Ellipse: Robust and imperceptible watermarking for tabular diffusion models

Toma Volentir[1]

Supervisor(s): Dr. Lydia Y. Chen[1]**, Jeroen M. Galjaard**[1]**, Chaoyi Zhu**[1]

[1]**EEMCS, Delft University of Technology, The Netherlands**

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 23, 2024

Name of the student: Toma Volentir
Final project course: CSE3000 Research Project
Thesis committee: Rihan Hai, Dr. Lydia Y. Chen, Jeroen M. Galjaard, Chaoyi Zhu

Abstract

Data in the form of tables is commonly used in the scientific and research industry, as it provides a compact, easy-to-understand and logical way of storing data. The advancement of diffusion models has significantly improved the quality of generated tabular data, but it also poses risks of misappropriation and copyright concerns. Thus, there is a need to control and monitor the data generated by diffusion models, to enable harm mitigation and protect intellectual property. This paper addresses the necessity for robust watermarking techniques specifically designed for tabular data generated by diffusion models. We propose `Ellipse`, a generalization of the Tree-Ring watermarking method, originally developed for square-shaped images, to handle rectangular shaped tables. We change the shape of the watermark from a circle—fit for square-shaped images—to an oval—fit for datasets of rectangle shape. Through comprehensive experiments on four real world datasets (Abalone, Adult, Default, and Diabetes), we demonstrate that the adapted watermarking technique has a negligible drop of 3.5% in data quality, measured through correlations between real and synthetic distributions, performance of downstream machine learning tasks, and discriminability between the real and synthetic data. This is a better result than the 12.46% drop in data quality offered by having a circle mask. `Ellipse` introduces a non-significant average drop of 0.4% in detection efficiency compared to having a circle mask. Our implementation also offers resilience against value skewing and deletion attacks on the rows and columns of the dataset. When exposed to attacks, `Ellipse` has a higher Area Under the Curve (AUC) than the circular mask of Tree-Ring by an average of 7.17%. The code for `Ellipse` is publicly available at https://github.com/6toma/ellipse-watermark.

1 Introduction

Recent advancements in diffusion models, like *Midjourney* [5], have led to an increase in the quality of data generation. This increased data fidelity, in turn, has led to an increase in abusive generation of content with malicious purposes, which motivates the development of *watermarks* for the output of diffusion models. Watermarks have multiple purposes, such as protecting the intellectual property of others. Another purpose is to mark content as being machine-generated, enabling harm mitigation, by identifying the source of the generated media—is it original/real, or is it fake/generated?

Algorithms have been created that watermark the output of diffusion models that generate images [15; 24; 26; 22; 30]. They are robust, imperceptible by humans and detectable by machines. However, they have trade-offs such as not being invariant to all types of post-editing attacks on the generated data. The current diffusion model watermarking techniques are intended for image-generating models, more specifically square-shaped images.

However, the issue is that watermarking for diffusion based tabular models has not yet been researched. For the same reason we need watermarking for images created by diffusion models (or models in general)—to mark content as artificially generated, enable harm mitigation, and protect intellectual property—we also need watermarking for tabular data generated by diffusion models. The aim is to reduce the risk of distributing harmful data generated by diffusion models.

State-of-the-art diffusion models and watermarking methods for them are centred around image generation. Because watermarking tabular diffusion models has not been done before, we aim to leverage existing image-based watermarking methods for diffusion models and adapt them for tabular data. For this purpose, we propose using the state-of-the-art Tree-Ring [26] watermark to a tabular diffusion model. However, Tree-Ring is **not** already fit for tabular data, because it is meant for watermarking images. When obtaining new images via diffusion, most common sizes for the output are 32×32, 64×64, and 256×256 [14; 23]. Thus, Tree-Ring has to be adapted to fit tabular data, which is commonly rectangular-shaped—the two dimensions being the number of columns and number of rows. Table data also has predominantly more rows than columns, making it have a narrow and long rectangular shape. Thus, we cannot simply add a bigger circular watermark on a dataset, as we risk impacting data quality significantly, at the cost of having strong detection. The main challenge is to adapt Tree-Ring to a tabular diffusion model, to verify its effectiveness as a watermarking algorithm for diffusion tabular synthesis.

We propose `Ellipse`, a watermarking algorithm, which changes the shape of the Tree-Ring *circle* mask to an *oval* mask, which better fits for tabular data. `Ellipse` addresses the following research question: *how to adapt the Tree-Ring algorithm for the synthetic tables generated by tabular diffusion models, that has a negligible impact on the synthetic data quality, is imperceptible by humans but detectable by the model owner, and has robust detection against post-editing of tables?* `Ellipse` first generates an oval shaped watermark patch and embeds it in the Fourier space of the noise vector needed for the diffusion pipeline. Afterwards, it samples a new table using the noise vector. To detect the watermark, we invert the sampling process to recover the initial noise vector, and check the L1 distance between our watermark patch and the recovered noise vector. It requires no training or fine-tuning, making it computationally inexpensive. It embeds the watermark in the Fourier space, making it imperceptible to humans. We bring the following contributions:

Negligible drop in data quality: Our proposed watermark `Ellipse` shows a negligible drop of 3.5% to data quality in terms of correlations between real and synthetic distributions, performance of downstream machine learning tasks (machine learning efficiency), and discriminability between real and synthetic data. This is a better result than the 12.46% drop in quality offered by a circle mask. `Ellipse` also has a lower impact on machine learning efficiency than on other metrics. `Ellipse`'s oval mask also outperforms Tree-Ring's circle

mask in machine learning efficiency on the bigger datasets, namely Adult and Default, by an impressive 27%.

Detection efficiency: `Ellipse` has an average Area Under the Curve (AUC) of 0.9 out of 1.00 , showing effective detection. Our implementation shows a negligible average drop of 0.4% in detection efficiency, measured by AUC, compared to the original circle mask . We showcase that $\texttt{Ellipse}_{rand}$ and $\texttt{Ellipse}_{zeros}$ work best on datasets with less columns, while $\texttt{Ellipse}_{ring}$ performs best on classification task datasets.

