Encoding Matters: Impact of Categorical Variable Encoding on Performance and Bias

Daniel Kopp[1], Benjamin Maudet[1,2], Lisheng Sun-Hosoya[2], and Kristin P. Bennett[3]
1- ChaLearn. 2- U. Paris-Saclay. 3- RPI.

Abstract.
Encoding categorical variables impacts model performance and can introduce bias in supervised learning, particularly affecting fairness when some groups are under-represented. We analyze the effects of different encoding methods on synthetic and real datasets to mitigate unintended model reliance on specific variables. We propose CaVaR (Categorical Variable Reliance) to quantify model reliance on variables and an Availability Index to measure CaVaR's sensitivity to partial encoding changes. A high Availability Disparity, measured by the standard deviation of the Availability Index across encodings, highlights potential bias from mixed encodings. The results suggest encoding all categorical variables uniformly, regardless of their ordinal or nominal nature, may reduce bias, with the choice guided by computational and performance considerations.

1 Background and motivations

In machine learning, it is common to encounter various data types that need to be properly encoded before being input into the learning model. Some models, such as linear models and neural networks, work well with quantitative data and require numerical inputs [3]. Other models, like decision trees and grid models, can also handle qualitative data, such as categorical variables [5]. Due to the increasing importance of neural networks in modern AI, it is crucial to correctly encode categorical variables to ensure these models function properly, achieve good generalization, and avoid bias[1] This is the focus of this paper.

Categorical variables are variables that can take one value from a finite set of possible values. Each possible value of a variable is referred to as a level. We consider two types of categorical variables: ordinal variables with a meaningful ordering, *e.g.,* education levels, and nominal variables, which are qualitative with no meaningful ordering, *e.g.,* race. Although nominal variables may be numbered (*e.g.,* user ID's), these numbers do not indicate any particular order. Textbooks and on-line resources[6, 10] usually recommend encoding ordinal and nominal variables differently. Typically, ordinal variables are encoded with a numeric value using their natural order (referred to as "Natural Ordinal" encoding). In contrast, nominal variables are encoded with "One-Hot" encoding, converting k-level categorical variables into a k-dimensional binary vector representation where only one of the levels is "hot" (set to 1) for each observation, while all the others are "cold" (set to 0) [11]. However, these "typical" encoding choices are

[1]To facilitate the reading of this paper, we added a glossary in appendix provided here: `https://github.com/danielkopp4/ESANN2025Appendix`. However, this paper should be readable without the appendix.

far from neutral. Changing the encoding of a variable can influence the model's reliance on that variable, thereby affecting how well the model generalizes to unseen data. For instance, a model may find it easier to learn a relationship with a variable that has a weaker dependence on the target compared to another, a phenomenon often referred to as shortcut learning [2]. In the case of protected variables (such as gender or race), shortcut learning can be associated with fairness issues by reinforcing or introducing societal biases.

2 Problem setting and methodology

Problem Setting. We study supervised learning problems where data samples $(\mathbf{x}, y) \in \mathcal{X}^d \times \mathcal{Y}$ are drawn from a joint distribution $P(\mathbf{x}, y) = P(\mathbf{x})P(y|\mathbf{x})$. The goal is to approximate $P(y|\mathbf{x})$ using machine learning (ML).

Training data $\{(\mathbf{x}^1, y^1), \ldots, (\mathbf{x}^n, y^n)\}$ are drawn *iid* from $P(\mathbf{x}, y)$. We treat the learning process, including encoding, as a black-box model M that outputs an estimator $\hat{P}_M(y|\mathbf{x})$ of $P(y|\mathbf{x})$. In our experiments, encoding is the only variable adjusted; all other factors, including hyperparameters, are held constant. If a variable is disproportionately emphasized in $\hat{P}_M(y|\mathbf{x})$, failing to reflect true statistical dependencies, bias may be introduced. Here, we assume all components of $\mathbf{x}$ are categorical variables and focus on encoding's effect on algorithmic bias.

Categorical variables are drawn from a discrete joint distribution, where a "cell" represents a specific combination of variable levels (*e.g.,* if $\mathbf{x}$ is composed of the two variables "race" and "education level", individuals with "graduate school" education and "Black" race form one cell). Each variable x_i can take k_i levels, and the discrete domain of inputs is represented by a tensor $\mathbf{X}$ of size $K = \prod_{i=1}^{d} k_i$, covering all cells.

Training data follow the natural distribution $P(\mathbf{x})$, while test data are drawn either from the same in-domain distribution or an out-of-domain uniform distribution where all cells in $\mathbf{X}$ have equal probability, including unseen cells.

Methodology. To assess the impact of disparate variable encoding on algorithmic bias, we introduce **CaVaR** (Categorical Variable Reliance), a metric that quantifies the reliance of a predictor $\hat{P}_M(y|\mathbf{x})$ on a subset S of categorical variables. This reliance is measured by evaluating prediction changes when values in S are permuted, neutralizing S's effect during inference. The formula is:

$$\mathrm{CaVaR}_S(M) = \frac{1}{\kappa \times K} \sum_{\pi \in \Pi(S)} \left\| \hat{P}_M(y = 1|\mathbf{X}) - \hat{P}_M(y = 1|\pi(\mathbf{X})) \right\|_1 , \quad (1)$$

where $\Pi(S)$ is the set of κ permutations of S (that can be subsampled for computational reasons), K is the number of cells in $\mathbf{X}$, and the L1-norm difference quantifies the model's reliance on S. This computation does not depend on ground-truth y, making it robust to under-represented cells in real data.

