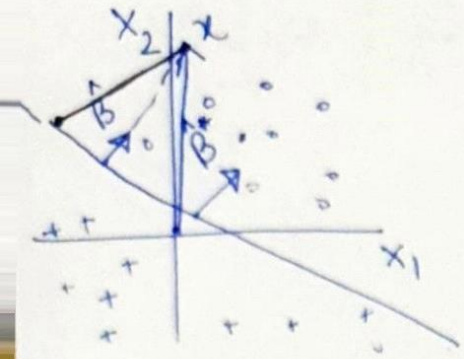
WEEK-4 - Lecture-20 - Perceptron Learning

Modelling a seperating hyperplane



(intercept)

$$f(x) = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 = 0 = L$$

$$= \beta_0 + \beta^T x = 0$$

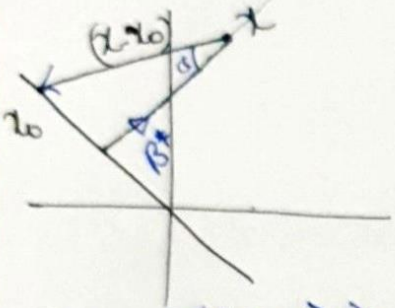
Properties:

1) $\beta^T(x_1 - x_2) = 0 \Rightarrow \beta^* = \frac{\beta}{\|\beta\|}$

2) $\beta^T x_0 = -\beta_0 \quad \forall x_0 \in L \Rightarrow$

3) $\beta^{*T}(x - x_0) = (\beta^{*T} x - \beta^{*T} x_0)$

$$= \frac{1}{\|\beta\|}(\beta^T x + \beta_0) = \frac{f(x)}{\|\beta\|}$$



$$\cos\theta = \frac{\vec{a}\cdot\vec{b}}{\|a\|\|b\|}$$

$$f'(x) = \beta$$

$$= \frac{f(x)}{\|f'(x)\|}$$

$$\frac{(x-x_0)\cdot\beta^*}{\|\beta^*\|\cdot\|x-x_0\|} = \cos\theta$$

Distance of x from hyperplane

signed distance

Perceptron Learning Algorithm

Minimize distance of misclassified points to decision boundary.

Assumption → $[y_i = 1 \Rightarrow x_i^T\beta + \beta_0 > 0]$ True classification

$y_i = 1$, is misclassified $\Rightarrow x_i^T\beta + \beta_0 < 0$

$y_i = -1$, is misclassified $\Rightarrow x_i^T\beta + \beta_0 > 0$

$$y_i(x_i^T\beta + \beta_0) \begin{cases} < 0, & y_i = 1 \\ > 0, & y_i = -1 \end{cases} \text{ Misclassified}$$

$$-y_i(x_i^T\beta + \beta_0) \begin{cases} > 0, & y_i = 1 \\ > 0, & y_i = -1 \end{cases} \text{ for sample } (x_i)$$

For all samples

$$D = -\sum y_i(x_i^T\beta + \beta_0) \rightarrow \text{Minimise}$$

[1]

Lecture-20

$D(\beta, \beta_0) = -\sum_{i \in M} y_i (x_i^T \beta + \beta_0)$, (M - samples are misclassified)

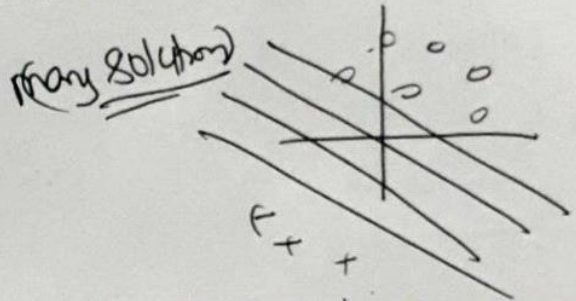
(Minimize $M \to 0$) our motive

$$\frac{\partial D}{\partial \beta} = -\sum_i y_i (x_i), \quad \frac{\partial D}{\partial \beta_0} = \sum_i y_i$$

$$\begin{pmatrix} \beta \\ \beta_0 \end{pmatrix} = \begin{pmatrix} \beta \\ \beta_0 \end{pmatrix} - (\alpha)\left(-\begin{pmatrix} \sum_i (y_i x_i) \\ \sum y_i \end{pmatrix}\right)$$ → (vectors)

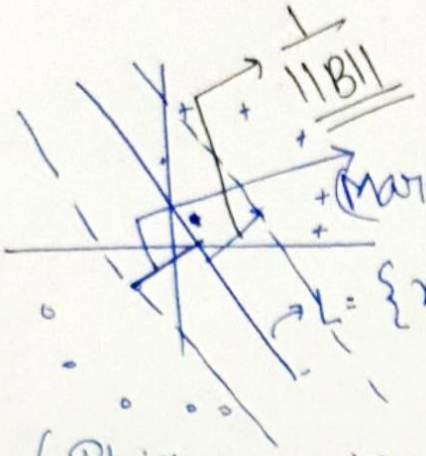
(problems)

- Linearly seperable: then Converge to some solutions.
- Can take along time if gap bet two classes are very less



- Not linearly seperable → it enters in (loop) (∞)

SVM FORMULATION Lecture-21



$f(x) = \beta_0 + \beta^T x$

Optimal Seperating hyperplane
Maximize the distance of closest point to hyperplane.

(Thickness should be same For both hyperplanes)

β_0, β_1, max M

$(||\beta||=1)$ ↓ β, should n't be larger.

subject to $y_i(x_i^T\beta + \beta_0) \geq M$ for all $i = 1, \cdots N$

↓ We need to maximise M, so that there is a maximum seperation between classes.

distance of any x_i from hyperplane

$$\frac{f(x)}{||f'(x)||} = \frac{(x_i^T\beta + \beta_0)}{||\beta||}$$

→ signed distance

↘ multiply by y_i to make positive

↙ Need n't worry about $||\beta||=1$

$$\frac{y_i(x_i^T\beta + \beta_0)}{||\beta||} \geq M \Rightarrow y_i(x_i^T\beta + \beta_0) \geq M||\beta||$$

Just make sure $||\beta|| = 1/M$ only change in direction.

$$\boxed{y_i(x_i^T\beta + \beta_0) \geq 1}$$

Max M ↓ Min β

$$\left(\min ||\beta|| \text{ s.t } y_i(x_i^T\beta + \beta_0) \geq 1\right)$$

[1]

SVM - interpretation & Analysis

s.t $y_i(x_i^T\beta+\beta_0) \geq 1$ ~~max~~ $\min \|\beta\| \Rightarrow \left\{\min \frac{1}{2}\|\beta\|^2\right\}$ by our choice so that it is easily differentiable.