Robustness against post-editing attacks: We show that $\texttt{Ellipse}_{rand}$ is unaffected by value skewing attacks—e.g., adding random noise or replacing values—on the data, maintaining an AUC of at worst 2.5% lower than the clean version, and at best an AUC of 1.00. We also demonstrate that the watermark is still detectable against row and column deletion attacks. `Ellipse` has, on average, an AUC that is 7.17% higher than Tree-Ring's circular mask against four types of attacks.

Regarding the choice of tabular diffusion model, we have chosen to adapt Tree-Ring to Tabsyn [29], a state-of-the-art tabular data synthesizer with score-based diffusion in latent space. Tabsyn offers generality towards data types, high-quality synthetic data, and fast synthesis speed.

2 Related Works

Recent efforts in generative modeling of tabular data predominantly focus on the usage of foundational (transformer-based), and diffusion-based modelling. These works show the ability to synthesize realistic tabular data that exceed earlier works such as CTGAN [27] and TVAE [16]. The success of these models have prompted malicious use of synthetic data—such as spreading generated data and claiming it to be real—leading to misinformation. Thus, research on watermarking has also been carried out, targeting generative models due to their impressive quality in data synthesis.

Generative tabular models Deep generative models have more notable success with computer vision and natural language processing than tabular data. However, tabular models have seen impressive improvements through generative adversarial network (GAN) [12] models. Up until recently, GANs, such as CTGAN [27], were considered to be state-of-the-art generative models, which were able to deal with imbalanced categorical features. Still, more recent advancements led to novel methods for synthetic tabular data generation. For example, diffusion models such as TabDDPM [20], STaSy [17], or Tabsyn [29], which "describe the spread of information, behaviors, or phenomena through a population and identify the factors influencing the diffusion process" [22], are considered state-of-the-art in terms of quality of the sampled data [11] and density estimation [18]. GOGGLE [21] is a large language model (LLM) based model, being the first to "model the dependency relationship between columns" [29]. Following the success of using LLM models to synthesize tabular data, GReaT [9] transforms the rows of a table into natural sentences, to then learn the sentence-level distributions with GPT2 [29]. Innovative sampling algorithms, such as Denoising Diffusion Probabilistic Models (DDPM) [14], use two Markov chains to progressively add, then remove Gaussian noise to obtain higher quality data. Denoising Diffusion Implicit Models (DDIM) [25] represents an improvement to DDPM, producing qualitative data in fewer steps. DDIM does not add additional noise like DDPM does, making the sampling process deterministic and invertible.

Watermarking generative models Generative models produce high-quality data and have state-of-the-art performance in synthesis tasks, but they are not "secure by design" [22]. Thus, generative models can be used with malicious purposes, such as spreading synthetic, model-generated data and claiming it to be real. Watermarking algorithms help relieve this problem by marking the data as machine-generated. Most common state-of-the-art watermarking methods are "green list" based [19; 13], or prompt based [22; 30]. The "green list" based watermarking normalizes the data and divides the range into equal intervals, selecting a subset to form a "green list" [13]. For each data point, the fractional part is checked against the green list intervals, and if it does not match, it is replaced with a value from the nearest green list interval. This ensures watermark embedding while maintaining data integrity. Detection is performed through statistical hypothesis testing to confirm the watermark's presence.

The prompt based procedure is common among diffusion models. The prompt based watermarking involves injecting a watermark into the model during the training or fine-tuning phase—depending on the implementation. Afterwards, a dataset with pairs of watermarked prompts (prompts that include the trigger word) and corresponding watermarked data, as well as pairs of clean prompts with clean data is created. This dataset will be used for training or fine-tuning, hereby integrating the watermark into the model's parameters. The watermark can later be detected by using a prompt that contains the trigger word in any position or, for a more sophisticated approach, a prompt with the trigger in a specific location.

Watermarking tabular diffusion models has not been studied previously, and the current available watermarking methods do not solve the issue of watermarking tabular diffusion models. The green-list approach has been shown to have an impact on the data quality, showing small degradations in fidelity [13]. It is also extremely vulnerable to random noise attacks on the numerical columns. The prompt-based approach is computationally expensive [22], prone to bias [22], and also cannot be trivially adapted, as tabular diffusion models do not take any prompt as input.

3 Preliminary

Diffusion models A diffusion model has three components, an *encoder*, the *diffusion pipeline*, and a *decoder*. The *encoder* takes as input the clean data and encodes it into vectors, called *latents*. The *decoder* takes as input the latents and outputs the new, synthesized data The *diffusion pipeline* works by simulating two Markov chains, namely a forward and a backwards process [10]. The forward process takes a clean data sample—in the form of latents—and sequentially corrupts it with random Gaussian noise to, in an infinite-time limit, transform it into pure noise [10]. The backward process attempts to reconstruct new clean data by sequentially removing the noise from the latents using a neural network [10].

Tree-Ring watermarking Tree-Ring [26] proposes embedding a pattern, also known as the *key*, into the Fourier space of the latents used for sampling *before* applying the diffusion process. The Fourier space is an abstract dimension in which the values of the data are mapped to their frequencies, making it invisible to human eyes. Thus, by adapting Tree-Ring to table data, we already start with an imperceptible watermark. A circular mask is applied on the latents, then the key is put over the mask. The key is embedded into the Fourier space, allowing the watermark to be robust, or even invariant, to certain transformations, such as rotations, flips, or crops. Tree-Ring allows three patterns for the key, each with certain advantages:

- zeros—an array of zeros invariant to certain post-editing attacks, but values dissimilar to Gaussian distribution
- rand—an array of values drawn from a Gaussian distribution, but not invariant to most post-editing attacks
- ring—multiple concentric rings with constant values drawn from a Gaussian distribution and invariant to certain post-editing attacks

The *ring* pattern shows promising robustness, allowing detection even after a series of post-editing attacks on the generated content [26]. The key can later be detected through inverting the diffusion process to retrieve the latents, then checking the distance between this and the embedded pattern. Thus, Tree-Ring requires a deterministic and invertible diffusion process, which is provided by implementing DDIM as the diffusion sampling process.