To study encoding effects on M, assume all variables use encoding a, except S, which uses b, represented by $M[a, S \leftarrow b]$. We introduce the notation:

$$\mathrm{CaVaR}_S(b;a) = \mathrm{CaVaR}_S(M[a, S \leftarrow b]) \ . \tag{2}$$

$\mathrm{CaVaR}_S(a;a)$ reflects reliance under uniform encoding a, while $\mathrm{CaVaR}_S(b;a)$ shows reliance when S uses b. Averaging the differences across all $b \neq a$ and subsets S yields the **Availability Index**:

$$\mathrm{Avail}(a) = \frac{1}{N_S \times (C-1)} \sum_S \sum_{b \neq a} (\mathrm{CaVaR}_S(a;a) - \mathrm{CaVaR}_S(b;a)) \ . \tag{3}$$

A high positive $\mathrm{Avail}(a)$ indicates greater reliance when variables are encoded with a compared to b. Variability in $\mathrm{Avail}(a)$ across encodings is summarized by the **Disparity Index**:

$$\mathrm{Disparity} = \mathrm{Std}(\mathrm{Avail}(a)). \tag{4}$$

If Disparity is high, using mixed encodings risks bias by favoring certain variables. Conversely, near-zero Disparity suggests mixed encodings are safe. When Disparity is significant, we recommend uniform encoding for all variables to avoid bias.

3 Experimental conditions

After evaluating various encoding strategies (Appendix C), we selected One-Hot, Ordinal, and Target Continuous encodings for illustration. We also introduce Random Ordinal and Target Ordinal, as specialized forms of Ordinal encoding.

One-Hot encoding is emphasized for its widespread use with nominal variables and its neutrality, as it avoids assumptions about category relationships. In contrast, Ordinal encodings incorporate prior information, which can either aid or hinder model learning. Random Ordinal represents a worst-case scenario, as arbitrary ordering introduces spurious continuity, implying nonexistent interpolation between levels. In our experiments, we average results over multiple random orderings. Target Ordinal assigns integers 0 to $k-1$ (where k is the number of levels) to maximize correlation with the target. Target Continuous encoding maps categories directly to their target variable means, often outperforming Target Ordinal unless tied means cause level merging.

Experiments were carried out on synthetic and real data, described in details in Appendix B. Briefly, the synthetic datasets consist of two-dimensional binary classification problems, with both input variables x_1 and x_2 having 8 levels. They were designed to illustrate various special cases:
Synthetic Simple: x_1 and x_2 are identical and identically dependent on y; any difference in variable availability can only arise from coding.
Synthetic Crossed: This example (similar the the XOR problem) is designed such that neither variable is individually correlated with the target and all values in the marginal distributions are tied. Target Continuous encoding is expected to fail on that example.

Synthetic Doughnut: This is another example of non-linearly separable classes, in which neither variable is correlated with the target. It is designed to show that Target Encoding can be beneficial even when variables are not correlated with the target, if values in the marginal distributions are not tied.

The real datasets, described in details in Appendix B, comprised of Adult Income [1], Bank Marketing [7], Cardiovascular Disease [4], Cook County Sentencing [8], and Mushroom [9]. Their number of features vary between 11 and 22. The number of samples is between 8124 and 266435 samples. The maximum number of levels is between 8 and 1316.

For all datasets, we used 50 examples of each class for training to exacerbate the difficulty of generalizing well out-of-domain. The choice of the same number of examples per class is to simplify the analysis and avoid adding a factor of variability that may confound the results. In future work, we could also vary the number of examples per class. For real data, we used all the remaining samples as test data (so called identically and independently distributed or *iid* test set) and for synthetic data, we drew 1000 samples from the same distribution as the training data. We use the test samples to evaluate the Balanced Accuracy (average of the accuracy on the positive class and the negative class). Additionally, we created a second test set to evaluate both the Availability and Disparity indices, populating the tensor $\mathbf{X}$, providing values for the levels in the ranges defined by the training data distribution, but enforcing uniform sampling.

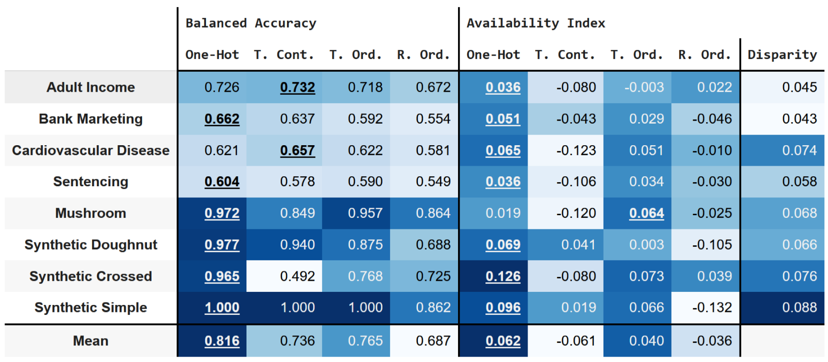
All experiments reported in the next section were carried out with a 2 layer neural network with 5 hidden units and weight decay regularization of 0.01, using the scikit-learn package. Experiments with various other models are provided as supplemental material in our Github Repo.

4 Results

Visual inspection of the two-dimensional Simple Synthetic data example (Appendix E) demonstrates that encoding variables in different ways can change their availability and therefore result in bias. This suggests that all variable should be encoded in the same way, regardless of whether they are nominal or ordinal, to avoid exacerbating or introducing bias. Because of space limitations, we only discuss in the body of the paper the results of Table 1, which present systematic experimental comparisons of a selection of codes on synthetic and real data, for the two-layer neural network.

We chose a neural network because encoding categorical variables as numerical values is required for such models. We chose two layers and 5 hidden units to have a small model that is capable of making non-linear separations. We also chose to train models on very few examples (50 of each class) so data sparsity will affect variable availability more strongly. We compute the balanced accuracy (average of the accuracy on the positive class and the negative class) and whether encoding affects variable availability with the Availability Index (Equation 3). The last column is encoding Disparity (Equation 4). A positive Disparity is a red flag for mixing codes.

