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

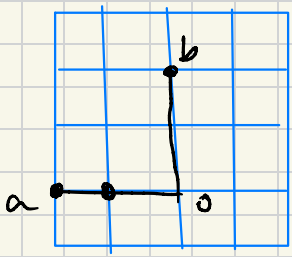
↓ L2 norm of a vector.

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

② Manhatten Distance :

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

(absolute value)



$||x_1 - x_2||_1$ L1 norm of a vector.

dist(a,b)
= dist(a,o) + dist(o,b).

Generalization to L1 & L2 norm —

③ Lp norm / Minkowski dist :

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

if p = 2 then Minkowski dist = Euclidean dist

if p = 1 then Minkowski dist = Manhatten dist.

④ Hamming Distance:

XOR between two boolean vectors.

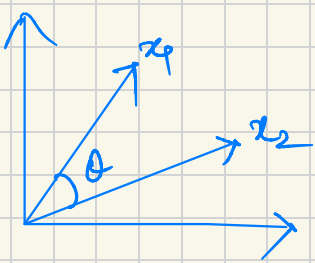
applicable to string also,

x_1 = a b c a d e f h i k

x_2 = d c b a d e g h i k

hamming dist $(x_1, x_2) = 4$.

⑤ Cosine-similarity:-



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

if x_1, x_2 is unit vector

then $\cos(\theta) = x_1 \cdot x_2$

⑥ Edit Distance:-

- The minimum edit distance between two strings.
- 3 operations allowed (Insert, Delete, substitute)
- Needed to transform one into the other.

Ex:

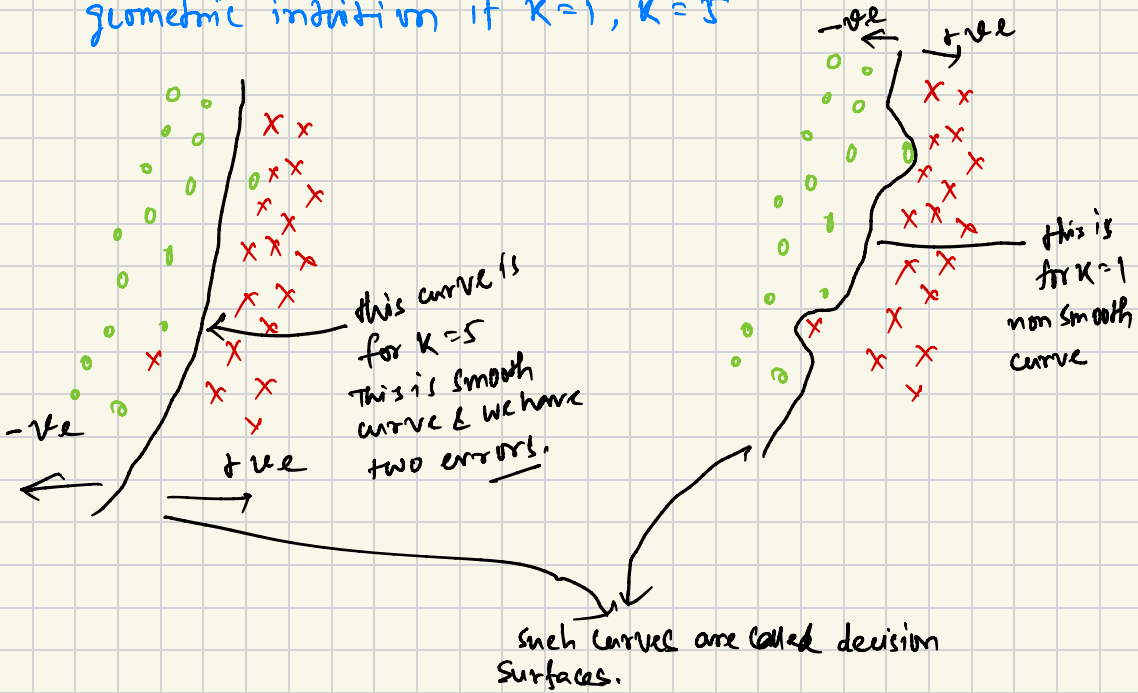
```
I N T E * N T I O N
| | | | | | | | | |
* E X E C U T I O N

d s s   i s
```

So, Distance between these two is 5.

how to decide K ?

