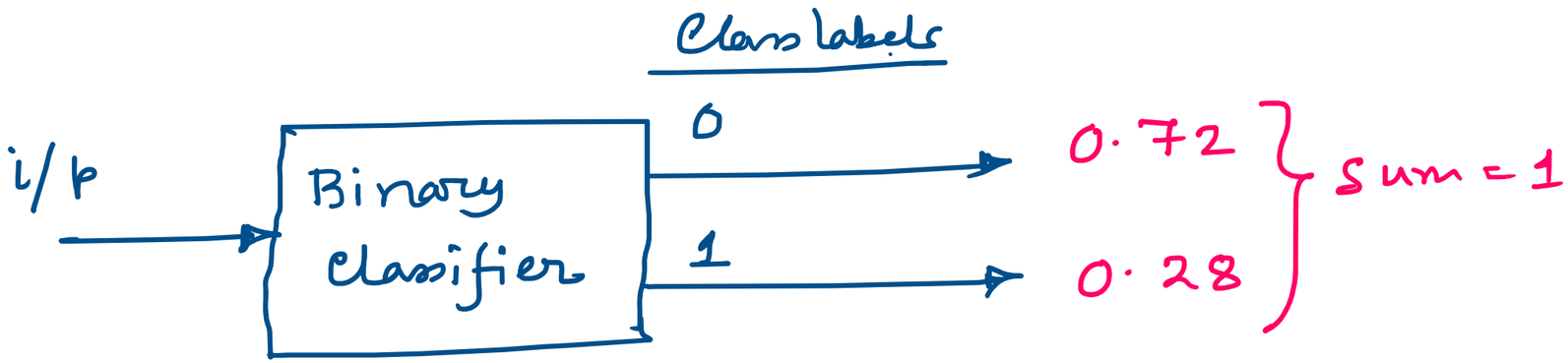
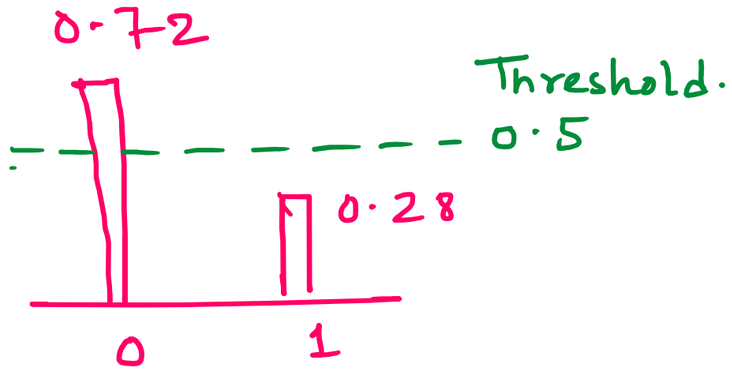
Binary Vs. Multiclass Classifier :-



