Thursday Learning Hour

eXtreme Gradient Boosting (XGBoost)

Prepared and Presented

by

Prabakaran Chandran

22 Oct 2020

Story Begins here:

Meet Mr. Bert , works as CEO of Namma Bengaluru Metro Water Corporation

He wants to estimate the Drinking water demand based on Population Density , so that he can plan water supply schemes

Bert has:

- **Historic Dataset of Areas, Population density , Water supply**

Bert has tried :

- **A Decision Tree algorithm**
- **But not optimal results**

Bert wants:

- **To implement and learn about XGBoost**

Learning starts here:

What’s the problem with the Single Predictor (say Decision tree , Linear Regression)

- High bias ,High Variance (Remember Bias Variance trade off)
- High bias is the problem in Decision tree
- Low performance on new data

Ok , then train multiple models? Will it work?

We can train multiple models instead of single models in parallel or sequential

Terminology alert: Ensemble Learning.
1.Bagging , 2. Boosting , 3. Stacking , 4. Blending.

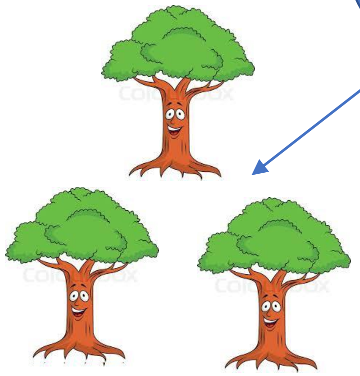
Journey for the Day : Boosting --> Gradient Boosting --> Extreme Gradient Boosting

Ensemble learning:

Weak learners/ predictors

Now, we have multiple learners , how can we train and test?

Bert might think this is a forest



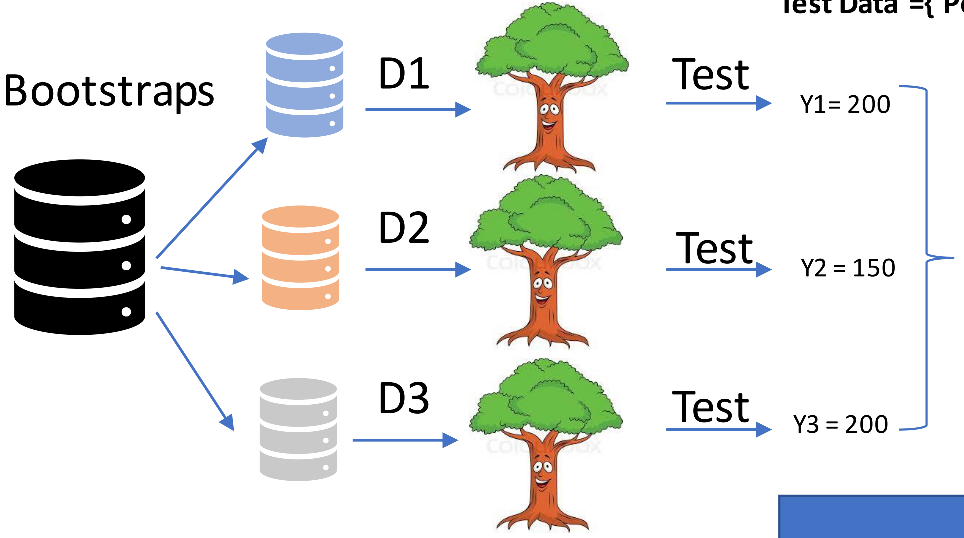
Train

Training Data

Population Density (K)	**Subblock**	**Block**	**Water Demand K.liters**
1.6	Urban	A	350
1.6	Urban	B	400
1.5	Rural	B	260
1.4	Rural	A	100

How? We have multiple ways --

Test Data ={ Pop Dens:600 , No of Conn:230}



Here we have a strong predictor
Aggregate
Maximum Votes for classification for our case - Regression Averaging

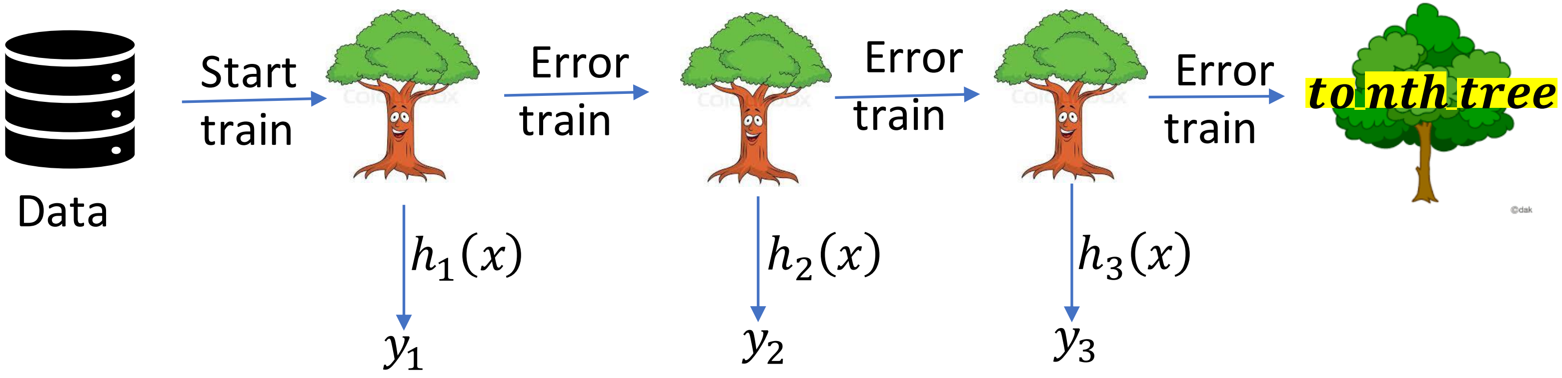
Terminology Alert:

Bootstrapped Aggregation (Bagging)
Parallel Training
Example : Random Forest

Bootstrap----------------Training

Bootstraps : Sampling with Replacement

If Bagging is Parallel , what about Boosting? Learn from failures..



Boosting:

- The idea of boosting is to train weak learners sequentially, each trying to correct its predecessor.
- Error is being corrected by weights or gradients (based on type of boosting)
- Ada Boost(mAdaBoost) , Gradient Boost { XGBoost , LGBoost, CatBoost},BrownBoost, LogitBoost