K in KNN is hyperparameter.

geometric intuition if K=1, K=5



if K is too large, let K=n

then if $n_1 > n_2$ [$n_1 + n_2 = n$]

In this case, any query point you take, it will be true. because you have to consider all n points & among n_1 & n_2, $n_1 > n_2$. so new point will be assigned to n_1 side.

Overfit
- It tries to cover noisy points also
- non-smooth
- no mistakes

K=1

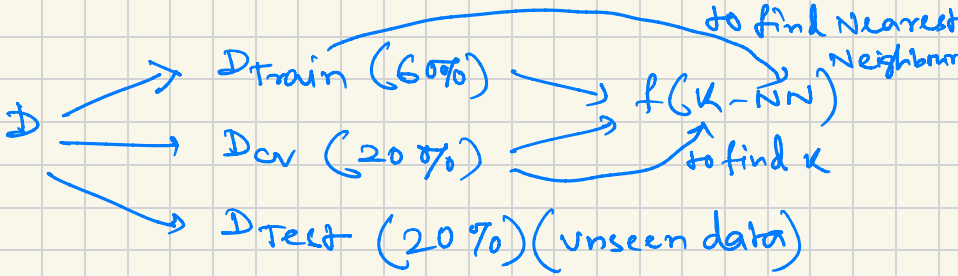
underfit
- Imperfect
- It cares about majority points. every points belongs to majority class.
- Imperfect

K=n

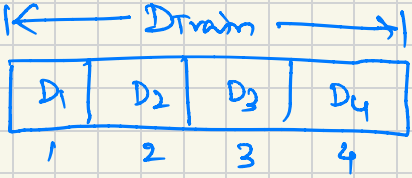
wellfit
- less prone to noise
- robust
- K=5 smooth.

** How to determine K ?

(1) We will devide the whole dataset into 3 parts.



(2)

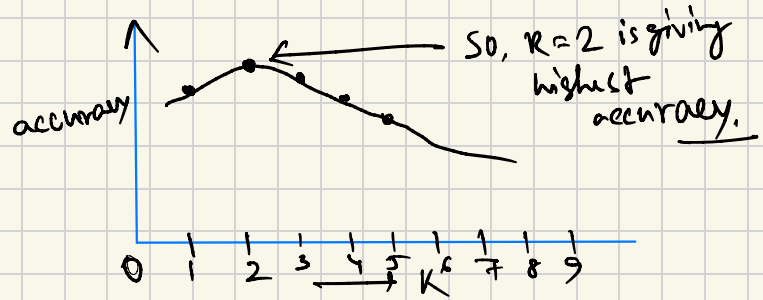


→ randomly divide into 4 parts with equal size. (4-Fold)

(3)

	D_{Train}	D_{CV}	Accuracy	
K=1	$D_1 D_2 D_3$	D_4	a_1	avg(a_1, a_2, a_3, a_4)
K=1	$D_1 D_2 D_4$	D_3	a_2	
K=1	$D_1 D_3 D_4$	D_2	a_3	
K=1	$D_2 D_3 D_4$	D_1	a_4	
K=2			a_1	avg(a_1, a_2, a_3, a_4)
K=2			a_2	
K=2			a_3	
K=2			a_4	

Considering 4-fold cross-validation we need to plot average accuracy.



Test/Evaluation time & space complexity:

Input: D_{Train}, K, $x_q \in R^d$

output: y_q

K is generally small, its 5 or 10.

KNNPts = [ ]

for each x_i in D_{Train}: → n points, d-dim each, so, time complexity = O(nd)

- Compute $d(x_i, x_q) \rightarrow d_i$
- Keep smallest K distance (x_i, y_i, d_i) → considering K is very small it will be O(1)

O(nd)

Count_pts = 0 ; Count_neg = 0

for each x_i in KNNpts:

- if y_i is true:
  - Count_pts + 1
- else Count_neg + 1

O(1)

- if Count_pts > Count_neg:
  - return y_q = 1 (+ve)
- else:
  - return y_q = 0 (-ve)

O(1)

Time Complexity = O(nd)

Space Complexity = O(nd) to keep training dataset.

