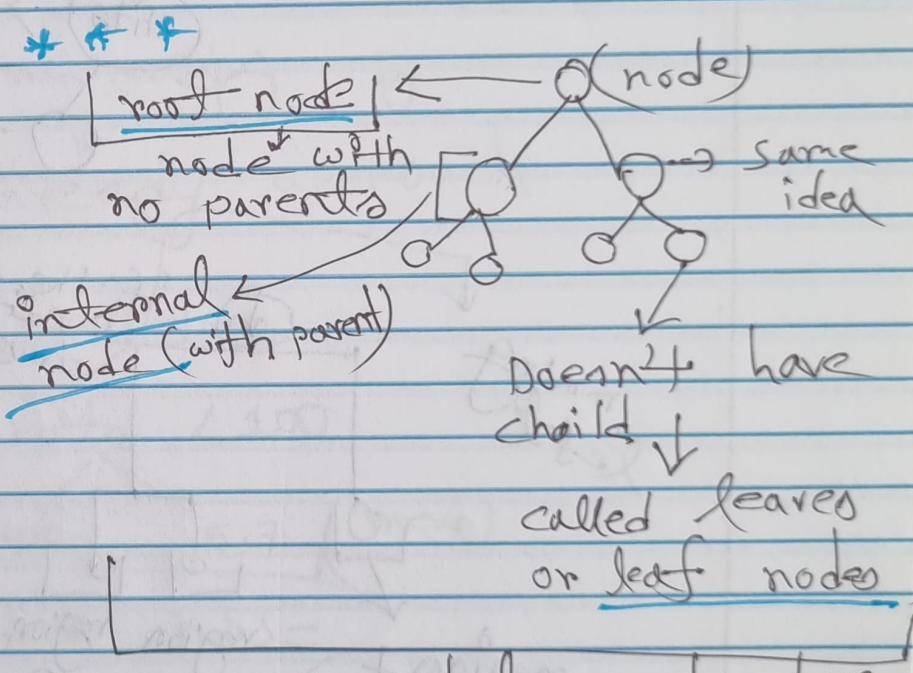
Lec: 15

Tree-Based Methods

Decision trees → relationship between Predictors and response

Tree-based methods
↓
can be applied both in linear and classification

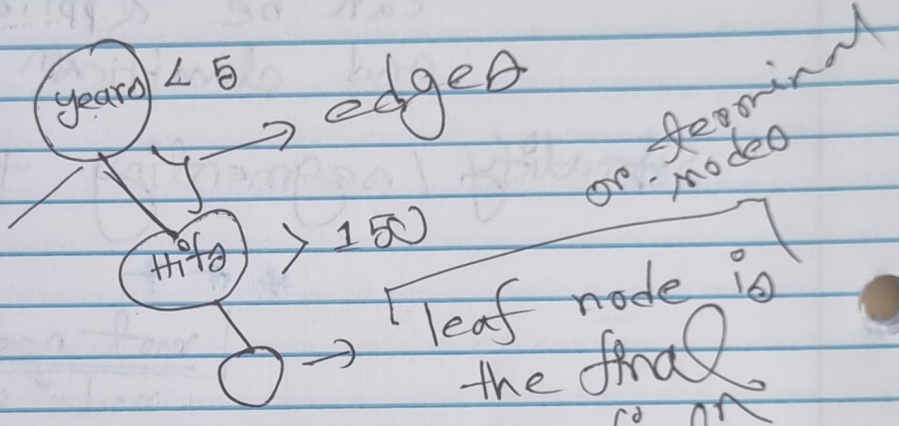
Stratify / segmenting the predictors

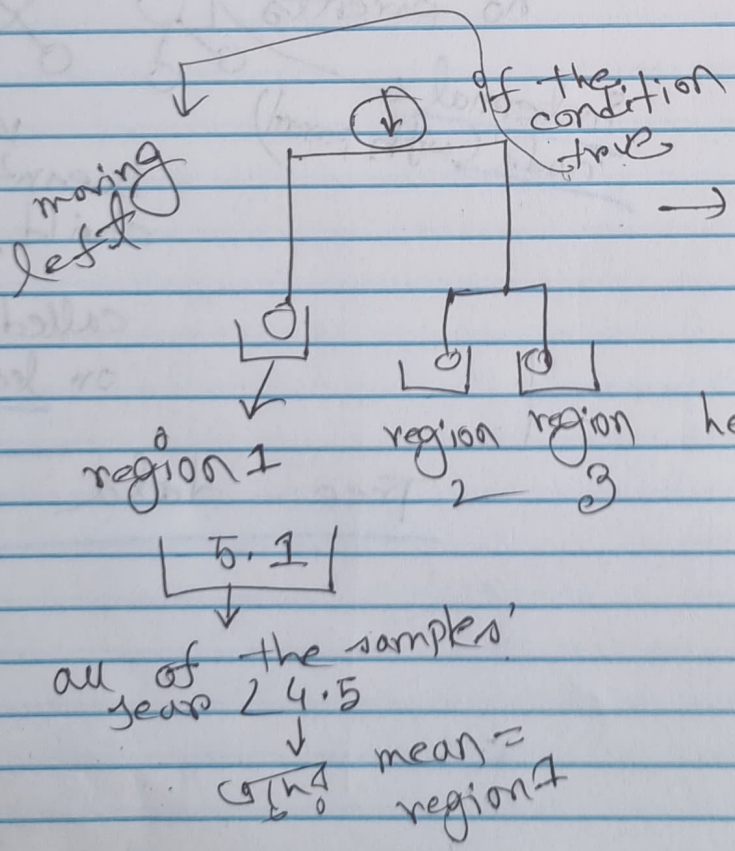


Tree-data structure

≠ Lower the num of years
↓
lower hits
↓
Lower salary

hits + years → features





