upGrad
#LifeKoKaroLift

Post-Graduate Diploma in ML/AI

upGrad

Course : Machine Learning

Lecture On : Boosting - Part 2

Instructor : Manish Kumar

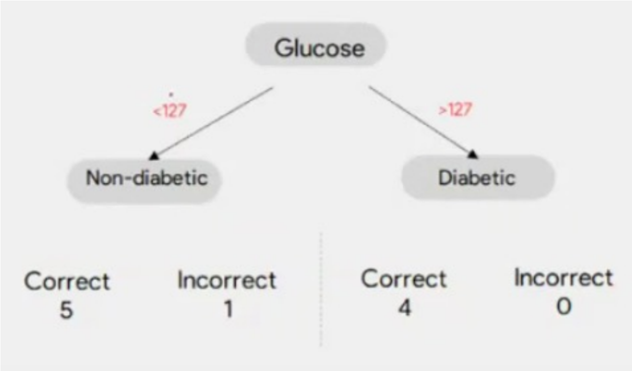
upGrad

Modules Included

- Quick Recap - AdaBoost
- Gradient Boosting
- Gradient Descent
- XGBoost – (pros /cons)

Ada Boost

upGrad



Glucose	Insulin	BMI	Age	Diabetes	Prediction	Sample weight
89	94	28.1	21	-1	-1	0.1
137	168	43.1	33	1	1	0.1
78	88	31	26	-1	-1	0.1
197	543	30.5	53	1	1	0.1
189	846	30.1	59	1	1	0.1
166	175	25.8	51	1	1	0.1
118	230	45.8	31	1	-1	0.1
103	83	43.3	33	-1	-1	0.1
115	96	34.6	32	-1	-1	0.1
126	235	39.3	27	-1	-1	0.1

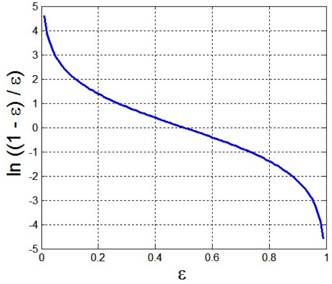
Error rate:
(i = index of classifier, j=index of instance)

$$\varepsilon_i = \frac{1}{N}\sum_{j=1}^{N} w_j \delta\left(C_i(x_j) \neq y_j\right)$$

The lower a classifier error rate, the more accurate it is, and therefore, the higher its weight for voting should be