** Limitations of KNN:

It takes huge time & Space. to reduce it we have some techniques

→ KD-tree

→ LSH (Locality sensitive Hashing)

weighted KNN:

Instead of majority votes, sometimes we can go with shortest distance.

let $K = 5$

for given x_q

$x_q, y_1, d_1 (0.1)$ −ve
$x_2, y_2, d_2 (0.2)$ −ve
$x_3, y_3, d_3 (1.0)$ +ve
$x_4, y_4, d_4 (2.0)$ +ve
$x_5, y_5, d_5 (4.0)$ +ve

these points are closest to x_q, but if we consider majority votes, it will be assigned to +ve. Do we need to consider distance also? why only majority votes? to eliminate this we can go with weighted KNN.

one way to assign weights is

$w_i = \frac{1}{d_i}$

$d_i \uparrow \; w_i \downarrow$

$d_i \downarrow \; w_i \uparrow$

Sum = 15 $(1 0.5)$

Sum = 1.75 $(1 + \frac{1}{2} + \frac{1}{4})$

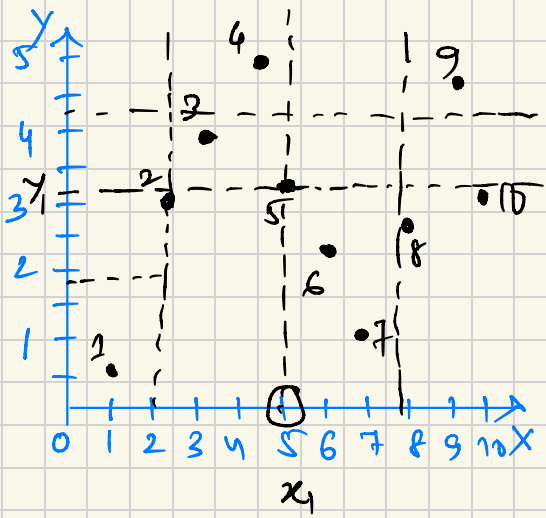
Here, $15 >> 1.75$ so, it will be assigned to −ve class.

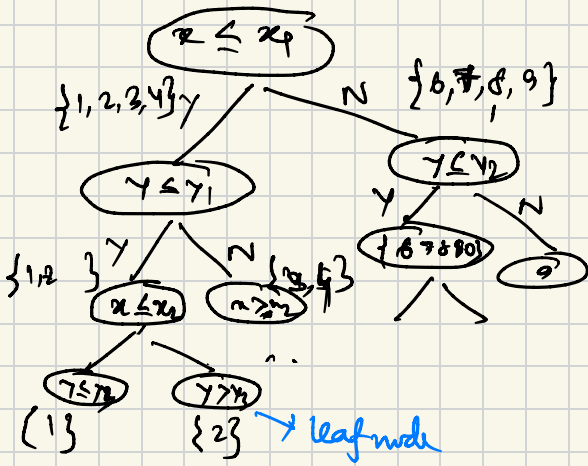
How to implement kd-tree using BST?

BST is → 1D [scalar]

↳ Extending BST to K-dimensional tree is called kd tree.

step 1: pick x-axis, project points into x-axis, compute median, split data into median.





You alternate each axis one after another till leaf nodes.

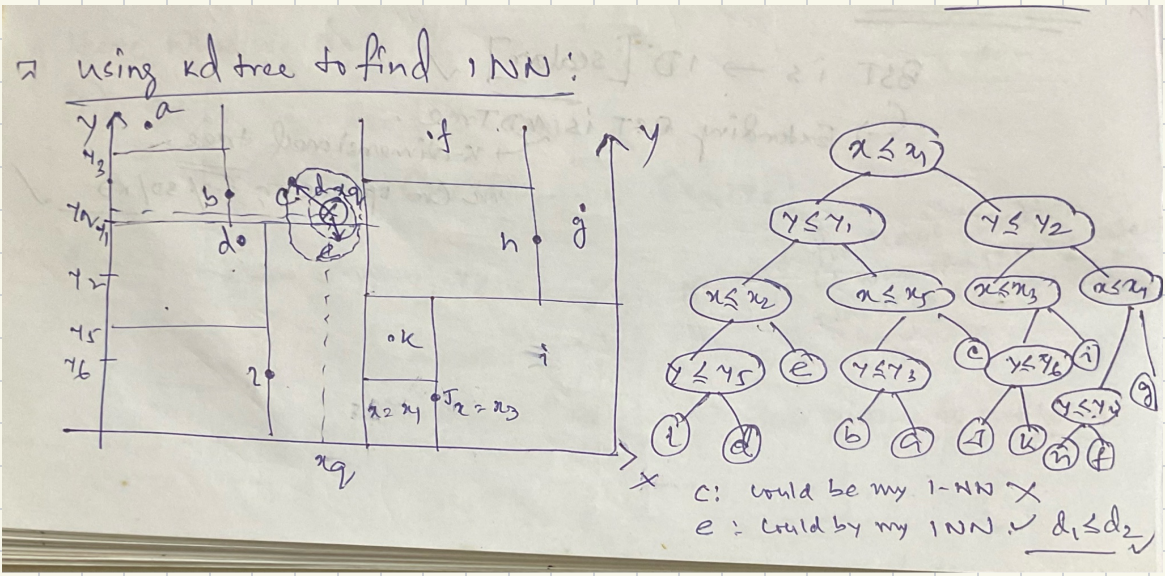
So, kd-tree breaks the space using axis parallel lines/planes into rectangle (2D) / cuboids (3D) / hypercuboid (nD).

