Final Project

CS 5350/6350 Jeonghoon Oh

December 2021

1 Problem definition and motivation

For the final project, I decided to compete in Kaggle competition for Income Level Prediction. The given dataset is provided with 14 features where 7 features are categorical features and 25000 cases for training dataset and 23842 cases for testing dataset. The training dataset contains a column that labels the case whether the case is labeled as income over 50k. Datawise, if the case is labeled as income over 50k it will be labeled as '1', else, it would be labeled as '0'. The goal of the competition is to predict the label for the testing dataset by creating a model using the provided features and labels in the training dataset and applying it on the testing dataset.

In order to make a model from this dataset, we need to use both categorical features and continuous features for building a prediction model. While making model out of both type of features is more complex to build then using single type of feature, it is very likely to happen in real life. Using such raw type of data makes the data scientist to think that they are solving the real world problem which would be the factor that makes it to be most motivating.

The biggest reason that the machine learning should be the way to solve the problem is that there exist 14 number of features that affects the final label and much more in the real world. As we integrate more and more features, it would be likely to approach closer to the perfect model. However

in a same time, the complexity of the model will grow exponentially which will be too complex for human to manually create, which is the reason we use machines to do exponential calculation for us.

2 Solution

In order to solve the problem, I have implemented the CatBoostRegressor from CatBoost library to the dataset. Originally, XGBoost was planned for implementation, but CatBoost was chosen as it was more adaptive to using multi type of features which would fit better for purpose of the problem. The type of solution takes integer mapping for categorical data in the pre-processing level and then create inequality decision node for those integer values. Even though these decision trees make recursive decision on same feature in depth, it would result in deeper tree and have high chance of overfitting. In this way, the model would be affected by the ordering of

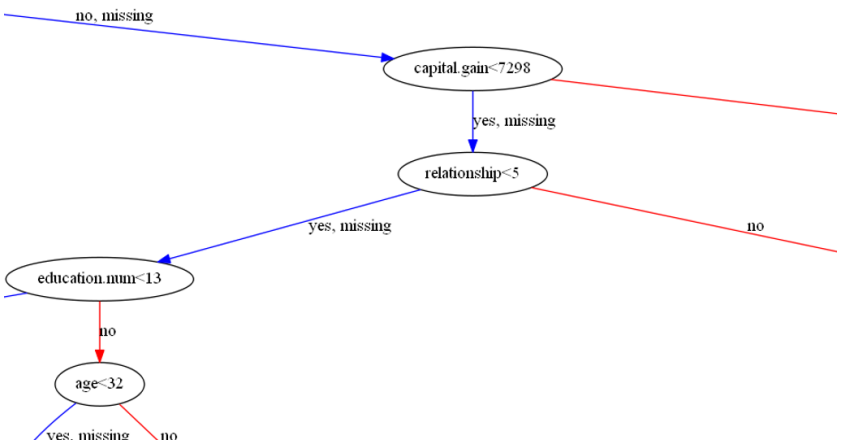

Figure 1: Decision node on categorical feature - XGBoost

feature mapping which is irrelevant to the solution and drop accuracy.

CatBoost is named after categorical boosting. As the name implies, it is designed typically for categorical variables and by it will transform categorical features with its nested functionality unlike creating dummy columns for

every attributes in the categorical features. User can label certain columns in the dataframe as cat_features for categorical variables.

The way CatBoost learns the model is by building gradient boosted tree. In each iteration, it will create sequential weak tree models where each tree models has their loss reduced by loss function attribute. There exists multiple type of loss function that could be plugged in as a parameter and the final model may end up in different shape based on which loss function the user choose.

CatBoostClassifier and CatBoostRegressor methods takes similar approach but uses different optimization techniques. I was not able to figure out exactly how those two metrics approach to the model but it was clear that for this problem, CatBoostRegressor metric built significantly better model.

3 Experimental results

In order to build a model, I have used CatBoostRegressor as a metric, number of estimators to be 200, loss function as MAPE and depth of tree to be 5. I have also set the categorical features labeled as categorical feature for training and testing data pool. Consequently, the model built from CatBoostRegressor showed great accuracy on the test, resulting 92.201% of accuracy. This was significant progress from my initial test using scikit-learn tree model (76.277%) and my XGBoost tree model (79.813%). Also, the speed of building model for CatBoostRegressor was extremely fast compared to the time taken for building model from scikit-learn library. The speed compared to time taken for building XGBoost tree model was somewhat slower but considering the improvement of accuracy, the time complexity was endurable.

