Evaluating categorical encoding methods on a real credit card fraud detection database

François de la Bourdonnaye, Fabrice Daniel

Artificial Intelligence Department of Lusis, Paris, France
http://www.lusisai.com

December 2021

ABSTRACT

Correctly dealing with categorical data in a supervised learning context is still a major issue. Furthermore, though some machine learning methods embody builtin methods to deal with categorical features, it is unclear whether they bring some improvements and how do they compare with usual categorical encoding methods. In this paper, we describe several well-known categorical encoding methods that are based on target statistics and weight of evidence. We apply them on a large and real credit card fraud detection database. Then, we train the encoded databases using state-of-the-art gradient boosting methods and evaluate their performances. We show that categorical encoding methods generally bring substantial improvements with respect to the absence of encoding. The contribution of this work is twofold: (1) we compare many state-of-the-art "lite" categorical encoding methods on a large scale database and (2) we use a real credit card fraud detection database.

Keywords: categorical encoding, gradient boosting, fraud detection

1 INTRODUCTION

Credit card frauds happen everyday and represent a major cost for banks. Consequently, it becomes beneficial for them to be able to automatically detect a large amount of these frauds and block them before having to reimburse clients. A good hint for this is the use of supervised learning techniques which have known major progresses over the past years [11], [8], [4]. Despite these hopes, several issues related to the concept drift, explainability or imbalanced dataset are still unsatisfactory addressed. In this paper, we are interested in the very specific problem of encoding categorical variables for fraud detection problems. Indeed, lot of input features of fraud detection problems are categorical-typed, e.g. the merchant category, the country in which the transaction takes place, the type of card, ...

Handling categorical data is not an easy task because we cannot naively input categorical data in machine learning pipelines without processing and hope that black boxes will do the trick in an optimal way. Several ways exist among which using target statistics, using polynomial coding, sum coding, one-hot encoding... In this paper, we mainly focus on methods using target statistics and weight of evidence because they transform categorical variables into a single numerical variable, which satisfies some size constraints: the fraud detection real dataset is huge, so the memory consumption of the encoded features should be as small as possible. In addition an important categorical feature like MCC (Merchant Category Code) has hundreds of unique values, making one-hot encoding irrelevant not only for a question of memory consumption but also for the resulting sparcity. The machine learning methods used in this study are based on gradient boosting. Three baseline methods are used: LightGBM, CatBoost and XGBoost. The choice of these techniques might be surprising given the fact literature often states that algorithms based on decision trees do not need a specific encoding for categories because of the "split" aspect, i.e. specific categories can be discriminated. We claim that this assumption may be inaccurate and will conduct experiments on real data to empirically show that models based on decision trees benefit from categorical encoding methods.

The studied encoding methods will be presented and compared against each other to check which method yields best performances. Our work contributes in that such categorical encoding methods have not been compared against each other on a huge real fraud detection database to our knowledge. Furthermore, these methods will be tested also against builtin methods when they exist so that we can conclude whether encoding categorical data with external methods is superior to builtin encoding. To finish with, the encoding methods are tested against the absence of encodings to ensure that metrics are better using categorical encoding and that our assumption of the usefulness of categorical encodings for gradient boosting trees is correct.

The remainder of the paper is organized as follows. First, we present theoretical knowledge that includes a presentation of gradient boosting, categorical encoding methods using target statistics, weight of evidence and builtin support for gradient-boosting based methods. Second, we present the experiments that are conducted. And finally, we will conclude the paper.

2 BACKGROUND

2.1 Gradient-boosting

2.1.1 Main idea

Boosting algorithms combine weak learners (generally decision trees) in an iterative fashion. Gradient boosting algorithms have a specific scheme where a new learner is optimized such that it fits residual of the previous learner.

Let's describe it in mathematical terms. We use the following notations:

- $F_i (i \in \{1, ..., m\})$ is a learner
- $D_{\text{train}} = \{(x_1, y_1), ..., (x_i, y_i), ..., (x_{n_{train}}, y_{n_{train}})\} = (X, Y)$ a training dataset.

At the first iteration, F_1 is fitted as a normal decision tree, i.e. on y. For the following iterations $i \in \{2, ..., m\}$, F_i is fitted on $y - F_{i-1}$. Thus, the rationale is to fit estimators on the residual error of the previous estimator.