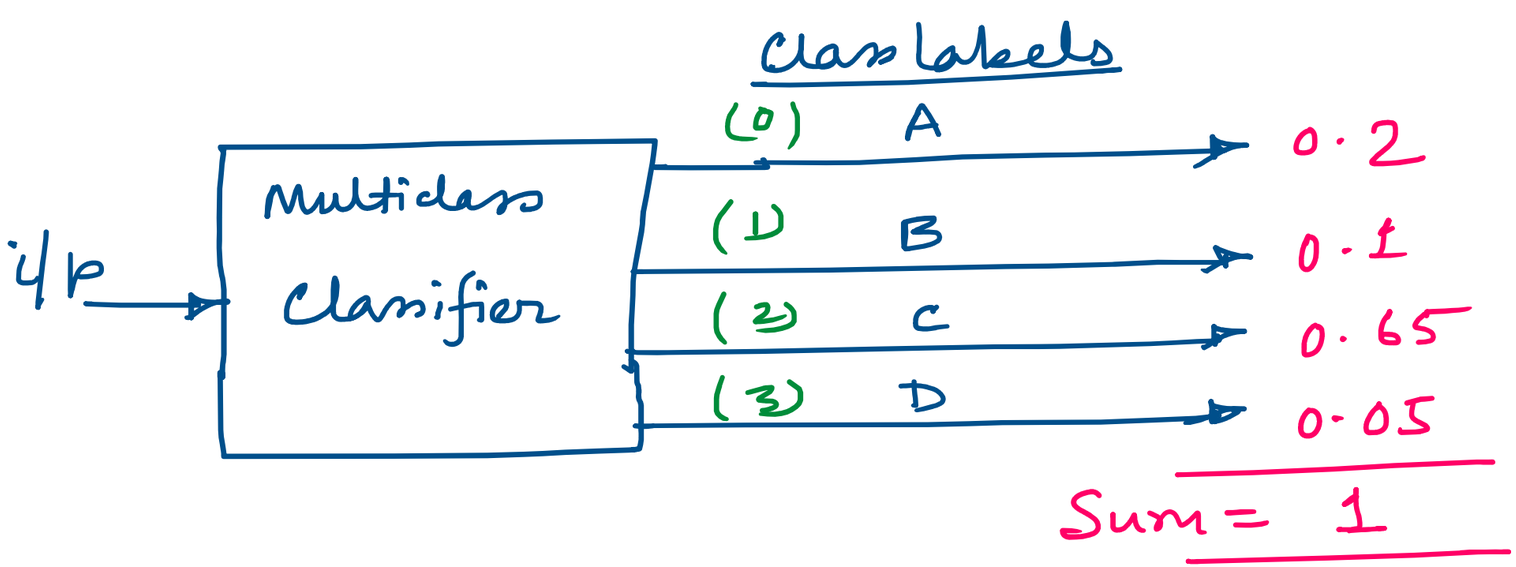


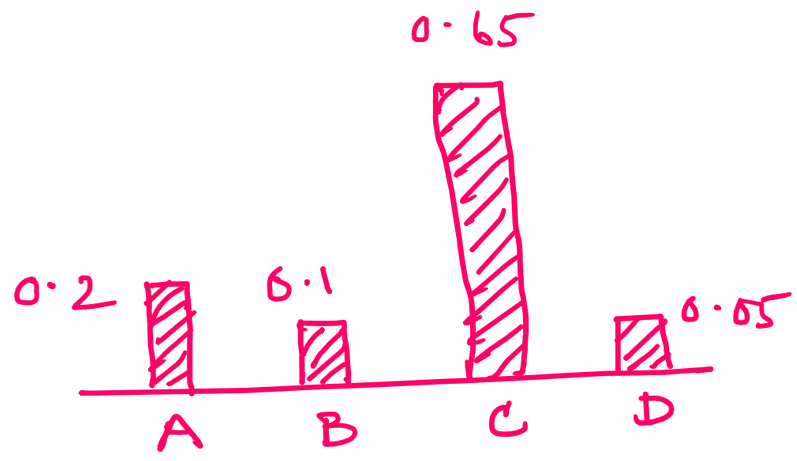
clf.fit(x_train, y_train)

clf.predict_proba(X_test) ⟶ [0.72, 0.28]

clf.predict(X_test, th=0.5) ⟶ 0

Note: If we change the threshold that might affect classifiers performance.





