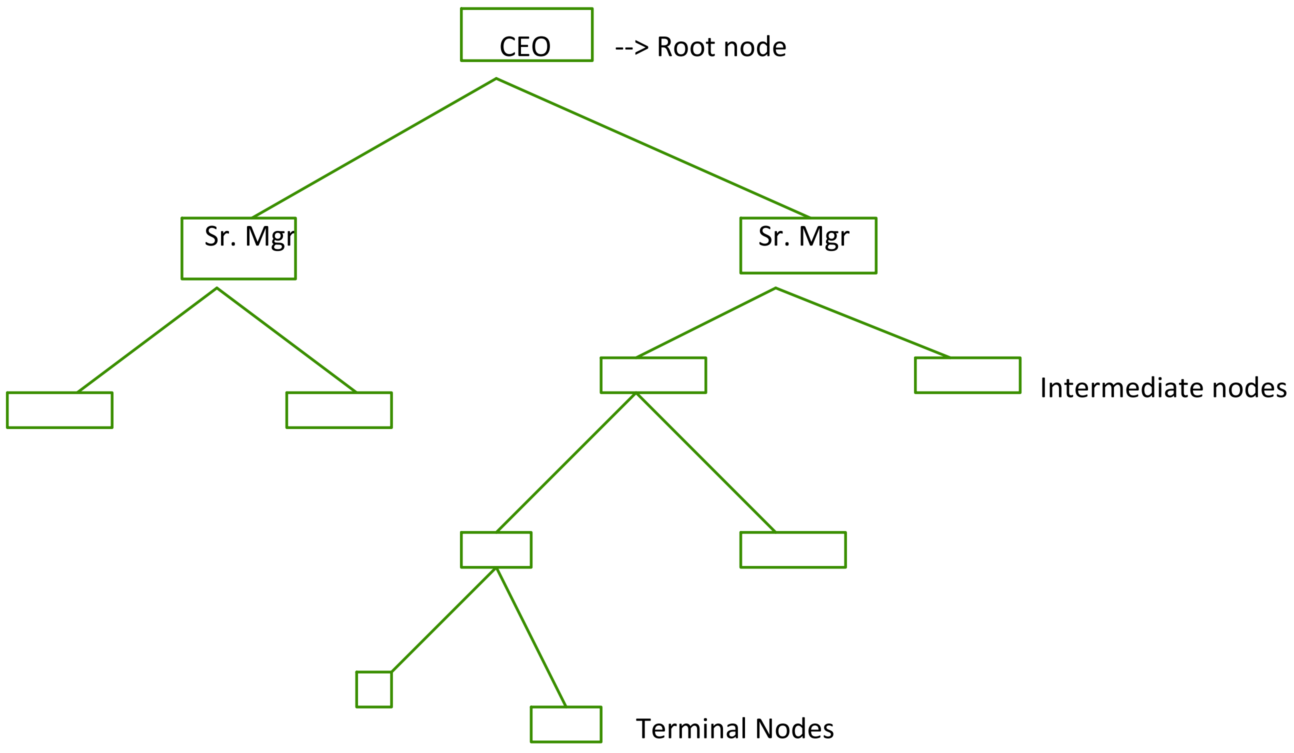
Decision Tree

18 April 2022 11:13

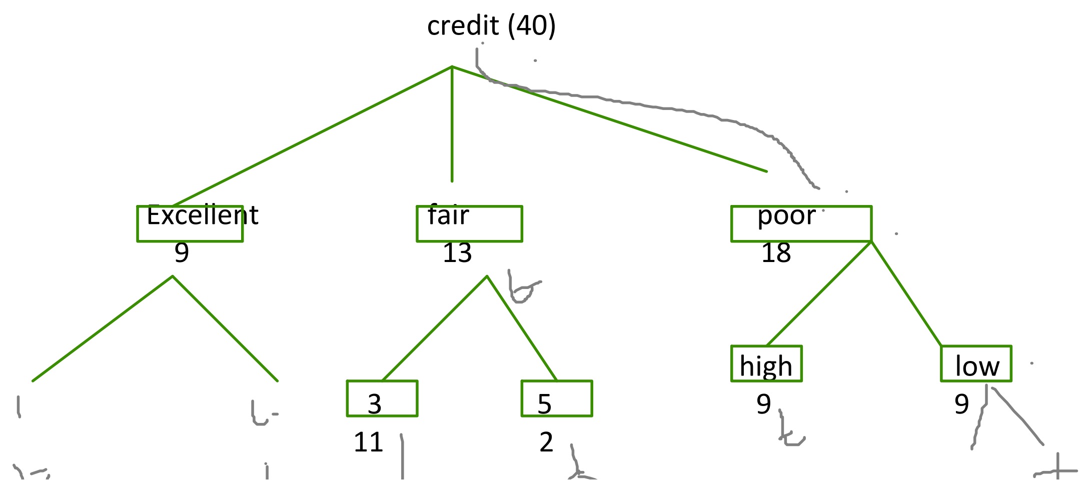
It can be applied for both kind of problems

