
2 Calculus
3 matrix / determinant
4 Stats
5 Probability theorem
Data → diff diff Assumption
Linear → Linear reg
Para / Non Para → log / svm
diff Algo
LR reg
Log reg
Optimization
Calculus
ANN
CNN
Data analysis
Naivebayes
DT
RF XB LGB XB
Text:- Naivebayes
Prob
Nonlinear - SVM / DT
SVG / SVM
SVM
Simple data
Kernel tricks
Linear
Non
S.M.n
K.S.A
Distance
distance strategy
KNN

K-Means Algo → Distance (Group)

= Unsupervised

K-NN :- K Nearest Neighbor → Supervised

Both are diff thing

Unsupervised

X_1 X_2 X_3 .. X_n

Grouping of the data

I.F D.f → target var

X (weight)	Y (Height)	
60	170	Regression
70	160	
80	115	
90	185	
100	155	

Classification

X_1 (W)	X_2 (H)	Y (Yes/No)
60	170	No
70	160	No
80	150	No
90	140	No
100	130	No

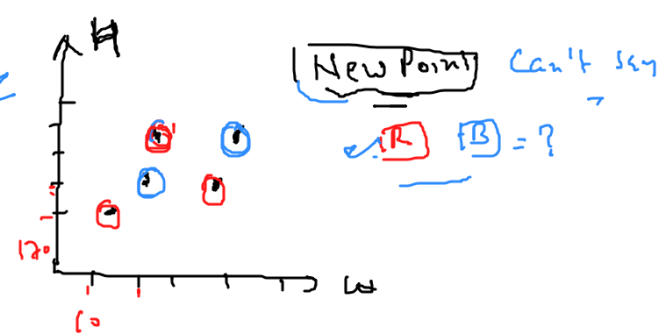
K-Nearest Neighbor :-

No of data Point

Near

Neighbor

KNN



New Point Can't say

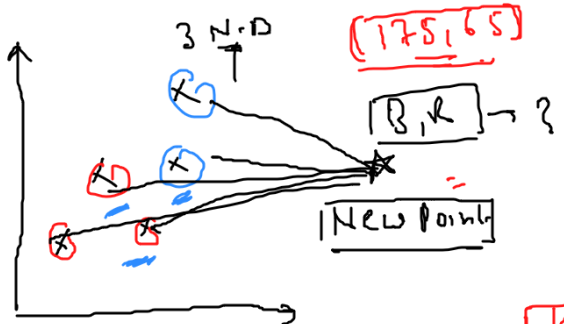
R B = ?

Hyperparameter → K = 1, 3, 5 ... N

KNN :- Data classification

Near Neighbor

(Near data point)



K = 3

K = 1, 3, 5, 7, 9, 11 → odd

K = 2, 4, 6, 8, 10 → even ✗ why ?

	X_1 (W)	X_2 (H)	Y (O/No)	Distance
((175, 65))				
1	60	175	05	5
2	70	170	NO	15.81
3	75	155	05	22.36
4	85	150	NO	26.92
5	95	160	08	30.41

3 Near Neighbor

→ 3 Near data Point

[1] I have to cal. distance to each Data Point

[2] we have to choose 3 Nearest Dist.

[3] Prob

[4] decide to which class it will belong.

(175, 65)

(1) (175, 60)

(2) (160, 70)

(3) (175, 75)

(150, 85)

(5) (180, 95)