Math Alert:
(Optional)
Final function:

$$\boldsymbol{F_{(x)} = \sum_{T=0}^{n} \alpha h_i(x)}$$

α $-$ *learning rate* (0 *to* 1) to emphasize weak learners

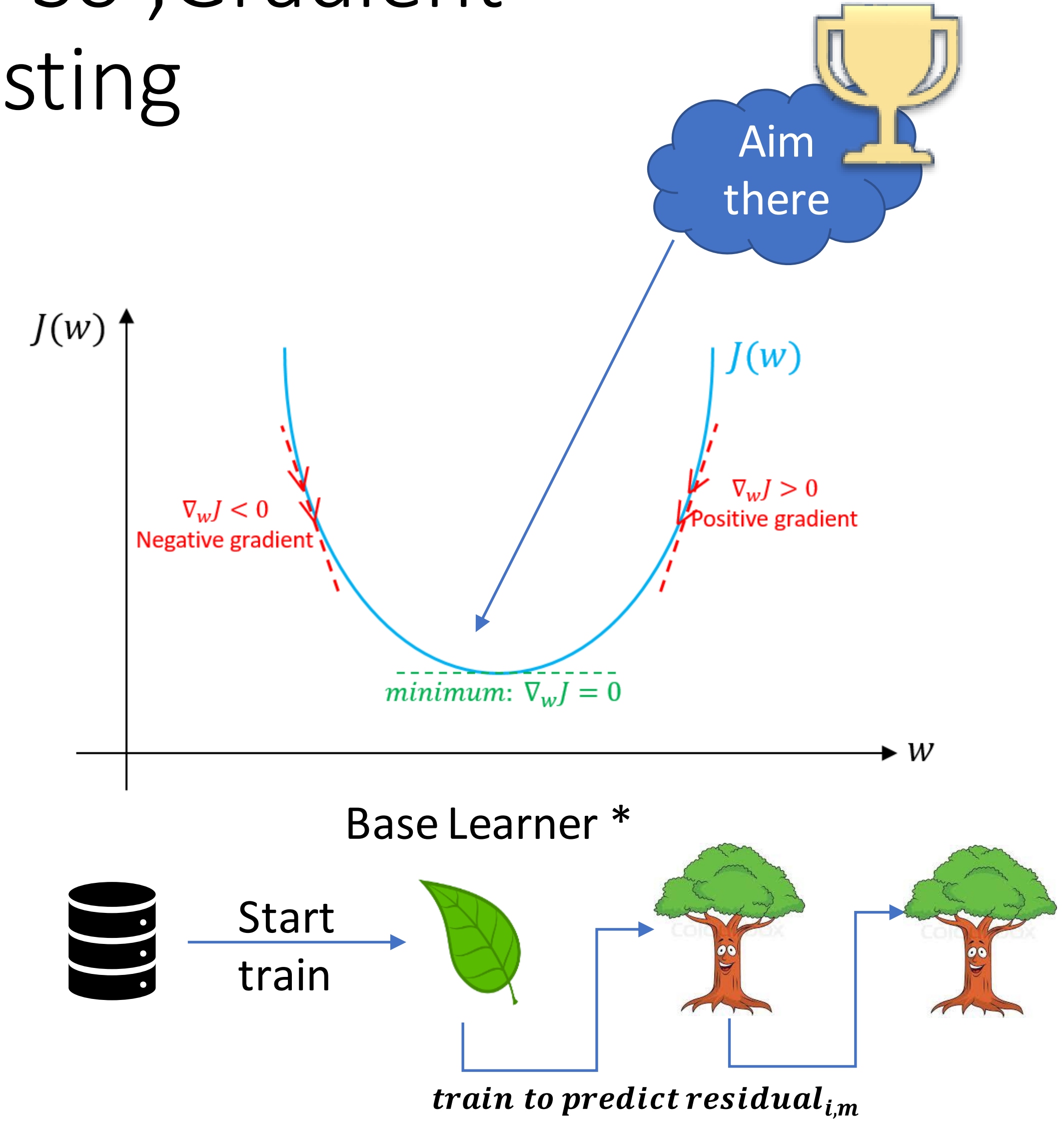
Loss or error or objective

$$\boldsymbol{J = \sum_i^n \mathcal{L}(y_i, y_i^p)}$$

y_i $-$ *actual* ,
y_i^p $-$ *predicted*

Let's add Gradient Descent to this? So ,Gradient Descent + Boosting = Gradient Boosting

- We know , even Bert Knows main Objective of all the ML algos is to reduce the Loss !
- One of the method is Gradient Descent ! Gradient : first order derivative of Loss (Slop of the curve , defines direction to minimal point)
- Let's take a derivative of a loss : (Math Alert!)
  - Loss = ½ (actual – predicted)^2 :: ½ MSE
  - $\frac{\partial L}{\partial y_p} = -(\text{actual} - \text{predicted}) = -\text{residual}$
  - So Gradient Boosting fits models of this residual instead of actuals.



Strong Learner = (Base-learner o/p)+
(learning rate x predicted residual from tree1)+ ....

Get back to Bert’s problem:

Remember the Gradient Boosting steps:

1. Initialize a constant value (Base learner)
2. $F_0(x) = 278$ { it is just a leaf of our DT}
3. Base learner will give 278 as output to all the $(\boldsymbol{x\ data\ points})$

Population Density in Ks	Sub Block	Block	Water Demand (actuals)	Base learner output	Residuals from base learner
1.6	Rural	A	350	278	72
1.6	Urban	B	400	278	122
1.5	Rural	B	260	278	-18
1.4	Urban	A	100	278	-178

Build the first weak learner with X on residuals

$$\begin{pmatrix} \boldsymbol{x : \{Desity, Conn\}} \\ \boldsymbol{y : residuals} \end{pmatrix}$$

$predicted\ residuals$

Math alert!:

To select an initial constant value

$$\boldsymbol{F_0(x) = arg\min_{\gamma} \sum_{i}^{n} \mathcal{L}(y_i, \gamma)}$$

The above equation is nothing but Loss only :

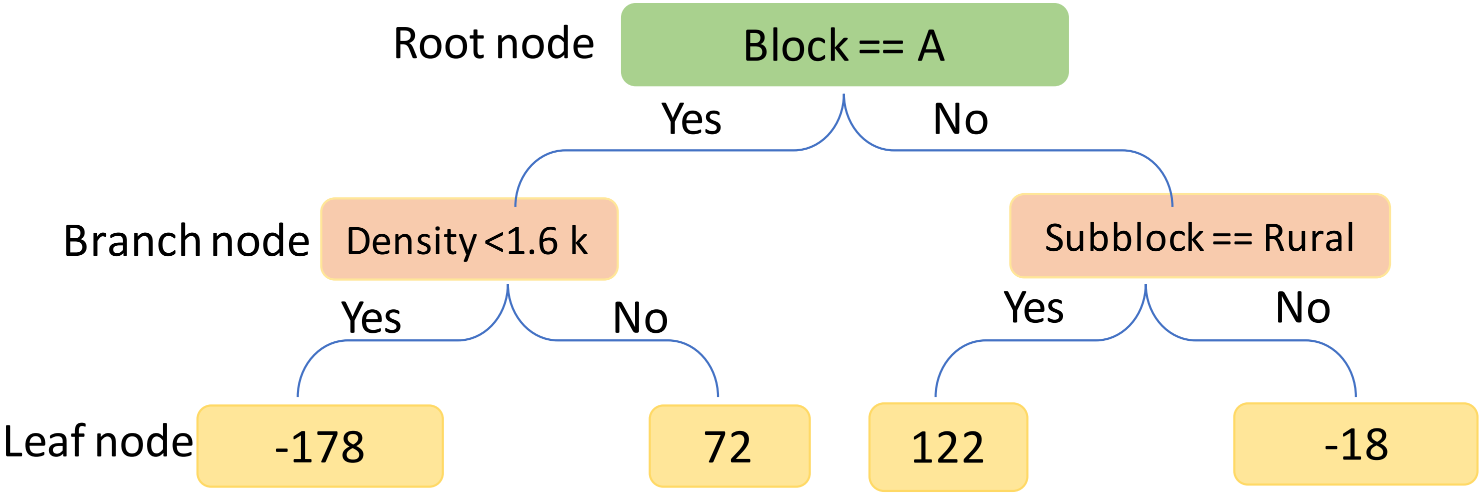
$\boldsymbol{½(350-\gamma)^2 + ½(400-\gamma)^2 + ½(260-\gamma)^2 + ½(100-\gamma)^2}$

By applying First order derivative to solve argmin problem: we will get

γ = 278 { average of actuals , approx.)