The name "gradient boosting" comes from the fact the residual is (up to a factor) the gradient of the mean squared error between the target and the output of the learner (see Equation 1 and 2).

$$L_{\text{MSE}} = \frac{1}{n_{train}} (y - F_m(x))^2 \tag{1}$$

$$\frac{\delta L_{\text{MSE}}}{\delta F_m(x)} = \frac{2}{n_{train}} (y - F_m(x)) \tag{2}$$

2.1.2 XGBoost

XGBoost [3] is one of the most famous gradient boosting libraries. Its keys innovations are based on a novel tree learning algorithm that efficiently handles sparse data, "a theoretically justified weighted quantile sketch procedure that enables handling instance weights in approximate tree learning", the efficient use of ressources for parallel computing, and "an effective cache-aware block structure or out-of-core tree learning".

2.1.3 LightGBM

The major difficulty of most gradient boosting algorithms is their scalability when the number of features increases. LightGBM [10] exhibits two techniques that are derived to overcome this difficulty:

- Gradient-based one side sampling (GOSS): this procedure cancels data that have no gradients.
- Exclusive feature bundling: this method bundles mutually exclusive features, i.e. features that never take 0 simultaneously.

Note that we won't use the GOSS feature for our experiments. LGBM exhibits also a builtin method to process categorical features. Categories are sorted with respect to the objective. Then, they use [9] to find an optimal split according to accumulated gradients (over Hessians) in reasonable time. See below for more details.

2.1.4 CatBoost

Catboost [6], [5] belongs to the class of gradient boosting algorithms and is specially suitable to input spaces that contain categorical features. Two innovations are implemented:

- ordered boosting
- algorithm that processes categorical features

The idea of ordered boosting is to compute gradients or residuals for a given sample of the training set based on a set of samples that does not include this particular training sample. Otherwise, the gradient appears to be biased. In practice, CatBoost establishes several permutations of the training set that are used for diverse training iterations. Residuals are computed based on these permutations, and models as well.

For processing categorical features, they take inspiration from target statistics (this converts categorical features into numerical values based on target averages, see below for more details). They proved that this technique is biased and came up with a solution based on ordered boosting (the solution is called ordered TS). Categorical features can then have different values according to the training iteration.

CatBoost is also innovative in a software view in the sense, it is said to be very efficient for training and inference both in CPU and GPU.

2.2 Builtin methods

2.2.1 LGBM

LGBMs have a specific way of dealing with categorical variables. At each split, categorical variables are sorted according to the training objective instead of categorical values (accumulated sum of gradient over accumulated sum of hessians for each categorical feature). The optimal split is made using [7].

2.2.2 CatBoost

The builtin method is derived from the CatBoost method presented below in Section 2.3.4.

2.2.3 XGBoost

There is no specific way to handle categorical variables using XGBoost. It is naturally achieved by splitting categories according to categorical values.

2.3 Encoding categorical variables using target statistics

2.3.1 General equation of target statistics

We assume that X is a categorical variable which takes I different categories and that Y is a binary target in a supervised learning context. We use the following notations:

- n_{iY} is the number of times $X = i$ when $Y = 1$
- n_i is the number of times $X = i$
- n_Y is the number of times $Y = 1$
- n_{TR} is the number of samples in the training set

Then, the prior probability of fraud (without taking categories into consideration) is given by:

$$p_{prior} = \frac{n_Y}{n_{TR}}. \tag{3}$$

and the probability of fraud given categories is given by:

$$p_{fraud,i} = \frac{n_{iY}}{n_i}. \tag{4}$$

Then, the equation that allows to compute S_i the encoded value for the category i can be computed as a weighted sum of p_{prior} and $p_{fraud,i}$ (taken from [12]):

$$S_i = \lambda(n_i)\, p_{fraud,i} + (1 - \lambda(n_i))\, p_{prior} = \lambda(n_i)\frac{n_{iY}}{n_i} + (1 - \lambda(n_i))\frac{n_Y}{n_{TR}}. \tag{5}$$

Various encoding methods are derived from this equation and they mainly differ in two aspects:

- the way $\lambda(n_i)$ is computed
- the way $p_{fraud,i}$ is computed

2.3.2 Target encoder

$\lambda(n_i)$ is computed this way (according to [12]):

$$\lambda(n_i) = \frac{1}{1 + e^{-\frac{(n_i - k)}{f}}}, \tag{6}$$

where k and f are two tunable parameters. k is half the minimal sample size for which we trust the estimate of $p_{fraud,i}$ and f is a smoothing parameter. In the target encoder package, the two default values are 1 and 1. The idea of this formula is simply to trust more $p_{fraud,i}$ when n_i is high with respect to the k hyper-parameter. In the following, the Target encoder is also called Barecca encoder from the name of one of its authors.

