**What’s so special about CatBoost?**

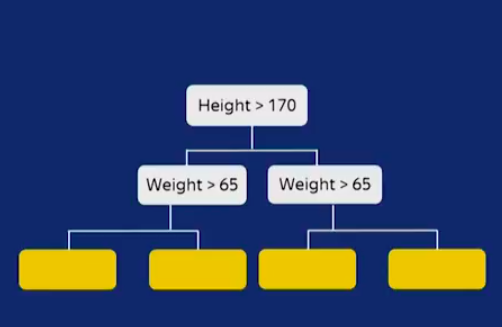
CatBoost is based on gradient boosting. A new machine learning technique developed by Yandex that outperforms many existing boosting algorithms like XGBoost, Light GBM.

While deep learning algorithms require lots of data and computational power, boosting algorithms are still needed for **most** business problems. However, boosting algorithms like XGBoost takes hours to train and sometimes you’ll get frustrated while tuning hyper-parameters.

On the other hand, CatBoost is easy to implement and very **powerful**. It provides excellent results in it’s very first to run. So, let’s find out what so special about CatBoost.

Base tree structure :

* One main difference between CatBoost and other boosting algorithms is that the CatBoost implements **symmetric trees**. This may sound crazy but helps in **decreasing prediction time**, which is extremely important for low latency environments.

Base Trees are symmetric in CatBoost

* Default max\_depth = 6

Procedure for other gradient boosting algorithms (XG boost, Light GBM)

Step 1: Consider**all**(or a sample ) the data points to train a highly biased model.

Step 2: Calculate residuals (errors) for each data point.

Step 3: Train another model with the **same** data points and corresponding residuals (errors) as class labels.

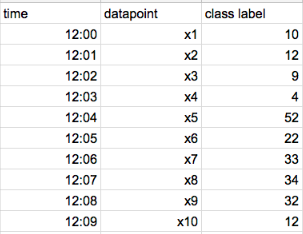
Step 4: Repeat Step 2 & Step 3 (*for n iterations*).

*This procedure is prone to overfitting, because we are calculating residuals of each data point by using the model that has already been trained on****same****set of data points.*

CatBoost Procedure

CatBoost does gradient boosting in a very elegant manner. Below is an explanation of CatBoost using a toy example.

Let’s say, we have 10 data points in our dataset and are **ordered in time**as shown below.

toy dataset

*If data****doesn’t****have time, CatBoost randomly creates an artificial time for each datapoint.*

* **Step 1**: Calculate residuals for each data point using a model that has been trained on all the **other data points at that time**(*For Example*, *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.
* **Step 2**: train the model by using the residuals of each data point as class labels
* **Step 3**: Repeat Step 1 & Step 2 (*for n iterations*)

For the above toy dataset, we should train 9 different models to get residuals for 9 data points. This is **computationally expensive** when we have many data points.

Hence 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.

* A model that has been trained on the first data point is used for calculating residuals of the second data point.
* Another model that has been trained on the first two data points is used for calculating residuals of third and fourth data points
* **and so on…**

*In the above toy dataset, now we calculate residuals of x5,x6,x7 and x8 using a model that has been trained on x1, x2,x3, and x4*.

All this procedure that I have explained until now is known as **ordered boosting.**

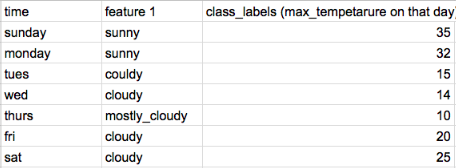
Random Permutations

CatBoost divides a given dataset into random permutations and applies ordered boosting on those random permutations. By default, CatBoost creates four random permutations. With this randomness, we can further stop overfitting our model. We can further control this randomness by tuning parameter *bagging\_temperature.*This is something that you have already seen in other boosting algorithms.

Handling Categorical Features.

* CatBoost has a very good vector representation of categorical data. It takes concepts of ordered boosting and applies the same to**response coding**.
* In response coding, we represent each categorical feature using the mean to the target values of **all**thedata points with the same categorical feature. We are representing a feature value of the data point with **its** class label. This leads to**target leakage.**
* CatBoost considers only the **previous data points to that time** and calculates the mean to the target values of those data points having the same categorical feature. Below is a detailed explanation with examples.

let’s take a toy dataset. **(all the data points are ordered in time/day)**

predicting max\_temperature using fearture\_1

*If data****doesn’t****have time, CatBoost randomly creates an artificial time for each datapoint.*

We have feature 1, a categorical feature that has 3 different categories.

