IITM-CS5691 : Pattern Recognition and Machine Learning — Release Date: 29th Apr, 2021
New Method Writeup — Due Date: 28th May, 2021

Roll No: EE18B035 — Name: Nithin Uppalapati

- Dear Student, You may have tried or thought of trying different methods for your data contest. Choose one of the methods that was not taught in class, and submit a writeup of this "new method" in the template provided below. This is an individual submission - i.e., while you would've done your kaggle submission as a team of two members or while you may've discussed this method with your teammate, **you will have to write about the new method in your own words independently and submit it individually**.
- **Template:** Fill in whatever fields are applicable for your algorithm (overall 1-2 page writeup; since some fields may not be applicable for certain methods, we haven't shown points below).

1. (points) [Name of Method, and its ML Problem and Paradigm: problem could be regression/classification/clustering/etc., and paradigm could be supervised/unsupervised/..., generative/discriminative/direct, linear/non-linear models, etc.)]:

   Solution:
   Name of the method: Gradient Boosting Machine Learning
   ML Problem: Regression (Gradient Boosted decision trees)
   Paradigm: Supervised, non-linear learning model

2. (points) [Brief introduction/motivation: One paragraph to describe briefly the new method (its name, what it does, its main application, etc.)]

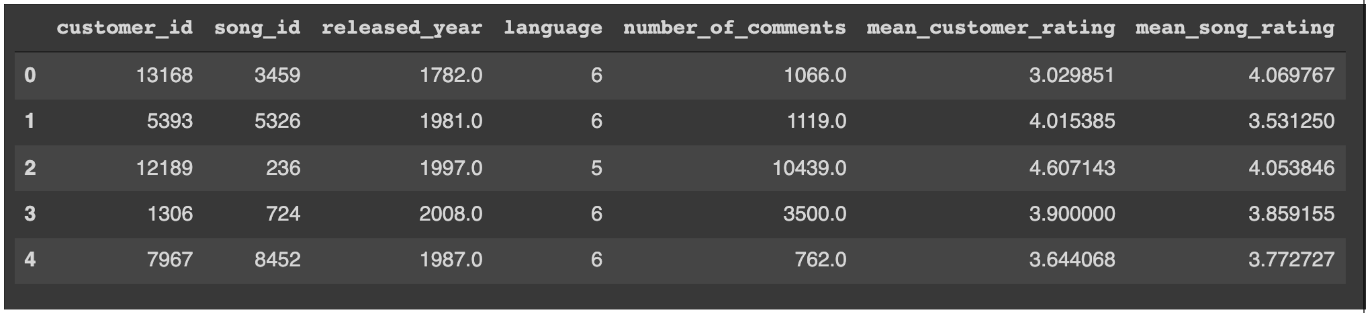
   Solution: The method we used in this contest is, CatBoostRegressor.
   Gradient boosting is a machine learning technique for regression and classification problems. CatBoost is based on gradient boosted decision trees. During training, a set of decision trees is built consecutively. Each successive tree is built with reduced loss compared to the previous trees. As gradient boosting fits the decision trees sequentially, the fitted trees will learn from the mistakes of parent trees and hence reduce the errors.

3. (points) [Closely related method seen in class, and relation of your selected new method to method you eventually used for the data contest]:

Solution: This method is related to Decision trees and Boosting, as discussed in class. It also involves bayesian hyperparameter tuning and grid search for finding optimal hyperparameters, which were also discussed in class.

4. (points) [Training Input and Output: (e.g., $\{x_i, y_i\}_{\{i=1...N\}}, x_i \in R^d, y_i \in \{-1, +1\}$, etc.]:

Solution:



	customer_id	song_id	released_year	language	number_of_comments	mean_customer_rating	mean_song_rating
0	13168	3459	1782.0	6	1066.0	3.029851	4.069767
1	5393	5326	1981.0	6	1119.0	4.015385	3.531250
2	12189	236	1997.0	5	10439.0	4.607143	4.053846
3	1306	724	2008.0	6	3500.0	3.900000	3.859155
4	7967	8452	1987.0	6	762.0	3.644068	3.772727

Training Input is a 7 dimensional vector, whose components are:

1st component is: Customer ID, a categorical variable.
2nd component is: Song ID, a categorical variable.
3rd component is: Released Year
4th component is: Language, a categorical variable.
5th component is: no. of comments, a categorical variable.
6th component is: mean customer rating
7th component is: mean song rating

