Chapter-34

Decision Trees

34.1 Geometric Intuition of decision tree: Axis Parallel hyperplanes :-

→ As we seen previously about

K-NN, NB, logistic-Regression, Linear-regression, SVM.

- K-NN → Instance based method
- NB → probabilistic method
- logistic-Regression, Linear-regression, SVM → geometric hyperplane.

→ Decision tree is simple if else classifier

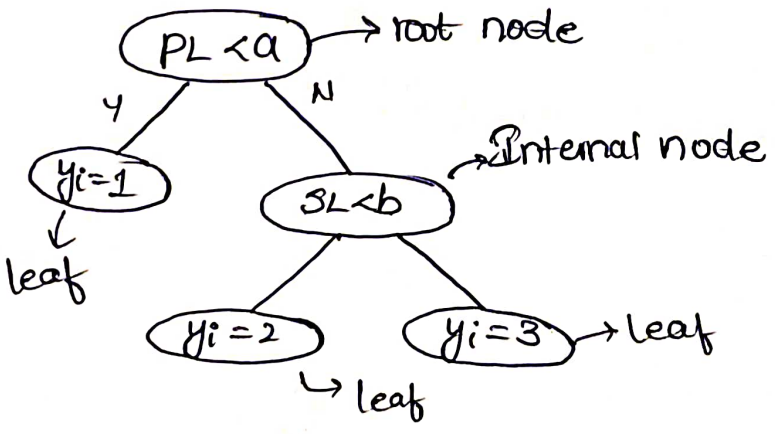
As we seen in IRIS dataset

$y_i = \{1, 2, 3\}$ (SL, SW, PL, PW)

$x_i = \langle SL, PL, SW, PL \rangle$

```
if PL < a
    yi = 1
else if SL < b
    Class = 2
else
    Class = 3
```

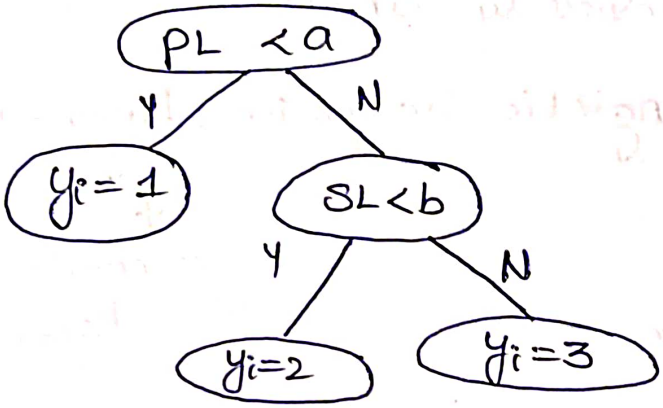
So diagramatically we can do like

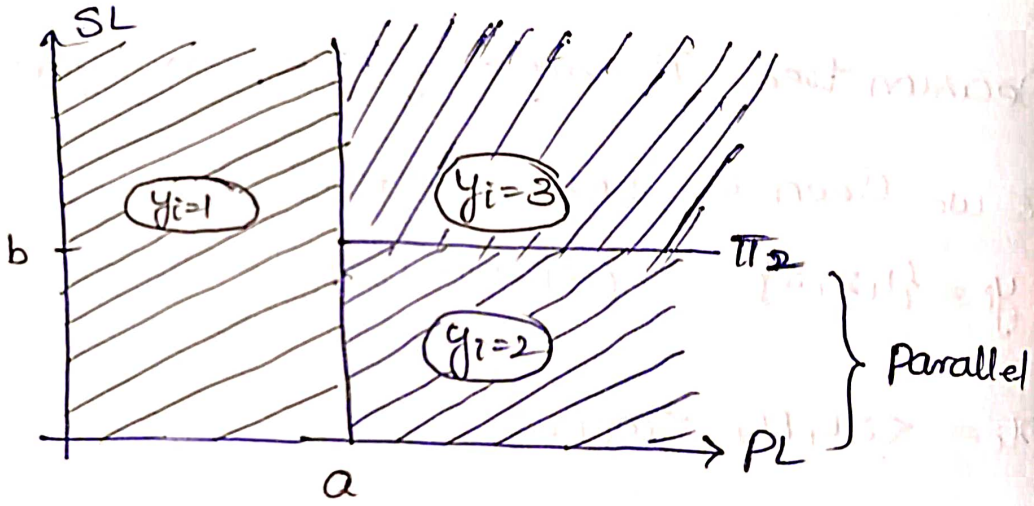


→ Usually we make decisions @ Root node and @ Internal nodes.

