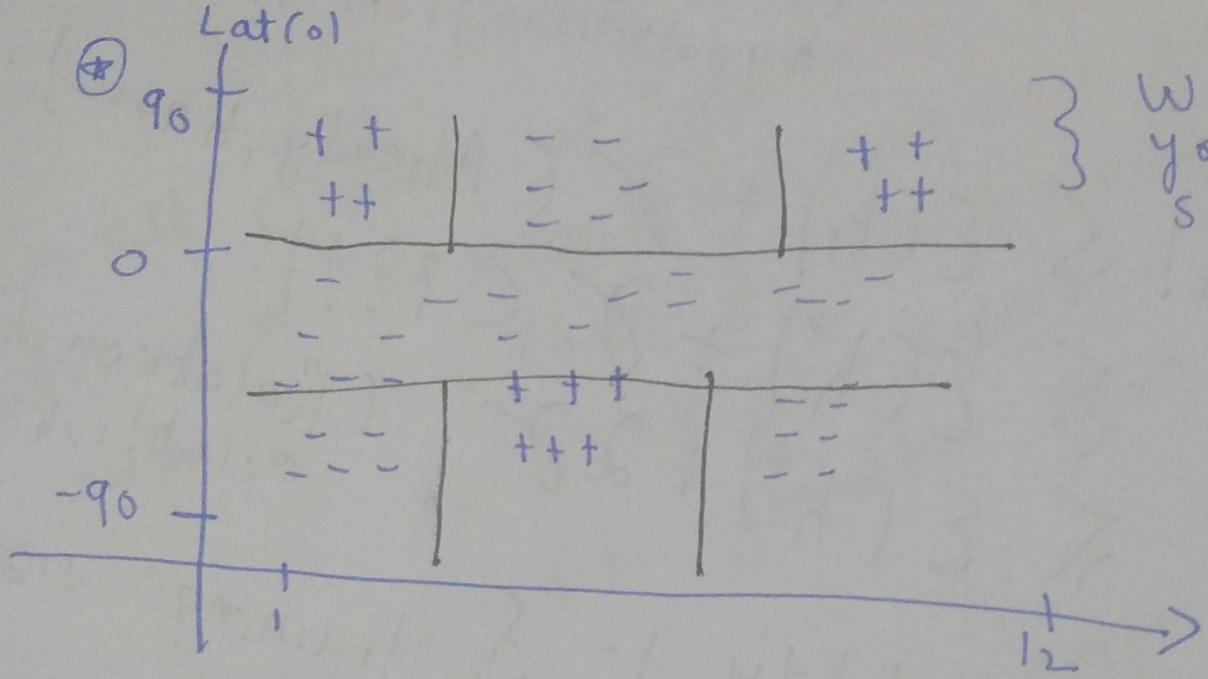
lecture-10

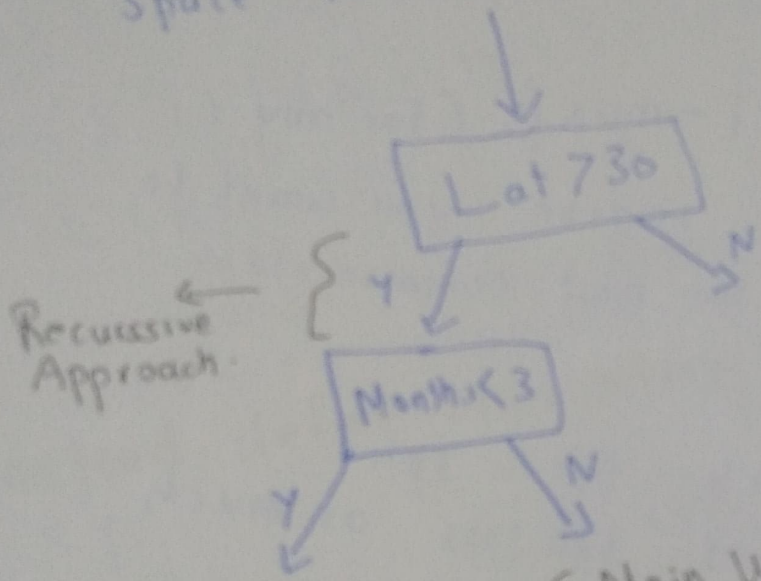
Decisions Tree:-



- Difficult to come up with linear decision boundry that is optimal
- Decision tree is optimal in this case.