Weight of a classifier C_i's vote is

$$\alpha_i = \frac{1}{2}\ln\left(\frac{1-\varepsilon_i}{\varepsilon_i}\right)$$



Ada Boost

upGrad

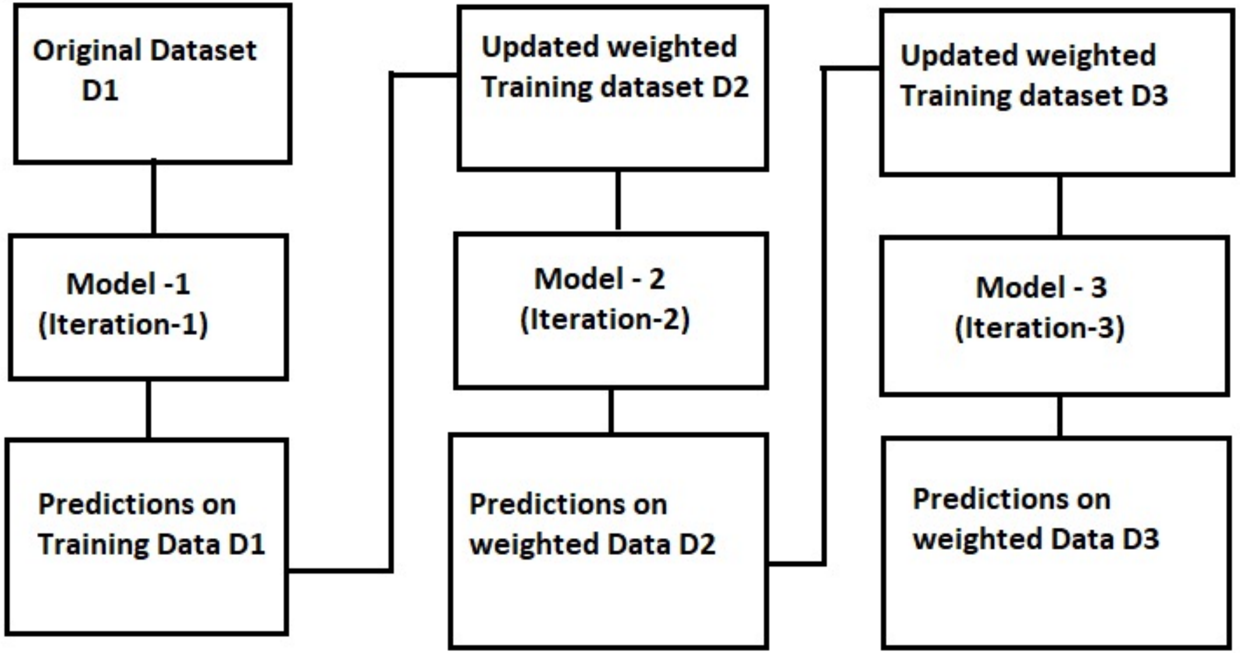
Glucose	Insulin	BMI	Age	Diabetes	Prediction	Sample weight	New weights	Norm weights
89	94	28.1	21	-1	-1	0.1	0.03	0.06
137	168	43.1	33	1	1	0.1	0.03	0.06
78	88	31	26	-1	-1	0.1	0.03	0.06
197	543	30.5	53	1	1	0.1	0.03	0.06
189	846	30.1	59	1	1	0.1	0.03	0.06
166	175	25.8	51	1	1	0.1	0.03	0.06
118	230	45.8	31	1	-1	0.1	0.30	0.50
103	83	43.3	33	-1	-1	0.1	0.03	0.06
115	96	34.6	32	-1	-1	0.1	0.03	0.06
126	235	39.3	27	-1	-1	0.1	0.03	0.06

$$w_j^{(i+1)} = \frac{w_j^{(i)}}{Z_i} \begin{cases} \exp^{-\alpha_i} & \text{if } C_i(x_j) = y_j \\ \exp^{\alpha_i} & \text{if } C_i(x_j) \neq y_j \end{cases}$$

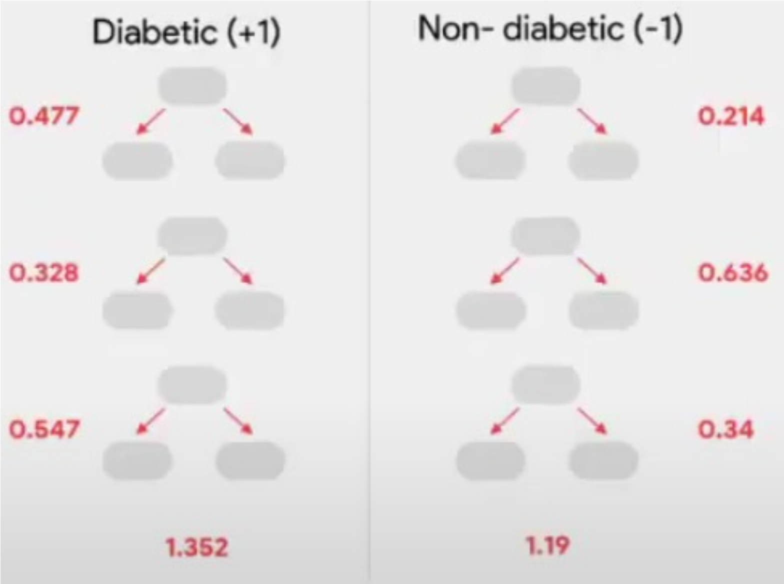
where Z_i is the normalization factor

Ada Boost

upGrad



upGrad



Testing:

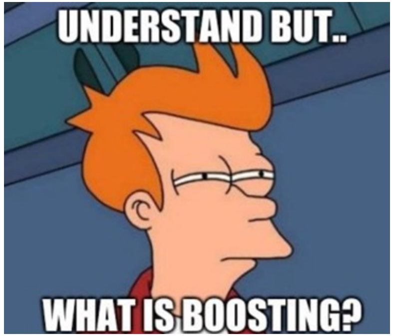
For each class c, sum the weights of each classifier that assigned class c to X (unseen data)

The class with the highest sum is the **WINNER!**

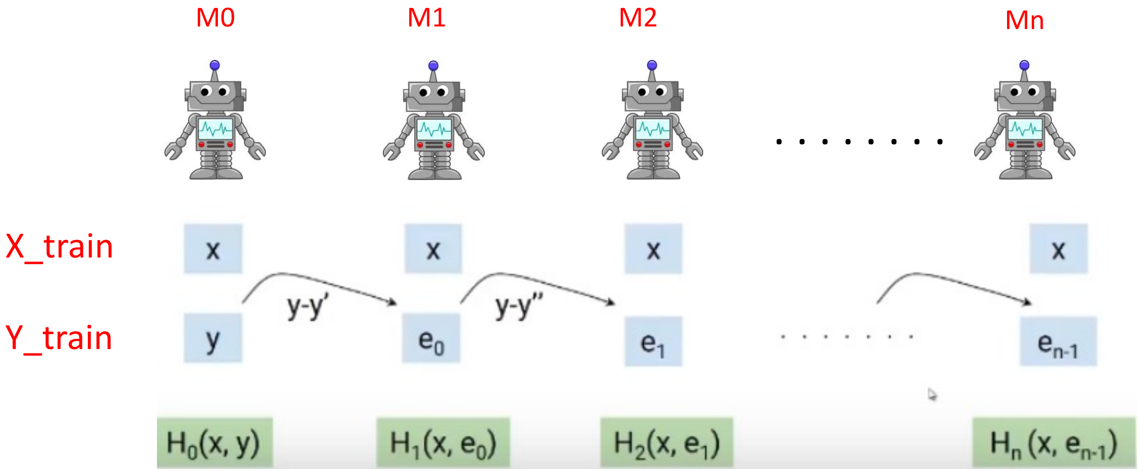
$$C^*(x_{test}) = \arg\max_{y} \sum_{i=1}^{T} \alpha_i \delta\left(C_i(x_{test}) = y\right)$$

Gradient Boosting Algorithm

Like AdaBoost in GBM as well we reduce the bias of the weak learners by building models sequentially where each of the subsequent models tries to reduce the error of the previous model. But How?

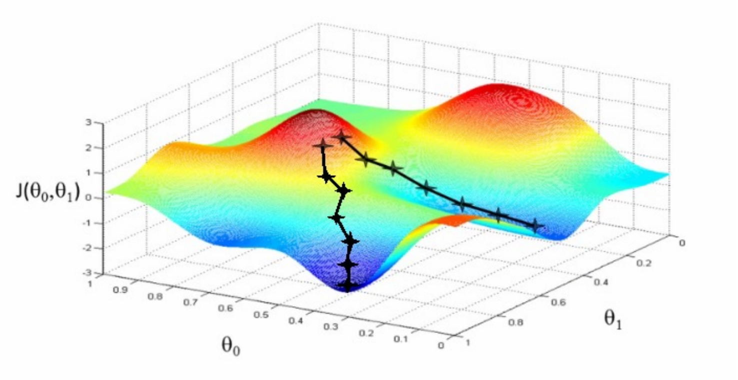


Gradient boosting approaches the problem a bit differently. Instead of adjusting weights of data points, Gradient boosting focuses on the difference between the prediction and the ground truth.