Table 1: Performance indices for various datasets, real (top) and synthetic (bottom). All error bars are on the last significant digit shown (see details in Appendix F). Disparity is significantly positive for all datasets, hinting that the same code should be used for all variables to avoid coding bias. One-Hot encoding is generally best, except in a few cases (see text). The significance of the effect was assessed using error bars, as explained in the appendix (Section F).



	Balanced Accuracy				Availability Index				
	One-Hot	T. Cont.	T. Ord.	R. Ord.	One-Hot	T. Cont.	T. Ord.	R. Ord.	Disparity
Adult Income	0.726	**0.732**	0.718	0.672	**0.036**	-0.080	-0.003	0.022	0.045
Bank Marketing	**0.662**	0.637	0.592	0.554	**0.051**	-0.043	0.029	-0.046	0.043
Cardiovascular Disease	0.621	**0.657**	0.622	0.581	**0.065**	-0.123	0.051	-0.010	0.074
Sentencing	**0.604**	0.578	0.590	0.549	**0.036**	-0.106	0.034	-0.030	0.058
Mushroom	**0.972**	0.849	0.957	0.864	0.019	-0.120	**0.064**	-0.025	0.068
Synthetic Doughnut	**0.977**	0.940	0.875	0.688	**0.069**	0.041	0.003	-0.105	0.066
Synthetic Crossed	**0.965**	0.492	0.768	0.725	**0.126**	-0.080	0.073	0.039	0.076
Synthetic Simple	**1.000**	1.000	1.000	0.862	**0.096**	0.019	0.066	-0.132	0.088
Mean	**0.816**	0.736	0.765	0.687	**0.062**	-0.061	0.040	-0.036	

Let us first analyze Balanced Accuracy results that are obtained by **coding ALL variables in the same way** (first 4 columns), to evaluate whether some codes yield better performance than others overall. At first glance, we notice One-Hot encoding is generally best and Random Ordinal worst. This is true on average for the datasets considered. There are only 2 exceptions: (1) For Cardiovascular disease, Target Encoding is better. This can be explained by the fact that the variables in that dataset are discretized versions of continuous variables, hence they are all ordinal. There is therefore no benefit of One-Hot encoding, known to be tailored to representing nominal variables well, but at the expense of loosing order information for ordinal variables. (2) Another interesting case is that of the Mushroom dataset: there, Target Continuous encoding does not do better than Random Ordinal but Target Ordinal encoding yields similar performance as One-Hot encoding. The good performance of One-Hot encoding is explained by the fact that all variables are nominal. The better performance of Target Ordinal compared to Target Continuous is more subtle. Intuitively, one might think the Target Continuous is a more informative code because Target Ordinal is a discretized code. However, when mean target values are tied, Target Continuous assigns the same value to multiple levels, resulting in a loss of information. Two competing factors determine why one code may be more informative than another, as demonstrated in our synthetic examples designed to highlight this effect. In the Synthetic Crossed example, perfect ties across all levels render Target Continuous non-informative (all levels map to the same value). In contrast, for the Synthetic Doughnut example, Target Continuous outperforms Target Ordinal.

Let us now turn our attention to the Disparity Index. The Disparity is always significant compared to 0, within the error bar, indicating that all variables should be encoded uniformly. This effect is more pronounced in synthetic examples, as real data lacks controlled joint distributions and variable redundancy. Examining Availability reveals a strong correlation with Balanced Accuracy. One-Hot encoding generally maximizes variable availability and is preferable, even with small datasets. Target Ordinal ranks second in Availability but performs worse in Balanced Accuracy, likely due to the issue of ties previously discussed.

5 Conclusion

Our findings indicate that encoding categorical variables differently may introduce bias and we advocate using the same code for all categorical variables to reduce out-of-distribution bias. The optimal code to be chosen depends on the predictive model, the number of levels of variables, the amount of available data, and the specific problem, and can be selected *e.g.,* by cross-validation using the performance metric (e.g. balanced accuracy).

Acknowledgements

Work partially supported by ANR Chair of AI HUMANIA ANR-19-CHIA-0022.

References

[1] B. Becker and R. Kohavi. Adult. UCI Machine Learning Repository, 1996.

[2] R. Geirhos, J.-H. Jacobsen, C. Michaelis, R. Zemel, W. Brendel, M. Bethge, and F. A. Wichmann. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2(11):665–673, nov 2020.

[3] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning.* MIT press, 2016.

[4] R. K. Halder. Cardiovascular disease dataset, 2020.

[5] G. James, D. Witten, T. Hastie, R. Tibshirani, et al. *An introduction to statistical learning*, volume 112, page 315. Springer, 2013.

[6] M. Kuhn and K. Johnson. *Feature engineering and selection: A practical approach for predictive models.* Chapman and Hall/CRC, 2019.

[7] R. P. Moro, S. and P. Cortez. Bank Marketing. UCI Machine Learning Repository, 2014.

[8] C. C. S. A. Office. Cook county sentencing data, 2020.

[9] J. Schlimmer. Mushroom. UCI Machine Learning Repository, 1981.

[10] scikit-learn developers. Encoding of categorical variables. `https://inria.github.io/scikit-learn-mooc/python_scripts/03_categorical_pipeline.html`.

[11] A. Zheng and A. Casari. *Feature engineering for machine learning: principles and techniques for data scientists.* " O'Reilly Media, Inc.", 2018.

The following supplemental material is not part of the paper and should not be construed as a violation of the page limit. The paper is self-contained without the supplemental material and reviewers are not obliged to take it into account.

