A THEOREMS

A.1 PROOF OF THEOREM 3.3

Proof. The proof is by induction on the number n of variables in Π. We recall that $\{x_1, \ldots, x_D\}$ is the set of variables and that we assumed, w.l.o.g., $\lambda(x_i) = i$.

For the base case $n = 0$, Π does not contain variables and, since Π is satisfiable, for each constraint $\sum_k w_k x_k + b \trianglerighteq 0$ in Π, (i) each $w_k = 0$, (ii) $b \trianglerighteq 0$, and (iii) each sample satisfies Π.

Assume that $n > 1$ variables appear in Π. Let x_k be the variable with the highest $\lambda(x_k)$ value occurring in Π.

Then, $x_D, x_{D-1}, \ldots, x_{k+1}$ do not occur in Π, $\Pi_D = \Pi_{D-1} = \ldots = \Pi_k = \Pi$, and

$$\Pi_{k-1} = \Pi_k \setminus (\Pi_k^- \cup \Pi_k^+) \cup \{red_k(\phi^1, \phi^2) \mid \phi^1 \in \Pi_k^-, \phi^2 \in \Pi_k^+\}. \tag{5}$$

Consider an arbitrary sample $\tilde{x}$.

In Π_{k-1}, by construction, less than n variables appear. Thus, for the inductive hypothesis, assume $\text{CL}(\tilde{x})$ satisfies Π_{k-1}. Proving that $\text{CL}(\tilde{x})$ satisfies Π_k is equivalent to proving that $\text{CL}(\tilde{x})$ satisfies both Π_k^+ and Π_k^-, since we know from eq. (5) that $\Pi_k \subseteq \Pi_{k-1} \cup \Pi_k^- \cup \Pi_k^+$.

The proof is in two steps. We first prove by contradiction that either (i) $lb_k < ub_k$ or (ii) $lb_k = ub_k$ and there exist two constraints $\phi_1 \in \Pi_k^-$ and $\phi_2 \in \Pi_k^+$ such that: $ub_k = \varepsilon_k^{\phi^1}(\text{CL}(\tilde{x}))$ and $lb_k = \varepsilon_k^{\phi^2}(\text{CL}(\tilde{x}))$, and both ϕ^1 and ϕ^2 are not strict inequalities. Then, in the second step, we prove that $\text{CL}(\tilde{x})$ satisfies each constraint $\phi \in \Pi_k^+ \cup \Pi_k^-$.

First step. Assume that either (i) $lb_k > ub_k$ or (ii) $lb_k = ub_k$ and at least one between ϕ^1 and ϕ^2 is a strict inequality. If $lb_k > ub_k$ or $lb_k = ub_k$, then $lb_k \neq -\infty$ and $ub_k \neq +\infty$ and thus both Π_k^- and Π_k^+ are not empty. Let ϕ^1 (resp. ϕ^2) be the constraint in Π_k^- (resp. Π_k^+) such that $ub_k = \varepsilon_k^{\phi^1}(\text{CL}(\tilde{x}))$ (resp. $lb_k = \varepsilon_k^{\phi^2}(\text{CL}(\tilde{x}))$). Such constraints ϕ_1 and ϕ_2 exist since Π is finite. Then, by definition, $red_k(\phi^1, \phi^2)$ is equivalent to $\varepsilon_k^{\phi^1} - \varepsilon_k^{\phi^2} \geq 0$ if both ϕ^1 and ϕ^2 are non-strict inequalities, and to $\varepsilon_k^{\phi^1} - \varepsilon_k^{\phi^2} > 0$ if at least one between ϕ^1 and ϕ^2 is a strict inequality.

We know that $red_k(\phi^1, \phi^2) \in \Pi_{k-1}$ and that, by the inductive hypothesis, $\text{CL}(\tilde{x})$ satisfies Π_{k-1}. Thus, $\text{CL}(\tilde{x})$ satisfies $red_k(\phi^1, \phi^2)$, which (taken together with the definition of $red_k(\phi^1, \phi^2)$ above) implies that $\varepsilon_k^{\phi^1}(\text{CL}(\tilde{x})) - \varepsilon_k^{\phi^2}(\text{CL}(\tilde{x})) \geq 0$ if both ϕ^1 and ϕ^2 are non-strict inequalities, and that $\varepsilon_k^{\phi^1}(\text{CL}(\tilde{x})) - \varepsilon_k^{\phi^2}(\text{CL}(\tilde{x})) > 0$ if at least one between ϕ^1 and ϕ^2 is a strict inequality. However, by our assumption, $ub_k = \varepsilon_k^{\phi^1}(\text{CL}(\tilde{x}))$ and $lb_k = \varepsilon_k^{\phi^2}(\text{CL}(\tilde{x}))$. Thus, we have that $ub_k \geq lb_k$ if both ϕ^1 and ϕ^2 are non-strict inequalities, and that $ub_k > lb_k$ if at least one between ϕ^1 and ϕ^2 is a strict inequality, deriving a contradiction.

Second step. We now prove that $\text{CL}(\tilde{x})$ satisfies each constraint $\phi \in \Pi_k^+$. The statement holds since

1. if ϕ is a non-strict inequality, by definition, we have that $\text{CL}(\tilde{x})_k \geq lb_k \geq \varepsilon_k^{\phi}(\text{CL}(\tilde{x}))$, and
2. if ϕ is a strict inequality, we have two cases. In the first one, $lb_k = \varepsilon_k^{\phi}(\text{CL}(\tilde{x}))$ and, since $lb_k < ub_k$, it is possible to choose an $\epsilon > 0$ such that $\text{CL}(\tilde{x})_k = lb_k + \epsilon < ub_k$, and thus $\text{CL}(\tilde{x})_k > lb_k = \varepsilon_k^{\phi}(\text{CL}(\tilde{x}))$. In the second case, $lb_k > \varepsilon_k^{\phi}(\text{CL}(\tilde{x}))$ and then $\text{CL}(\tilde{x})_k \geq lb_k > \varepsilon_k^{\phi}(\text{CL}(\tilde{x}))$.

Analogously, $\text{CL}(\tilde{x})$ satisfies each constraint $\phi \in \Pi_k^-$.

□

A.2 PROOF OF THEOREM 3.5

Let $\tilde{x}$ be a sample satisfying the constraint in Π. We recall that, w.l.o.g., we assume $\lambda(x_i) = i$. We will prove by contradiction that if $\tilde{x}$ satisfies Π, then $\text{CL}(\tilde{x}) = \tilde{x}$.

Assume $\text{CL}(\tilde{x}) \neq \tilde{x}$. Let i be the lowest index such that $\text{CL}(\tilde{x})_i \neq \tilde{x}_i$. Then, $\text{CL}(\tilde{x})_i = \min^i(\max^i(\tilde{x}_i, lb_i), ub_i) \neq \tilde{x}_i$, and thus either $\tilde{x}_i < lb_i$ or $\tilde{x}_i > ub_i$, both cases being impossible given

1. the definitions of lb_i and ub_i,
2. the hypothesis that $\tilde{x}$ satisfies the constraints in Π, and
3. the fact that all the constraints in $\Pi_i^- \cup \Pi_i^+$ are entailed by Π and thus satisfied by $\tilde{x}$.

A.3 PROOF OF THEOREM 3.6

We prove the two statements of the theorem in separate lemmas, after a first introductory lemma.

Lemma A.1. *In the hypotheses of Theorem 3.6, CL$^{\geq}(\tilde{x})$ is optimal with respect to $\Pi^{\geq}$.*

Proof. Let $\tilde{x}$ be a sample. We recall that, w.l.o.g., we assume $\lambda(x_i) = i$.

We show that for every $i = 1, \ldots, D$, there does not exist another sample $\tilde{x}'$ satisfying $\Pi^{\geq}$ such that for each $1 \leq j < i$, $\tilde{x}'_j = \text{CL}^{\geq}(\tilde{x})_j$ and $|\tilde{x}'_i - \tilde{x}_i| < |\text{CL}^{\geq}(\tilde{x})_i - \tilde{x}_i|$. The proof is by induction on i.

For the base case ($i = 1$), we know that in $\Pi_1^{\geq}$ the only variable that can appear is x_1, thus we have four cases:

1. $\Pi_1^{\geq}$ is either empty or contains a constraint $c \geq 0$ which is always satisfied since Π (and thus $\Pi^{\geq}$) is satisfiable. In this case, $\text{CL}^{\geq}(\tilde{x})_1 = \tilde{x}_1$ and the statement trivially holds;
2. $\Pi_1^{\geq}$ is equivalent to a single constraint $\{x_1 + a \geq 0\}$ in which case if $\tilde{x}_1 \geq -a$ then $\text{CL}^{\geq}(\tilde{x})_1 = \tilde{x}_1$ otherwise $\text{CL}^{\geq}(\tilde{x})_1 = -a$ and the statement trivially holds;
3. $\Pi_1^{\geq}$ is equivalent to a single constraint $\{-x_1 + b \geq 0\}$ in which case if $\tilde{x}_1 \leq b$ then $\text{CL}^{\geq}(\tilde{x})_1 = \tilde{x}_1$ otherwise $\text{CL}^{\geq}(\tilde{x})_1 = b$ and the statement trivially holds;
4. $\Pi_1^{\geq}$ is equivalent to a pair of constraints $\{x_1 + a \geq 0, -x_1 + b \geq 0\}$. Since Π is satisfiable, $a + b \geq 0$. Then, if $\tilde{x}_1 < -a$ then $\text{CL}^{\geq}(\tilde{x})_1 = -a$, if $-a \leq \tilde{x}_1 \leq b$ then $\text{CL}^{\geq}(\tilde{x})_1 = \tilde{x}_1$, and if $\tilde{x}_1 > b$ then $\text{CL}^{\geq}(\tilde{x})_1 = b$. For each of the three cases, the statement trivially holds.

Assume $i = j + 1 > 1$. Assume by contradiction that there exists another sample $\tilde{x}'$ satisfying $\Pi^{\geq}$ such that for each $1 \leq j < i$, $\tilde{x}'_j = \text{CL}^{\geq}(\tilde{x})_j$ and $|\tilde{x}'_i - \tilde{x}_i| < |\text{CL}^{\geq}(\tilde{x})_i - \tilde{x}_i|$. By construction of $\Pi_i^{\geq}$, we know that only $x_1, \ldots, x_i$ appear in $\Pi_i^{\geq}$. If we substitute each variable x_j with $j < i$ with $\text{CL}^{\geq}(\tilde{x})_j$ in $\Pi_i^{\geq}$, then (i) in the resulting set of constraints the only variable is x_i, (ii) we are back to the base case, and (iii) the statement follows.

□

Lemma A.2. *In the hypotheses of Theorem 3.6, CL$(\tilde{x})$ is optimal if CL$(\tilde{x}) =$ CL$^{\geq}(\tilde{x})$.*

Proof. The proof is a direct consequence of the facts that (i) $\text{CL}(\tilde{x})$ satisfies the constraints in Π, (ii) $\text{CL}(\tilde{x}) = \text{CL}^{\geq}(\tilde{x})$, (iii) $\text{CL}^{\geq}(\tilde{x})$ is optimal wrt $\Pi^{\geq}$ (from Lemma A.1), and (iv) the samples satisfying Π are a subset of the samples satisfying $\Pi^{\geq}$. □

Lemma A.3. *In the hypotheses of Theorem 3.6, CL$(\tilde{x})$ tends to CL$^{\geq}(\tilde{x})$ as the ϵ values used to compute CL$(\tilde{x})$ tend to 0.*

Proof. Let $\epsilon_1, \ldots, \epsilon_k$ ($k \geq 0$) be the ϵ values (assumed, w.l.o.g., to be distinct) in $\text{CL}(\tilde{x})$. If $k = 0$ then $\text{CL}(\tilde{x}) = \text{CL}^{\geq}(\tilde{x})$, and the statement trivially holds.

Consider $\text{CL}(\tilde{x})_i$, $i = 1, \ldots, D$, and substitute ϵ_j with a newly introduced variable v_j. $\text{CL}(\tilde{x})_i$ is a composition of continuous functions in the newly introduced variables and, thus, it is continuous. For an arbitrary continuous function $f : \mathbb{R}^k \to \mathbb{R}$ we know that $\lim_{x \to x_0} f(x) = f(x_0)$. Thus,

$$\lim_{[v_1,\ldots,v_k] \to [0,\ldots,0]} \text{CL}(\tilde{x})_i = \text{CL}^{\geq}(\tilde{x})_i,$$

and hence

$$\lim_{[v_1,\ldots,v_k] \to [0,\ldots,0]} \text{CL}(\tilde{x}) = \text{CL}^{\geq}(\tilde{x}).$$

□

B Experimental analysis settings

B.1 Models

In our experimental analysis, we use five base models:

- **WGAN** (Arjovsky et al., 2017) is a GAN model trained with Wasserstein loss in a typical generator discriminator GAN-based architecture. In our implementation, WGAN uses a MinMax transformer for the continuous features and one-hot encoding for categorical ones. It has not been designed specifically for tabular data.
- **TableGAN** (Park et al., 2018) is among the first GAN-based approaches proposed for tabular data generation. In addition to the typical generator and discriminator architecture for GANs, the authors proposed adding a classifier trained to learn the relationship between the labels and the other features. The classifier ensures a higher number of semantically correct produced records. TableGAN uses a MinMax transformer for the features.
- **CTGAN** (Xu et al., 2019) uses a conditional generator and training-by-sampling strategy in a generator-discriminator GAN architecture to model tabular data. The conditional generator generates synthetic rows conditioned on one of the discrete columns. The training-by-sampling ensures that the data are sampled according to the log-frequency of each category. Both help to better model the imbalanced categorical columns. CTGAN transforms discrete features using one-hot encoding and a mode-based normalization for continuous features. A variational Gaussian mixture model (Camino et al., 2018) is used to estimate the number of modes and fit a Gaussian mixture. For each continuous value, a mode is sampled based on probability densities, and its mean and standard deviation are used to normalize the value.
- **TVAE** (Xu et al., 2019) was proposed as a variation of the standard Variational AutoEncoder to handle tabular data. It uses the same transformations of data as CTGAN and trains the encoder-decoder architecture using evidence lower-bound (ELBO) loss.
- **GOGGLE** (Liu et al., 2022) is a graph-based approach to learning the relational structure of the data as well as functional relationships (dependencies between features). The relational structure of the data is learned by building a graph where nodes are variables and edges indicate dependencies between them. The functional dependencies are learned through a message passing neural network (MPNN). The generative model generates each variable considering its surrounding neighborhood.

