*International Journal of Combinatorial Optimization Problems and Informatics, 15(2), May-Aug 2024, 160-172. ISSN: 2007-1558.*
*https://doi.org/10.61467/2007.1558.2024.v15i2.456*

www.editada.org

# Statistical Evaluation of Categorical Encoders for Pattern Preservation in Machine Learning Tasks

*Eric Valdez-Valenzuela[1], Angel Kuri-Morales[2] and Helena Gomez-Adorno[3]*
[1] Posgrado en Ciencia e Ingeniería de la Computación, Universidad Nacional Autónoma de México
[2] Instituto Tecnológico Autónomo de México
[3] Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas, Universidad Nacional Autónoma de México
Emails: ericvaldez@comunidad.unam.mx, akuri@itam.mx, helena.gomez@iimas.unam.mx

Abstract. Categorical attributes are prevalent in many datasets used for training Machine Learning models. However, most ML models are designed to handle only numerical inputs. Therefore, converting these categorical attributes into numerical values is necessary to utilize them effectively. During this conversion process, it is essential to preserve the underlying patterns. A loss of such information could adversely affect the performance of ML algorithms. Several encoding techniques have been developed to map categorical instances to numbers. This study evaluates commonly used encoders alongside CESAMO, a novel encoder designed to capture relationships between categorical attributes and other variables using what is referred to as Pattern Preserving Codes. We conducted a statistically supported assessment of these categorical encoders using synthetic data and compared the encoders' performance. The results show that CESAMO outperforms all other evaluated encoding techniques, confirming its ability to identify patterns in categorical data effectively.

Keywords. Categorical encoding, synthetic data, machine learning, data preprocessing

Article Info
*Received May 23, 2024*
*Accepted Jun 1, 2024*

## 1 Introduction

A significant number of machine learning (ML) algorithms exclusively process numerical attributes for analysis. There are, nonetheless, some exceptions. Tree-based model algorithms and Naive Bayes can theoretically manage both numeric and categorical data (Kuhn & Johnson, 2019). However, when these algorithms are implemented through commonly used libraries like Scikit-learn (Pedregosa et. al., 2011), and XGBoost (Chen & Guestrin, 2016), the necessity to transform categorical instances into numeric values arises anyway since these libraries exclusively accept numeric inputs.

Real-world datasets typically comprise a mix of both categorical and numerical attributes. To effectively employ these categorical variables in ML models, they must be processed through an encoding method, mapping all categorical instances into numerical values. It is crucial that this mapping retains the patterns in the data. Otherwise, there is a potential risk of information loss that can negatively affect the performance of ML algorithms. This implies that selecting an appropriate encoding technique affects the model's overall performance, as shown in Valdez et. al. (2021).

Transforming categorical attributes into numerical values is particularly crucial in clustering scenarios. Clustering problems, usually unsupervised, lack prior information about a dependent variable or target. Instead, clustering algorithms group unlabeled data into distinct clusters based on inherent patterns or similarities. Failing to preserve the underlying relationships in categorical attributes can lead to inaccurate outcomes. Hence, it is essential to employ encoders that effectively preserve the patterns.

Considering the importance of categorical encoders in ML tasks and their diverse impact on the performance of ML models, assessing the effectiveness of these encoders in such scenarios is essential. While studies have investigated the impact of encoding techniques on ML models, they often lack statistical rigor. These studies typically select a fixed number of datasets and draw conclusions based on them. To the best of our knowledge, there is a shortage of studies that extensively assess the performance of categorical encoders using statistical criteria to support the results.

In this study, our main goal is to conduct a quantifiable statistical assessment of the performance of encoder techniques and compare them. To achieve this, we propose the use of synthetic datasets, as supporting our findings requires evaluating the encoders on many datasets until reaching statistical significance. Moreover, the assessment includes a novel type of encoder that captures the relations between categorical attributes and other attributes in a dataset through Pattern Preserving Codes (PPCs). Our results reveal that the encoders mapping categorical instances to PPCs outperform other encoders.

The paper is structured as follows: Section 2 covers previous studies that have measured and compared encoders. Section 3 outlines the methodology used to evaluate the encoders' performance. Section 4 presents the experimental results. Finally, in Section 5, we summarize our conclusions.