From the Figure 2, we are able to observe that loss function tend to converge after around 20th iterations of estimators. We can learn from the fact that CatBoost use loss function to quickly drop their loss of weak tree model to

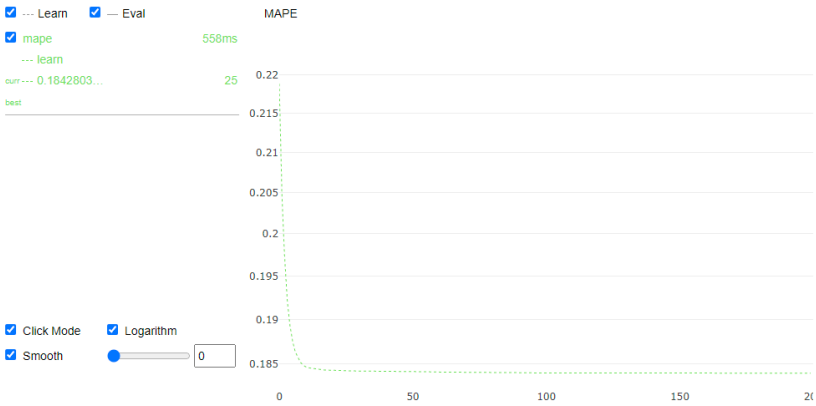

Figure 2: Loss for every iteration - MAPE

convergence and suggest that it is a method that only requires small amount of weak tree model to achieve ideal model.

In the Figure 3, we can observe how CatBoostRegressor creates its classifying nodes. instead of having single classifying features in the node, it contains (total depth - current depth) amount of classifying features where there exist total depth amount of unique features. In this way, I believe it was able to create ideal model without under-fitting.

4 Future Project

I am definitely satisfied with the model I got. However, I also have some ambiguity on the metrics I have used and I wish I could solve those ambiguity if I had some more time. First, I am not exactly clear of the difference between CatBoostClassifier and CatBoostRegressor. I know that they are using different optimization techniques and use different loss function because of such reason. I wish to know how exactly those two metrics approach to the model and what factors in the loss function prevents cross usage. For each of those loss functions, I would want to know how those loss functions result in different model and in what cases those loss function

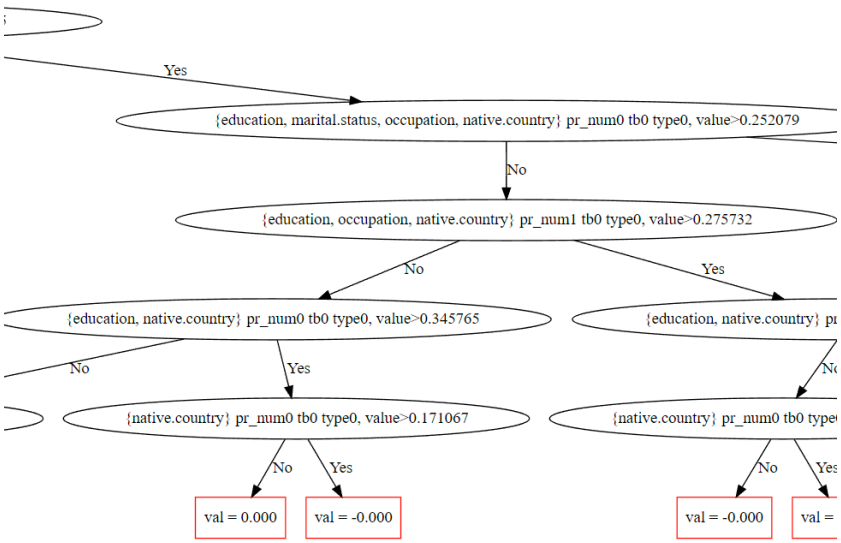

Figure 3: CatBoost Tree Model

would be best utilized. Finally, I want to know more about parameters for CatBoost metrics. There exists 96 amount of parameters for CatBoostRegressor and 101 amount of parameters for CatBoostClassifier. I believe that if I could have better understandings for the parameters for both metrics, I could tune the metrics to have better model as a result.

5 Git

`https://github.com/ihoon9203/ML_Project.git`