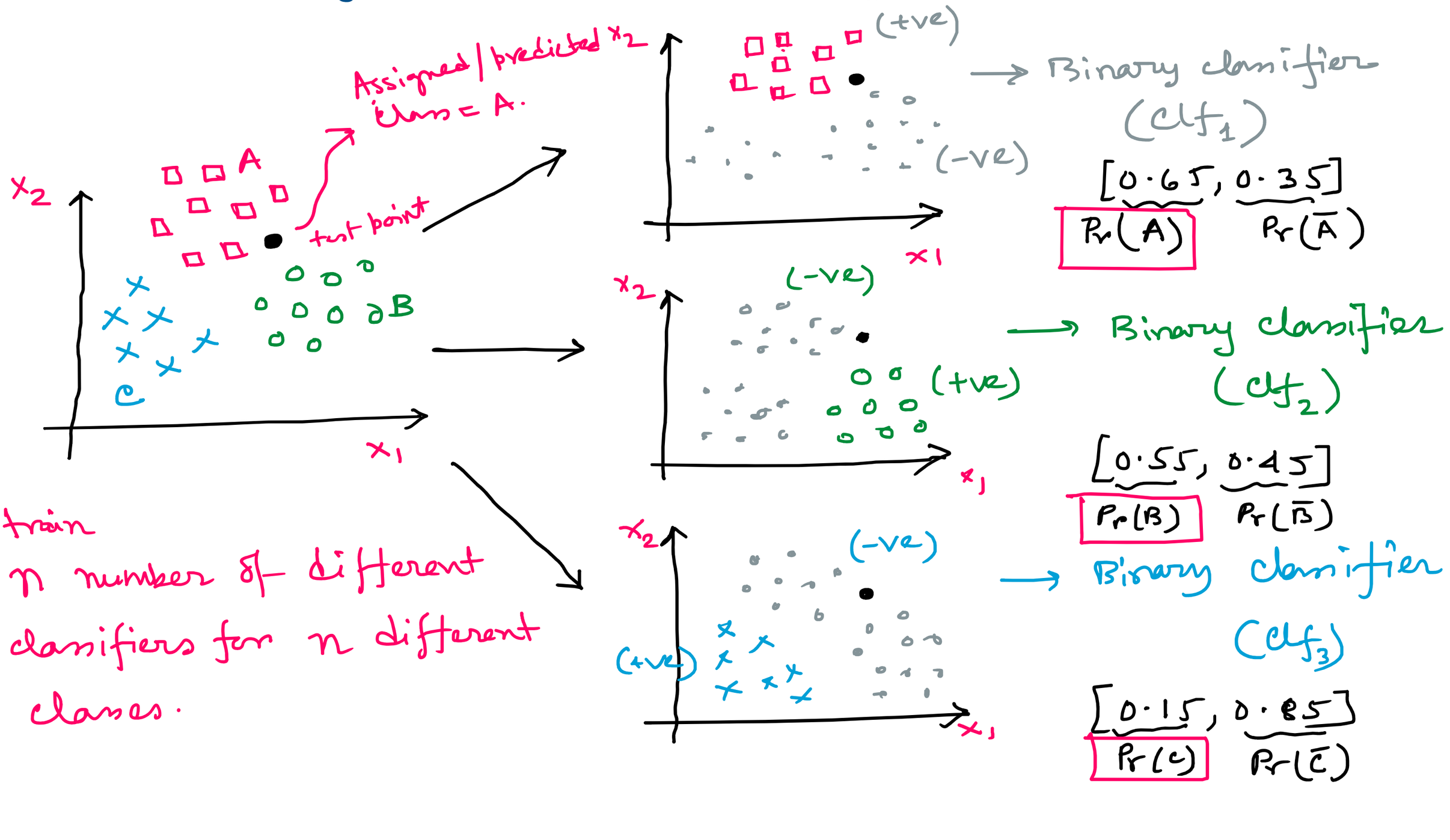
clf. fit(X_train, y_train)

clf. predict_proba(X_test) → [0.2, 0.1, 0.65, 0.05]

A B C D
(0) (1) (2) (3)