→ Geometric Intuition :-

If we consider





decision ; TT

→ Geometrically Decision trees are the set of axis parall hyperplane

→ Axis parallel usually can be the parallel lines which are parallel to any axis (either x or y)

34.2 Sample Decision Tree :-

← Play tennis Example →

34.3 Building a decision tree : Entropy

Entropy : is a metric to measure the impurity in given attribute

$$H(Y) = -\sum_{i=1}^{K} P(y_i) \log_b (P(y_i))$$; where b can be

b = 2

or Oiler no. b = 2.718

Note :- $\log_2 = \lg$

$\log_e = \ln$

& $P(y_i) = P(Y = y_i)$

If we see play tennis example

Y : Play tennis

$y_+, y_- \Rightarrow P(Y+) = 9/14$

$P(Y-) = 1 - P(Y+) = 5/14$

$$H(Y) = -\sum_{i=1}^{K} P(y_i) \log_2 (P(y_i))$$

$$H(Y) = \underbrace{-\frac{9}{14}}_{P(Y+)} \log\left(\frac{9}{14}\right) - \underbrace{\frac{5}{14}}_{P(Y-)} \lg\left(\frac{5}{14}\right) = 0.94$$

$P(Y+)$ = no of +ve pts / Total no of pts = % of +ve pts in D

$P(Y-)$ = % age of -ve pts in D

best-Split (X, Y)

~~It~~ bet Entropy helps to select the right feature / root node among all the features to construct a decision tree

OK entropy value < 1

Properties of Entropy:-

- To understand this let us assume we have 2 cla Dataset or 2 category Dataset.

$$Y \rightarrow Y+, Y-$$

Case1 :-

$$D \begin{cases} \nearrow Y_+ \rightarrow 99\% \\ \searrow Y_- \rightarrow 1\% \end{cases} \Big\} H(Y) = -0.99 \log 0.99 - 0.01 \log 0.01$$
$$= 0.0801$$

Case2

$$D \begin{cases} \nearrow Y_+ \rightarrow 50\% \\ \searrow Y_- \rightarrow 50\% \end{cases} \Big\} H(Y) = -0.5 \log(0.5) - 0.5 \log(0.5)$$
$$= 1$$

Case3

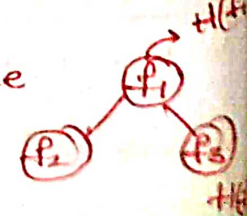
$$D \begin{cases} \rightarrow Y_+ \rightarrow 0\% \\ \rightarrow Y_- \rightarrow 100\% \end{cases} \Big\} H(Y) = 0$$

→ from this we can say if both the classes are equally probable then we get maximum entropy ie. 1. (Impure Subsplit)

→ If one is fully dominant we can say that Entropy is minimum ie ~~low~~ 0. (Pure Subsplit)

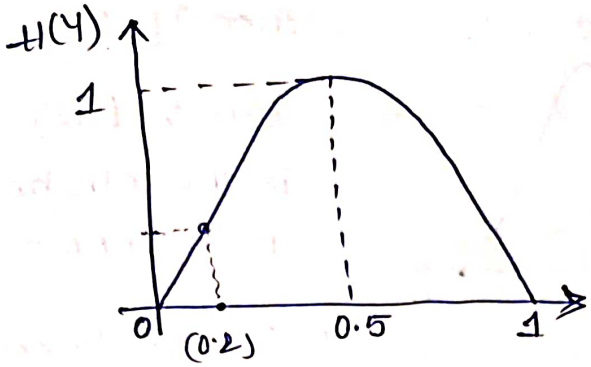
→ As we go from maximum ↓ our Entropy tends to decrease