B.2 Datasets

We use 6 real-world datasets covering both classification and regression tasks. An overview of these datasets' statistics can be found in Table 5. For the selection, we focused on datasets with at least three feature relationship constraints that either were provided with the description of the datasets or we could derive with our domain expertise. The selected datasets are listed below:

Table 5: Datasets statistics.

Dataset	# Train	# Val	# Test	# Features	# Cat.	# Cont.	Task (# classes)
URL	7K	2K	2K	64	20	44	Binary classification
WiDS	22K	6K	7K	109	9	100	Binary classification
LCLD	494K	199K	431K	29	8	21	Binary classification
Heloc	8K	2K	0.2K	24	8	16	Binary classification
FSP	2K	0.2K	0.2K	28	0	28	Multi-class class. (7)
News	31K	7K	1K	60	14	46	Regression

- URL[3] (Hannousse & Yahiouche, 2021) is used to perform webpage phishing detection with features describing statistical properties of the URL itself as well as the content of the page.
- WiDS[4] is used to predict if a patient is diagnosed with a particular type of diabetes named Diabetes Mellitus, using data from the first 24 hours of intensive care.
- LCLD[5] is used to predict whether the debt lent is unlikely to be collected. In particular, we use the feature-engineered dataset from Simonetto et al. (2022), inspired from the LendingClub loan data. The dataset captures features related to the loan as well as client history.
- FICO's Home Equity Line of Credit dataset (Heloc[6]) from the FICO xML Challenge is used to predict whether customers will repay their credit lines within 2 years. Similarly to LCLD, the dataset has features related to the credit line and the client's history.
- FSP[7](Buscema, 1998) is used to predict 7 types of surface defects in stainless steel plates. The features approximately capture the geometric shape of the defect and its outline.
- News[8] (Fernandes et al., 2015) is used to predict the number of times a news article will be shared on social networks. The features capture properties of the text, as well as the publishing time.

For URL, WiDS, and LCLD, we used the train-val-test splits provided by Simonetto et al. (2022). For Heloc we used the train-test split of 80-20 and 20% of the training set was later split for validation. Finally, for FSP and News we split the data into 80-10-10% sets, and for the former (which is a multiclass classification dataset) we preserved the class imbalance.

B.3 Constraints Datasheet

Here we give an overview of the structure of our constraints. To this end, we define F to be the number of features appearing in at least one constraint, and we remind that D is the total number of features. Then, given a constraint ϕ, we define F_ϕ^+ (resp. F_ϕ^-) to be the number of features appearing positively (resp. negatively) in ϕ, and $F_\phi = F_\phi^+ + F_\phi^-$ to be the number of features appearing in ϕ.

As we can see from Table 6, the constraints characteristics greatly vary from one dataset to the other. Indeed, we can have from 4 to 31 constraints annotated for each dataset, with as little as 15% of the variables appearing in at least a constraint in News to as much as 56.88% in WiDS. Further, we can see that for almost all datasets, the average number of features appearing per constraint is 2.00, except for URL, which has a constraint where as many as 17 different features appear, and LCLD where we have 2 constraints where a single variable appears.

B.4 Evaluation protocol

For evaluating the utility of the DGM/C-DGM models presented in our paper, we followed closely the protocol from Kim et al. (2023) which we also reproduce here.

[3]Link to dataset: https://data.mendeley.com/datasets/c2gw7fy2j4/2
[4]Link to dataset: https://www.kaggle.com/competitions/widsdatathon2021
[5]Link to dataset: https://figshare.com/s/84ae808ce6999fafd192
[6]Link to the dataset: https://huggingface.co/datasets/mstz/heloc
[7]Link to dataset: https://www.kaggle.com/datasets/uciml/faulty-steel-plates
[8]Link to dataset: https://archive.ics.uci.edu/dataset/332/online+news+popularity

Table 6: Constraints statistics.

Dataset	# Constr.	F / D	Avg. F_ϕ	Avg. F_ϕ^+	Avg. F_ϕ^-
URL	8	24 / 64	4.25	1.00	3.25
WiDS	31	62 / 109	2.00	1.00	1.00
LCLD	4	5 / 29	1.50	0.75	0.75
Heloc	7	10 / 24	2.00	1.00	1.00
FSP	4	7 / 28	2.00	1.00	1.00
News	5	9 / 60	2.00	1.00	1.00

1. First, we generate a synthetic dataset, split into training, validation and test partitions using the same proportions as the real dataset.
2. Then, we perform a hyperparameter search using the synthetic training data partition to train different classifiers/regressors.

   For the binary classification datasets (i.e., URL, WiDS, LCLD, and Heloc) we use: Decision Tree (Wu et al., 2008), AdaBoost (Schapire, 2013), Multi-layer Perceptron (MLP) (Haykin, 1994), Random Forest (Ho, 1995), XGBoost (Chen & Guestrin, 2016), and Logistic Regression (Cox, 1958) classifiers. For multi-class classification datasets, (i.e., FSP) we use Decision Tree, MLP, Random Forest, and XGBoost classifiers. For the regression dataset, (i.e., News) we use MLP, XGBoost, and Random Forest regressors and linear regression. For all the classifiers and regressors above, we considered the same hyperparameter settings as those from Table 26 of (Kim et al., 2023) and picked the best hyperparameter configuration using the real validation set according to the F1-score.
3. Finally, we tested the selected best models on the real test set and averaged the results across all the classifiers/regressors to get performance measurements for the DGM/C-DGM predictions according to three different metrics: F1-score, weighted F1-score, and Area Under the ROC Curve.

The above procedure was repeated 5 times for each DGM/C-DGM model, and the results were averaged separately for each of the metrics.

For evaluating the DGM/C-DGM models in terms of detection, we slightly adapted the procedure presented by Kim et al. (2023). We first created the training, validation, and test sets by concatenating real and synthetic data, including their targets as usual features, and adding a new target column that specifies whether the data is real or not. By construction, these datasets are binary classification datasets and, thus, are suitable for the hyperparameter search procedure presented in Kim et al. (2023) using the 6 different binary classifiers mentioned above. We proceeded to pick the best model using the newly-created validation set, and then we obtained the final detection performance on the newly-created test data (which combines the real and the synthetic data).

B.5 Hyperparameter search

Prior to experimenting with our C-DGM methods, we conducted an extensive hyperparameter search to reveal the best configurations. We always chose the best settings according to the utility performance measured either by the average over the F1-score, weighted F1-score, and Area Under the ROC Curve for the binary and multi-class classification datasets or by the Mean Absolute Error for the regression dataset.

Initial phase. For GOGGLE, we used the same optimiser and learning rate set as Liu et al. (2022). Specifically, we used Adam (Kingma & Ba, 2015) with a set of 5 different learning rates: $\{1 \times 10^{-3}, 5 \times 10^{-3}, 1 \times 10^{-2}\}$. For TVAE, we used Adam with a set of 5 different learning rates: $\{5 \times 10^{-6}, 1 \times 10^{-5}, 1 \times 10^{-4}, 2 \times 10^{-4}, 1 \times 10^{-3}\}$. And for each of the other DGM models, we used three different optimizers, Adam, RMSProp (Hinton, 2014), SGD (Ruder, 2016), each with a different set of learning rates:

- for WGAN $\{1\times10^{-4}, 1\times10^{-3}\}$, $\{5\times10^{-5}, 1\times10^{-4}, 1\times10^{-3}\}$, and $\{1\times10^{-4}, 1\times10^{-3}\}$, respectively.

- for TableGAN, $\{5 \times 10^{-5}, 1 \times 10^{-4}, 2 \times 10^{-4}, 1 \times 10^{-3}\}$, $\{1 \times 10^{-4}, 2 \times 10^{-4}, 1 \times 10^{-3}\}$ and $\{1 \times 10^{-4}, 1 \times 10^{-3}\}$, respectively.
- for CTGAN, $\{5 \times 10^{-5}, 1 \times 10^{-4}, 2 \times 10^{-4}\}$, $\{1 \times 10^{-4}, 2 \times 10^{-4}, 1 \times 10^{-3}\}$ and $\{1 \times 10^{-4}, 1 \times 10^{-3}\}$, respectively.

Then, for each of the above optimizer-learning rate pairs, we tested three different batch sizes, depending on the DGM model: $\{64, 128, 256\}$ for WGAN, $\{128, 256, 512\}$ for TableGAN, $\{70, 280, 500\}$ for CTGAN and TVAE, and $\{64, 128\}$ for GOGGLE. The batch sizes for CTGAN are multiples of 10, to allow for using the CTGAN's recommended PAC (Lin et al., 2018) value of 10, among other values.

Further search. The initial phase of hyperparameter search allowed us to narrow down the space of configurations and focus on further investigating the most promising one. For the second phase, we varied the following parameters for each DGM, keeping fixed the rest from the best model of the initial phase:

- for WGAN we varied initially PAC values to 4, 8, and 16 and then discriminator iterations to 1,2, and 10.
- for TableGAN we varied separately *i)* generator layers dimensions to 128 from 100 initially and *ii)* embedding dimensions by doubling the value.
- for CTGAN we varied the value of PAC within $\{1, 5, 15\}$.
- for TVAE we varied the multiplier of the loss within $\{1, 2, 3, 4\}$.

The best hyperparameters settings are presented in Table 7. We used these configurations for the experiments presented in our paper. The same hyperparameters were then used for C-DGMs. Furthermore, for C-DGMs we reported the variable ordering hyperparameter that needs to be given to our CL. We give a full description of the orderings and of the impact they have on the performance in Appendix C.3.

C RESULTS

C.1 BACKGROUND KNOWLEDGE ALIGNMENT

Besides constraints violation rate (CVR) presented in the main text, we measure two additional metrics: (i) *constraints violation coverage* (CVC), which, given a set of samples $\mathcal{S}$ and Π, represents the percentage of constraints in Π that have been violated at least once by any of the samples in $\mathcal{S}$, and (ii) samplewise *constraints violation coverage* (sCVC), which represents the average over the samples in $\mathcal{S}$ of the percentage of the constraints violated by each sample.

Table 8 shows that for 5 out of 6 datasets, all the samples generated by unconstrained DGMs violate at least 50% of the constraints, with CVC reaching up to 100% for WiDS, FSPand News. This entails that the models actually struggle with the majority of the constraints specified, and that the problem cannot be solved by just fixing a very small subset of the available constraints. Meanwhile, all our C-DGMs guarantee that not a single constraint is violated by any of the samples.

Regarding sCVC, the results in Table 9 show that for all models and datasets, on average each sample can violate up to 29.8% of the constraints. As a reminder, even a single constraint violated indicates an unfeasible example in a real setting. The unconstrained DGM models learn different representations that produce different rankings for sCVC. For example, WGAN violates on average the most constraints per samples in LCLD. But, for TableGAN, CTGAN, TVAE and GOGGLE, LCLD is among the datasets with the lowest sCVC. Again, all our C-DGMs ensure that sCVC is 0.0, meaning not a single constraint is violated for all the samples.

We complement our quantitative analysis of background knowledge alignment for the generated synthetic samples with some visualizations. In particular, we consider three different constraints where only two variables appear, and then for each one of them, we create four two-dimensional scatter plots with the variables appearing in the constraint on the axes. The first scatter plot represents the real data, while the other three represent the samples generated by each of the DGMs and the C-DGMs. We now consider them one by one.

Table 7: Best hyperparameter settings used for DGMs/C-DGMs in our experiments.

Model/Dataset	Hyperparameter	URL	WiDS	LCLD	Heloc	FSP	News
WGAN	Batch size	64	128	128	64	128	128
	Optimiser	Adam	RMSProp	SGD	Adam	RMSProp	RMSProp
	Learning rate	0.0001	0.001	0.001	0.0001	0.001	0.001
	Epochs	300	80	30	200	20	50
	Discriminator iters	10	10	5	10	10	10
	Ordering	Rnd	KDE	KDE	KDE	KDE	KDE
TableGAN	Batch size	128	128	128	128	128	128
	Optimiser	Adam	RMSProp	Adam	Adam	Adam	RMSProp
	Learning rate	0.001	0.001	0.001	0.001	0.001	0.001
	Epochs	300	50	25	200	200	50
	Ordering	Corr	Corr	KDE	Corr	KDE	KDE
CTGAN	Batch size	500	500	500	500	500	500
	Optimiser	Adam	RMSProp	Adam	Adam	Adam	Adam
	Learning rate	0.0002	0.0005	0.0002	0.0002	0.0002	0.0002
	Epochs	150	50	20	500	300	100
	Ordering	KDE	Corr	Rnd	Corr	Corr	Corr
TVAE	Batch size	70	70	500	500	70	70
	Optimiser	Adam	Adam	Adam	Adam	Adam	Adam
	Learning rate	0.0002	0.000005	0.00001	0.000005	0.00001	0.00001
	Epochs	150	50	40	150	500	50
	Loss factor	2	2	4	2	1	3
	Ordering	KDE	KDE	Corr	Corr	Corr	Corr
GOGGLE	Batch size	128	128	128	64	128	64
	Optimiser	Adam	Adam	Adam	Adam	Adam	Adam
	Learning rate	0.005	0.01	0.001	0.001	0.005	0.001
	Epochs	1000	200	200	1000	1000	250
	Patience	50	50	50	50	50	50
	Ordering	Random	Corr	Corr	Random	Corr	KDE

Table 8: Constraints violation coverage (CVC) for all datasets and models under study.