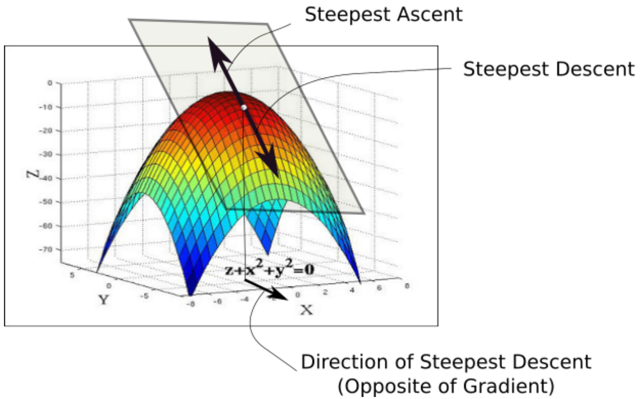
Gradient Descent

upGrad



While the direction of the gradient tells us which direction has the steepest ascent, it's magnitude tells us how steep the steepest ascent/descent is. So, at the minima, where the contour is almost flat, you would expect the gradient to be almost zero. In fact, it's precisely zero for the point of minima.

Gradient descent is an optimization algorithm used to minimize some function by iteratively moving in the direction of steepest descent as defined by the negative of the gradient



Gradient Boosting Algorithm

upGrad

Input: training set $\{(x_i, y_i)\}_{i=1}^{n}$, a differentiable loss function $L(y, F(x))$, number of iterations M.

Algorithm:

1. Initialize model with a constant value:

$$F_0(x) = \underset{\gamma}{\arg\min} \sum_{i=1}^{n} L(y_i, \gamma).$$

2. For m = 1 to M:
   1. Compute so-called *pseudo-residuals*:

   $$r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)} \quad \text{for } i = 1, \ldots, n.$$

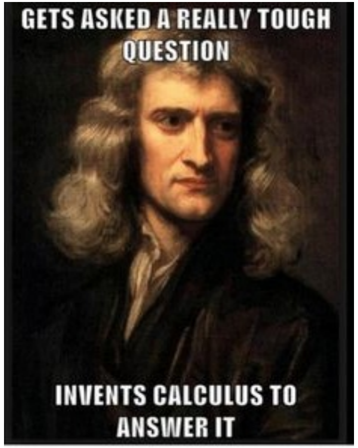
   2. Fit a base learner (or weak learner, e.g. tree) $h_m(x)$ to pseudo-residuals, i.e. train it using the training set $\{(x_i, r_{im})\}_{i=1}^{n}$.
   3. Compute multiplier γ_m by solving the following one-dimensional optimization problem:

   $$\gamma_m = \underset{\gamma}{\arg\min} \sum_{i=1}^{n} L\left(y_i, F_{m-1}(x_i) + \gamma h_m(x_i)\right).$$

   4. Update the model:

   $$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x).$$

3. Output $F_M(x)$.



You don't just ask Issac Newton a tough question..

https://statweb.stanford.edu/~jhf/ftp/trebst.pdf

The algorithm can be then described as the following, on a dataset (x, y) with x the features and y the targets, with a differentiable loss function $\mathcal{L}$:

$\mathcal{L} = \frac{1}{2}(\mathcal{O}bs - \mathcal{P}red)^2$, called the Squared Residuals. Notice that since the function is differentiable, we have :

$$\frac{\delta}{\delta Pred}\mathcal{L} = -1 \times (\mathcal{O}bs - \mathcal{P}red)$$

When the lecturer skips 10 steps in the derivation

The proof is trivial

Step 1 : Initialize the model with a constant value : $F_0(x) = argmin_\gamma \sum_i \mathcal{L}(y_i, \gamma)$. We simply want to minimize the sum of the squared residuals (SSR) by choosing the best prediction γ.

If we derive the optimal value for γ :

$$\frac{\delta}{\delta\gamma}\sum_i \mathcal{L}(y_i, \gamma) = -(y_1 - \gamma) + -(y_2 - \gamma) + -(y_3 - \gamma) + \ldots = 0$$

$$\sum_i y_i - n * \gamma = 0$$

$$\gamma = \frac{\sum_i y_i}{n} = \bar{y}$$

Gradient Boosting Algorithm

upGrad

Area(Sq-ft)	Room(BHK)	Furnished	Rent(K)	Average(K)
480	2	Y	15	27.7
255	1	Y	10	27.7
800	1	N	35	27.7
785	2	N	39	27.7
900	3	Y	55	27.7
350	1	Y	12	27.7

Step 1 : Calculate the average of the target label

For regression, we start with a leaf that is the average value of the target variable. This leaf will be used as a baseline to approach the correct solution in the next steps.