A Glossary

Glossary

Algorithmic bias This is the tendency of a machine learning training algorithm to favor certain predictions due to its own design, the training data, or the model design. 2, 7

Bias Systematic errors or deviations that affect the performance or fairness of a model. In machine learning, three types of bias are encountered: algorithmic bias, statistical bias, and data bias . 1, 2

Categorical variable Represents data grouped into distinct categories or classes rather than numeric values. Each category is mutually exclusive, meaning an instance can belong to only one category at a time. 1, 7, 8

Data bias The error introduced in data by human biases including confirmation bias, selection bias. 7

Disparate impact Analyzing whether a model's decisions result in different outcomes across groups. If a model inadvertently results in a negative impact for a particular group, it may be considered unfair even if the model itself does not explicitly include bias. 7

Fairness Refers to the principle that a model should make equitable decisions, predictions, or classifications across diverse groups and not systematically disadvantage or favor certain groups based on characteristics like race, gender, age, socioeconomic status, or other Protected attributes. The goal is to ensure that the model's outcomes are free from harmful biases and that it treats all individuals or groups in a just and unbiased manner. Two cases of fairness includes group fairness, disparate impact. 2, 7

Group fairness Ensuring that model performance is similar across different demographic or protected groups. For instance, a classifier is considered fair under group fairness if it predicts outcomes at similar rates for different groups, such as ethnic or gender groups.. 7

Level One specific category or distinct value within a categorical variable. 8, 9

Nominal variable Type of categorical variable, where the categories or values represent distinct groups with no inherent order or ranking. Each category within a nominal variable is simply a label, and there is no logical sequence or comparison between them. 1

One-Hot Encoding encodes a categorical variable x_i with k_i distinct levels into a binary vector. Each index of the vector corresponds to one level of the variable. When $x_i = l$, the resulting vector has a "1" ("hot") at the index representing level l and "0"s at all other indices. The resulting encoded representation has a dimension equal to k_i, and this process is repeated for all levels.. 3

Ordinal Encoding Encodes of categorical variable $x_i = l$ as a bijection of the set of possible levels to $\{0, \dots, k_i - 1\}$ where k_i is the number of levels of x_i. The specific choice of bijection will affect the ordering of the levels. 3, 8, 9

Ordinal variable Type of categorical variable where the categories or levels have a meaningful order or ranking, but the intervals between them are not consistent or defined. Unlike nominal variables, ordinal variables allow for a hierarchy or sequence among the categories, though they do not specify exact differences between them.. 1

Protected Variable In the context of fairness, a protected variable refers to a characteristic or attribute of individuals that is legally or ethically considered sensitive and should not be used in a discriminatory or unfair manner. These variables include demographic attributes (e.g., race, gender, age, ethnicity, religion, sexual orientation, disability) or socioeconomic factors (e.g., income, education level, family background). They are often safeguarded by laws, ethical guidelines, or policies to prevent discrimination.. 2, 7

Random Ordinal Encoding Special case of Ordinal encoding which chooses the ordering to be selected at random. This strategy shows the case of an uninformative ordering. 3

Reliance The extent to which a model depends on a specific variable or set of variables when making predictions. It measures the change in a model's predictions when a variable or set of variables is altered or removed. When a model relies heavily on a certain set of variables, it uses those variables as a basis for its decisions, sometimes at the expense of other potentially informative variables. 2

Robustness A model's ability to maintain reliable, consistent performance when faced with variations, distortions, or challenges that were not present in the training data. A robust model performs well not only on the data it was trained on but also in conditions that differ from its training environment, such as with noisy data, minor adversarial attacks, or shifts in data distribution. 9

Shortcut learning A model's tendency to rely on simple, easily learnable features, patterns or spurious correlations in the data (shortcuts) rather than learning the more complex, genuine features that are essential for robust and accurate generalization. These shortcuts allow the model to achieve high performance on the training data or certain test sets, but the model may fail when presented with new data that does not contain these same superficial patterns. 2

Statistical bias The error introduced by approximating a complex real-world problem with a simplified model family. 7

Systematic errors Consistent, repeatable errors that occur predictably due to flaws in data collection, measurement, or model assumptions. These errors arise at the source (cause) as biases in the process and propagate to the evaluation (consequence), skewing predictions or performance metrics. In the bias-variance tradeoff, systematic errors correspond to bias—the model's inability to capture the true relationship—leading to consistently inaccurate outcomes. Unlike random errors (variance), systematic errors shift results in one direction and require improvements in design, data, or assumptions to mitigate. . 7

Target Continuous Encoding Encodes a level of categorical variable $x_i = l$ as the mean of the target variable y for all samples that satisfy $x_i = l$. This process is repeated for all levels of x_i. This encoding can produce a loss of information if the mean of y is close in value for pairs of levels. 3

Target Ordinal Encoding Special case of Ordinal encoding where the levels of a categorical variable are ordered based on the Pearson correlation coefficient between each level (treated as a binary indicator) and the target variable y.. 3

B Datasets

B.1 Synthetic data

We describe in some details the generating process of our synthetic datasets.

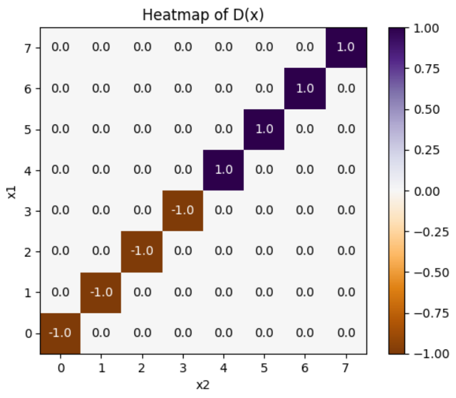
B.1.1 Synthetic Simple

In this synthetic dataset, x_1 and x_2 are identical and exhibit the same dependency on the target variable y. Any observed difference in variable availability can only result from differences in encoding methods.