* Entropy will be Calculated for each Node



* $P(y_t) = 1 - P(y_i)$

Graphically we can define as



If $y_t \to 0.2$ that means $y_- \to 0.8$

* Ily if we have a n class R.V

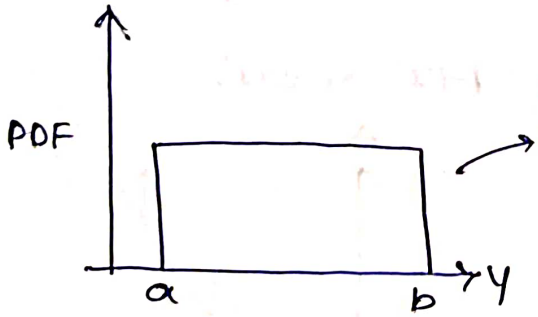
$Y_1 = \underbrace{y_1, y_2, \ldots y_k}$

If all of them are equally probable

Entropy is maximum.

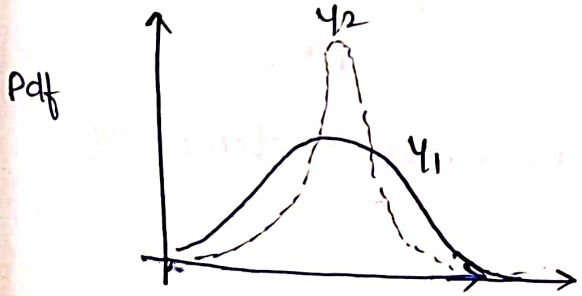
Ily if $y_1 \to$ most probable
$y_2, y_3 \ldots y_k \to$ '0'
} Entropy is minimum.

*If we consider PDF



→ uniform distribution

Entropy is maximum
(∵ all points are equally probables)



$y_1 \to$ less peaked
≈ uniform distribution

$y_2 \to$ very peaked

so $H(y_2) < H(y_1)$

→ More Peaked distribution is, less is its entropy.

34.5 Building a decision Tree : Information Gain

for a Random Variable Y if that Random Variab divided into Small sub set.

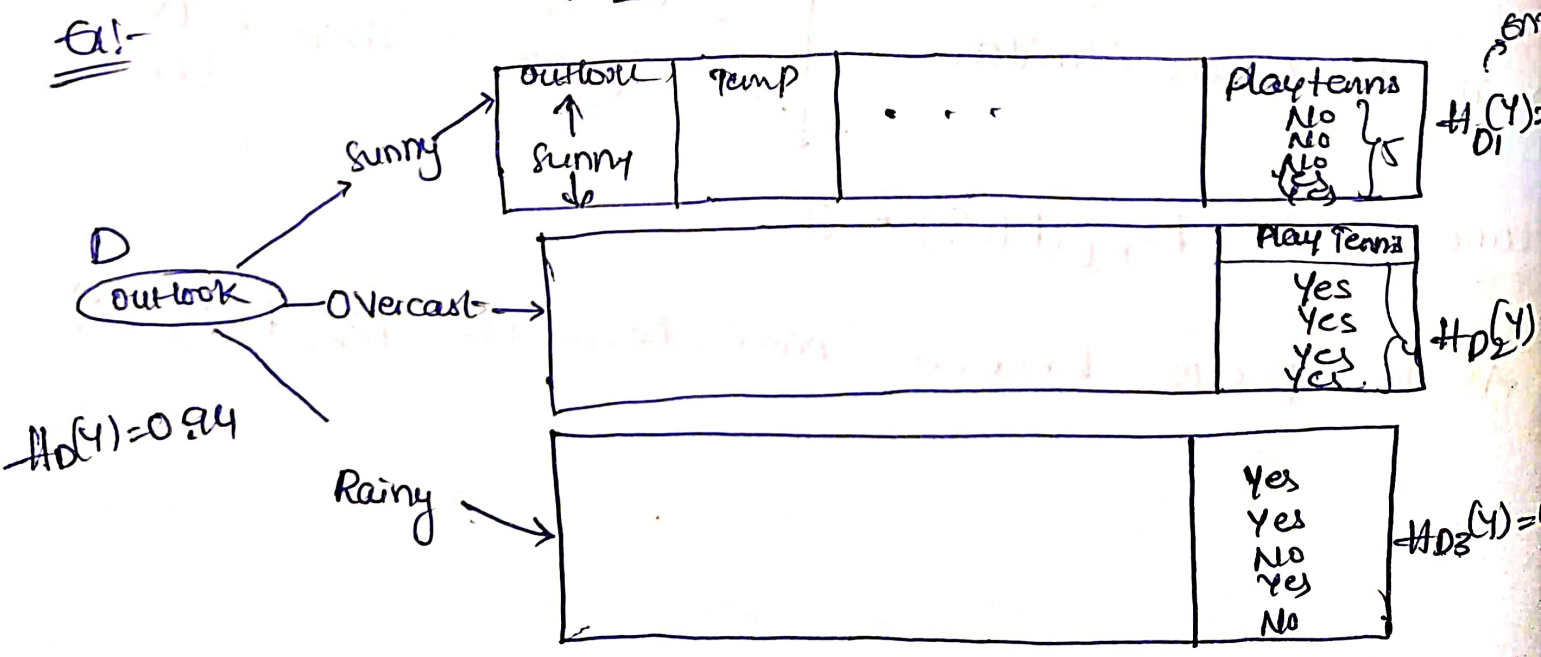
$$Y \xrightarrow{\text{Variation}} y_1, y_2 \cdots y_k$$