## 2 Related Work

Multiple studies have measured and compared the performance of various encoding techniques. As follows are listed the most relevant and extensive ones that we found:

- Zhu et. al. (2024) explored how different encoders affect machine learning model performance with categorical variables. They categorize models into ATI, tree-based, and other models. The study evaluates 14 encoders, including one-hot and target, across 8 common machine-learning models on 28 datasets, showing that the one-hot encoder is optimal for ATI models and target encoders for tree-based models.
- Pargent et. al. (2019) investigated encoding strategies for high cardinality features in predictive modeling. Through a benchmark, the study compares 9 encoders with 5 machine learning algorithms and 27 datasets across regression, binary, and multiclass classification tasks. They found that regularized target encoding emerges as the most effective strategy across all algorithms, outperforming traditional approaches like label encoding.
- Matteucci et. al. (2024) presented a comprehensive benchmark of categorical encoders, addressing limitations in previous studies by evaluating 32 configurations of encoders, using 48 combinations of experimental factors, and testing on 50 datasets. Their findings revealed that Weight of Evidence performed the best for decision trees, while for logistic regression, the Sum, One-Hot, Binary, and Weight of Evidence consistently achieved higher ranks
- Seca & Mendes (2021) tested 16 encoding methods on 15 regression datasets using 7 predictive models. According to their results, the top-performing general-purpose encoders identified were Catboost, LeaveOneOut, and Target (all of these are supervised encoders).

While these works offer strong evidence indicating that some encoders' performance is better than others, their limitation lies in the lack of statistical support. In these studies, encoders are evaluated by subjectively selecting a predetermined number of real-world datasets, with the largest evaluation comprising 50 datasets. However, for more solid conclusions, evaluations should rely on statistical criteria.

Based on the previously mentioned disadvantage of the analyzed related works, arises the need for a method to evaluate and compare encoding techniques with statistical rigor. This work proposes a methodology supported by statistical analysis to objectively evaluate encoding techniques. The methodology is explained as follows.

## 3 Encoder Evaluation Methodology.

To evaluate and compare the categorical encoders, we aimed to determine their *Minimum Error* (ME), a statistical measure that quantifies the performance of any method. Each encoder is associated with an *Error Distribution* (ED), representing the population of all potential errors that may arise when applying the technique to problems. The ED is characterized by its mean $\mu_{ED}$ and $\sigma_{ED}$. The ME is defined as the value positioned at *k* standard deviations to the left of the mean within the error distribution:

$$ME = \mu_{ED} - k\, \sigma_{ED} \tag{1}$$

Given that the ME depends on knowing the $\mu_{ED}$ and $\sigma_{ED}$, it is required to determine first the ED of an encoding technique. To achieve this, we assess the encoders across several datasets until a statistical criterion is satisfied. Essentially, this criterion is based on obtaining errors until their mean normalizes. This will always occur as per the Central Limit Theorem. The subsequent sections delve into a more detailed explanation of the rationale behind this methodology, and the implementation of the activities to evaluate the encoders.

**Minimum Error Metric**

The ME is the metric we employed to statistically compare the encoders. This metric is based on Chebyshev's inequality, which states that for any data set, at least a certain proportion of values can be found at a specified *k* distance from the mean, regardless of the distribution shape, whether normal, bimodal, uniform or otherwise. Let *X* be a random variable with mean *μ* and standard deviation *σ*. The Chebyshev's inequality is stated as follows:

$$P(|X - \mu| \geq k\sigma) \leq \frac{1}{k^2} \tag{2}$$

Selecting $k = \sqrt{5}$ , the probability of *X* deviating from its mean by more than $\sqrt{5}\sigma$ standard deviations is ~20%. This is illustrated in Figure 1, where the red vertical dotted line denotes the values at $-\sqrt{5}\sigma$, and the green one at $\sqrt{5}\sigma$ . The shaded areas in the tails of the distributions represent ~20% of the data. The ~80% falls between these two vertical lines.

Let us assume that the distribution shown in Figure 1 represents the error distribution of an encoding technique. As we are looking for a statistical metric for comparing the encoders, our focus is on the error situated at -*k* standard deviations from the mean (the smaller the error, the better the encoding technique). We define the ME as the value located at $\mu - \sqrt{5}\sigma$ , which corresponds to the red vertical dotted line in Figure 1.