Euclidean distance

$$\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}$$

New data Point → OB, OB, No → observe → Probability → majority

KNN = yahoo Game

K-NN → Machine learning algo? Not

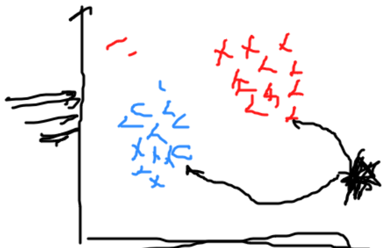
No ⇒ Lazy learner

2/3 1/3

2/3 OB

K-Nearest Neighbor

4 = Near data Point, select



Distance

(1) first distance with all Point

(2) K-value → hyperParameter

1, 3, 5, 7, 9, 11, 13 -- N → odd

2, 4, 6, 8, 10, 12, -- N → X even

even value

Red

Bag 7R, 4; 10, 3 - R; 3 Nearest

K=4 → B B / R R ← ? New data Point belongs to which class

B, B, R, R, R ↓ R

B, B, B, R, R ↓ B

B, R, R, R, R ↓ Red

Eger learner

training - calculating

Machine learning Algo :- 1) calculation

2) loss

3) Optimization

LR - Cal - $y = mx + c$

$loss = (y - \hat{y})$

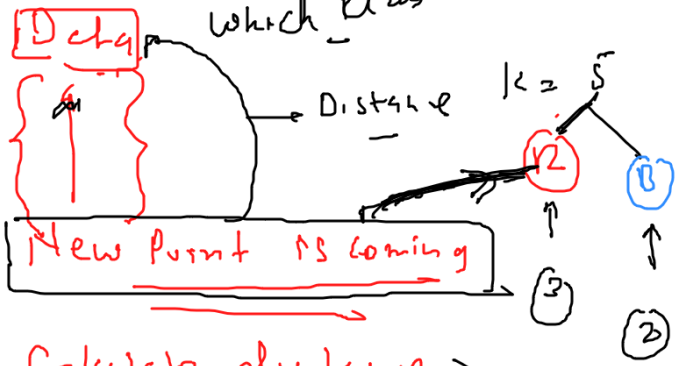
optimization - gradient descent.

KNN : 1D

$x_1 x_2 x_3 x_4 - x_n$ Y → problem (B C)

Log reg - Cal → Prob → sigmoid

(training)



(log loss

optimization ⇒ Gradient descent

or

Maximum

Likelihood

Estimation

SVM :- Cal

loss

optimization

DT :- Prob: best Node.

[No Loss]

[No optimization]