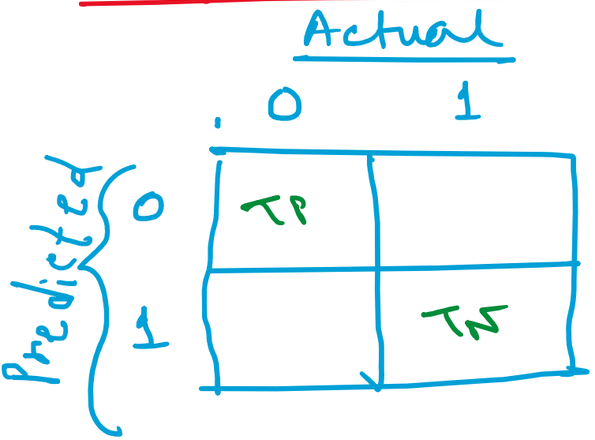
clf. predict(X_test) → 2 → class-c

One_vs_Rest (OVR) Multiclass classification



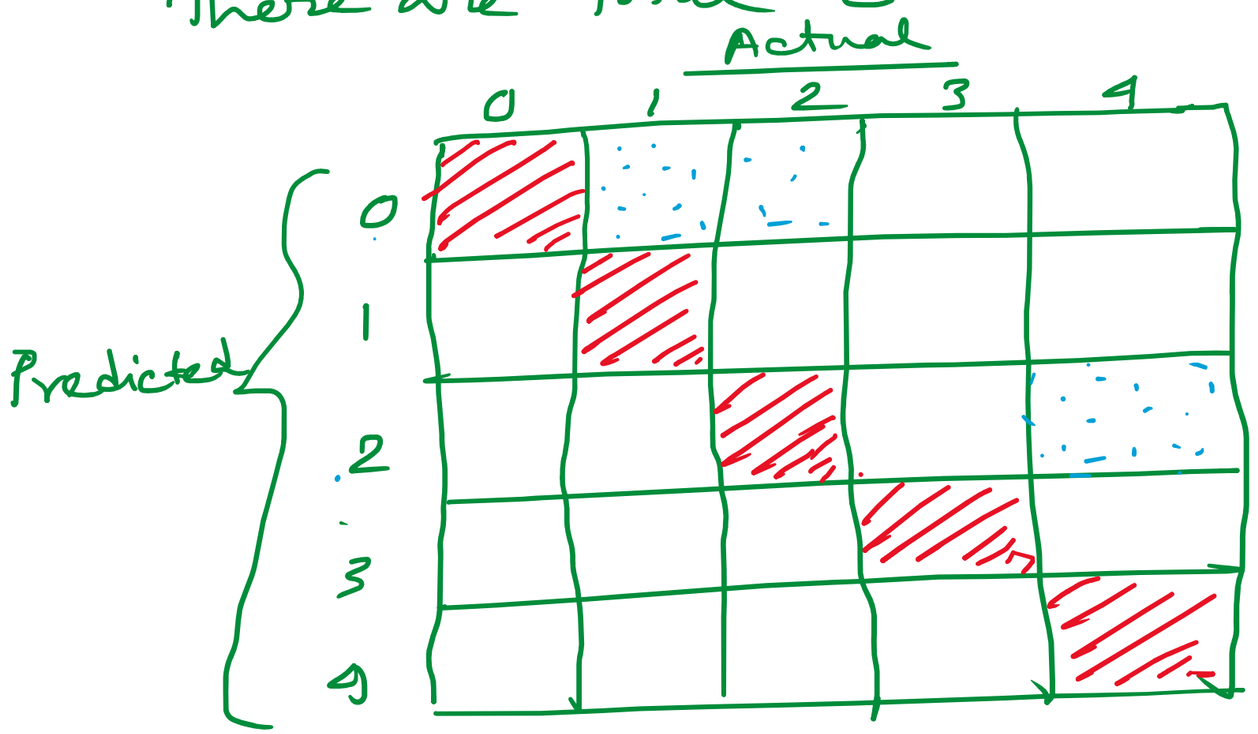
train
n number of different classifiers for n different classes.