or considering a Dataset D if that Dataset is divided into k Datasub sets

$$D \longrightarrow D_1, D_2, \cdots D_k$$

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

Ex:-



simply if a Dataset D is given, we need to divide the Datasets into n Sub Datasets using a feature. (D_1, D_2, D_3)

. After that find Entropy of the Datasets, (That is for the Dataset D and for Datasub set D_1, D_2, D_3.

. Then we can get Information gain as

Information Gain = Entropy (Parent) - [weighted Average Entropy of Child nodes].

where

weighted Average Entropy of Child nodes $= \frac{|D_1|}{|D|} \times H_{D_1}(Y) + \frac{|D_2|}{|D|} \times H_{D_2}(Y) + \frac{|D_3|}{|D|} \times H_{D_3}(Y)$

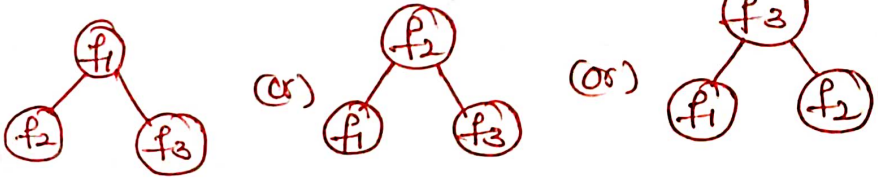
Entropy of parent $= H_D(Y)$

• for the above example

$$0.97 - \frac{5}{14} \times 0.97 + \frac{4}{14} \times 0 + \frac{5}{14} \times 0.97.$$

Note: Information gain helps us to select the pattern how the decision trees to be

fit



* The higher the Information gain that particular model / Structure need to be considered.

34.6 Building a decision Tree : Gini Impurity :-

Gini Impurity is similar to Entropy.

$$I_G(Y) = 1 - \sum_{i=1}^{K} [P(y_i)]^2 \qquad Y \to y_1, y_2, y_3 \cdots y_k$$

Compared to Entropy Gini Impurity is less computatio complex and be have less Time complexity as Entropy uses logarithms.

$I_G(Y)$	$H(Y)$
$1 - P(Y+)^2 + P(Y-)^2$	$-P(Y+)\log_2 P(Y+)$ $-P(Y-)\log_2(P(Y-))$