Lagrangian:

$$L_P = \frac{1}{2}\|\beta\|^2 - \sum_{i=1}^{N}\alpha_i\left[y_i(x_i^T\beta+\beta_0)-1\right]$$

Setting derivatives to '0'.

$$\frac{\partial L_P}{\partial \beta} = 0 \Rightarrow \frac{1}{2}\cdot 2\cdot\beta - \sum_{i=1}^{N}\alpha_i[y_i x_i] = 0 \quad -(1)$$

$$\beta = \sum_{i=1}^{N}\alpha_i x_i y_i$$

$$\frac{\partial L_P}{\partial \beta_0} = 0 - \sum_{i=1}^{N}\alpha_i y_i = 0 \quad -(2)$$

Dual:

$$L_D := \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{k=1}^{N}\alpha_i\alpha_k y_i y_k x_i^T x_k \quad -(3)$$

Subject to $\alpha_i \geq 0$, $\forall i$

$\alpha_i\left[y_i(x_i^T\beta+\beta_0)-1\right]=0 \quad \forall i \quad -(4)$

$\alpha_i = 0$ For non-marginal points → which are very far from hyperplane.
↓
will not contribute to calculating 'β'.

$\alpha_i \neq 0$, x_i is on the margin. → (support vectors)
↳ only points on the margin plane contribute to β.
↓ points which lies on margin plane.

Calculate β_0 from here.

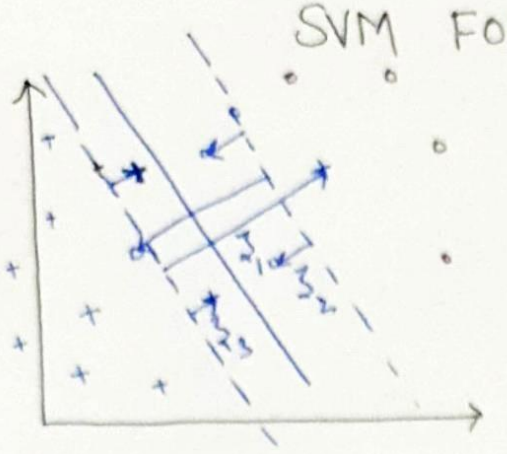
$f(x) = x^T\hat{\beta} + \hat{\beta_0}$
$= \sum\alpha_i y_i x_i^T x + \beta_0$

SVM is stable
↓
less variance

points outside the margin will not affect the SVM. only support vectors will affect SVM.

Lecture-23

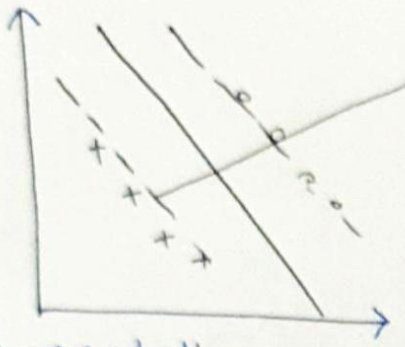
SVM FOR linear Non-seperable data



1. Minimize these distances.
   ξ_1, ξ_2, ξ_3 → miselassified samples

Qe? why not minimize the maximum distance rather than minimizing sum of distances.

Ans: classifier will try to minimize the noise, and hyperplane will get shifted towards Noise like.



Formulation:

$$y_i(x_i^T\beta + \beta_0) \geq M(1-\xi_i)$$ ← slack variable

1. Ideally we want to maximise M, and $\xi_i = 0$
2. but I want some sort of relaxation in maximising M.
   why not $(M-\xi_i)$ Ans: Non-Convex optimization problem.

ξ_i: ~~what~~ fraction of margin (M)

① $\forall_i \ \xi_i \geq 0$, ② $\sum_{i=1}^{N} \xi_i \leq$ Constant
↑ we don't want ξ_i to be very large.

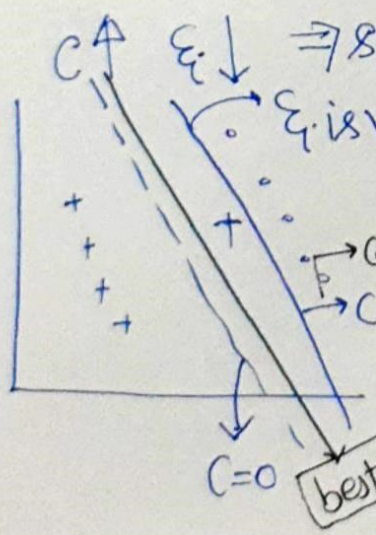
$$\min \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{N}\xi_i, \text{ Sub. to } y_i(x_i^T\beta+\beta_0) \geq (1-\xi_i)$$

since $M = \frac{1}{\|\beta\|}$ like the previous formulation

C ↑ ξ_i ↓ ⇒ smaller margin. $\xi_i \geq 0$

ξ_i is very small. → less robust

Desirable ξ_i should be more.



→ Can capture Noise.
→ $C \to \infty$

C should be less.

tradoff between increasing C and ξ_i.

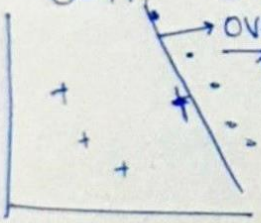
$C \uparrow \xi_i \downarrow$, $C \downarrow \xi_i \uparrow$

NOTE:

introducing ξ_i makes SVM more stable and less prone to noisy data. Suppose: we put $\xi_i = 0$, then we will try to maximize M. as shown in figure.

→ we should provide some slackness, because M is calculated via support vectors which lies on the boundary. It doesn't care about the points which lies within the margin. so, we need to provide some slackness in variable M. by introducing ξ_i which is distance of points which are within the margin from margin plane. If ~~C~~ ~~make~~ ~~ξ~~ we need sum of ξ_i ~~ξ_i~~ to be constant so it shouldn't overshoot.

If we make $C = \infty$, it means all ξ_i are very small, since classifier will try to minimize loss, in doing so, it has to make all $\xi_i = 0$, ⇒ ~~M~~ ~~try~~ margin will be very small. overfitting will capture Noise also as shown below



If $C = 0$, ξ_i will be larger, more biased SVM classifier.

underfitting.

Lecture-24 SVM KERNELS

$f(x) = x^T\beta + \beta_0 \Rightarrow \sum_{i=1}^{N} \alpha_i y_i \boxed{x_i^T x} + \beta_0$

$\boxed{x_i^T x}$ → inner product.

is there any good way to compute inner product.