Confusion Matrix for multiclass classifier

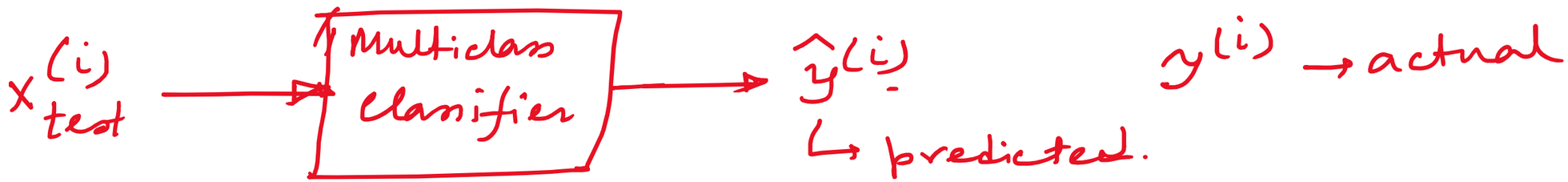


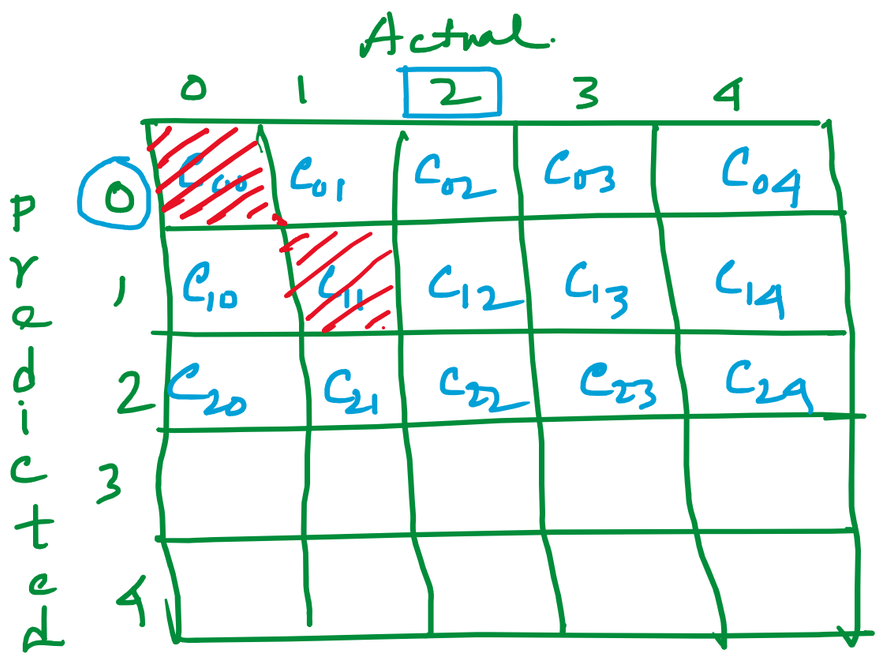
2×2 matrix for binary clf.

There are total 'C' classes (C = 5 for ex.)



(i, j)
↓
no of outputs whose predicted class label is 'i' & Actual class label is 'j'.





C_{03} = number of instances which are incorrectly classified as '0' but actually belong to class-3.

C_{0j}, $j \neq 0$ = number of instances which are classified as '0' but actually belong to class-j

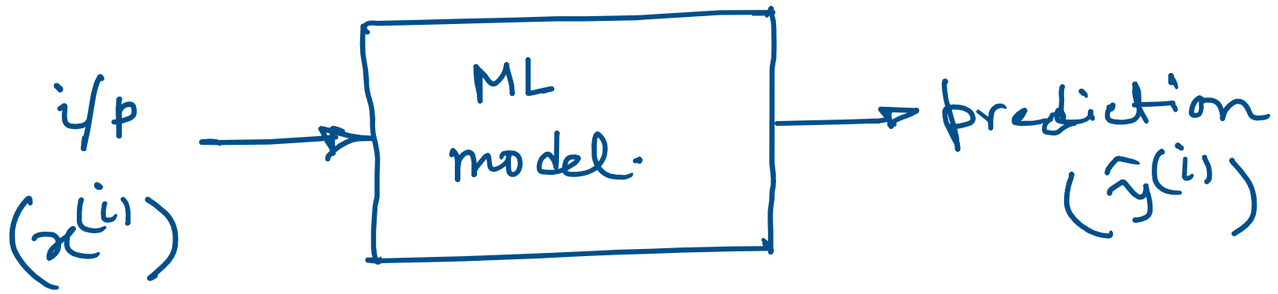
C_{ij}, $i \neq j$ = number of instances which are classified as 'i' but actually belong to class-j