We have built our base learner , let's build a weak learner 1

$$\begin{pmatrix} x : \{Desity, Conn\} \\ y : residuals \end{pmatrix} \longrightarrow$$  $$\longrightarrow \; predicted\ residuals$$



Leaves are called as Terminal Regions - $R_{jth\ leaf, mth\ tree}$

Output at each leaf γ=

$$\arg\min_{\gamma} \sum_{R\,(j,m)}^{n} \mathcal{L}(y_i, (F_{m-1}(x) + \gamma))$$

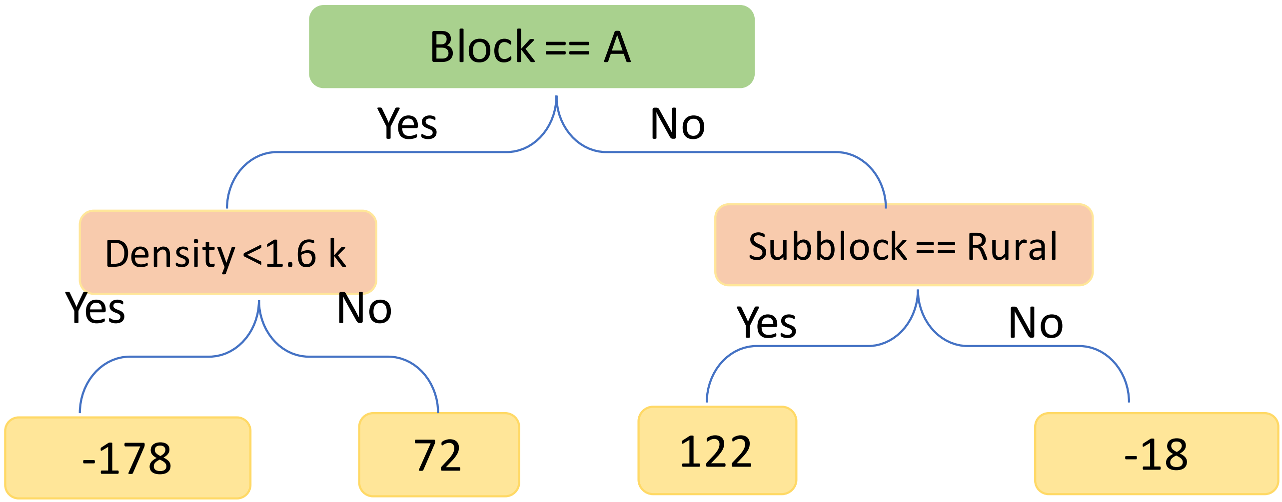
If we simplify that as usual: Output at each leaf γ = average of residuals at each leaf

Even if we have one residual at a leaf = residual/1

Note :

- Here Trees are built based on CART Algorithm (CART to build Decision Trees)
- Algorithm follows Gini Index / Entropy (Impurity Indices), Information gain to structure the Decision trees (Details omitted to avoid the confusion)
- Refer DT – CART algo for more info.

Output of weak learner 1



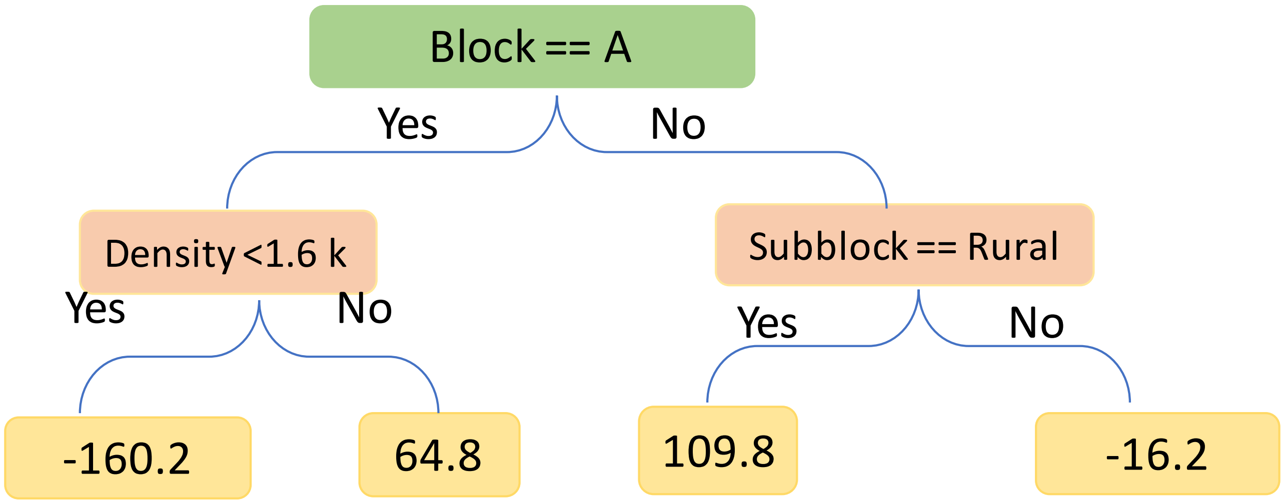
Leaves are called as Terminal Regions - $R_{jth\ leaf, mth\ tree}$

Inference:

1. **Predicted Residuals move towards zero**
2. **Predicted outputs move towards actuals**

Population Density in Ks	**Sub Block**	**Block**	**Water Demand (actuals)**	**Weak Learner output (predict. res)**	**O.P = 278+(0.1*pred.res)**	**Residuals from Weak learner 1 (to trainer wl 2)**
1.6	Rural	A	350	72	278+7.2= **285.2**	**64.8**
1.6	Urban	B	400	122	278.5+12.2= **290.2**	**109.8**
1.5	Rural	B	260	-18	278-1.8 = **276.2**	**-16.2**
1.4	Urban	A	100	-178	278-17.8= **260.2**	**-160.2**

Repeat until m= M., (that’s what ML does ☺)