With response coding, we represent cloudy = (15 +14 +20+25)/4 = 18.5

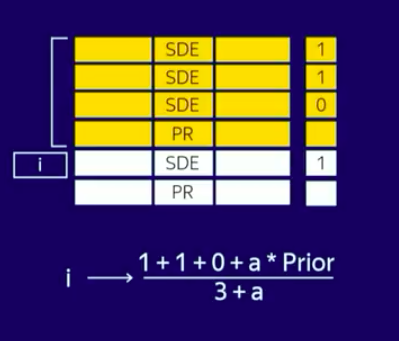
*This actually leads to target leakage. Because we are vectorising a data point using target value of the same datapoint.*

CatBoost vectorize all the categorical features without any target leakage. Instead of considering all the data points, it will consider **only data points that are past in time to a data point.** For Example,

* On Friday, it represents cloudy = *(15+14) /2* = *15.5*
* On Saturday, it represents cloudy = *(15+14+20)/3 = 16.3*
* But on Tuesday, it represents cloudy = 0/0?

To overcome this, we all knew what Laplace smoothing does in Naive Bayes. CatBoost implements the same.

Below is one another neat example,



In the above dataset, we have a feature with two categories(SDE, PR) and let’s assume that all the data points are ordered in time. For **i***th* data point, we represent SDE as (*with some constant added to the numerator and denominator to overcome 0/0 error*).

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.



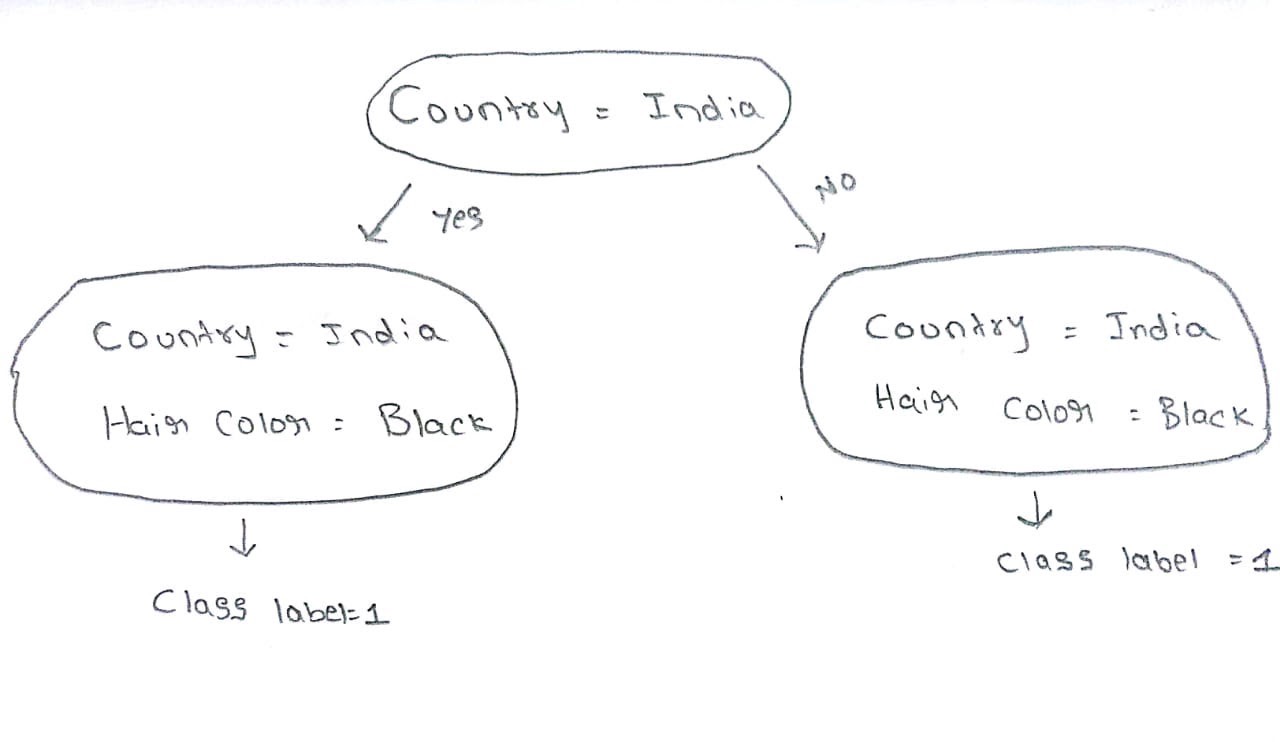
In this dataset, there are two features (country and hair length). We can easily observe that whenever a person is from India, his/her hair color black. We can represent those two features into a single feature. In the real world, many categorical features can represent a single feature.

CatBoost does feature combinations by building a base tree with the root node consisting only a single feature and for the child nodes, it randomly selects the other best feature and **represents** it along with the feature in the root node.

Below is the neat diagram of CatBoost representing two features as a single feature at level 2 of the tree.

Image for post

Image for post

the symmetric tree structure in cat boost

At the very first level of the tree, we have a single feature. When the level of tree increases, the number of categorical feature combinations increases proportionally.

