~~[illegible]~~ -

Classification models

→ Logistic Regression

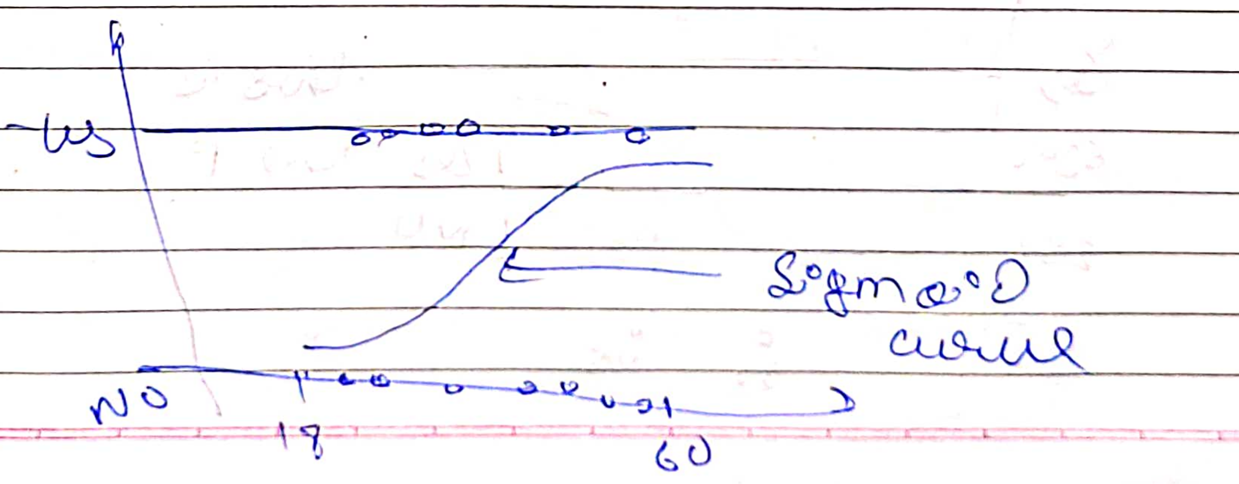
Suppose we want to predict whether someone will buy insurance or not based on age

So "

So independent var = age

Dependent var = Yes / NO

So graph ~~looks~~ looks like



Here we can't use simple regression i.e

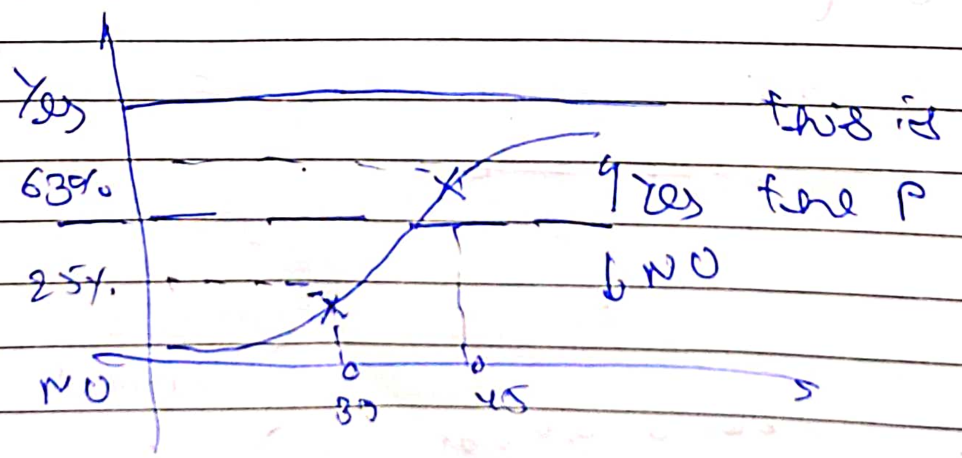
$y = b_0 + b_1 x$ But

Here we use Logistic Regression where equation is

$$\ln \frac{p}{1-p} = b_0 + b_1 x_1$$

(P is probability)

So Logistic Regression will give us the probability of someone saying yes or NO



we can also define a Boundary anything above that is 'yes' and ~~below~~ Below that is NO

we can also have multiple independent variables for them eg odds like

$$\ln \frac{p}{1-p} = b_0 + b_1x_1 + b_2x_2 + b_3x_3 + b_4x_4$$

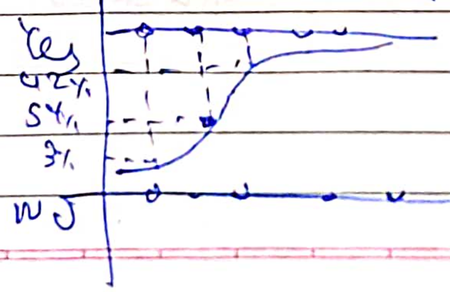
→ maximum ~~likelyhood~~ likelyhood

Just like linear Regression we need to find the best curve that fits our dataset

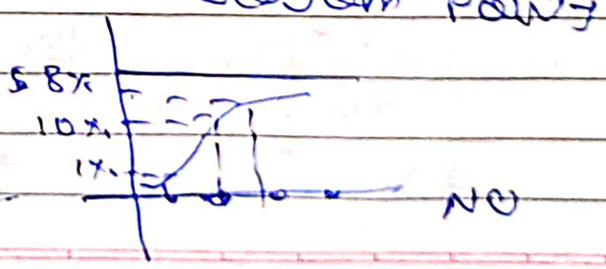
in linear Regression we used sum of least squares here we use maximum likelyhood

first we compute the likelyhood

top points



Bottom points



we find the value calculated if data point was not present and multiply likelihood

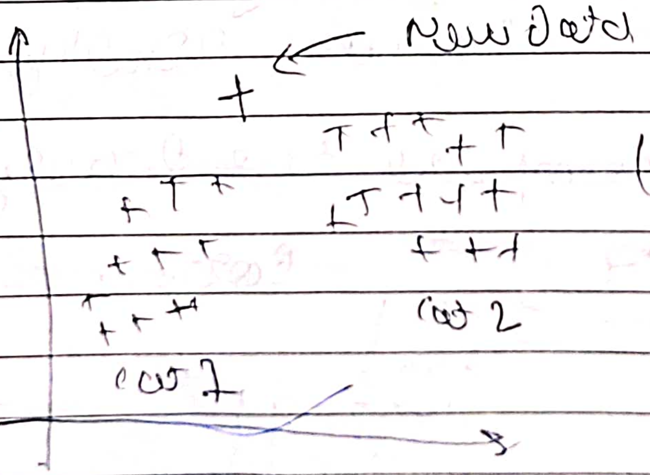
Likelihood

$$= 0.03 \times 0.05 \times 0.54 \times (1 - 0.01) \times (1 - 0.10) \times (1 - 0.58)$$

then we compute all possible curves and take the curve with max likelihood

[Note in reality not all curves are actually computed]