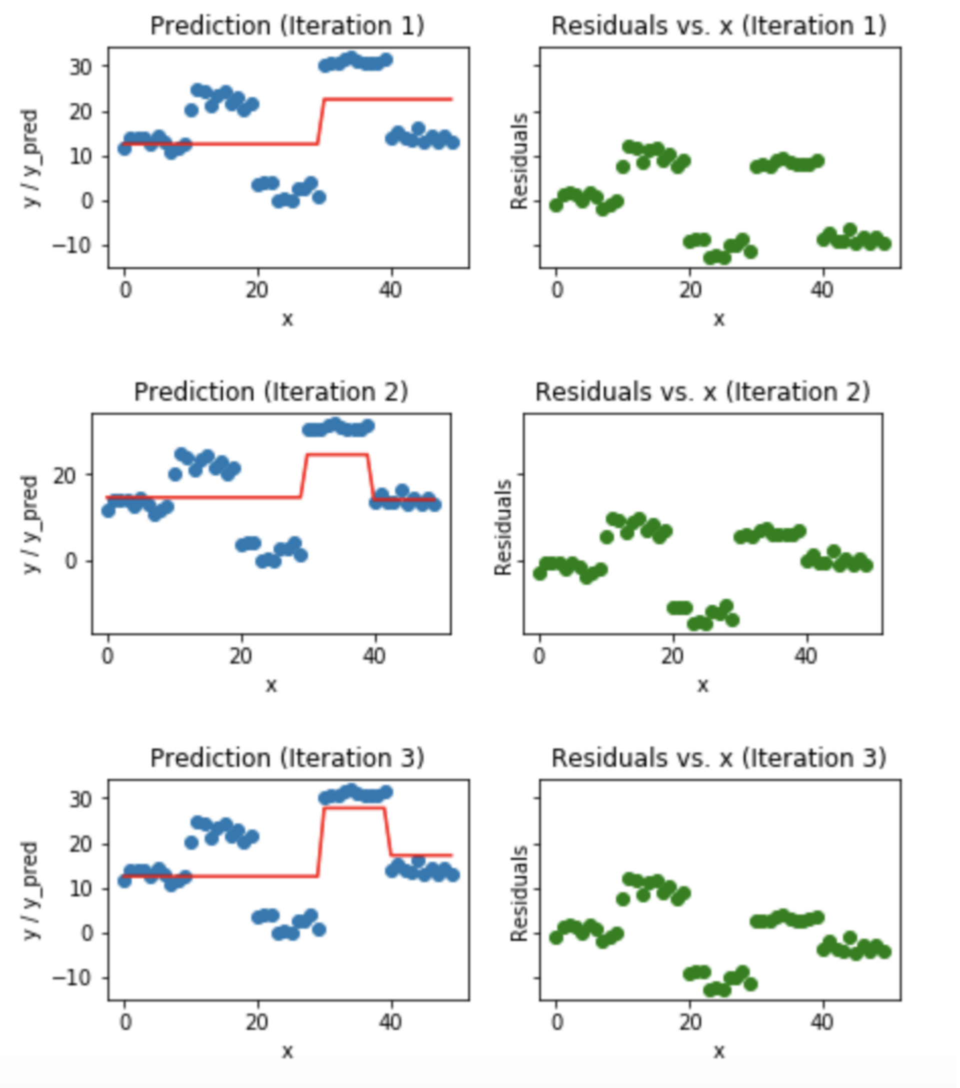


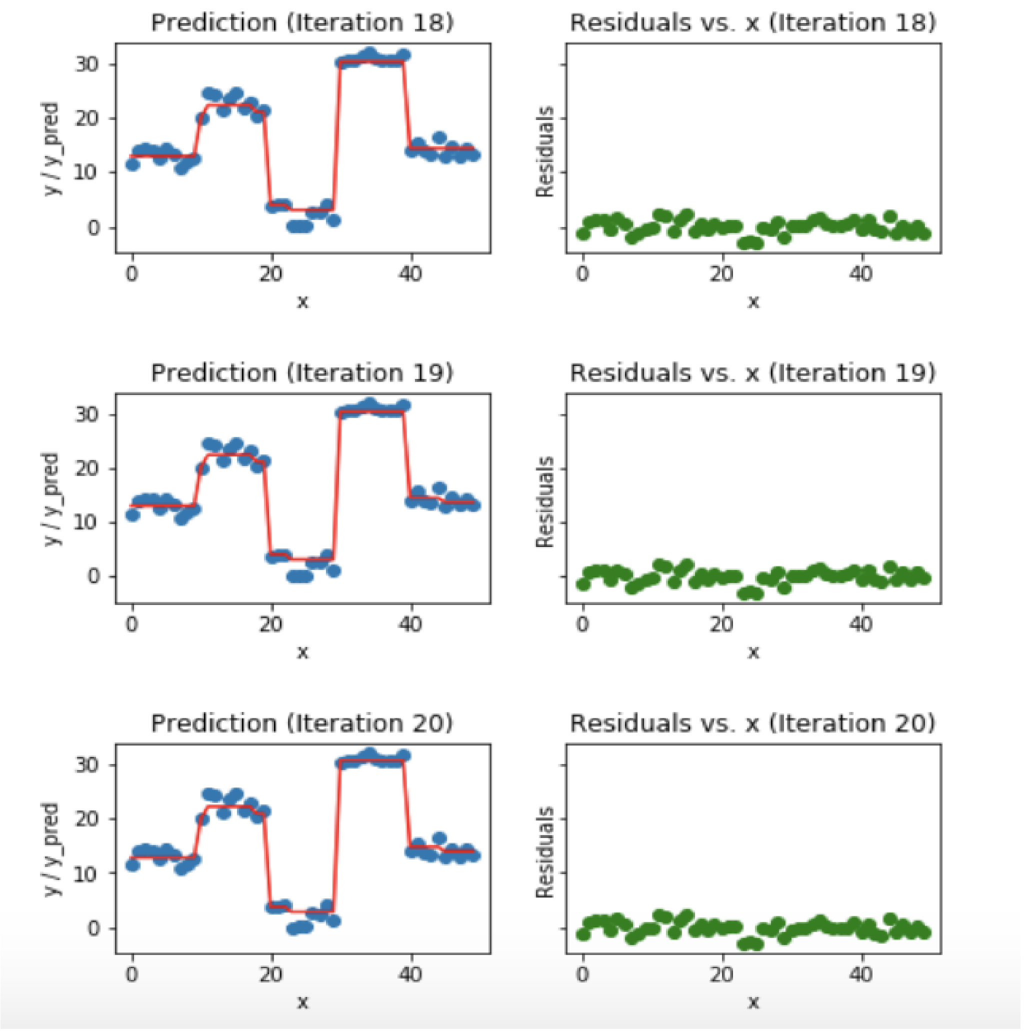
Leaves are called as Terminal Regions - $R_{jth\ leaf, mth\ tree}$

Inference:
1. Outputs and Residuals movements are in correct direction
Term alert:
M is max number of Trees/estimators default 100 in py packages

Population Density in Ks	Sub Block	Block	Water Demand (actuals)	Weak Learner2 output (predict. res)	O.P = wl1_O.P+(0.1*pred. res)	Residuals from Weak learner 2 (to trainer wl 3)
1.6	Rural	A	350	64.8	**285.2+6.48= 291.7**	**59.3**
1.6	Urban	B	400	109.8	**290.2+10.98= 301.12**	**98.88**
1.5	Rural	B	260	-16.2	**276.2-1.62 = 274.58**	**-14.5**
1.4	Urban	A	100	-160.2	**260.2-16.02= 244.18**	**-144.18**

Dummy Illustration:





eXtreme Gradient Boosting

a pro player in Kaggle Competitions

XGBoost: Execution and Features promote a statistical methods to ML algorithm

- GBM + Regularization
- Boosted trees
- Structure evaluation using similarity score