$\sum_{j, j \neq i} C_{ij}$ = total number of instances those are wrongly classified as class-i

$$\text{Precision of- Class}(i) = \frac{C_{ii}}{C_{ii} + \sum_{j, j \neq i} C_{ij}} = \frac{C_{ii}}{\sum_{j} C_{ij}}$$

$$\text{Recall}(i) = \frac{C_{ii}}{\sum_{i} C_{ij}}$$

Gradient Boosting :-



Suppose considering regression problem.

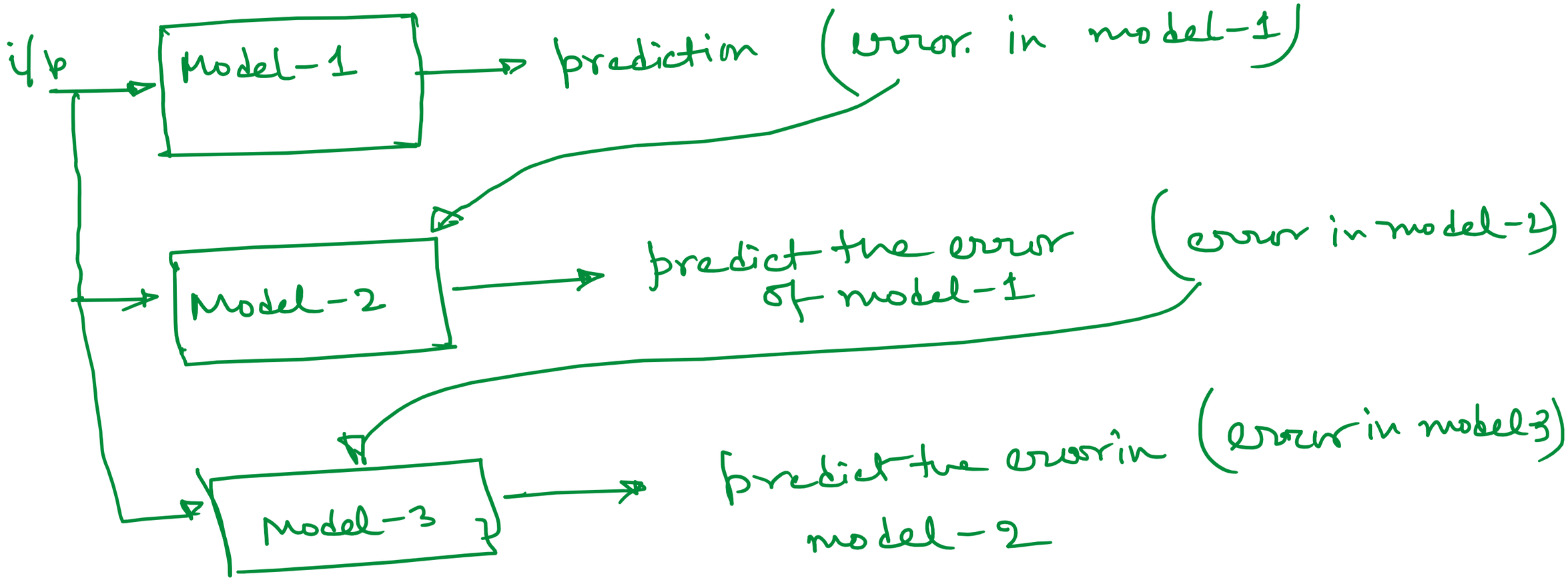
mean square error $(e^{(i)}) = \frac{1}{2}\left(y^{(i)} - \hat{y}^{(i)}\right)^2$

$$\frac{\Delta e^{(i)}}{\Delta \theta} \propto \left(\hat{y}^{(i)} - y^{(i)}\right)$$