→ 2 internal nodes
leaf nodes → 3
height = from root to the leaf distance

separating predictors space
↓
called decision tree

How to build?

Step 1 : non-overlapping region of Pr

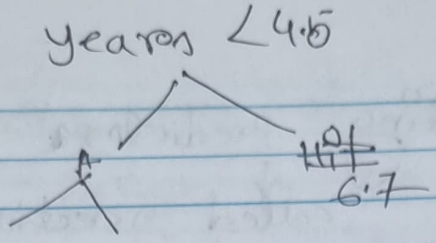
" 2 : Sample mean in every space region

" 3 : MSE
↓
minimizing by
↓
Greedy approach

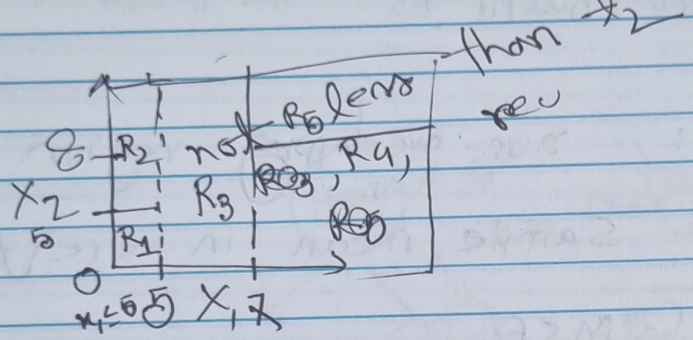
#① selecting predictors
↓
Hits < 100
↙ ↘
4.0 5.7 (mean)