$x \rightarrow \boxed{h(x)}$ "Transformation on x"

$L_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \langle h(x_i), h(x_j)\rangle$

similarily

$f(x) = \sum_{i=1}^{N} \alpha_i y_i \boxed{\langle h(x), h(x_i)\rangle} + \beta_0$

$\langle h(x), h(x_i)\rangle = K(x, x_i)$ – similarity measures

Kernel – symmetric $x^T A x \geq 0$

(+ve) semi definite ↓ semi definite

"popular choice of K"

poly : $(1 + \langle x, x'\rangle)^d$

RBF : $\exp(-\gamma \|x - x'\|^2)$

ANN : $\tanh(K_1\langle x, x'\rangle + K_2)$

K_1 → Constant, K_2 → Constant

poly

$(1 + \langle x_i, x\rangle)^d = (1 + x_1 x_1' + x_2 x_2')^2, \quad d = 2$

$= 1 + \underbrace{2x_1 x_1'}_{2} + \underbrace{2x_2 x_2'}_{3} + \underbrace{(x_1 x_1')^2}_{4} + \underbrace{(x_2 x_2')^2}_{5} + \underbrace{2x_1 x_1' x_2 x_2'}_{6}$

$\boxed{X = (x_1, x_2)}$

$h_1(x) = 1, \quad h_2(x) = \sqrt{2}x_1, \quad h_3(x) = \sqrt{2}x_2$

$h_4(x) = x_1^2, \quad h_5(x) = x_2^2, \quad h_6(x) = \sqrt{2}x_1 x_2$

(6-dimension space)

Primal

Lecture-23

$$L_P = \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\alpha_i[y_i(x_i^T\beta+\beta_0)-(1-\xi_i)] - \sum_{i=1}^{N}\mu_i\xi_i \qquad \alpha_i,\mu_i \geq 0$$

Setting derivatives to '0'.

$$\beta = \sum \alpha_i y_i x_i \quad (1) \qquad 0 = \sum_{i=1}^{N}\alpha_i y_i \quad (2)$$

$$\alpha_i = C - \mu_i \quad (3)$$

we don't Need $\sum \xi_i \leq$ Constant

↓

It's taken Care by 'C' and optimization equation

Dual:

$$L_D = \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j x_i^T x_j$$

Subject to $\boxed{0 \leq \alpha_i \leq C < \sum_{i=1}^{N}\alpha_i y_i = 0}$

KKT Condition

$$C\xi_i = 0, \quad (4) \qquad \alpha_i[y_i(x_i^T\beta+\beta_0)-(1-\xi_i)] \geq 0, \quad (5) \qquad \mu_i\xi_i = 0 \quad (6)$$

$y_i(x_i^T\beta+\beta) - (1-\xi_i) \geq 0$ (1) If $\alpha_i = 0 \Rightarrow x_i$ far away since $\xi_i = 0$ for farther points.

if $y_i(x_i^T\beta+\beta_0) > 1 \Rightarrow \alpha = 0$ ($\xi = 0$)

if $y_i(x_i^T\beta+\beta_0) = 1 \Rightarrow 0 < \alpha_i < C$ ($\xi = 0$)

if $y_i(x_i^T\beta+\beta_0) < 1 \Rightarrow \xi_i > 0, \Rightarrow \mu_i = 0, \Rightarrow \alpha = C$

(2) $\alpha < \alpha_i < \infty, \xi_i = 0 \Rightarrow y_i(x_i^T\beta+\beta_0) = (1-\xi_i)$ ↘ support vector

(3) $y_i(x_i^T\beta+\beta_0) - (1-\xi_i) < 1$ if $\xi_i \neq 0$ ⇓ ⇒ $\mu_i = 0$. ⇓

$\boxed{\alpha_i = C}$ Misclassified

in both cases $\alpha_i \neq 0 \Rightarrow$ those x_i's are support vectors.

[1]

Lecture-24 SVM kernel

Using kernels, we don't need to compute the basis function to transform vectors.

like in previous example, we don't need to compute $h(x)$.

C-SVM → we call it C-SVM - whatever we learnt so far.

Different type of SVM's are possible where people use different optimisation constraints.

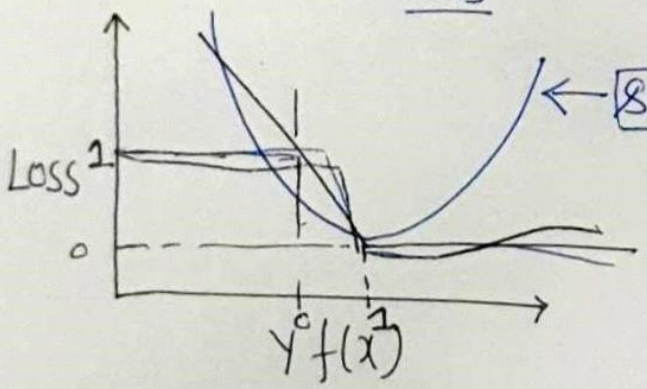
Lecture-25 - Hinge Loss Formulation

$$L_p = \sum_{i=1}^{N} \alpha_i + \frac{1}{2}\|\beta\|^2 - \sum_{i=1}^{N}[\alpha_i(y_i(x_i^T\beta+\beta_0)) - 1]$$

$$= \min_{\beta,\beta_0} \sum_{i=1}^{N} [1 - y_i f(x_i)]_+ + \frac{\lambda}{2}\|\beta\|^2$$

$[\ ]_+$ ↓ Count only when it's positive.

Loss | Penalty



← squared error

$(1 - y f(x_i))^2$

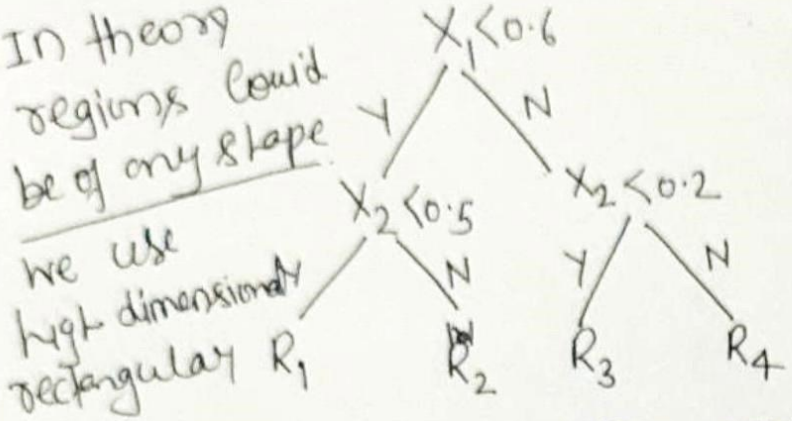
Lecture-33-1 Decision Trees - Introduction

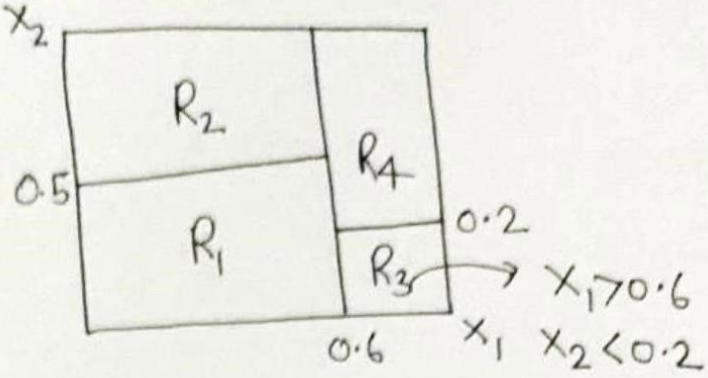
Querying properties of the Data

- Partition I/P space into rectangles.

In theory regions could be of any shape

We use high dimensional rectangular boxes.