If we consider the difference blw Entropy & Gini Impurity

let $Y \to y_+, y_-$

Case1:- $P(Y+) = 0.5$

$P(Y-) = 0.5$

$I_G(Y) = 1 - (0.25 + 0.25) = 0.5$

$H(Y) = 1$

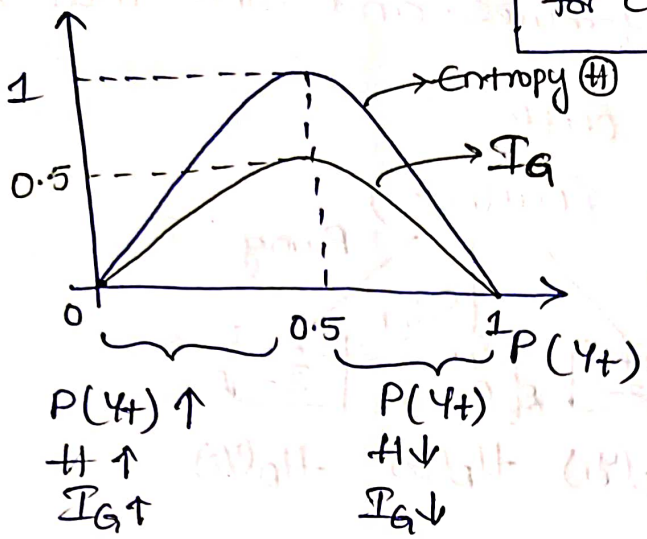
Case: 2 $P(Y+) = 1$

$P(Y-) = 0$

$I_G(Y) = 1 - (1+0) = 0$ $H(Y) = 0$

Geometrically

2- Category case :- Y_+, Y_-



Gini Impurity (Programatically):

```
1 - sum((np.sum(y==c)/m)**2
for c in np.unique(y))
```

Tree node → Gini impurity → Building DT

Note:

Gini Impurity Notation = $I_G(Y)$

Information Notation = $IG(Y)$

347 Building a Decision Tree: Constructing a Decision Tree:-

#Considering the example of ~~Decision tree~~. play tennis Dataset.

#To Construct a Decision tree,

Inputs:-

X, Y, max-depth, min-sample-Split = 2

min-samples-leaf = 1

Step 1:-

1. Considering any feature in the Dataset and find its Entropy

num-samples-per-class =

$\sum_i y_i \quad P(y_i)$

Predicted class = $\arg\max [P(y_i)]$

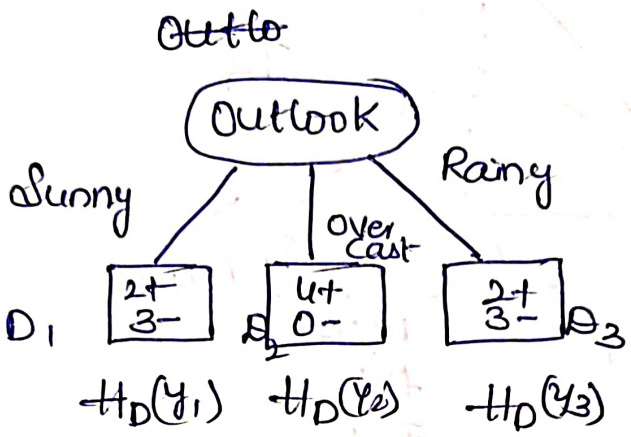
~~(a)~~ Outlook

D ⟶ 9+, 5-

$H_D(Y) = 0.94$

Step2

Now Divide the feature Data into 'n' Sub data se
using internal features. and find their Entropies th



and after that find ~~the~~ their weighted Entropy

In this example = 0.69

Step3

find IG (Y, Outlook) [that means find the Information By using the feature 1] using ~~the~~ HD(Y) & weighted Entropy.

In this Example IG(Y, Outlook) = 0.94 - 0.69 = 0.25

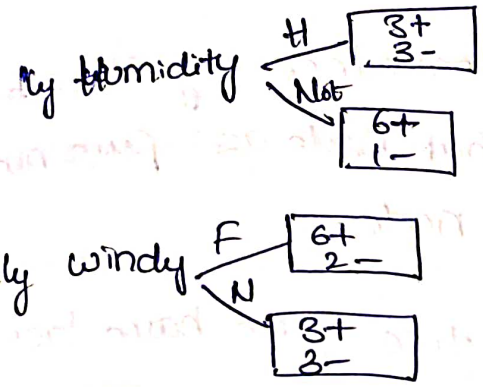
Step4

Repeat the same above steps and find the IG for every feature in the Dataset.

In this example

IG(Y, Temp) = □

Temperature
- n → 2+ 2- D1 = HD1(Y)
- m → 4+ 2- D2 = HD2(Y)
- c → 3+ D3 HD3(Y)