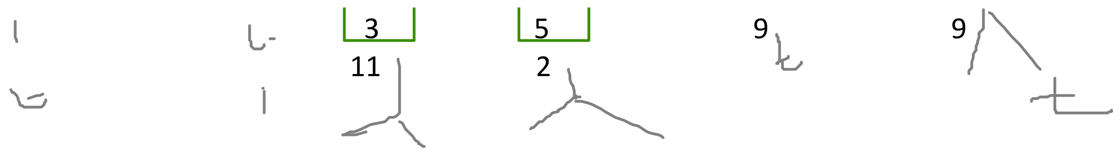
1. Regression technique
2. Classification technique



To identify the Root variables, we have many techniques

1. Miss classification Error (R only)
2. Gini Index (R and Python) --> Classification
3. Information gain / Entropy (Python) --> Classification
4. Mean square error (Python) --> Regression
5. ID3
6. CHAID
7. Variance reduction technique
8. C4.5





Gini Index

Gini =

$$\sum_{i=1}^{C}(p_i)^2$$

$\Sigma\ Wi(p2+q2)$

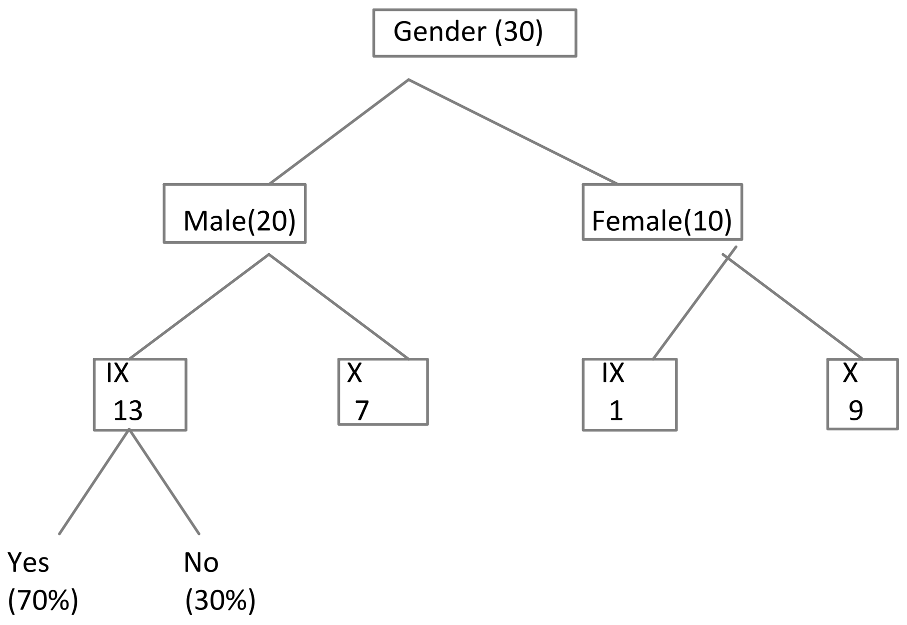
Gini index will be calculated for each of the X variable and will see which X variable contains highest Gini index that variable will become Root Variable.