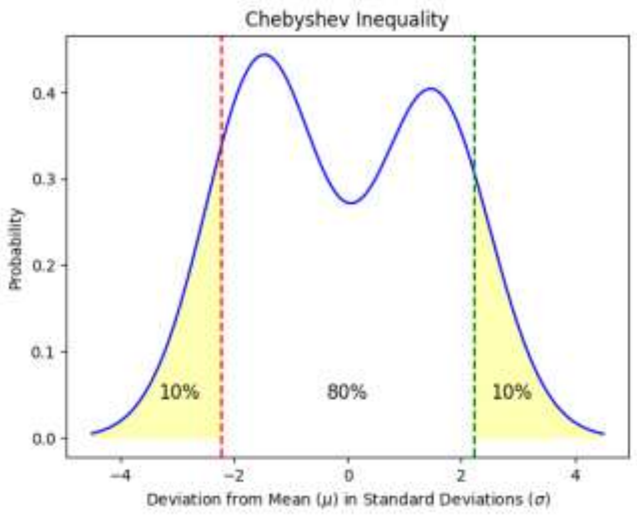

Figure 1. Chebyshev inequality selecting $k = \sqrt{5}$

The ME indicates that the probability of encountering an error smaller than this when using an encoding technique is less than ~10%, as the right side of the distribution comprises ~90% of the errors.

## Synthetic Data Generation

Determining the Error Distribution (ED) of an encoding technique involves evaluating it with datasets until a stopping criterion is met. Doing this with real-world datasets represents a challenge since statistical support for the results demands a large number of datasets, and we lack prior knowledge regarding the exact number needed. To circumvent this, we employed synthetic datasets.

The synthetic datasets were created through polynomial functions involving both numerical and categorical attributes (code available: https://github.com/celestun/polynomial-dataset-generator). To emulate properties found in datasets derived from real-world sources, we examined 10 real-world datasets to define their properties, such as the total number of numerical and categorical attributes, dataset size, complexity of the target (linear, no-linear), and others. The analyzed real-world datasets can be found in Appendix A, Table 3. These properties define the parameters of the polynomials that will generate the synthetic datasets. The parameters of the polynomials, expressed in set-builder notation, are shown in Table 1.

Table 1. Polynomial parameters to generate synthetic data

| Parameter name | Range | Description |
|---|---|---|
| Number of variables: $x$ | $\{x \mid v \in N, 5 \leq v \leq 34\}$ | Number of attributes in the synthetic dataset. |
| Variables domain: $d$ | $\{d \mid d \in \mathrm{R}, 0 \leq d \leq 1\}$ | The domain of the attributes. |
| Polynomial degrees: $P_d$ | $\{P_d \mid P_d \in N, 0 \leq P_d \leq 11\}$ | The exponent of each variable in the polynomial. |
| Polynomial terms: $P_t$ | $\{P_t \mid P_t \in \mathrm{N}, 1 \leq P_t \leq 13\}$ | The number of terms of the polynomial. |
| Coefficient values: $c$ | $\{c \mid c \in \mathrm{R}, -1 \leq c \leq 1\}$ | The value that multiplies each term. |

| | | |
|---|---|---|
| Number of variables in each term: $n_x$ | $\{n_x \mid n_x \in \mathrm{N},\ 1 \leq n_x \leq 5\}$ | The number of variables in each term. |
| Dataset size: $n$ | $\{n \mid n \in \mathrm{N},\ 1000 \leq n \leq 3000\}$ | The count of tuples |

To create the synthetic datasets, the polynomial parameters were assigned random values within predefined ranges (shown in Table 1), ensuring each generated dataset was unique. Given that the purpose of the synthetic datasets was to evaluate categorical encoders, categorical attributes were included in the datasets. It was observed that approximately 30% of the variables in the analyzed real-world datasets are categorical.

## Evaluated encoders

In this study, the most commonly used encoding techniques were evaluated, which are described below.

- *Ordinal or label encoding* assigns integer values to each unique category instance. It is suitable for ordinal categorical instances with a natural order. However, it is not recommended for nominal categorical variables, which can result in poor performance. According to Hancock and Khoshgoftaar (2020), applying ordinal encoding to nominal data introduces non-existent orders among categorical variables, potentially causing scale issues due to arbitrary assignments of order and scale.
- *One-hot encoding* transforms categorical instances into binary vectors, where each unique category corresponds to a binary digit (0 or 1) within the vector. This technique is well-suited for nominal data. However, it doesn't capture similarities between distinct categorical values since it assigns only 0 or 1 to its representations. Drawbacks of this method, as mentioned in Zheng and Casari (2018), include potential computational inefficiency, limited adaptability to new categories, and the curse of dimensionality.
- *Binary encoding* is similar to One-Hot Encoding, but instead of adding a binary vector for each distinct instance, it transforms the instance into a binary format by initially assigning an integer value to each category and subsequently converting it into a binary representation. In the case of a feature with *d* unique values, this process yields a total of $log_2(d)$ discrete values (Seger, 2018).
- *Count Encoder* assigns to each category instance the count or frequency of that category in the dataset. It is a simple yet effective way to represent categorical data as numerical values. It takes advantage of the frequency information of each category. The assumption is that the number of observations per category is somewhat characteristic of the target (Galli, 2022).
- The *Hashing Encoder* hashes categorical instances into a fixed-size space, typically resulting in a smaller dimensionality than One-Hot Encoding. It utilizes a hash function to map each category to a specific index in the hash space (Weinberger et. al., 2009). While efficient in terms of memory usage and computation, this method may lead to collisions where different categories are mapped to the same index, potentially causing information loss.