Now once we get the all the IG values

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

$IG(Y, \text{Temp}) =$ ____

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

$IG(Y, \text{windy}) =$ ____

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

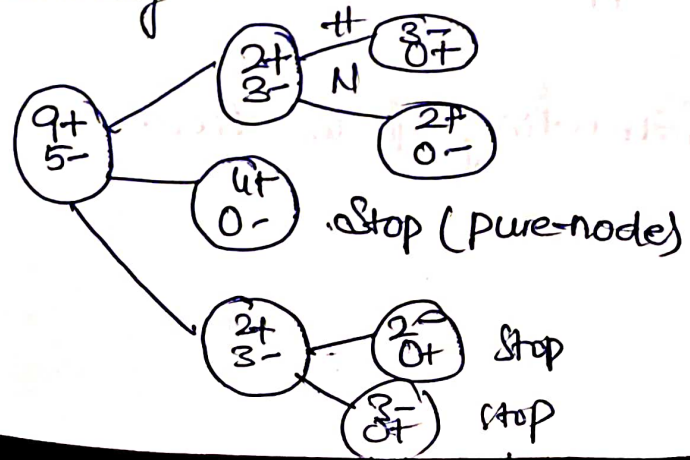
Choose the feature which has highest value and make that as parent node.

IG(Y, f) = entropy @ parent level

− weighted entropy @ Child level.

Step 5 :-

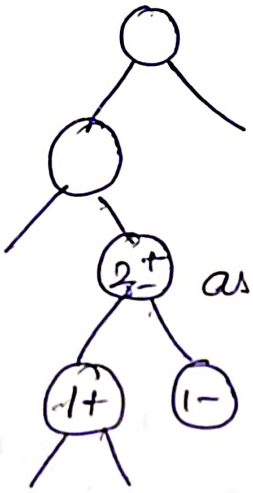
Recursively break each node using IG as the Criterion



The node which as maximum dominance by the class points in the feature we call that node as pure node and we need to stop @ that node.

and we also can't grow the tree if we have lack of points.

for example if we have very points @ a node



as there are only 2 points which are very fewer

∴ They are very fewer and If they are noisy points we end up with wrong o/

So Typically If we got any of the scenario's while constructing a decision tree

① pure node ✓ `num_samples_per_class.count(0)==0`

② few pts @ a node ✓ `if depth < max_depth and`

③ If we are too depp ✓

We need to stop constructing further tree.

* depth of the tree ↑ : Overfitting ↑ (few pts)

depth is small ⇒ underfit.

In decision tree hyperparameter is depth and which can findout by Cross Validation (CV).

34.8 Building a Decision Tree: Splitting Numerical features :-

Construct a Decision tree:-

For the above example we have used Information gain for splitting the node.

for finding Information Gain we can use Entropy Instead of Entropy we can also use Gini Impurity which is Computational efficient

IG: → Entropy
→ Gini Impurity.

Till now we have applied Decision tree for the Data which has Categorical features / Discrete RV.

If we have Numerical featured data.

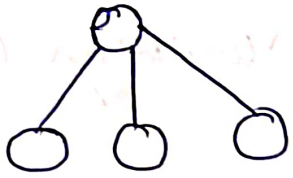
f_1	y
2.2	1
2.6	1
3.5	0
3.8	0
4.6	1
5.3	0

where f_1 : numerical
→ Integer(s) real valued

→ previously we tried to split the data based on categorical variable like If we have feature

f_2 : 3 categories.

Then we can split into



And If we have numerical data then for every value

there would be only one node. which is not preferable

~~Hack~~ General way to split based on numerical features:

Step 1:- Sort the Numerical feature in ascending order

f_1	Y
2.2	1
2.6	1
3.5	0
3.8	0
4.6	1
5.3	0

n-pts

Step2 Consider the features based on range

$f_1 < 2.2$
$f_1 < 2.6$
$f_1 < 3.5$
$f_1 < 3.8$
$f_1 < 4.6$
$f_1 < 5.3$

like this we get n possible variations.

→ Based on Threshold we divide the numerical features

$f_1 < T_1$ ⟨ D_1, D_2

$f_2 < T_2$ ⟨ D_1, D_2

$f_3 < T_3$ ⟨ D_1, D_2

⋮

$f_3 < T_n$ ⟨ D_1, D_2

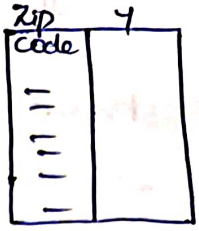
f_1	Y	
2.2	1	D_1
2.6	1	D_1
3.5	0	D_2
3.8	0	D_2
4.6	1	D_2
5.3	0	D_2

Like this we n possible splitting based on Criteria and Then we finalize nodes based on Information Gain value, the feature which has highest IG will be considered to be the node

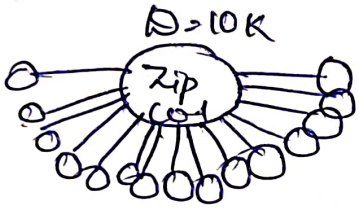
But this General process is the time taking process so we have some hacks to workaround.