2.3.3 M-estimate

$\lambda(n_i)$ is computed this way (according to [12]):

$$\lambda(n_i) = \frac{n_i}{m + n_i}, \tag{7}$$

which gives the following formula for S_i:

$$S_i = \frac{n_i \times p_{fraud,i} + p_{prior}m}{n_i + m} = \frac{n_{iY} + p_{prior}m}{n_i + m}, \tag{8}$$

where m is a tunable parameter.

The rationale of M-estimate is the same as for the target encoder. When $n_i >> m$ then $\lambda(n_i)$ tends to 1 and the estimate of $p_{fraud,i}$ becomes more important.

2.3.4 CatBoost encoding

For CatBoost, several ideas are mixed. First, the basic formula is given by M-estimate:

$$S_i = \frac{n_{iY} + p_{prior}m}{n_i + m}. \tag{9}$$

CatBoost developers identified that using the sample n to compute $S_{i,n}$ leads to target shift. To avoid using the sample n, they use permutations of training sets and take systematically all the relevant elements of the permuted dataset before the new position of sample n to compute n_i or n_{iY}.

2.3.5 Pozzolo

We take inspiration from [13] which designs a categorical encoding method specially designed for the problem of credit card fraud detection. $\lambda\left(n_i\right)$ is computed this way:

Let α_{n_i} defined as:

$$\alpha_{n_i} = \frac{n_i}{n_{TR}}. \tag{10}$$

Then, we define two ways of computing $\lambda\left(n_i\right)$:

$$\lambda 1(n_i) = \frac{\alpha_{n_i} - \min_{i \in I}(\alpha_{n_i})}{\max_{i \in I}(\alpha_{n_i}) - \min_{i \in I}(\alpha_{n_i})}, \tag{11}$$

and

$$\lambda 2(n_i) = \frac{log(\alpha_{n_i} - \min_{i \in I}(\alpha_{n_i}))}{log(\max_{i \in I}(\alpha_{n_i}) - \min_{i \in I}(\alpha_{n_i}))}. \tag{12}$$

The log computation is made such that the coefficients take into account the fact that the values of categorical variables are not uniformly distributed over the categories, which is often the case for categorical variables in fraud detection data.

For instance, let's say that X has three categories 'A', 'B' and 'C'. 'A' is called 99 % of the samples, and 'B' and 'C' 0.5 % of the samples. Then, using $\lambda 1(n_i)$ will yield very large values for 'A' and very small ones for 'B' and 'C'. The log computation used for $\lambda 2(n_i)$ makes these differences smaller.

2.3.6 James-Stein

This encoder directly comes from the Stein's paradox [9] (the methods based on target statistics have a connection with the paradox) which tells that combined estimators are more accurate on average with respect to a classical Monte-Carlo estimator.

The classical James Stein approach gives the following formula for $1 - \lambda\left(n_i\right)$:

$$1 - \lambda\left(n_i\right) = \frac{(k-3)var(p_{fraud,i})}{\sum\left(x_i - p_{prior}\right)^2}, \tag{13}$$

where k is the number of groups.

$(k-3)$ has been replaced by $k-1$ and we found the following formula:

$$1 - \lambda\left(n_i\right) = \frac{(k-1)var(p_{fraud,i})}{\sum\left(x_i - p_{prior}\right)^2} = \frac{var(p_{fraud,i})}{\frac{\sum(x_i - p_{prior})^2}{k-1}} = \frac{var(p_{fraud,i})}{var(p_{fraud,i}) + var(p_{prior})}. \tag{14}$$

Then, $\lambda\left(n_i\right)$ is computed this way:

$$\lambda\left(n_i\right) = \frac{var(p_{prior})}{var(p_{fraud,i}) + var(p_{prior})}, \tag{15}$$

where λ depends on n_i because $var(p_{fraud,i})$ depends on n_i. Indeed, in practice, squared standard errors are used to compute these variances. Here, in order to compute λ, we take into account variances instead of only the number of samples taking i as a category.