Data :- KNN

Near data Point

we don't know

hyper Parameter — 1, 3, 5, 7, 9, 11 →

K

2, 4, 6, 8, 10 X

higher accuracy → Plot { K value / accuracy } → K_means algo

1, 3, 5, 7, 9, 11

Machine learning → training → Generate pattern → CNN → training

Data → if test data is coming

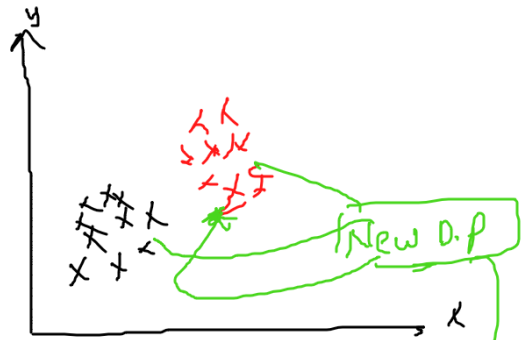
Distance, N. Neighbour

Lazy learner

Rest of also eager learner

KNN :- Classification — majority of the class

Regression :- mean value



K = 3

majority vote

2R, 1B = R

$\frac{18+19+19.5}{3}$

$\downarrow$

18.83

Nearest distance

K = 3

18, 19, 19.5

? final

K-Nearest Neighbour

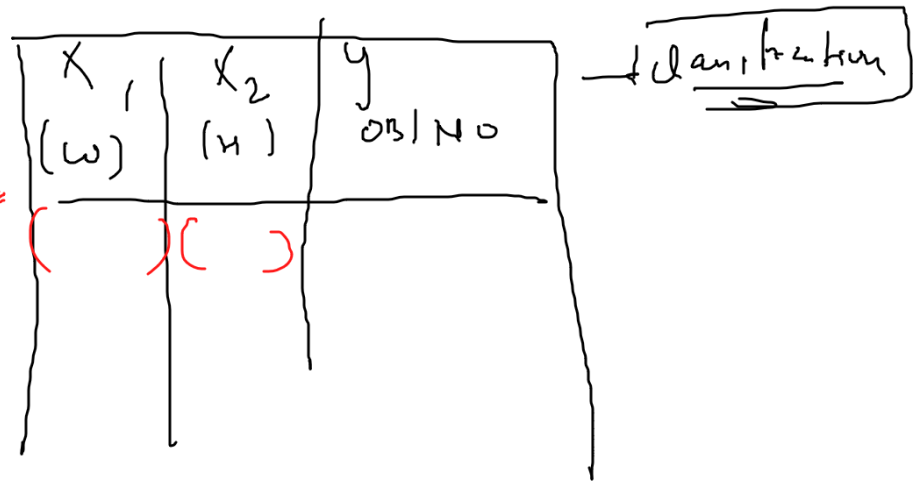
$\frac{18+19+19.5+20}{5}$

? final

	X_1 (W)	X_2 (H)	Y (BMI)
1	70	160	18
2	80	170	15
3	50	180	19.5
4	100	150	20
5	120	200	22
	130	140	?

18.83

Machine learning → Reg
→ Clas



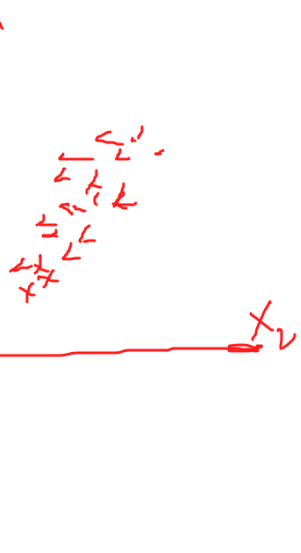
New Point

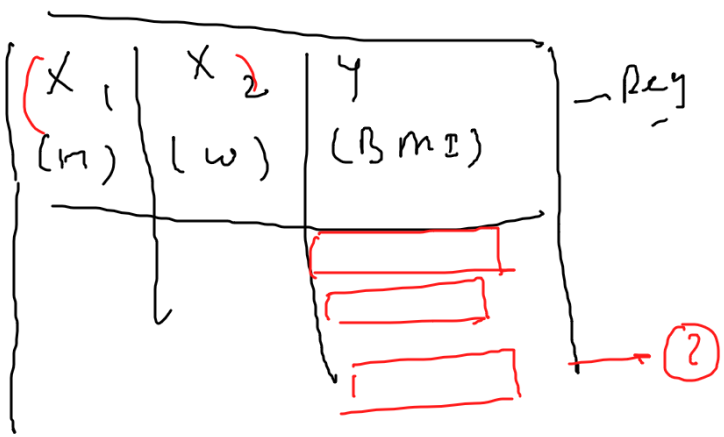
(X1, X2)