→ Decision Tree works by greedy, Top-Down - Recursive Partitioning

→ Decision is gonna ask question in the Space to create isolated regions (Small Sptial regions).



→ More Formally :- (Main Working)

Region R_p

Looking for a split S_p

$S_p(j, t) = (\{X | X_j < t, X \in R_p\})$ R①

$= (\{X | X_j \geq t, X \in R_p\})$ R②

Feature to be used for split → j

Threshold → t

How to choose splits:-

Define $L(R)$ Loss on R

Given C classes, define $\hat{P}_c$ to be the propotion of examples in R that are of class c.

- $L_{misclass} = 1 - \max_c \hat{P}_c$ (For any Sub-divided region you want to pick up the most common class there).

- Want to pick the split which decreases the loss as much as possible:-

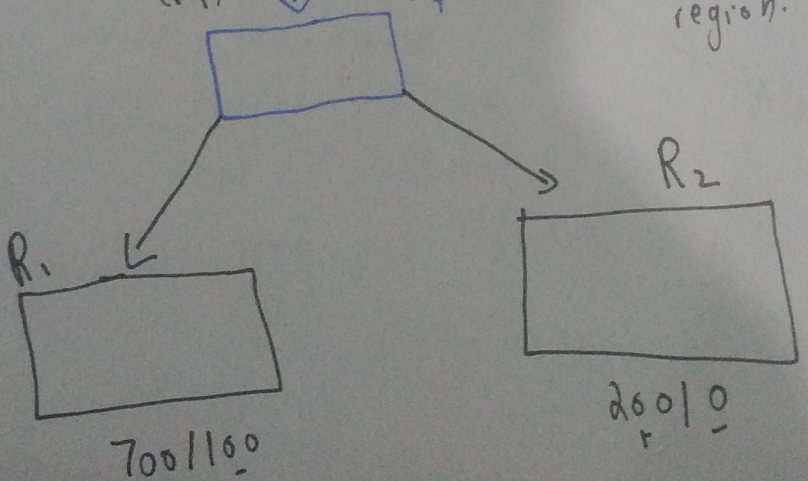
$$\max_{j,t} \; L(R_p) - \left(L(R_1) + L(R_2) \right)$$

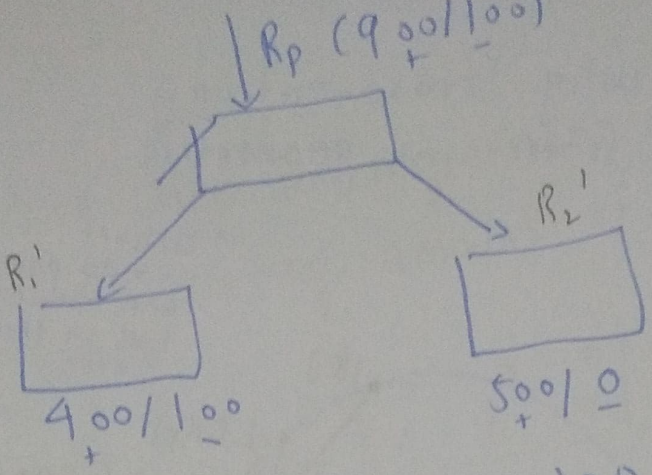
$L(R_p)$: parent loss

$L(R_1) + L(R_2)$: Children loss

Misclassification Loss has Issues:-

(R_p) $\downarrow$ $[400/100]$ → Number of examples in this region.





$L(R_1) + L(R_2) = 100 + 0 = 100$

$L(R_1') + L(R_2') = 100 + 0 = 100$

→ Loss Remains the same in above two splits.