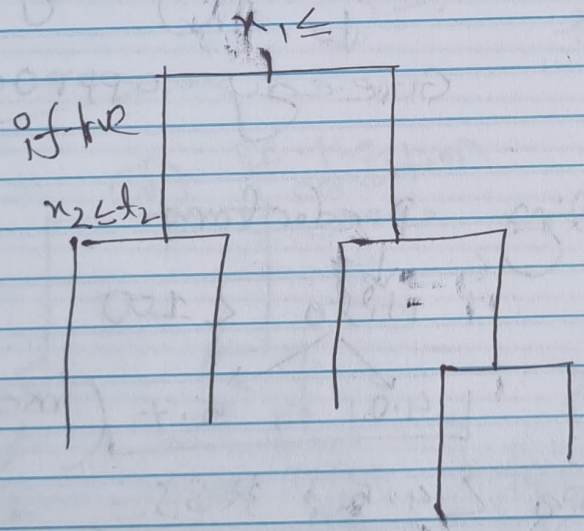
years < 4.5 RSS=
^
5.2 6.5

RSS = → based on lowest num
↓
root node
selection



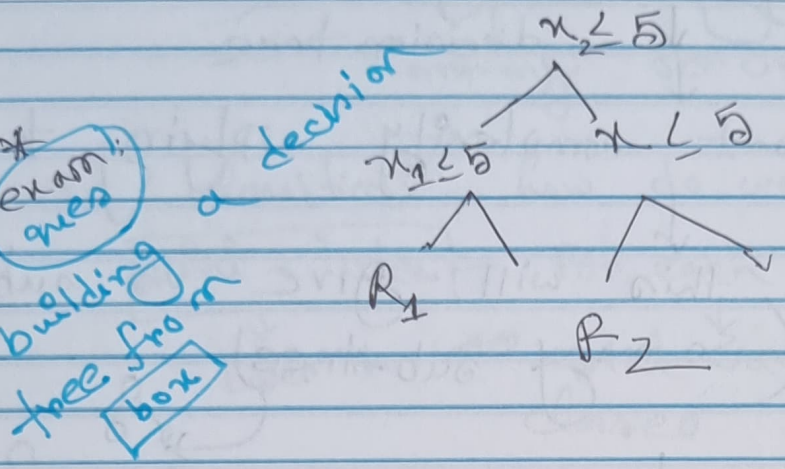
Recursive process





$t_1 = 5$
$t_2 = 5$
$t_3 = 7$
$t_4 = 8$
$t_5 = 10$

root node = 0



* exam: ques
a decision
building a tree from box

Pruning a tree: cutting the unwanted parts

cost complexity pruning :→ or weakest link pruning

T = Height
= 50

$\alpha = 0.01$
need to be tested

During cutting → too much pruning
↓
useful branch can be lost

optimal α value using cross-validation

training very big model (50 predictors)

↓ decision tree

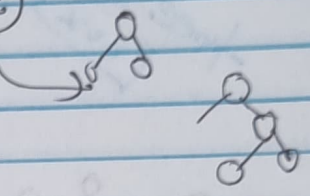
↓

cost complexity applying to pruning

↓

this will give us subset

(so many sub-trees)



↓

Then cross-validation

on each and individual tree

↓

best one with the minimum error

Baseball example → 3 is good

instead of 10

Lec: 16
03-13-24

classificatn tree
↓
commonly occurring names

In classificatn, how do we calculate errors?

Gini - index → for each internal node
↓ ↓
Lower the better Total variance across all the classes

purity of node

root node
↓
↑↑ Gini - index
↓
but at the bottom of decision tree
↓
Gini index ↓↓

cross - entropy :

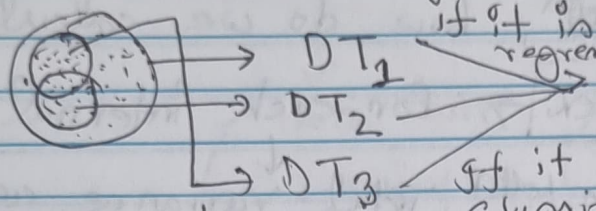
Gini index → underfitting → ↑↑ value ↑↑ like root node
→ overfitting
↓
too low (overconfidently)
yes / no

Disadva → not much accurate

advantage: → easy to explain / understand
→ " graphically representatn + easy interpretatn
→ can easily handle qualitative predictors

Bagging decision tree :

Bagging: B = 3 sample randomly from the dataset 3 times



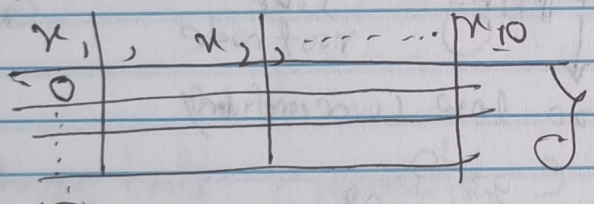
we are reducing the variance + Good

Problem → the way we are sampling

↓

repeating samples in different DT →

3 features:



random sampling with replacement

0, 1, 2, 3, 4 → DT1

0, 2, 4, 5, 6 → DT2

In the bag ↓ repeating sampling

↓

can't reduce the variance

0, 1, 2, 3, 4

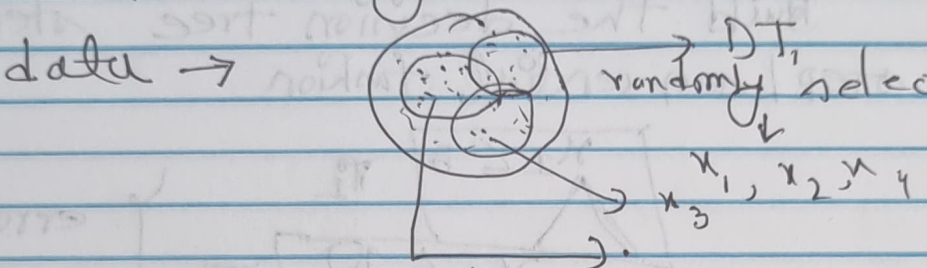
Others than this sampling called out of bag