In 2D → axis parallel lines → rectangles

In 3D → axis parallel planes → cuboids.

In nD → axis parallel hyperplane → hypercuboid.

using KD-tree to find 1NN:



# of comparison to find 1-NN

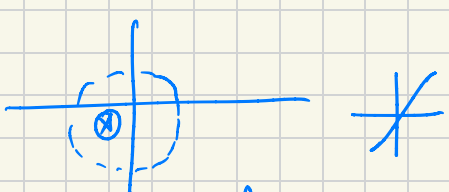
best case: $O(\log n)$

worst case: $O(n)$

if it is K-NN, the best case $O(K * \log n)$

worst case $O(K * n)$.

Limitations of KD-tree:



When d is not small the time complexity $O(2^d \log n)$

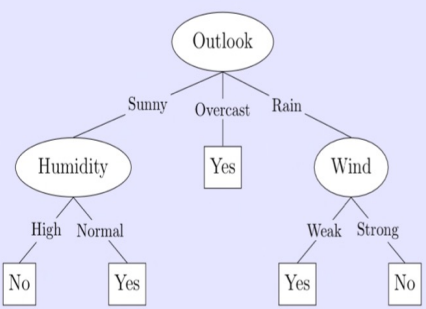
for 2D if need 4 regions to cheer

Decision Tree:

Day	Outlook	Temp	Humidity	Wind	Tennis?
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	**No**
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

$= \{D\}$

given a new query you just need to pass your data to DT to find your output.



$\Rightarrow$ DT

Algorithm 1 DECISIONTREETRAIN(*data*, *remaining features*)

```
guess ← most frequent answer in data                    // default answer for this data
if the labels in data are unambiguous then              (only single class in the leaf node)
    return LEAF(guess)                                  // base case: no need to split further
else if remaining features is empty then                (if no features left to split then stop)
    return LEAF(guess)                                  // base case: cannot split further
else                                                    // we need to query more features
    for all f ∈ remaining features do
        NO ← the subset of data on which f=no
        YES ← the subset of data on which f=yes
        score[f] ← # of majority vote answers in NO
                   + # of majority vote answers in YES
                                                        // the accuracy we would get if we only queried on f
    end for                                             → Storing score of a feature (Information Gain)
    f ← the feature with maximal score(f)
    NO ← the subset of data on which f=no
    YES ← the subset of data on which f=yes
    left ← DECISIONTREETRAIN(NO, remaining features \ {f})
    right ← DECISIONTREETRAIN(YES, remaining features \ {f})
    return NODE(f, left, right)
end if
```

How to calculate these scores?

Entropy: Let's say $Y = Y_1, Y_2 \cdots Y_K$ is random variable.

