World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:13, No:1, 2019

Comparison between XGBoost, LightGBM and CatBoost Using a Home Credit Dataset

Essam Al Daoud

Abstract**—**Gradient boosting methods have been proven to be a very important strategy. Many successful machine learning solutions were developed using the XGBoost and its derivatives. The aim of this study is to investigate and compare the efficiency of three gradient methods. Home credit dataset is used in this work which contains 219 features and 356251 records. However, new features are generated and several techniques are used to rank and select the best features. The implementation indicates that the LightGBM is faster and more accurate than CatBoost and XGBoost using variant number of features and records.

Keywords**—**Gradient boosting, XGBoost, LightGBM, CatBoost, home credit.

I. Introduction

DESPITEthe recent re-rise and popularity of artificial neural networks (ANN), boosting methods are still more useful for a medium dataset because the training time is relatively very fast and it does not require a long time to tune its parameters.

Boosting is an ensemble strategy that endeavors to make an accurate classifier from various weak classifiers. This is done by dividing the training data and using each part to train different models or one model with a different setting, and then the results are combined together using a majority vote. AdaBoost was the first effective boosting method discovered for binary classification by [1]. When AdaBoost makes its first iteration, all records are weighted identically, but in the next iterations, more weight is given to the misclassified records, and the model will continue until an efficient classifier is constructed. Soon after AdaBoost was presented, it was noted that even if the number of iterations is increased, the test error does not grow [2]. Thus, AdaBoost is a suitable model for solving the overfitting problem. In recent years, three efficient gradient methods based on decision trees are suggested: XGBoost, CatBoost and LightGBM. The new methods have been used successfully in industry, academia and competitive machine learning [3].

The rest of this paper is organized as follows: Section II provides a short introduction about the gradient boosting algorithms and the recent developments. Section III explores the home credit dataset and exploits the knowledge of the domain to generate new features. Section IV implements gradient boosting algorithms and discusses a new mechanism to generate useful random features, and the conclusion is provided in Section V.

II. Related Work

Gradient boosting methods construct the solution in a stage-wise fashion and solve the over fitting problem by optimizing the loss functions. For example, assume that you have a custom base-learner *h(x, θ)* (such as decision tree), and a loss function $\psi(y, f(x))$; it is challenging to estimate the parameters directly, and thus, an iterative model is suggested such that at each iteration. The model will be updated and a new base-learner function $h(x, \theta_t)$is selected, where the increment is guided by:

$$g_t(x) = E_y\left[\frac{\partial\psi(y, f(x))}{\partial f(x)} |x\right]_{f(x)=\hat{f}^{t-1}(x)}$$

This allows the substitution of the hard optimization problem with the usual least-squares optimization problem:

$$(\rho_t, \theta_t) = \arg min_{\rho,\theta} \sum_{i=1}^{N}[-g_t(x_i) + \rho\, h(x_i, \theta)]^2$$

Algorithm 1summarizes the Friedman algorithm.

Algorithm 1 Gradient Boost

```
1-  Let $\hat{f}_0$be a constant
2-  For i= 1 to M
        a.  Compute $g_i(x)$ using eq()
        b.  Train the function $h(x, \theta_i)$
        c.  Find $\rho_i$ using eq()
        d.  Update the function
                $\hat{f}_i = \hat{f}_{i-1} + \rho_i h(x, \theta_i)$
3-  End
```

The algorithm starts with a single leaf, and then the learning rate is optimized for each node and each record [4]-[6].

eXtreme Gradient Boosting (XGBoost) is a highly scalable, flexible and versatile tool; it was engineered to exploit resources correctly and to overcome the limitations of the previous gradient boosting. The main difference between XGBoost and other gradient boosting is that it uses a new regularization technique to control the overfitting. Therefore, it is faster and more robust during the model tuning. The regularization technique is done by adding a new term to the loss function, as:

$$L(f) = \sum_{i=1}^{n} L(\hat{y}_i, y_i) + \sum_{m=1}^{M} \Omega(\delta_m)$$

with

$$\Omega(\delta) = \alpha|\delta| + 0.5\beta\|w\|^2$$

E. Al-Daoud is with faculty of Information Technology, Computer Science Department, Zarka University, Jordan (phone: +96279668000, e-mail: essamdz@zu.edu.jo).

where |δ| is the number of branches, *w* is the value of each leaf and *Ω*is the regularization function. XGBoost uses a new gain function, as:

$$G_j = \sum_{i \in I_j} g_i$$
$$H_j = \sum_{i \in I_j} h_i$$
$$Gain = \frac{1}{2}\left[\frac{G_L^2}{H_L+\beta} + \frac{G_R^2}{H_R+\beta} - \frac{(G_R+G_L)^2}{H_R+H_L+\beta}\right] - \alpha$$

where

$$g_i = \partial_{\hat{y}_i} L(\hat{y}_i, y_i)$$

and

$$h_i = \partial^2_{\hat{y}_i} L(\hat{y}_i, y_i)$$

G is the score of the right child, *H* is the score of the left child and*Gain* is the score in the case no new child [7].