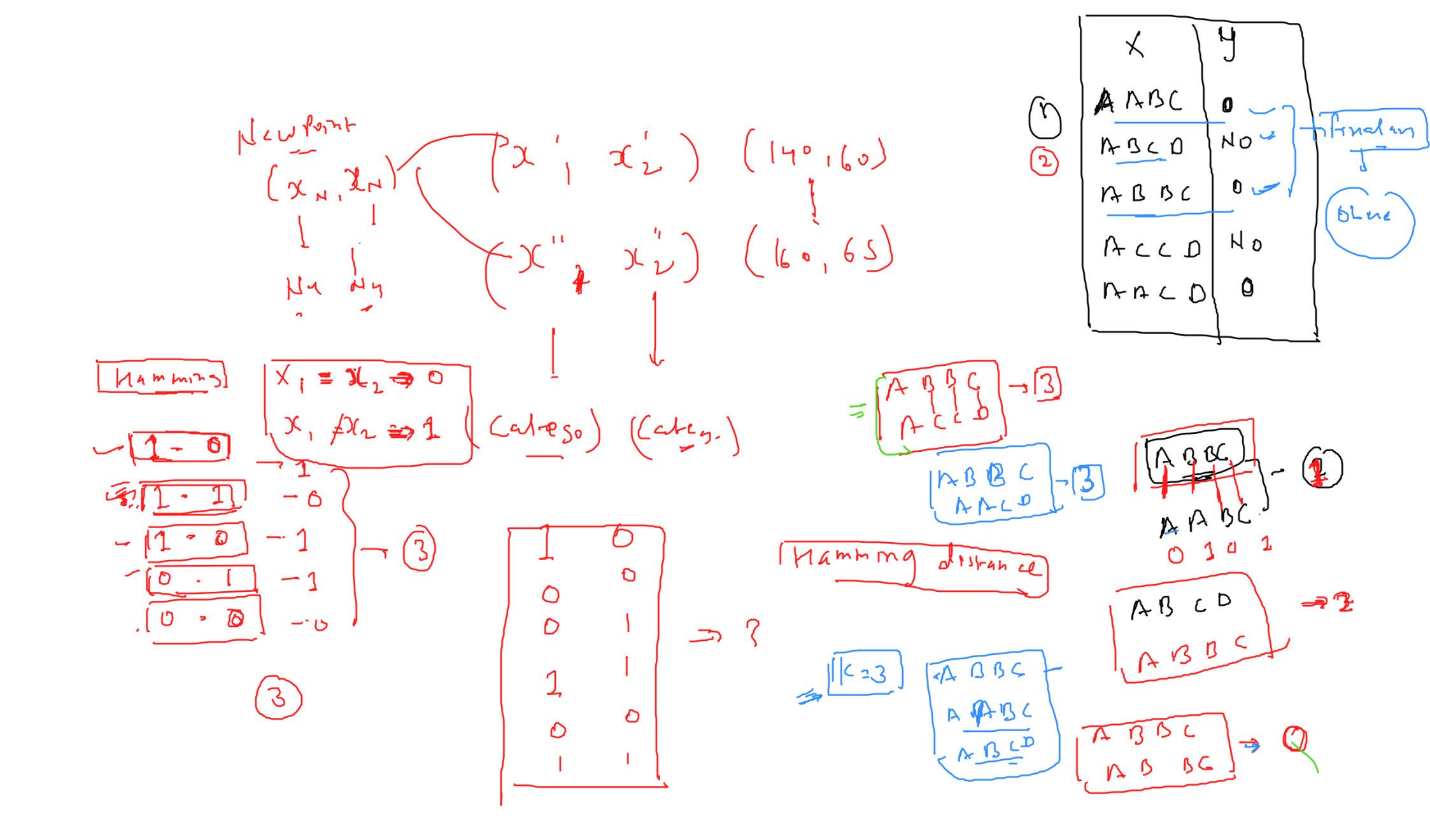
(Xw, Xh)