2.4 Encoding categorical variables using weight-of-evidence

This encoding method does not depend on the equation of target statistics and comes from the credit scoring world where log ratio of percentages of bad customers and percentages of good customers [14] are computed.

Let's define first weight of evidence as a log ratio between percentages of non-events (non-fraud) and events (fraud):

$$woe = \log\left(\frac{\% of nonfrauds}{\% of frauds}\right). \tag{16}$$

We apply this definition for each category i and S_i is computed:

$$S_i = \log\left(\frac{\frac{n_{i\bar{Y}}}{n_{\bar{Y}}}}{\frac{n_{iY}}{n_Y}}\right). \tag{17}$$

In a nutshell, this encoding method groups categories whose ratio between percentages of non-frauds and frauds is similar.

In practice, the real implemented formula is:

$$S_i = \log\left(\frac{\frac{n_{i\bar{Y}}+\gamma}{n_{\bar{Y}}+2\gamma}}{\frac{n_{iY}+\gamma}{n_Y+2\gamma}}\right), \tag{18}$$

where γ is a regularization parameter.

2.5 Toy example

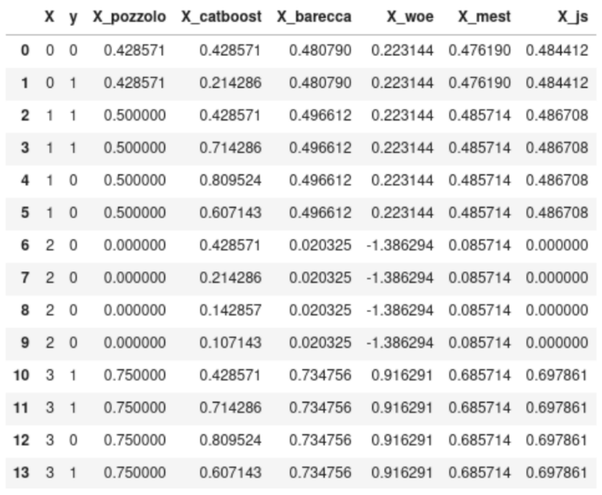
Figure 1 shows a toy example of encodings of categorical variables. The columns X and y represent the toy inputs and outputs respectively, and the other columns stand for encodings of Pozzolo, CatBoost, Barecca (Target encoder), Weight of evidence, M-estimate and James-Stein.

3 EXPERIMENTS

3.1 Settings

3.1.1 Datasets

We use real data from a major French bank. More precisely, we use a subset of 10 millions of transactions using raw



	X	y	X_pozzolo	X_catboost	X_barecca	X_woe	X_mest	X_js
0	0	0	0.428571	0.428571	0.480790	0.223144	0.476190	0.484412
1	0	1	0.428571	0.214286	0.480790	0.223144	0.476190	0.484412
2	1	1	0.500000	0.428571	0.496612	0.223144	0.485714	0.486708
3	1	1	0.500000	0.714286	0.496612	0.223144	0.485714	0.486708
4	1	0	0.500000	0.809524	0.496612	0.223144	0.485714	0.486708
5	1	0	0.500000	0.607143	0.496612	0.223144	0.485714	0.486708
6	2	0	0.000000	0.428571	0.020325	-1.386294	0.085714	0.000000
7	2	0	0.000000	0.214286	0.020325	-1.386294	0.085714	0.000000
8	2	0	0.000000	0.142857	0.020325	-1.386294	0.085714	0.000000
9	2	0	0.000000	0.107143	0.020325	-1.386294	0.085714	0.000000
10	3	1	0.750000	0.428571	0.734756	0.916291	0.685714	0.697861
11	3	1	0.750000	0.714286	0.734756	0.916291	0.685714	0.697861
12	3	0	0.750000	0.809524	0.734756	0.916291	0.685714	0.697861
13	3	1	0.750000	0.607143	0.734756	0.916291	0.685714	0.697861

Figure 1: Toy example for categorical feature computation

and derivative features. The raw features consist of 8 categorical features and 2 numerical features. The derivative features consist of a small subset of 20 numerical features among more than 200 computed using count, mean and diff aggregations.