To reduce the implementation time, a team from Microsoft developed the light gradient boosting machine (LightGBM) in April 2017 [8]. The main difference is that the decision trees in LightGBM are grown leaf-wise, instead of checking all of the previous leaves for each new leaf, as shown in Figs. 1 and 2. All the attributes are sorted and grouped as bins. This implementation is called histogram implementation. LightGBM has several advantages such as better accuracy, faster training speed, and is capable of large-scale handling data and is GPU learning supported.

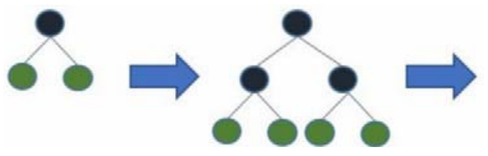


Fig. 1 XGBoost Level-wise tree growth

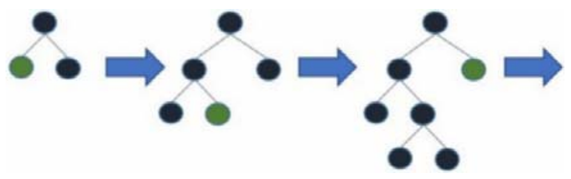


Fig. 2 LightGBM Leaf-wise tree growth

CatBoost (for "categorical boosting") focuses on categorical columns using permutation techniques, one_hot_max_size (OHMS), and target-based statistics. CatBoost solves the exponential growth of the features combination by using the greedy method at each new split of the current tree. For each feature that has more categories than OHMS (an input parameter), CatBoost uses the following steps:

1. Dividing the records to subsets randomly,
2. Converting the labels to integer numbers, and
3. Transforming the categorical features to numerical, as:

$$avgTarget = \frac{countInClass + prior}{totalCount + 1}$$

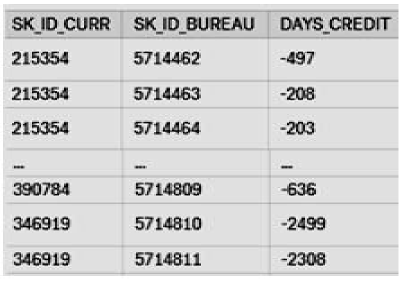
where, *countInClass* is the number of ones in the target for a given categorical feature, *totalCount*is the number of previous objects and *prior* is specified by the starting parameters [9]-[11].

III. Home Credit Dataset

The aim of the home credit dataset is to predict the capabilities of the clients repayment by using a variety of alternative data [1], [12]. Due to shortage or non-existent records of loan repayment, home credit attempts to expand the safe borrowing experience for the unbanked clients by collecting and extracting more information about the clients from different resources as follows:

1- Application_{train|test}.csv: Each row in this file is considered one loan, the file application_train.csv contains a target column, while application_test.csv does not contain a target column. The number of the clients in this file is 307511, and the number of the features is 123 such as: *SK_ID_CURR, NAME_CONTRACT_TYPE,CODE_GENDER, FLAG_OWN_CAR, FLAG_OWN, CNT_CHILDREN, AMT_INCOME, AMT_CREDIT, AMT_ANNUITY, TARGET,* etc. The target variable defines whether the loan was repaid or not.

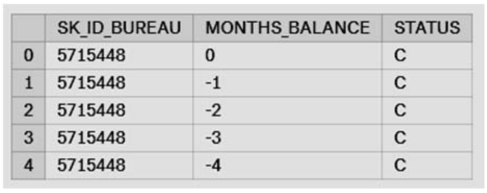
2- Bureau.csv: The previous applications about each client from other financial institutions, a client could have several applications, thus the number of the records in this file more than the number of the clients. This file has 1716428 rows and 17 features. Fig. 3 shows a snapshot of this data.



SK_ID_CURR	SK_ID_BUREAU	DAYS_CREDIT
215354	5714462	-497
215354	5714463	-208
215354	5714464	-203
--	--	--
390784	5714809	-636
346919	5714810	-2499
346919	5714811	-2308

Fig. 3 Snapshot of Bureau data

3- Bureau_balance.csv: The balance of each month for every previous credit. This file has 27299925 rows and three features. Fig. 4 shows a snapshot of this data.



