Decision Trees, Random Forest, XGBoost

1. How is best split chosen in decision trees/ Random Forests:
   a. What metrics are used?
      i. Gini Gain or Gini Impurity - Represents quality of feature to create a split
      ii. Information Gain - Represents quality of feature to create a split
   b. How is Gini Gain or Gini Impurity calculated?
   Example: we have 20 balls – 10 blue and 10 green – all balls have same weight but blue balls are larger in diameter than green balls. We want to find which feature is right to classify a ball as B or G.

   Impurity in the dataset:
   Dataset has equal number of B and G balls. Hence Probab of picking B = 0.5 and Probab of picking G = 0.5

   =>Probability of wrongly classifying a ball = Probab(picking a Blue ball)xProbab(classifying as Green Ball) + Probab(picking a Green ball)xProbab(classifying as Blue Ball)
   =>Probability of wrongly classifying a ball = P(Blue)xP(Green) + P(Green)xP(Blue)
   => Probability of wrongly classifying a ball = 0.5x0.5 + 0.5x0.5 = 0.5
   Hence Gini impurity of **Dataset** is 0.5.
   0.5 is the worst Gini Impurity you can have.

   Imperfect Split:
   Now if we use Weight as our feature and split the 20 balls into 2 branches; because all 20 balls have same weight – each branch would end up with 10 B & 10 G balls (most likely).

   Now take branch 1:
   =>Probability of wrongly classifying a ball = Probab(picking a Blue ball)xProbab(classifying as Green Ball) + Probab(picking a Green ball)xProbab(classifying as Blue Ball)
   =>Probability of wrongly classifying a ball = P(Blue)xP(Green) + P(Green)xP(Blue)
   => Probability of wrongly classifying a ball = 0.5x0.5 + 0.5x0.5 = 0.5
   Hence Gini impurity of Branch 1 is 0.5.

   Similarly, we can show that Gini impurity of Branch 2 is also 0.5.

   0.5 is the worst Gini Impurity you can have.

   Weighted Gini impurity = Fraction of total elements in branch1 * Gini-Impurity of branch1 + Fraction of total elements in branch2 * Gini-Impurity of branch2
   Weighted Gini impurity = (10/20)*0.5 + (10/20)*0.5 = 0.5

Gini gain = Impurity before splitting – Weighted Impurity after splitting
= 0.5 (of raw dataset) – 0.5 (weighted) = 0
Hence, we didn't witness any Gini gain => Weight is probably not the right criteria.

Perfect Split:

Suppose if we take weight as feature, then we would correctly able to classify all Blue balls in branch1 and all Green balls in branch 2.

Now take branch 1:
=>Probability of wrongly classifying a ball = Probab(picking a Blue ball)xProbab(classifying as Green Ball) + Probab(picking a Green ball)xProbab(classifying as Blue Ball)
=>Probability of wrongly classifying a ball = P(Blue)xP(Green) + P(Green)xP(Blue)
=> Probability of wrongly classifying a ball = 1x0 + 0x1 = 0
Hence Gini impurity of Branch 1 is 0 => there is no impurity

Similarly, we can show that Gini impurity of Branch 2 is also 0.

0 is the best/ideal Gini Impurity you can have.

Weighted Gini impurity = Fraction of total elements in branch1 * Gini-Impurity of branch1 + Fraction of total elements in branch2 * Gini-Impurity of branch2
Weighted Gini impurity = (10/20)*0 + (10/20)*0 = 0
Gini gain = Impurity before splitting – Weighted Impurity after splitting
= 0.5 (of raw dataset) – 0 (weighted) = 0.5
Hence, we witnessed a significant Gini gain => Diameter is probably a very good right criteria for splitting.

Semi-Perfect split:

Suppose 2 G balls have same diameter has B balls. Now If we use diameter as feature to split the dataset, we would have 12 balls (10B + 2G) in branch1 and branch2 has all 8G balls.

Now take branch 1:
=>Probability of wrongly classifying a ball = P(Blue)xP(Green) + P(Green)xP(Blue)
=> Probability of wrongly classifying a ball = (10/12)x(2/12) + (2/12)x(10/12) = 40/144 = 0.278
Hence Gini impurity of Branch 1 is 0.278

Now take branch 2:
=>Probability of wrongly classifying a ball = P(Blue)xP(Green) + P(Green)xP(Blue)
=> Probability of wrongly classifying a ball = (0/8)x(8/8) + (8/8)x(0/8) = 0
Hence Gini impurity of Branch 2 is 0