Training Output is a 1-Dimensional vector (a number which lies between 1 - 5)

5. (points) [Training Objective function (e.g., loss function that is to be optimized) or probabilistic model (over which MLE or Bayesian inference done):]

Solution: The objective function to minimize is, loss $\ell(y, F(x))$ with $F(\cdot)$ being ensemble trees

$$F^* = \underset{F}{\operatorname{argmin}} \sum_{i=1}^{n} \ell\left(\mathbf{y}_i, F\left(\boldsymbol{x}_i\right)\right) \text{ with } F(\boldsymbol{x}) = \sum_{m=1}^{T} f_m(\boldsymbol{x})$$

(each f_m is a decision tree).

Direct loss minimization: at a given stage m, find best function to minimize loss.
$f_m = \text{argmin}_{f_m} \sum_{i=1}^{N} \ell\left(y_i, F_{m-1}\left(\boldsymbol{x}_i\right) + f_m\left(\boldsymbol{x}_i\right)\right)$
update $F_m \leftarrow F_{m-1} + f_m$ $F_m(\boldsymbol{x}) = \sum_{j=1}^{m} f_j(\boldsymbol{x})$
is the prediction of $\boldsymbol{x}$ after m iterations.

To approximate the current loss function by a quadratic approximation,

$$\begin{aligned}\sum_{i=1}^{n} \ell_i\left(\hat{y}_i + f_m\left(x_i\right)\right) &\approx \sum_{i=1}^{n}\left(\ell_i\left(\hat{y}_i\right) + g_i f_m\left(x_i\right) + \frac{1}{2} h_i f_m\left(x_i\right)^2\right) \\ &= \sum_{i=1}^{n} \frac{h_i}{2}\left\|f_m\left(x_i\right) - g_i / h_i\right\|^2 + \text{ constant}\end{aligned}$$

where $g_i = \partial_{\hat{y}_i} \ell_i\left(\hat{y}_i\right)$ is gradient, $h_i = \partial_{\hat{y}_i}^2 \ell_i\left(\hat{y}_i\right)$ is second order derivative.

Finding $f_m\left(\boldsymbol{x}, \theta_m\right)$ by minimizing the loss function:

$$\underset{f_m}{\text{argmin}} \sum_{i=1}^{N}\left[f_m\left(\boldsymbol{x}_i, \theta\right) - g_i / h_i\right]^2 + R\left(f_m\right)$$

6. (points) [Training Algorithm: Brief description of key aspects of the algorithm]

Solution: The algorithm used is: CatboostRegressor.train(), from Catboost library.

It is a gradient boosting machine learning. One of the key aspect of catboostregressor is it implements symmetric trees, which helps in decreasing prediction times. It also has inbuilt overfitting detector, which stop the further increase in iterations of boosting rounds and hence finds the optimal iterations in for new trees formed. The nodes are split in such a way that maximum informational gain is achieved.

Steps implemented by algorithm:
1. Computing the current gradient for each $\hat{y}_i$.
2. Building a base learner (decision tree) to fit the gradient.
3. Updating current prediction $\hat{y}_i = F_m(x_i)$ for all i.

So, each base learner is a decision tree and each regression tree approximates the functional gradient.

7. (points) [Testing Input and Output: (e.g., $x \in R^d$, $y \in \{-1, +1\}$)]

Solution: The testing input is the testing set of X vectors, same structure as that of training input, whereas, the training output is the predictions of scores of the testing

8. (points) [Testing Algorithm: Brief description of key aspects of the algorithm]

Solution: Based on the model trained on the train dataset, we give the testing input to the model and then we compute the predictions based on the model trained earlier.

9. (points) [Critique of the method: (1-2 paragraphs discussing its strengths and weaknesses in your own words)]

Solution: Strengths:
This algorithm has capability to directly handle categorical variables. Additionally it has a optional parameter for pointing out the categorical variables. We can also club multiple categorical features into a lesser number of categorical features. It also reduces the need for extensive hyper-parameter tuning and lower the chances of over-fitting also which leads to more generalized models. It also has better interpret-ability. The model with default parameters will give us a good results to start with.

Weaknesses:
In case there are dependencies in data, it needs to build deep decision trees to recover those dependencies and especially in case of features with high cardinality, which takes significant amount of compute power and also might consume more time. So, generally these models are slow in training setup. Whereas, faster in testing setup.
These models also tend to overfit the training data, and hard for implementing a general loss.