$$H(Y) = -\sum_{i=1}^{K} P(Y_i) \log_b(P(Y_i))$$

for categorical values

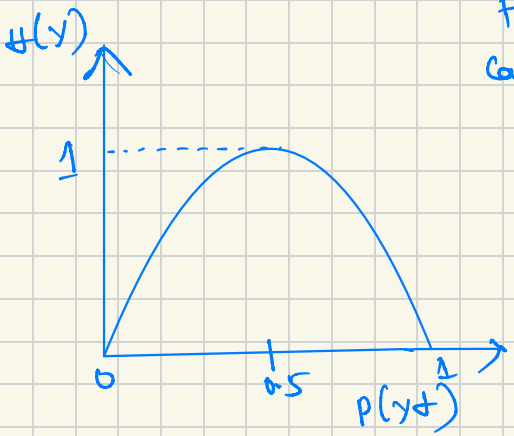
$$H(Y) = -\int P(Y_i) \log_b(P(Y_i))\, dy$$

for real values

Here $b = 2$ or $b = 2.718$

$\log_2 = \lg$

$\log_e = \ln$



for two class classification, Entropy can never be more than 1.

$$P(y+) = 1 - P(y-)$$

$y = y_1, y_2 \cdots y_k$

↳ if they are equiprobable then entropy is maximum.

Ex- let $P(y+) = P(y-) = 0.5$

$$E(y) = -P(y+)\log(P(y+)) - P(y-)\log(P(y-))$$

$$= -0.5 \log 0.5 - 0.5 \log 0.5$$

$$= -(\log 0.5) = -(\log 1 - \log 2)$$

$$= 1.$$ ✓

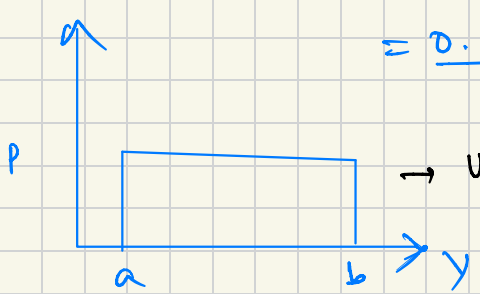
If any one of random variable is most probable & others are 0 then, ⟶ entropy is minimal.

let $P(y+) = 1 \quad P(y-) = 0$

$$E(y) = -P(y+)\log(P(y+)) - P(y-)\log(P(y-))$$

$$= -1 * 0 - 0 * \log 0$$

$$= 0.$$



↳ Uniform distribution, so, if r.v y is equiprobable then it has maximum Entropy.

Y_1 → less peaked
Y_2 → very peaked

So, Y_1 has more data variance
Whears Y_2 has low data variance

So, $H(Y_1) > H(Y_2)$

entropy of Y_1 always greater than Y_2.

Partition of Data:

Day	Outlook	Temp	Humidity	Wind	Tennis?
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	**No**
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

initial Entropy $H_D(Y) = -\frac{8}{14} \times \log\left(\frac{8}{14}\right) - \frac{6}{14} \times \log\left(\frac{6}{14}\right)$

$= -.57 \times (-.8074) - .42 \times (-1.2224)$

$= .46 + .52$

$= .98$

Let, D → D_1 (5 rows) $H_{D_1}(Y) = 0.97$
D → D_2 (4 rows) $H_{D_2}(Y) = 0$
D → D_3 (5 rows) $H_{D_3}(Y) = .97$

Information gain (IG):

$$\left(.97 \times \frac{5}{14} + 0 \times \frac{4}{14} + .97 \times \frac{5}{14}\right) - .98$$

weight entropy after partitioning dataset — entropy before partitioning

$$IG(Y, var) = H_D(Y) - \left[\sum_{i=1}^{k} \frac{|D_i|}{|D|} * H_{D_i}(Y)\right]$$

Gini Impurity : (I_G)

~ Similar to entropy

$$I_G(Y) \neq IG(Y)$$

for random variable $Y = Y_1, Y_2 \dots Y_k$

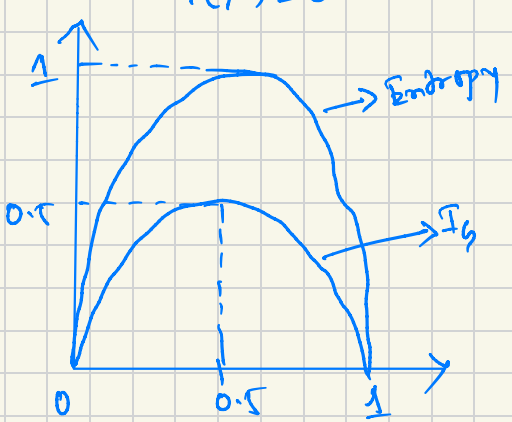
$$I_G(Y) = 1 - \sum_{i=1}^{k} (P(Y_i))^2$$

Case 1: $P(Y+) = 0.5$, $P(Y-) = 0.5$

$$I_G(Y) = 1 - [(0.5)^2 + (0.5)^2]$$
$$= 1 - [.25 + .25]$$
$$= .5$$

Case 2: $P(Y+) = 1$, $P(Y-) = 0$

$$I_G(Y) = 1 - [1^2 + 0^2]$$
$$= 1 - 1 = 0$$



max value for Gini Impurity is 0.5.