Weighted Gini impurity = Fraction of total elements in branch1 * Gini-Impurity of branch1 + Fraction of total elements in branch2 * Gini-Impurity of branch2
Weighted Gini impurity = (12/20)*0.278 + (8/20)*0 = 0.167
Gini gain = Impurity before splitting – Weighted Impurity after splitting
= 0.5 (of raw dataset) – 0.167 (weighted) = 0.333
Hence, we witnessed some Gini gain => Diamater may be a good criteria for splitting.

c. What is Information Gain?
It is similar to Gini Impurity. It uses entropy concept to quantify randomness in the split.

$$E = -\sum_{i}^{C} p_i \log_2 p_i$$

So, continuing the same example from above for Semi-perfect split.
Entropy in the Dataset:
As dataset has 10B and 10G = P(B) = 0.5, P(G) = 0.5
Entropy = -(0.5*$\log_2$(0.5) + 0.5* $\log_2$(0.5)) = 1

Entropy of 1 is the worst and Entropy of 0 is the best

Now take branch 1:
Branch 1 has 12 balls – 10B + 2G; P(B) = 10/12=0.83 & P(G) = 2/12=0.167
Entropy of Branch1 = -(0.83*$\log_2$(0.83) + 0.167* $\log_2$(0.167)) = 0.65
Hence Entropy of Branch 1 is 0.65

Now take branch 2:
Branch 2 has 8 balls – 0B + 8G; P(B) = 0 0.83 & P(G) = 1
Entropy of Branch2 = -(0 *$\log_2$(0) + 1 * $\log_2$(1)) = 0
Hence Entropy of Branch 2 is 0

Weighted Entropy = Fraction of total elements in branch1 * Entropy of branch1 + Fraction of total elements in branch2 * Entropy of branch2
Weighted Entropy = (12/20)*0.65 + (8/20)*0 = 0.39
Information gain = Entropy before splitting – Entropy after splitting
=1 (of raw dataset) – 0.39 (weighted) = 0.61
Hence, we witnessed some Information gain => Diameter may be a good criteria for splitting.

2. How is feature importance calculated?

    a. It is calculated by measuring the total decrease in node impurity averaged across all the trees. (Refer Q1 for more details on these metrics are calculated)

3. Boosting vs Bagging
   Bagging – **B**ootstrap **Agg**regating
    a. What is bootstrapping
        i. It is a sampling procedure to create subset of data. Particularly, in RF, we create m subsets with all n examples with replacement.
    b. Differences
        i. Boosting trains different machine learning models one after another (**sequentially**) to get the final result, while bagging trains them in **parallel**.
        ii. We know, Error in a model = Bias2 + Variance + Noise. **Bagging reduces Variance while Boosting reduces Bias**. (We know Decision Trees have high variance. So, we perform bagging to reduce variance)
        iii. Each tree in Bagging has same weight, while Boosting involves a set of weights which it assigns to each tree. So, at prediction stage, in Bagging – final prediction is average of all trees (or majority vote) while for Boosting it is weighted average of all trees predictions
        iv. In **Bagging** different training data subsets are randomly drawn with replacement from the entire training dataset. In **Boosting** every new subsets contains the elements that were misclassified by previous models.
        v. If the classifier is unstable (high variance), then we should apply **Bagging**. If the classifier is stable and simple (high bias) then we should apply **Boosting**.
        vi. **Bagging** is extended to Random forest model while **Boosting** is extended to **Gradient boosting**.
    c. When to choose Boosting and when to choose Bagging?
       Normally their selection depends on problem at hand.
        i. Bagging and Boosting decrease the variance of your single estimate as they combine several estimates from different models. So, the result may be a model with higher stability.
        ii. If the problem is that the single model gets a very low performance (aka underfitting, high bias), Bagging will rarely get a better bias. However, Boosting could generate a combined model with lower errors as it optimizes the advantages and reduces pitfalls of the single model.
        iii. By contrast, if the difficulty of the single model is over-fitting, then Bagging is the best option. Boosting for its part doesn't help to avoid over-fitting.
        iv. In fact, this technique is faced with this problem itself. For this reason, Bagging is effective more often than Boosting.
    d. What are similarities between Bagging and Boosting
        i. Both are ensemble learning method
        ii. Both makes Decision Trees stable
        iii. Both are good at reducing variance
    e. What is gradient boosting?

4. Can Boosting reduce bias?

    a. Yes. This is why they are popular.

5. Difference between GBDT and XGBoost. How do you parameter tune them?
6. Adaboost
7. More resources:
    https://victorzhou.com/blog/gini-impurity/
    https://victorzhou.com/blog/information-gain/
    https://www.kaggle.com/code/prashant111/bagging-vs-boosting