Improving bagging:

Random forests:

wanna handle randomness which is common in bagging ↓

→ introducing randomness



though repeating occurs, but they are unique

random sampling and features } creating forests

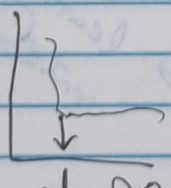
16 predictors : $\sqrt{16} = 4$

$m \approx 4$

small trees coming together to build a forest

↓

error is minimized in this than the bagging and other method



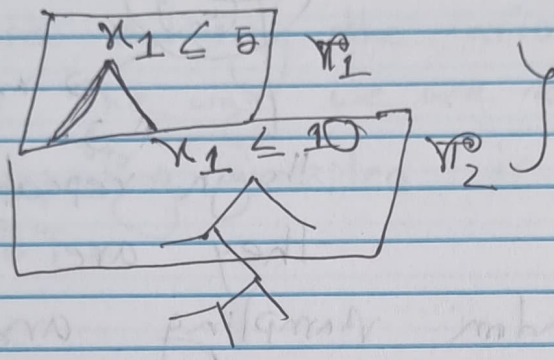
at 90 error is minimum

90 trees will be better than the 300-trees

disadvan.

Boosting:

Build the decision tree step by step / sequential fashion



error difference r_i should be less

Bagging + random for

(i) → number of trees (by giving different values of B by cross-validation)

(ii) Shrinkage parameter in boosting controls how boosting learns

small positive numbers

learning rate

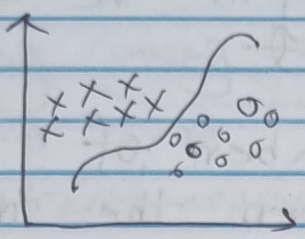
(iv) number of opts

Final exam
- 20 – Multiple question
- 2 → case study → per 5

Lec: 17
04-01-24

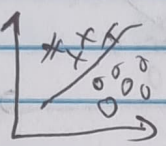
"Support vector machine"

2 class / 2 labels within the dataset



logistic regression for seperating the data

But in SVM → seperating by using a line

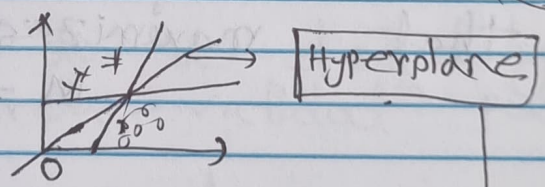


If gets tough to separate

we will use SVM

feature expansion → polynomial lines to separate the data

Hyperplane:



coefficient doesn't matter

From many, which one can we select?

criteria for selecting best hyperplane:

1) Maximal Margin classifier

i) Maximal Margin classifier:

finds all possible hyperplanes

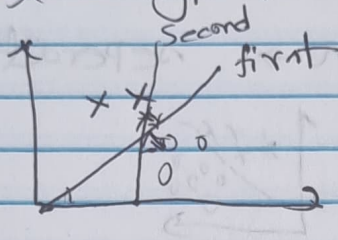
↓

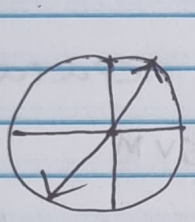
depending on the size of the dataset

→ create k number of hyperplanes

margin between the immediate sample

→ hyperplan



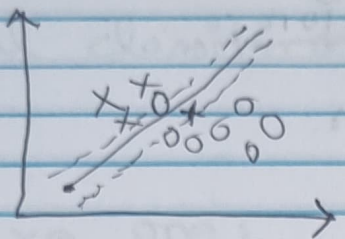


→ select the best hyperplane that maximizes M

M = Largest Margin

cons: It can't be used if the data isn't perfectly seperable.

