Stanford University

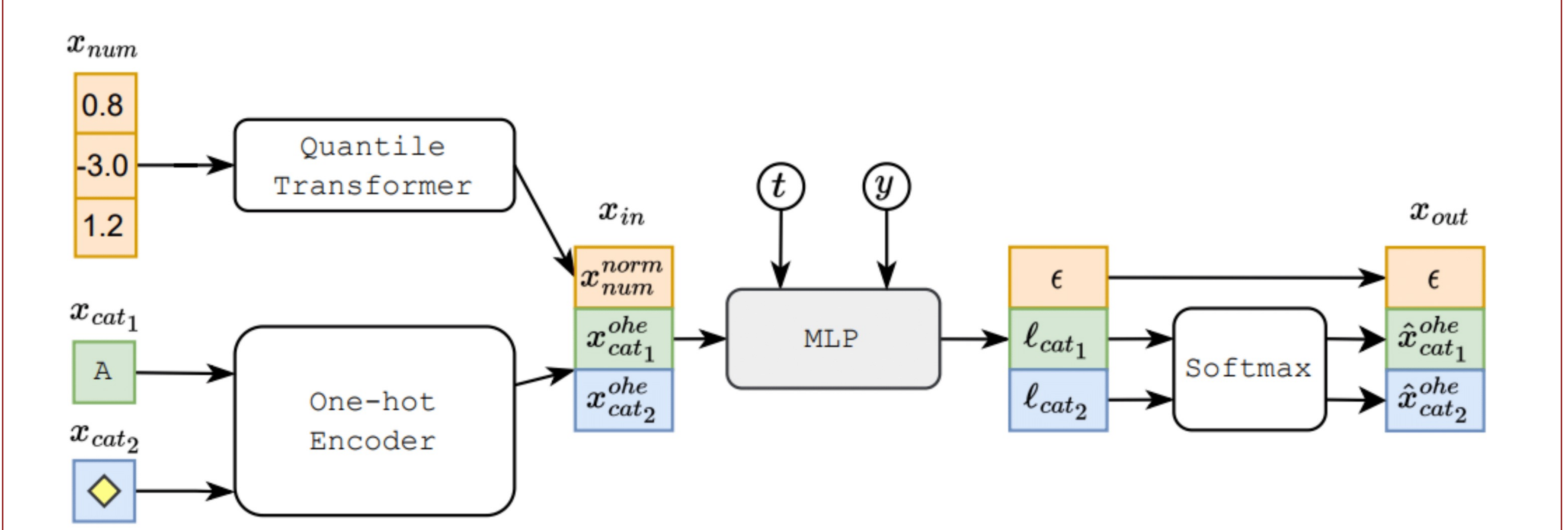
# Diffusion Models for Data Imputation

## Problem:

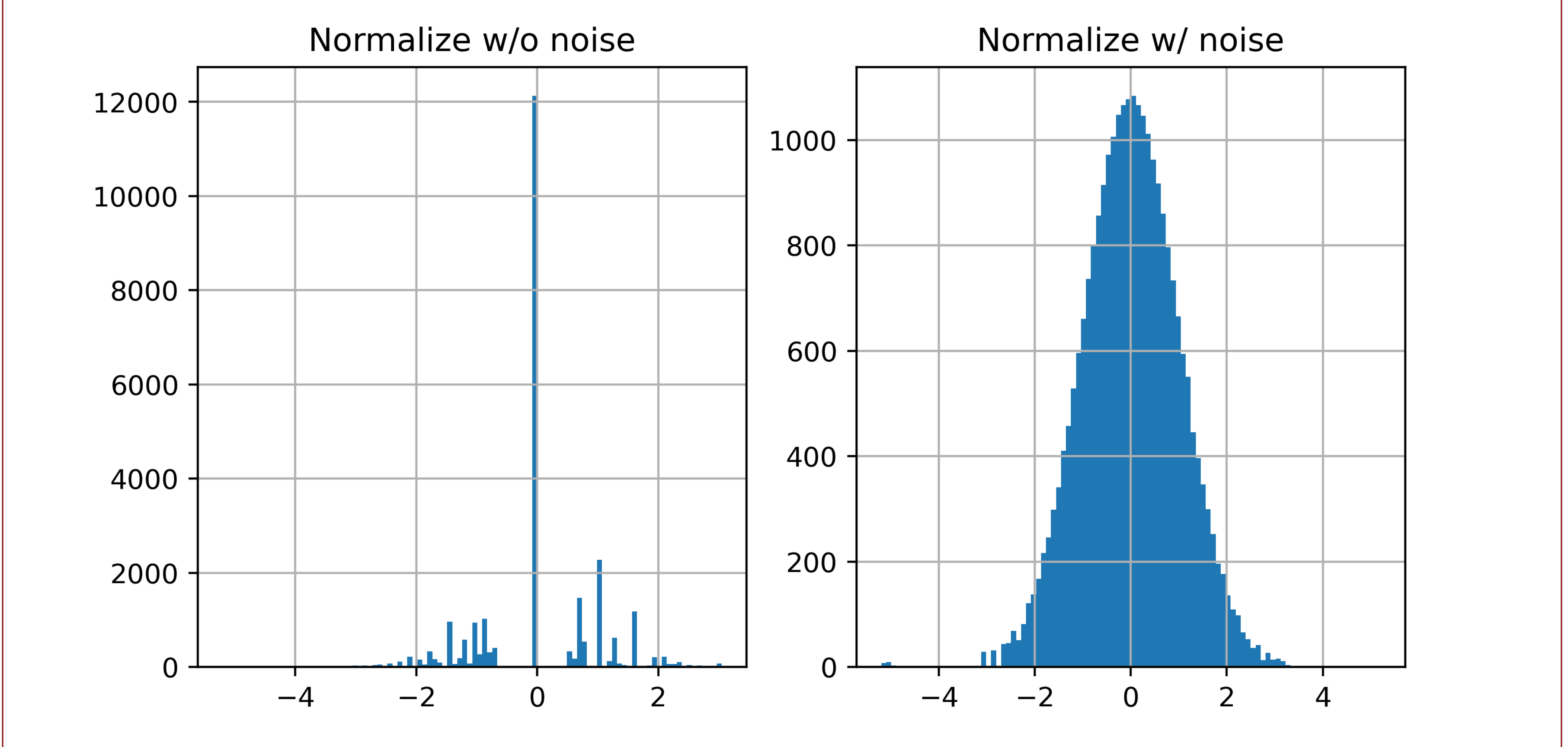
Missing Data imputation is commonly done using methods which can be implemented quickly and easily, like mean/mode imputation and random forest imputation. More complex methods, namely generative methods like variational autoencoders (VAEs) and generative adverserial models (GANs)[2] have also been shown to achieve success when applied to data imputation problems. Diffusion models are a trending generative technique that operate differently than VAEs and GANs [3]. Diffusion models essentially learn how to reconstruct a data point of a target distribution from noise. While this type of model has been most successful in image domains, Kotelnikov et al., 2022 introduced the TabDDPM model to apply the diffusion process to the generation of synthetic tabular data [1]. In this project, we adapted TabDDPM for data imputation.

## Data:

Our method was applied to a total of sixteen datasets. Fifteen of these were used by the authors to test the performance of synthetic data generation using TabDDPM. The hyperparameters of the TabDDPM model had already been tuned on these datasets, allowing for an expedited training process. We imported and tuned the hyperparameters of one additional dataset, JapaneseVowels, to ensure that we could apply our method to datasets as yet unseen by the TabDDPM model.



Taken from Kotelnikov et al., 2022 [1]



## References:

[1] Kotelnikov, Baranchuk, Rubachev, and Babenko. Tabddpm: Modelling tabular data with diffusion models. arXiv preprint arXiv:2209.15421, 2022
[2] Sebastian Jäger, Arndt Allhorn, and Felix Bießmann. A benchmark for data imputation methods. Frontiers in big Data, page 48, 2021.
[3] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. CoRR, abs/1503.03585, 2015.