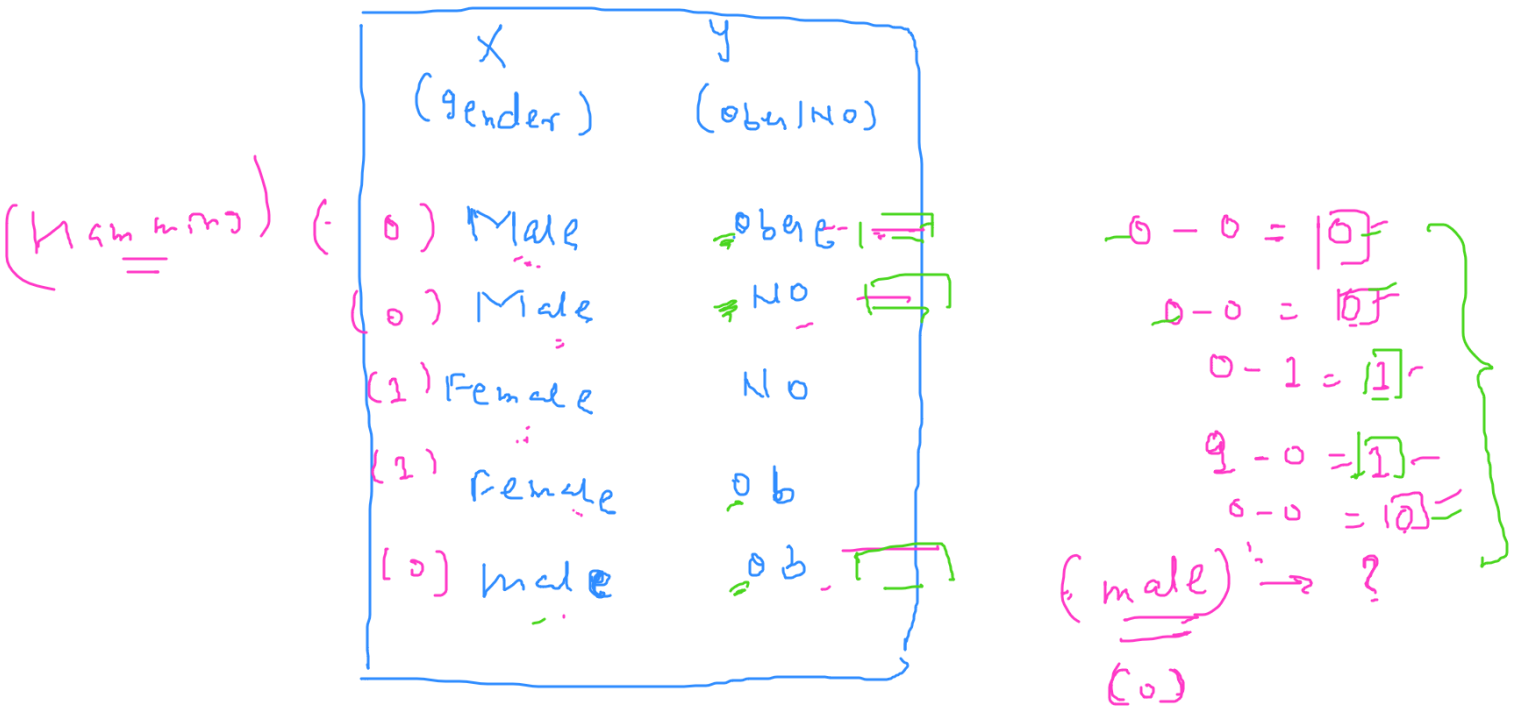
Distance → Category

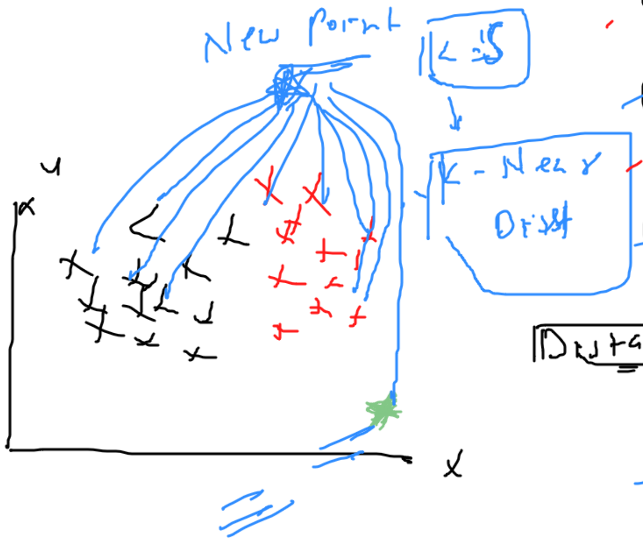
final value with X1






New point
(x_N, x_N)
N_4 N_4
$(x_1' \; x_2')$ $(140, 60)$
$(x'' \; x_2'')$ $(60, 65)$
Hamming
$x_1 = x_2 \Rightarrow 0$
$x_1 \neq x_2 \Rightarrow 1$
Calcso) (Calcs..)
1 - 0
1 - 1
1 - 0
0 - 1
0 - 0
1
- 0
- 1
- 1
- 0
3
3
X
Y
ABC 0
ABCD NO
ABBC 0
ACCD NO
AACD 0