Model/Dataset	URL	WIDS	LCLD	HELOC	Faults	News
WGAN	11.1 ± 1.6	100.0 ± 0.0	75.0 ± 0.0	100.0 ± 0	75.0 ± 0.0	84.8 ± 8.7
TableGAN	24.5 ± 3.7	100.0 ± 0.0	50.0 ± 0.0	100.0 ± 0	75.0 ± 0.0	100.0 ± 0.0
CTGAN	16.5 ± 3.8	100.0 ± 0.0	50.0 ± 0.0	99.4 ± 1.3	75.0 ± 0.0	100.0 ± 0.0
TVAE	12.5 ± 0.0	100 ± 0.0	50.0 ± 0.0	100.0 ± 0.0	75.0 ± 0.0	100.0 ± 0.0
GOGGLE	17.5 ± 6.9	99.2 ± 1.7	50.0 ± 0.0	100.0 ± 0.0	75.0 ± 0.0	100.0 ± 0.0
All C-models	$\mathbf{0.0 \pm 0.0}$	$\mathbf{0.0 \pm 0.0}$	$\mathbf{0.0 \pm 0.0}$	$\mathbf{0.0 \pm 0.0}$	$\mathbf{0.0 \pm 0.0}$	$\mathbf{0.0 \pm 0.0}$

1. In Figure 4, we consider again the constraint $\mathit{MaxHemoglobinLevel} - \mathit{MinHemoglobinLevel} \geq 0$ appearing in WiDS dataset from Section 4.2 in the main text. We visualize the outputs for the remaining models WGAN, CTGAN, TVAE and GOGGLE and their constrained counterparts C-WGAN, C-CTGAN, C-TVAE and C-GOGGLE.

2. In Figure 5, we illustrate the behavior of the real and synthetic samples with respect to a constraint from the Heloc dataset which requires that the number of trades that have been insolvent for 60 days must be greater than the number of trades that have been insolvent for 90 days, i.e., $\mathit{NumInsolventTradesGreaterThan60Days} \geq \mathit{NumInsolventTradesGreaterThan90Days}$.

3. In Figure 6, we illustrate the behavior of the real and synthetic samples with respect to a constraint from the FSP dataset that requires $\mathit{X\_Maximum} \geq \mathit{X\_minimum}$, where $\mathit{X\_minimum}$ and $\mathit{X\_maximum}$ refer to the value for the X coordinate in images captured of steel plates.

Table 9: Samplewise constraints violation coverage (sCVC) for all models and datasets under study.

Model/Dataset	URL	WIDS	LCLD	HELOC	Faults	News
WGAN	1.4 ± 0.2	21.2 ± 1.3	29.8 ± 0.2	10.5 ± 2.5	23.3 ± 4.3	12.1 ± 2.4
TableGAN	0.6 ± 0.2	14.4 ± 2.8	1.5 ± 0.2	9.0 ± 3.3	22.9 ± 3.2	24.1 ± 3.3
CTGAN	0.4 ± 0.3	21.6 ± 0.5	3.0 ± 0.7	7.6 ± 0.3	24.4 ± 2.4	15.8 ± 3.9
TVAE	0.4 ± 0.1	21.0 ± 0.2	1.0 ± 0.1	9.4 ± 0.2	21.5 ± 1.5	14.3 ± 1.1
GOGGLE	0.8 ± 0.9	10.6 ± 2.4	3.3 ± 0.7	17.2 ± 4.9	18.8 ± 5.3	10.7 ± 1.6
All C-models	$\mathbf{0.0 \pm 0.0}$	$\mathbf{0.0 \pm 0.0}$	$\mathbf{0.0 \pm 0.0}$	$\mathbf{0.0 \pm 0.0}$	$\mathbf{0.0 \pm 0.0}$	$\mathbf{0.0 \pm 0.0}$

The Figures visually demonstrate that the models that incorporate our constrained layer (C-WGAN, C-CTGAN, C-TableGAN, C-TVAE and C-GOGGLE) are guaranteed to produce outputs only within the feasible regions of the real data.

Finally, to further investigate the properties of the generated data by C-DGMs, we study the impact of constraint reparation on the boundary population for the three cases considered above. We defined a band around the boundary whose width w we set to be proportional to the range of the values of the considered features in the real dataset. As in all the considered constraints only two features appear, we set $w = \sqrt{(r_1 p)^2 + (r_2 p)^2}$, where r_1 (resp. r_2) represents the range of the first (resp. second) feature, while p represents the proportion of the range of values each feature can assume. If $p = 1$, then the width is equal to the diagonal of the rectangle defined by the two points with minimum and maximum coordinates.

The results in Table 10 show the percentage of generated samples that lay on the boundary when $p = \{1\%, 5\%, 10\%\}$ for the real dataset (last row), individual DGMs and C-DGMs, as well as their averages (third and second to last row, respectively). Notice that C-DGMs populate the boundary at a much closer rate to the real data than their unconstrained counterparts. Indeed the maximum difference between the percentage of real data points laying at the boundary and the average number of samples generated by C-DGMs is equal to 14.7% (for dataset WiDS and $p = 1\%$). On the contrary, the maximum difference between the number of real data points laying at the boundary and the average number of samples generated by DGMs is equal to 51.5% (for dataset FSP and $p = 1\%$)

C.2 Feature Distribution Analysis

Following Kotelnikov et al. (2023) and Zhao et al. (2021) we perform a comparative analysis of the distributions of data generated by the models under study and the real data. The results are presented in Table 11 for continuous features, for which we measure the difference between the real data and generated data distributions using the Wasserstein distance, and in Table 12 for categorical features, for which we measure the difference between the real data and generated data distributions using the Jensen-Shannon divergence.

As we can see from Table 11, for every model and dataset there is very little difference in the Wasserstein distance obtained with the standard DGMs, the P-DGMs, and the C-DGMs. The only big gap is registered for GOGGLE on the WiDS dataset, where the Wasserstein distance obtained with C-GOGGLE is equal to 0.05 while the one obtained with GOGGLE and P-GOGGLE is equal to 0.22. Regarding the results for the categorical features in Table 12 we can see that the differences between DGMs and C-DGMs are very small, while, as expected, there is no difference between the results obtained with the DGMs and the P-DGMs. Indeed, our datasets are annotated with no constraints over the categorical features.

C.3 Variable Ordering Study

In this subsection, we first discuss how we picked the variable orderings tested in our experimental analysis, and then we conduct an in-detail experimental analysis of how the different orderings perform with each model.

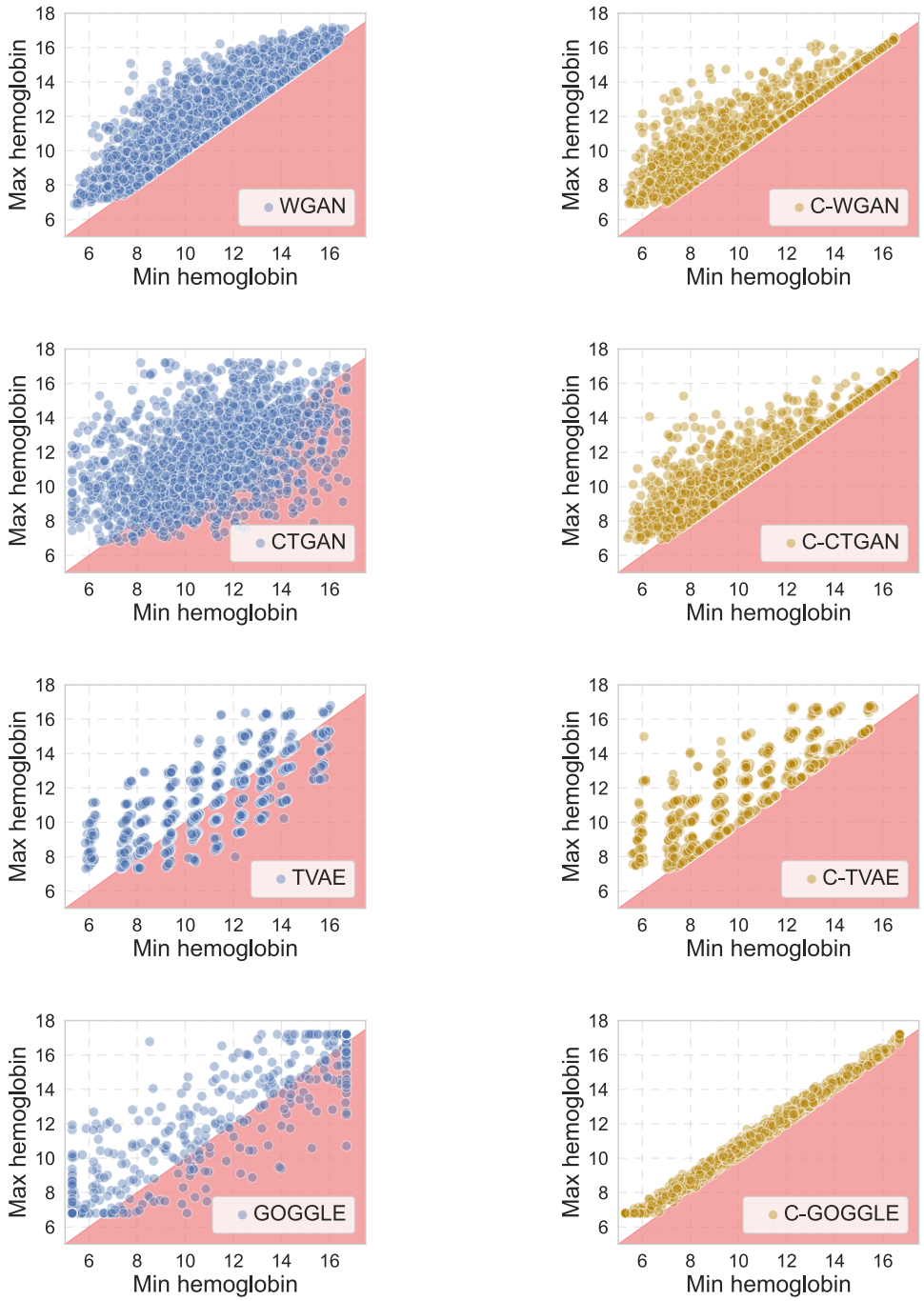

Figure 4: Samples generated by WGAN, CTGAN and TVAE, GOGGLE and their constrained versions for WiDS. The real and TableGAN distributions are given in Section 4.2 in the main text.

It is easy to see that given a sample $\tilde{x}$, the value of $\text{CL}(\tilde{x})$ may depend on the selected variable ordering, i.e., that different variable orderings may lead to different $\text{CL}(\tilde{x})$. Ideally, we would like the orderings to help in generating the highest quality samples possible. Thus, our intuition has been that features whose distributions are easier to learn should be assigned a lower ranking, so that their value can be fruitfully used to compute the value associated with features with higher ranking. With such intuition, we designed two different heuristics to choose such orderings. To

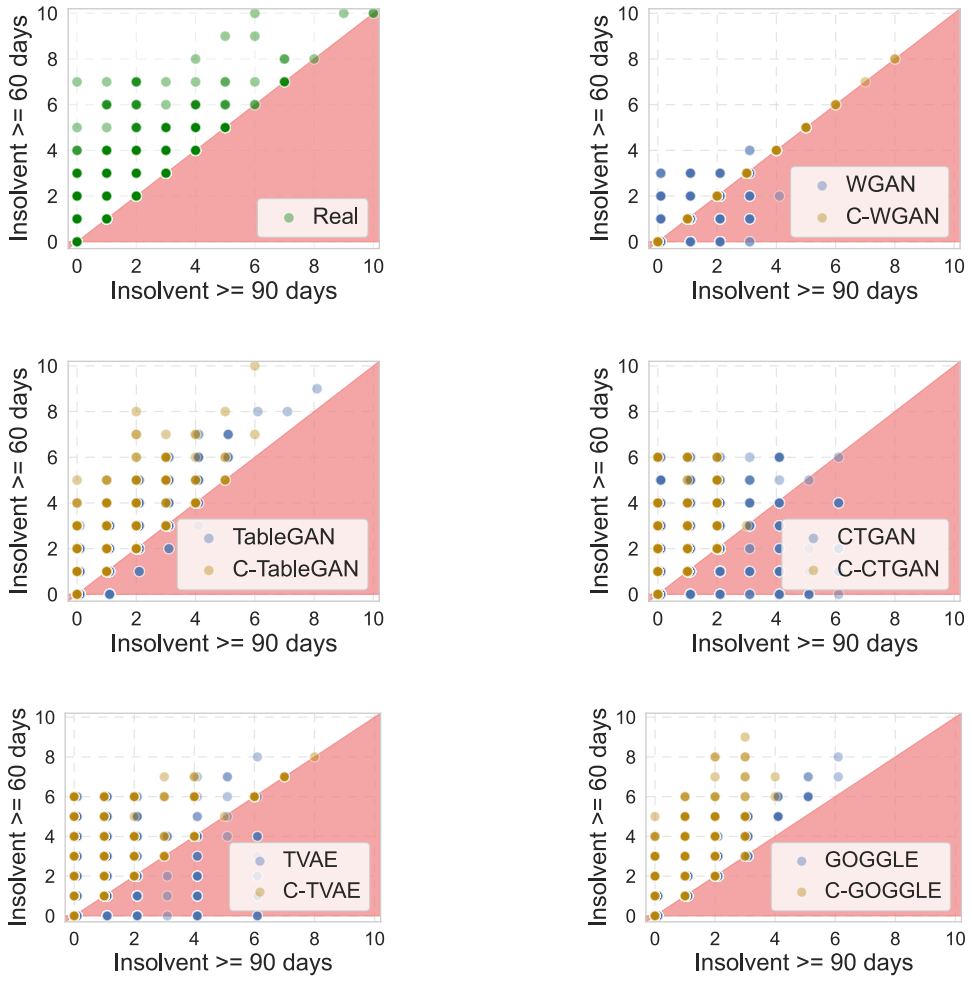

Figure 5: Real data and samples generated by WGAN, TableGAN, CTGAN, TVAE, GOGGLE and their constrained versions for Heloc.

obtain the two orderings described below, for each C-DGM, we first need to train the respective standard unconstrained DGM, and then generate a set of samples $\mathcal{S}$.

The first ordering we propose is a *correlation-based (Corr.) ordering*. Given the initial real data dataset $\mathcal{D}$ and $\mathcal{S}$, for each variable x_i we compute the score $\sigma_i = |\sum_{j \neq i} \text{corr}(C_i^{\mathcal{D}}, C_j^{\mathcal{D}}) - \sum_{j \neq i} \text{corr}(C_i^{\mathcal{S}}, C_j^{\mathcal{S}})|$, where $C_j^{\mathcal{D}}$ (resp. $C_j^{\mathcal{S}}$) is the vector of the values of the jth column in $\mathcal{D}$ (resp. $\mathcal{S}$). The lower the value of σ_i is, the lower the position x_i gets in the variable ordering, and the sooner its value is computed (as its relations with the other features should be easier to compute for the DGM).

