Adaboost

→ In a forest of trees made with adaboost, the trees are usually just a node and two leaves.

→ A tree with one node and 2 leaves is called stumps.

→ In random forests ⇒ individual decision tree votes all are taken into account equally. Whereas, in adaboost some stumps have more weightage compared to others. (Amount of say)

→ likewise, all DT's are independent in RF, but in adaboost the error made by 1 stump influences the other.

→ Amount of say $= \frac{1}{2} \log\left(\frac{1 - \text{Total Eu}}{\text{Total Eu}}\right)$

→ New sample weight = sample weight $\times e^{(\text{amount of say})}$

Down sample weight = sample weight $\times e^{(-\text{amount of say})}$

Gradient Boost

→ GB starts by making a leaf (based on the average of all the values.

→ Based on this leaf GB starts to build a tree. (Its made based on the error from the leaf.

→ Tree from GB is larger than stump from adaboost.

→ The trees are subsequently built based on the error made from the previous tree and are scaled.

→ The trees are built until the no of trees specified are built or if the fit of the model doesn't improve further.

→ As a rule of thumb generally 8 to 32 leaves are used to build a tree in GB.

→ Main ideas:

i) In GB regression take the response column and find the average of them values.

ii) This average is the initial prediction made by the first tree.

iii) Now, calculate the residuals and update it in the new column

iv) Based on the calculated residuals build a tree to predict the residuals.

v) Adding the previous prediction to the newly predicted residual is the new predicted value, but this value overfits the data.

vi) In order to minimize the overfit we scale it with learning rate which is typically between $[0, 1]$.

vii) And then residuals are calculate again, and new prediction is the sum of all predictions from first tree.

(Pseudo residuals)

Loss function for GB is:

$$\frac{1}{2}(\text{Observed} - \text{Predicted})^2$$

1) Initialize model with constant value

$$F_0(x) = \underset{\gamma}{\operatorname{argmin}} \sum_{i=1}^{n} L(y_i, \gamma)$$

$\gamma \Rightarrow$ Predicted values

2) for $m = 1$ to M: ↗ calculates the residuals

i) $$r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]$$ For $i = 1, \ldots, n$

$F(x) = F_{m-1}(x)$

$i \Rightarrow$ sample number, $m \Rightarrow$ tree

ii) Fit a regression tree to the r_{im} values and create terminal regions

$R_{j,m}$, for $j = 1 \ldots J_m$

iii) For $j = 1 \ldots J_m$ compute

$$\gamma_{jm} = \underset{\gamma}{\operatorname{argmin}} \sum_{x_i \in R_{ij}} L(y_i, F_{m-1}(x_i) + \gamma)$$

↓ using previous prediction

iv) Update:

$$F_m(x) = F_{m-1}(x) + \alpha \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$$

3) Output $F_m(x)$ ⇒ Findl prediction

Gradient Boost for classification:

→ Here instead of average we calculate the log (odds) of the response variable.

→ Convert the log (odds) to probability for the purpose of classification.

→ calculate the pseudo residuals same as the ones in regression. (calculated from prob)

→ Use the columns to predict the residuals.

→ Here the residuals are obtained from probability and the previous predicted output is log (odds) value, so both the values cannot be added.

→ Performing transformation:

$$\frac{\sum \text{Residual}_i}{\sum [\text{Previous Probability}_i \times (1 - \text{Previous Probability}_i)]}$$

→ The above expression gives the final value from the leaf.

→ new pred = previous log (odds) prediction + α × output value from tree
(log(odds) Pred)

α ↓ learning rate

→ Convert new log (odds) pred into Probability.

→ Repeat the same process again.

→ Repeat the process until reached specified no of trees (or) residual becomes really small.

→ To get final classification beginning from the first tree (leaf) add all the log (odds) value, to get the final log (odd) values from all the trees.

→ convert this log (odds) into a probability. If the prob > 0.5 (belongs to class 1) if the less than 0.5 (belongs to class 0).