→ KNN (K th nearest neighbour)



(lets say we have 2 independent variables) we need to decide where will the new data point go? will it go into cat 1 or cat 2

to find this we use the kth nearest neighbour algorithm

Step 1:- Choose number of K neighbours (generally K = 5)

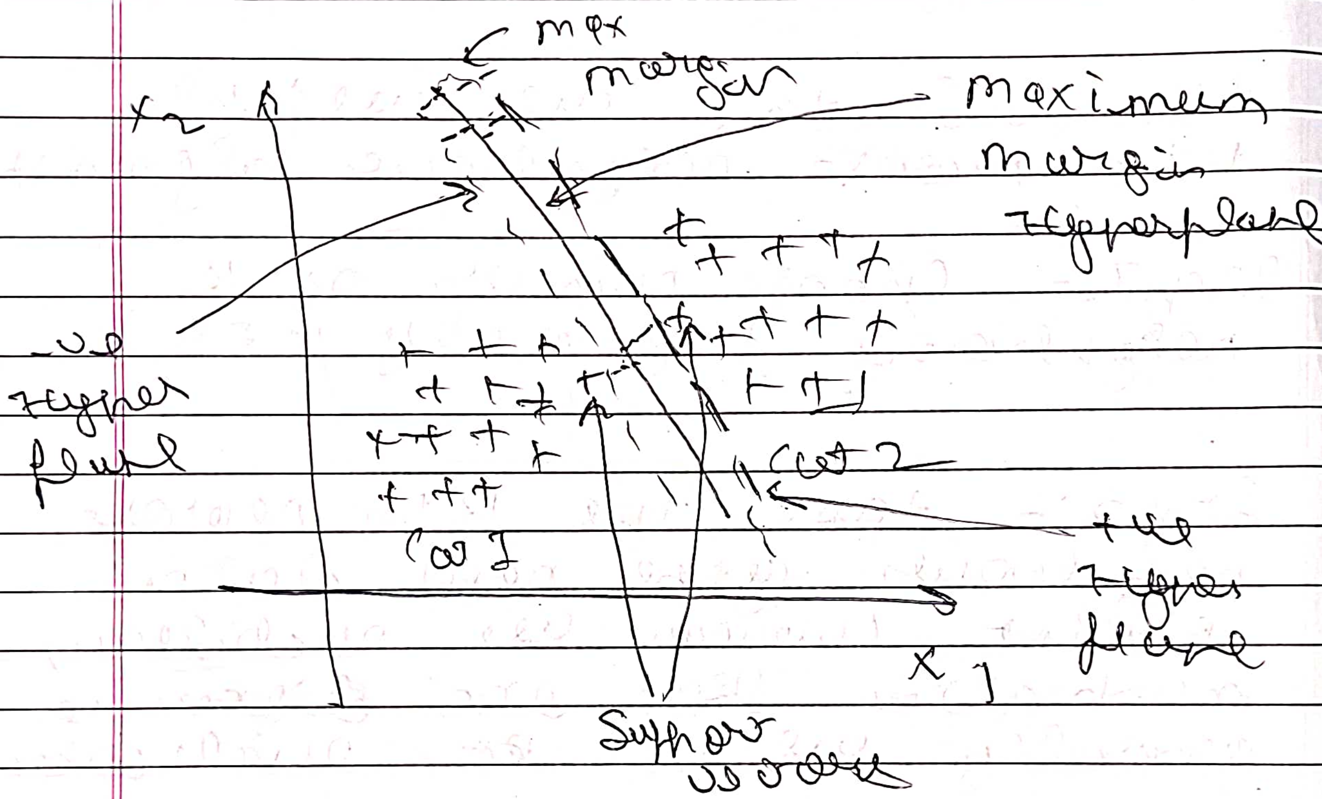
Step 2:- take the Kth nearest neighbours of the new data point (we can use euclidean, manhattan dist etc distance generally we use euclidean)

Step 3:- count no of ~~datapoints~~ data points in each categories from each of the neighbours.

Step 4:- assign the datapoint to the category which has the max count among the kth nearest neighbours

→ Support Vector Machine (Linear)

Suppose we have



The line is drawn equidistant from the support vector

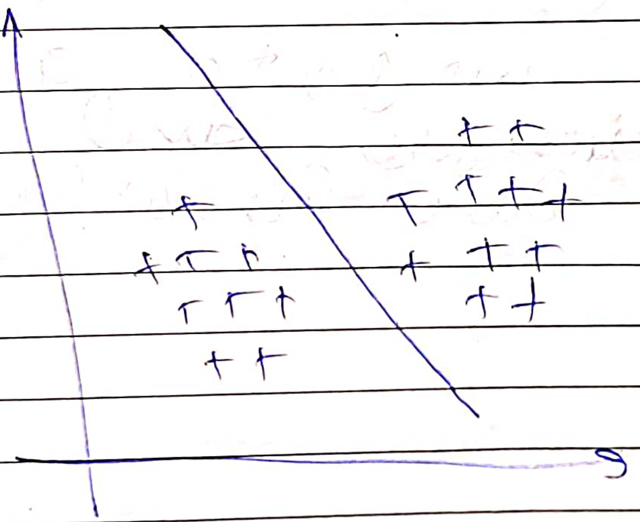
Anything to right of +ve hyperplane will be in cat 2

Anything to left of -ve hyperplane will be in cat 1

If we want to classify apples and oranges this will look at the most apple which looks more like an orange and orange which looks more like apple and make it the support vector and comparative analysis from there