- highly Intrepretable
- universal approximator
- Non-parametric

Regression Trees

- Fit a constant in each region.

$$\hat{f}(x) = \sum_{m=1}^{4} c_m I\{(x_1, x_2) \in R_m\}$$

(x_i, y_i) : $i = 1, \dots N$, $y_i \in \mathbb{R}$ $\forall i$

$x_i = \{x_{i1}, x_{i2}, \dots x_{ip}\}$

$$\Rightarrow f(x) = \sum_{m=1}^{M} c_m I(x \in R_m)$$

$$= c_1 I(x \in R_1) + c_2 I(x \in R_2) + c_3 I(x \in R_3) + c_4 I(x \in R_4)$$

- Determine the 'M' regions
- Given regions, find Response. $c_m \rightarrow$ response for region 'm'.

Minimise RSS:

$$= \sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2$$

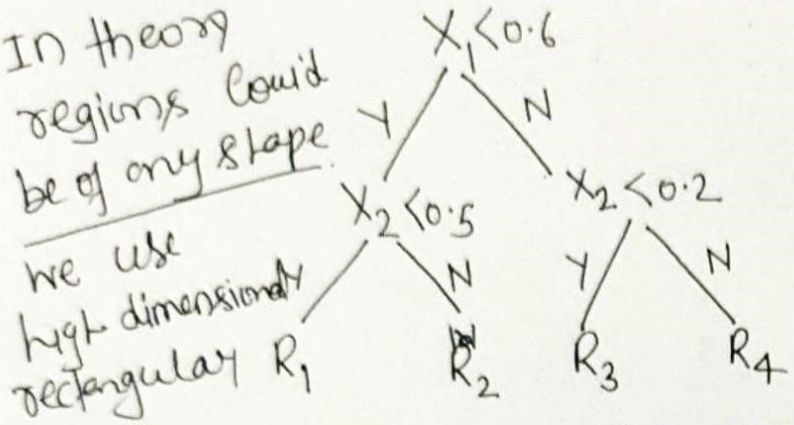
$$\hat{y}_{R_m} = \hat{c}_m = \sum_{i \in R_m} f(x_i) \rightarrow$$ Mean response For training observations

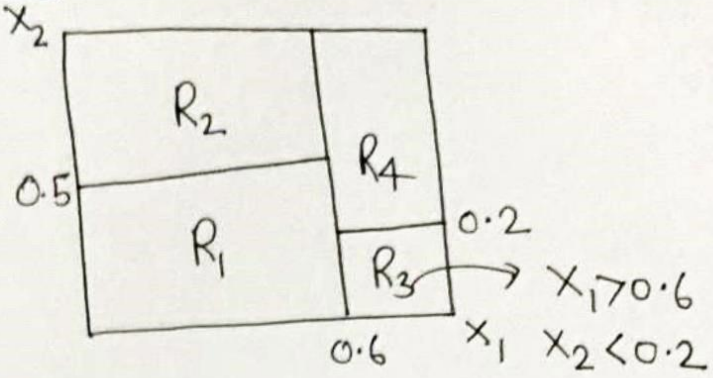
Lecture-33-1 Decision Trees - Introduction

Querying properties of the Data

- Partition I/p space into rectangles.

In theory regions could be of any shape. We use high dimensional rectangular boxes.





- highly Intrepretable
- universal approximator
- Non-parametric

Regression Trees

- Fit a constant in each region.

$$\hat{f}(x) = \sum_{m=1}^{4} c_m I\{(x_1, x_2) \in R_m\}$$

$(x_i, y_i) : i = 1, \dots N, \quad y_i \in \mathbb{R} \ \forall i$

$x_i = \{x_{i1}, x_{i2}, \dots x_{ip}\}$

$$\Rightarrow f(x) = \sum_{m=1}^{M} c_m I(x \in R_m)$$

$$= c_1 I(x \in R_1) + c_2 I(x \in R_2) + c_3 I(x \in R_3) + c_4 I(x \in R_4)$$

$c_m \rightarrow$ response for region 'm'.

- Determine the 'M' regions
- Given regions, find Response.

Minimise RSS:

$$= \sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{Y}_{R_j})^2$$

$$\hat{y}_{R_m} = \hat{c}_m = \sum_{i \in R_m} f(x_i) \rightarrow \text{Mean response For training observations}$$

very difficult to divide feature space into J regions.

Approach (greedy)

1. all observations belong to a single region
2. Split feature space, i.e split into two new branches.
3. At each step of tree building process, best split is made.
4. Consider a splitting variable j and cutpoint s, such the splitting the space into regions $R_1: \{X | X_j < s\}$, $R_2: \{X | X_j > s\}$ leads to greatest possible reduction in RSS.

$$R_1(j,s) = \{X | X_j \leq s\}, \quad R_2(j,s) = \{X | X_j > s\}$$

Seek the value of J and S such that

$$\min_{J,S} \left[\sum_{i: x_i \in R_1(J,S)} (y_i - \hat{y}_{R_1})^2 + \sum_{i: x_i \in R_2(J,S)} (y_i - \hat{y}_{R_2})^2 \right]$$

$\hat{y}_{R_1} \rightarrow C_1 \rightarrow$ Mean response

$\hat{y}_{R_2} \rightarrow C_2 \rightarrow$ (Mean response)

↓

we will do seperately.

fix J and find S

$j \in \{1, 2, \cdots p\}$

Strategy to find S → choose values from data points. Sort the points in ascending order and choose 's'.

n → Total points. & Tota time: $n * p$ → For one split

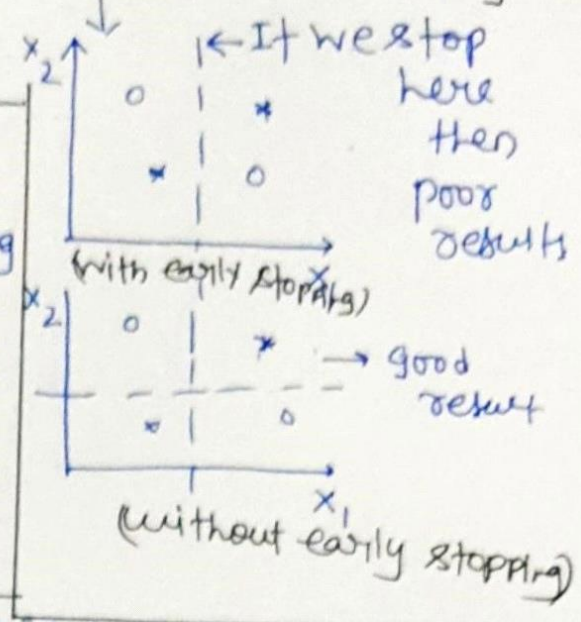
p → J.

J → Can be repeated while growing trees.

Lecture 35 - Stopping criteria & pruning

*Early stopping - May lead into poor decision boundary.