$L(R_p) = 100$

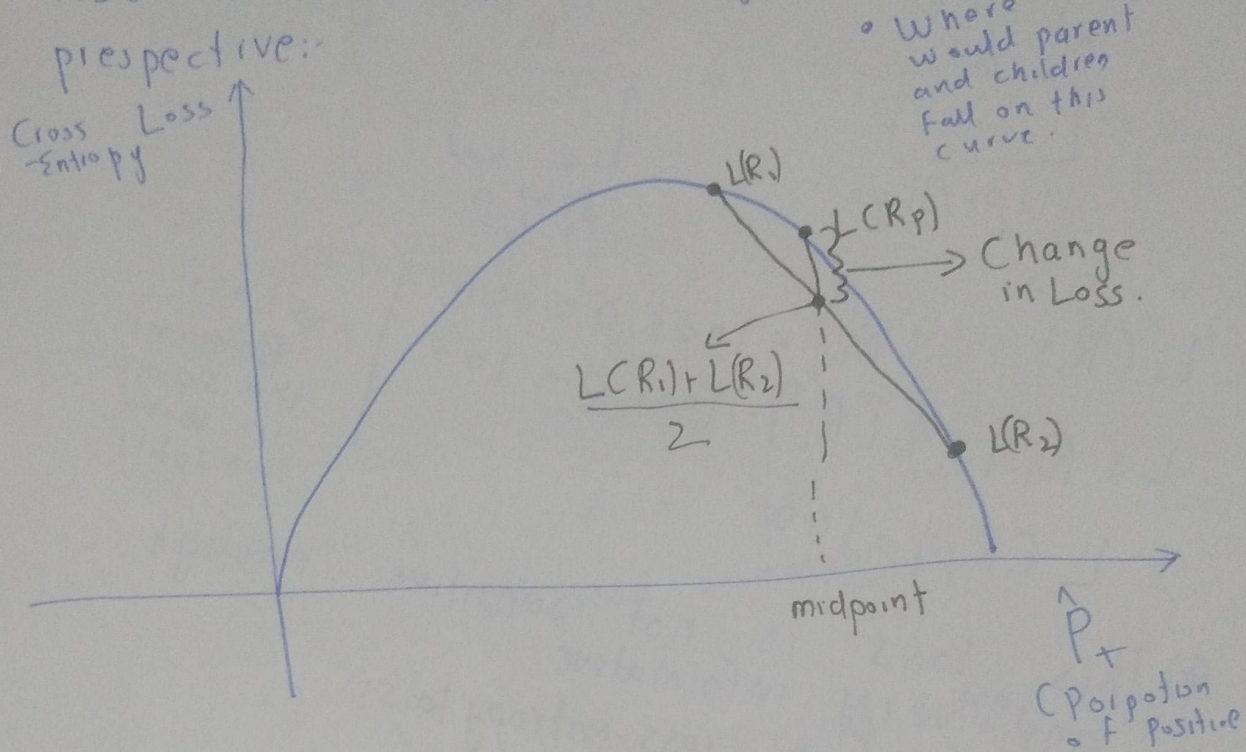
* This loss is not sensitive enough and not very informative

* Instead, define cross-entropy loss:

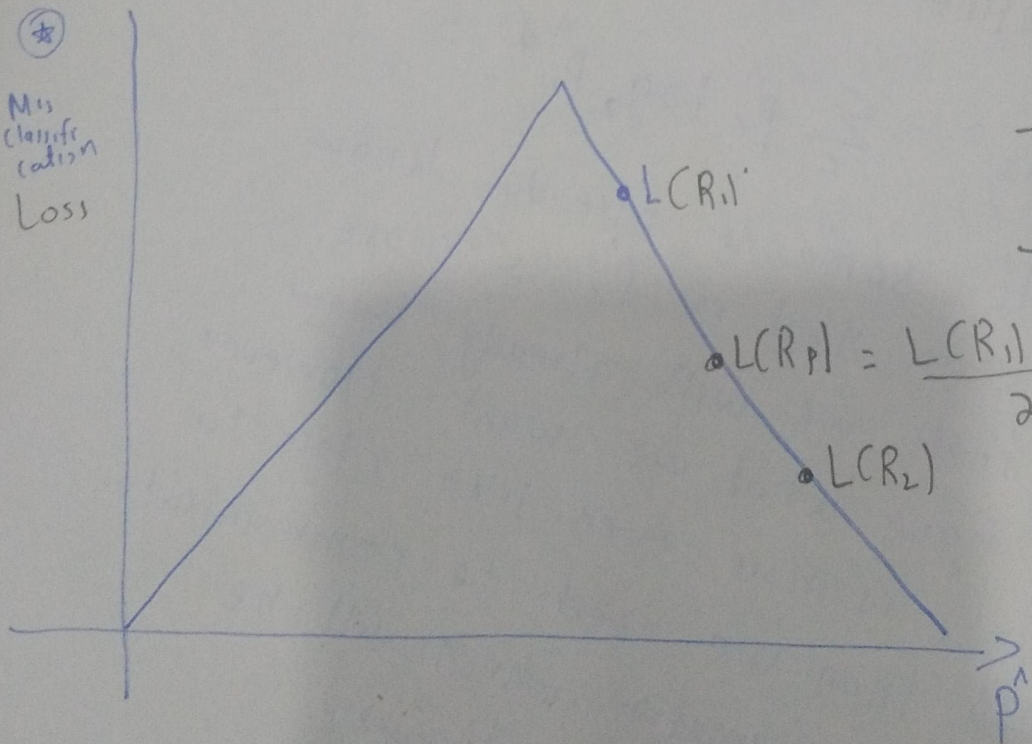
$$L_{cross} = -\sum_c \hat{P}_c \log_2 P_c \quad (\#) \rightarrow$$

If someone already know the class that is $\hat{P}_c$ is close one that we will already have what we want. However if $\hat{P}_c$ is more even like close to 0.5 than we need to communicate more information which will be visible by larger L_{cross}.

* We can see why cross-entropy loss is better from a geometric prespective:-



- Where would parent and children fall on this curve.



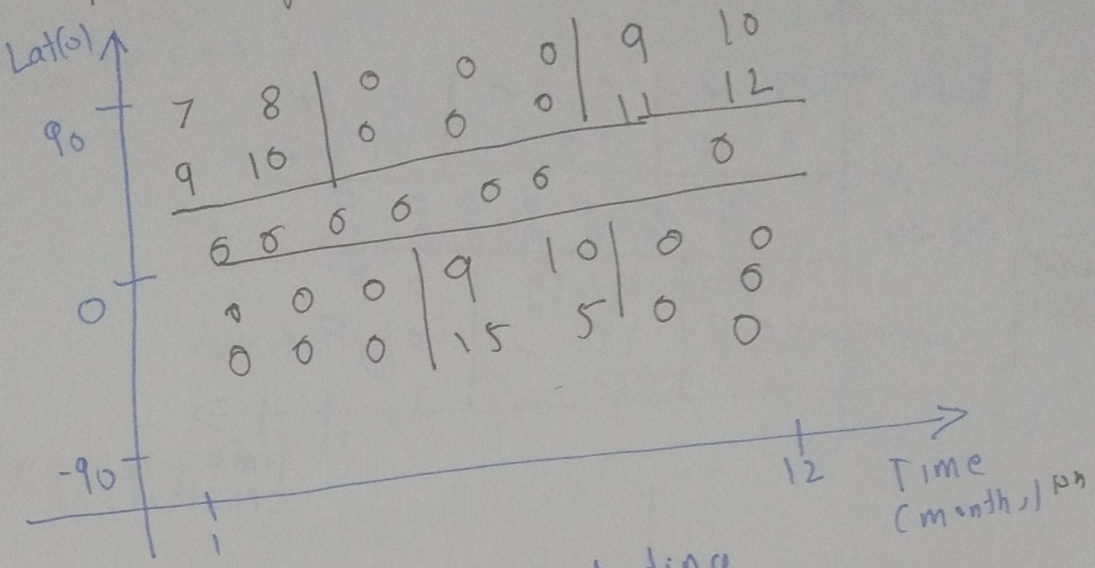
] → For misclassification loss its not very sensitive

* Gini Loss:-

$$\sum_{c} \hat{p}_c(1-\hat{p}_c)$$ ⇝ Curve is similar to Cross-entropy loss

(*) Decision Tree for regression:

→ Predicting amount of snowfall:- (in inches).



↳ Instead of predicting the majority of values → you can predict the mean of values present in the region.