A drawback of these methods (Ordinal, One-hot, Binary, Count, and Hashing encoders) is their lack of consideration for potential patterns between the categorical attribute and other attributes in the dataset. However, alternative techniques are specifically designed to find such relationships using codes that preserve these patterns.

We hence define Pattern Preserving Codes (PPCs) as those numerical values that retain the possible relations between one selected categorical attribute and the remaining ones. Let's consider a collection of *n*-dimensional tuples (let's call it *U*) with a total of *m* elements. There are *n* unknown functions of *n-1* variables. We denote such function as $f_k$:

$$f_k(v_1, \ldots, v_{k-1}, v_{k+1}, \ldots, v_n); \; k = 1, \ldots, n \qquad (3)$$

Let us further suppose that there exists a method allowing us to approximate $f_k$ (from the tuples) with $F_k$. Represent the resulting n functions of *n-1* independent variables as $F_i$, consequently:

$$F_k \approx f_k(v_1, \ldots, v_{k-1}, v_{k+1}, \ldots, v_n); \; k = 1, \ldots, n \tag{4}$$

The difference between $f_k$ and $F_k$ will be denoted as $e_k$ such that, for attribute *k* and the *m* tuples in the dataset

$$e_k = max[abs(f_{ki} - F_{ki})]; \; i = 1, \ldots, m \tag{5}$$

PPCs minimize $e_k$ for all *k*. This is so because only those codes that maintain the relationships between variable *k* and the remaining $n-1$ variables (while doing so for every variable in the collection) effectively preserve the entire set of relations (i.e., patterns) found in the database, as in (6).

$$\Xi = min[max(e_k; \; k = 1, \ldots, n)] \tag{6}$$

Notice that this is a multi-objective optimization problem because complying with condition *k* in (5) for any given value of *k* may induce non-compliance for a different possible *k*. Using the min-max expression of (6) corresponds to selecting a particular point in Pareto's front (Deb et. al., 2000).

Some encoders are designed to find these PPCs. One example is CENG: *Categorical Encoding with Neural Networks and Genetic Algorithms* (Kuri, 2015). While this technique demonstrates accurate identification of PPCs using neural networks and genetic algorithms, it is computationally expensive in terms of processing power and time. To reduce the computational cost related to CENG, two algorithms were proposed: CESAMO (Kuri, 2018) and its multivariable version CESAMMO (Valdez et. al., 2022).

CESAMO encoder, which stands for Categorical Encoding by Statistical Applied Modeling, relies on statistical and numerical principles. The algorithm seeks PPCs for each categorical attribute in a dataset. To do so, it randomly selects numerical values that we will refer to as candidate codes (*cc*). The *cc* are evaluated in an approximation function (*F*) that finds the relationship between the categorical attribute and the other attributes in a dataset. The evaluation yields an error (*e*). Candidate codes are evaluated until a stopping criterion is met. The PPCs are defined as those that obtained the smallest error when evaluated in *F*. Finally, the categorical instances are replaced by the corresponding PPCs. The algorithm of CESAMO can be found in Appendix B.

Two questions may arise: a) How to define the approximation function, and b) How to establish the stopping criterion to determine when to cease sampling *cc*? Regarding A) the algorithm uses a mathematical model considering high-order relations, which consists of a universal polynomial approximation:

$$F(x) = c_0 + \sum_{i=1}^{6} c_i x^{2i-1} \tag{7}$$

It was shown in Kuri & Cartas (2014) that any continuous function may be approximated with a linear combination of monomials which has constant terms of odd degree. Regarding b), irrespective of the error distribution, the means of the errors will converge to a Normal distribution, indicating statistical stability. Additional sampling will have minimal impact on the characterization of the error population.

## Categorical Encoders Evaluation

We aimed to assess the encoders' ability to retain patterns in categorical attributes. The better the encoder preserves information within categorical attributes, the more effectively ML models perform when using these encoded datasets.