4 `Ellipse`: Elliptical Rings for Rectangular Latents

We *generalize* the Tree-Ring watermark by introducing elliptical rings which are compatible with any rectangular shaped latents. We do this by changing the shape of the watermark to an oval instead of a circle, introducing two radii instead of one. The radii are based on the 'height' and 'width' of the latents. Thus, when the two radii are equal—the latents are square shaped—the resulting watermark is the original Tree-Ring, in the same way a rectangle becomes a square when all its sides are equal. The pipeline for watermarking the diffusion latents is illustrated in Figure 1.

We want to benchmark the performance of Tree-Ring when used on a tabular diffusion model, where the shape of the data is rectangular. Tree-Ring is intended to be used for square-shaped image synthesis tasks, and is thus not compatible with tabular data, which has a rectangular shape. Consequently, we first adapt Tree-Ring's implementation [7] to Tabsyn's implementation [6]. Tabsyn uses score-based sampling, but Tree-Ring requires a deterministic and invertible sampling process to obtain the initial latents. Thus, we implement DDIM sampling for Tabsyn. Still, Tree-Ring uses a circle mask to watermark the intended input of square shaped images. Thus, we have to adapt the circle mask to a more suitable shape for the rectangular shaped tabular data, namely an oval. Finally, we change the shape of the latents from 2-dimensional to 4-dimensional. Image-based latents are 4-dimensional, having a batch size, the number of channels, the width and the height, while table-based latents only have the height and width as dimensions. After making the aforementioned modifications to make Tabsyn and Tree-Ring compatible, we adapt the watermark injection and detection algorithms to fit tabular data.

Watermarking pipeline We inject and detect a watermark following a certain pipeline, which is shown in Figure 1. Due to the properties of the diffusion model, insertion and detection of a watermark bear similarity, differing mainly in the order in which the process is performed. First, we generate a watermark key based on a given pattern (*ring*, *rand*, or *zeros*). We use this key to watermark the latents in the Fourier space. Afterwards, the latents go through the data sampling process—in this case, DDIM—to obtain new, synthesized latents, which are decoded back to CSV format. The detection process is similar, but in reverse, starting with the encoding of the CSV into latents. Consecutively, the DDIM inverse process is applied on the synthetic latents to obtain the initial latents. In the Fourier space again, we look at where the watermark key would be located on the latents and check the L1 distance between this section of the latents and the predefined key. If the L1 distance falls below an empirically chosen threshold τ, the data is considered *watermarked*.

Adapting the shape of the Tree-Ring mask to fit tabular data As previously mentioned, Tree-Ring was meant for square shaped data, but tabular data has a long and narrow rectangular shape with its height being much longer than its width. In order to have an appropriate size for our watermark, `Ellipse` modifies the shape of the mask from a circle to an ellipse. Thus, we consider the following implementation from Algorithm 1 for obtaining an elliptical mask. We first extract the height and the width from the shape of the latents, to determine the center of the shape. Afterwards, we use the equation of an ellipse on line 7 to determine where the circumference is—i.e., where to place the ring.

Algorithm 1 Implementation of elliptical mask to fit any 2D shape

1: **function** ELLIPTICALMASK(shape, x_radius, y_radius, x_offset, y_offset)
2: $height, width \leftarrow shape$
3: $y, x \leftarrow$ grid of coordinates from 0 to $height - 1, width - 1$
4: $y \leftarrow$ reverse order of y
5: $x\_center \leftarrow \frac{width}{2} + x\_offset$
6: $y\_center \leftarrow \frac{height}{2} + y\_offset$
7: **return** $\left(\frac{(x-x\_center)^2}{x\_radius^2} + \frac{(y-y\_center)^2}{y\_radius^2}\right) \leq 1$
8: **end function**

Re-shaping watermarks requires additional steps, due to the rectangular shape of the tabular latent, whereas image based diffusion models commonly utilize square latents [14; 23]. Most common sizes for image based latents include 32×32, 64×64, and 256×256 [14; 23].

`Ellipse` injection and detection methods As with the original implementation of Tree-Ring, we embed our water-

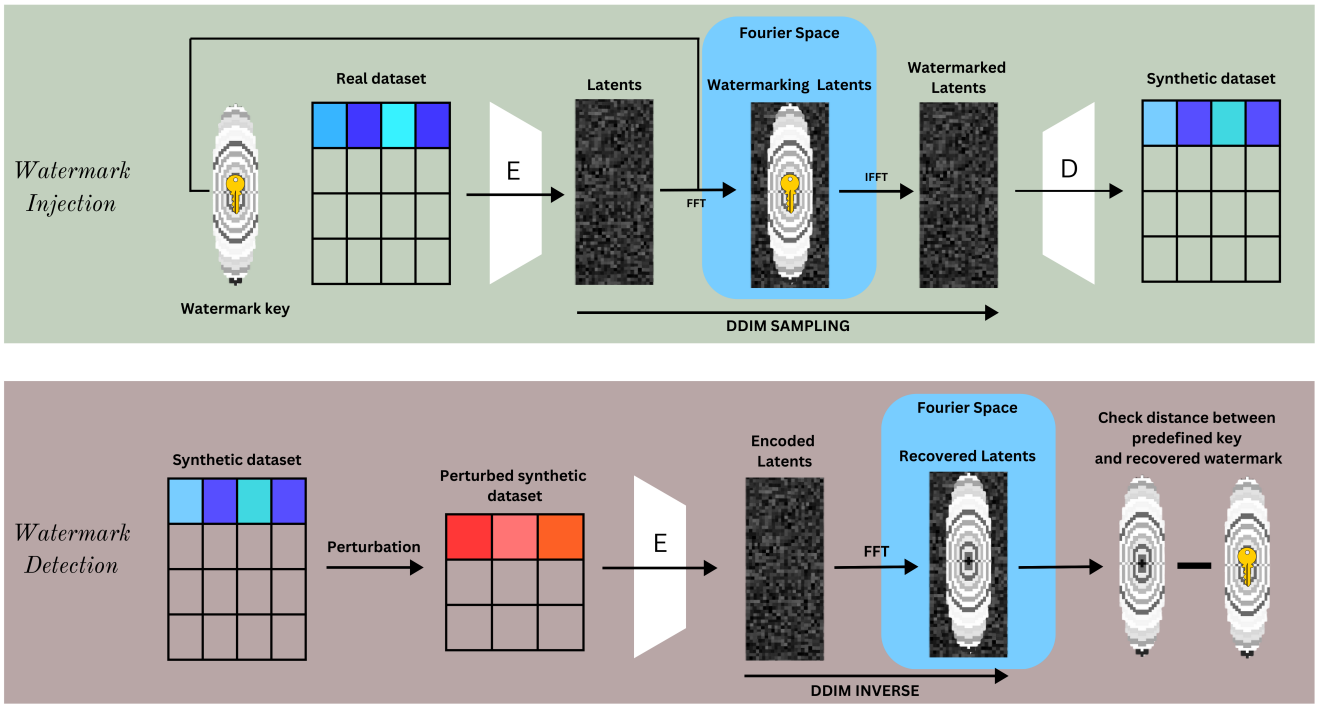

Figure 1: Pipeline for applying Tree-Ring Watermarking on diffusion latents. A diffusion model generation is watermarked and later detected through ring-patterns in the Fourier space of the initial noise vector.