	SK_ID_BUREAU	MONTHS_BALANCE	STATUS
0	5715448	0	C
1	5715448	-1	C
2	5715448	-2	C
3	5715448	-3	C
4	5715448	-4	C

Fig. 4 Snapshot of Bureau balance data

4- POS_CASH_balance.csv: The snapshots of monthly balance for every previous point of sales (POS). This file has 10001358 rows and eight features.

5- Credit_card_balance.csv: The snapshots of monthly balance for every credit with home credit. This file has 3840312 rows and 23 features.

6- Previous_application.csv: Each row in this file represents a previous application related to client loans. This file has 1670214 rows and 37 features.

7- Installments_payments.csv: The history of the previous repayments in home credit, where some rows outline missed installments, and other rows describe payments made. This file has 13605401 rows and eight features.

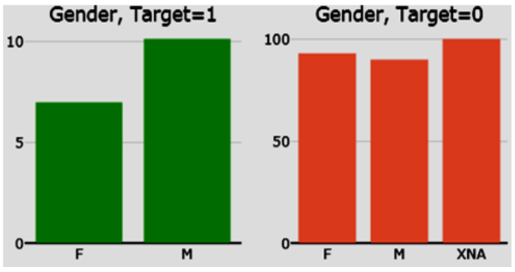
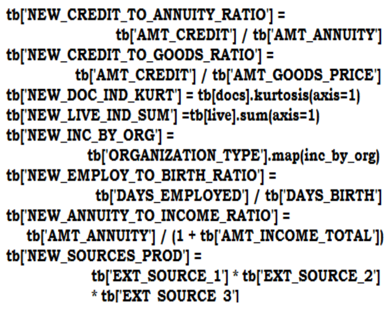


Fig. 5 Gender differences in repaying the loan



```
tb['NEW_CREDIT_TO_ANNUITY_RATIO'] =
                    tb['AMT_CREDIT'] / tb['AMT_ANNUITY']
tb['NEW_CREDIT_TO_GOODS_RATIO'] =
             tb['AMT_CREDIT'] / tb['AMT_GOODS_PRICE']
tb['NEW_DOC_IND_KURT'] = tb[docs].kurtosis(axis=1)
tb['NEW_LIVE_IND_SUM'] =tb[live].sum(axis=1)
tb['NEW_INC_BY_ORG'] =
               tb['ORGANIZATION_TYPE'].map(inc_by_org)
tb['NEW_EMPLOY_TO_BIRTH_RATIO'] =
                tb['DAYS_EMPLOYED'] / tb['DAYS_BIRTH']
tb['NEW_ANNUITY_TO_INCOME_RATIO'] =
   tb['AMT_ANNUITY'] / (1 + tb['AMT_INCOME_TOTAL'])
tb['NEW_SOURCES_PROD'] =
                tb['EXT_SOURCE_1'] * tb['EXT_SOURCE_2']
                * tb['EXT SOURCE 3']
```

Fig. 6 Snapshot of feature generation using python

When the home credit dataset is explored, we can note that the target label is an imbalanced, where the target column in the most of the records have the value 0 (about 91%), which means that the client did his installments successfully, and 24000 applicants (about 9%) had difficulties in repaying the loan. Another important observation can be exploited is that males, more than the females, are more prone to failure to repay the loan or make installments successfully, as shown in Fig. 5.

TABLE I
THE NUMBER OF FEATURES BEFORE AND AFTER FEATURE GENERATION

File	#Records	#features	#Feature after generation
Application	356251	123	240
Bureau	1716428	17	80
Bureau_balance	27299925	3	15
Pos-cash	10001358	8	18
Credit card balance	3840312	23	113
Previous applications	1670214	37	219
Installments payments	13605401	8	36
Total		219	721

More features can be generated by using the domain knowledge and aggregations, as shown in Fig. 6. Table I summarizes the number of features before and after feature generation.

IV. EXPERIMENTAL RESULTS

To compare between the gradient methods, the home credit dataset is used and tested by implementing XGBoost, LightGBM and CatBoost. The number of rows is reduced by deleting any row with missing values more than 75% or has a low importance rank. Five-fold validation is applied on a variant number of rows. Tables II-IV show that LightGBM has the best area under the curve (AUC) and the fastest training time, while XGBoost has the worst training time, and CatBoost has the worst AUC. However, these results cannot be generalized to other datasets. For example, if the dataset has more categorical features, we expect CatBoost will outperform the other methods; the implementation time seems to be more independent and has low correlation with the features type.