The dataset consists of three categorical variables: x_1, x_2, and y, where y is the target variable and x_1 and x_2 are input variables. The training data is generated under the following rules:

- $x_1 \sim \mathcal{U}(\{0, 1, \ldots, 7\})$,
- $x_2 = x_1$,
- $y = 1$ if $x_2 \geq 4$, and $y = -1$ if $x_2 < 4$.

This setup ensures there are 8 unique (x_1, x_2) pairs, and the dataset contains a total of 80 training samples, with 10 samples for each pair. The structure of this dataset is visualized in Figure 1, which shows a heatmap of $D(x)$. The diagonal region $(x_1 = x_2)$ corresponds to high-density training samples and probabilities of $\{1, -1\}$, while off-diagonal regions have zero density, thus resulting in zero $D(x)$ values, as no training samples are generated there.



x1 \ x2	0	1	2	3	4	5	6	7
7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
6	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
5	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
4	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0
3	0.0	0.0	0.0	-1.0	0.0	0.0	0.0	0.0
2	0.0	0.0	-1.0	0.0	0.0	0.0	0.0	0.0
1	0.0	-1.0	0.0	0.0	0.0	0.0	0.0	0.0
0	-1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Fig. 1: **Synthetic Simple Dataset:** Training samples are generated as follows: $x_1 \sim \mathcal{U}(\{0, 1, \ldots, 7\})$, $x_2 = x_1$, and $y = \mathbf{1}(x_2 - 3.5)$, where $\mathbf{1}(x) = 1$ if $x > 0$ and -1 otherwise. To visualize the density of training examples, we present a heatmap of $D(\mathbf{x})$, defined as $D(\mathbf{x}) = (2P(y = 1|\mathbf{x}) - 1)\frac{\rho(\mathbf{x})}{\rho_{\max}}$ where $\rho(\mathbf{x})$ is number of samples at $\mathbf{x}$, $\rho_{\max}$ is the maximum sample number, and $P(y = 1 \mid \mathbf{x})$ is the probability of the positive class. This equation scales the probability of class $y = 1$ to the range $[-1, 1]$, weighted by the relative density of samples. The diagonal region $(x_1 = x_2)$ shows probabilities of $\{1, -1\}$ due to the data generation process, while off-diagonal regions have zero density, thus resulting in zero $D(\mathbf{x})$ values, as no training samples are generated there.

B.1.2 Synthetic Crossed

In this case, we have three variables: x_1, x_2, y where y is the target and x_1, x_2 are the input variables. The training data is generated such that $x_1, x_2 \in \{1, 2, \cdots, 8\}^2$ and when $x_1 = x_2$, $y = 1$, when $x_1 = 8 - x_2$, $y = -1$. All other cases have no training samples. This example is similar to the XOR problem where there is no independent dependence between the input variables and the target. The only relationship that can be determined from the inputs require the joint relationship between x_1, x_2 and y. This relationship is non-linearly separable. Shown in figure 2.

B.1.3 Synthetic Crossed

In this case, we have three variables: x_1, x_2, y where y is the target and x_1, x_2 are the input variables. The training data is generated such that $x_1, x_2 \in \{1, 2, \cdots, 8\}^2$ and when $x_1 = x_2$, $y = 1$, when $x_1 = 8 - x_2$, $y = -1$. All other cases have no training samples. This example is similar to the XOR problem where there is no independent dependence between the input variables and the target. The only relationship that can be determined from the inputs require the joint relationship between x_1, x_2 and y. This relationship is non-linearly separable. Shown in figure 2.

B.1.4 Synthetic Crossed

In this case, we have three variables: x_1, x_2, y where y is the target and x_1, x_2 are the input variables. The training data is generated such that $x_1, x_2 \in \{1, 2, \cdots, 8\}^2$ and when $x_1 = x_2$, $y = 1$, when $x_1 = 8 - x_2$, $y = -1$. All other cases have no training samples. This example is similar to the XOR problem where there is no independent dependence between the input variables and the target. The only relationship that can be determined from the inputs require the joint relationship between x_1, x_2 and y. This relationship is non-linearly separable. Shown in figure 2.

B.1.5 Synthetic Crossed

In this case, we have three variables: x_1, x_2, y where y is the target and x_1, x_2 are the input variables. The training data is generated such that $x_1, x_2 \in \{1, 2, \cdots, 8\}^2$ and when $x_1 = x_2$, $y = 1$, when $x_1 = 8 - x_2$, $y = -1$. All other cases have no training samples. This example is similar to the XOR problem where there is no independent dependence between the input variables and the target. The only relationship that can be determined from the inputs require the joint relationship between x_1, x_2 and y. This relationship is non-linearly separable. Shown in figure 2.

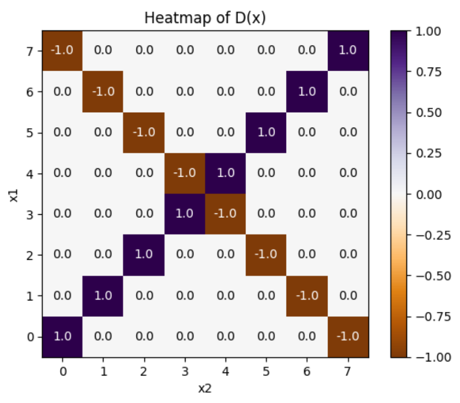


Fig. 2: **Synthetic Synthetic Crossed Dataset**,, $D(\mathbf{x}) = (2P(y=1|\mathbf{x}) - 1)\frac{\rho(\mathbf{x})}{\rho_{\max}}$ where $\rho(\mathbf{x})$ is number of samples at $\mathbf{x}$, $\rho_{\max}$ is the maximum sample number, and $P(y=1 \mid \mathbf{x})$ is the probability of the positive class. The training data is generated as follows: $x_1, x_2 \in \{1, 2, \ldots, 8\}$, $y = 1$ when $x_1 = x_2$, and $y = -1$ when $x_1 = 8 - x_2$. No training samples are generated for all other cases, resulting in zero $D(\mathbf{x})$ values in those regions. Similar to the XOR problem, there is no independent dependence between (x_1, x_2) and (y). The only discernible relationship requires the joint interaction between x_1, x_2, and y.

