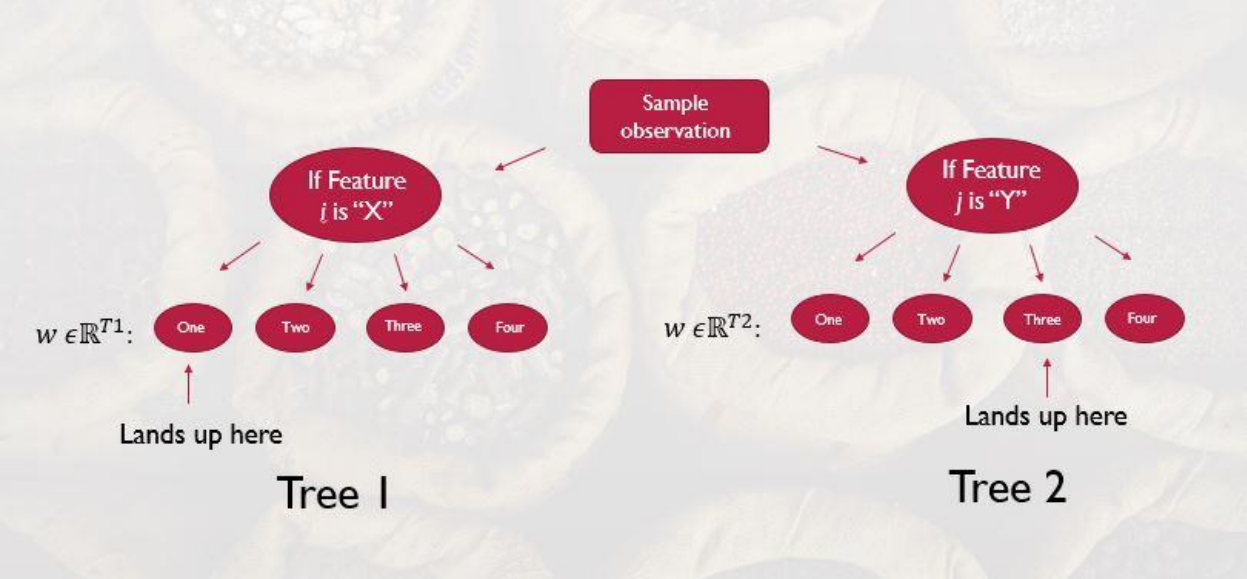
eXtreme Gradient Boosting (XGBoost)

XGBoost stands for eXtreme Gradient Boosting. The motivation for boosting was a procedure that combi nes the outputs of many "***weak***" classifiers to produce a powerful "***committee***." The purpose of boosting i s to sequentially apply the weak classification algorithm to repeatedly modified versions of the data, there by producing a ***sequence of weak classifiers.*** Boosting is a way of fitting an additive expansion in a set of elementary "***basis***" functions where the basis functions are the individual classifiers, in our case. The choi ce of weak learner for boosting is evident from the following diagram

Characteristic	Neural Nets	SVM	Trees	MARS	k-NN, Kernels
Natural handling of data of "mixed" type	▼	▼	▲	▲	▼
Handling of missing values	▼	▼	▲	▲	▲
Robustness to outliers in input space	▼	▼	▲	▼	▲
Insensitive to monotone transformations of inputs	▼	▼	▲	▼	▼
Computational scalability (large N)	▼	▼	▲	▲	▼
Ability to deal with irrel- evant inputs	▼	▼	▲	▲	▼
Ability to extract linear combinations of features	▲	▲	▼	▼	◆
Interpretability	▼	▼	◆	▲	▼
Predictive power	▲	▲	▼	◆	▲

To carry out the supervised learning using boosted trees we need to redefine 'tree'. In a way, Regression t ree is a function that maps the attributes to the score. A tree can be defined a vector of leaf scores and a le af index mapping function. The ***structure*** of individual tree ($q(x)$) guides a sample to a leaf and the ***assoc iated score*** w_i is assigned as the prediction for that sample for that tree.
The diagram and the equation explains above statements.

$$f_t(x) = w_{q(x)}, \quad w \in \mathbf{R}^T, q : \mathbf{R}^d \to \{1, 2, \cdots, T\}$$



On the basis, the ***prediction model*** ($\hat{y}$) can be written as the aggregation of all the prediction score for eac h tree for a ***sample*** (x). Particularly for i-th sample,

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i), f_k \in \mathcal{F}$$

where $\boldsymbol{K}$ is the number of trees, $\boldsymbol{f}$ is the function in the functional space $\boldsymbol{\mathcal{F}}$ and $\mathcal{F}$ is the all possible set of tr ees having prediction score in each leaf, slightly different from decision tree which only contains decisio n values. As the modelling is done, we need to optimize certain objective function to choose the best mod el for the training data. Here, we encourage a model to have high predictive power as well as to have a si mple in nature (deals with less number of features). As we know minimizing loss function ($L(\Theta)$) encoura ges predictive models as well as optimizing regularization ($\Omega(\Theta)$) encourages simpler model to have smal ler variance in future predictions, making prediction stable. The closed form of the objective is given belo w:

$$Obj(\Theta) = L(\Theta) + \Omega(\Theta)$$

For our tree ensemble model, we have our objective function which is to minimize as below:

$$obj(\Theta) = \sum_{i}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k)$$

For multiclass classification, the loss function is loss for multiclass prediction. In XGBoost model, we spe cifically optimized the softmax objective for which the objective loss function is:

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{j=1}^{k} 1\left\{y^{(i)} = j\right\} \log \frac{e^{\theta_j^T x^{(i)}}}{\sum_{l=1}^{k} e^{\theta_l^T x^{(i)}}}\right]$$

Since the regression is over a set of functions, usual Stochastic Gradient Descent wouldn't work here. To solve this, an ***additive*** strategy has been taken to add a new tree at each iteration. Starting from constant pr ediction (usually 0), the task is to add a new function each iteration.

$$
\begin{aligned}
\hat{y}_i^{(0)} &= 0 \\
\hat{y}_i^{(1)} &= f_1(x_i) = \hat{y}_i^{(0)} + f_1(x_i) \\
\hat{y}_i^{(2)} &= f_1(x_i) + f_2(x_i) = \hat{y}_i^{(1)} + f_2(x_i) \\
&\dots \\
\hat{y}_i^{(t)} &= \sum_{k=1}^{t} f_k(x_i) = \hat{y}_i^{(t-1)} + f_t(x_i)
\end{aligned}
$$

The objective can be broken into parts and written as dependent only on current iteration *t*:

$$
\begin{aligned}
Obj^{(t)} &= \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t)}) + \sum_{i=1}^{t} \Omega(f_i) \\
&= \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \Omega(f_t) + constant
\end{aligned}
$$

For square loss,

$$
\begin{aligned}
Obj^{(t)} &= \sum_{i=1}^{n} (y_i - (\hat{y}_i^{(t-1)} + f_t(x_i)))^2 + \sum_{i=1}^{t} \Omega(f_i) \\
&= \sum_{i=1}^{n} [2(\hat{y}_i^{(t-1)} - y_i) f_t(x_i) + f_t(x_i)^2] + \Omega(f_t) + constant
\end{aligned}
$$

Using Taylor expansion,

$$
Obj^{(t)} = \sum_{i=1}^{n} [l(y_i, \hat{y}_i^{(t-1)}) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \Omega(f_t) + constant
$$

where

$$
\begin{aligned}
g_i &= \partial_{\hat{y}_i^{(t)}} l(y_i, \hat{y}_i^{(t-1)}) \\
h_i &= \partial^2_{\hat{y}_i^{(t)}} l(y_i, \hat{y}_i^{(t-1)})
\end{aligned}
$$

As the derivation says, the new form of optimizing goal is:

$$
\sum_{i=1}^{n} [g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \Omega(f_t)
$$

The advantage of this is any loss function can be applied to optimize as the calculation just depends upon the first and second Taylor series coefficients.

The model complexity for a tree comprises of ***number of leaves*** in a tree and ***L2 norm of the scores on le aves*** which ensures normalization of leaf scores. Even intuitively the expression is obvious since as the nu mber of leaves is optimized (minimized) the tree will have a simpler model. For optimizing the L2 norm o f the leaf scores would try to normalize the score on each leaf and will prevent having a high score on a pa rticular leaf. Two constants, gamma and lamba are the Lagrangian multipliers and can be tuned for accura cy.

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2$$

To derive an expression for structure score, the re-written objective function in terms of scores would be:

$$\begin{aligned} Obj^{(t)} &\approx \sum_{i=1}^{n} [g_i w_q(x_i) + \frac{1}{2} h_i w_{q(x_i)}^2] + \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2 \\ &= \sum_{j=1}^{T} [(\sum_{i \in I_j} g_i) w_j + \frac{1}{2} (\sum_{i \in I_j} h_i + \lambda) w_j^2] + \gamma T \end{aligned}$$

which immediately gives the optimal score to optimize the objective function:

$$w_j^* = -\frac{G_j}{H_j + \lambda}$$

(The objective is now a quadratic function of scores)

$$G_j = \sum_{i \in I_j} g_i \;\; H_j = \sum_{i \in I_j} h_i$$

In this way, in each iteration, we are able to choose an optimized tree which optimizes the objective functi on which has been already optimized partly up to previous iteration, which ensures better accuracy. The o ptimal score is the best score function for a given structure of tree and ***optimal objective reduction*** measur es how good is a tree structure for a particular iteration so that it could minimize the objective function wh ich is given below.

$$Obj^* = -\frac{1}{2} \sum_{j=1}^{T} \frac{G_j^2}{H_j + \lambda} + \gamma T$$

Due to impossibility of enumerating all the tree from the function space, a greedy approach is of practical use which ensure ***an optimal split***. The gain for a split can be formulated as:

$$Gain = \frac{1}{2} \left[\frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{(G_L + G_R)^2}{H_L + H_R + \lambda} \right] - \gamma$$

The components are the score on the new left leaf, the score on the new right leaf, the score on the original leaf and the complexity cost by introducing additional leaf (gamma). It is obvious that, if gain is smaller than γ, we would better not to add that branch, which is nothing but ***pruning***.

Given the description of XGBoost, it is time to differentiate the model with Random Forest:

- The fact is Random Forest and Boosted Trees are not different in terms of model, the difference is how we train them.
- The major reason is in terms of training objective, Boosted Trees tries to add new trees (additive training) that compliments the already built ones. This normally gives you ***better accuracy with less trees*.**
- In Random Forest the regularization factor is missing, hence if the gain in splitting is greater than epsilon where epsilon is an infinitesimally small positive number, the split will happen. Whereas, in Boosted trees, there is control on model complexity which reduces overfitting.