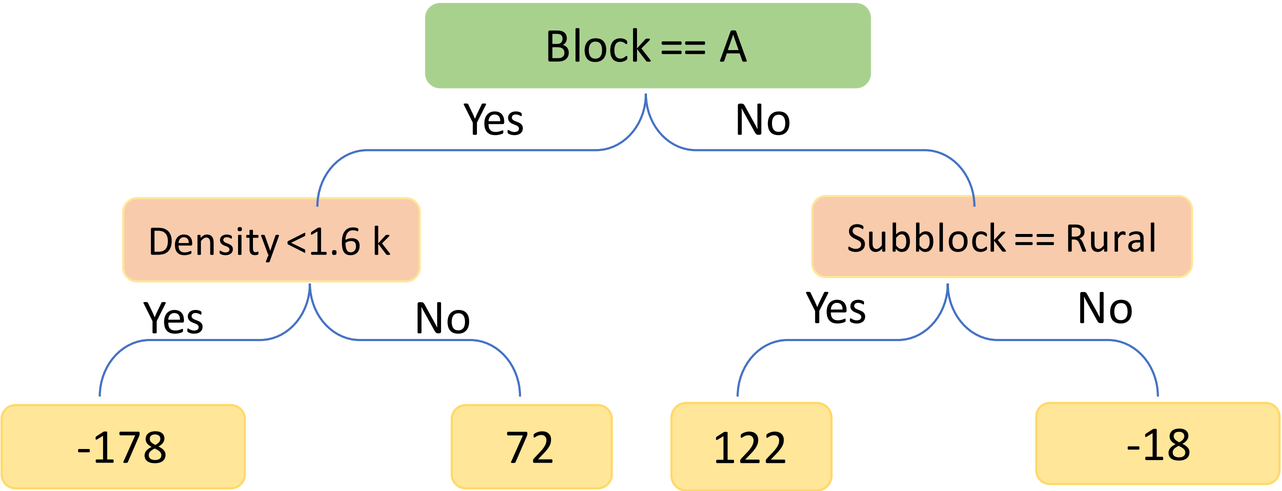
- Approximated Greedy algorithm
- Weighted Quantile Search
- Sparsity aware split finding – missing values handling

- Parallel Computation
- Taylor approx. to reduce the computation cost
- Bootstrapped training for Larger datasets

- Cache aware computation
- Blocks for out of core computation

Now Bert is ready to learn XGBoost:

- Let's add a regularization term to avoid
- Training Loss + Regularization



Leaves are called as Terminal Regions - $R_{jth\ leaf, mth\ tree}$

Usual Loss function:

$\sum_i^n \mathcal{L}(y_i, p_i)$; $where\ p_i\ - predicted$

Manuscript of XGBoost adds Regularization parameters:

$$\sum_{i}^{n} \mathcal{L}(y_i, p_i) + \gamma T + 1/2\lambda O_{value}^2;$$

where $\gamma\ - pruning\ penalty$, $\lambda - Regularization\ (Ridge)$

XGBoost uses second order Taylor approximation to avoid complex computation

Optimal Output value :

$$O_{value} = \frac{\sum sum\ of\ residuals}{\sum number\ of\ residuals + \lambda}$$

$say:\ \lambda = 1$

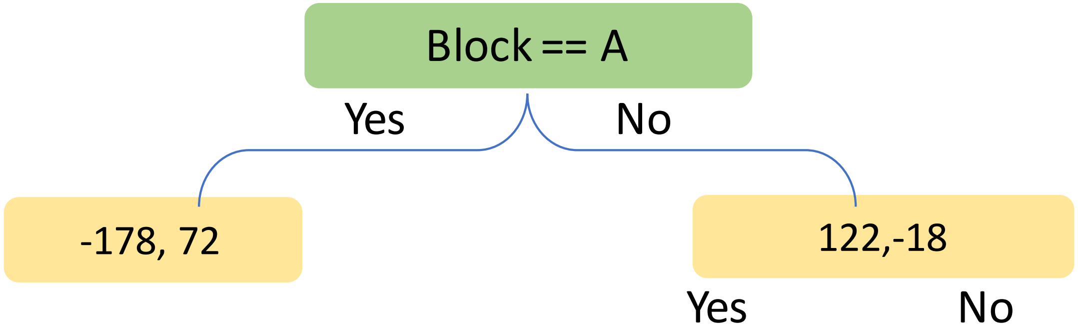
Penalty will give goodness in the future

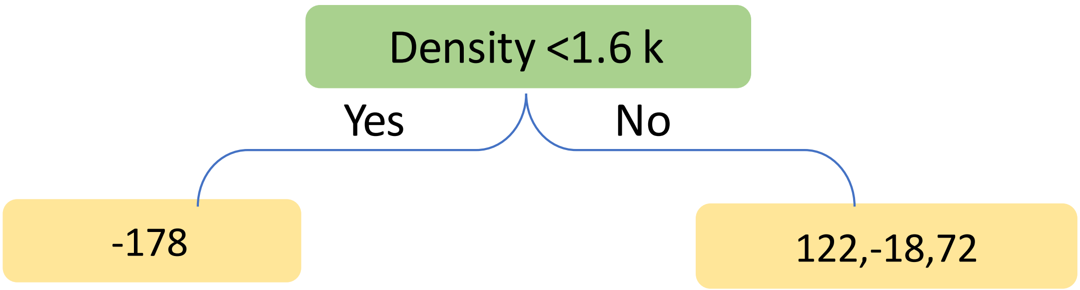
Population Density in Ks	Sub Block	Block	Water Demand (actuals)	Weak Learner output (predict. res)	O.P = 278+(0.1*pred.res)	Residuals from Weak learner 1 (to trainer wl 2)
1.6	Rural	A	350	72/2 = 36	278+3.6= **281.6**	**68.4**
1.6	Urban	B	400	122/2 =61	278+6.1= **264.1**	**235.9**
1.5	Rural	B	260	-18/2 =-9	278-0.9 = **278.9**	**--18.9**
1.4	Urban	A	100	-178/2 =-89	278-8.9=**269.1**	**-169.1**

Penalty term slows down the residual's movement , but will help the model better in future

- Note :
  - Rest of the Actions are same as unextreme Gradient boosting
  - Penalty factor is just a hyper parameter , it depends on the ML engineer to fix the value and take the decision to use

XGBoost uses Similarity Score to evaluate structures





Similarity score= $\frac{\sum \textit{sum of residuals ,squared}}{\sum \textit{number of residuals}}$

Tree 1:
Root similarity Score : 1
Left similarity score : 5618
Right similarity score : 5408
Gain = Left + Right- Root = 11025

Tree2 is having better split.