B.1.6 Synthetic Doughnut

In this case, we have three variables: x_1, x_2, y where y is the target and x_1, x_2 are the input variables. This data is generate from sklearn's `make_circles` function with parameters of `noise=0` and `factor=0.7`. This data was then discretized into 8 bins uniformly. There are no cells containing two different classes of y.This example has neither a correlation between x_1 and y nor x_2 and y. This dataset is non-linearly separable. Shown in figure 3.

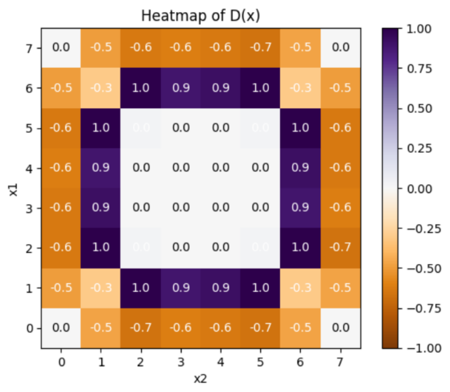


Fig. 3: **Synthetic Doughnut Dataset**, $D(\mathbf{x}) = (2P(y=1|\mathbf{x}) - 1)\frac{\rho(\mathbf{x})}{\rho_{\max}}$ where $\rho(\mathbf{x})$ is number of samples at $\mathbf{x}$, $\rho_{\max}$ is the maximum sample number, and $P(y=1 \mid \mathbf{x})$ is the probability of the positive class. The training data is generated as follows: generate data using sklearn then discretized into 8 levels.

B.2 Real data

The experiments on real data utilize five datasets, covering classification tasks, as summarized in Table 2. Continuous variables, such as age, are binned to convert them into categorical variables. Below is a brief description of each dataset:

- **Adult Income:** This dataset focuses on predicting whether an individual's income exceeds \$50K per year based on demographic and employment-related attributes such as age, education, and occupation.
- **Bank Marketing:** Collected from a Portuguese banking institution, this dataset predicts whether a client will subscribe to a term deposit. Features include job type, marital status, and details of marketing campaigns.
- **Cardiovascular Disease:** This dataset involves predicting the presence or absence of cardiovascular disease using medical features such as age, cholesterol levels, and blood pressure.
- **Sentencing:** This dataset predicts sentencing outcomes based on judicial data such as defendant demographics, charges, and case details.
- **Mushroom:** This dataset classifies mushrooms as edible or poisonous using features like cap shape, odor, and habitat.

Table 2: Summary of real datasets used in the analysis. Complexity represents the input dimension divided by the number of training samples.

Dataset	Input Features	Samples	Total Cells	Max. Levels	Input Dim.	Complexity	Target
Adult Income [1]	13	45222	1.728×10^{10}	41	129	12.9	Income
Bank Marketing [7]	16	45211	4.212×10^{10}	31	107	10.7	Subscription
Cardio. Disease [4]	11	70000	1.769×10^{6}	8	48	4.8	Cardio. Disease
Sentencing [8]	11	266435	4.090×10^{13}	1316	1647	1647	Sentence Type
Mushroom [9]	22	8124	1.218×10^{14}	12	117	11.7	Poisonous

C Types of encoding

In this section, we review common types of encoding used in machine learning an statistics and highlight their similarities and differences.

C.1 Label Encoding

Each unique category is assigned a numeric value.
Example:

```
["Red", "Green", "Blue"] → [0, 1, 2]
```

Introduces an ordinal relationship that may not exist, which may lead to misinterpretation when applied to nominal variables.

C.2 One-Hot Encoding

Creates a new binary column for each category.

Example:

```
["Red", "Green", "Blue"] →
Red   Green  Blue
1      0      0
0      1      0
0      0      1
```

Suitable for nominal variables with no ordinal relationship. Destroys the information stemming from the order, for ordinal variables. Blows up the dimensionality of input space and therefore can be problematic for variables with many levels.

C.3 Ordinal Encoding

Similar to label encoding but relies on a meaningful ordinal relationship between categories.

Example:

```
["Low", "Medium", "High"] → [1, 2, 3]
```

In this paper, we do not make a distinction between “label encoding” and “ordinal encoding” and use them interchangeably.

C.4 Binary Encoding

Converts categories into binary digits and then encodes them into fewer columns.

Example:

```
["A", "B", "C", "D"] →
A: 1 → [0, 1]
B: 2 → [1, 0]
C: 3 → [1, 1]
D: 4 → [1, 0]
```

Reduces dimensionality compared to one-hot encoding. This reduces dimensionality compared to one-hot encoding but introduces distances between levels, which may not reflect their actual order (for ordinal variables) or similarity (for nominal variables). In this paper, we do not consider binary encoding.

C.5 Target Continuous Encoding

Replaces each category with the mean of the target variable (for regression problems) or the frequency of the positive class (for binary classification problems).

Example:

```
Category: "Red", Target: [1, 0, 1, 1] → Red: Mean target = 0.75
```

Since this encoding uses the target variable, mean values must be computed strictly using training data to avoid data leakage. While effective for high-cardinality features, it only captures univariate associations between the variable and the target, potentially destroying multivariate dependency information. In cases of ties, it may lose information by merging levels arbitrarily.