mark in the Fourier space. This is because injecting the watermark directly in latent space might affect the correlations of the data and introduce statistical deviations from the original distribution. We start by generating our watermark key, through Algorithm 2. This algorithm generates a random watermark patch in the Fourier space through lines 2 and 3. Depending on the chosen pattern, further modifications are done to the watermark patch. More notably, for the 'ring' pattern, we create multiple concentrical ellipses. The watermark is *injected* in the Fourier space of the latents by first creating a binary elliptical mask. The latents undergo a 2D Fast Fourier Transform (FFT) with the zero-frequency component shifted at the center, on top of which the pattern is applied. After, the inverse Fourier Tranform (IFFT) yields the watermarked latents. After the sampling process, in Fourier space the algorithms embed part of the latents with the secret, thereby distributing the secret over the entire latents, 'hiding in plain sight'. The secret that overrides in Fourier Space is either an elliptical 'ring' pattern, or binary mask. The former consist of gradually smaller concentric *ellipses* (see Algorithm 1), centered around the origin in Fourier space. Whereas the latter binary mask sets overrides values at pre-determined indices, as shown in Algorithm 1. To inject the watermark, specific elements in the FFT latents are replaced by corresponding values from the watermark patch. More formally, given the latents L, watermark patch P and mask M, the watermark is applied on the Fourier transformed latent by:

$$L_{i,j} = \begin{cases} P_{i,j}, & \text{if } M_{i,j} = 1, \\ L_{i,j}, & \text{if } M_{i,j} = 0. \end{cases} \tag{1}$$

We *detect* the watermark by first reversing the sampling process to recover the initial latents. Again, we create the binary elliptical mask to see where precisely in the Fourier space to look for the watermark patch. Afterwards, the recovered latents are mapped to Fourier space. Here, the values at indices where the binary mask is present are considered to calculate the L1 distance between the watermark patch and the FFT recovered latents. The distance is then checked to be under a threshold τ, which is chosen empirically based on the dataset. Formally, given the recovered latents R, predefined watermark patch P, and mask M, we define the distance as:

$$distance = \frac{1}{|M|} \sum_{i,j\,;\,M_{i,j}=1} |P_{i,j} - R_{i,j}| \tag{2}$$

Choosing an appropriate threshold τ We use a quantitative study of the data to choose the appropriate threshold for the detection of the watermark. All datasets and `Ellipse` patterns have different properties that make them produce different distances between the recovered latents and the watermark keys. Thus, we choose to have a unique threshold on each pattern on each dataset—i.e., each dataset has three thresholds, one for each of the three `Ellipse` patterns. We run the pipeline described above 100 times on each pattern on each dataset, for a total of 1200 runs. We obtain enough data

Algorithm 2 Pattern Generation Algorithm

function GENERATEPATCH(shape, pattern)
  $gt\_init \leftarrow$ Random tensor of shape (shape)
  $gt\_patch \leftarrow$ fftshift(fft2(gt_init))
  if $pattern$ is 'zeros' **then**
    $gt\_patch \leftarrow gt\_patch \cdot 0$
  else if $pattern$ is 'ring' **then**
    $height, width \leftarrow shape$
    $min\_dim \leftarrow width \div 2$
    $max\_dim \leftarrow height \div 2$
    for $i \leftarrow min\_dim, 0, -1$ **do**
      $x\_rad \leftarrow i$
      $y\_rad \leftarrow max\_dim \times \frac{x\_rad}{min\_dim}$
      $tmp\_mask \leftarrow$ elliptical_mask(shape, x_rad, y_rad)
      for $j \leftarrow 0$ to gt_patch.shape[1] $- 1$ **do**
        $gt\_patch[:, j, tmp\_mask] \leftarrow gt\_patch[0, j, 0, i]$
      end for
    end for
  end if
  return gt_patch
end function

to generate ROC curves, to find suitable thresholds for the dataset-pattern combinations. The ROC curves can be seen in Figure 2 of Appendix C. We then pick the best threshold by limiting the false positive rate (FPR) at 1%, as is reported in Tree-Ring [26].

5 Evaluating `Ellipse`

Datasets We use four real-world datasets—two larger, and two smaller—which contain both numerical and categorical data. We use these datasets to evaluate the impact of the watermark on the quality of the synthetic data, the watermark's detection efficiency, and the watermark's robustness against post-editing attacks. We select as representative datasets the following: Abalone [1], Adult [2], Default [3], and Diabetes [4]. The statistical details and descriptions of these datasets can be found in Appendix A.

Evaluation metrics To benchmark the performance of `Ellipse`, we evaluate the impact on the quality of the synthetic data, the efficiency of the watermark detection, and the robustness to four different types of perturbations.

We assess the quality of the synthetic data using three metrics. We use *resemblance* (↑) to measure the correlation between the real and synthetic data. *Machine learning efficiency (MLE)* (↑) evaluates the performance of synthetic data on downstream tasks. For *MLE*, we train a model on synthetic data and test it on real data. Lastly, we use *discriminability* (↑) to measure the difficulty of a trained binary classifier to distinguish between real and synthetic data. In Appendix B a more detailed description of these metrics is available. The results of these scores can be seen in Table 1. An interesting, but expected, result is that the *rand* pattern performs best for the resemblance metric. This is expected as this pattern provides a patch of values sampled from a Gaussian distribution, the same as the unmodified latents.