The synthetic datasets were converted into a classification problem, maintaining a balanced 50%-50% distribution of the targets. The encoders processed each dataset. Following the encoding stage, the dataset was solved as a classification problem by a supervised ML model. Given that the datasets were balanced, we used the Accuracy ($acc$) as the performance metric for the ML models, and then we defined the error ($e$) as $e = 1 - acc$.

We employed five distinct ML models to avoid biasing the evaluation towards a single supervised ML model: a Multilayer Perceptron Neural Network, a Logistic Regression, a Support Vector Machine, and the Gaussian Naive-Bayes model. These were implemented using the Sklearn and XGBoost libraries. The evaluated encoders included CESAMO, Binary, Hashing, One-hot, Ordinal, and Count encoders (code available: https://github.com/celestun/categorical-encoders-benchmark). For each model-encoder combination, we followed the next general steps to obtain the performance metrics:

- 1: Create a synthetic dataset using the polynomial function.
- 2: Apply the encoding technique and replace the categorical instances in the dataset.
- 3: Solve the encoded dataset with an ML algorithm and record the resulting error.
- 4: Repeat steps 1, 2, and 3 until 36 errors are reached.
- 5: Compute and record the mean of the errors.
- 6: Verify if the distribution of the errors' mean is normal. If true, proceed to step 7; otherwise, go to step 1.
- 7: Conclude the sampling process, and determine the mean and standard deviation from the errors' mean distribution

In step 4, the selection of a sample size of 36 follows a general rule of thumb grounded in the Central Limit Theorem, which states that as the sample size increases, the sample distribution tends to approximate a normal distribution, irrespective of the underlying population distribution (Hogg et. al., 1977). In step 6, to verify if the distribution is normal, the Л Distribution was implemented.

## Distribution

Goodness-of-fit (GoF) is a statistical test that assesses how well sample data aligns with an expected distribution, such as a normal distribution (in our case). Multiple methods have been proposed for assessing Normality. The Chi-Squared ($\chi^2$), Shapiro-Wilk, Anderson-Darling, and Kolmogorov-Smirnov are the most frequently employed.

The constraint of the tests outlined in the preceding paragraph lies in their emphasis on either rejecting or not the null hypothesis based on the significant evidence found. Failure to reject the null hypothesis does not confirm that the distribution is normal; instead, it suggests insufficient evidence to reject the null hypothesis. The absence of evidence, however, does not imply confirmation of normality. In our context, the objective is to confirm a normal distribution.

The Л distribution was proposed in Kuri and Lopez (2017) to tackle this constraint. It ensures that a sample is derived from a normal distribution with a certain probability. The foundation of Л distribution lies in the question: How probable is it to compute an experimental value of Л larger than ξ for a sample that is expected to be normal?, where Л is the value obtained after evaluating the data under equation 8, and ξ is a predefined critical value. To answer this question, the sample distribution is divided into quantiles. Each quantile spans the same area under the distribution curve. Figure 2 provides an example where the distribution is segmented into deciles (10 quantiles).

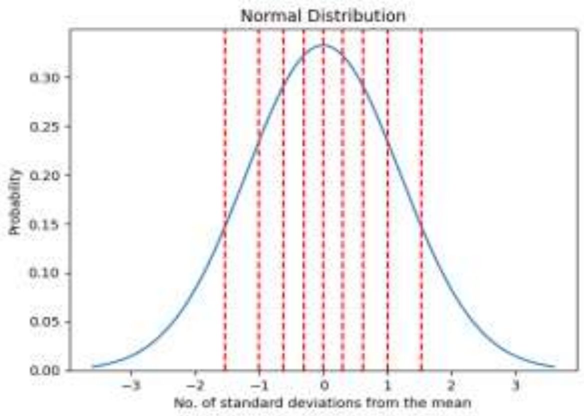

Figure 2. Normal distribution segmented into deciles. Each decile spans 10% of the area under the curve.

Let $Q$ represent the number of quantiles, $O_i$ denotes the number of observed events in the *i-th* quantile. $E_i$ is the number of expected observations in the *i-th* quantile, and $\Phi$ is the minimum number of observations required per quantile. Also, let $p$ denote the probability that Л exceeds ξ when data is normally distributed, and there are at least $\Phi$ events in all quantiles. Then:

$$Л = \sum_{i=1}^{Q} \frac{(Q_i - E_i)^2}{E_i} \wedge [O_i \geq \Phi \, \forall \, i] \tag{8}$$

## 4 Experimental results

The evaluation of the encoders required around 3,000 datasets for each encoder-model combination to satisfy the statistical criterion, achieving normality. Once the *Mean of the Error Distribution* (MED) became Normal, we determined its $\mu_{MED}$ and $\sigma_{MED}$ . With these values, the ED mean $\mu_{ED}$ and its standard deviation $\sigma_{ED}$ were determined as follows:

$$\mu_{ED} \approx \mu_{MED} \, , \quad \sigma_{ED} = \sqrt{ss} \cdot \sigma_{MED} \tag{9}$$

Where the sample size *ss* was 36. Table 2 shows the obtained values after calculating $\mu_{ED}$ and $\sigma_{ED}$.

