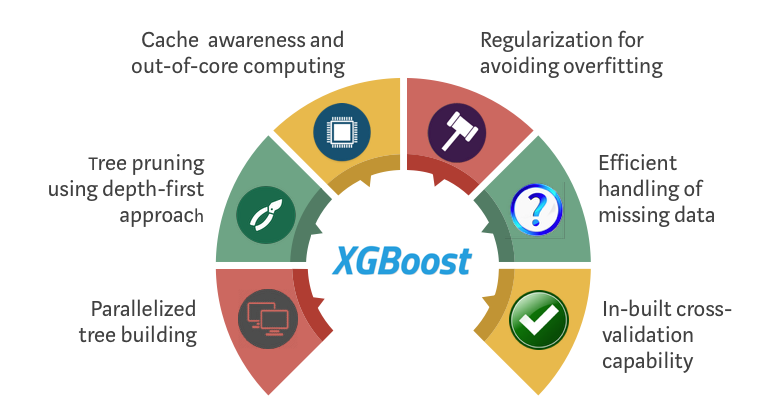
XGBOOST( Extreme Gradient Boosting )

Gradient boosting is a type of machine learning boosting. **It relies on the intuition that the best possible next model, when combined with previous models, minimizes the overall prediction error**.

XGBOOST Work on Input Data and Residual Values and forms a decision tree to find out the root node.

Advantages of XGBOOST over DECISION TREE,

* XGBoost **builds one tree at a time so that each data pertaining to the decision tree is taken into account and the data is filled if there are any missing data**.
* XGBoost offers a few technical advantages over other gradient boosting approaches, including **a more direct route to the minimum error, converging more quickly with fewer steps, and simplified calculations to improve speed and lower compute costs**



Now Let’s try to understand XGBOOST with very practical example mentioned below

sal = [ '49K', '49K', '49K', '51K', '51K', '51K', '49K' ]

credit = ['Bad', 'Good', 'Good', 'Bad', 'Good', 'Normal', 'Normal' ]

approval = [0, 1, 1, 0, 1, 1, 0]

d = { 'SalaryCat' :sal , 'Credit' : credit, 'Approval' : approval }

data = pd.DataFrame( d )

>>> data

SalaryCat Credit Approval

0 49K Bad 0

1 49K Good 1

2 49K Good 1

3 51K Bad 0

4 51K Good 1

5 51K Normal 1

6 49K Normal 0

In approval column, we have only 2 possible outcomes that is : 0 or 1.

P( 0 ) = P( 1 ) = 0.5 = 50%

We need to calculate residual first to understand how much the person is lagging or in safer side from approval status.

>>> data['Residual'] = data['Approval'] - 0.5

>>> data

SalaryCat Credit Approval Residual

0 49K Bad 0 -0.5

1 49K Good 1 0.5

2 49K Good 1 0.5

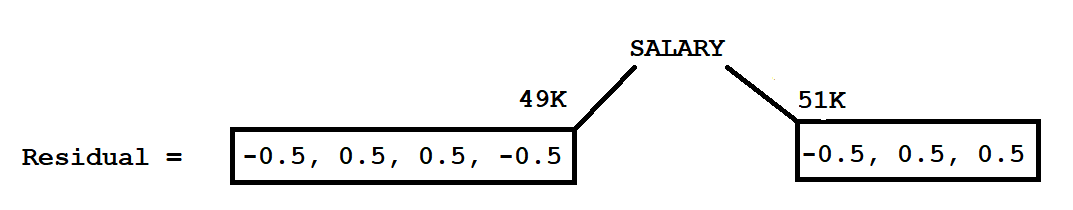
3 51K Bad 0 -0.5

4 51K Good 1 0.5

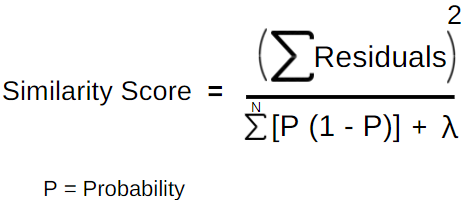
5 51K Normal 1 0.5

6 49K Normal 0 -0.5

XGBOOST will try to form a decision tree based on input columns and residuals. One thing we need to keep in mind is XGBOOST always does binary classification at any level of decision tree.



Now XGBOOST will calculate the similarity weight.



Similarity Score for 49K Category

Probability of Category 49K = 0.5

Numerator = ( -0.5 + 0.5 + 0.5 - 0.5 )2 = 0

Denominator = 0.5( 1-0.5) + 0.5( 1-0.5) + 0.5( 1-0.5) + 0.5( 1-0.5) + Lambda( Constant = 0 )

= 0.52 + 0.52 + 0.52 + 0.52 = 4 \* 0.52 = 4 \* 0.25 = 1

Similarity Score of 49K Category = 0/1 = 0

Similarity Score for 51K Category

Probability of Category 51K = 0.5

Numerator = ( -0.5 + 0.5 + 0.5 )2 = 0.25

Denominator = 0.5( 1-0.5) + 0.5( 1-0.5) + 0.5( 1-0.5) + Lambda( Constant = 0 )

= 0.52 + 0.52 + 0.52 = 3 \* 0.52 = 3 \* 0.25 = 0.75

Similarity Score of 51K Category = 0.25/0.75 = 1/3 = 0.3334

Similarity Score for whole salary

Probability of each category = 0.5

Numerator = ( -0.5 + 0.5 + 0.5 -0.5 -0.5 + 0.5 + 0.5 )2 = 0.52 = 0.25

Denominator = 0.5( 1-0.5) + 0.5( 1-0.5) + 0.5( 1-0.5) + 0.5( 1-0.5) + 0.5( 1-0.5) + 0.5( 1-0.5) + 0.5( 1-0.5) + Lambda( Constant = 0 )

= 0.52 + 0.52 + 0.52 + 0.52 + 0.52 + 0.52 + 0.52 = 7 \* 0.52 = 7 \* 0.25 = 1.75

Similarity Score of whole Category = 0.25/1.75 = 1/7 = 0.14285 = 0.14

Now XGBOOST will calculate the Gain = Similarity Score of 49K

+ Similarity Score of 51K

- Similarity Score of whole salary

= 0 + 0.33 - 0.14 = 0.19

Like this we need calculate the Similarity Score for Credit Columns also. Imagine we have calculated Similarity Score for Credit as well but Winner was Salary and we have kept the root node of tree as Salary,

data[ data['Credit'] == 'Bad' ]

SalaryCat Credit Approval Residual

0 49K Bad 0 -0.5

1 49K Good 1 0.5

2 49K Good 1 0.5

6 49K Normal 0 -0.5

>>> data[ data['SalaryCat'] == '51K' ]

SalaryCat Credit Approval Residual

3 51K Bad 0 -0.5

4 51K Good 1 0.5

5 51K Normal 1 0.5