H(Y) is computationally intensive due to log calculation whereas computing square has less computational cost compare to log computation.

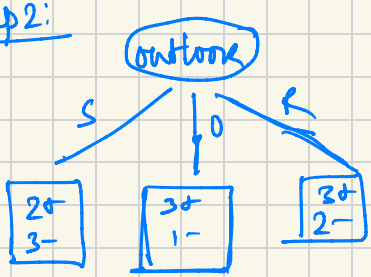
So, I_G is preferable.

** Construct a DT:

Step 1: $D = \{y^+, y^-\} = \{8, 6\}$ & $H(y) = .98$

Day	Outlook	Temp	Humidity	Wind	Tennis?
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	**No**
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

Step 2:



$$\left[-\frac{5}{14}\left(\frac{2}{5}\log\left(\frac{2}{5}\right)+\frac{3}{5}\log\left(\frac{3}{5}\right)\right)\right.$$

$$-\frac{4}{14}\left(\frac{3}{4}\log\left(\frac{3}{4}\right)+\frac{1}{4}\log\left(\frac{1}{4}\right)\right)$$

$$\left.-\frac{5}{14}\left(\frac{3}{5}\log\left(\frac{3}{5}\right)+\frac{2}{5}\log\left(\frac{2}{5}\right)\right)\right]-.98$$

$$=\left[-\frac{5}{7}\left(\frac{2}{5}\log\left(\frac{2}{5}\right)+\frac{3}{5}\log\left(\frac{3}{5}\right)\right)-\frac{4}{14}\left(\frac{3}{4}\log\left(\frac{3}{4}\right)+\frac{1}{4}\log\left(\frac{1}{4}\right)\right)\right]$$

$$-.98$$

$$=-\frac{5}{7}(-.528-.4422)+\frac{4}{14}(-.3112-.5)$$

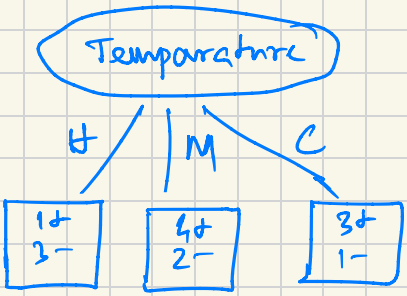
$$-.98$$

$$=(.3914+.2317)-.98$$

$$=|-.6231-.98|$$

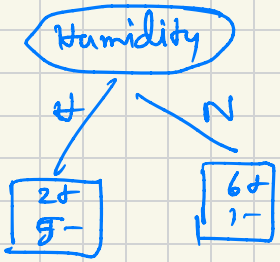
$$=.35$$

Step 3:



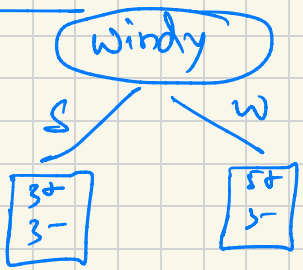
find IG as above

Step 4:



find IG as above

Step 5:



find IG as above

Step 6:

$IG(Y, \text{outlook}) = 0.35$

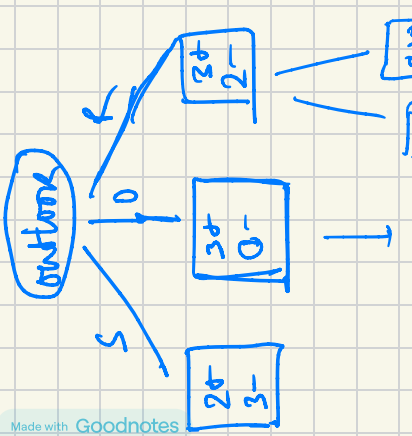
$IG(Y, \text{Temparature}) = \_\_$

$IG(Y, \text{Humidity}) = \_\_$

$IG(Y, \text{Windy}) = \_\_$

We will choose the heighest IG as root.

$$IG(Y,f) = H_D(Y) - \sum_{i=1}^{K} \frac{|D_i|}{|D|} * H_{D_i}(Y)$$

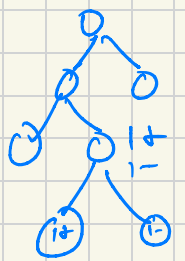


We are recursively branching each nodes based on I_G.