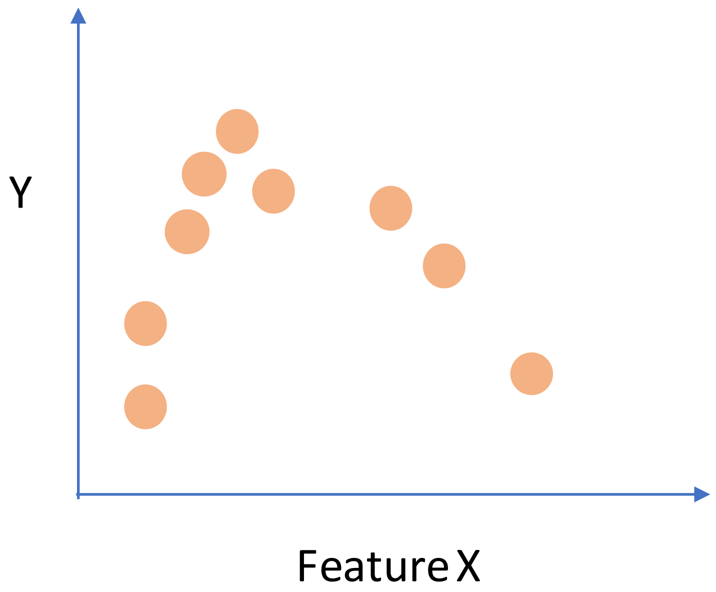
Tree 2:
Root similarity Score : 1
Left similarity score : 31684
Right similarity score : 10325
Gain = Left + Right- Root = 42009

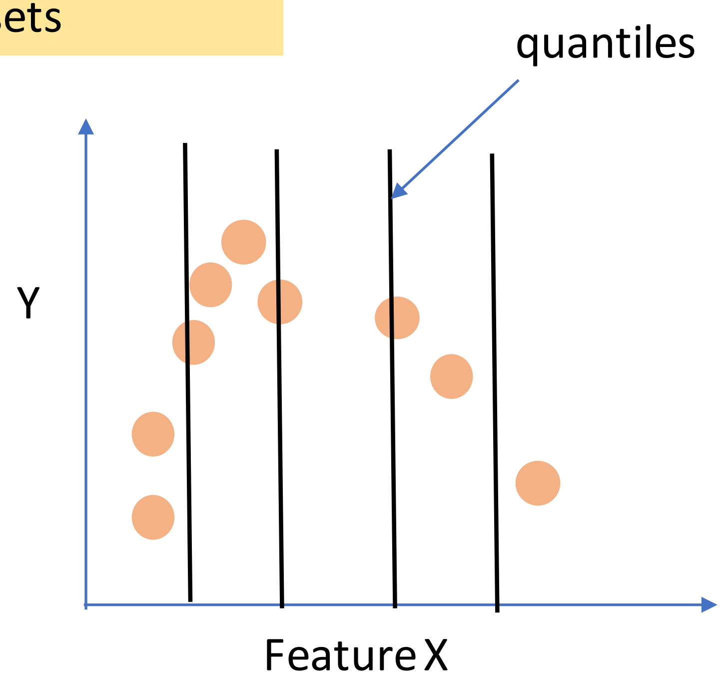
XG Boost uses Similarity score to evaluate the splits , which is faster than Gini/Entropy based one and it also linked with Approximate Greedy algorithm

Approximate Greedy algorithm reduces the Split Evaluation number.

GBM checks the split quality for each and every features and their data point , whereas in Approximated Greedy algorithm uses Quantiles.

Its implemented when we have large datasets



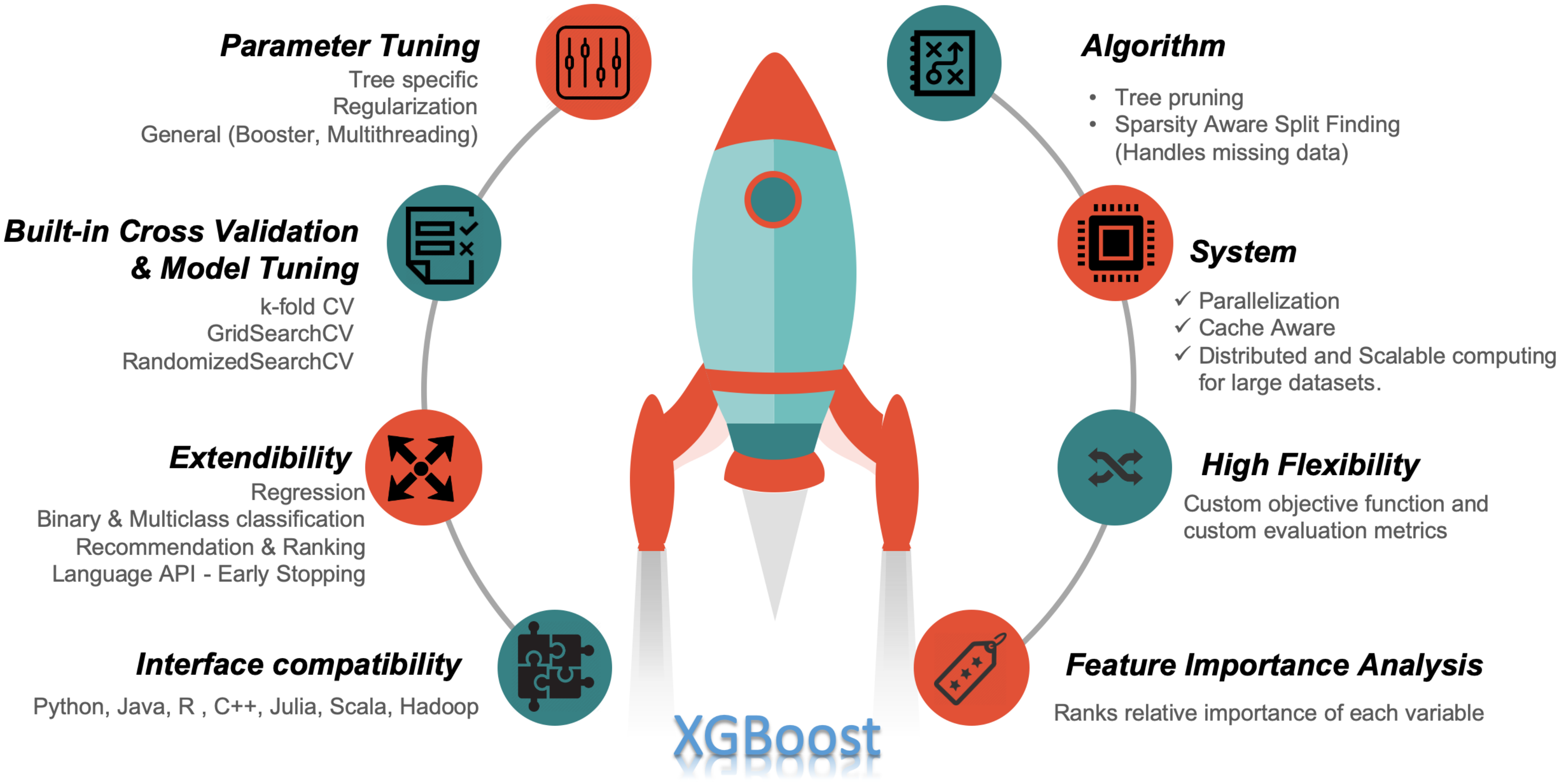


Approximate Greedy algorithm builds splits only on Quantiles and reduces the number of thresholds and splits evaluation

Speeds up the computation

* Weighed Quantiles and Sketch algorithms are also being used

Consolidated Features of XGBoost



Thank you

Questions!