The second ordering we propose is a *kernel density estimation (KDE)-based ordering*, which can be obtained by following the protocol outlined below:

1. Using scikit-learn's [9] (Pedregosa et al., 2011) implementation of KDE, we estimated a probability density function of the multidimensional real data.
2. We then computed the log-likelihood of each sample belonging to $\mathcal{D}$ and $\mathcal{S}$ under the estimated model, using again scikit-learn. It is worth noting that scikit-learn returns probability values normalised over the sample spaces under evaluation. Using these, we treated

[9] https://scikit-learn.org/stable/index.html

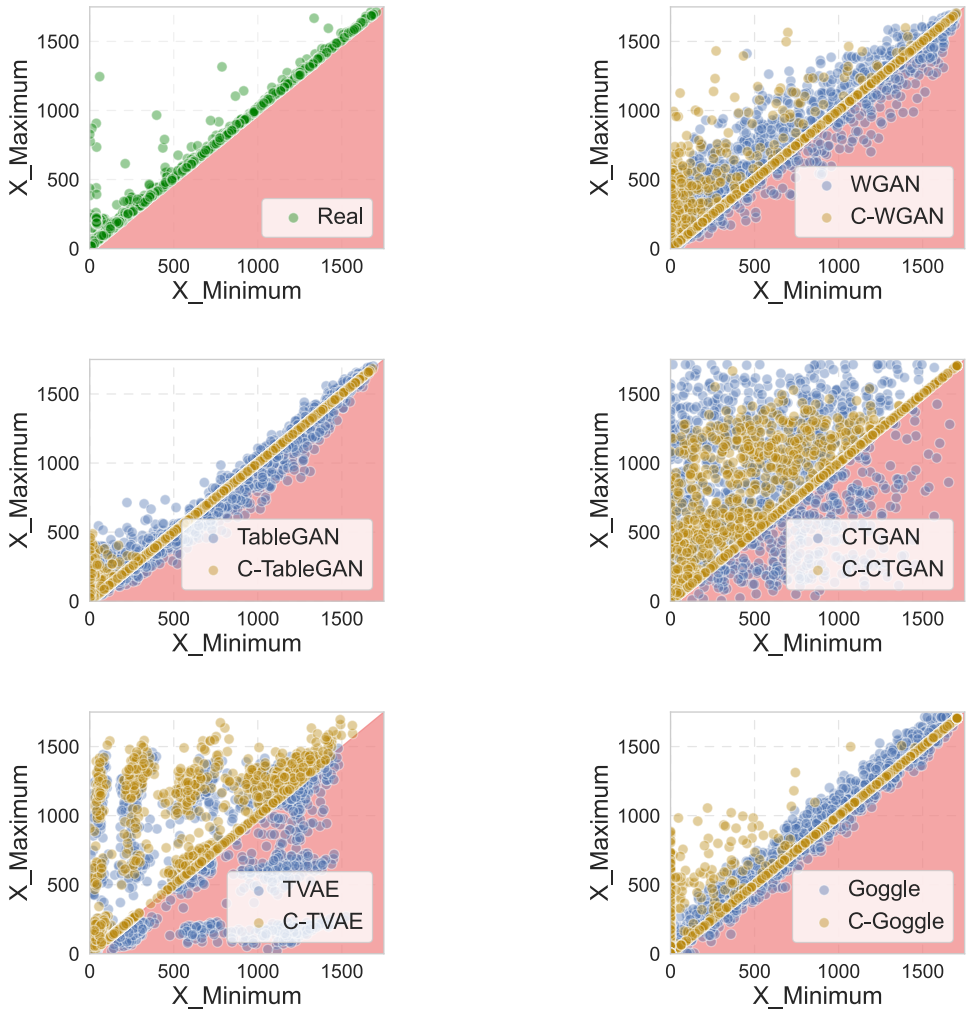

Figure 6: Real data and samples generated by WGAN, TableGAN, CTGAN, TVAE, GOGGLE and their constrained versions for FSP.

the problem in a discrete setting and approximated the marginal probability mass function of each variable. More specifically, for each feature x_i, we determined its unique values from $\mathcal{D} \cup \mathcal{S}$ and for each unique value u_{ij} we summed the KDE scores of the samples for which $x_i = u_{ij}$, separately for $\mathcal{D}$ and $\mathcal{S}$. In case no sample was observed with $x_i = u_{ij}$, we set the value of the marginal probability mass function of x_i to 0.

3. Hence, we obtain two marginal probability mass functions for each feature x_i (one for the real data D and one for the generated samples $\mathcal{S}$) and compute the Kullback-Leibler divergence (KL) between them to get a single value d_i.
4. Finally, we rank the features x_i based on their KL scores d_i in ascending order.

One disadvantage of this method is that the marginals are computed in a discrete setting. As future work, one interesting direction would be to use a higher-fidelity approximation of the marginal distributions, based on which the KL divergence scores, and hence the variable ranks in the ordering, are computed.

To assess the impact of using different orderings for each C-DGM model, we compared the two orderings proposed above with a random ordering (Rnd). Table 13 summarises the average utility

Table 10: Percentage of the generated samples that lie on the boundary.

	WiDS			Heloc			FSP		
	$p = 1\%$	$p = 5\%$	$p = 10\%$	$p = 1\%$	$p = 5\%$	$p = 10\%$	$p = 1\%$	$p = 5\%$	$p = 10\%$
WGAN	28.5±0.4	45.2±0.5	67.0±0.5	22.5±0.4	83.9±0.2	99.4±0.1	14.8±1.3	55.6±1.7	81.2±0.7
C-WGAN	62.1±0.3	72.1±0.2	83.1±0.2	100.0±0.0	100.0±0.0	100.0±0.0	76.8±1.0	83.1±0.8	89.9±0.5
TableGAN	19.7±0.1	71.9±0.2	88.7±0.1	37.4±0.2	94.7±0.1	99.8±0.0	25.3±1.2	76.0±0.9	95.8±0.5
C-TableGAN	72.2±0.2	81.1±0.3	88.7±0.3	83.8±0.3	96.1±0.2	99.1±0.1	75.4±0.5	86.2±0.5	94.3±0.3
CTGAN	6.5±0.0	30.5±0.3	53.2±0.2	80.2±0.4	93.1±0.5	96.8±0.3	2.6±0.3	16.4±0.4	36.7±1.3
C-CTGAN	54.2±0.6	62.4±0.6	73.8±0.5	83.2±0.4	96.0±0.1	98.3±0.1	49.2±0.8	62.3±1.1	72.9±1.4
TVAE	20.2±0.1	42.0±0.2	62.4±0.3	69.4±0.5	90.1±0.4	96.5±0.2	6.5±0.7	37.6±1.6	56.7±1.8
C-TVAE	31.0±0.4	56.9±0.5	69.7±0.3	79.2±0.7	92.7±0.4	97.1±0.2	38.4±1.2	57.5±0.4	77.4±0.9
GOGGLE	0.9±0.1	29.1±0.2	33.1±0.3	46.3±0.4	99.6±0.1	100.0±0.0	35.3±0.0	82.2±0.0	98.9±0.0
C-GOGGLE	7.7±0.1	77.2±0.2	99.1±0.1	83.9±0.3	92.4±0.3	97.0±0.1	81.1±0.0	85.4±0.0	89.7±0.0
DGMs	15.2±0.2	43.8±0.3	60.9±0.3	51.1±0.4	92.3±0.3	98.5±0.1	16.9±0.7	53.5±0.9	73.9±0.9
C-DGMs	45.4±0.3	69.9±0.4	82.9±0.3	86.0±0.3	95.4±0.2	98.3±0.1	64.2±0.7	74.9±0.6	84.8±0.6
Real	60.1±0.0	73.3±0.0	85.1±0.0	85.7±0.0	96.5±0.0	98.9±0.0	68.4±0.0	82.8±0.0	98.9±0.0

Table 11: Wasserstein distance between numerical features.

	URL	WiDS	LCLD	Heloc	FSP	News
WGAN	0.01±0.00	0.04±0.00	0.02±0.00	0.03±0.00	0.04±0.00	0.02±0.00
P-WGAN	0.02±0.00	0.04±0.00	0.02±0.00	0.03±0.00	0.07±0.00	0.02±0.00
C-WGAN	0.03±0.00	0.05±0.00	0.02±0.00	0.03±0.00	0.07±0.00	0.02±0.00
TableGAN	0.01±0.00	0.02±0.00	0.01±0.00	0.01±0.00	0.02±0.00	0.02±0.00
P-TableGAN	0.02±0.00	0.02±0.00	0.02±0.00	0.01±0.00	0.03±0.00	0.02±0.00
C-TableGAN	0.01±0.00	0.02±0.00	0.01±0.00	0.02±0.00	0.02±0.00	0.02±0.00
CTGAN	0.01±0.00	0.05±0.00	0.03±0.00	0.02±0.00	0.06±0.01	0.01±0.00
P-CTGAN	0.02±0.00	0.05±0.00	0.03±0.00	0.02±0.00	0.08±0.00	0.01±0.00
C-CTGAN	0.01±0.00	0.04±0.00	0.03±0.00	0.02±0.00	0.06±0.01	0.02±0.00
TVAE	0.01±0.00	0.01±0.00	0.02±0.00	0.01±0.00	0.02±0.00	0.01±0.00
P-TVAE	0.02±0.00	0.02±0.00	0.02±0.00	0.02±0.00	0.04±0.00	0.01±0.00
C-TVAE	0.01±0.00	0.02±0.00	0.01±0.00	0.01±0.00	0.03±0.00	0.01±0.00
GOGGLE	0.03±0.01	0.22±0.10	0.04±0.00	0.05±0.01	0.12±0.01	0.08±0.01
P-GOGGLE	0.03±0.01	0.22±0.10	0.04±0.00	0.05±0.01	0.12±0.02	0.08±0.01
C-GOGGLE	0.04±0.01	0.05±0.01	0.08±0.02	0.07±0.02	0.14±0.02	0.13±0.03

performance over the binary and multiclass classification datasets (i.e., all except the News) and the average detection performance over all datasets.

For detection, we can see that the KDE-based ordering performs best for C-WGAN, C-TableGAN and C-CTGAN — according to all three metrics (i.e., F1-score, weighted F1-score, and Area Under the ROC Curve) for the first two, and according to two out of three metrics (i.e. weighted F1-score and Area Under the ROC Curve) for C-CTGAN. On the other hand, the correlation-based ordering yields better results for C-TVAE across all three metrics. Only C-GOGGLE gets best detection results across all three metrics using the random ordering. For utility, we see again that different C-DGM models work best with different orderings. For instance, the KDE-based ordering performs best w.r.t. all three utility metrics for C-WGAN, C-TVAE and C-CTGAN. However, for C-TableGAN and C-GOGGLE, the correlation-based ordering outperforms the other orderings.

Overall, the utility and detection results we obtained using KDE-based or correlation-based orderings as opposed to random ordering highlight the importance of choosing an ordering that takes into account the data distribution and/or the feature relations captured by the C-DGM models in their predictions. Additionally, we can see that a trend emerges for C-WGAN, C-TableGAN, and C-TVAE: each of these models has a clear preference for the ordering that gives the highest overall

Table 12: Jensen-Shannon divergence between categorical features.

	URL	WiDS	LCLD	Heloc	FSP	News
WGAN	0.76±0.00	0.79±0.00	0.60±0.00	0.58±0.01	0.33±0.01	0.77±0.00
P-WGAN	0.76±0.00	0.79±0.00	0.60±0.00	0.58±0.01	0.33±0.01	0.77±0.00
C-WGAN	0.77±0.00	0.80±0.01	0.60±0.00	0.58±0.00	0.33±0.01	0.77±0.00
TableGAN	0.78±0.00	0.81±0.00	0.69±0.01	0.62±0.02	0.35±0.01	0.77±0.00
P-TableGAN	0.78±0.00	0.81±0.00	0.69±0.01	0.62±0.02	0.35±0.01	0.77±0.00
C-TableGAN	0.78±0.00	0.80±0.00	0.60±0.01	0.59±0.01	0.37±0.02	0.77±0.00
CTGAN	0.76±0.00	0.75±0.00	0.49±0.01	0.56±0.00	0.33±0.00	0.77±0.00
P-CTGAN	0.76±0.00	0.75±0.00	0.49±0.01	0.56±0.00	0.33±0.00	0.77±0.00
C-CTGAN	0.76±0.00	0.75±0.00	0.49±0.00	0.56±0.01	0.33±0.00	0.77±0.00
TVAE	0.77±0.00	0.79±0.00	0.50±0.00	0.57±0.00	0.33±0.00	0.77±0.00
P-TVAE	0.77±0.00	0.79±0.00	0.50±0.00	0.57±0.00	0.33±0.00	0.77±0.00
C-TVAE	0.77±0.00	0.80±0.00	0.50±0.00	0.57±0.01	0.33±0.00	0.77±0.00
GOGGLE	0.71±0.02	0.76±0.03	0.49±0.01	0.58±0.01	0.37±0.03	0.76±0.00
P-GOGGLE	0.71±0.02	0.76±0.03	0.49±0.01	0.58±0.01	0.37±0.03	0.76±0.00
C-GOGGLE	0.71±0.01	0.82±0.01	0.49±0.01	0.57±0.01	0.39±0.01	0.76±0.01

Table 13: Variable orderings comparison for C-DGMs. The best results are in bold.