TABLE II
TIME AND AUC USING XGBOOST

#Rows	AUC	Time
307507	0.788320	4306
250000	0.784516	3550
200000	0.781219	2892
150000	0.773347	2098
100000	0.772771	1219
50000	0.768899	9487

TABLE III
TIME AND AUC USING LIGHTGBM

#Rows	AUC	Time
307507	0.789996	786
250000	0.788589	638
200000	0.786344	512
150000	0.786215	393
100000	0.782477	263
50000	0.777649	121

TABLE IV
TIME AND AUC USING CATBOOST

#Rows	AUC	Time
307507	0.787629	1803
250000	0.784402	1257
200000	0.782895	851
150000	0.780762	567
100000	0.776168	442
50000	0.770666	286

Table V illustrates the effect of the features preprocessing on the time and AUC. From the table, it can be noted that normalization, collinear or deleting the features which have missing values less than 75%,is unfeasible. Figs. 7 and 8 show the features ranking using LightGBM and CatBoost, respectively.

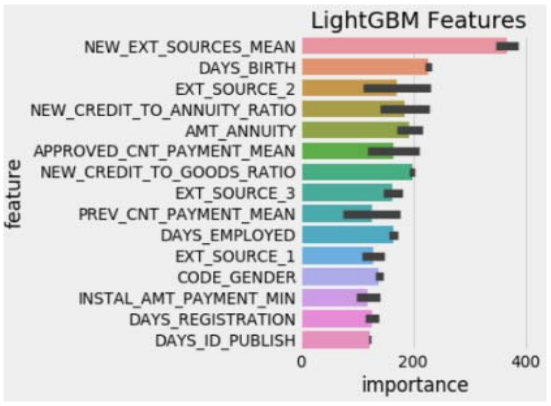


Fig. 7 Feature ranking using LightGBM

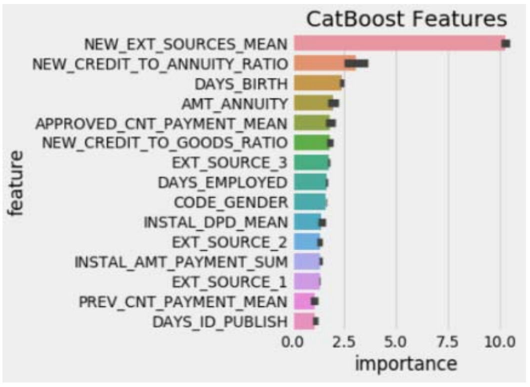


Fig. 8 Feature ranking using CatBoost

Fig. 9 shows the distribution of a high rank feature (EXT_SOURCE1) and a low rank feature (BURO_DAYS).

TABLE V
THE EFFECT OF THE FEATURES PREPROCESSING ON LIGHTGBM PERFORMANCE

	# Features	AUC	Time
Full data	721	0.789804	1748
Miss 75%	696	0.789933	1685
Miss 75% normalization	696	0.789868	1716
Miss 80 Importance 1	392	0.790115	1437
Miss 75 Importance 5	200	0.789996	786
Miss 75 Importance 7	158	0.789897	645
Miss 75 Importance 10 Collinear 95	113	0.788780	515
Miss 50 Importance 7 Collinear 95	105	0.779643	432
Miss 50 Importance 7	122	0.77310	533

Discovering new features can enhance the accuracy significantly; however, the knowledge of the domain is not sufficient to find all the important features. Thus, a random features generation mechanism is adopted using random operations (*, ^, /, +, - , max, …) with two or three of the top features. To prevent the exponential growth of the random features, a simple and fast rejection technique is used such as a signal to noise feature ranking. By using the combination of the above operations, thousands of the new features are generated. However, only 150 features are found which have acceptable rank; therefore, the AUC is improved after adding the new discovered features, and became 0.79304. Fig. 10 shows a new random feature (b1n11) among the top features using LightGBM ranking.