The detection efficiency is calculated using two metrics, the results of which are available in Table 2. We use the *Area Under the Curve (AUC)* (↑) score for a classification task that distinguishes between non-watermarked and watermarked data. We also use the value of the *True Positive Rate (TPR)* (↑), when the False Positive Rate (FPR) is capped at 1%, as is done in Tree-Ring [26]. We note that $\texttt{Ellipse}_{rand}$ consistently has the highest TPR when the FPR is capped at 1%. The $\texttt{Ellipse}_{rand}$ pattern also maintains a high AUC score across all datasets, and an AUC score of 1.00 for three of the four datasets. The $\texttt{Ellipse}_{zeros}$ pattern also shows good performance—in terms of detection efficiency—across each of the four datasets.

Experimental setting The scores of the evaluation metrics are presented in Table 1, while the detection efficiency is showcased in Table 2. Tabular data has been sampled where the diffusion latents—needed for sampling tabular data using Tabsyn—have been watermarked in turn using the *rand*, *zeros*, and *ring* patterns. This process was then carried out on each dataset. We use 1000 inference steps for training and sampling the data. For the *ring* pattern, we use an x_radius equal to a quarter of the width of the latents, and a y_radius equal to a quarter of the height of the latents. We run the data quality tests 30 times on each pattern on each dataset, for a total of 720 trials. For the detection efficiency we run 100 trials on each pattern on each dataset, for a total of 2,400 runs.

Our method offers only a 3.5% drop in data quality, compared to Tree-Ring's circle mask having a higher 12.46% drop in quality. Interestingly, `Ellipse` outperforms Tree-Ring in MLE score by an impressive 27% for Adult and Default, which both have a large number of rows and columns compared to Abalone and Diabetes. It is notable that $\texttt{Ellipse}_{rand}$ generally has the best performance in terms of resemblance and detection efficiency. This pattern also has a maximum AUC of 1.00 for three of four datasets. $\texttt{Ellipse}_{zeros}$ also has an AUC of 1.00 for Abalone and Diabetes, which are smaller datasets, having less than 4.000 rows. We note that $\texttt{Ellipse}_{rand}$'s and $\texttt{Ellipse}_{zeros}$'s detection efficiency are smallest for the Default dataset, which has the highest number of columns, most of which numerical. Meanwhile, $\texttt{Ellipse}_{ring}$'s performs better on classification datasets, having an AUC 27% higher on average than on the regression dataset.

Robustness against post-editing attacks To assess the robustness of `Ellipse` against post-editing attacks we consider four possible distortions of the data. The results of the robustness to post-editing data perturbations can be seen in Table 3. `Ellipse` has a better overall robustness than Tree-Ring's circle mask, maintaining a high average AUC of 0.906 after each of the four different attacks, which is 7.17% higher than the circular mask. It is notable that the $\texttt{Ellipse}_{rand}$ pattern is immune to value skewing attacks, having a maximum AUC of 1.000 for all attack strengths. Given an attack strength of *s*, with *s* between 0 and 1, we consider the following attacks on tabular data:

1. **Numerical skewing** For each numerical column, we add random noise of strength *s* to all values in the column.

Table 1: Main results for the data quality, measured through resemblance, MLE, and discriminability. For each result, the means and standard deviations of 30 trials are presented. *Circle* represents the original Tree-Ring implementation, the baseline. `Ellipse` represents our implementation. The ↑ means that the aim is to maximize the score of the metrics. **Bold Face** represents the best score on each dataset, excluding the 'No watermark' baseline.

		No watermark	$Tree-Ring_{Rand}$		$Tree-Ring_{Zeros}$		$Tree-Ring_{Ring}$	
			Circle	`Ellipse`	Circle	`Ellipse`	Circle	`Ellipse`
Abalone	Resemblance ↑	90±0.0	**84±0.0**	**84±0.0**	76±0.0	82±0.0	77±0.0	82±0.0
	MLE ↑	95±1.394	94.6±0.596	94±0.73	85.8±1.301	**94.93±0.853**	93.2±0.853	94.8±0.748
	Discriminability ↑	85±0.636	82.9±0.512	**83.16±0.636**	74.7±0.64	81.8±0.832	75±0.813	81.76±0.76
Adult	Resemblance ↑	97±0.0	**91±0.0**	**91±0.0**	86±0.0	89±0.0	86±0.0	89±0.0
	MLE ↑	99±0.134	70.8±4.712	90.2±4.621	76.8±3.724	99.93±0.359	74.8±3.783	**100±0.0**
	Discriminability ↑	98±0.7	**96±0.0**	**96±0.0**	72.2±0.422	88.7±0.525	72.2±0.476	88.83±0.453
Default	Resemblance ↑	90±0.0	**91±0.0**	**91±0.0**	82±0.0	87±0.0	82±0.0	87±0.0
	MLE ↑	64.26±4.829	46.4±1.384	**63.83±4.2626**	42.4±1.113	63.8±4.819	42.6±1.38	62.76±5.541
	Discriminability ↑	96±0.0	**96±0.179**	**96±0.0**	64.86±0.498	88.3±0.458	65.4±0.611	88.6±0.489
Diabetes	Resemblance ↑	85±0.0	**85±0.0**	**85±0.0**	78±0.0	83±0.0	78±0.0	83±0.0
	MLE ↑	83.4±2.735	81.5±3.145	**83.3±2.793**	76.8±4.865	80.5±3.757	75.9±4.761	81.9±3.284
	Discriminability ↑	35.33±0.906	34.76±0.843	34.7±0.69	**35.66±1.324**	34.73±0.727	34.6±0.663	34.66±0.788

Table 2: Main results for the detection efficiency, measured through the AUC, and the T1F, which stands for the TPR when FPR is capped at 1%. *Circle* represents the original Tree-Ring implementation, the baseline. `Ellipse` represents our implementation. The ↑ means that the aim is to maximize the score of the metrics. **Bold Face** represents the best score on each dataset.