3.1.2 Methodology

The datasets are splitted into training and validation sets according to a $\frac{2}{3}/\frac{1}{3}$ policy. For each gradient boosting model (LGBM, XGBoost, CatBoost), we test the encoding methods that are mentioned above. Furthermore, hyperparameters of these gradient boosting models have been optimized using the following approach:

The optimization happened using 3-fold cross validation on the training set (the optimization criterion was the PR AUC) using the Optuna package [1] and the Tree-Parzen-Estimator as the sampling strategy [2]. For this optimization, we did not use categorical encoding.

For each method, we apply encoding computed using training data only. The methods are evaluated using various metrics:

- Area under Precision-recall curve (PR AUC)
- Precision
- Recall
- F1

As ranks between methods are not stable over the seeds given as input to the boosting models, we average each setting over 10 seeds. We did not use the accuracy metric since it does not make much sense in the context of fraud detection database which is highly imbalanced.

The gradient boosting hyperparameters have been found only for the case of the absence of encoding. Then, we willingly favor the absence of encoding compared with the use of categorical encoding.

3.2 Results

We display the results metrics by using a table gathering average results of each method. More precisely, we only display percentage of variations with respect to the absence of encoding. We willingly omit raw values for confidentiality purposes.

3.2.1 LGBM

	PR AUC	Precision	Recall	F1
Pozzolo	+9%	-1%	+59%	+53%
M-estimate	+6%	-7%	+61%	+53%
Builtin	**+11%**	+0%	+36%	+32%
James Stein	+5%	-4%	**+64%**	**+56%**
Barecca	+9%	+0%	+57%	+50%
Weight of evidence	+8%	+1%	+56%	+50%
CatBoost	+5%	**+3%**	+41%	+37%

Table 1: Results for LGBM

For LGBM, we establish several observations:

- We observe that all the encoding methods improve PR AUC, RECALL and F1.
- We observe that only CatBoost and Weight of Evidence improve the Precision metric.
- The builtin method gives very interesting results without implementation effort due to the encoding.

3.2.2 CatBoost

	PR AUC	Precision	Recall	F1
Barecca	+27%	-8%	+106%	+96%
Builtin	**+34%**	-9%	**+133%**	**+119%**
Pozzolo	+26%	-8%	+91%	+82%
James-Stein	+24%	-11%	+114%	+102%
Weight of evidence	+22%	**-4%**	+102%	+92 %
M-estimate	+22%	-11%	+95%	+86%

Table 2: Results for CatBoost

For CatBoost, we establish several observations:

- Precision is slightly deteriorated by all the methods though all the other metrics are improved.
- The builtin Catboost method is the best one according to all the metrics (with the exception of Precision) and should be chosen in the future since it can be applied in an effortless way.

3.2.3 XGBoost

For XGBoost, we establish several observations:

- The CatBoost encoder method is the only one to improve all the metrics, precision included.
- Another interesting method is the target encoder (Barecca).

To summarize, all these results show us on the one hand that gradient boosting algorithms applied on real fraud detection problems benefit from categorical encoding. On the other hand, we notice that the CatBoost encoder seems to be a very good categorical encoding method for all the tested gradient boosting models.

	PR AUC	Precision	Recall	F1
M-estimate	+5%	-5%	+49%	+44%
CatBoost	**+6%**	**+4%**	+36%	+29%
Pozzolo	+4%	-5%	+40%	+36%
James Stein	0%	-9%	**+53%**	**+47%**
Barecca	+6%	-4%	+46%	+41%
Weight of evidence	+1%	-9%	+36%	+32%

Table 3: Results for XGBoost

4 CONCLUSION

We have tested several categorical encoding approaches for different gradient boosting methods. First, we have observed that it is always interesting and beneficial to handle categorical variables in a specific way as metrics are all improved with the exception of precision in some cases. Then, we back up the claim that gradient boosting algorithms also benefit from the encoding of categorical variables. Second, we remind the reader of the fact that these results are only valid for the specific database, for the specific categorical encoding parameters and for the gradient-boosting methods along with its parameters.