Table 2. $\mu$ and $\sigma$ of the errors from the evaluated categorical encoders

| Encoder | Mean error | Standard deviation |
|---|---|---|
| CESAMO | 0.2994374188 | 0.0912776249 |
| Hashing | 0.3069885601 | 0.1028621334 |
| Binary | 0.3105365472 | 0.0909518922 |
| One hot | 0.3299986654 | 0.1008621334 |
| Ordinal | 0.3315438146 | 0.1014268199 |
| Count | 0.3572900108 | 0.1106407878 |

As can be seen in Table 2, CESAMO was the encoder that got the lowest mean error among all encoding techniques. We calculated the ME, which is at $-\sqrt{5}\sigma_{ED}$ to the left of $\mu_{ED}$. Results are shown in Figure 3.

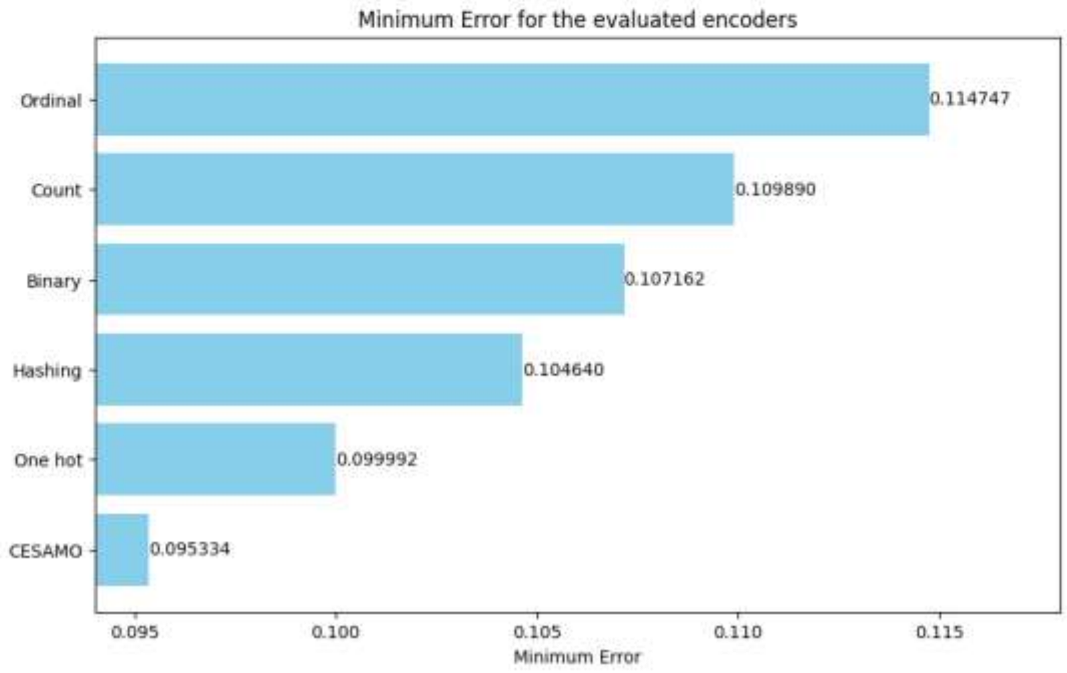

Figure 3. Minimum Error ($\mu_{ED} - \sqrt{5}\sigma_{ED}$) found for the evaluated encoders

Figure 3 shows the Minimum Error obtained by the evaluated encoders. As can be seen, CESAMO had the smallest ME among all the encoders, followed by the One-hot encoder.

Excluding the CESAMO encoder, which is extensively evaluated for the first time, this finding is consistent with some of the related works analyzed. For instance, Zhu et al., 2024, also observed that one-hot encoding performs better for models based on Multilayer Perceptron, and Matteucci et al., 2024, similarly found that the one-hot encoder achieved superior performance with the logistic regression model. While these studies identify specific model-encoder combinations that yield the best results, our approach involves calculating averages per encoder (without maintaining the resolution of the model-encoder relationship). Nonetheless, both studies confirm that the one-hot encoder consistently ranks higher.