Gradient Boosting Algorithm

upGrad

Step 2 : For m = 1 to M (the maximum number of trees specified, e.g 100)

- a) Compute the pseudo-residuals for every sample :

$$r_{im} = -\frac{\delta \mathcal{L}(y_i, \mathcal{F}(x_i))}{\delta F(x_i)} = -(-1 \times (Obs - F_{m-1}(x))) = (Obs - F_{m-1}(x)) = (Obs - Pred)$$

This derivative is called the Gradient. The Gradient Boost is named after this.

Gradient Boosting Algorithm

upGrad

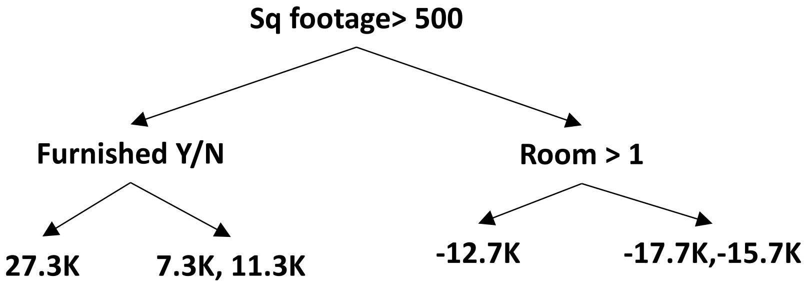
Area(Sq-ft)	Room(BHK)	Furnished	Rent(K)	Average(K)	Residual
480	2	Y	15	27.7	-12.7
255	1	Y	10	27.7	-17.7
800	1	N	35	27.7	7.3
785	2	N	39	27.7	11.3
900	3	Y	55	27.7	27.3
350	1	Y	12	27.7	-15.7

Step 2 : Calculate the residual for every sample in the dataset

Gradient Boosting Algorithm

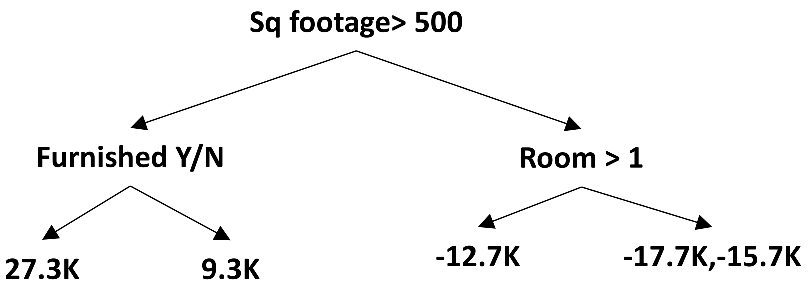
upGrad

Area(Sq-ft)	Room(BHK)	Furnished	Rent(K)	Average(K)	Residual
480	2	Y	15	27.7	-12.7
255	1	Y	10	27.7	-17.7
800	1	N	35	27.7	7.3
785	2	N	39	27.7	11.3
900	3	Y	55	27.7	27.3
350	1	Y	12	27.7	-15.7



Gradient Boosting Algorithm

Area(Sq-ft)	Room(BHK)	Furnished	Rent(K)	Average(K)	Residual
480	2	Y	15	27.7	-12.7
255	1	Y	10	27.7	-17.7
800	1	N	35	27.7	7.3
785	2	N	39	27.7	11.3
900	3	Y	55	27.7	27.3
350	1	Y	12	27.7	-15.7



Gradient Boosting Algorithm

upGrad

Area(Sq-ft)	Room(BHK)	Furnished	Rent(K)	Average(K)	Residual	Residual
480	2	Y	15	27.7	-12.7	-12.7
255	1	Y	10	27.7	-17.7	-16.7
800	1	N	35	27.7	7.3	9.3
785	2	N	39	27.7	11.3	9.3
900	3	Y	55	27.7	27.3	27.3
350	1	Y	12	27.7	-15.7	-16.7

Step 3 : Construct the model on top of residuals

We will now build a model(decision tree) which is aimed at predicting residuals. In other words, every leaf will contain a prediction as to the value of the residual