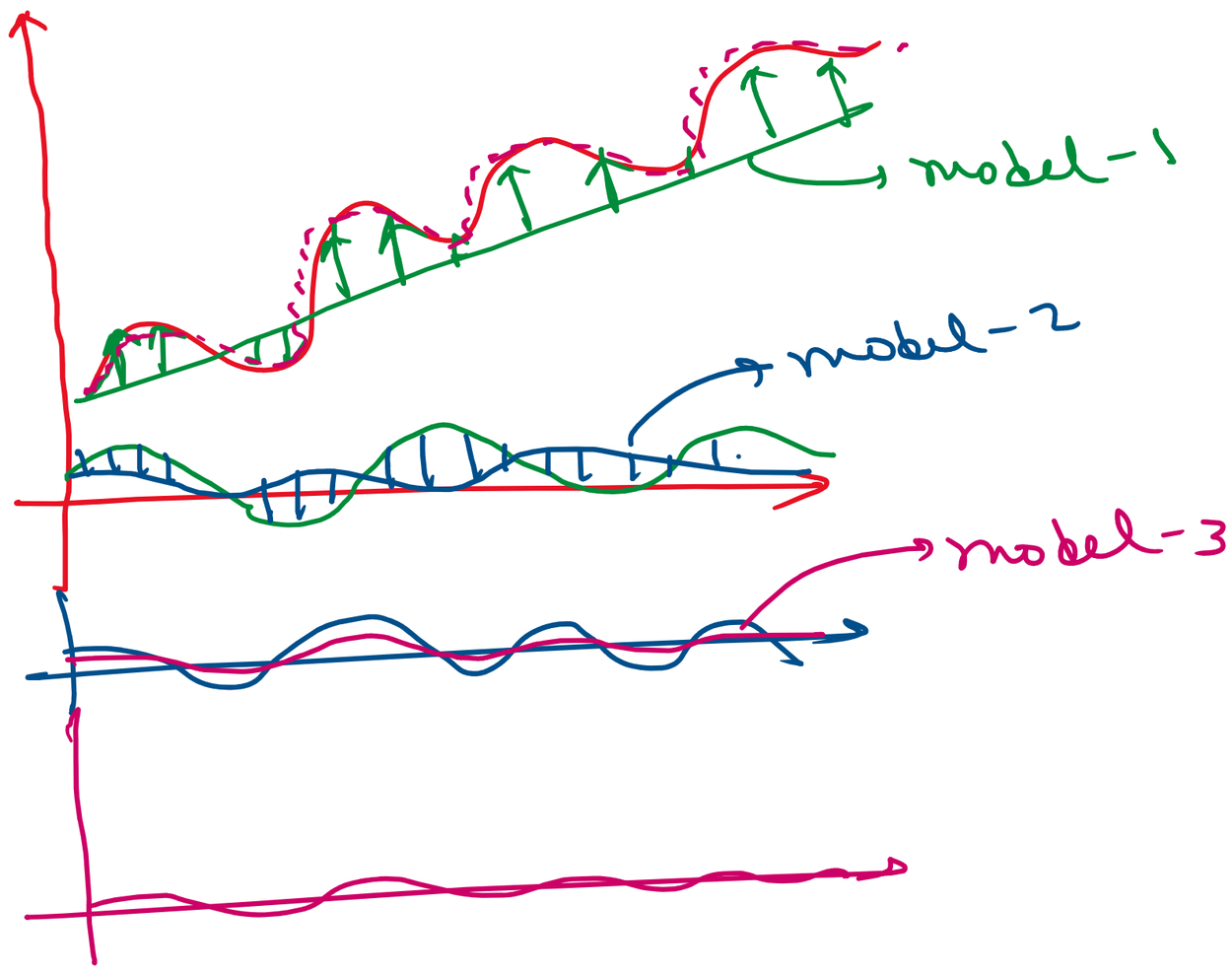
$$\Delta e^{(i)} = -\left(y^{(i)} - \hat{y}^{(i)}\right)\left(\frac{\partial \hat{y}^{(i)}}{\partial \theta}\right) \times \theta$$

$$= \left(\hat{y}^{(i)} - y^{(i)}\right) \frac{\partial \hat{y}^{(i)}}{\partial \theta} \cdot \Delta\theta$$

difference between actual & predicted (error in prediction)