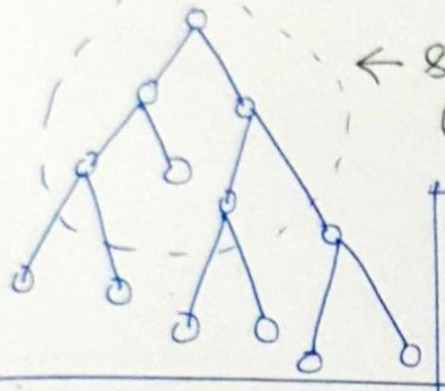
*leaf of a tree is a region



Tree pruning

Above process will lead to overfitting
→ Tree will be very complex.
→ variance higher
— Simple Tree
less regions → lower variance.
better interpretation

Strategy: Grow Tree T, and the prune it back in order to obtain subtree. → using (cross-validation)



← seleeting best subtree will be very time consuming. → No. of subtrees are very large.

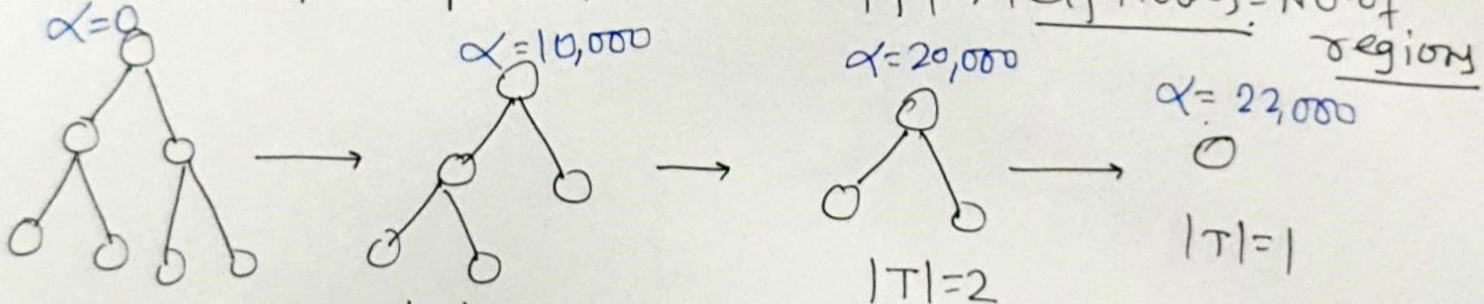
Cost-Complexity pruning: weakes link pruning

$C_\alpha(T)$ = prediction error + $\alpha|T|$

$= \sum_{m=1}^{|T|} (y_i - \hat{y}_{R_m})^2 + \alpha|T|$ ← Complexity penality

As $\alpha \uparrow$ ⇒ Numbere of Nodes pruned.

$|T|$ → leaf Nodes = No. of regions

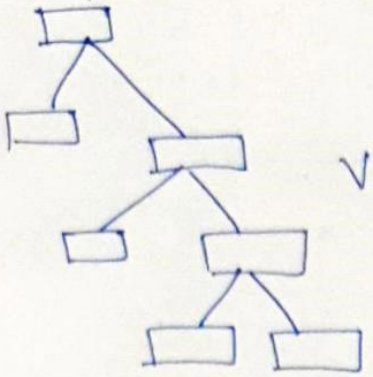


$\alpha \uparrow$ → (More penalty)

~~We want to penalise tree more having less no. of regions~~

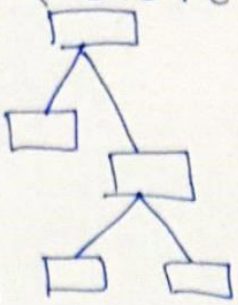
To Compare
① we already know that pruned tree will have more error.
② we need to be introduce one more term by which we can compare these trees.
by introducing Complexity penalty.

SSR = 543.8



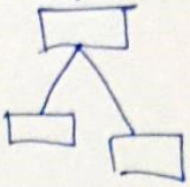
vs

SSR = 5494.8



vs

SSR = 19243



vs

SSR = 28897.2

Let's take $\alpha = 10{,}000$

Tree score = 40,543.8

Total score = SSR + 10,000 * T

Tree score = 35,494

Tree score = 39,243.7

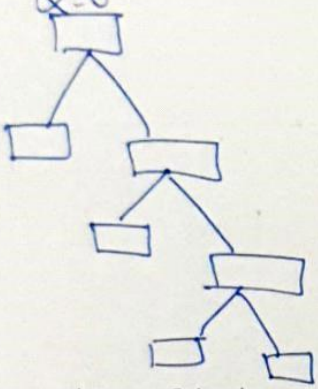
Tree score = 38,897.2

optimal subtree

using different α can give different result.
we need to find best value for alpha.

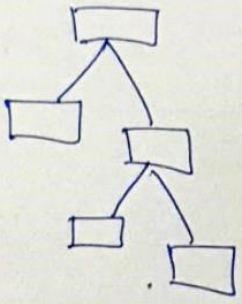
$\alpha = 0$, ⇒ Full sized tree will have lower Tree score

$\alpha = 0$



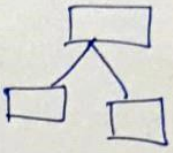
lowest tree score

$\alpha = 10{,}000$



lower tree score

$\alpha = 15{,}000$



(lower tree score)

$\alpha = 22{,}000$

↓

gives lowest tree score.

use testing data to get the optimal subtrees for different values of |T|.

Lecture 36

Decision Tree for classification - Loss function.

$$\hat{p}_{mk} = \frac{1}{|R_m|} \sum_{x_i \in R_m} I\{y_i \neq k\}$$

$\hat{p}_{mk}$ ↓ prediction

$\sum I\{y_i \neq k\}$ → Total Number of ~~Correct predictions~~

$|R_m|$ ↓ Total Number of data points belonging to class k in region 'm'

R_2	R_3
R_1	R_4.

3 classes

$p_{41}, p_{42}, p_{43}, p_{44}, p_{21}, p_{22}, p_{23}, p_{24}$
$p_{11}, p_{12}, p_{13}, p_{14}, p_{31}, p_{32}, p_{33}, p_{44}$

if $x_i \in R_m$ then check which is greater.
$[p_{41}, p_{42}, p_{43}, p_{44}]$

$$k(m) = \arg\max_k \hat{p}_{mk} = \max_k [p_{41}, p_{42}, p_{43}, p_{44}]$$

$k(4) =$ $=$

↓ label for region '4'

$$\text{classification error} = \frac{1}{|R_m|} \sum_{x_i \in R_m} I(y_i \neq k(m)) = (1 - \hat{p}_{m k(m)})$$
$$= (1 - \max_k \hat{p}_{mk})$$

Gini Index $:= \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$ – Measure of variance across K-classes for a particular region.

Cross entropy. $-\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}$ – (information gain criteria)

$\hat{p}_{mk}$ ↓ estimated

$\log \hat{p}_{mk}$ ↳ label

Lecture-37.