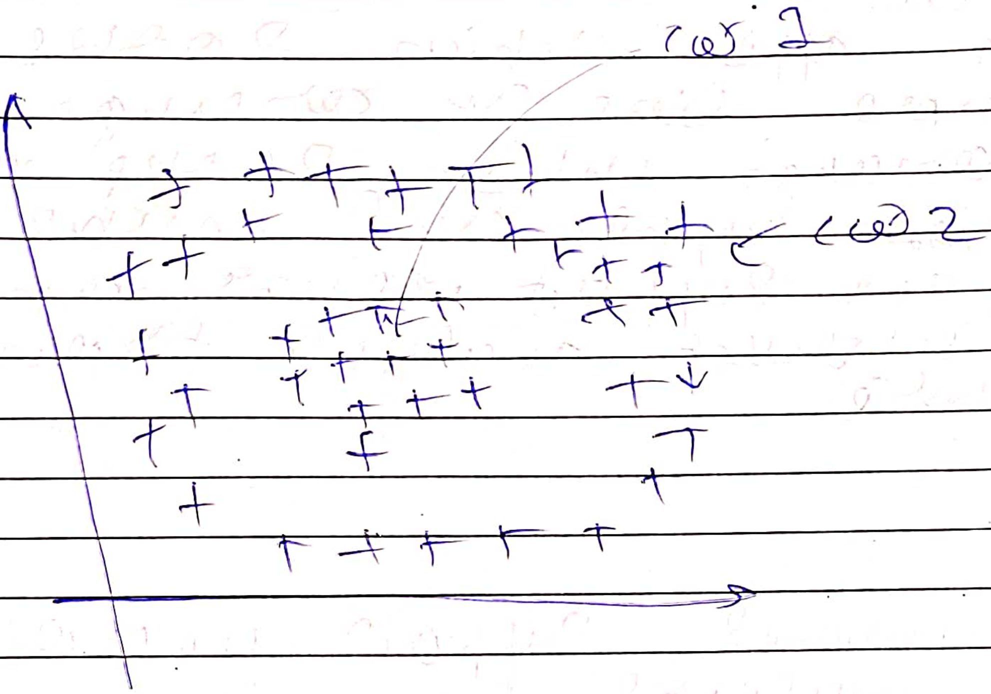
→ Kernel Support vector machine

→ for data points like these



we can easily find a boundary between them

However when we have data points like this



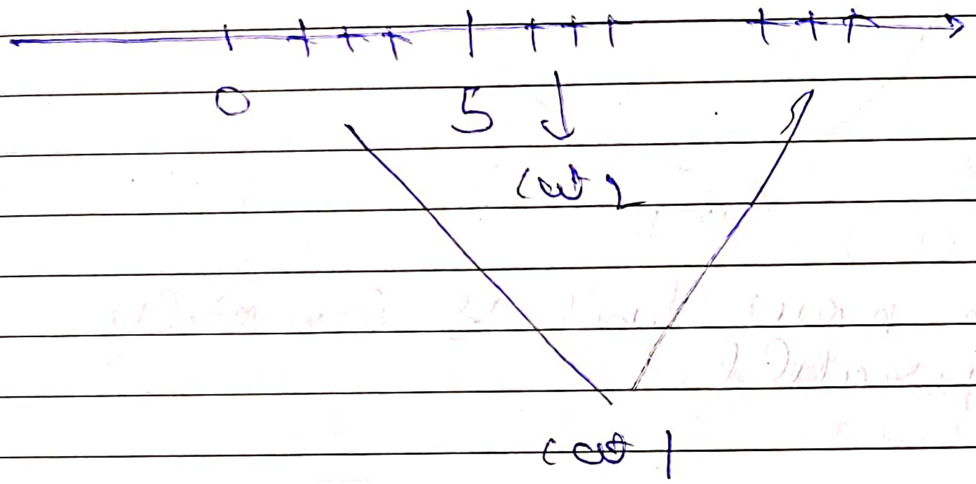
we cannot separate them linearly so we cannot use linear SVM to do this

To do this we go to a higher dimension and make this data linearly separable

→ mapping to Higher dimension

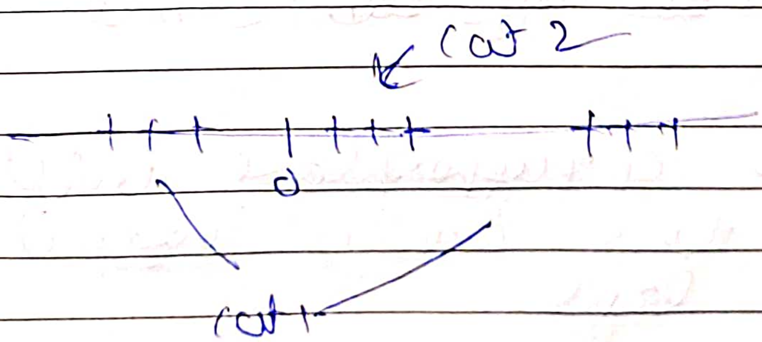
eg if we have

one Dimension Dataset

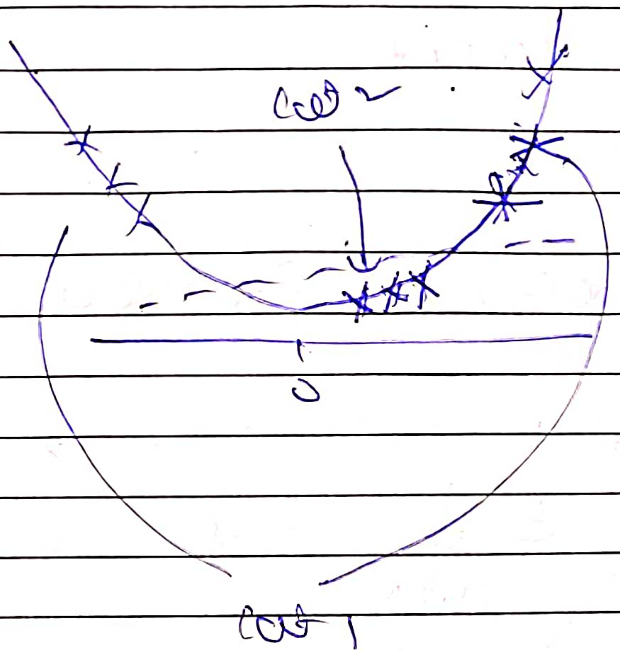


now if we do X−5

we get



if we now do $(x-5)^2$



∴ now this is linearly seperable

we do similar to if we have 2D dataset

~~mapping~~ mapping functions can be anything not always same

in 2D a Hyperplane will seperate the Data instead of a line

But mapping to higher dimension is very compute-intensive to avoid this we use the kernel trick