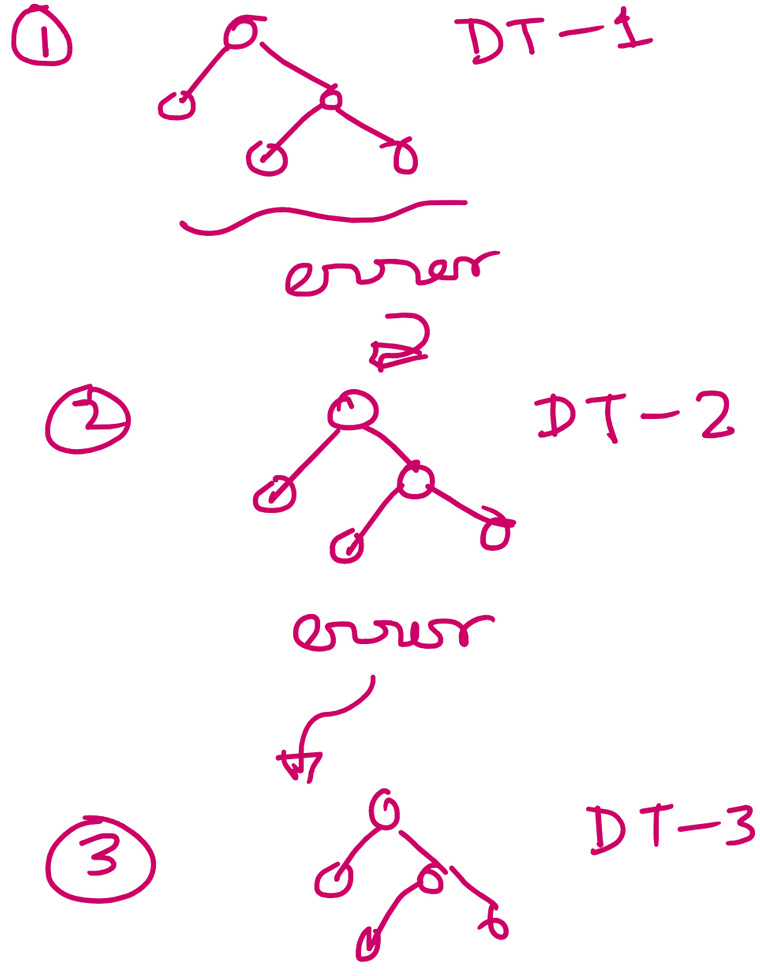


Model : $\text{Model-1} + \underbrace{\eta}_{\text{learning rate.}} \times \text{Model-2} + \eta \times \text{model-3} + \cdots + \eta \times \text{model-k}$



Gradient Boosted Trees.

For clarification:-



eXtreme Gradient Boosting (XGBoost)

→ XGBoost is a very high performing ML algorithm that outperforms most of the traditional ML algorithms in case of tabular data.

→ We have both XGBoost Regressor & XGBoost Classifier.

Salient features of XGBoost :-

1. Regularization: XGBoost uses regularization (L1, L2) to reduce overfitting.

   i.e. Why XGBoost is called regularized boosting technique.

2. Parallel Processing :- GBM is sequential, where as XGBoost core algorithm implements parallel processing. & is very fast.

The parallel processing is actually happening in building the tree (not in boosting).

parallelize → node building at each level.

parallelize split finding on each node.

3. High flexibility: XGBoost allows users to define custom optimization objectives & evaluation criteria.

4. Handling missing values :- XGBoost can handle missing values by inbuilt algorithm.

5. Tree Pruning , 6. It has Built in cross validation