One-hot Encoding in CatBoost

By default, CatBoost internally represents all the categorical features with One-hot encoding if and only if a categorical feature has two different categories.

* If you would like to implement One-hot encoding on a categorical feature that has **N** different categories then you can change parameter ***one\_hot\_max\_size****=****N****.*

Handling Numerical Features

CatBoost handles the numerical features in the same way that other tree algorithms do. We select the best possible split based on the Information Gain.

Limitations

* At this point, CatBoost **doesn’t**support sparse matrices**.**
* When the dataset has many numerical features, CatBoost takes more time to train than Light GBM.

CatBoost in various situations:

While *Hyper-parameter*tuning is not an important aspect for CatBoost. The most important thing is to set the right *parameters* based on the problem we are solving. Below are a *few* important situations.

1. When data is changing over time

We are living in the 21*st* century where the distribution of data changes recklessly over time. Especially in most of the internet companies, user preferences change over time to time. There are many situations in the real world where data changes over time. CatBoost can perform very well in these situations by setting parameter **has\_time = True.**

2. Low latency requirements

Customer satisfaction is the most important aspect of every business. A user usually expects very fast service from the website/model. CatBoost is the only boosting algorithm with very less prediction time. Thanks to its symmetric tree structure. It is comparatively **8x faster** then XGBoost while predicting.

3. Weighting data points

There are some situations where we need to give more importance to certain data points. Especially when you do temporal train-test split, you need the model to train mostly on the **earlier** data points. When you give more weightage to a data point, It has a higher chance of getting selected in the random permutations.

We can give more weightage to certain data points by setting parameter

For example, you can give **linear** weightage all the datapoints

***sample\_weight****=*[ x **for** x in **range**(train.shape[0])]

4. Working with small datasets

There are some instances when you have less number of data points and you need minimal Log-loss. In those situations you can set parameters ***fold\_len\_multiplier as close as to*1***(must be >1)*and ***approx\_on\_full\_history*** =**True** *.*With these parameters, CatBoost calculates residuals for each data point using a different model.

5. Working with large datasets

For large datasets, you can train CatBoost on GPUs by setting parameter **task\_type = GPU**. It also supports multi-server distributed **GPU**s. CatBoost also supports older GPUs that you can train it in Google Colabs.

6. Monitoring Errors / Loss function

It’s a very good practice to monitor your model for every iteration. You can monitor any metrics of your choice along with your optimizing loss function by setting parameter ***custom\_metric=****[‘AUC’,‘Logloss’]*.

You can visualize all the metrics that you did choose to monitor. Make sure that you have installed ***ipywidgets*** using pip to visualize plots in Jupyter Notebook and set parameter **plot = True.**

7. Staged prediction & Shrinking Models

This is again one powerful method provided by CatBoost library. You have trained a model and you want to know how your model predicts at a particular iteration. You can call the ***staged\_predict(***) method to check how your model performs at that stage. If you did notice that in a particular stage that the model is performing better than your final trained model, then you can use a ***shrink****( )* method to shrink the model to that particular stage. Check [documentation](https://catboost.ai/)for more info.

8. Handling different situations

Whether it’s a festival season or week-end or a normal day, the model should predict the best results in every given situation.

For this kind of problem, you can train different models ondifferent cross-validation datasets and blend all the models with some weights assigning to each model using the ***sum\_models*( )**method. Later **based** on the situation you could change the weights of each model.

Many More…

* By default, CatBoost has an overfitting detector that it stops training when CV error starts increasing. You can set parameter **od\_type** = **Iter**to stop training your model **after** few iterations.
* Like other algorithms, we can alsobalance**an**imbalanced dataset with the **class\_weight** parameter.
* CatBoost gives not only important features. But it also tells us that for a given data point what are the important features.
* Code for training CatBoost is simply straight forwarded and it is almost similar to the sklearn module. I have explained only some important aspects of CatBoost. You can further read the full documentation of CatBoost [here](https://tech.yandex.com/catboost/doc/dg/concepts/about-docpage/) for a better understanding.

Goodbye to Hyper-parameter tuning?

CatBoost which is implemented by powerful theories like ordered Boosting, Random permutations. It makes sure that we are not overfitting our model. It also implements symmetric trees which eliminate parameters like (min\_child\_leafs ). We can further tune with parameters like *learning\_rate*, *random\_strength*, *L2\_regulariser,* but the**results don’t vary much**.

EndNote:

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. You can further practice CatBoost on the assignments that are provided by the CatBoost team [here](https://catboost.ai/docs/concepts/tutorials.html). A model that is robust to over-fitting and with very powerful tools, what else you are waiting for? Start working on CatBoost !!!

References:

* <https://arxiv.org/abs/1706.09516>
* <https://papers.nips.cc/paper/7898-catboost-unbiased-boosting-with-categorical-features.pdf>