Gaussian / Radial Basis function

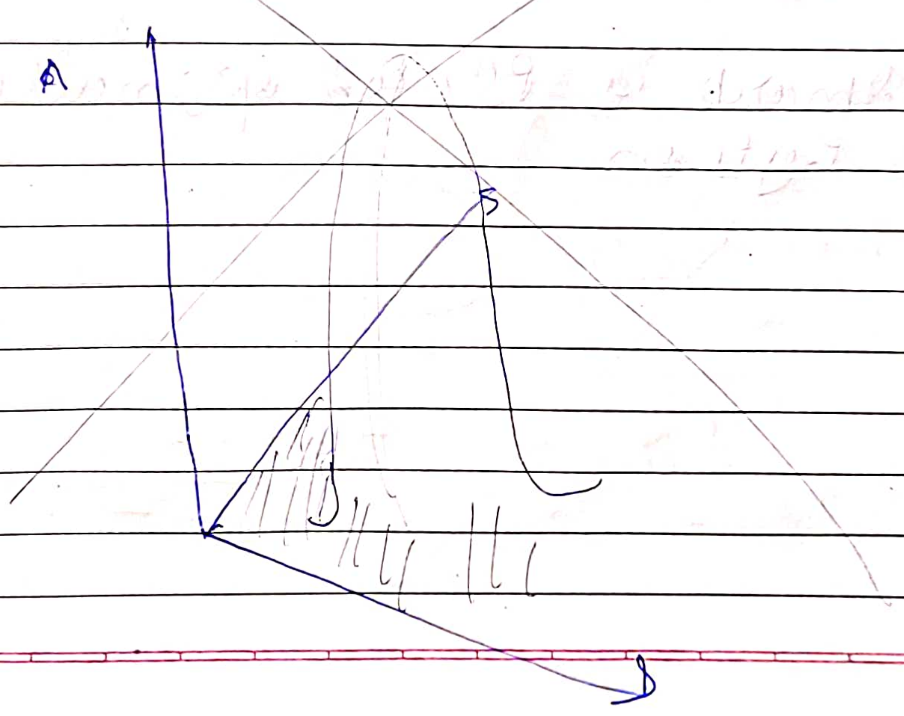
$$K(\vec{x}, \vec{l}^{\,i}) = e^{-\frac{||\vec{x} - \vec{l}^{\,i}||^2}{2\sigma^2}}$$

(Dis Between x and landmark)^2

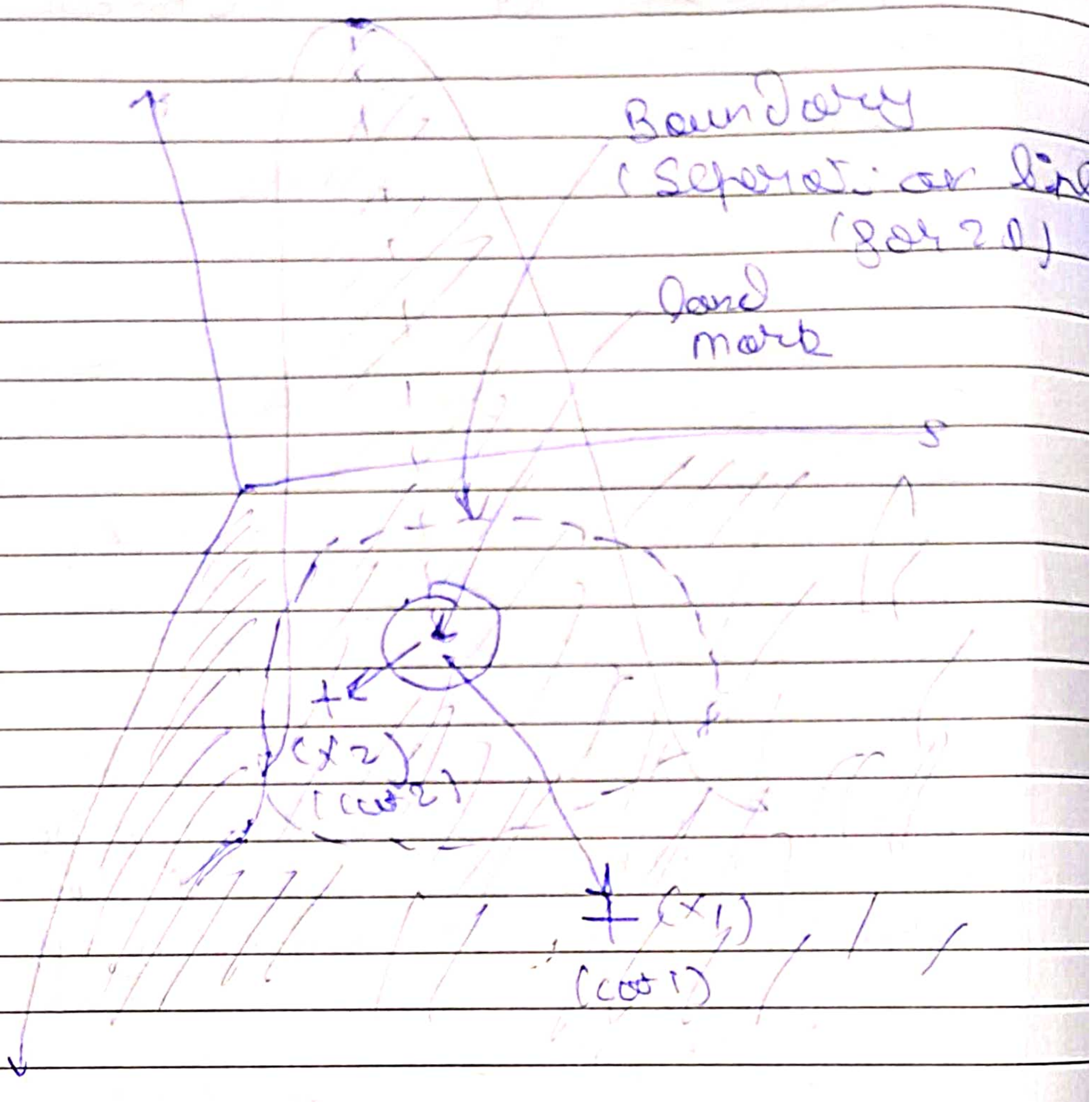
Point in dataset — landmark

[landmark is some fixed point and x varies]