Steps to Calculate Gini for a split

1. Calculate Gini for sub-nodes, using formula sum of square of probability for success and failure (p^2+q^2).
2. Calculate Gini for split using weighted Gini score of each node of that split

3. To calculate the exact Gini score by multiplying Total(step 1 * step 2)

We have to verify same calculative method for each of the x variable and finalized with one of the X variable.



When you run this decision trees, our accuracies are actually not in stable.

One single decision tree will be not enough to learn.

Ensemble methods:

1. Parallel ensemble methods
2. Sequential ensemble methods.

Parallel ensemble methods:

1. Bagging
2. Random Forests

Bagging: Bagging classifier, Bagging Regressor

n_estimators = 100, data size = (1000, 10), Max_sample = 0.6, base_estimator = Decision Tree

1. Tree1 ---> 0.6 --> (600, 10) ---> Accuracy ----> 0.79
2. Tree2 ---> 0.6 --> (600, 10) ---> Accuracy ----> 0.83
3. Tree3 ---> 0.6 --> (600, 10) ---> Accuracy ----> 0.85

100. Tree3 ---> 0.6 --> (600, 10) ---> Accuracy ----> 0.82

Average (0.81, 0.83, 0.82,……0.82) = 0.82

Max_sample is tuning parameter

0.1, 0.2, 0.3, 0.4, 0.5……..0.9 --> what percentage of samples we get higher average accuracy that is final.

Note: when more number of samples which are greater than 10,000 are there in data set, prefer Bagging method.

Random Forests: This is exclusively for Decision Tree only

RandomForestsClassifier, RandomForestsRegressor

n_estimators = 100, data size = (1000, 10), Max_features = 0.6,

1. Tree1 ---> 0.6 --> (1000, 6) ---> Accuracy ----> 0.79
2. Tree2 ---> 0.6 --> (1000, 6) ---> Accuracy ----> 0.83
3. Tree3 ---> 0.6 --> (1000, 6) ---> Accuracy ----> 0.85

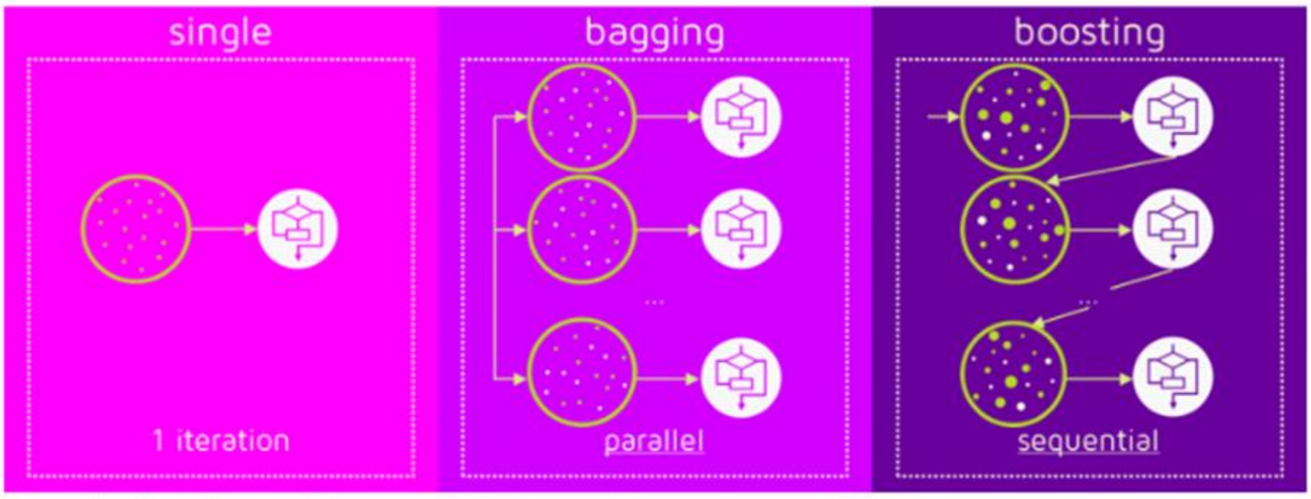
4. Tree3 ---> 0.6 --> (1000, 6) ---> Accuracy ----> 0.82