		Utility (↑)			**Detection (↓)**		
		F1	*w*F1	AUC	F1	*w*F1	AUC
C-WGAN	Rnd	0.453	0.477	0.735	0.936	0.933	0.950
	Corr	0.475	0.497	0.746	0.919	**0.915**	0.936
	KDE	**0.485**	**0.504**	**0.748**	**0.917**	**0.915**	**0.935**
C-TableGAN	Rnd	0.346	0.409	0.710	0.908	0.904	0.926
	Corr	**0.361**	**0.422**	**0.717**	0.897	**0.893**	**0.916**
	KDE	0.349	0.413	0.706	**0.895**	**0.893**	**0.916**
C-CTGAN	Rnd	0.509	0.530	0.770	0.894	0.896	0.924
	Corr	0.513	0.536	**0.773**	**0.887**	0.891	0.920
	KDE	**0.516**	**0.537**	**0.773**	0.888	**0.889**	**0.919**
C-TVAE	Rnd	0.487	0.522	0.766	0.873	0.870	0.901
	Corr	**0.504**	0.534	0.771	**0.868**	**0.868**	**0.898**
	KDE	**0.504**	**0.536**	**0.774**	0.875	0.875	0.905
C-GOGGLE	Rnd	0.384	0.406	0.653	**0.922**	**0.916**	**0.937**
	Corr	**0.409**	**0.424**	**0.661**	0.929	0.918	0.940
	KDE	0.393	0.416	**0.661**	0.928	0.926	0.941

improvements w.r.t. utility and detection. It is also worth noticing that it is not always the case that an ordering improves both utility and detection, as we have seen for C-TableGAN and C-TVAE.

We also show the effect of using different orderings on P-DGM models, in Table 14. As opposed to the trends we noticed for our C-DGMs, here we find that it is harder to establish clear patterns in the preference of the P-DGMs towards any of the orderings. This is mainly due to the very small differences between the results, as it can be seen from the Table.

Exploring other variable orderings. In our work, we explored two variable orderings and conducted extensive experiments with them. However, there are various other ways to define these orderings. One advantage of customizing the order is that it can be tailored to the user's needs. For example, if users require orderings that consider how closely the generated data distribution matches the real data distribution, there are numerous ways to achieve this. In our detailed analysis, we examined a KDE-based ordering that compares the joint distributions of the features. However, a simpler alternative method would be to use the Wasserstein distance, a metric discussed in Section C.2,

Table 14: Variable orderings comparison for P-DGM. Best results are in bold.

		Utility (↑)			Detection (↓)		
		F1	*w*F1	AUC	F1	*w*F1	AUC
P-WGAN	Rnd	0.456	0.484	0.731	**0.930**	**0.928**	**0.945**
	Corr	0.462	**0.489**	**0.732**	0.931	0.930	0.947
	KDE	**0.463**	**0.489**	0.731	0.933	0.931	0.948
P-TableGAN	Rnd	**0.328**	0.398	0.705	0.902	0.901	0.925
	Corr	**0.328**	**0.399**	0.703	**0.900**	**0.900**	**0.922**
	KDE	0.326	0.398	**0.706**	0.901	0.901	0.924
P-CTGAN	Rnd	0.507	0.524	0.769	**0.898**	**0.901**	**0.926**
	Corr	**0.508**	**0.527**	**0.770**	0.899	0.902	0.928
	KDE	0.507	0.524	0.769	0.899	0.903	0.927
P-TVAE	Rnd	0.490	0.521	0.764	0.879	0.879	0.905
	Corr	0.493	0.523	0.763	**0.876**	0.879	0.904
	KDE	**0.494**	**0.524**	**0.767**	**0.876**	**0.875**	**0.901**
P-GOGGLE	Rnd	0.347	0.373	**0.626**	**0.925**	**0.925**	**0.943**
	Corr	**0.348**	**0.375**	0.625	0.929	0.926	0.945
	KDE	0.347	0.374	**0.626**	0.929	0.927	0.944

which differs from the KDE approach in that it compares individual distributions of the features rather than joint ones. In yet another scenario, users could define orderings without relying on the outputs of the unconstrained model by considering relations between the features in the real data, which is a different approach to all the previously-mentioned orderings. For instance, one way to do this is to order the variables based on the cause-effect relations between them, which we refer to as a causal-based ordering. More precisely, in such an ordering, there is no instance of a cause-effect pair of features (x_i, x_j) in which x_j appears before x_i in the ordering.

To assess the performance of our layer when constructed using different orderings, we conducted experiments with Wasserstein- and causal-based orderings on the WiDS dataset. For the Wasserstein-based ordering, for each feature we calculated the Wasserstein distance between the real data distribution and the generated data distribution obtained from an unconstrained DGM. We then arranged the features in ascending order based on the calculated distances, such that the features with generated data distributions furthest from the real data distributions would be corrected last by our constraint layer.

We note that the Wasserstein distance can only be defined on a metric space, making it impractical for use with categorical features without additional modifications (i.e. it is possible to define distance metrics for each of the categorical features and then apply Wasserstein distance on these features). However, here we assume that the constraint layer can only be applied on continuous features and, thus, the position of the categorical features in the orderings will not impact the final results, as our constraints exclude such features. As an alternative, we investigated using the Jensen-Shannon divergence to derive another ordering that compares individual feature distributions. And while this method offers has the advantage that it can be applied on both continuous and categorical features, we found that it was unstable in its results, aligning with the observations made by Zhao et al. (2021).

To compute the causal-based ordering, we obtained a directed acyclic graph (DAG), capturing the cause-effect relations between the features, and topologically sorted the features. Since none of the datasets we experimented with provided any information on causal relations between features, we utilized DAG-GNN (Yu et al., 2019) to obtain such relations in the training partition of each dataset. DAG-GNN is a gradient-based causal discovery method employing graph neural networks to learn a DAG structure which encodes cause-effect relations. Importantly for our application, we require that the graph has a topological ordering, implying the graph should be a DAG. However, determining a DAG structure poses a challenge in the causality domain. In fact, the DAG-GNN method does not guarantee that its output structure is a DAG, as the DAG constraint is embedded in a loss term minimised during training. In our experiments, we observed that DAG-GNN produced a few self-loops (edges connecting a node to itself) for most datasets and one cycle for half of the datasets.

Table 15: Comparing DGMs with their C-DGM versions using 5 different orderings, when trained on the WiDS dataset. Best results are in bold.

		Utility (↑)			**Detection (↓)**		
		F1	*w*F1	AUC	F1	*w*F1	AUC
C-WGAN	Rnd	0.303	0.360	0.797	0.995	0.996	0.999
	Corr	0.284	0.343	0.796	0.975	0.976	0.989
	KDE	**0.316**	**0.372**	**0.815**	0.975	0.975	0.989
	WD	0.273	0.331	0.770	**0.932**	**0.925**	**0.948**
	Caus	0.275	0.334	0.775	0.944	0.929	0.956
C-TableGAN	Rnd	0.213	0.279	**0.777**	0.984	0.984	0.996
	Corr	**0.246**	**0.309**	0.775	**0.956**	**0.957**	**0.974**
	KDE	0.208	0.274	0.767	0.962	0.963	0.979
	WD	0.242	0.305	0.770	0.961	0.962	0.977
	Caus	0.225	0.289	0.770	0.962	0.963	0.979
C-CTGAN	Rnd	0.364	0.408	0.836	0.990	0.990	0.997
	Corr	0.365	0.409	0.826	0.988	0.988	0.995
	KDE	0.360	0.403	0.832	0.986	0.986	0.994
	WD	**0.368**	**0.411**	**0.842**	**0.953**	**0.955**	**0.970**
	Caus	0.365	0.409	0.838	0.954	0.956	**0.970**
C-TVAE	Rnd	0.248	0.311	0.773	0.965	0.965	0.982
	Corr	0.305	0.363	0.804	0.959	0.960	0.977
	KDE	**0.321**	**0.378**	**0.816**	0.961	0.962	0.979
	WD	0.297	0.356	0.794	**0.958**	**0.959**	**0.976**
	Caus	0.316	0.373	0.808	**0.958**	**0.959**	**0.976**
C-GOGGLE	Rnd	0.139	0.210	0.643	**0.965**	**0.965**	**0.979**
	Corr	0.185	0.253	0.675	0.972	0.971	0.984
	KDE	0.171	0.239	0.678	0.975	0.975	0.984
	WD	**0.207**	**0.273**	**0.705**	0.980	0.980	0.990
	Caus	0.175	0.244	0.681	0.966	**0.965**	**0.979**

To address these issues, we eliminated the self-loops and broke the cycles by randomly selecting an edge from each cycle and removing it from the graph. This approach enabled us to obtain a DAG structure and, consequently, a topological ordering that we eventually used as the causal-based ordering in our experiments.

For our experiments we considered the medium-sized datasets and selected WiDS. Our choice was mainly guided by the levels of difficulty involved when manually checking the causality relations found by the DAG-GNN model. Compared to the other datasets, WiDS' columns had a clearer and more explanatory description, which allowed us to determine whether the cause-effect relations between the columns were sensible. In Tables 15 and 16 we compared the 5 different orderings on the WiDS dataset for all C-DGMs and P-DGMs, respectively. As we can see in Table 15, both the causal- and Wasserstein distance-based orderings achieved better results than the random ordering in 12 (and 11) out of 15 cases for detection (and utility). On the other hand, we notice that when used during postprocessing, these two new orderings can only bring small performance improvements. However, similarly to the other orderings we explored, it is harder to distinguish trends where the new orderings might be more helpful than the random ordering.

Future work on variable orderings. As we can see, using custom-made orderings can improve the performance of C-DGMs which use a random ordering of the variables w.r.t. both utility and detection. However, it is not straightforward to determine which ordering works best in which scenario. To leverage the constraint layer's capabilities to the maximum, for future work we plan to investigate when different feature orderings might work best and uncover patterns in the models' and datasets' preferences towards certain orderings.

Table 16: Comparing DGMs with their P-DGM versions using 5 different orderings, when trained on the WiDS dataset. Best results are in bold.

		Utility (↑)			**Detection (↓)**		
		F1	*w*F1	AUC	F1	*w*F1	AUC
P-WGAN	Rnd	0.319	0.372	0.777	0.979	0.980	0.992
	Corr	0.332	0.384	0.781	**0.976**	**0.977**	**0.991**
	KDE	**0.334**	**0.386**	**0.783**	0.978	0.978	**0.991**
	WD	0.332	0.384	0.776	0.979	0.979	0.992
	Caus	0.332	0.383	0.774	0.980	0.981	0.992
P-TableGAN	Rnd	0.173	0.242	**0.749**	**0.962**	**0.963**	0.980
	Corr	0.174	0.242	0.731	0.963	0.964	0.981
	KDE	0.171	0.240	0.735	**0.962**	**0.963**	0.980
	WD	**0.181**	**0.250**	0.738	**0.962**	**0.963**	**0.979**
	Caus	0.171	0.240	0.741	0.963	0.964	0.980
P-CTGAN	Rnd	0.364	0.407	**0.837**	0.991	0.991	**0.997**
	Corr	0.365	0.407	0.835	**0.989**	**0.989**	**0.997**
	KDE	0.357	0.400	0.835	0.991	0.991	**0.997**
	WD	**0.375**	**0.419**	0.836	0.990	0.990	**0.997**
	Caus	0.363	0.406	0.835	0.990	0.990	**0.997**
P-TVAE	Rnd	0.266	0.327	0.785	**0.962**	0.963	**0.978**
	Corr	0.282	0.342	0.787	**0.962**	**0.962**	0.979
	KDE	**0.285**	**0.345**	**0.797**	0.963	0.964	0.979
	WD	0.280	0.340	0.778	**0.962**	**0.962**	0.979
	Caus	0.269	0.329	0.793	**0.962**	**0.962**	0.979
P-GOGGLE	Rnd	**0.192**	**0.202**	0.665	0.988	0.988	**0.993**
	Corr	0.191	0.201	0.667	0.988	0.988	**0.993**
	KDE	0.189	0.197	0.667	**0.987**	**0.987**	**0.993**
	WD	0.191	0.200	0.663	0.988	0.988	**0.993**
	Caus	0.191	0.200	**0.669**	0.988	0.988	0.994

C.4 Full results on DGMs vs. C-DGMs

To assess the impact of constraining DGMs with our method, we conducted an extensive experimental analysis where we compared the performance of our C-DGM models with the baseline DGM models in terms of utility and detection, as specified in Section 4.1. In Tables 17-23 we present full utility and detection results. More specifically, in each table we report results for each dataset, each DGM and each C-DGM, using 3 different orderings (i.e., random, correlation-based and KDE-based). Additionally, for each result in the table, we report the standard deviation from the mean. As a reminder, we obtained the results by running each experiment with 5 different seeds. As shown, in most cases at least one of the orderings used for the C-DGMs outperforms the DGMs, particularly for detection.

Comparing WGAN with C-WGAN, we note that the only dataset for which we do not get an improvement in utility with either of the three different metrics is LCLD. However, the gap between the WGAN and the best C-WGAN is small for all the 3 different metrics: 0.7%, 0.1%, 0.2% for F1-score, weighted F1-score and Area Under the ROC Curve, respectively. In most cases, with C-WGAN we see significant utility improvements, e.g. 7.7%, 5% and 3.8% for Heloc in F1-score, weighted F1-score and Area Under the ROC Curve, respectively.

For TableGAN, we see that C-TableGAN outperforms the unconstrained models for all binary and multiclass classification datasets in terms of utility, according to at least 2 out of 3 metrics. For the cases where the performance is not improved under some metric, we notice that the difference in performance between TableGAN and C-TableGAN is small, e.g., 0.5% for the FSP dataset and 0.2% for the Heloc dataset in Area Under the ROC Curve. It is worth noticing that out of all the C-DGMs, C-TableGAN brings the highest improvements overall. For example, C-TableGAN outperforms the

unconstrained TableGAN by at least 4.5% according to the F1-score in 4 datasets, with the highest improvement, of 7.5%, recorded for WiDS.

As opposed to C-TableGAN, C-CTGAN does not show the same trend, with most models giving either moderate improvements or performing close to the unconstrained CTGAN. However, it is still the case that most of the datasets (four out of six) show improvements over all three metrics (with the remaining two datasets showing improvements on two out of three metrics). Notably, with C-CTGAN we did observe a large improvement of 64.9 points in the mean absolute error for utility when training on the News dataset, which is the highest improvement we observed with any of the C-DGM models.