		$Tree-Ring_{Rand}$		$Tree-Ring_{Zeros}$		$Tree-Ring_{Ring}$	
		Circle	`Ellipse`	Circle	`Ellipse`	Circle	`Ellipse`
Abalone	AUC ↑	**1.00**	**1.00**	**1.00**	**1.00**	0.60	0.59
	T1F ↑	**1.00**	**1.00**	**1.00**	**1.00**	0.09	0.02
Adult	AUC ↑	**1.00**	**1.00**	0.99	0.99	0.81	0.79
	T1F ↑	**1.00**	**1.00**	0.90	0.89	0.03	0.10
Default	AUC ↑	0.89	0.86	0.87	**0.95**	0.82	0.86
	T1F ↑	0.48	0.48	**0.54**	0.48	0.35	0.48
Diabetes	AUC ↑	**1.00**	**1.00**	**1.00**	**1.00**	0.81	0.80
	T1F ↑	**1.00**	**1.00**	**1.00**	**1.00**	0.09	0.11

The random noise follows a Gaussian distribution.

2. **Categorical skewing** For each categorical column, we take *s*% of the values in the column and replace them with *s*% other values from the same column
3. **Row deletion** We delete *s*% of all total rows
4. **Column deletion** We delete *s*% of all total columns

For the deletion attacks, we have to be careful as the diffusion pipeline expects a certain shape of the latents. When deleting rows or columns, we essentially change the latents' shape, which will then become incompatible with the diffusion. Thus, we need to ensure the latents maintain the same shape. We add an extra step for the detection, where we change the shape of the data to be the same as in the input. If the data has undergone a deletion attack, we fill in the missing data in the following way:

- **Row deletion** To return to the original shape, we sample random rows from the current attacked data, essentially duplicating rows. We sample as many rows from the data as are missing from it after the attack.

Table 3: AUC under each of the four attacks, calculated as an average score of the AUC for each attack on the four individual datasets. Higher values are better. *Circle* represents the original Tree-Ring implementation, the baseline. `Ellipse` represents our implementation. *s* is the strength of the attack 'col' refers to column, 'num' refers to numerical, 'cat' refers to categorical, and 'del' stands for deletion. **Bold Face** represents the best average AUC on the four datasets.

	s	$Tree-Ring_{Rand}$		$Tree-Ring_{Zeros}$		$Tree-Ring_{Ring}$	
		Circle	`Ellipse`	Circle	`Ellipse`	Circle	`Ellipse`
Clean	–	0.972	0.965	0.965	**0.985**	0.760	0.760
Num skew	10%	0.900	**1.000**	**1.000**	**1.000**	0.822	0.920
	25%	0.940	**1.000**	0.902	0.960	0.827	0.927
Cat skew	10%	0.880	**1.000**	0.990	0.987	0.795	0.867
	25%	0.872	**1.000**	0.990	0.987	0.795	0.867
Row del	10%	0.747	0.867	**0.965**	0.952	0.717	0.800
	25%	0.742	0.815	**0.917**	0.900	0.805	0.725
Col del	10%	0.885	**0.982**	0.777	0.895	0.695	0.840
	25%	0.735	**0.905**	0.772	0.762	0.722	0.792

- **Column deletion** When deleting columns, we cannot sample other columns from the current attacked data. Instead, we sample columns from a synthetic, non-watermarked, non-attacked dataset. We copy the corresponding missing columns into our attacked table.

We run 100 trials on each of the six patterns on each dataset for the given attacks and their strengths, for a total of 19,200 runs. The presented results from Table 3 show the average AUC of the four datasets.

Impact of watermarking area on detection The size of the watermark patch is an important factor in the detection efficiency. If the size is too small, the patch becomes harder to detect. The size of our watermark depends on the radius scaling, the height and the width of the table. The radius scaling dictates the size of the x_{radius} and y_{radius} based on the height and width of the table. For example, when the radius scaling is equal to 1/2, the x_{radius} will be half the width, and the y_{radius} will be half the height. In Table 4 we aggregate

Table 4: Main results for the detection efficiency based on the radii used in generation, measured through the AUC, and the T1F, which stands for the TPR when FPR is capped at 1%. The results are averaged over the four datasets. RS stands for Radius Scaling, and represents the scaling used for the `Ellipse`'s oval mask radii—e.g., 1/2 means radii equal to half the width and half the length of the table were used. The ↑ means that the aim is to maximize the score of the metrics.

RS	Metric	$\texttt{Ellipse}_{rand}$	$\texttt{Ellipse}_{zeros}$	$\texttt{Ellipse}_{ring}$
1/2	AUC ↑	0.96	0.98	0.76
	T1F ↑	0.80	0.73	0.30
1/4	AUC ↑	0.96	0.98	0.76
	T1F ↑	0.87	0.84	0.17
1/8	AUC ↑	0.90	0.88	0.74
	T1F ↑	0.78	0.64	0.08
1/16	AUC ↑	0.94	0.80	0.70
	T1F ↑	0.76	0.27	0.21
1/32	AUC ↑	0.89	0.68	0.68
	T1F ↑	0.30	0.22	0.17
1/64	AUC ↑	0.74	0.52	0.55
	T1F ↑	0.11	0.03	0.06

the average detection scores of 100 runs with different radius scalings on the four datasets. We note that when the radii become smaller, the detection efficiency decreases, with the TPR approaching zero, and the AUC approaching 0.5, which makes the detection algorithm almost as unreliable as random guessing. A higher radius scaling allows better detection efficiency, but a radius scaling of 1/2 seems to decrease the TPR when FPR is capped at 1%, being lower than for a radius scaling of 1/4 for $\texttt{Ellipse}_{rand}$ and $\texttt{Ellipse}_{zeros}$.

6 Responsible Research

Reproducibility Our work has been made public and available on GitHub, being accessible through https://github.com/6toma/ellipse-watermark. The process of adapting Tree-Ring to Tabsyn is thoroughly explained in section 4. Tabsyn's fine-tuned weights are also available in our repository, along the real, synthesized and perturbed datasets, ensuring the results can be obtained by following the setup explained in section 5. Although generative models make use of randomization, we use manual seeding for our experiments, ensuring deterministic results. The package and library requirements, along their versions, for our implementation are all provided on GitHub. We have made considerable efforts to ensure that our research is fully transparent and reproducible in its entirety.