Gradient Boosting Algorithm

upGrad

Area(Sq-ft)	Room(BHK)	Furnished	Rent(K)	Average(K)	Residual	Pedicted
480	2	Y	15	27.7	-12.7	26.4
255	1	Y	10	27.7	-17.7	26
800	1	N	35	27.7	7.3	28.6
785	2	N	39	27.7	11.3	28.6
900	3	Y	55	27.7	27.3	30.4
350	1	Y	12	27.7	-15.7	26

- Make a new prediction for each sample by updating, accoridng to a learning rate $lr \in (0,1)$: $F_m(x) = F_{m-1}(x) + lr \times \sum_j \gamma_{jm} I(x \in R_{jm})$. We compute the new value by summing the previous prediction and all the predictions γ into which our sample falls.

By increasing the learning rate, we tend to overfit. However, if the learning rate is too low, it takes a large number of iterations to even approach the underlying structure of the data.

Step 4 : Compute the predictions

27.7K(Average rent) + 0.1(LR)(Residual of the sample) = New Prediction

27.7 +0.1*(-12.7) = 26.4

Gradient Boosting Algorithm

upGrad

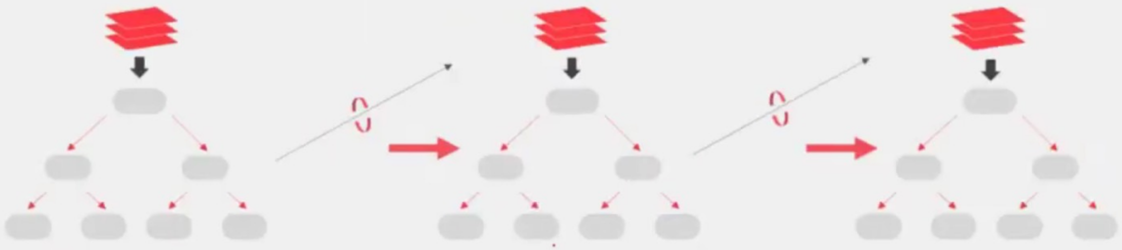
Area(Sq-ft)	Room(BHK)	Furnished	Rent(K)	Average(K)	Residual	Pedicted	New-Residual
480	2	Y	15	27.7	-12.7	26.4	-11.4
255	1	Y	10	27.7	-17.7	26	-16
800	1	N	35	27.7	7.3	28.6	6.4
785	2	N	39	27.7	11.3	28.6	10.4
900	3	Y	55	27.7	27.3	30.4	24.6
350	1	Y	12	27.7	-15.7	26	-14

Step 5 : Compute the new residuals

New residuals of the sample = Actual rent – predicted rent

Gradient Boosting Algorithm

upGrad



Prediction : Use all of the trees in the ensemble to make a final prediction

27.7K + 0.1*(-12.7) + 0.1*(-11.4) + …

upGrad

Step 1 : Make the first guess

The initial guess of the Gradient Boosting algorithm is to *predict the log of the odds of the target y*, the equivalent of the average for the logistic regression.

$$odds = log(\frac{P(Y=1)}{P(Y=0)}) = log(\frac{3}{1}) = log(3)$$

How is this ratio used to make a classification? We apply a softmax transformation!

$$P(Y=1) = \frac{e^{odds}}{1+e^{odds}} = \frac{3}{4} = 0.75$$

If this probability is greater than 0.5, we classify as 1. Else, we classify as 0.