Decision Trees - Categorical attributes

q- unordered attributes:

▷ Divide into 2 groups. Total No. of grps possible:

$(^{q}C_1 + {}^{q}C_2 + {}^{q}C_3 + \cdots {}^{q}C_{q/2}) = 2^{q-1} - 1$ → Not feasible to go and pick split points.

in case of regression: you have to look in' points.

n → No. of datapoints.

0/1 - binary classification.

Suppose q=5 For color attribute:

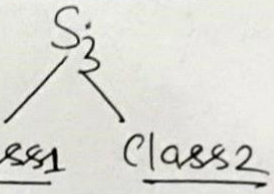
q=0 Red,
q=1 green
q=3 blue
q=4 yellow
q=5 Magenta.

P(class1/Red), P(class1/green),
P(class1/blue), P(class1/yellow),
P(class1/Magenta) → Ascending order. and then make the split..

0.2 | 0.3 | 0.4 | 0.45 | 0.55 ←

R | G | B | Y | M

S_1 | S_2 | S_3 | S_4 ← split point.



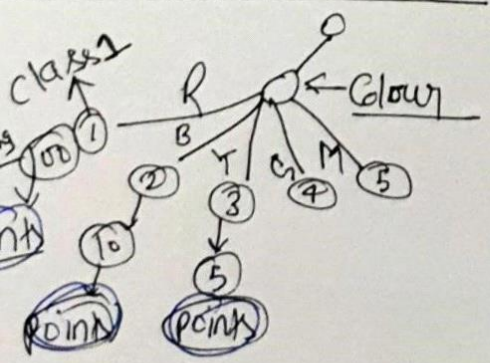
No - Need to consider $(2^{q-1}-1)$ splits.

Lecture-38. Multiway splits.

Multiclass

Heuristics: -

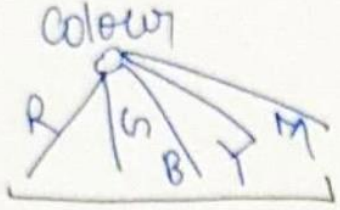
Multiway-splits

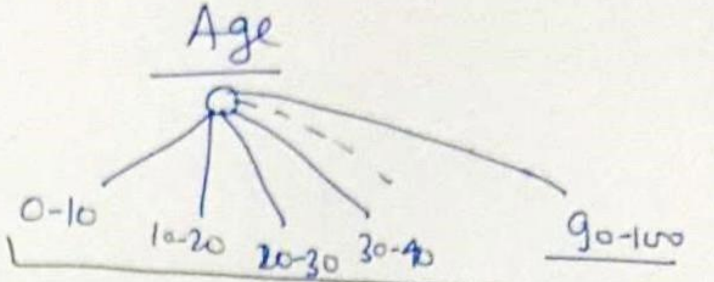


- Problem
  - → Sprawling (spreading) (hard to interpret)
  - → lost interpretality
  - → Variance more.
  - → At each split, data points will be lower..
  - → Tree will be become sparse.
  - → overfitting is the problem.

Lecture-38

- Favours attributes with more values.





Error Color > Error Age. Less error in splitting

C4.5 - Algorithm - gain-Ratio

↓

what is the information gain after splitting into 10 parts rather than 6.

→ we can also use binary split to arrive at same
→ function which is preferred by multiway splits.

↳ binary split is always better > multiway split because

↓

we can use recursive approach to make multi-way split.

Multiway splits → avoids the choosing variable (S) (split point)

Lecture-39.

Missing values, imputation & Surrogate result

Imputation:

Mean, Regression. [predict missing points]

└ Full information imputation.

Using all other feature to impute values.

Multiple imputation: - use distribution to predict values.

↓

Follows 3 steps.

1. similar to single Imputation, missing values are imputed. However, the imputed values are drawn m times from a distribution rather than just once. → gives (m - datasets)
2. Analysis: At end of this step, there should be m - analyses.
3. pooling: m results are consolidates into one result by calculating mean, variance, confidence intervals of the variable.

↓

~~by~~ i single Imputation does not take care of uncertainity in imputed values where as multiple imputation accounts for the uncertainity and range of values that the true value could have been.

→ Fitting it into a distribution and then sampling values using mean and variance of distribution

↑imputation. Lecture-39: Surrogate splits

— New Categorical ~~feature~~ value "missing".

Age
20
30
--
--
40

⇒

Age
20
30
"missing"
"missing"
40

There are some reasons, why there is missing values in our data. If we impute it, we are going to add NOise in the feature. So, It's best to just ~~leave~~ replace those values by "missing".

Surrogate splits

In decision tree, when a split is made, It depends on one variable. but what if that variable is missing. Suppose, we want to predict whethers customers will be kept or lost next year.

order ≤ 6.6

Y / N

(Lose) (Keep)

→ Surrogate split will try to predict actual split:

→ Another decision tree will be created to predict your split.

order ≤ 6.5 orders ≥ 6.5 → Fake decision tree.

— Fragment (specific to trees)

$X_3 < 5$?

Lecture-40

Instability, Smoothness & Repeated Subtrees

Instable :- More variance. Prone to Noises

Lack of smoothness :- since decision tree has piecewise rectangular boxes. clear cut boundar

Repeated subtrees -:- same set of split made at different points in decision tree.

Loss $L_{KK'}$: $\sum_{k=1}^{K} \hat{p}_{mk}(1-\hat{p}_{mk}) = \sum_{k=1}^{K} \hat{p}_{mk}\hat{p}_{mk'}$

(K: True, K': estimated)

Advantages :-

- simple to understand and interpret.
- handle both numerical and categorical data.
- performs well with large datasets.

disadvantages :-

- Non-Robust : A small change in the training data can result in a large change in the tree. and consequently final Predictions.
- For data including categorical variables with different number of levels, Information-gain is biased in favour of attributes with more levels.
- Decision tree can learn very complex decision boundary. Overfitting.

Decision Tree tutorial

Dataset

age	income	student	credit-rating	target buys_computer
Youth	high	yes	fair	
middleaged	medium	NO	excellent	
Senior	low			

Total_data = 14

: impurity measure :

- cross entropy | gini-Index

Multi-way split using cross entropy

Consider Attribute 'Age'

Youth < 3-NO, 2-Yes

Senior < 3-Yes, 2-NO

middleaged < 4 Yes, 0-NO

cross_entropy: $-\sum_{K=1}^{K} p_{mK} \log_m p_{mK}$ $\quad K=2$

initially $m=1$

$K=1 \rightarrow$ class Yes

$K=2 \rightarrow$ class NO

$-(p_{11} \log p_{11} + p_{12} \log p_{12})$

For attribute age = Youth $p_{11} = \frac{2}{5}$, $p_{12} = \frac{3}{5}$

$\text{cross}_{\text{entropy}_{Age}} = +(5/14)\left(-\frac{2}{5}\log\frac{2}{5} - \frac{3}{5}\log\frac{3}{5}\right) + \frac{4}{14}\left(-\frac{4}{5}\log\frac{4}{5}\right) + \frac{5}{14}\left(-\frac{3}{5}\log\frac{3}{5} - \frac{2}{5}\log\frac{3}{5}\right)$