Similarly, to the C-WGAN case, there is only one dataset for which C-TVAE does not improve the utility performance in any of the three metrics, namely the Heloc dataset. Nevertheless, the difference between the overall best C-TVAE model (i.e. using the KDE ordering) and TVAE is small for all the 3 different metrics: 0.4%, 0.2%, 0.1% for F1-score, weighted F1-score and Area Under the ROC Curve, respectively. In most cases, with C-TVAE we see major utility improvements, e.g. 4.0%, 3.6% and 1.5% for WiDS in F1-score, weighted F1-score and Area Under the ROC Curve, respectively, and 2.6%, 2.9% and 1.4% for FSP in F1-score, weighted F1-score and Area Under the ROC Curve, respectively.

For C-GOGGLE, we see that five out of six datasets outperform the standard GOGGLE results on all three metrics, giving major improvements (with the largest improvement being of 17.1% in the F1-score for utility when training on the URL dataset). Only the FSP dataset does not show any improvements on any of the orderings when using the C-GOGGLE version.

For detection, with C-WGAN we get an improvement (or do not change the performance, as it happens in a few cases) over all datasets in all metrics, the only exception being the Area Under the ROC Curve for the URL dataset. With C-TableGAN, C-CTGAN, and C-TVAE, we see an improvement in 4 out of 6 datasets, where we outperform the unconstrained models (i) according to all metrics for 3 datasets and (ii) according to at least two out of three metrics for 1 dataset, respectively. With C-GOGGLE we see again an improvement in 4 out of 6 datasets, where we outperform the unconstrained models (i) according to all metrics for 2 datasets and (ii) according to at least one out of three metrics for 2 datasets, respectively.

C.4.1 DGMs vs. C-DGMs: Utility Results

In Tables 17-19 we present the utility results on all datasets using 3 metrics in the following order: F1-score, weighted F1-score, and Area Under the ROC Curve. Separately, in Table 20, we report the performance of real and synthetic data for the News dataset (which is a regression dataset), using 4 different metrics: Explained Variance (XV) and Mean Absolute Error (MAE).

C.4.2 DGMs vs. C-DGMs: Detection Results

In Tables 21-23 we present the detection results on all datasets using 3 metrics in the following order: F1-score, weighted F1-score and Area Under the ROC Curve.

C.5 Full results on DGMs vs. P-DGMs

Our constrained layer uses the constraints to correct the predictions both at training and inference time, however, the constraints can simply be used at inference time to correct the predictions only once. This latter approach, which we call P-DGM, is a way of putting guardrails on the output space of the DGMs and could be the preferred option for users who do not want to make any modifications to their models or retrain them. Both methods guarantee that the constraints are satisfied by the predictions, however, their impact on the models' performance might differ. To see this, we compared DGMs with P-DGMs whose predictions have been corrected according to three different label orderings: random, correlation-based, and KDE-based. In each of the following tables, we report the mean and standard deviation from the mean.

C.5.1 DGMs vs. P-DGMs: Utility Results

In Tables 24, 25 and 26 we compare the performance of the DGM and P-DGM models in terms of utility and report the results w.r.t. 3 different metrics: F1-score, weighted F1-score, and Area Under the ROC Curve. Separately, in Table 27, we report the performance of the synthetic data for the News dataset, using two metrics: XV and MAE.

As we can see, putting guardrails on the output space of the models can help increase the performance, however, the overall gains are not as high as those used by our C-DGM models which correct the outputs at training time.

C.5.2 DGMs vs. P-DGMs: Detection results

In Tables 28, 29 and 30 we compare the performance of the DGM and P-DGM models in terms of detection. Similarly to the utility case, post-processing the outputs of the unconstrained models can help slightly increase the overall performance.

Table 17: Binary (macro) F1-score utility results with their corresponding stddevs for binary (multi-class) classification datasets.

		URL	WiDS	LCLD	Heloc	FSP
WGAN	-	0.756±0.030	**0.329**±0.059	**0.239**±0.026	0.634±0.099	0.357±0.020
	Rnd	0.792±0.021	0.302±0.075	0.196±0.026	0.643±0.060	0.333±0.028
C-WGAN	Corr	0.790±0.026	0.284±0.040	0.232±0.050	0.704±0.039	**0.367**±0.027
	KDE	**0.801**±0.004	0.316±0.044	0.232±0.050	**0.711**±0.030	**0.367**±0.027
TableGAN	-	0.562±0.051	0.171±0.037	0.123±0.041	0.593±0.058	0.199±0.044
	Rnd	0.534±0.096	0.213±0.027	0.148±0.058	0.636±0.073	0.199±0.031
C-TableGAN	Corr	**0.611**±0.111	**0.246**±0.047	0.130±0.036	**0.638**±0.061	0.179±0.036
	KDE	0.576±0.074	0.207±0.043	**0.174**±0.063	0.582±0.114	**0.208**±0.053
CTGAN	-	0.822±0.017	0.362±0.033	0.247±0.087	0.736±0.035	0.374±0.034
	Rnd	0.833±0.006	**0.365**±0.019	0.235±0.057	**0.744**±0.010	0.370±0.006
C-CTGAN	Corr	0.830±0.009	**0.365**±0.020	0.260±0.081	0.730±0.027	**0.383**±0.019
	KDE	**0.836**±0.002	0.360±0.022	**0.265**±0.040	0.736±0.010	0.381±0.030
TVAE	-	0.810±0.008	0.282±0.029	**0.185**±0.021	**0.735**±0.010	0.473±0.016
	Rnd	0.824±0.004	0.249±0.038	0.143±0.018	0.720±0.014	**0.501**±0.012
C-TVAE	Corr	**0.826**±0.007	0.305±0.021	0.158±0.011	0.733±0.017	0.496±0.018
	KDE	0.824±0.004	**0.322**±0.041	0.146±0.023	0.731±0.008	0.498±0.014
GOGGLE	-	0.622±0.094	**0.189**±0.038	0.163±0.119	0.596±0.072	**0.152**±0.003
	Rnd	0.782±0.035	0.139±0.068	0.157±0.085	**0.723**±0.018	0.117±0.008
C-GOGGLE	Corr	**0.793**±0.013	0.185±0.108	**0.219**±0.023	0.714±0.013	0.136±0.024
	KDE	0.788±0.014	0.171±0.036	0.167±0.082	0.708±0.025	0.134±0.016

Table 18: Weighted F1-score utility results with their corresponding stddevs for binary and multi-class classification datasets.

		URL	WiDS	LCLD	Heloc	FSP
WGAN	-	0.764±0.018	**0.381**±0.057	**0.359**±0.019	0.599±0.050	0.339±0.021
	Rnd	0.782±0.025	0.360±0.068	0.339±0.020	0.585±0.079	0.316±0.029
C-WGAN	Corr	0.785±0.011	0.344±0.036	0.358±0.037	0.648±0.022	**0.349**±0.028
	KDE	**0.790**±0.007	0.373±0.040	0.358±0.037	**0.649**±0.011	**0.349**±0.028
TableGAN	-	0.659±0.035	0.240±0.034	0.286±0.029	0.615±0.030	0.199±0.043
	Rnd	0.644±0.061	0.280±0.025	0.306±0.043	0.618±0.042	0.198±0.030
C-TableGAN	Corr	**0.695**±0.071	**0.309**±0.042	0.292±0.029	**0.633**±0.036	0.180±0.034
	KDE	0.670±0.052	0.274±0.039	**0.314**±0.029	0.596±0.048	**0.209**±0.051
CTGAN	-	0.799±0.033	0.405±0.034	0.379±0.061	0.675±0.015	0.372±0.034
	Rnd	0.818±0.008	0.408±0.019	0.369±0.044	0.684±0.005	0.369±0.007
C-CTGAN	Corr	0.816±0.012	**0.409**±0.019	0.388±0.060	0.688±0.010	0.379±0.020
	KDE	**0.820**±0.008	0.403±0.023	**0.392**±0.030	**0.690**±0.010	**0.380**±0.032
TVAE	-	0.802±0.012	0.342±0.027	**0.330**±0.016	**0.696**±0.006	0.463±0.017
	Rnd	**0.817**±0.007	0.311±0.034	0.301±0.012	0.687±0.006	**0.492**±0.013
C-TVAE	Corr	0.816±0.007	0.363±0.019	0.311±0.011	0.690±0.009	0.488±0.019
	KDE	0.816±0.008	**0.378**±0.038	0.305±0.016	0.694±0.003	0.490±0.014
GOGGLE	-	0.648±0.074	0.198±0.067	0.315±0.086	0.566±0.050	**0.139**±0.002
	Rnd	0.741±0.032	0.210±0.061	0.314±0.065	**0.663**±0.012	0.103±0.008
C-GOGGLE	Corr	**0.752**±0.024	**0.253**±0.098	**0.357**±0.020	0.637±0.023	0.121±0.024
	KDE	0.749±0.029	0.239±0.032	0.318±0.060	0.656±0.014	0.119±0.017

Table 19: Area under the ROC curve utility results with their corresponding stddevs for binary and multiclass classification datasets.

		URL	WiDS	LCLD	Heloc	FSP
WGAN	-	0.839±0.016	0.775±0.038	**0.618**±0.011	0.677±0.033	**0.742**±0.007
	Rnd	0.860±0.018	0.798±0.028	0.616±0.009	0.672±0.037	0.728±0.010
C-WGAN	Corr	**0.865**±0.011	0.796±0.017	0.612±0.016	0.714±0.017	**0.742**±0.022
	KDE	0.858±0.011	**0.816**±0.015	0.612±0.016	**0.715**±0.011	**0.742**±0.022
TableGAN	-	0.843±0.020	0.740±0.021	0.587±0.027	**0.707**±0.007	**0.642**±0.026
	Rnd	0.853±0.018	**0.778**±0.017	0.593±0.019	0.692±0.029	0.637±0.022
C-TableGAN	Corr	**0.868**±0.007	0.775±0.015	**0.605**±0.016	0.705±0.030	0.630±0.037
	KDE	0.865±0.010	0.767±0.016	0.587±0.017	0.676±0.037	0.635±0.028
CTGAN	-	0.859±0.040	0.835±0.012	**0.651**±0.020	0.744±0.009	**0.760**±0.014
	Rnd	**0.880**±0.004	**0.837**±0.003	0.626±0.028	0.749±0.008	0.757±0.008
C-CTGAN	Corr	0.877±0.010	0.827±0.013	0.643±0.015	**0.755**±0.007	**0.760**±0.009
	KDE	**0.880**±0.007	0.833±0.007	0.641±0.015	0.751±0.011	0.759±0.012
TVAE	-	0.863±0.011	0.800±0.016	0.631±0.004	**0.752**±0.003	0.789±0.007
	Rnd	0.876±0.008	0.773±0.036	0.630±0.002	0.750±0.005	**0.803**±0.009
C-TVAE	Corr	0.878±0.005	0.803±0.014	**0.633**±0.003	0.749±0.005	0.791±0.015
	KDE	**0.879**±0.007	**0.815**±0.028	0.632±0.003	0.751±0.002	0.794±0.009
GOGGLE	-	0.742±0.071	0.656±0.049	0.543±0.039	0.600±0.056	**0.577**±0.010
	Rnd	0.800±0.029	0.643±0.088	0.569±0.055	**0.719**±0.005	0.535±0.016
C-GOGGLE	Corr	0.794±0.030	0.675±0.051	**0.593**±0.046	0.696±0.022	0.546±0.011
	KDE	**0.802**±0.016	**0.678**±0.029	0.572±0.051	0.713±0.021	0.538±0.020

Table 20: Utility performance comparison between DGM and C-DGM models trained on News, using XV and MAE. In the first row, we report the real data performance.

		XV	MAE
Real data	-	-0.001±0.001	3001.1±55.0
WGAN	-	-0.006±0.003	3133.6±242.1
	Rnd	-0.008±0.012	3093.0±202.3
C-WGAN	Corr	**-0.002**±0.002	**3014.0**±79.1
	KDE	-0.004±0.001	3070.0±98.4
TableGAN	-	**-0.001**±0.001	**2992.2**±35.8
	Rnd	**-0.001**±0.001	3033.1±53.9
C-TableGAN	Corr	-0.002±0.002	3015.9±40
	KDE	-0.002±0.002	3016.8±70.8
CTGAN	-	**-0.002**±0.001	3043.8±37.8
	Rnd	**-0.002**±0.001	3034.6±29.4
C-CTGAN	Corr	**-0.002**±0.001	**2978.9**±28.7
	KDE	-0.003±0.001	3029.1±72.7
TVAE	-	**-0.001**±0.000	3021.3±10.0
	Rnd	-0.002±0.001	3067.7±71.5
C-TVAE	Corr	**-0.001**±0.001	**3005.0**±56.2
	KDE	-0.002±0.001	3011.7±44.2
GOGGLE	-	-0.004±0.003	3054.5±44.7
	Rnd	**-0.001**±0.001	3042.6±54.1
C-GOGGLE	Corr	**-0.001**±0.001	3026.0±34.6
	KDE	**-0.001**±0.000	**2999.0**±27.3

Table 21: Binary F1-score detection results with their corresponding stddevs for all datasets.