Nevertheless, we observe that for LGBM, Catboost and Weight-of-evidence methods yield best performances. The builtin method is also interesting and can be applied effortlessly. For CatBoost, the builtin method is by far the best one. For XGBoost, CatBoost and Barecca encoding methods seem to be the best ones. Consequently, we claim that the CatBoost method seems to be the most promising one for encoding categorical variables in our real fraud detection problem.

For future work, we wish to consolidate the results using different datasets and categorical encoding hyper-parameter optimization.

REFERENCES

[1] T. Akiba et al. "Optuna: A Next-generation Hyperparameter Optimization Framework". In: *Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2019.

[2] J. Bergstra et al. "Algorithms for Hyper-Parameter Optimization". In: *Advances in Neural Information Processing Systems*. Ed. by J. Shawe-Taylor et al. Vol. 24. Curran Associates, Inc., 2011. URL: `https://proceedings.neurips.cc/paper/2011/file/86e8f7ab32cfd12577bc2619bc635690-Paper.pdf`.

[3] T. Chen and C. Guestrin. "XGBoost". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Aug. 2016). DOI: `10.1145/2939672.2939785`. URL: `http://dx.doi.org/10.1145/2939672.2939785`.

[4] A. Dal Pozzolo et al. "Credit Card Fraud Detection: A Realistic Modeling and a Novel Learning Strategy". In: *IEEE Transactions on Neural Networks and Learning Systems* PP (Sept. 2017), pp. 1–14. DOI: `10.1109/TNNLS.2017.2736643`.

[5] A. V. Dorogush, V. Ershov, and A. Gulin. "CatBoost: gradient boosting with categorical features support". In: *CoRR* abs/1810.11363 (2018). arXiv: `1810.11363`. URL: `http://arxiv.org/abs/1810.11363`.

[6] A. V. Dorogush et al. "CatBoost: unbiased boosting with categorical features". In: *CoRR* abs/1706.09516 (2017). arXiv: `1706.09516`. URL: `http://arxiv.org/abs/1706.09516`.

[7] W. D. Fisher. "On Grouping for Maximum Homogeneity". In: *Journal of the American Statistical Association* 53.284 (1958), pp. 789–798. DOI: `10.1080/01621459.1958.10501479`. eprint: `https://www.tandfonline.com/doi/pdf/10.1080/01621459.1958.10501479`. URL: `https://www.tandfonline.com/doi/abs/10.1080/01621459.1958.10501479`.

[8] J. Frery et al. "Efficient Top Rank Optimization with Gradient Boosting for Supervised Anomaly Detection". In: *Machine Learning and Knowledge Discovery in Databases*. Ed. by M. Ceci et al. Cham: Springer International Publishing, 2017, pp. 20–35. ISBN: 978-3-319-71249-9.

[9] W. James and C. Stein. "Estimation with Quadratic Loss". In: *Breakthroughs in Statistics: Foundations and Basic Theory*. Ed. by S. Kotz and N. L. Johnson. New York, NY: Springer New York, 1992, pp. 443–460. ISBN: 978-1-4612-0919-5. DOI: `10.1007/978-1-4612-0919-5_30`. URL: `https://doi.org/10.1007/978-1-4612-0919-5_30`.

[10] G. Ke et al. "LightGBM: A Highly Efficient Gradient Boosting Decision Tree". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS'17. Long Beach, California, USA: Curran Associates Inc., 2017, pp. 3149–3157. ISBN: 9781510860964.

[11] Y. Lucas and J. Jurgovsky. *Credit card fraud detection using machine learning: A survey*. 2020. arXiv: `2010.06479 [cs.LG]`.

[12] D. Micci-Barreca. "A Preprocessing Scheme for High-Cardinality Categorical Attributes in Classification and Prediction Problems". In: *SIGKDD Explor. Newsl.* (2001).

[13] A. D. Pozzolo and G. Bontempi. "Adaptive Machine Learning for Credit Card Fraud Detection". In: 2015.

[14] G. Zeng. "A Necessary Condition for a Good Binning Algorithm in Credit Scoring". In: *Applied Mathematical Sciences* Vol. 8 (July 2014), pp. 3229–3242. DOI: `10.12988/ams.2014.44300`.