34.10 Building a decision tree: Categorical features with many possible values:-

let us Consider we have a feature Pincode / Zipcode with 1000's of values.



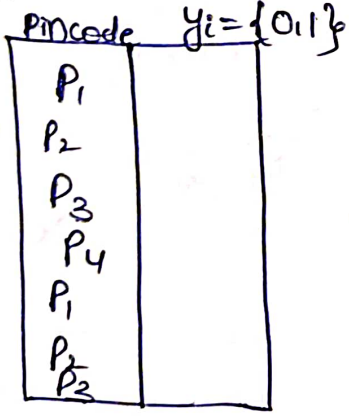
If we have like this we end up having thousands of node under a node



So we endup having thousands of node @ a level.

So to Overcome This

One of the hack is to do feature Engineering the Categorical data and convert it into Numerical data.



To convert it into numerical feature

$$P(y_i=1 \mid P_j)$$

This means no of time we get $y_i=1$ for a particular Pincode (P_j) (let be P_2) / total no of time P_j occured.

$= P(y_i=1 / P_j)$

$= \frac{\text{no of \# } y_i=1 \text{ for a given } P_j}{\# P_j}$

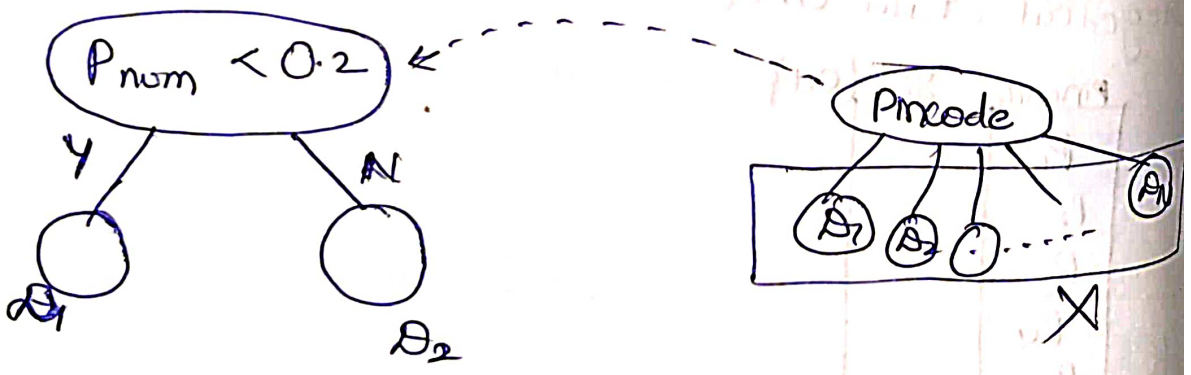
let we have 20 times P_j and in that 19 times $y_i=1 \Rightarrow 19/20$

P_j (Categorical feature) $\longrightarrow$ Numerical feature $= P(y_i=1 / P_j)$

D → remove pincode features

Numerical features

So by this we can make decision tree like



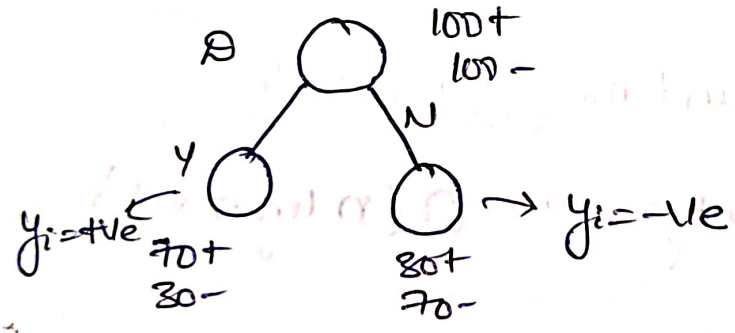
34.11 Overfitting and Underfitting

If the depth ↑!
possibility of having very few points @ a leaf node ↑

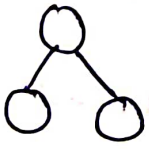
By this whole model becomes noisy, Overfitting my model to noise.