On the contrary, Figure 3 also shows that the encoder exhibiting the highest Minimum Error was the Ordinal Encoder, surprisingly one of the most frequently utilized encoders in practice. The count (frequency) encoder ranked second lowest in performance. This finding is consistent with the results of Pargent et al., 2019, who also observed poor performance from Ordinal and Count encoders overall.

## 5 Conclusions

In this paper, we evaluated the performance of categorical encoders in preserving patterns during the conversion of categorical attributes into numerical values. The evaluation was conducted with synthetic datasets, enabling us to validate our findings statistically. The metric employed to measure and compare the performance of the categorical encoders was the Minimum Error, which represents the value situated at $\mu - \sqrt{5}\sigma$ of the Error Distribution of the encoders.

Categorical encoders were combined with 5 Machine Learning models to measure their performance, assuming that the better a categorical encoder preserves patterns, the better the performance of an ML model when using the encoded dataset. On average, each *encoder-model* combination was evaluated on approximately 3000 datasets until the error mean reached normality.

The results indicated that CESAMO achieved the lowest Minimum Error (indicating the best performance), followed by the One-Hot encoder. Conversely, the Ordinal encoder, a widely used technique, exhibited the poorest performance among the encoders. These findings suggest that CESAMO outperformed the other encoders due to its ability to identify Pattern Preserving Codes, which maintain the relationship between the categorical attribute and other attributes of the dataset. In contrast, the Ordinal encoder often introduces non-existent patterns when converting categorical attributes into numbers, as it arbitrarily assigns integer values to categorical instances. These results align with other extensive related works.

## Acknowledgments

This work has been carried out with partial support from the DGAPA UNAM-PAPIIT project number IN104424 and CONAHCYT project number CF-2023-G-64.

## References

Kuhn, M., & Johnson, K. (2019). Feature engineering and selection: A practical approach for predictive models. Chapman and Hall/CRC.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, É. (2011). Scikit-learn: Machine learning in Python. the Journal of machine Learning research, 12, 2825-2830.

Chen, T., & Guestrin, C. (2016, August). Xgboost: A scalable tree boosting system. In Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining (pp. 785-794).

Valdez-Valenzuela, E., Kuri-Morales, A., & Gomez-Adorno, H. (2021). Measuring the effect of categorical encoders in machine learning tasks using synthetic data. In Advances in Computational Intelligence: 20th Mexican International Conference on Artificial Intelligence, MICAI 2021, Mexico City, Mexico, October 25–30, 2021, Proceedings, Part I 20 (pp. 92-107). Springer International Publishing.

Zhu, W., Qiu, R., & Fu, Y. (2024). Comparative Study on the Performance of Categorical Variable Encoders in Classification and Regression Tasks. arXiv preprint arXiv:2401.09682.

Pargent, F., Bischl, B., & Thomas, J. (2019). A benchmark experiment on how to encode categorical features in predictive modeling. München: Ludwig-Maximilians-Universität München.

Matteucci, F., Arzamasov, V., & Böhm, K. (2024). A benchmark of categorical encoders for binary classification. Advances in Neural Information Processing Systems, 36.

Seca, D., & Mendes-Moreira, J. (2021, March). Benchmark of encoders of nominal features for regression. In World Conference on Information Systems and Technologies (pp. 146-155). Cham: Springer International Publishing.

Hancock, J. T., & Khoshgoftaar, T. M. (2020). Survey on categorical data for neural networks. Journal of big data, 7(1), 28.

Zheng, A., & Casari, A. (2018). Feature engineering for machine learning: principles and techniques for data scientists. " O'Reilly Media, Inc.".

Seger, C. (2018). An investigation of categorical variable encoding techniques in machine learning: binary versus one-hot and feature hashing.

Galli, S. (2022). Python feature engineering cookbook: over 70 recipes for creating, engineering, and transforming features to build machine learning models. Packt Publishing Ltd.

Weinberger, K., Dasgupta, A., Langford, J., Smola, A., & Attenberg, J. (2009, June). Feature hashing for large scale multitask learning. In Proceedings of the 26th annual international conference on machine learning (pp. 1113-1120).

Deb, K., Agrawal, S., Pratap, A., & Meyarivan, T. (2000). A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In Parallel Problem Solving from Nature PPSN VI: 6th International Conference Paris, France, September 18–20, 2000 Proceedings 6 (pp. 849-858). Springer Berlin Heidelberg.