		URL	WiDS	LCLD	Heloc	FSP	News
WGAN	-	0.865±0.035	**0.975**±0.002	0.999±0.001	0.964±0.004	0.914±0.018	0.954±0.022
	Rnd	**0.864**±0.046	0.996±0.003	**0.916**±0.013	0.970±0.024	0.900±0.078	0.969±0.015
C-WGAN	Corr	0.879±0.027	**0.975**±0.006	0.923±0.005	0.964±0.007	**0.843**±0.037	0.928±0.023
	KDE	0.877±0.016	**0.975**±0.002	0.923±0.005	**0.957**±0.018	**0.843**±0.037	**0.926**±0.011
TableGAN	-	0.831±0.029	0.963±0.006	0.895±0.024	**0.923**±0.011	0.909±0.021	**0.927**±0.009
	Rnd	0.840±0.020	0.984±0.002	0.870±0.016	0.963±0.021	0.839±0.061	0.953±0.010
C-TableGAN	Corr	0.849±0.025	**0.956**±0.004	0.874±0.011	0.952±0.010	**0.818**±0.012	0.933±0.011
	KDE	**0.828**±0.031	0.962±0.007	**0.869**±0.014	0.948±0.018	0.830±0.015	0.933±0.011
CTGAN	-	0.850±0.027	0.990±0.002	0.848±0.016	0.914±0.024	0.926±0.011	**0.901**±0.018
	Rnd	0.834±0.026	0.990±0.001	0.863±0.022	0.910±0.013	0.859±0.022	0.909±0.016
C-CTGAN	Corr	**0.820**±0.025	0.987±0.003	**0.845**±0.025	0.903±0.012	0.861±0.021	0.905±0.019
	KDE	0.838±0.033	**0.986**±0.002	0.848±0.017	**0.897**±0.023	**0.857**±0.022	0.902±0.014
TVAE	-	**0.813**±0.037	**0.926**±0.001	0.842±0.019	0.914±0.011	**0.843**±0.027	0.877±0.013
	Rnd	0.826±0.037	0.965±0.001	**0.796**±0.030	**0.905**±0.022	0.874±0.015	0.874±0.018
C-TVAE	Corr	0.815±0.037	0.959±0.003	0.808±0.030	**0.905**±0.014	0.855±0.020	**0.863**±0.008
	KDE	0.814±0.030	0.961±0.002	0.799±0.020	0.907±0.021	0.879±0.008	0.890±0.019
GOGGLE	-	0.892±0.019	0.987±0.018	**0.890**±0.017	**0.924**±0.006	0.912±0.010	**0.949**±0.023
	Rnd	**0.890**±0.044	**0.965**±0.006	0.904±0.011	0.939±0.008	**0.866**±0.020	0.967±0.009
C-GOGGLE	Corr	0.898±0.021	0.972±0.005	0.910±0.017	0.939±0.010	0.884±0.009	0.968±0.015
	KDE	0.903±0.019	0.975±0.013	0.911±0.017	0.938±0.008	0.887±0.014	0.957±0.020

Table 22: Weighted F1-score detection results with their corresponding stddevs for all datasets.

		URL	WiDS	LCLD	Heloc	FSP	News
WGAN	-	0.856±0.008	0.976±0.002	0.999±0.001	0.964±0.004	0.910±0.027	0.954±0.020
	Rnd	**0.851**±0.017	0.996±0.003	**0.912**±0.013	0.970±0.024	0.900±0.072	0.969±0.016
C-WGAN	Corr	0.861±0.016	0.975±0.006	0.917±0.007	0.964±0.007	**0.845**±0.022	0.929±0.022
	KDE	0.871±0.010	**0.975**±0.002	0.917±0.007	**0.957**±0.019	**0.845**±0.022	**0.925**±0.013
TableGAN	-	**0.822**±0.007	0.964±0.006	0.895±0.022	**0.925**±0.011	0.906±0.028	**0.929**±0.013
	Rnd	0.830±0.011	0.984±0.002	**0.859**±0.015	0.963±0.021	0.834±0.054	0.953±0.011
C-TableGAN	Corr	0.835±0.011	**0.957**±0.003	0.866±0.022	0.952±0.010	**0.813**±0.020	0.936±0.010
	KDE	0.827±0.006	0.963±0.007	0.871±0.013	0.947±0.019	0.817±0.021	0.934±0.011
CTGAN	-	0.862±0.014	0.990±0.002	0.850±0.019	0.915±0.023	0.927±0.016	0.896±0.023
	Rnd	0.843±0.017	0.990±0.001	0.867±0.015	0.911±0.013	**0.861**±0.022	0.904±0.018
C-CTGAN	Corr	**0.839**±0.015	0.988±0.002	0.846±0.024	0.904±0.013	0.865±0.028	0.901±0.023
	KDE	0.849±0.020	**0.986**±0.002	**0.837**±0.014	**0.897**±0.023	0.868±0.009	**0.894**±0.018
TVAE	-	0.831±0.013	**0.927**±0.001	0.832±0.010	0.916±0.010	**0.847**±0.006	0.855±0.019
	Rnd	**0.829**±0.014	0.966±0.001	**0.795**±0.025	**0.908**±0.020	0.868±0.018	0.856±0.012
C-TVAE	Corr	0.832±0.011	0.960±0.003	**0.795**±0.014	**0.908**±0.013	0.857±0.011	**0.854**±0.010
	KDE	0.829±0.014	0.962±0.002	0.797±0.019	0.909±0.020	0.873±0.014	0.882±0.024
GOGGLE	-	**0.880**±0.012	0.987±0.018	**0.892**±0.014	**0.926**±0.005	0.916±0.011	**0.955**±0.019
	Rnd	0.891±0.030	**0.965**±0.005	0.905±0.010	0.940±0.008	0.823±0.030	0.970±0.008
C-GOGGLE	Corr	0.898±0.014	0.971±0.005	0.912±0.017	0.939±0.009	**0.817**±0.004	0.972±0.012
	KDE	0.902±0.017	0.975±0.013	0.913±0.018	0.939±0.008	0.865±0.033	0.963±0.015

Table 23: Area under the ROC curve detection results with their corresponding stddevs for all datasets.

		URL	WiDS	LCLD	Heloc	FSP	News
WGAN	-	**0.872**±0.008	**0.989**±0.002	1.000±0.000	0.983±0.004	0.916±0.016	0.964±0.021
	Rnd	0.877±0.017	0.999±0.001	**0.941**±0.009	0.983±0.018	0.918±0.060	0.981±0.010
C-WGAN	Corr	0.879±0.009	**0.989**±0.002	0.946±0.006	0.982±0.007	**0.874**±0.020	0.945±0.020
	KDE	0.883±0.009	**0.989**±0.001	0.946±0.006	**0.975**±0.018	**0.874**±0.020	**0.941**±0.011
TableGAN	-	0.850±0.007	0.980±0.004	0.926±0.020	**0.953**±0.009	0.907±0.022	**0.940**±0.011
	Rnd	0.851±0.006	0.995±0.001	**0.900**±0.012	0.978±0.016	0.866±0.048	0.964±0.012
C-TableGAN	Corr	0.856±0.004	**0.974**±0.002	0.904±0.019	0.970±0.008	**0.845**±0.005	0.950±0.010
	KDE	**0.849**±0.005	0.978±0.005	0.909±0.012	0.965±0.017	0.849±0.012	0.947±0.010
CTGAN	-	0.879±0.008	0.996±0.001	**0.896**±0.011	0.953±0.016	0.932±0.007	**0.912**±0.021
	Rnd	0.865±0.013	0.997±0.000	0.908±0.011	0.950±0.008	0.896±0.016	0.925±0.021
C-CTGAN	Corr	**0.864**±0.009	**0.995**±0.001	0.900±0.020	0.946±0.004	0.894±0.025	0.922±0.019
	KDE	0.867±0.015	**0.995**±0.001	0.897±0.012	**0.943**±0.018	**0.893**±0.007	0.919±0.022
TVAE	-	0.854±0.007	**0.935**±0.002	0.861±0.009	0.947±0.005	**0.872**±0.008	0.881±0.019
	Rnd	0.854±0.008	0.982±0.001	0.844±0.013	**0.942**±0.015	0.904±0.015	0.883±0.012
C-TVAE	Corr	0.853±0.009	0.977±0.001	**0.840**±0.011	0.943±0.011	0.896±0.008	**0.880**±0.014
	KDE	**0.849**±0.013	0.979±0.001	0.843±0.012	0.944±0.014	0.904±0.013	0.909±0.030
GOGGLE	-	**0.891**±0.009	0.993±0.013	**0.928**±0.010	**0.949**±0.003	0.920±0.006	0.973±0.013
	Rnd	0.898±0.028	**0.979**±0.007	0.938±0.010	0.958±0.008	0.871±0.009	0.977±0.007
C-GOGGLE	Corr	0.906±0.020	0.984±0.005	0.943±0.013	0.960±0.007	**0.865**±0.011	0.980±0.012
	KDE	0.907±0.018	0.985±0.009	0.944±0.013	0.960±0.007	0.880±0.014	**0.973**±0.009

Table 24: Binary (macro) F1-score utility results with their corresponding stddevs for binary (multi-class) classification datasets when postprocessing the unconstrained predictions.

		URL	WiDS	LCLD	Heloc	FSP
WGAN	-	0.756±0.030	0.329±0.059	**0.239**±0.026	0.634±0.099	0.357±0.020
	Rnd	**0.767**±0.034	0.318±0.053	0.201±0.030	**0.641**±0.113	0.355±0.020
P-WGAN	Corr	0.766±0.036	0.332±0.050	0.214±0.021	**0.641**±0.115	**0.358**±0.024
	KDE	**0.767**±0.035	**0.334**±0.059	0.214±0.021	**0.641**±0.112	**0.358**±0.024
TableGAN	-	0.562±0.051	0.171±0.037	0.123±0.041	0.593±0.058	**0.199**±0.044
	Rnd	**0.565**±0.048	0.173±0.049	0.115±0.028	**0.594**±0.058	0.195±0.046
P-TableGAN	Corr	0.553±0.050	**0.174**±0.045	**0.125**±0.032	**0.594**±0.060	0.194±0.037
	KDE	0.556±0.060	0.171±0.042	0.119±0.025	0.592±0.060	0.193±0.042
CTGAN	-	0.822±0.017	0.362±0.033	0.247±0.087	0.736±0.035	**0.374**±0.034
	Rnd	0.819±0.014	0.364±0.032	**0.255**±0.089	0.735±0.038	0.365±0.021
P-CTGAN	Corr	0.814±0.008	**0.365**±0.038	0.246±0.087	**0.743**±0.032	0.372±0.033
	KDE	**0.823**±0.017	0.357±0.040	0.248±0.077	0.736±0.031	0.371±0.035
TVAE	-	0.810±0.008	0.282±0.029	**0.185**±0.021	0.735±0.010	**0.473**±0.016
	Rnd	0.810±0.011	0.266±0.020	0.172±0.011	**0.739**±0.004	0.464±0.019
P-TVAE	Corr	**0.815**±0.009	0.283±0.026	0.176±0.028	0.730±0.006	0.462±0.018
	KDE	**0.815**±0.009	**0.285**±0.034	0.171±0.030	0.735±0.007	0.463±0.021
GOGGLE	-	0.622±0.094	0.189±0.038	0.163±0.119	0.596±0.072	0.152±0.003
	Rnd	**0.626**±0.098	**0.193**±0.042	0.164±0.121	0.600±0.060	0.151±0.007
P-GOGGLE	Corr	0.623±0.089	0.191±0.038	**0.169**±0.126	**0.601**±0.063	**0.155**±0.010
	KDE	**0.626**±0.096	0.189±0.039	**0.169**±0.126	0.597±0.066	**0.155**±0.011

Table 25: Weighted F1-score utility results with their corresponding stddevs for binary and multi-class classification datasets when post-processing the unconstrained predictions.

		URL	WiDS	LCLD	Heloc	FSP
WGAN	-	0.764±0.018	0.381±0.057	**0.359**±0.019	0.599±0.050	0.339±0.021
	Rnd	0.768±0.028	0.372±0.052	0.342±0.019	0.600±0.057	0.337±0.022
P-WGAN	Corr	**0.769**±0.027	0.384±0.048	0.352±0.014	0.598±0.054	**0.341**±0.026
	KDE	0.762±0.028	**0.386**±0.057	0.352±0.014	**0.603**±0.059	**0.341**±0.026
TableGAN	-	0.659±0.035	0.240±0.034	**0.286**±0.029	**0.615**±0.030	**0.199**±0.043
	Rnd	**0.660**±0.035	**0.243**±0.045	0.281±0.022	0.614±0.028	0.195±0.046
P-TableGAN	Corr	0.657±0.035	**0.243**±0.041	0.285±0.023	0.614±0.030	0.195±0.036
	KDE	0.658±0.040	0.240±0.039	0.285±0.018	0.613±0.029	0.193±0.042
CTGAN	-	0.799±0.033	0.405±0.034	0.379±0.061	0.675±0.015	**0.372**±0.034
	Rnd	0.794±0.031	**0.407**±0.032	**0.383**±0.070	0.671±0.015	0.363±0.019
P-CTGAN	Corr	0.801±0.008	**0.407**±0.040	0.379±0.068	**0.678**±0.010	0.370±0.033
	KDE	**0.802**±0.032	0.399±0.042	0.380±0.058	0.671±0.010	0.368±0.033
TVAE	-	0.802±0.012	0.342±0.027	**0.330**±0.016	**0.696**±0.006	**0.463**±0.017
	Rnd	0.802±0.015	0.327±0.018	0.324±0.009	**0.696**±0.004	0.454±0.020
P-TVAE	Corr	**0.806**±0.008	0.342±0.024	0.325±0.018	0.691±0.004	0.452±0.019
	KDE	**0.806**±0.008	**0.344**±0.031	0.322±0.020	0.694±0.004	0.453±0.022
GOGGLE	-	0.648±0.074	0.198±0.067	0.315±0.086	**0.566**±0.050	0.139±0.002
	Rnd	0.645±0.071	**0.202**±0.070	0.315±0.086	**0.566**±0.047	0.139±0.006
P-GOGGLE	Corr	0.645±0.070	0.201±0.067	**0.318**±0.090	**0.566**±0.047	**0.142**±0.009
	KDE	**0.650**±0.072	0.197±0.069	**0.318**±0.090	0.565±0.048	0.141±0.011

Table 26: Area under the ROC curve utility results with their corresponding stddevs for binary and multiclass classification datasets when post-processing the unconstrained predictions.

