**CATBOOST**

CatBoost name comes from two words Category and Boosting. CatBoost is a decision tree based ensemble machine learning algorithm that uses a [gradient boosting](https://en.wikipedia.org/wiki/Gradient_boosting) framework. CatBoost can be used for both regression and classification problems.

An Ensemble is a classifier built by combining many similar classifiers or possibly different types of classifiers. For catboost this would mean running catboost classifier many times and taking as the final class label the most common prediction from all the classifiers. Generally, you want to have some variation between each of the classifiers that make up the ensemble - the best results will come when the errors that each classifier makes are uncorrelated. This happens when each of the classifiers are as different as possible.

CatBoost is freaking fast and it outperforms all the gradient boosting algorithms. It’s a good choice to train if most of the features in your dataset are categorical

**HOW IS CATBOOST DIFFERENT**

- Missing value support

Just leave NaN. It can treat missing values automatically.

- Feature and Data point importance

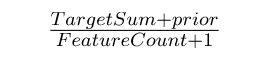
CatBoost gives not only important features. But it also tells us that for a given data point what are the important features.

- Handling Categorical Features

CatBoost relies on the ordering principle also called Target-Based with prior (TBS). By aligning the categorical features with the relative values of the target variables, the least amount of information loss is maintained in the implicit conversion of categorical features into a vector

CatBoost introduces an artificial time, a random permutation σ1 of the training examples. Then, for each example, it uses all the available history to compute its Target Statistic. Note that, using only one random permutation, results in preceding examples with higher variance in Target Statistic than subsequent ones. To this end, CatBoost uses different permutations for different steps of gradient boosting. Instead of considering all the data points, it will consider only data points that are past in time to a data point and calculates the mean to the target values of those data points having the same categorical feature.

Categorical feature values are encoded using the following formula:



TargetCount: It is sum of the target value for that particular categorical feature (upto the current one).

Prior: It is a constant value determined by (sum of target values in the whole dataset)/(total number of observations (i.e. rows) in the dataset)

FeatureCount: Total number of categorical features observed upto the current one with the same value as the current one.

For Example, if we have categorical feature column with values

Color = [“red”, “blue”, “blue”, “green”, “red”, “red”, “black”, “black”, “blue”, “green”]

Target= [1, 2, 3, 2, 3, 1, 4, 4, 2, 3]