final an
Blue
1)
2)
ABC
ACD
3
ABBC
AACD
3
ABBC
AABC
0 1 0 1
1
ABCD
ABBC
2
Hamming distrance
C=3
ABBC
AABC
ABCD
ABBC
ABBC
AB BC
1
0
1 0
0 0
0 1
1 1
1 0
0 0
1 1
→ ?


X
(gender)
y
(obesity)
(0) Male
(0) Male
(1) Female
(1) Female
(0) Male
obese
NO
NO
ob
ob
(Kamini)
0 - 0 = 0
0 - 0 = 0
0 - 1 = 1
1 - 0 = 1
0 - 0 = 0
(male) → ?
(0)



Distance

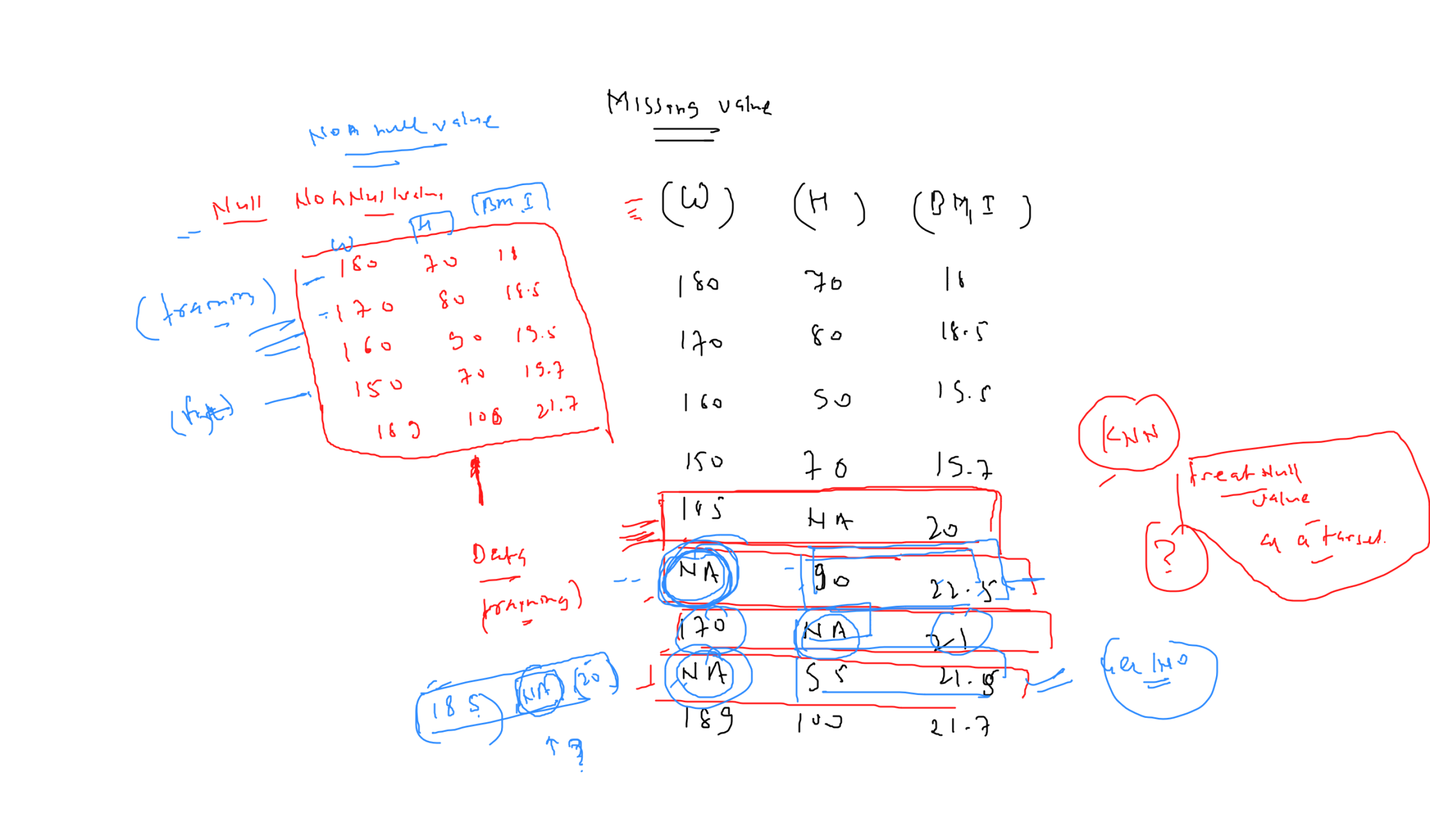
(KNN)