Visually



visually



landmark is placed optimally
By Python

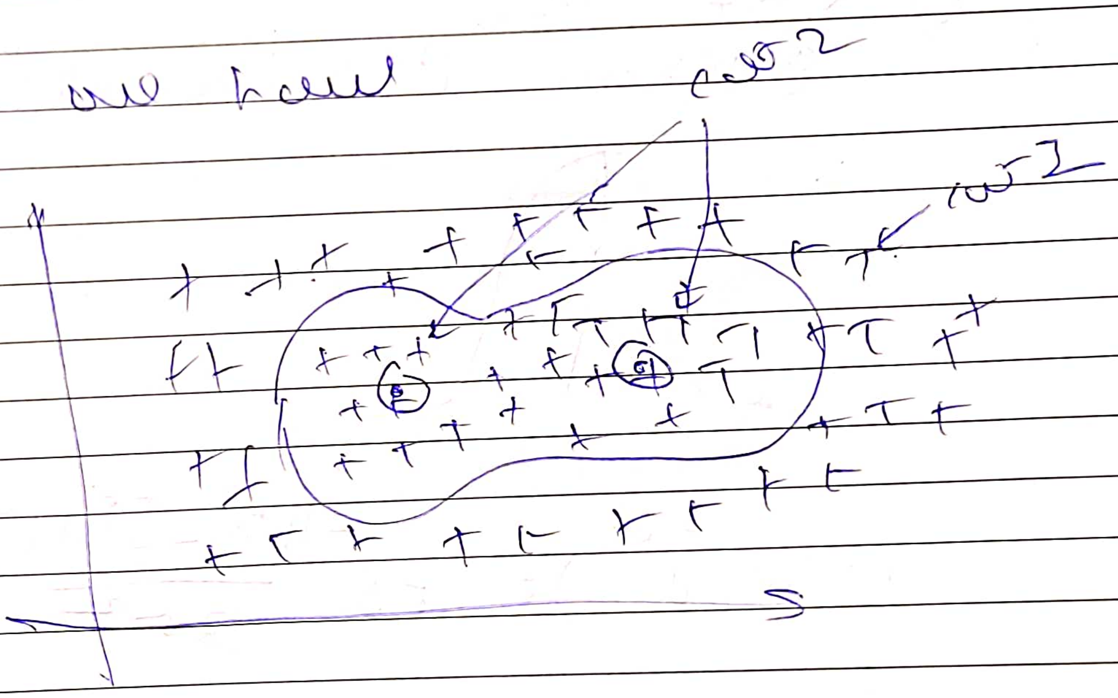
So anything within the circle will be in class 2 and beyond that it will be in class 1

σ is used to control the circumference of circle

(So this is how we avoid mapping to higher dimensions and use ~~this~~ this)

[Points ~~on~~ beyond circle get value close to zero and ~~beyond~~ in it will get values only in one thing as ~~we go~~ farther, we come close to landmark]

If we have



we can use 2 kernels

$K(\vec{x}, \vec{l^1}) + K(\vec{x}, \vec{l^2})$

case 2

$K(\vec{x}, \vec{l^1}) + K(\vec{x}, \vec{l^2}) > 0$

case 1

$K(\vec{x}, \vec{l^1}) + K(\vec{x}, \vec{l^2}) = 0$

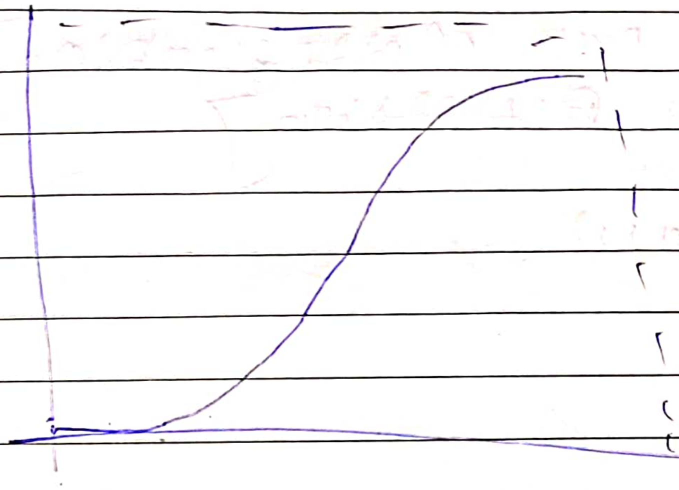
→ Types of kernel function

i) ~~RBF kern~~
gaussian RBF kernel

$$K(x, \vec{l^1}) = e^{\frac{-\|\vec{x} - \vec{l^1}\|^2}{2\sigma^2}}$$

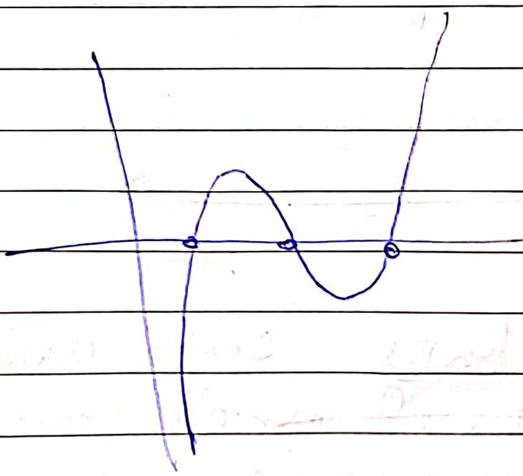
PAGE No.
DATE / /

2) Sigmoid kernel



$$K(X, Y) = \tanh(\gamma \cdot X^T Y + r)$$

3) Polynomial kernel



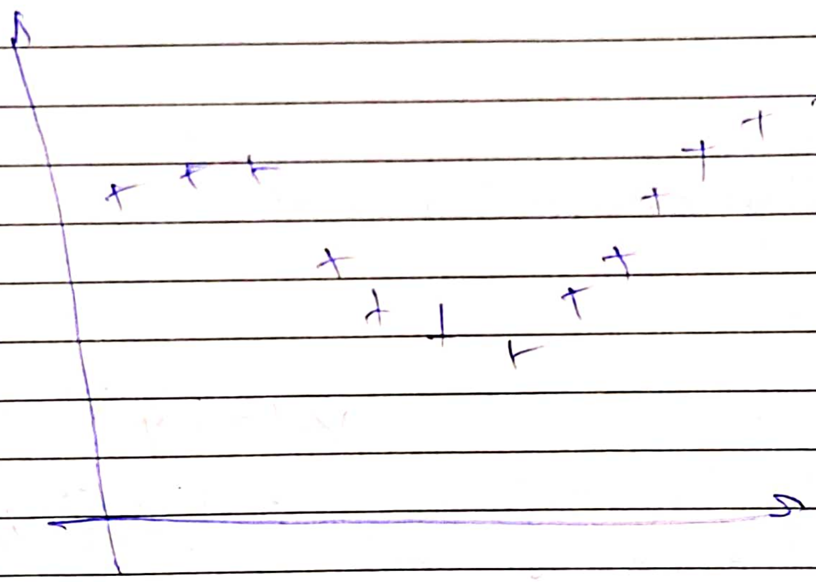
$$K(X, Y) = (\gamma \cdot X^T Y + r)^d,$$