ii) Support vector classifier :



allowing (wrong samples) in the restricted area → SVM

C - regularized/parameter :

if C ↑↑ → we are giving more space for the wrong samples / errors

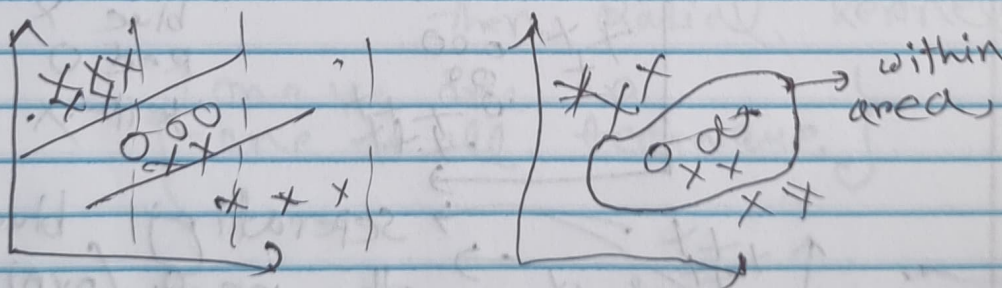
if C ↓↓ → ↓↓ error

$\epsilon_i \leq C$

Tradeoff : if C ↑↑ → error ↑↑ but model will be robust

but if C ↓↓ → error ↓↓ but model will be less robust

so need to be checked by checking with the cross-validation parameter



feature - expansion:

amplifying the predictors.

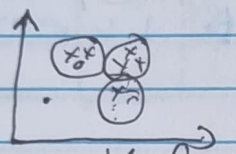
③ SVM:

kernels = hyperplane function

nonlinearities and kernels:

Radial kernel:

overfitting may occur:



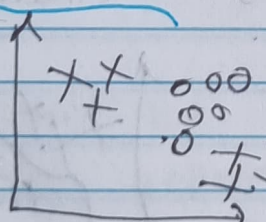
within radial kernel → α-value is for controlling the parameters

SVM → works for 2 classes:

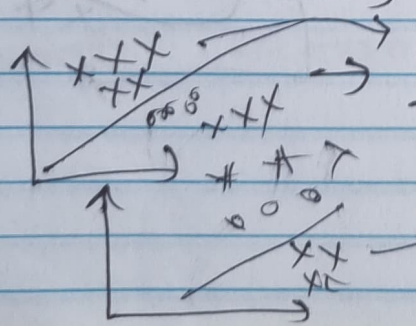
$k=2$

how can we extend it?

a) one vs all:



y
blue X
Red O
yellow X



seperating blue from the rest (OVA)

seperating red form the rest

└ yellow

final classification based on majority:

b) one vs one:

one vs others
Blue vs Red
Blue vs yellow
Red vs yellow

* combination will take time → in case of bigger sample.

separation:

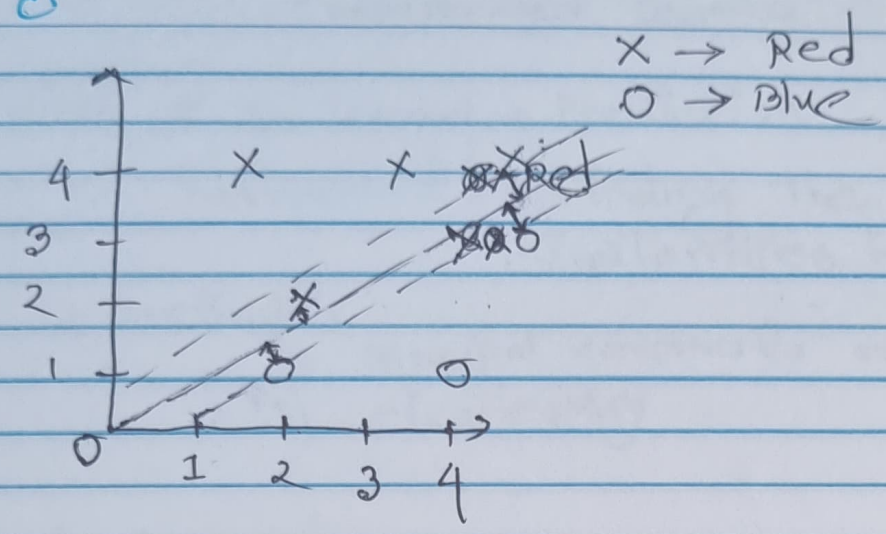
Summary:

Maximal margin → if perfect is necessary
↓
if not, then allowing for errors
↓
by SV classifiers
controlling margin with C-
Samples that are on the edges

{non-linear} → SVM → kernel
↓
then Radial kernel

{one vs one or one vs all technique}

class Problem:



① sketch the optimal seperating hyperplane

② Indicate the SV