** three points where to stop growing the tree —

(1) pure node → stop growing the tree.

(2) Can't grow the tree anymore due to lack of points



In this case no point of growing the node.
Suppose, your dataset has 10k points & you have only one points which follows one path. That point may be noisy. (It will overfit)

(3) if we are too deep → stop growing the tree.

As, the depth of the tree ↑ → chance of overfitting ↑

depth is small → underfit.

In DT → depth of the tree is a hyperparameter.
↳ we can use CV to find it.

Pruning: Remove subtrees for better generarization (decrease variance)

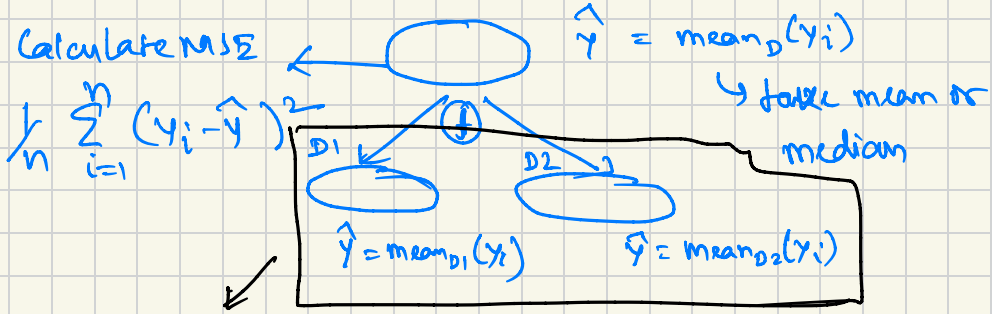
Prepruning: Early stopping

Postpruning: Grow the whole tree then prune subtrees which overfit on the pruning set

Prepruning is faster but postpruning is more accurate.

DT for Regression:

- IG used for classification
- MSE or MAE is used for regression



find weighted mean square

$w_1 MSE_{D1} + w_2 MSE_{D2}$

Now subtract $\frac{1}{n}\sum_{i=1}^{n}(y_i-\hat{y})^2 - \left[w_1 MSE_{D1} + w_2 MSE_{D2}\right]$

whichever split minimizes MSE, should consider for split.

Dimensionality Reduction 8

volume of a d-dim hyperspace

$$V_d(r) = \frac{r^d \pi^{d/2}}{\Gamma\left(\frac{d}{2}+1\right)} \qquad V_3(r) = \frac{r^3 \pi^{d/2}}{\Gamma\left(\frac{5}{2}\right)}$$

$$\Gamma(5/2) = \int_0^{5/2} e^{-x} x^{\frac{5}{2}-1} dx$$

$$V_3(r) = \frac{r^3 \pi^{3/2}}{\frac{3}{4}\sqrt{\pi}}$$

$$\Gamma\left(1+\frac{3}{2}\right) = \frac{3}{2}\Gamma\frac{3}{2} = \frac{3}{2}\times\frac{1}{2}\sqrt{\pi} = \frac{3}{4}\sqrt{\pi}$$

$$V_3 r = \frac{4}{3}\pi r^3 \quad \checkmark$$

$$\frac{\text{vol(sphere)}}{\text{vol(cube)}} = \frac{r^d \pi^{d/2}}{\Gamma\left(\frac{d}{2}+1\right)} \Big/ (2r)^d$$

$$\frac{\pi^{d/2}}{2^d \frac{d}{2}\Gamma\frac{d}{2}}$$

Gamma function

$$\Gamma(n) = \int_0^\infty e^{-x} x^{n-1} dx$$

$$\Gamma(1) = 1, \quad \Gamma(1/2) = \sqrt{\pi}$$

$$\Gamma(n+1) = n\Gamma(n)$$

$$\Gamma(n+1) = n!$$

$$\Gamma(n)\Gamma(1-n) = \frac{\pi}{\sin n\pi}$$