$$\gamma > 0$$

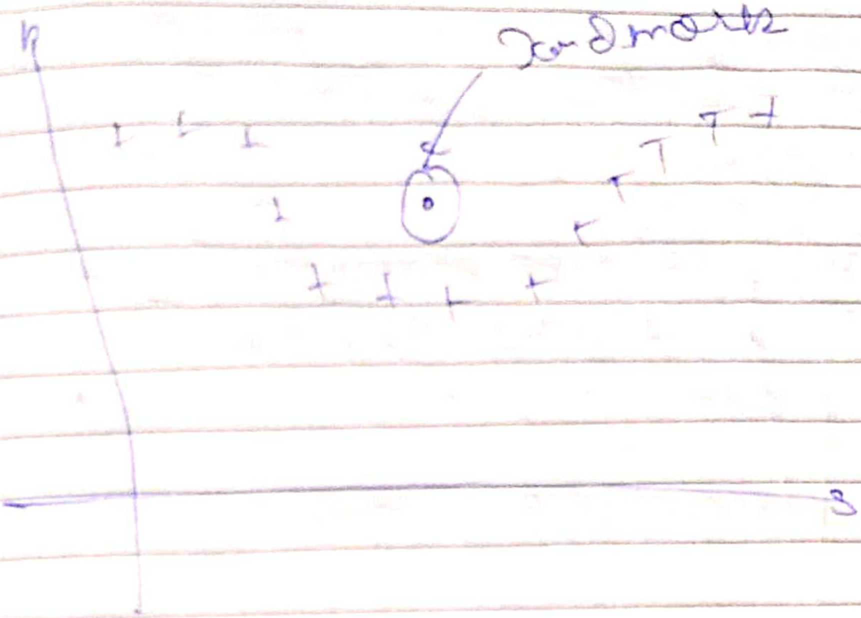
→ non linear Support Vector Regression

[this is not classification
this is Regression]