C.6 Target Ordinal Encoding

Replaces each category with an integer number between 0 and $L - 1$, where L is the number of levels, such that the variable maximally correlates with the target.

Similar to "Target Continuous" but replaces the continuous values by discrete integer numbers and breaks ties. Like "Target Continuous" since the code uses the target, it should be built using training data only, to avoid data leakage and because it uses univariate associations between a variable and the target, it may destroy multivariate dependency information. However, unlike "Target Continuous" it does not destroys information by arbitrarily fusing levels because of ties.

C.7 Frequency Encoding

Encodes categories based on their frequency in the dataset.

Example:

```
["Red", "Green", "Blue", "Red"] →
Red: 2, Green: 1, Blue: 1
```

Useful when the frequency of occurrence is important. While not considered in this paper for brevity, it could be interesting as it encodes variables uniformly with a single scalar for both nominal and ordinal variables. However, it destroys order information for ordinal variables and may introduce ties, further losing information.

C.8 Hash Encoding and embeddings

Hash encoding hashes categories into a fixed number of columns using a hash function. Example: Categories are hashed to numerical values based on a hash function. Embeddings, often used in neural networks, can be considered a form of multivariate hash encoding, as they use learned dense vector representations of categories. Advantages: Efficient for high-cardinality data. However, it induces proximity between levels that may introduce unintended biases. Not considered in this paper for brevity.

C.9 Cyclic Encoding

Cyclic Encoding transforms a cyclical categorical variable into two numerical features that preserve the variable's cyclical nature. It typically uses sine and cosine transformations to represent the cyclical pattern.

Example

```
Month_cos = cos(2  (m - 1) / 12)
Month_sin = sin(2  (m - 1) / 12)
where m is a number from 1 to 12 representing January to December.
```

Not considered in this paper, because too specific of cyclical variables.

C.10 Contrast Encoding (Helmert, Sum, Polynomial)

Contrast encoding is similar to one-hot encoding but is used in statistical modeling to represent differences between categories. Commonly applied in linear regression or ANOVA analysis. Example: Helmert encoding contrasts each level with the mean of subsequent levels. Not considered in this paper, as it is most relevant for linear regressions and otherwise similar to one-hot encoding.

C.11 Custom or Manual Encoding

Custom encoding involves manually mapping categories based on domain knowledge. Example:

```
{"Male": 0, "Female": 1, "Other": 2}
```

D Pseudo-code

In this section we provide pseudo-code for the main algorithms implemented. Algorithm 1 describes the computation of CaVaR values, and Algorithm 2 explains the evaluation of variable Availability index. The hyperparameter values used in our experiments are also provided as example values.

[2]Recursive Feature Elimination from sklearn-learn: `https://scikit-learn.org/dev/modules/generated/sklearn.feature_selection.RFE.html`

Algorithm 1 Calculate CaVaR values

```
1: Input:
2:       Datasets $D$
3:       Feature selector $\leftarrow$ RFE$^2$
4:       Models $\leftarrow$ [Neural Net]
5:       Regularizations $\leftarrow [10^{-2}]$
6:       Seeds $\leftarrow 20$
7:       Metric $\leftarrow$ balanced accuracy
8:       NumBins $\leftarrow 8$
9:       TestSize $\leftarrow$ 50 per class
10:      NumPermutation $\leftarrow 20$
11: Output:
12:      CaVaR values (CaVaR$_1$, CaVaR$_2$)
13:      Metric scores (e.g., balanced accuracy)
14: Initialization:
15:      Initialize $\text{CaVaR}_1, \text{CaVaR}_2 \leftarrow 0$
16: for each dataset $D$ do
17:    Discretize continuous variables into NumBins.
18:    Split variables into $V_1$ (top $v/2$ by *FeatureSelector*) and $V_2$ (rest).
19:    Rebalance: $V_1 \leftarrow$ half most, half least informative; $V_2 \leftarrow$ the reverse.
20:    for each seed $S$ in Seeds do
21:       Seed the generator with $S$ and split data using TestSize.
22:       Rebalance $V_1$ and $V_2$ randomly to create sets $V_1'$ and $V_2'$ such that on
          average $V_1'$ contains half of variables from $V_1$ and half from $V_2$ with the
          same being true for $V_2'$.
23:       for each encoder combination ($e_1$, $e_2$), regularization $R$, model $M$ do
24:          Fit $e_1, e_2$, train $M$ on encoded data, and compute Metric.
25:          for each iteration up to NumPermutation do
26:             Create all cells:
27:                $C$ (original cells), $C_1$ ($V_1$ shuffled), $C_2$ ($V_2$ shuffled).
28:             Generate prediction probabilities: $P$ (on $C$), $P_1$ (on $C_1$), $P_2$ (on
                $C_2$).
29:             Update CaVaR values:
30:                $\text{CaVaR}_1 \leftarrow \text{CaVaR}_1 + \frac{\sum |P-P_1|}{\text{cells} \times \texttt{NumPermutation}}$.
31:                $\text{CaVaR}_2 \leftarrow \text{CaVaR}_2 + \frac{\sum |P-P_2|}{\text{cells} \times \texttt{NumPermutation}}$.
32:          end for
33:          Save CaVaR$_1$, CaVaR$_2$, Metric.
34:       end for
35:    end for
36: end for
```

Algorithm 2 Availability Index Reporting

```
1: Input:
2:       CaVaR $\leftarrow$ CaVaR for each encoding (a,b) of a given variable
3: Output:
4:       Availability Index table
5: Initialization:
6:       Initialize difference table CaVaRDiff[initial,final] $\leftarrow$ 0 for each pair of
   encodings
7: for each pair of encodings (initial, final) do
8:     CaVaRDiff[inital,final]) $\leftarrow$ CaVar(inital,inital) - CaVaR(inital,final).
9: end for
10: for each initial encoding do
11:     Avail(initial) $\leftarrow$ mean(CaVaRDiff[inital,final] for all final $\neq$ initial.)
12: end for
```

E Two Dimensional Experiment

Here, we present a plots of the generalization of a two layer neural network for the Synthetic Simple case B. We show this for three encoding types: One-Hot, Target Continuous and the Natural Ordinal case. The top row presented is the identical codings and the bottom row uses mixed encodings. The values of these heatmaps represent the predicted probabilities rescaled to between -1 and 1.