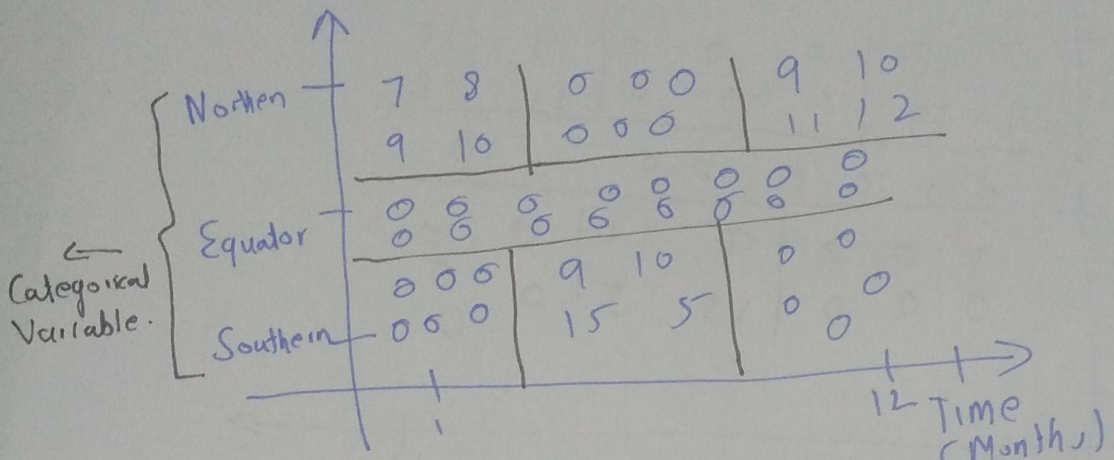
Sub space. R_m

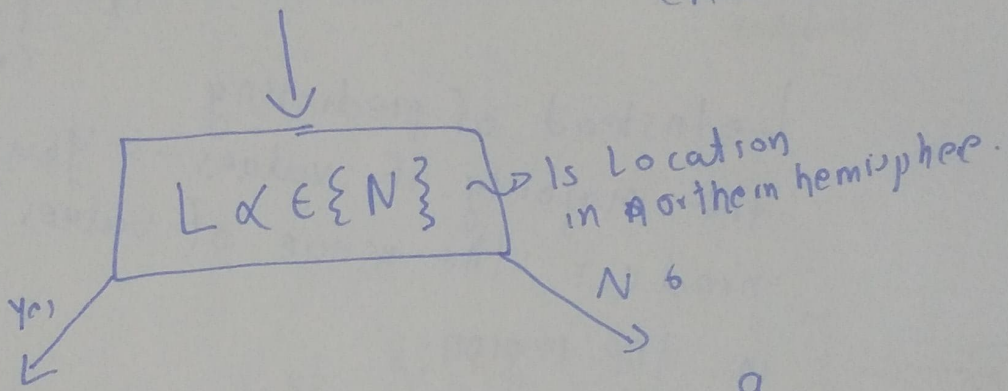
Predict $\hat{y}_m = \frac{\sum_{i \in R_m} (y_i - \hat{y}_m)^2}{|R_m|} y_i$ (Mean is the prediction).

$L_{square} = \frac{\sum_{i \in R_m} (y_i - \hat{y}_m)^2}{|R_m|}$ (Loss of specific region).

(117)

(*) Categorical Variables can be dealt efficently:-





→ However, remeber if you have 9 categories than you need to convider 2^9 splits which is computationally efficent.

* If you let your tree grow indefinately then you would have separate region for each point
  - ↳ Probably you are overfitting
  - ↳ Generally decision trees have tendency for high variance.

- Regularization of Decision Trees :-
  1) Minimum Leaf size → You stop splitting
  2) Max depth
  3) Max number of nodes
  4) Minimum decrease in Loss
     - ↳ You might stop too early.
  5) Grow the full Decision Tree and then Prune it backwards. (By using misclassification with validation Set).