$= \boxed{0.6935}$

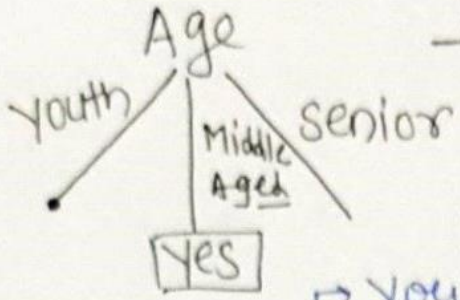
cross_entropy_credit_range (D) = $\boxed{0.8922}$

cross_entropy_income: $\boxed{0.9111}$

cross_entropy_student: $\boxed{0.7885}$

less entropy →

Decision Tree

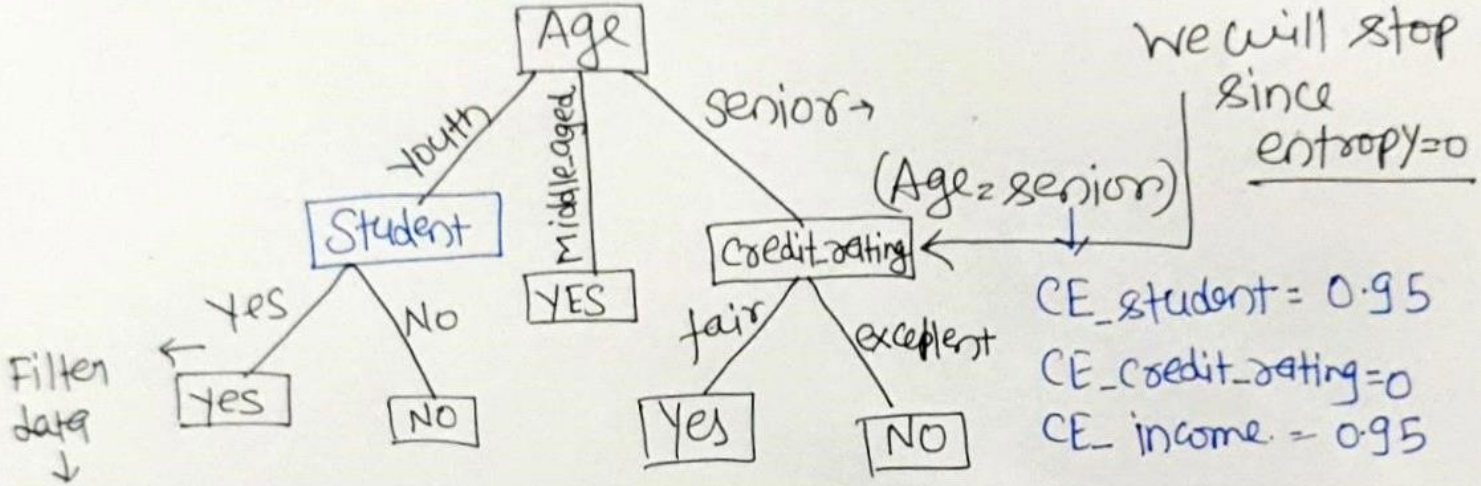


↳ You will filter dataset where age=youth

cross_entropy (Age=youth) = income

cross_entropy (Age=youth) = 0 → Minimum value of entropy
student

cross_entropy (Age=youth) =
credit_score



We will stop since entropy=0

(Age=senior)

CE_student = 0.95
CE_credit_rating=0
CE_income = 0.95

Filter data
↓
where
(Age=youth) $
(Student=yes)

Decision-Tree:

Binary split using Gini index

Consider 'Age'

Total data points = 14

(+ve) class_proportion: Age = Youth : 2/5

Age = Midd : 1

Age : Senior : 3/5

Ordering the probability.

Youth, senior, middle

$$\text{Gini-index} = \sum_{K=1}^{K} p_{mk}(1 - p_{mk})$$

Possible splits

{Youth}, {senior, middle} | {youth, senior}, {middle}

⇒ We need to calculate impurity measure for both the above splits.

$$\text{Gini}_{age \in \{youth\}}(D) = 2p(1-p)$$

$$= \frac{5}{14}\left(2 * \frac{2}{5} * \frac{3}{5}\right) + \frac{9}{14}\left(2 * \frac{7}{9} * \frac{2}{9}\right)$$

$$= 0.6508$$

For '2' class problem $G_1 = p(1-p) + (1-p)p$

$$= \boxed{2p(1-p)}$$

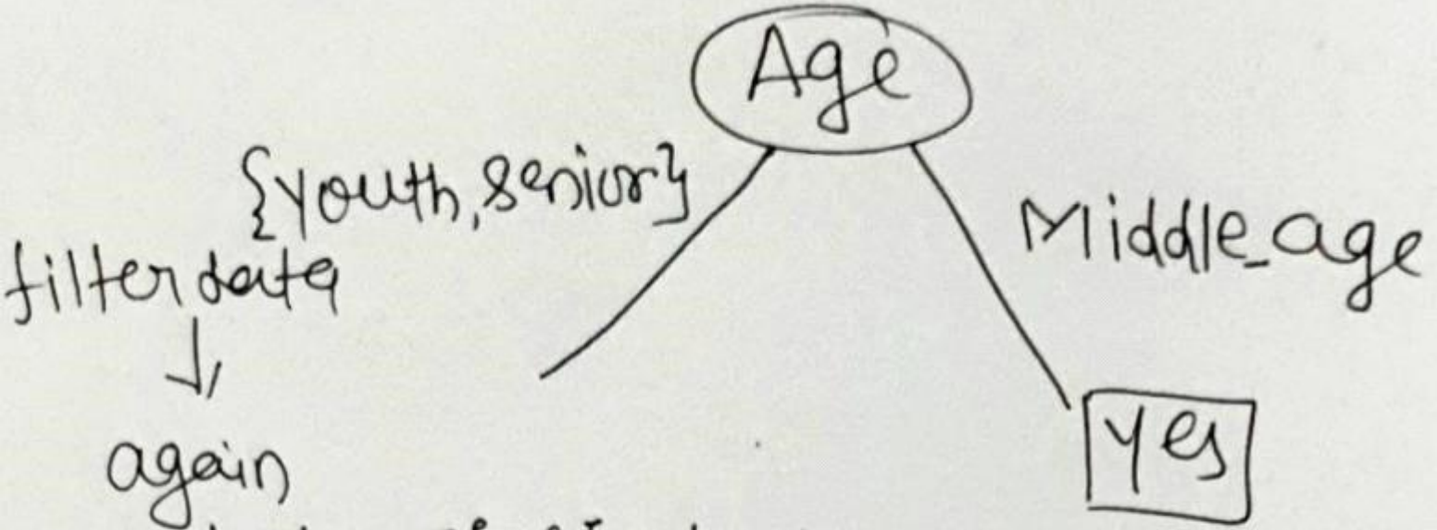
Gini-age = {youth, senior} =

Gini-student {Yes} = $7/14\left(2 * \frac{3}{7} * \frac{4}{7}\right) + 7/14\left(2 * \frac{2}{4} * \frac{1}{2}\right)$

Gini-income → Two splits =

Gini-credit rating {~~fair~~} = $\frac{8}{14}\left(2 * 6/8 * 2/8\right) + \frac{6}{14}\left(2 * \frac{3}{6} * \frac{3}{6}\right)$

Choose the one having less impure : lesser value


Age
{youth, senior}
Middle_age
filter data
yes
again
Calculate gini_index for all other attribute other than age.