Kuri-Morales, A. F. (2015, July). Categorical encoding with neural networks and genetic algorithms. In WSEAS Proceedings of the 6th International Conference on Applied Informatics and. Computing Theory (pp. 167-175).

Kuri-Morales, A. (2018). Pattern discovery in mixed data bases. In Pattern Recognition: 10th Mexican Conference, MCPR 2018, Puebla, Mexico, June 27-30, 2018, Proceedings 10 (pp. 178-188). Springer International Publishing.

Valdez-Valenzuela, E., Kuri-Morales, A., & Gomez-Adorno, H. (2022, October). CESAMMO: Categorical Encoding by Statistical Applied Multivariable Modeling. In Mexican International Conference on Artificial Intelligence (pp. 173-182). Cham: Springer Nature Switzerland.

Kuri-Morales, A., & Cartas-Ayala, A. (2014). Polynomial multivariate approximation with genetic algorithms. In Advances in Artificial Intelligence: 27th Canadian Conference on Artificial Intelligence, Canadian AI 2014, Montréal, QC, Canada, May 6-9, 2014. Proceedings 27 (pp. 307-312). Springer International Publishing.

Hogg, R. V., Tanis, E. A., & Zimmerman, D. L. (1977). Probability and statistical inference (Vol. 993). New York: Macmillan.

Kuri-Morales, A. F., & López-Peña, I. (2017). Normality from monte carlo simulation for statistical validation of computer intensive algorithms. In Advances in Soft Computing: 15th Mexican International Conference on Artificial Intelligence, MICAI 2016, Cancún, Mexico, October 23–28, 2016, Proceedings, Part II 15 (pp. 3-14). Springer International Publishing.

**Appendix A**

Table 3. Real world datasets analyzed to generate the synthetic data

| **Dataset name** | **Description** |
|---|---|
| Abalone data | It consists of biological measurements, including length, diameter, height, and weight of abalones, along with the number of rings representing their age. |
| Car Evaluation Database | It assesses the acceptability of cars based on various attributes. It includes features such as price, maintenance cost, number of doors, seating capacity, and safety. |
| Hepatitis Domain | It contains information related to the medical domain of hepatitis. It includes various clinical and laboratory attributes such as age, sex, symptoms, and blood test results. |
| breast cancer wisconsin | It is focused on breast cancer diagnosis. It encompasses features derived from digitized images of breast cancer biopsies, including characteristics like cell nucleus properties. |
| cpu performance | It is related to computer hardware performance. It includes attributes such as clock cycle time, cache size, and other technical specifications. |
| Yeast Data Set | It is centered around the study of yeast protein localization patterns. It comprises attributes related to the amino acid sequences and other characteristics of yeast proteins. |
| Servo Data Set | It focuses on the control of a servo system. It includes attributes related to the input and output signals of the system |
| Ecoli Data Set | It is dedicated to the study of the subcellular localization of proteins in Escherichia coli (E. coli) bacteria. It includes attributes related to various protein properties |
| Adult Dataset | Also known as the "Census Income" dataset, it contains demographic information about adults, and the task is to predict whether an individual earns more than $50,000 per year. |
| Bank Marketing Dataset: | It contains information related to the direct marketing campaigns of a Portuguese banking institution. The task is to predict whether a client will subscribe to a term deposit (binary outcome: yes/no). |

All these datasets are available at: https://archive.ics.uci.edu/

## Appendix B

CESAMO algorithm to transform categorical variables into numbers.

Algorithm 1: CESAMO encoder

```
1   For ca_to_map in ca_list:
2       While e_avg distribution != Gaussian:
3           cc ← randomly_generate_cc( )
4           assign_cc_to_all_instances_in_ca_to_map( )
5           i_var ← randomly_select_independent_var( )   # cannot be ca_to_map
6           e ← F(ca_to_map)   # apply approx. function
7           e_list.append(e)
8           e_avg ← avg(e_list)
9           determine_if_distribution_became_gaussian(e_avg)
10          If e_avg distribution == Gaussian Then break Else continue
11      ppcs = set_ppcs(e_list)   # set as ppcs those cc that yielded the min error
```

Where

*ds* = dataset
*ca_to_map* = categorical attribute to map to a numerical var
*a_list* = list with all the attributes in *ds*
*ca_list* = list with all the categorical attributes
*i_var* = independent variable to be used in the approximation, cannot be ca_to_map
*e* = error
*e_avg* = errors mean
*cc* = candidate code
*F* = Approximation function