Gradient Boosting Algorithm

upGrad

Age	Sex	Pclass	Survived	Initial Pred
22	M	3	0	0.57
38	F	1	1	0.57
26	F	1	1	0.57
35	F	1	1	0.57
54	M	3	0	0.57
2	M	3	1	0.57
43	M	3	0	0.57

Step 1 : Calculate the initial Prediction for every individual -> log(odds)

We can't use the value directly, since it needs to be converted to a probability value. To do so, we will use the logistic function to transform it to a probability.

$$\log(4/3) = 0 \cdot 29$$

Probability of surviving

$$\frac{e^{4/3}}{1 + e^{4/3}} = 0 \cdot 57$$

Gradient Boosting Algorithm

upGrad

Age	Sex	Pclass	Survived	Initial Pred	Residual
22	M	3	0	0.57	-0.57
38	F	1	1	0.57	0.43
26	F	1	1	0.57	0.43
35	F	1	1	0.57	0.43
54	M	3	0	0.57	-0.57
2	M	3	1	0.57	0.43
43	M	3	0	0.57	-0.57

Step 2 : Calculate the pseudo-residual for every sample dataset

Gradient Boosting Algorithm

upGrad

Age	Sex	Pclass	Survived	Initial Pred	Residual
22	M	3	0	0.57	-0.57
38	F	1	1	0.57	0.43
26	F	1	1	0.57	0.43
35	F	1	1	0.57	0.43
54	M	3	0	0.57	-0.57
2	M	3	1	0.57	0.43
43	M	3	0	0.57	-0.57

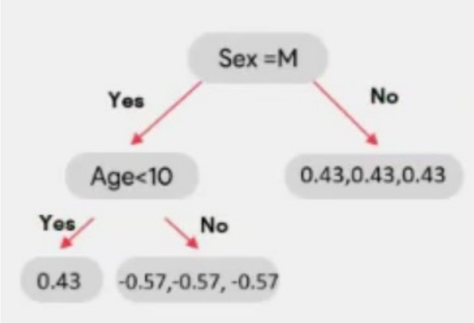
Step 3 : Construct the model on top of residuals

We will now build a model(decision tree) which is aimed at predicting residuals. In other words, every leaf will contain a prediction as to the value of the residual

Gradient Boosting Algorithm

upGrad

Age	Sex	Pclass	Survived	Initial Pred	Residual
22	M	3	0	0.57	-0.57
38	F	1	1	0.57	0.43
26	F	1	1	0.57	0.43
35	F	1	1	0.57	0.43
54	M	3	0	0.57	-0.57
2	M	3	1	0.57	0.43
43	M	3	0	0.57	-0.57



Step 3 : Construct the model on top of residuals

$$\gamma_{i+1} = \frac{\sum_i Residuals_i}{\sum(\gamma_i \times (1 - \gamma_i))}$$

Gradient Boosting Algorithm

upGrad

Age	Sex	Pclass	Survived	Initial Pred	Residual
22	M	3	0	0.57	-0.57
38	F	1	1	0.57	0.43
26	F	1	1	0.57	0.43
35	F	1	1	0.57	0.43
54	M	3	0	0.57	-0.57
2	M	3	1	0.57	0.43
43	M	3	0	0.57	-0.57

Step 3 : Construct the model on top of residuals

$$\gamma_{i+1} = \frac{\sum_i Residuals_i}{\sum(\gamma_i \times (1-\gamma_i))}$$

$$\frac{-0\cdot 57 - 0\cdot 57 - 0\cdot 57}{0\cdot 57(1-0\cdot 57)+0.57(1-0\cdot 57)+0\cdot 57(1-0\cdot 57)} = -2.3$$

Gradient Boosting Algorithm

upGrad

Age	Sex	Pclass	Survived	Initial Pred	Residual	Leaf Output	New log Odds
22	M	3	0	0.57	-0.57	-2.3	-1.09
38	F	1	1	0.57	0.43	1.75	1.34
26	F	1	1	0.57	0.43	1.75	1.34
35	F	1	1	0.57	0.43	1.75	1.34
54	M	3	0	0.57	-0.57	-2.3	-1.09
2	M	3	1	0.57	0.43	1.75	1.34
43	M	3	0	0.57	-0.57	-2.3	-1.09