(160, 85
170, 190)

(16 8
17 5)

(KNN = ?) 10, 20, 10, 50

1. K-NN => No. of Point
2. R or C => Yes.
3. Is KNN ML algo → No | Yes
   - Lazy => no training, No loss, No optimizing
   - eager =>
4. Data (large) Y | N — (Why?) — lots of memory (Lazy learner)
5. Robust to outlier? Yes.
6. In KNN → do we req. Scaling = ?
   - Distance → Scaling — (standard scaling) (min-max)
7. Can we use this missing value?
8. Can we reduce time complexity of KNN?
   - which method :- kd tree, Ball, BF
   - ↓ Can you explain


Missing value
NaN null value
Null
Non Null value
BMI
W
H
160 70 18
170 80 18.5
160 50 19.5
150 70 19.7
160 100 21.7
(training)
(test)
Data
(training)
W H BMI
180 70 16
170 80 18.5
160 50 15.5
150 70 15.7
185 NA 20
NA 90 22.5
170 NA 21
NA 55 21.5
189 100 21.7
185 NA 20
?
KNN
Great null value
as a target.
?
Regression

Ball tree

K-D tree

if K-NN - lazy learner

optimize

traversing - tome

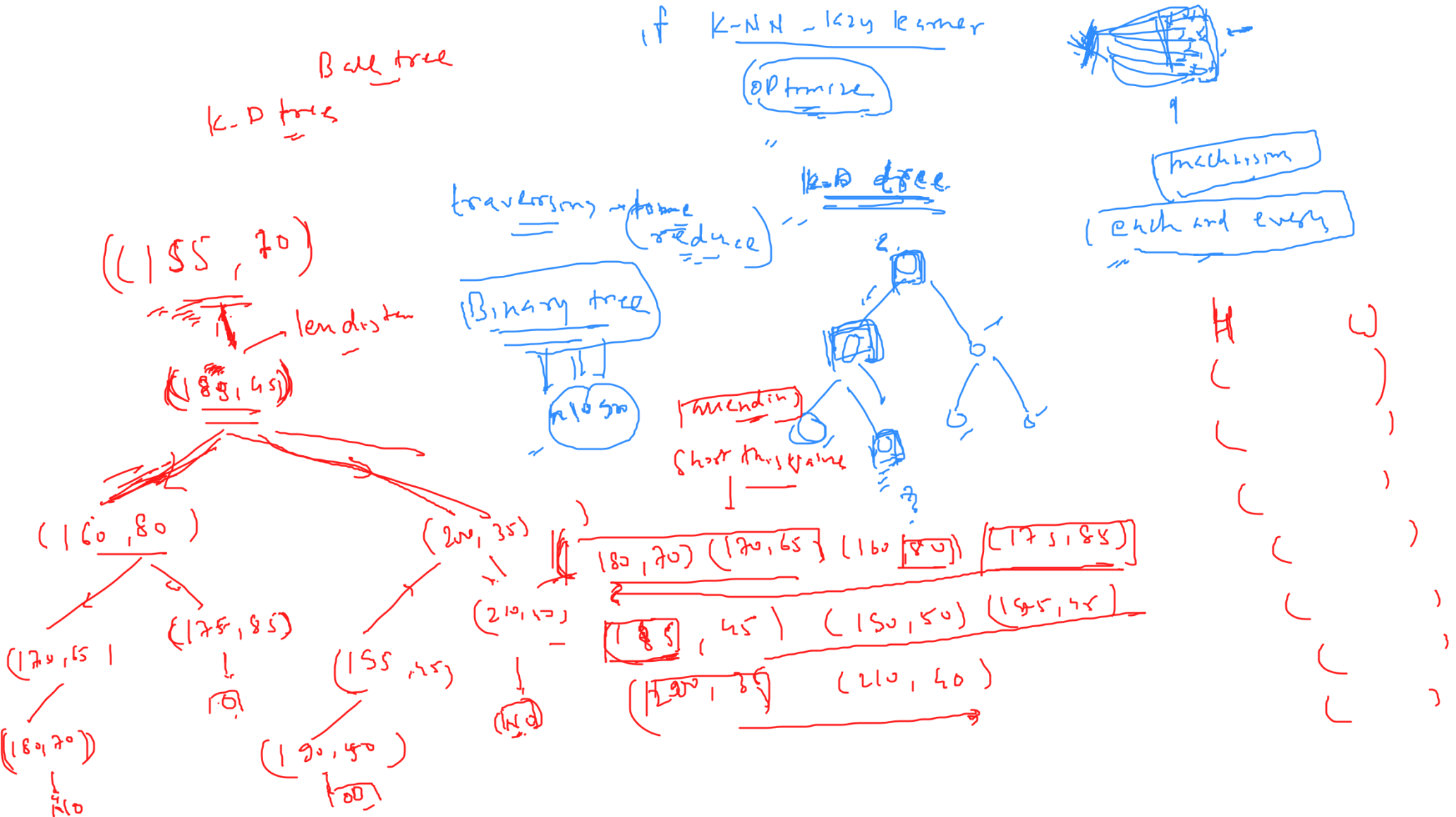
reduce

K-D tree

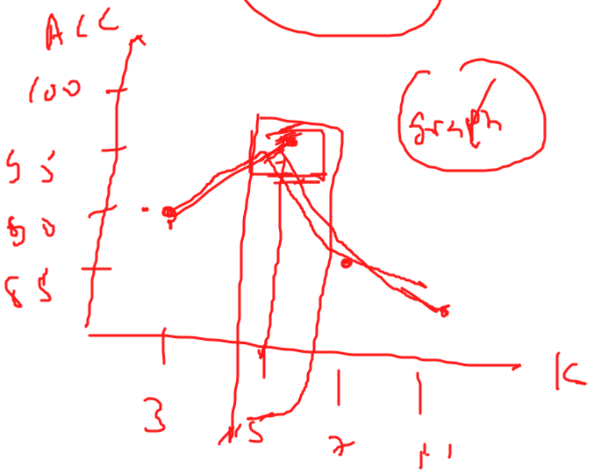
Binary tree

mechanism

each and every



Acc | k



Machine learning — missing → known

K-value — 1, 3, 5, 7, 9, 14 = [illegible]

Experimental — |Data should small|

Data

3

training | fit ← 3 test ⇒ 90%

— 5 = 95%

— 7 — 89%

— 11 — 85%

NAIVE BAYES

ML → (Experimental) |to calculate|

KNN

time

|Rest of algo|

|Compt Power|