Appendix : Maths

Instead of direct 2nd order derivation , we can use Taylor approx.:

$-\!\!\rightarrow \sum \boldsymbol{\mathcal{L}}(\boldsymbol{y_i}, \boldsymbol{p_i}' + \boldsymbol{O\ value}) \approx \sum \boldsymbol{\mathcal{L}}\,(\boldsymbol{y_i}, \boldsymbol{p_i})\,;$

$$\boldsymbol{\mathcal{L}}\,(\boldsymbol{y_i}, \boldsymbol{p_i}) + \frac{\mathbf{d}}{\mathbf{d}\boldsymbol{p_i}}\boldsymbol{\mathcal{L}}\,(\boldsymbol{y_i}, \boldsymbol{p_i})^{*}\boldsymbol{O_{value}} + \frac{\mathbf{d}^2}{\mathbf{d}\boldsymbol{p_i}}\boldsymbol{\mathcal{L}}\,(\boldsymbol{y_i}, \boldsymbol{p_i})^{*}\ \boldsymbol{O^2_{value}}$$

- First order derivative of the loss is gradient –g
- Second order loss is hessian – h
- This improves the computation speed – no need to perform differentiation
- Calculation of gradients and hessians taken place in Cache memory – faster execution

So Optimal Output value :

$$\boldsymbol{O_{value}} = \frac{\sum \boldsymbol{sum\ of\ residuals}}{\sum \boldsymbol{number\ of\ residuals}}$$

Boosting in Machine Learning:

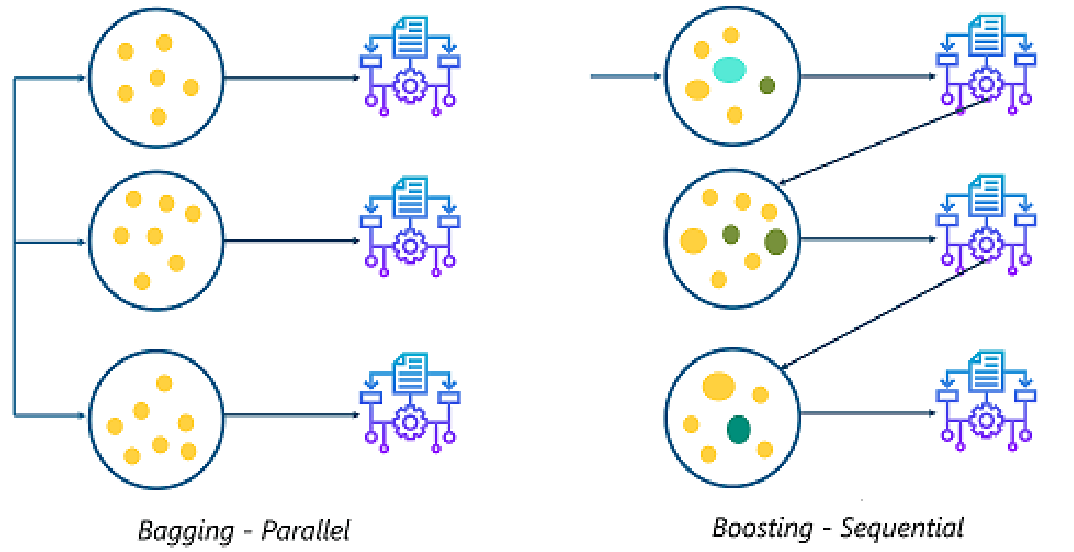
- Boosting is a type of ensemble machine learning technique. predictors are not made independently, but sequentially

Ensemble Learning :

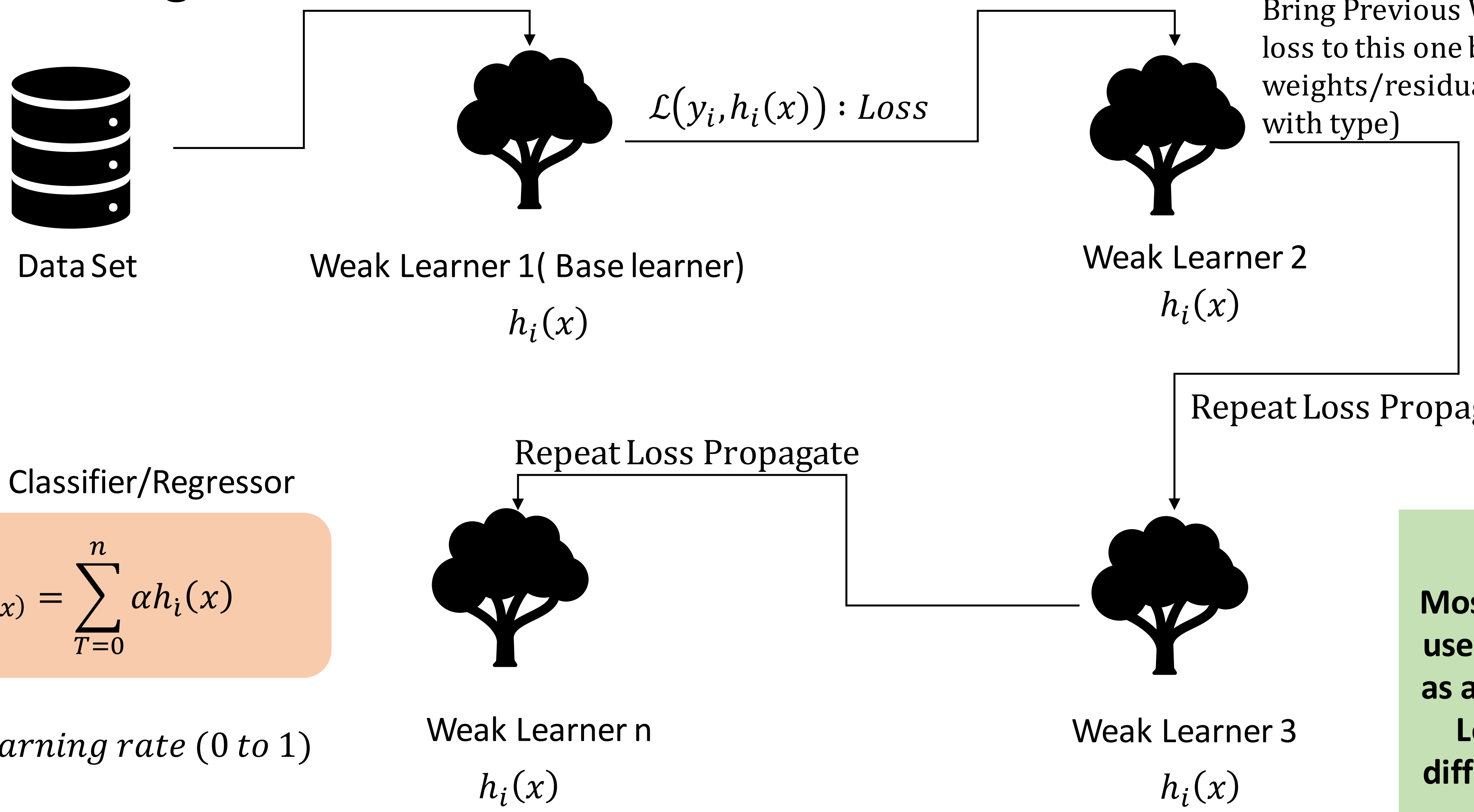
- Single model can lead to High bias and High variance
- Multiple models (often called "weak learners") are trained to solve the same problem and combined to get better results
- Strong Learner = $\Sigma\ weak\ learners$; $\Sigma\ denotes\ ensembling\ action$
- Bagging(Bootstrapped aggregation) , Boosting(Today's topic) , Stacking and Blending(uses meta-models)

Boosting:

- The idea of boosting is to train weak learners sequentially, each trying to correct its predecessor.
- Error is being corrected by weights or gradients (based on type of boosting)
- Ada Boost(mAdaBoost) , Gradient Boost { XGBoost , LGBoost, CatBoost},BrownBoost, LogitBoost



Boosting:



Final Classifier/Regressor

$$F_{(x)} = \sum_{T=0}^{n} \alpha h_i(x)$$

α $- learning\ rate$ $(0\ to\ 1)$

Strong Learner = (Base-learner) + (learning rate x Weaklearner1) + ...