⊛ Runtime:

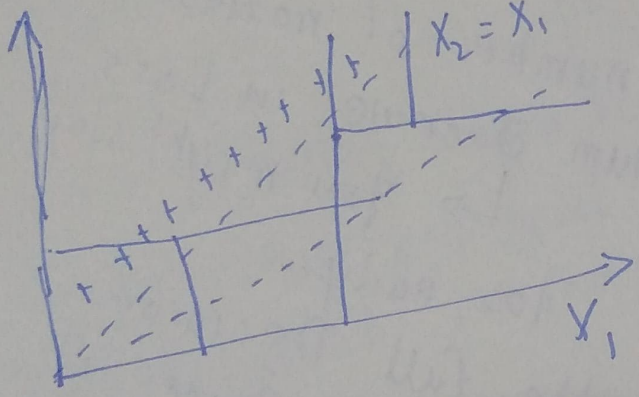
→ n examples
→ f features
→ d depth

Test time: → Pretty Quick
O(d) → Runtime only your depth; typically $d < \log n$

Train Time:

→ Each point is part of O(d) nodes
Cost of point at each node is O(f).
Total Cost: O(nfd)
↳ Not that slow

- Decision trees **don't have additive structure:**



- Decision tree have problem of over complicating when features follow an additive structure.

⊛ Recap:

⊕ Benefits:
- ⊕ Easy to explain
- ⊕ Intrepretable
- ⊕ Can deal with categorical variable
- ⊕ Fast

⊖ Negative:
- ⊖ High Variance
- ⊖ Bad at additive
- ⊖ Low predective accuracy.

④ You can make decision tree stronger through **ensembling**

→ Ensembling:-

→ Take X_i's which are random variable (RV) that are independent identically distributed (IID's)

→ $Var(X_i) = \sigma^2$; $Var(\bar{X}) = Var\left(\frac{1}{n}\sum_i X_i\right)$

$= \frac{\sigma^2}{n}$

$\bar{X}$ ↓ Mean of many of these variables

n ↓ Increase in Y_i's decrease variance

→ What if you drop the independent assumption

So now X_i's are id(identically distributed)

Y_i's correlated by ρ

$$Var(\bar{X}) = \rho\sigma^2 + \frac{1-\rho}{n}\sigma^2$$

↳ This shows we want more examples but make sure that as many examples are decorrelated.

⊙ Different Ways to ensemble:-

1) Different Algorithm
2) Different Training Sets.
3) Bagging → Random Forest
4) Boosting → Ad Boost, Xg boost

Our Focus.

⊛ Bagging:-

⊛ Boostrap aggregation
↳ Typical used in statistic to measure the uncertainty of your estimate.

→ What is Bootstrap aggregation?

→ Have a true population P
→ Training set S ~ P (S sampled from P).

→ Assume your population is training sample (S = P).
→ ↳ Bootstrap Samples $\boxed{Z}$ ~ S:-
↳ You sample N example from your training set S with replacement.

→ Train your models on Z and see the variability that model is making based on different bootstrap examples.

→ We are going to bootstrap sample, train different models and average their output (Bootstrap aggregation).

→ Lets make it more formal:

- Bootstrap Samples $Z_1, \dots Z_M$
- Train model G_m on Z_m.

$$G(m) = \frac{\sum_m G_m(x)}{M}$$

→ Aggregating output of all models.

→ Total Number of Models

→ Bias-Variance Analysis:

Remeber

$$Var(X) = \rho\sigma^2 + \frac{1-\rho}{N}\sigma^2$$

- Bootstrapping is driving down ρ
- Bootstrap decreases M.
  → Less Variance in your model (Less Overfitting).
- But you are increase the bias of model because you are training on less data (Each model which reduces its complexity).

Decrease in variance is greater than increase in Bias!!!

⑦ DT + Bagging :-

→ Recall :- DT are high vairance & low bias → Suitable for Bagging (Ideal fit).

→ Random Forest ⇒ DT + Bagging but it includes more randomization in each tree.