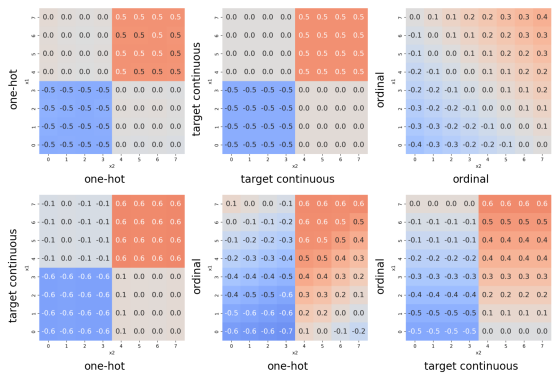
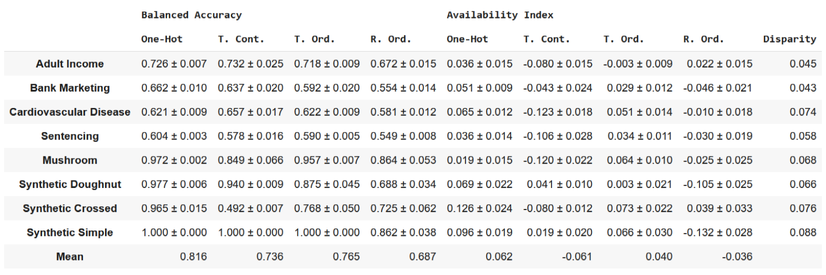


Fig. 4: Classifier predictions, $2\hat{P}_M(y = 1|\mathbf{x}) - 1$, are presented for *test data* (64 cells as shown) trained on the Synthetic Simple dataset B (where $x_1 = x_2$) using a neural network. Each heatmap corresponds to a different combination of encoding of the two variables x_1 and x_2 which are the only differences between them. As x_1 is identical to x_2, there should be no difference in a prediction should we swap a value of x_1 for a value of x_2. As the only training data lies on the diagonal $x_1 = x_2$, all values off-diagonal show the out-of-domain predictions. The top row of the heatmaps show encoding pairs that encode x_1 and x_2 identically. We see that this results in heatmaps that are symmetric about $x_1 = x_2$ or any $\hat{P}_M(y = 1|x_1 = i, x_2 = j) = \hat{P}_M(y = 1|x_1 = j, x_2 = i)$. This shows that one variable is not relied upon more than another. The second row shows pairs that encode x_1 and x_2 differently. In all three heatmaps, we see an asymmetry about $x_1 = x_2$ or that x_1 and x_2 are used differently by the model. We also see a vertical separation with more negative values (blue) on the left and more positive values (red) on the right. This is an indication that x_2 is being relied on more in these cases. Varying x_1 makes less difference in the prediction than does varying x_2. In the last two heatmaps, the prediction is almost fully determined by x_2.

F Extended Experimental Results

Here we show the amount of error present in the tests shown in 1. The error shown represents the standard deviation divided by the square root of the number of seeds each test was run on.



	Balanced Accuracy				Availability Index				
	One-Hot	T. Cont.	T. Ord.	R. Ord.	One-Hot	T. Cont.	T. Ord.	R. Ord.	Disparity
Adult Income	0.726 ± 0.007	0.732 ± 0.025	0.718 ± 0.009	0.672 ± 0.015	0.036 ± 0.015	-0.080 ± 0.015	-0.003 ± 0.009	0.022 ± 0.015	0.045
Bank Marketing	0.662 ± 0.010	0.637 ± 0.020	0.592 ± 0.020	0.554 ± 0.014	0.051 ± 0.009	-0.043 ± 0.024	0.029 ± 0.012	-0.046 ± 0.021	0.043
Cardiovascular Disease	0.621 ± 0.009	0.657 ± 0.017	0.622 ± 0.009	0.581 ± 0.012	0.065 ± 0.012	-0.123 ± 0.018	0.051 ± 0.014	-0.010 ± 0.018	0.074
Sentencing	0.604 ± 0.003	0.578 ± 0.016	0.590 ± 0.005	0.549 ± 0.008	0.036 ± 0.014	-0.106 ± 0.028	0.034 ± 0.011	-0.030 ± 0.019	0.058
Mushroom	0.972 ± 0.002	0.849 ± 0.066	0.957 ± 0.007	0.864 ± 0.053	0.019 ± 0.015	-0.120 ± 0.022	0.064 ± 0.010	-0.025 ± 0.025	0.068
Synthetic Doughnut	0.977 ± 0.006	0.940 ± 0.009	0.875 ± 0.045	0.688 ± 0.034	0.069 ± 0.022	0.041 ± 0.010	0.003 ± 0.021	-0.105 ± 0.025	0.066
Synthetic Crossed	0.965 ± 0.015	0.492 ± 0.007	0.768 ± 0.050	0.725 ± 0.062	0.126 ± 0.024	-0.080 ± 0.012	0.073 ± 0.022	0.039 ± 0.033	0.076
Synthetic Simple	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	0.862 ± 0.038	0.096 ± 0.019	0.019 ± 0.020	0.066 ± 0.030	-0.132 ± 0.028	0.088
Mean	0.816	0.736	0.765	0.687	0.062	-0.061	0.040	-0.036	

Table 3: Error for Tests in shown in Table 1