Most of the Algos use Decision tree as a weak learner Loss fucn will differ for Reg and Clasfn

Gradient Boosting: (Gradient Descent + Boosting) with residuals

- Math and Example:
- Data : $\{(x_i, y_i): i \text{ to } n\}$ Weight has to be predicted (Regression problem)

Height	Color	Gender	Weight
1.6	Blue	Male	88
1.6	Green	Female	76
1.5	Blue	Female	56
1.4	Green	Male	66

Our objective is to minimize the Loss function:

$$\sum_i^n \mathcal{L}(y_i, y_i^p) \rightarrow y_i - actual\,, y_i^p = predicted$$

$$\text{Loss} = ½\ (\text{actual} - \text{predicted})^2 :: ½\ \text{MSE}$$

½ is added to reduce the complexity while differentiating

When diff the Loss (Gradient descent Logic) :

$$\frac{\partial L}{\partial y_p} = -(\text{actual} - \text{predicted}) = -\text{residual}$$

Let’s start Boosting

- A. Initialize a Model with constant value by following condition: $\boldsymbol{F_0(x)} = \boldsymbol{arg\min_{\gamma} \sum_i^n \mathcal{L}(y_i, \gamma)}$

The above equation is nothing but Loss only : **$\frac{1}{2}(88-\gamma)^2 + \frac{1}{2}(76-\gamma)^2 + \frac{1}{2}(56-\gamma)^2 + \frac{1}{2}(66-\gamma)^2$**

By applying First order derivative to solve argmin problem: we will get γ = 71 { average of actuals)

We have our base model $F_0(x) = 71$
{ it is just a leaf of our DT}

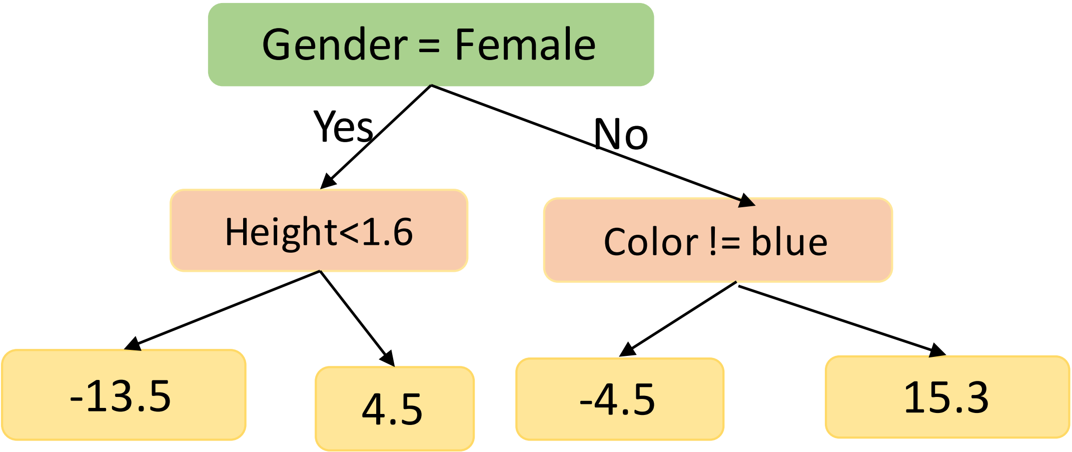
For M trees m = 1 to M:

- Calculate the Residual to build our first weak learner: actual - 71

Height	Color	Gender	Weight	Weight – 71	$residual_{i,m}$
1.6	Blue	Male	88	88-71	17
1.6	Green	Female	76	76-71	5
1.5	Blue	Female	56	56-71	-15
1.4	Green	Male	66	66-71	-5

Gradient boosting Completed:

Repeat: until m= Max no of tree



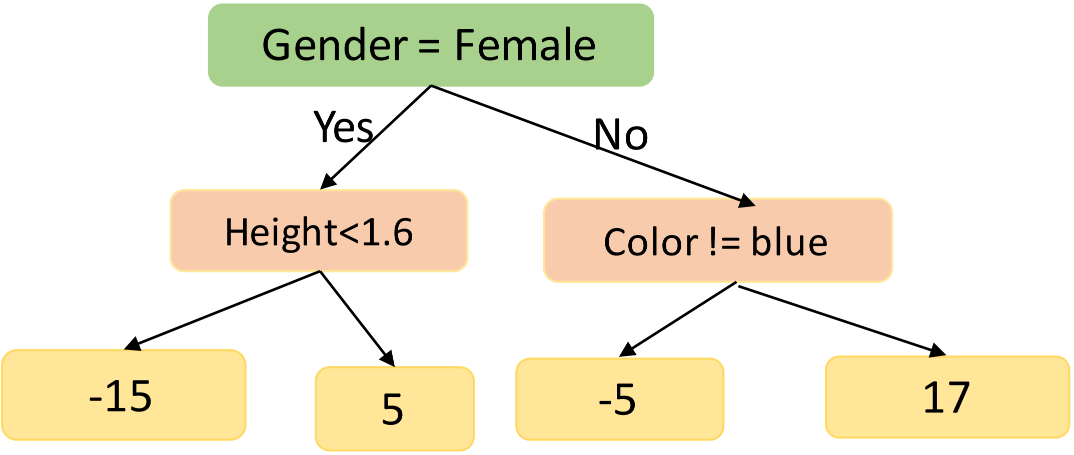
Leaves are called as Terminal Regions - $R_{jth\ leaf, mth\ tree}$

Height	Color	Gender	Weight	Leaf output	71+ 0.1(leaf output)	New Residuals
1.6	Blue	Male	88	15.3	72.7+1.53 =74.23	13.77
1.6	Green	Female	76	4.5	71.5+0.45 =71.95	4.05
1.5	Blue	Female	56	-13.5	69.5-1.35 =68.15	-12.15
1.4	Green	Male	66	-4.5	70.5-0.45 =70.05	--4.05

Note : Residuals are continuing their movement toward zero. Default GBM will have 100 trees , but we can control it by number of estimators M value.

cont.:

- Calculate the output of the tree 1(predicted residual) consider learning rate = 0.1 , get new



Leaves are called as Terminal Regions - $R_{jth\ leaf,mth\ tree}$

Height	Color	Gender	Weight	Leaf output	71+ 0.1(leaf output)	New Residuals
1.6	Blue	Male	88	17	71+1.7 = 72.7	15.3
1.6	Green	Female	76	5	71+0.5=71.5	4.5
1.5	Blue	Female	56	-15	71-1.5=69.5	-13.5
1.4	Green	Male	66	-5	71-0.5 = 70.5	-4.5

Note : Residuals have started their movement toward zero.

Strong Learner = (Base-learner) + (learning rate x Weaklearner1) + ...