Ethical aspects Our implementation has been intensively tested against four real-world datasets. The datasets—Abalone [1], Adult [2], Default [3] and Diabetes [4]—are publicly available and contain no personally identifiable information. We also limit False Positive Rates for our watermark to 1%, ensuring that the chance of false alarms is minimized, while still maintaining significance of results. We acknowledge the limitations of our watermark and do not falsely claim performance that does not exist or cannot be proven.

Scientific integrity and use of LLM During our research, we have limited our use of Large Language Models (LLMs) to only rephrasing and enhancing our writing. We have not used LLMs to obtain from scratch any result, idea or paragraphs for the purpose of our research.

7 Limitations and Future Work

Our method has only been tested on Tabsyn, hence future evaluation on other tabular diffusion models, such as TabDDPM [20] could yield more results. `Ellipse` requires the use of DDIM during the sampling process, which leads to a drop in data quality compared to the original Tabsyn score-based sampling. Future work that uses exact diffusion inversion via bi-directional integration approximation [28] (BDIA) could improve the performance of `Ellipse`'s detection efficiency. We also acknowledge that the watermark is by design verifiable only by the model owner, as the "model parameters are needed to perform the inversion process" [26]. This can restrict third parties from detecting the watermark freely and without the use of an API, but it offers protection against adversaries checking whether their battery of attacks successfully removed the watermark. We also suggest the idea of creating a tabular-specific pattern for `Ellipse` that leverages the properties of tables—in the same way *ring* was created for images.

8 Conclusion

We identify an absence of research in the area of watermarking tabular diffusion models. Current generative model watermarking techniques are computationally expensive and prone to bias. Tree-Ring is a state-of-the-art, training-free watermarking algorithm for image diffusion models, that works for square shaped data. We pinpoint a major issue in adapting Tree-Ring to a tabular diffusion model, namely the shape of the data being incompatible, as tabular data has a rectangular shape. We propose `Ellipse`, a generalization of Tree-Ring that allows watermarking rectangular shaped data by changing the mask shape from circle to oval. We showcase the performance of `Ellipse` by making use of Tabsyn, a state-of-the-art score-based tabular diffusion model, using four real-world datasets. Our oval mask has a negligible drop of 3.5% in data quality—measured by resemblance, machine learning efficiency (MLE) and discriminability metrics—which is lower than the 12.46% drop in data quality offered by using a circular mask. We show strong detection possibility with a high average Area Under the Curve (AUC) of 0.9 out of 1, having a negligible average drop of 0.4% in AUC compared to Tree-Ring's circle mask. For future work, we suggest implementing exact diffusion inversion via bi-directional integration approximation (BDIA) to improve the detection efficiency.

References

[1] Abalone dataset. https://www.kaggle.com/datasets/rodolfomendes/abalone-dataset.

[2] Adult dataset. https://www.kaggle.com/datasets/wenruliu/adult-income-dataset.

[3] Default dataset. https://archive.ics.uci.edu/dataset/350/default+of+credit+card+clients.

[4] Diabetes dataset. https://www.kaggle.com/datasets/mathchi/diabetes-data-set.

[5] Midjourney. https://www.midjourney.com/home.

[6] Tabsyn implementation github repository. https://github.com/amazon-science/tabsyn.

[7] Tree-ring implementation github repository. https://github.com/YuxinWenRick/tree-ring-watermark.

[8] Xgboost classifier. https://xgboost.readthedocs.io/en/stable/parameter.html.

[9] Vadim Borisov, Kathrin Seßler, Tobias Leemann, Martin Pawelczyk, and Gjergji Kasneci. Language models are realistic tabular data generators. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

[10] Minshuo Chen, Song Mei, Jianqing Fan, and Mengdi Wang. An overview of diffusion models: Applications, guided generation, statistical rates and optimization. *CoRR*, abs/2404.07771, 2024.

[11] Prafulla Dhariwal and Alexander Quinn Nichol. Diffusion models beat gans on image synthesis. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 8780–8794, 2021.

[12] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial networks. *CoRR*, abs/1406.2661, 2014.

[13] Hengzhi He, Peiyu Yu, Junpeng Ren, Ying Nian Wu, and Guang Cheng. Watermarking generative tabular data. *CoRR*, abs/2405.14018, 2024.

[14] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[15] Jeffrey Bloom Jessica Fridrich Ingemar Cox, Matthew Miller and Ton Kalker. *Digital Watermarking and Steganography*. Morgan Kaufmann Publishers Inc., 2007.

[16] Haque Ishfaq, Assaf Hoogi, and Daniel Rubin. Tvae: Triplet-based variational autoencoder using metric learning. *arXiv preprint arXiv:1802.04403*, 2023.

[17] Jayoung Kim, Chaejeong Lee, and Noseong Park. Stasy: Score-based tabular data synthesis. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

[18] Diederik P. Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. *CoRR*, abs/2107.00630, 2021.

[19] John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. A watermark for large language models. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 17061–17084. PMLR, 2023.

[20] Akim Kotelnikov, Dmitry Baranchuk, Ivan Rubachev, and Artem Babenko. Tabddpm: Modelling tabular data with diffusion models. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 17564–17579. PMLR, 2023.

[21] Tennison Liu, Zhaozhi Qian, Jeroen Berrevoets, and Mihaela van der Schaar. GOGGLE: generative modelling for tabular data by learning relational structure. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

[22] Yugeng Liu, Zheng Li, Michael Backes, Yun Shen, and Yang Zhang. Watermarking diffusion model. *arXiv preprint arXiv:2305.12502*, 2023.

[23] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 8162–8171. PMLR, 2021.

[24] Joseph Ó Ruanaidh and Thierry Pun. Rotation, translation and scale invariant digital image watermarking. In *Proceedings 1997 International Conference on Image Processing, ICIP '97, Santa Barbara, California, USA, October 26-29, 1997*, pages 536–539. IEEE Computer Society, 1997.

[25] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[26] Yuxin Wen, John Kirchenbauer, Jonas Geiping, and Tom Goldstein. Tree-ring watermarks: Fingerprints for diffusion images that are invisible and robust. *CoRR*, abs/2305.20030, 2023.

[27] Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. Modeling tabular data using conditional GAN. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc,

Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 7333–7343, 2019.

[28] Guoqiang Zhang, John P. Lewis, and W. Bastiaan Kleijn. Exact diffusion inversion via bi-directional integration approximation. *CoRR*, abs/2307.10829, 2023.