Interpretability of the model also ↓
because
if ( ) & ( ) & ( ) & ( ) ... get ↑
for a $x_q \rightarrow y_q$ by this Interpretability decreases.

similarly If the depth is very low (min depth would be 1)



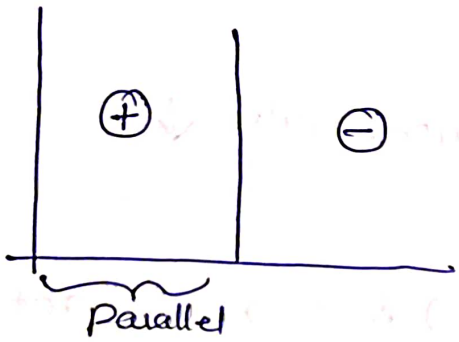
so by this we end up getting non of the node is pure so If we given a Query point, the class label y_q would take majority among them.

Note  This type of tree Called decision Stump.

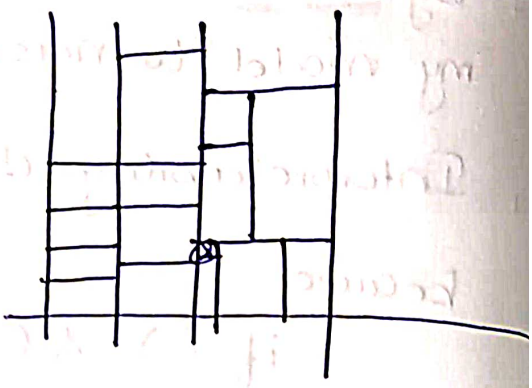
So if we go too deep → Overfit
too shallow → Underfit

Geometrically

Underfit



Overfit



So we use Cross Validation 'CV' to better determin
The depth to be taken into consideration.

34.12 Train and Runtime complexity :-

Train time Complexity ~ $O(n \log n * d)$

n = no of points Dtrain

d = dimensions

When we have numerical features It becomes mor
computational complex to solve, but there are
many algorithmic methods or hacks to solve.

n logn in the complexity is mostly used for
Sorting

After training @ Runtime the space needed to store my Decision Tree is by a simple nested if else is small.

simply In nested if else we use no of internal nodes + no of leaf nodes.

Typically we train Decision tree with depth 5 or 10.

as depth ↑ ⇒ Interpretability ↓

So the Runtime ~~time~~ space complexity is reasonable.

whereas Run time time complexity: would do K decisions where K is max depth of any leaf node.

run time time complexity: O(depth).

Decision ~~trees~~ are good

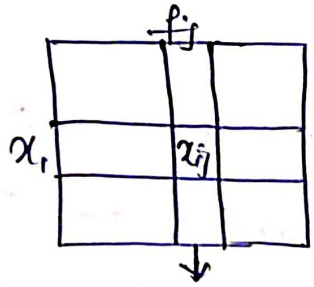
→ when we have large data, and dim is small

→ and also when we have to make low latency as the time complexity for DT is O(depth)

34.9 Feature Standardization :-

In the case of Logistic Regression, SVM, & kNN as all are distance based algorithms we need to perf feature Standardization to make all the values under a feature into same scale.

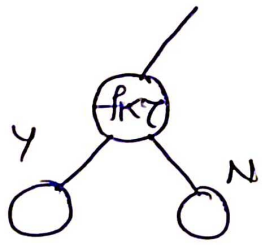
In Logistic Regression

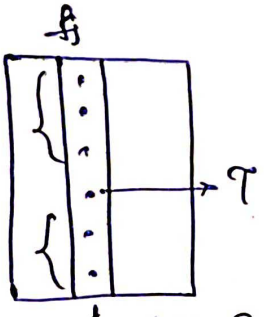


$$x_{ij}' = \frac{x_{ij} - \mu_j}{\sigma_j}$$

where as Decision trees are not distance based methods.

Based on Threshold value we define a node and split it based on $<, >, =$ values





↳ we Sort this column in order

So we do not need to perform feature Standardization, as it does not depends on scale rather it only consider the values Not on distances that ~~are~~ are based on scales.