		URL	WiDS	LCLD	Heloc	FSP
WGAN	-	0.839±0.016	0.775±0.038	**0.618**±0.011	0.677±0.033	**0.742**±0.007
	Rnd	**0.840**±0.019	0.777±0.037	**0.618**±0.010	0.682±0.042	0.736±0.011
P-WGAN	Corr	0.839±0.020	0.782±0.035	0.617±0.008	**0.683**±0.038	0.740±0.010
	KDE	0.834±0.021	**0.783**±0.039	0.617±0.008	0.682±0.041	0.740±0.010
TableGAN	-	0.843±0.020	0.740±0.021	**0.587**±0.027	**0.707**±0.007	0.642±0.026
	Rnd	0.848±0.019	**0.749**±0.019	0.585±0.027	0.703±0.008	0.641±0.030
P-TableGAN	Corr	0.846±0.020	0.731±0.022	**0.587**±0.028	**0.707**±0.011	**0.645**±0.031
	KDE	**0.855**±0.016	0.736±0.016	**0.587**±0.027	**0.707**±0.011	0.644±0.035
CTGAN	-	0.859±0.040	0.835±0.012	0.651±0.020	0.744±0.009	**0.760**±0.014
	Rnd	0.859±0.038	**0.837**±0.012	**0.652**±0.024	0.743±0.008	0.753±0.008
P-CTGAN	Corr	**0.866**±0.007	0.835±0.012	0.650±0.023	**0.745**±0.008	0.754±0.011
	KDE	0.863±0.035	0.836±0.012	0.651±0.018	0.743±0.009	0.753±0.009
TVAE	-	0.863±0.011	**0.800**±0.016	**0.631**±0.004	**0.752**±0.003	**0.789**±0.007
	Rnd	0.866±0.013	0.785±0.017	0.630±0.007	0.751±0.003	0.787±0.011
P-TVAE	Corr	**0.870**±0.007	0.787±0.009	0.629±0.004	**0.752**±0.004	0.779±0.014
	KDE	**0.870**±0.007	0.797±0.016	0.630±0.002	0.750±0.001	0.788±0.010
GOGGLE	-	**0.742**±0.071	0.656±0.049	0.543±0.039	0.600±0.056	0.577±0.010
	Rnd	0.738±0.067	0.665±0.043	**0.548**±0.035	0.602±0.056	0.575±0.008
P-GOGGLE	Corr	0.740±0.063	**0.667**±0.044	0.542±0.036	0.601±0.058	0.576±0.010
	KDE	0.741±0.067	**0.667**±0.060	0.542±0.036	**0.603**±0.058	**0.578**±0.014

Table 27: Utility performance comparison between DGM and P-DGM models trained on News.

		XV	MAE
WGAN	-	-0.006±0.003	3133.6±242.1
	Rnd	**-0.005**±0.002	3151.3±219.8
P-WGAN	Corr	-0.012±0.011	**3109.2**±220.4
	KDE	-0.006±0.003	3159.7±210.7
TableGAN	-	**-0.001**±0.001	2992.2±35.8
	Rnd	**-0.001**±0.001	**2981.9**±32.3
P-TableGAN	Corr	**-0.001**±0.001	3015.3±43.9
	KDE	**-0.001**±0.001	3022.7±23.6
CTGAN	-	**-0.002**±0.001	3043.8±37.8
	Rnd	-0.003±0.004	3013.4±36.2
P-CTGAN	Corr	-0.006±0.002	**2999.3**±27.3
	KDE	-0.003±0.003	3004.8±47.4
TVAE	-	**-0.001**±0.000	3021.3±10.0
	Rnd	**-0.001**±0.001	3040.5±60.4
P-TVAE	Corr	-0.002±0.001	**3003.8**±22.7
	KDE	**-0.001**±0.001	3032.2±39.3
GOGGLE	-	**-0.004**±0.003	3054.5±44.7
	Rnd	-0.006±0.005	3024.1±37.2
P-GOGGLE	Corr	-0.005±0.002	3077.9±54.1
	KDE	**-0.004**±0.003	**3012.3**±36.6

Table 28: Binary F1-score detection results with their corresponding stddevs for all datasets when post-processing the unconstrained predictions.

		URL	WiDS	LCLD	Heloc	FSP	News
WGAN	-	0.865±0.035	**0.975**±0.002	0.999±0.001	**0.964**±0.004	0.914±0.018	**0.954**±0.022
	Rnd	**0.856**±0.037	0.979±0.003	**0.916**±0.008	**0.964**±0.004	**0.910**±0.015	**0.954**±0.022
P-WGAN	Corr	0.857±0.033	0.977±0.002	0.921±0.004	0.968±0.005	0.911±0.018	**0.954**±0.026
	KDE	0.862±0.037	0.978±0.003	0.921±0.004	0.967±0.004	0.911±0.018	0.957±0.019
TableGAN	-	**0.831**±0.029	0.963±0.006	**0.895**±0.024	0.923±0.011	0.909±0.021	0.927±0.009
	Rnd	0.843±0.034	**0.962**±0.006	0.896±0.026	0.923±0.010	0.858±0.031	0.932±0.010
P-TableGAN	Corr	0.833±0.035	0.963±0.005	0.900±0.024	0.922±0.011	0.860±0.029	**0.920**±0.009
	KDE	0.835±0.041	**0.962**±0.006	0.901±0.019	**0.921**±0.011	**0.857**±0.029	0.928±0.004
CTGAN	-	0.850±0.027	0.990±0.002	0.848±0.016	0.914±0.024	0.926±0.011	0.901±0.018
	Rnd	**0.845**±0.028	0.990±0.002	**0.839**±0.015	0.916±0.028	0.902±0.029	**0.898**±0.012
P-CTGAN	Corr	0.855±0.014	**0.989**±0.003	0.845±0.019	**0.913**±0.028	**0.895**±0.030	**0.898**±0.006
	KDE	0.850±0.023	0.991±0.002	0.841±0.019	**0.913**±0.027	**0.895**±0.032	0.902±0.010
TVAE	-	0.813±0.037	**0.926**±0.001	0.842±0.019	0.914±0.011	**0.843**±0.027	0.877±0.013
	Rnd	**0.806**±0.039	0.962±0.002	0.835±0.018	0.914±0.007	0.881±0.007	0.873±0.012
P-TVAE	Corr	**0.806**±0.044	0.962±0.001	**0.829**±0.022	0.912±0.011	0.881±0.011	**0.866**±0.014
	KDE	**0.806**±0.044	0.964±0.001	0.837±0.010	**0.909**±0.011	0.874±0.012	0.868±0.012
GOGGLE	-	0.892±0.019	**0.987**±0.018	**0.890**±0.017	**0.924**±0.006	0.912±0.010	**0.949**±0.023
	Rnd	**0.884**±0.025	0.988±0.017	0.892±0.012	0.927±0.006	**0.907**±0.005	0.952±0.021
P-GOGGLE	Corr	0.897±0.010	0.988±0.016	0.902±0.005	0.926±0.006	0.909±0.006	0.951±0.021
	KDE	0.898±0.015	**0.987**±0.017	0.902±0.005	0.928±0.007	**0.907**±0.009	0.951±0.020

Table 29: Weighted F1-score detection results with their corresponding stddevs for all datasets when post-processing the unconstrained predictions.

		URL	WiDS	LCLD	Heloc	FSP	News
WGAN	-	0.856±0.008	**0.976**±0.002	0.999±0.001	**0.964**±0.004	0.910±0.027	**0.954**±0.020
	Rnd	**0.852**±0.011	0.979±0.003	**0.907**±0.010	0.965±0.004	**0.907**±0.009	0.957±0.020
P-WGAN	Corr	0.853±0.008	0.977±0.002	0.914±0.002	0.968±0.004	0.912±0.018	0.955±0.025
	KDE	0.855±0.010	0.978±0.003	0.914±0.002	0.967±0.004	0.912±0.018	0.959±0.017
TableGAN	-	**0.822**±0.007	0.964±0.006	0.895±0.022	0.925±0.011	0.906±0.028	0.929±0.013
	Rnd	0.823±0.005	**0.963**±0.006	**0.888**±0.031	0.924±0.011	0.873±0.018	0.934±0.012
P-TableGAN	Corr	0.826±0.006	0.964±0.005	0.898±0.025	0.924±0.011	**0.868**±0.016	**0.923**±0.009
	KDE	0.824±0.005	**0.963**±0.006	0.896±0.021	**0.923**±0.011	0.873±0.022	0.928±0.006
CTGAN	-	0.862±0.014	0.990±0.002	0.850±0.019	0.915±0.023	0.927±0.016	0.896±0.023
	Rnd	**0.859**±0.017	0.990±0.002	**0.840**±0.014	0.917±0.027	0.905±0.026	**0.894**±0.006
P-CTGAN	Corr	0.860±0.012	**0.989**±0.003	0.850±0.009	0.914±0.027	**0.902**±0.017	**0.894**±0.014
	KDE	0.861±0.017	0.991±0.002	0.847±0.006	**0.913**±0.027	0.905±0.025	0.899±0.014
TVAE	-	0.831±0.013	**0.927**±0.001	0.832±0.010	0.916±0.010	**0.847**±0.006	0.855±0.019
	Rnd	**0.828**±0.012	0.963±0.002	0.832±0.010	0.915±0.007	0.877±0.010	0.856±0.017
P-TVAE	Corr	0.829±0.012	0.963±0.001	**0.825**±0.011	0.914±0.011	0.882±0.012	0.860±0.015
	KDE	0.829±0.012	0.964±0.001	0.831±0.014	**0.910**±0.011	0.870±0.016	**0.848**±0.014
GOGGLE	-	0.880±0.012	**0.987**±0.018	0.892±0.014	**0.926**±0.005	0.916±0.011	**0.955**±0.019
	Rnd	0.878±0.018	0.988±0.017	**0.889**±0.015	0.928±0.005	0.911±0.007	0.957±0.017
P-GOGGLE	Corr	**0.875**±0.021	0.988±0.016	0.901±0.006	0.927±0.005	0.907±0.012	0.957±0.017
	KDE	0.883±0.011	**0.987**±0.017	0.901±0.006	0.929±0.006	**0.906**±0.011	0.956±0.017

Table 30: Area under the ROC curve detection results with their corresponding stddevs for all datasets when post-processing the unconstrained predictions.

		URL	WiDS	LCLD	Heloc	FSP	News
WGAN	-	0.872±0.008	**0.989**±0.002	1.000±0.000	**0.983**±0.004	**0.916**±0.016	**0.964**±0.021
	Rnd	**0.867**±0.003	0.992±0.002	**0.936**±0.010	0.985±0.002	0.927±0.006	0.966±0.020
P-WGAN	Corr	0.869±0.004	0.991±0.002	0.942±0.001	0.985±0.003	0.931±0.013	0.966±0.024
	KDE	0.870±0.003	0.991±0.002	0.942±0.001	0.985±0.002	0.931±0.013	0.969±0.017
TableGAN	-	0.850±0.007	0.980±0.004	0.926±0.020	0.953±0.009	0.907±0.022	0.940±0.011
	Rnd	0.850±0.006	0.980±0.004	**0.925**±0.024	0.953±0.009	0.894±0.016	0.946±0.012
P-TableGAN	Corr	**0.848**±0.007	0.980±0.003	0.929±0.023	0.953±0.008	**0.886**±0.018	**0.937**±0.011
	KDE	0.852±0.009	**0.979**±0.003	0.933±0.016	**0.952**±0.008	0.890±0.022	0.939±0.006
CTGAN	-	0.879±0.008	**0.996**±0.001	0.896±0.011	0.953±0.016	0.932±0.007	**0.912**±0.021
	Rnd	**0.876**±0.012	0.997±0.001	**0.892**±0.008	0.953±0.021	0.925±0.019	0.915±0.012
P-CTGAN	Corr	0.878±0.004	0.997±0.001	0.896±0.005	**0.952**±0.018	**0.922**±0.013	0.921±0.017
	KDE	**0.876**±0.013	0.998±0.001	0.893±0.004	**0.952**±0.020	0.925±0.017	0.919±0.021
TVAE	-	0.854±0.007	**0.935**±0.002	0.861±0.009	0.947±0.005	**0.872**±0.008	0.881±0.019
	Rnd	**0.849**±0.010	0.978±0.002	0.862±0.008	0.948±0.004	0.907±0.008	0.887±0.021
P-TVAE	Corr	0.850±0.009	0.979±0.001	**0.858**±0.007	0.946±0.007	0.910±0.011	0.884±0.014
	KDE	0.850±0.009	0.979±0.001	0.861±0.005	**0.944**±0.006	0.895±0.013	**0.874**±0.018
GOGGLE	-	0.891±0.009	**0.993**±0.013	0.928±0.010	**0.949**±0.003	0.920±0.006	**0.973**±0.013
	Rnd	**0.889**±0.017	0.994±0.013	**0.923**±0.008	0.952±0.004	0.924±0.011	0.976±0.012
P-GOGGLE	Corr	0.895±0.010	0.994±0.012	0.931±0.005	0.952±0.003	0.919±0.011	0.977±0.011
	KDE	0.894±0.017	**0.993**±0.012	0.931±0.005	0.952±0.003	**0.917**±0.010	0.976±0.012

C.6 Real data performance

To ensure that the comparisons between our C-DGM models and the baseline unconstrained DGMs are meaningful, we conducted a hyperparameter search as detailed earlier in Section B.5, allowing us to get close to (and sometimes even surpass) the real data utility performance. We report the latter in Table 31 using the same three metrics we used for measuring the synthetic data utility performance (i.e., F1-score, weighted F1-score, and Area Under the ROC Curve) following the same protocol as the one described in Appendix B.4. Comparing the results here with those for the synthetic data in Tables 17-19 we can see that, overall, C-WGAN and C-CTGAN models got utility scores similar to those obtained on the real data. For C-TableGAN and C-TVAE we notice several cases where our method helped in bringing the performance of the synthetic data closer to the real data. In particular, for LCLD we notice that the real data got an F1-score 4.8% higher than the unconstrained TableGAN, but our C-TableGAN model was able to match the real data performance (scoring slightly better than it, i.e., by 0.3%). It is also worth noticing that the gap between real and synthetic performance was reduced the most for the LCLD and WiDS datasets, both of which are highly unbalanced. For instance, all C-WGAN, WGAN, C-CTGAN, and CTGAN models yield a higher F1-score than the real LCLD data. We notice similar trends for the other metrics, weighted F1 and Area Under the ROC Curve. On the other hand, none of the DGM or C-DGM models get close to the real FSP data. One possible reason is the datasets' small size and multiclass nature, which makes it harder for the DGM models to capture patterns that lead to the correct different targets.

Table 31: Utility scores calculated on real data.

	F1	*w*F1	AUC
URL	0.884±0.007	0.875±0.014	0.903±0.009
WiDS	0.383±0.021	0.434±0.020	0.832±0.009
LCLD	0.171±0.030	0.316±0.013	0.645±0.007
Heloc	0.772±0.003	0.662±0.011	0.707±0.008
FSP	0.662±0.011	0.659±0.009	0.848±0.010