[29] Hengrui Zhang, Jiani Zhang, Balasubramaniam Srinivasan, Zhengyuan Shen, Xiao Qin, Christos Faloutsos, Huzefa Rangwala, and George Karypis. Mixed-type tabular data synthesis with score-based diffusion in latent space. *CoRR*, abs/2310.09656, 2023.

[30] Yunqing Zhao, Tianyu Pang, Chao Du, Xiao Yang, Ngai-Man Cheung, and Min Lin. A recipe for watermarking diffusion models. *CoRR*, abs/2303.10137, 2023.

A Datasets

Abalone [1] contains information about the physical features of abalone with the purpose of predicting their age. It has 4,177 samples, with 8 numerical columns and 1 categorical column. The **Adult** [2] dataset contains the demographic and employment related features about individuals, with the purpose of predicting whether their income exceeds 50,000. The dataset is large with 48.842 samples and a mix of 6 numerical columns and 9 categorical. **Default** [3] has information about default payments, credit data, and bill statements of credit card clients in Taiwan from April 2005 to September 2005, with the purpose of predicting whether the client will default payment the following month. This dataset contains 30,000 samples, 14 numerical columns, and 11 categorical columns. The **Diabetes** [4] dataset contains information about the health 21 years old female patients of Pima Indian heritage, with the purpose of predicting whether the patient has diabetes. This relatively small dataset has 768 rows, and 9 columns—8 numerical and 1 categorical.

B Evaluation metrics

The data quality metrics are:

- **Resemblance (↑)** The resemblance metric measures the correlations between the real and synthesized data and the similarity between their distributions. Here, we use two metrics and average their scores. Specifically, we use
  - Theil's U between a real and synthetic column and average the score over all columns
  - Jensen-Shannon divergence between the distributions of the real and synthetic data
- **MLE** (↑) Using MLE, we evaluate the performance of synthetic data in downstream machine learning tasks. Training is done on synthetic data, then testing is performed on the real data. The score is represented by the AUC score for classification tasks and RMSE for regression tasks.
- **Discriminability (↑)** The discriminability score measures the difficulty of a trained binary classifier to distinguish between real and synthetic data. We first concatenate the two datasets, then split into training and testing sets with a test size of 20%. Then, we train an XGBClassifier [8] from XGBoost on the training set and evaluate it on the test set. The result contains the F1 score of the predicted labels, the probability mean square error of the predicted values, and the ROC AUC score.

C ROC curves with the distances between the latents and the key for the four datasets

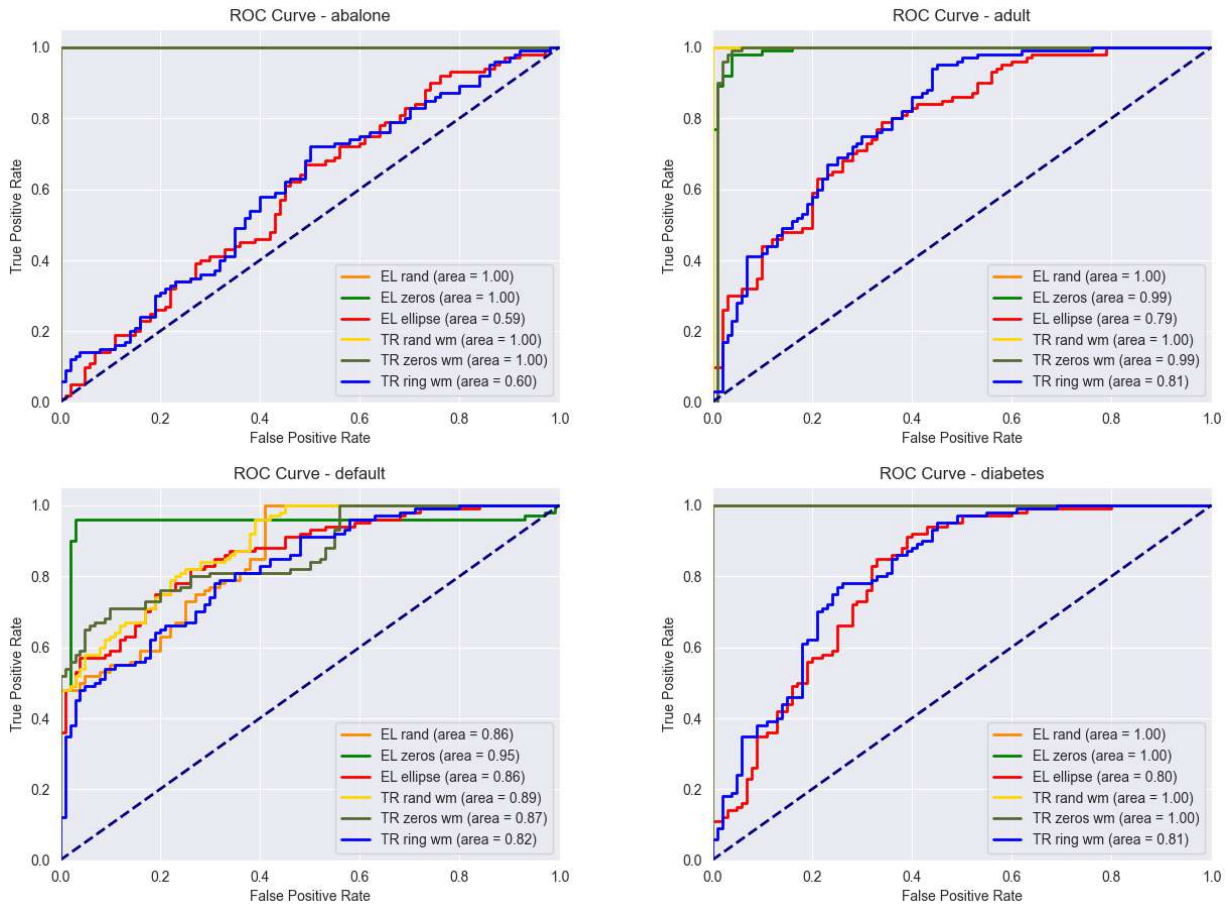

Figure 2: ROC Curves for the distances between the latents and the watermark key for the Abalone, Adult, Default and Diabetes datasets