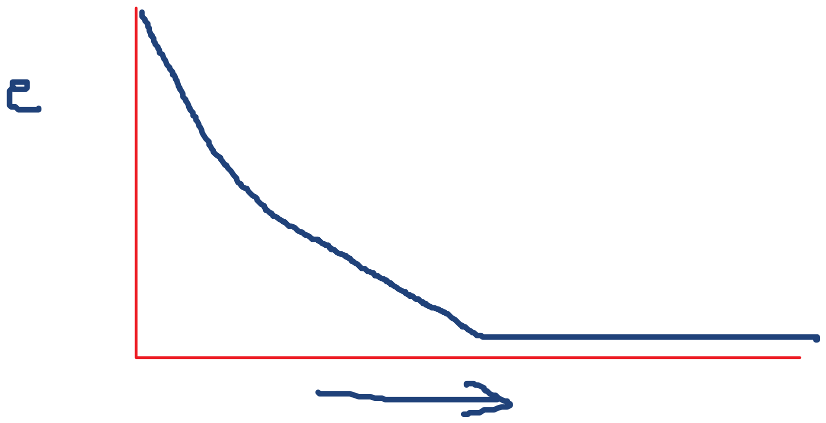
Average (0.79, 0.83, 0.85,......0.82) = 0.83

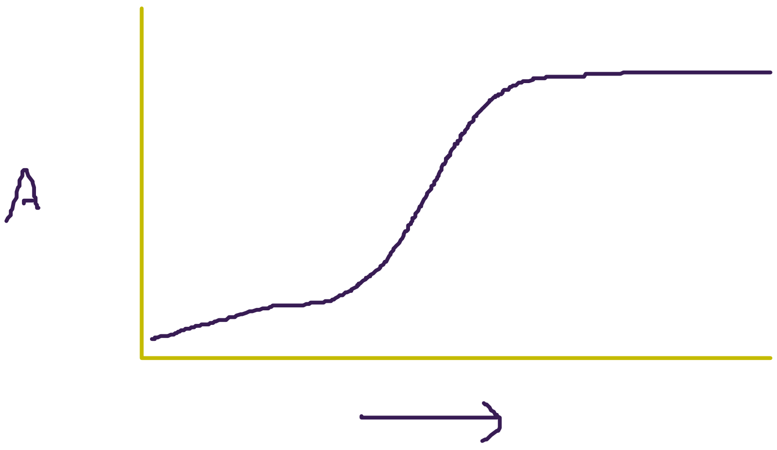
0.1, 0.2, 0.3, 0.4, 0.5........0.9 --> what percentage of variables we get higher average accuracy that is final.

Note: when more number of variables are there in data set, prefer Random Forests.

Sequential ensemble methods.







Sequential ensemble methods:

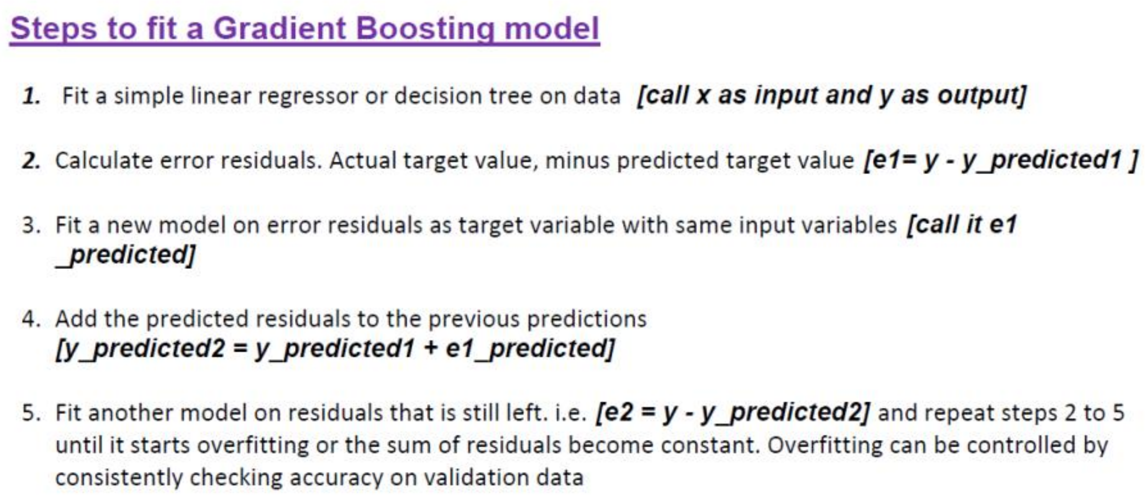
1. Gradient Boosting
2. Adative Boosting --> AdaBoost
3. Extreme Gradient Boosting --> XGBoost
4. LP Boosting
5. Brown Boosting

