**Solution Sheet**

1. Which model have you used for Total IPL 2020 Runs

prediction for each player? Explain your model.

**XGBoost (Regressor) model is used for the total ipl 2020 runs predication for each player**

**XGBoost is a ensemble technique**

We regularly come across many game shows on television and you must have noticed an option of “Audience Poll”. Most of the times a contestant goes with the option which has the highest vote from the audience and most of the times they win. We can generalize this in real life as well where taking opinions from a majority of people is much more preferred than the opinion of a single person. Ensemble technique has a similar underlying idea where we aggregate predictions from a group of predictors, which may be classifiers or regressors, and most of the times the prediction is better than the one obtained using a single predictor. Such algorithms are called Ensemble methods and such predictors are called Ensembles.

**Types of Ensemble technique**

1. Bagging
2. Stacking
3. Boosting
4. Voting

Since Xg boost is a boosting technique, here we discus about boosting

**Boosting**

Boosting is an ensemble approach(meaning it involves several trees) that starts from a weaker decision and keeps on building the models such that the final prediction is the weighted sum of all the weaker decision-makers.

The weights are assigned based on the performance of an individual tree.

Ensemble parameters are calculated in \*\*stagewise way\*\* which means that while calculating the subsequent weight, the learning from the previous tree is considered as well.

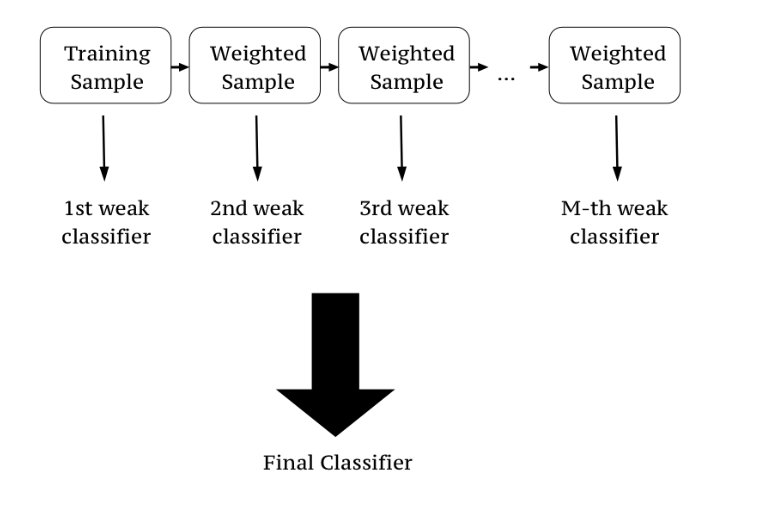


Fig 01 :- Bosting \_Basic Example

From Fig 01 we can observe that the 1st training sample has some weights , these weights are carried for subsequent samples and the decision of a 2nd sample taken based on the 1st sample weights . Till Mth samples similar things happen and final output is given

**XGBoost Intution**

It was developed by Tianqi Chen in C++ but now has interfaces for Python, R, Julia.

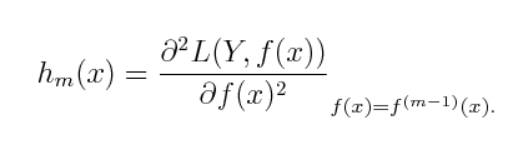
XGBoost's objective function is the sum of loss function evaluated over all the predictions and a regularisation function for all predictors (j trees). In the formula means a prediction coming from the tree.

Loss function depends on the task being performed (classification, regression, etc.) and a regularization term is described by the following equation:

First part is responsible for controlling the overall number of created leaves, and the second term watches over the scores.

Mathematics Involved

Unlike the other tree-building algorithms, XGBoost doesn’t use entropy or Gini indices. Instead, it utilises gradient (the error term) and hessian for creating the trees. Hessian for a Regression problem is the \*number of residuals\* and for a classification problem. Mathematically, Hessian is a second order derivative of the loss at the current estimate given as:

<

where L is the loss function.

- Initialise the tree with only one leaf.

- compute the similarity using the formula

Similarity=

Where is the regularisation term.

- Now for splitting data into a tree form, calculate

Gain= left similarity +right similarity -similarity for root

- For tree pruning, the parameter is used. The algorithm starts from the lowest level of the tree and then starts pruning based on the value of $\gamma$.

If Gain-$, remove that branch. Else, keep the branch

- Learning is done using the equation

New Value= old Value+ ξ \* prediction

Where is the learning rate