Step 4 : Compute the predictions

New log(odds) = initial log(odds) + LR(Output value by D.T) -> log(4/3) = 0.29

New log(odds) = 2.9 +0.6(-2.3) = -1.09

New log(odds) = 0.29+0.6(1.75) = 1.34

Gradient Boosting Algorithm

upGrad

Age	Sex	Pclass	Survived	Initial Pred	Residual	Leaf Output	New log Odds	New Pred	New Residual
22	M	3	0	0.57	-0.57	-2.3	-1.09	0.25	-0.25
38	F	1	1	0.57	0.43	1.75	1.34	0.8	0.2
26	F	1	1	0.57	0.43	1.75	1.34	0.8	0.2
35	F	1	1	0.57	0.43	1.75	1.34	0.8	0.2
54	M	3	0	0.57	-0.57	-2.3	-1.09	0.25	-0.25
2	M	3	1	0.57	0.43	1.75	1.34	0.8	0.2
43	M	3	0	0.57	-0.57	-2.3	-1.09	0.25	-0.25

Step 5 : Build another tree on top of this residual

Gradient Boosting Algorithm

upGrad

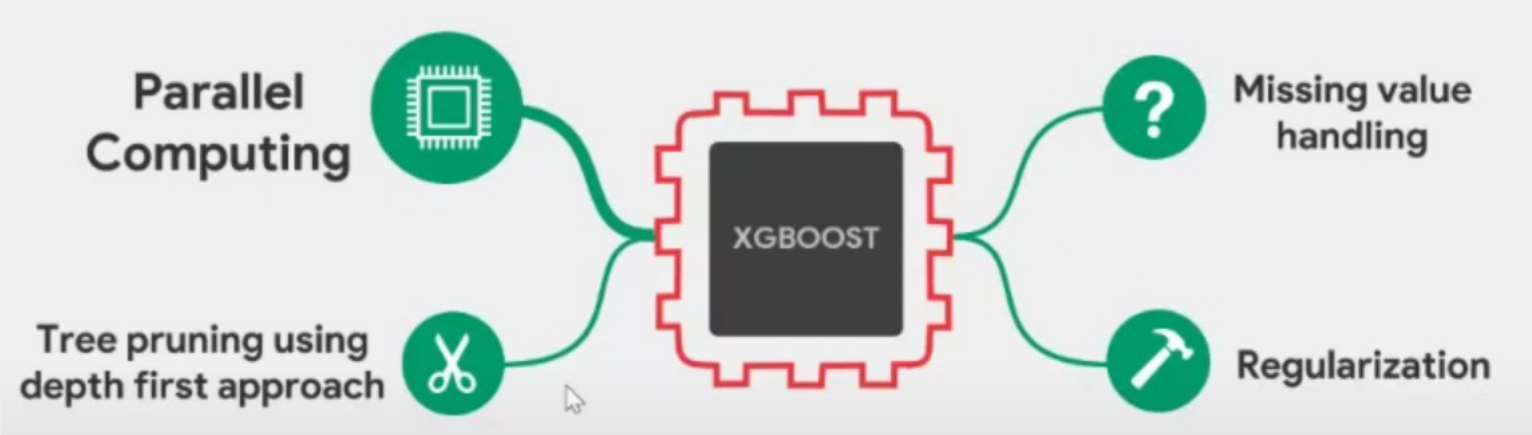
Age	Sex	Pclass	Survived	Initial Pred	Residual	Leaf Output	New log Odds	New Pred	New Residual
22	M	3	0	0.57	-0.57	-2.3	-1.09	0.25	-0.25
38	F	1	1	0.57	0.43	1.75	1.34	0.8	0.2
26	F	1	1	0.57	0.43	1.75	1.34	0.8	0.2
35	F	1	1	0.57	0.43	1.75	1.34	0.8	0.2
54	M	3	0	0.57	-0.57	-2.3	-1.09	0.25	-0.25
2	M	3	1	0.57	0.43	1.75	1.34	0.8	0.2
43	M	3	0	0.57	-0.57	-2.3	-1.09	0.25	-0.25

$$y_{pred} = odds + lr \times y_{res} + lr \times y_{res_2} + lr \times y_{res_3} + lr \times y_{res_4} + \ldots$$

And classifiy using :

$$P(Y=1) = \frac{e^{y_{pred}}}{1 + e^{y_{pred}}}$$

XGBOOST


Parallel Computing
Tree pruning using depth first approach
XGBOOST
Missing value handling
Regularization

#LifeKoKaroLift

Thank You!