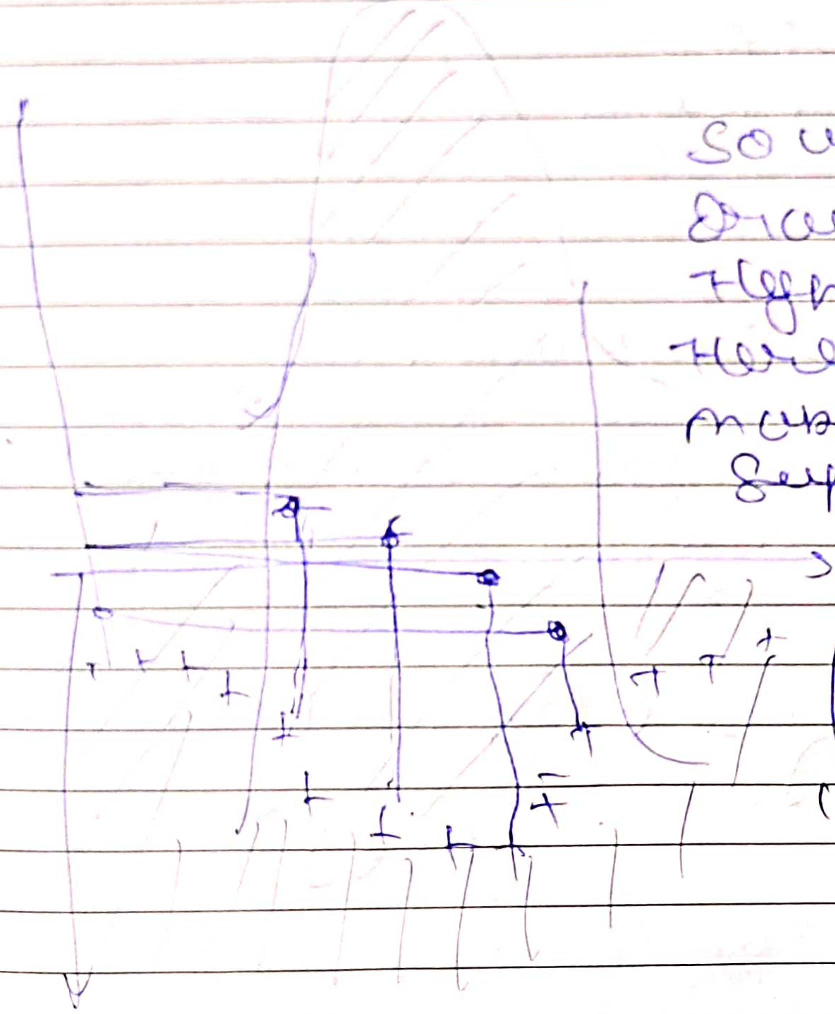
if we have



Data like this we will select a kernel and landmark and apply a kernel trick



now compute kernel trick



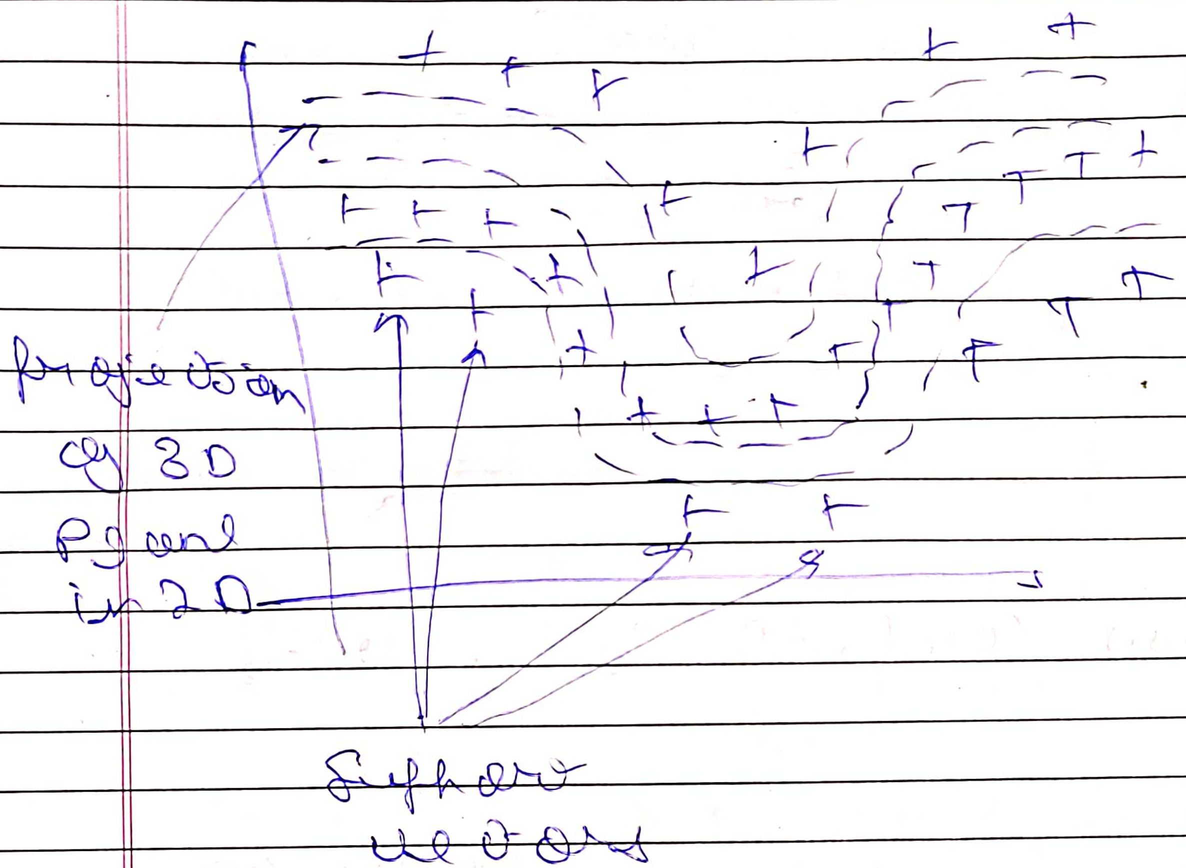
So we can draw a Hyperplane here and make our support vector

[we make 3 Hyperplane (like linear sum)]

PAGE No.

DATE / /

end Result



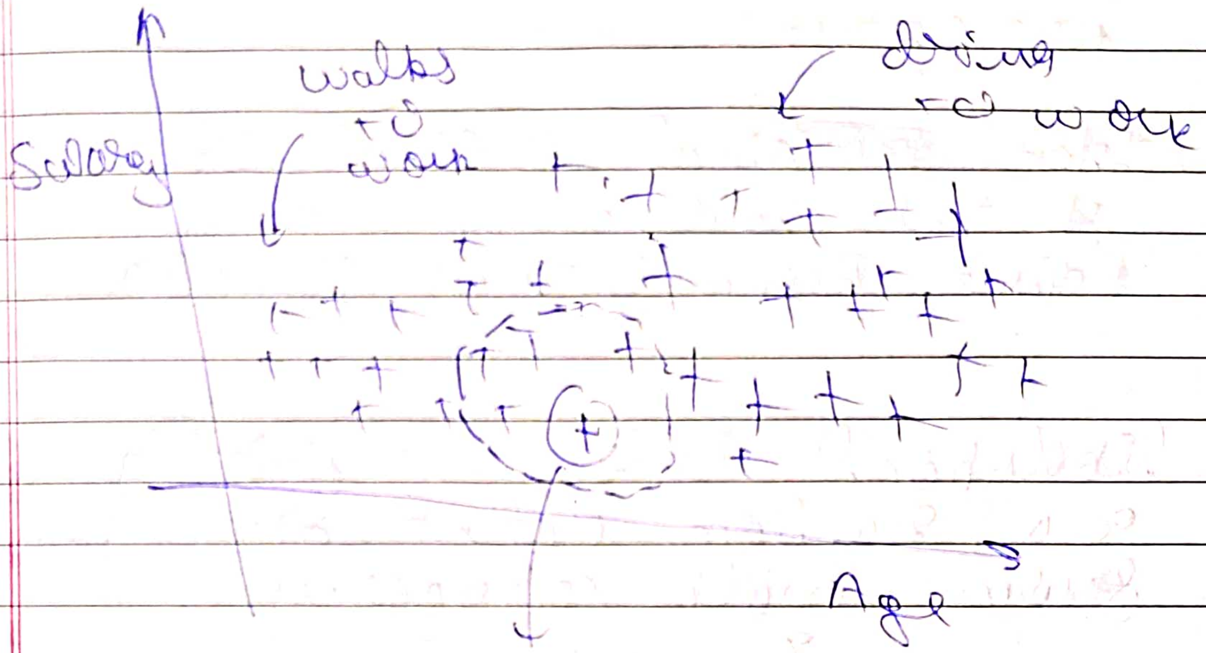
-> Bayes Theorem

used to find conditional probability

$$P(A/B) = \frac{P(B/A) \times P(A)}{P(B)}$$

→ Naive Bayes Classifier

Suppose we have Dataset



new Data Point
(age = 25
Salary = $3000)

Now we will calculate conditional probability of new Data Point for both walks and driving

$$\underbrace{P(walks/x)}_{\text{Posterior probab}} = \frac{\overbrace{P(x/walks)}^{\text{likelyhood}} \times \overbrace{P(walks)}^{\text{Prior Probab}}}{\underbrace{P(x)}_{\text{(marginal likelihood)}}}$$

we need to select a Radius around a data point (length is decided by us)

anything in the Radius will be considered similar to the point

then marginal probability is the probability of a point being in that circle

likelyhood it is the probability of a random point in from walk category (in this case) falls in the circle

So we calculate

P(walks/x) vs
P(drives/x)

and take the max and add it in that class.

PAGE No.
DATE / /

→ why is the naive Bayes ~~Bayes~~ : Because it relies on some assumptions that might not be correct, but we assume it to be correct

it requires the variables to be independent but age and salary might be dependent