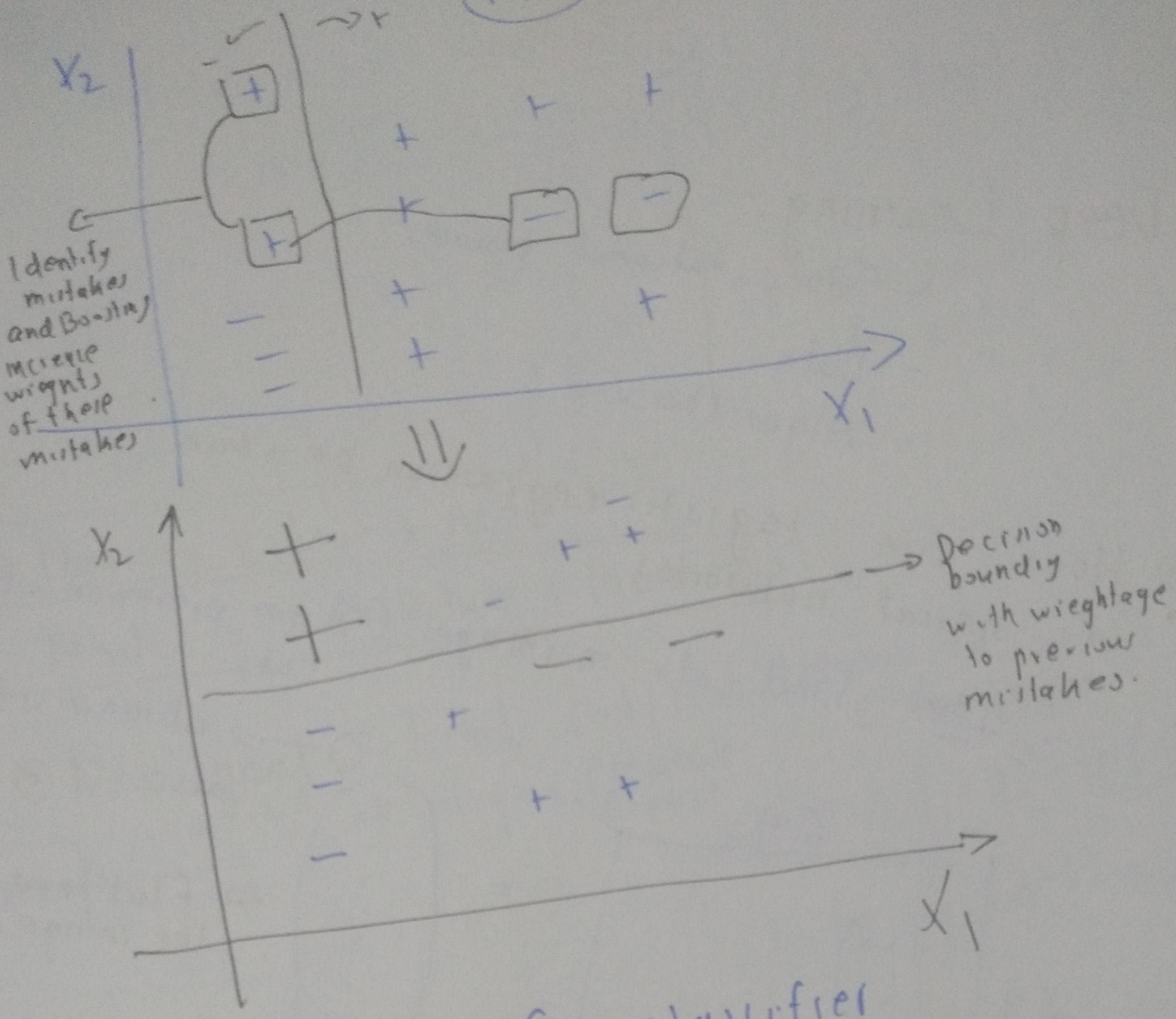
↳ At each split, consider only fraction of your features. (total features) · (Decrease correlation p) → Make sure different DT in random forest are decorrelated.

⑧ Boosting :-

→ Decreasing Bias of your model

→ More additive in nature

→ You train one and you add that prediction into your ensemble.



→ Determine for classifier G_m a wieght α_m (wieght for each model trained, higher α_m for good model, lower α_m for bad model).

↳ In adaboost for example α_m is porpotional to: $\log\frac{1-err_m}{err_m}$

Finally:

$$G(x) = \sum_m \alpha_m G_m$$

Each G_m trained on re-wrighted training set.