StatQuest

Random Forest - Part 1
Building using, and Evaluating

- Decision trees are not flexing in order to classify new samples.

Building Random Forest

Step1: create a "bootstraped" dataset.

original-dataset (Target)

Chest pain	Good Blood	F3	F4	F5
No	No	No	125	Yes
Yes	Yes	Yes	180	No
Yes	Yes	No	210	No
Yes	No	Yes	167	Yes

Bootstrapped Dataset

Chest pain	Good blood	F3	F4	F5
Yes	Yes	Yes	180	No
No	No	No	125	Yes
Yes	No	Yes	167	Yes
Yes	Yes	Yes	180	No

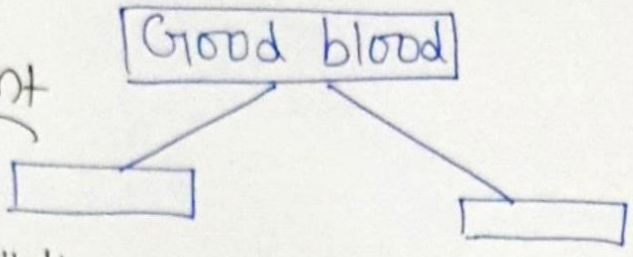
a) To create bootstrapped dataset, ie of same size of original dataset, we just randomly select samples from the original dataset.

b) we are allowed to pick the same sample more than once.

Step-2: create a decision tree using the bootstrapped dataset, but only use a random subset of variables (features) at each step.

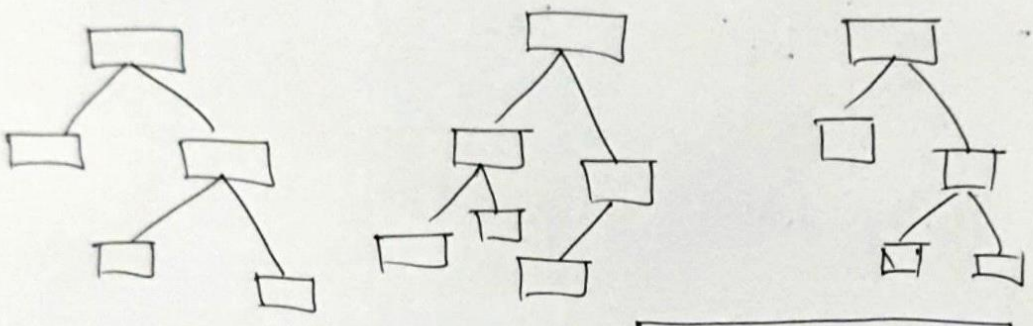
In this case, we will randomly select two variables, "Chest" and "Good blood" as a candidate for root node.

Assume, "Good blood" has lowest cross-entropy,



at this point also, randomly select two variables from "3" variables left.
and build the tree as usual.

Now, go back to step 1 and repeat: Make a new bootstrapped dataset and build a tree consider random subset of variables at each split.



This will result in ~~wide variety~~ of decision trees.

↓

Makes Random forest more effecting than individual decision trees.

How to predict?

Run the sample on all tree we have created. and calculates voting of Yes vs No

Heart_disease(Target)	
Yes	No
5	1

⇒ Class = Yes

Random Forests- Part 2

Missing data and sample clustering

Random forest Consider two types of missing data

1. Missing data in original dataset.
2. Missing data in New Sample.

↳ we can make initial guess → refine our guess.
↓
using mean, median or other imputation methods.

initial guess

Chest pain	Good blood	Blocked Artery	Weight	Target
Yes	Yes	No	167.5	No

How to refine our guesses

Step-1 : Build Random forest...

Step-2 : Run all data in all of distinct decision trees.

Find the samples which end up at same 'leaf Node' as our data-sample which we want to make a guess.

No. of Rows = No. of Columns = Total samples

Build Proximity_Matrix

	1	2	3	4
1		2	1	1
2	2		1	1
3	1	1		8
4	1	1	8	

Suppose → 3 and 4 end up in same leaf Node, we will add 1 in P[3][4] and P[4][3]

→ we divide each value by 4 Total No of Trees.

	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		0.8
4	0.1	0.1	0.8	

Random Forest

Terminology:

Bootstrapping the data plus using the aggregate to make a decision is called "Bagging".

→ Typically $1/3^{rd}$ samples does not end up in bootstrapped dataset.

33% of data → called the (out-of bag dataset)

↓ doesn't appear in bootstrapped dataset.

*Since out of bag-datasets were not used in building decision tree, we will use it to evaluate our model.

* The proportion of out of bag samples incorrectly classified by Random Forest is known as "out of bag error".

1. Build Random Forest (Change No. of variables)
2. Estimate Accuracy of Random Forest

Usually, we start with $\sqrt{\text{No. of variables}}$.

Random Forest Part 2-2

Use proximity matrix to calculate missing values.

original dataset

Chest_pain	Good blood Circ.	Blocked Arteries	weight	Heart_disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	???	??!	No

frequency(Yes) = $\frac{1}{3}$, f(No) = $\frac{2}{3}$

weighted_f(Yes) = $\frac{1}{3}$ × (The weight for Yes) (0.1) = 0.03

The weight of Yes = $\frac{\text{Proximity of Yes}}{\text{All Proximities}}$ = Proximity value for sample 2

1	2	3	4
0.1	0.1	0.8	

$= \frac{0.1}{0.1+0.1+0.8} = \frac{0.1}{1} = 0.1$

weighted_f(No) = $\frac{2}{3}$ * (The weight for No) (0.9) = 0.6

The weight for No = $\frac{\text{Sample 1 and sample 3}}{\text{All}}$

$\frac{0.8+0.1}{0.1+0.8+0.1} = \frac{0.9}{1} = 0.9$

(No > Yes)

→ New guess = No

weightage_average of (weight) = 125 * (proximity weight) + (180) * (proximity_weight) + (210) * (proximity weight)

Proximity weight (125) = $\frac{0.1}{0.1+0.1+0.8} = 0.1$

proximity weight (180) = $\frac{0.1}{0.1+0.1+0.8} = 0.1$

Proximity_weight (210) = $\frac{0.8}{0.1} = 0.8$

$= 125 \times (0.1) + 180 \times (0.1) + 210 \times (0.8) = (198.5)$

After refining guesses, we run random forest again to refine guesses, until missing values converge. (6 to 7 times).