1 Catboost

4 types of importance

1. **Feature importance by Prediction Values Change (Internal Feature Importance)** — Internal Feature Importance is calculated using the exact same formula as Prediction Values Change. The difference between the importances is that Internal Feature Importance also returns the importance of automatically added combinations based on categorical features.

   $\sum_{trees, leafs_F}(v_1 - avr) \cdot c_1 + (v_2 - avr) \cdot c_2$

   $avr = \frac{v_1 \cdot c_1 + v_2 \cdot c_2}{c_1 + c_2}$

   - c_1, c_2 represent the total weight of objects in the left and right leaves, respectively. This weight equals the number of objects in each leaf if the dataset does not have specified weights.
   - v_1, v_2 represent the model's predictions. v_1 is the tree's response for examples that meet the splitting condition, and v_2 is the response for the remaining examples.

2. **Prediction Difference** – the feature importance for comparing decision-making between two specific objects.

   Definition Consider a CatBoost model $cb(x)$, trained to solve a binary classification problem. We have two objects from the test set: x_1 and x_2, with their true labels y_{true} both equal to class "0". Our interest lies in the feature with index 10, which varies within the interval $[0, 10]$. We will fix this feature and divide its range of values into n parts (bins).

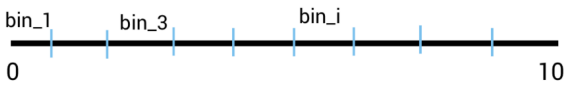

Figure 1: Bins

   (a) **For each bin from** 1 **to** n**:**

   (b) Change the value of the feature x_{10} so that it falls within the bin (i.e., $x_{10} \in \text{bin}_i, \forall i$), resulting in x'_{10}.

   (c) Compute the model prediction for each new x'_{10}.

(d) For each new obtained value, calculate $\text{difference}_{i1} = y_1 - cb(x'_{10})$ and $\text{difference}_{i2} = y_2 - cb(x'_{10})$.

After aggregating the obtained values across all bins and for the pair of objects, we compute the average change in the model's prediction when the feature value is changed. This average change reflects the importance of the feature of interest.

3. **Loss function change** — a method of calculating feature importance that computes the difference between the model loss with and without the feature.

   - $E_i v$ — the expected value of the model's prediction trained without the i-th feature;
   - v — the vector of predictions for the original dataset;
   - $metric$ — the loss function specified in the training parameters.

   Depending on the task, we compute the best value of the metric as:

   $$bestValue = \pm(metric(E_i v) - metric(v))$$

   Then, the feature importance according to the Loss Function Change is:

   $$featureImportance_i = |(metric(E_i) - bestValue| - |metric(v) - bestValue|$$

2 XgBoost and LightGbm

3 types of importance and 2 respectively.

1. **Cover importance** — represent the relative number of observations associated with this feature.

   Example Suppose we have 100 observations, 6 features, and 3 trees. Also, let feature $feature_1$ be used to determine the terminal node for 13, 5, and 2 observations in $tree_1$, $tree_2$, and $tree_3$ respectively.

   Then we calculate the cover of this object as:

   $$cover_1 = 13 + 5 + 2 = 20$$

   This will be calculated for all 6 features, and the final cover will be equal to 20, expressed as a percentage of the cover of all features. Thus:

   $$cover_{fi} = \frac{cover_i}{\sum_i^n cover_n}$$

where n is the number of features. In our case, $n = 3$.

Indeed, if we sum up all the covers, we get a value of approximately 100.

Important: Such a simple interpretation is valid only for a quadratic loss function (i.e., for linear regression). But explaining this metric for classification is simpler.

In the case of another loss function, it is the sum of the second-order gradient on the training data classified by the feature in the leaves. For this, the Hessian of the function is used.

2. **Gain importances** — represent the importance by gain in the nodes of the tree. It is analogous to the importance calculated in decision trees.

   In XGBoost, for gain importances, you can also obtain both the total sum and the sum normalized by the number of trees. In LightGBM, only the total sum is available. It is constructed according to the following algorithm:

   For each tree in the ensemble:

   - Calculate the contribution of each feature to the purity in the nodes of each tree in the model.
   - Average over the number of trees.

   A higher value of this metric compared to another feature indicates that it is more important for making predictions on average across the trees.

3. **Frequency or weight importance** — another metric that is computed out-of-the-box for XGBoost and LightGBM. It represents the number of times a particular feature appears in the trees of the model. The difference is that in LightGBM, weight importance is called split importance.

   Example:

   Let's consider we have 100 observations, 4 features, and 3 trees. Also, suppose feature $feature_2$ is used to determine the terminal node for some observations in $tree_1$, $tree_2$, and $tree_3$ respectively, and let $feature_2$ participates in 3, 2, and again 3 splits in trees $1, 2, 3$.

   Then its weight is calculated as:

   $weight = splits_1 + splits_2 + splits_3 = 3 + 2 + 2 = 7.$

   In general, the weight of a feature can be written as:

   $weight = splits_1 + splits_2 + ... + splits_j + ... + splits_n$, where n is the number of trees in the ensemble.

@telegram: sabrina_sadiekh