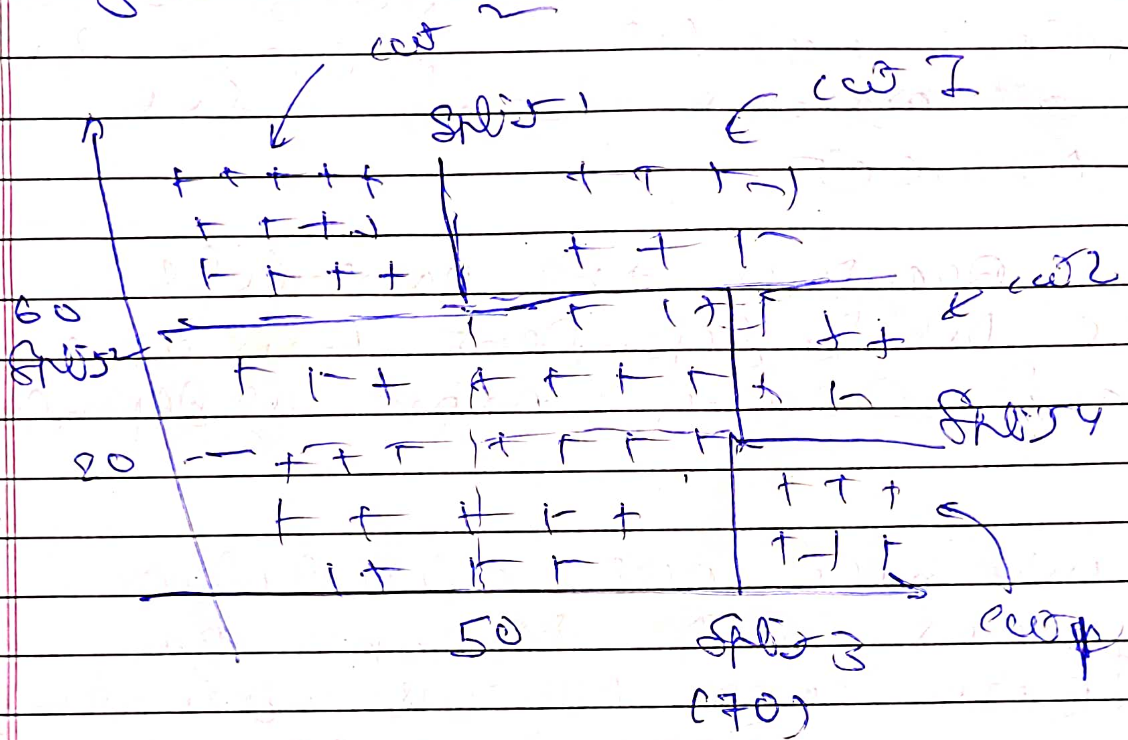
→ we need not calculate P(x) as it is same in both cases (priors and weights) [okay if we are comparing] marginal likelihood

However while calculating we need P(x)

→ when we have more than 2 features we need to calculate all and probability and take the max

→ Decision Tree Classification

if we have

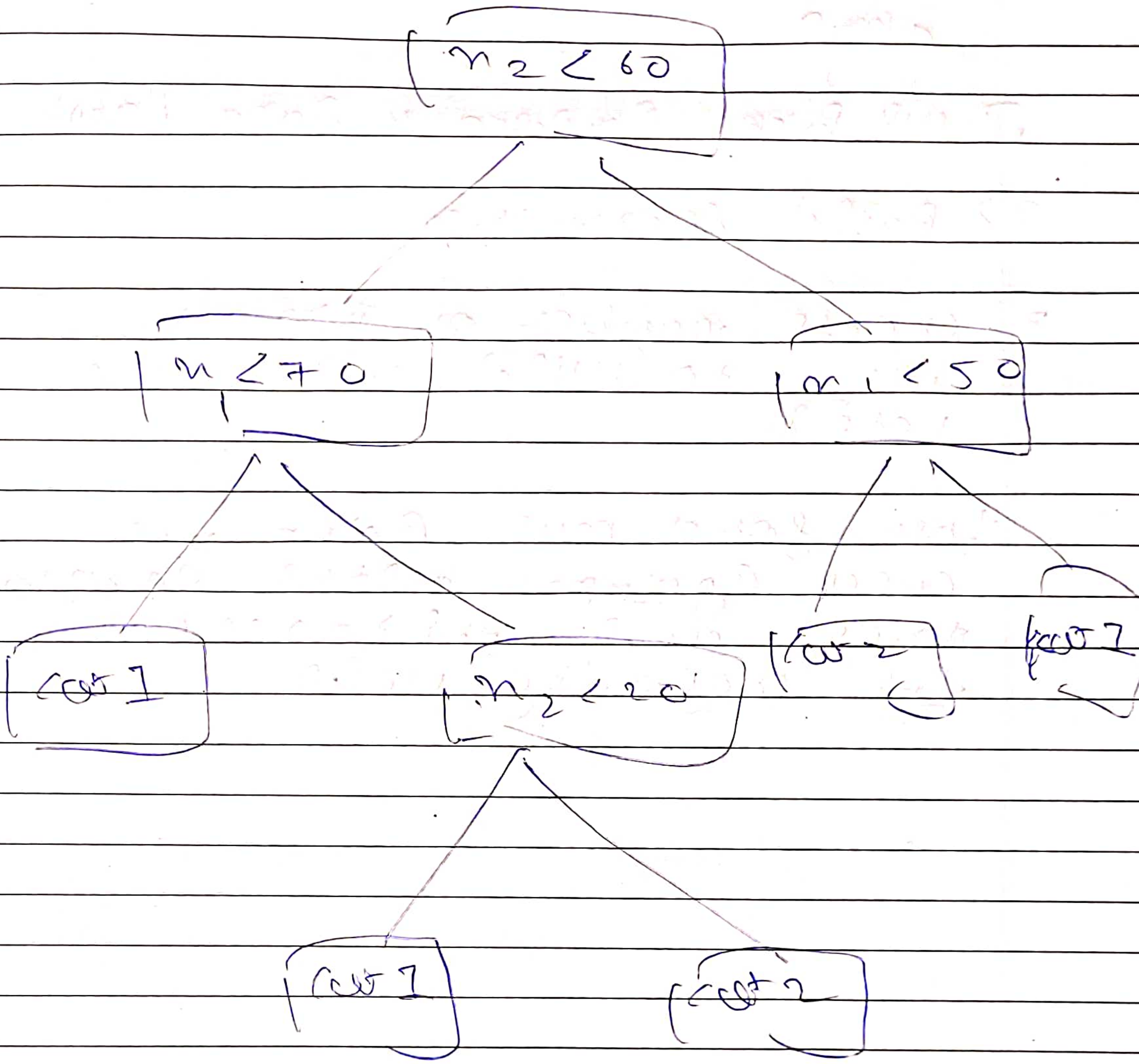


Splits are done in such a way that max of a category is present in any leaf

PAGE No.
DATE / /

Decision

Decision Tree



Sometimes we can't reach the Bottom so in that case we see what is the max probability of it being in cat 1 or 2 in non leaf node

→ Random forest classification

we use ensemble learning here

1) we pick random K data points

2) Build decision tree

3) choose number of N trees we to build and repeat 1 and 2

then for a new point we check against all the decision trees and put prediction the majority vote