Gradient Boosting:

We should have the knowledge of Gradient Descent

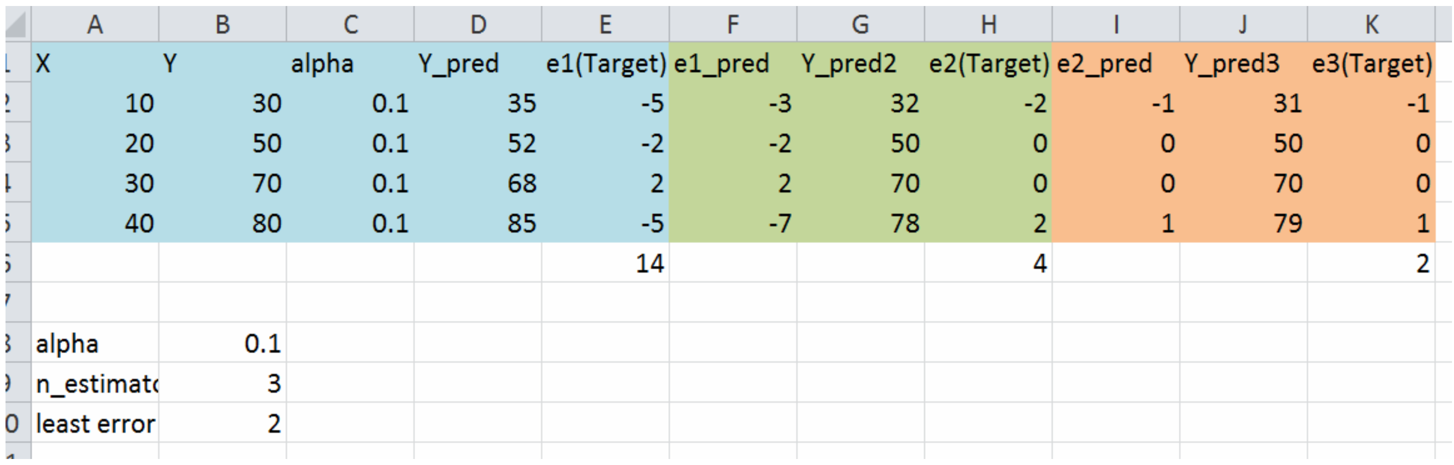
2x + 1

Steps for Gradient Boosting:



Steps to fit a Gradient Boosting model

1. Fit a simple linear regressor or decision tree on data ***[call x as input and y as output]***
2. Calculate error residuals. Actual target value, minus predicted target value ***[e1= y - y_predicted1]***
3. Fit a new model on error residuals as target variable with same input variables ***[call it e1 _predicted]***
4. Add the predicted residuals to the previous predictions ***[y_predicted2 = y_predicted1 + e1_predicted]***
5. Fit another model on residuals that is still left. i.e. ***[e2 = y - y_predicted2]*** and repeat steps 2 to 5 until it starts overfitting or the sum of residuals become constant. Overfitting can be controlled by consistently checking accuracy on validation data



A	B	C	D	E	F	G	H	I	J	K
X	Y	alpha	Y_pred	e1(Target)	e1_pred	Y_pred2	e2(Target)	e2_pred	Y_pred3	e3(Target)
10	30	0.1	35	-5	-3	32	-2	-1	31	-1
20	50	0.1	52	-2	-2	50	0	0	50	0
30	70	0.1	68	2	2	70	0	0	70	0
40	80	0.1	85	-5	-7	78	2	1	79	1
				14			4			2
alpha	0.1									
n_estimat	3									
least error	2									