Prior = 25/10 = 2.5

For the “red” category

TargetCount will be 1+3+1 = 5

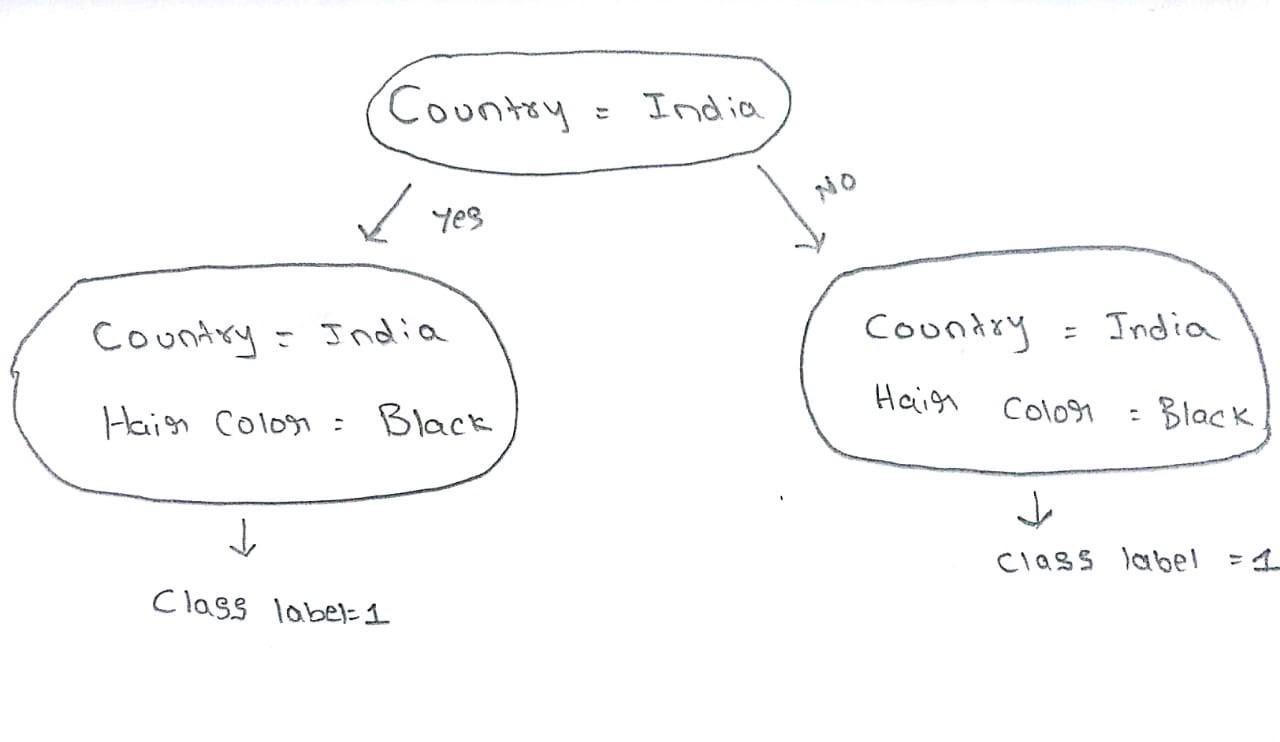
FeatureCount = 3

Encoded value for “red” = (5+2.5) / (3+1) = 1.875

- Categorical Feature Combinations

CatBoost combines multiple categorical features. For the most number of times combining two categorical features makes sense. CatBoost does this for you automatically.

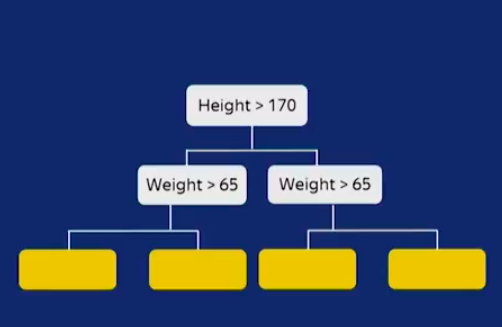




the symmetric tree structure in cat boost

- Base Tree Structure

CatBoost implements symmetric trees. This may sound crazy but helps in decreasing prediction time which is extremely important for low latency environments.



- Overfitting detector

By default, CatBoost has an overfitting detector that it stops training when CV error starts increasing. On small datasets, the GB is quickly over fitted. In CatBoost there is a special modification for such cases. That is, on those datasets where other algorithms had a problem with over fitted you won’t observe the same problem on CatBoost

- GPU support

Catboost can be efficiently trained on a GPU.

**HOW CATBOOST WORKS**

CatBoost has two modes for choosing the tree structure

- Plain

Plain mode corresponds to a combination of the standard gradient boosting decision tree algorithm (used in XGBoost) with an ordered Target Statistic

- Ordered

Under ordered tree structure, catboost algorithm make use of ordered boosting and random permutation. It divides a given dataset into random permutations and applies ordered boosting on those random permutations. By default, CatBoost creates four random permutations.

Here we calculate residuals for each data point using a model that has been trained on all the other data points at that time (To calculate residual for x5 datapoint, we train one model using x1, x2, x3, and x4). Hence we train different models to calculate residuals for different data points. In the end, we are calculating residuals for each data point that the corresponding model has never seen that datapoint before. We then train the model by using the residuals of each data point as class labels. This process is repeated for specified number of iterations

In ordered boosting method, we should train n different models to get residuals for n data points. This is computationally expensive when we have many data points. Hence there exists a modification and by default instead of training different models for each data point, it trains only log (num\_of\_datapoints) models. Now if a model has been trained on n data points then that model is used to calculate residuals for the next n data points. This procedure is called ordered boosting.