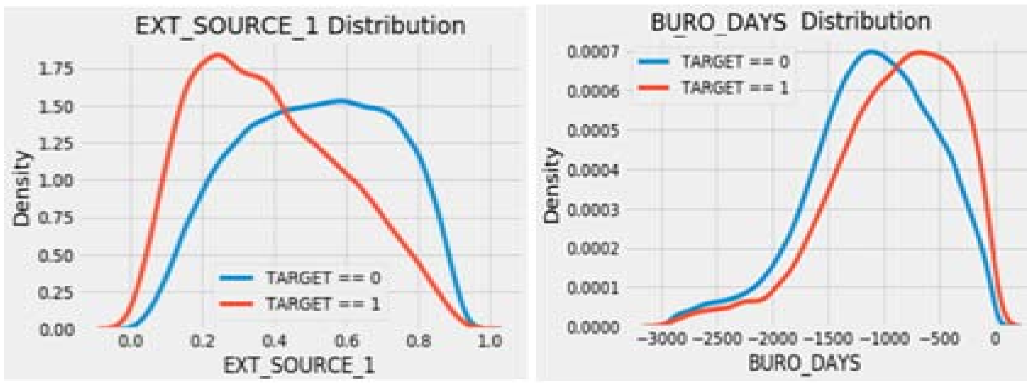


Fig. 9 The distribution of low and high rank features

V. CONCLUSION

Boosting methods iteratively train a set of weak learners, where the weight of the records are updated according to the regression results of the loss function of the previous learners. In this study, we compared between three state-of-the-art gradient boosting methods (XGBoost, CatBoost and LightGBM) in terms of CPU runtime and accuracy. LightGBM seems to be significantly faster than the other gradient boosting methods and more accurate using the same time budget of hyper-parameters optimization. The results can be improved by generating new features and selecting the best set.

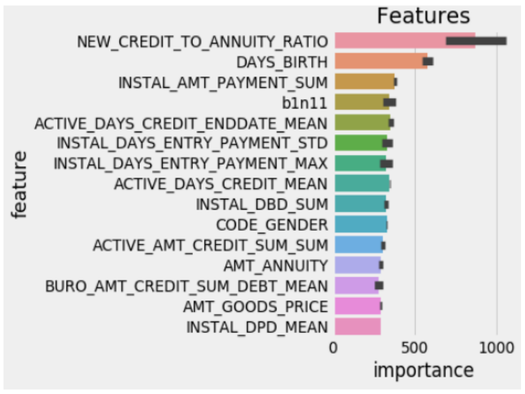


Fig. 10 New features ranking using LightGBM

REFERENCES

[1] Y. Freund, R. E. Schapire, “A decision-theoretic generalizationof online learning and an application to boosting," *Journal of Computer andSystem Sciences*, vol. 55, no. 1, pp 119-139, 1997.

[2] P. Kontschieder, M. Fiterau, A. Criminisi, S. Rota Bulo. “Deep neural decision forests,” *In Proceedings of the IEEE International Conference on Computer Vision,* pp 1467–1475, 2015.

[3] J. C. Wang, T. Hastie, “Boosted varying-coefficient regression models for product demand prediction,” *Journal of Computational and Graphical Statistics,* vol. 23, no. 2, pp 361–382, 2014.

[4] E Al Daoud, “Intrusion Detection Using a New Particle Swarm Method and Support Vector Machines,” *World Academy of Science, Engineering and Technology,* vol. 77, 59-62, 2013.

[5] E. Al Daoud, H Turabieh, “New empirical nonparametric kernels for support vector machine classification,” *Applied Soft Computing,* vol. 13, no. 4, 1759-1765, 2013.

[6] E. Al Daoud, "An Efficient Algorithm for Finding a Fuzzy Rough Set Reduct Using an Improved Harmony Search," *I.J. Modern Education and Computer Science*, vol. 7, no. 2, pp16-23, 2015.

[7] Y. Zhang, A. Haghani. “A gradient boosting method to improve travel time prediction. Transportation Research Part C,” *Emerging Technologies*, vol. 58, 308–324, 2015.

[8] K. Guolin, M. Qi, F. Thomas, W. Taifeng, C. Wei, M. Weidong, Y. Qiwei, L. Tie-Yan, "LightGBM: A Highly Efficient Gradient Boosting Decision Tree," *Advances in Neural Information Processing Systems* vol. 30, pp. 3149-3157, 2017.

[9] A. Dorogush, V. Ershov, A. Gulin "CatBoost: gradient boosting with categorical features support," *NIPS*, p1-7, 2017.

[10] M. Qi, K. Guolin, W. Taifeng, C. Wei, Y. Qiwei, M. Weidong, L. Tie-Yan, "A Communication-Efficient Parallel Algorithm for Decision Tree," *Advances in Neural Information Processing Systems,* vol. 29, pp. 1279-1287, 2016.

[11] A. Klein, S. Falkner, S. Bartels, P. Hennig, F. Hutter, “Fast bayesian optimization of machine learning hyperparameters on large datasets,” *In Proceedings of Machine Learning Research PMLR*, vol. 54, pp 528-536,2017.

[12] J. H. Aboobyda, and M. A. Tarig, “Developing Prediction Model Of Loan Risk In Banks Using Data Mining,” *Machine Learning and Applications: An International Journal (MLAIJ)*, vol. 3, no. 1, pp 1–9, 2016.