## Experiments:

- ❑ One randomly chosen column per dataset chosen for imputation experiment
- ❑ Tested MCAR ("missing completely atrandom"), MNAR ("missing not at random"), and MAR ("missing at random") missingness patterns
- ❑ Tested 1%, 10%, 30%, and 50% missing data proportions.
- ❑ Used RMSE for numeric feature performance evaluation and F1 score for categorical feature performance evaluation
- ❑ Compared performance to mean/mode and random forest imputation baselines
- ❑ Resulted in 192 experiments overall, 72 on categorical data, 120 on numeric data

## Model:

The high level structure of the TabDDPM model is shown on the left. The model takes one-hot encoded categorical features and normalized numerical features as input [1]. The model is evaluated by an ML efficiency metric, which compares how well a learning algorithm performs after training on synthetic data generated by the model to how well it performs using true data. The hyperparameters of TabDDPM are tuned on the performance of a downstream task particular to each dataset [1]. We accepted the authors' argument that using the CatBoost learning algorithm for tuning produces the most accurate synthetic data and that doing so does not bias the model to producing data which favors the CatBoost model specifically [1], and therfore used CatBoost to tune our hyperparameters for each dataset. We then trained the model in the same manner as the authors did. The authors used quantile normalization to transform every numerical feature to be as close as possible to a normal distribution. This is important, as TabDDPM assumes all numerical features follow a gaussian distribution. However, the scikit-learn implementation used in the paper has trouble mapping numerical distributions with few observed unique values to gaussians. For this reason, we temporarily added a small amount of noise to the training data to spread out replicate values, allowing the quantile transformer to perform better, as seen on the left, leading to a small improvement in model performance. To simulate missing data, we applied a mask to a column of our testing set and filled masked values with NaN values, which we then aimed to impute accurately. To understand how we used the TabDDPM model to achieve this goal, it is instructional look at how the authors model their diffusion process. For numerical features $x^n$, they write [1]:

$$q(x_t^n|x_{t-1}^n) := \mathcal{N}\left(x_t^n; \sqrt{1-\beta_t}x_{t-1}^n, \beta_t I\right)$$
$$q(x_T^n) := \mathcal{N}(x_T^n; 0, I)$$
$$p_\theta(x_{t-1}^n|x_t^n) := \mathcal{N}(x_{t-1}^n; \mu_\theta(x_t^n, t), \Sigma_\theta(x_t^n, t))$$

For categorical features $x^c$ [1]:

$$q(x_t^c|x_{t-1}^c) := Cat\left(x_t^c; (1-\beta_t)x_{t-1}^c + \beta_t/K\right)$$
$$q(x_T^c) := Cat(x_T^c; 1/K)$$

For synthetic data generation, we begin by choosing some random vector $x_T$ whose numerical features are sampled from $\mathcal{N}(x_T^n; 0, I)$, and whose categorical features are sampled from $Cat(x_{T,cat}^c; 1/K)$. We would then feed $x_{t=T}$ and the timestep $t$ into the neural network, which produces the parameters of the probability distributions from which to sample $x_{t-1}$. In other words: $x_{t-1} \sim p_\theta(x_{t-1}|x_t, t)$. Repeating this process until $t = 0$ would then yield a generated sample.

For imputation, our input is a given row of data with some values known and some unknown, not a random vector. From thisinput, we want to generate a sample that matches the known values and produces reasonable estimations of missing values. We found the most effective approach for this to be the following: we use the known forward diffusion model $q(x_t|x_0, t)$ to replace our known values at time $t$, and the learned diffusion model $p_\theta(x_{t-1}|x_t, t)$ to fill in the unknown variables at time $T$ that we wish to impute. Programmatically, we can write:

$$x_{t,i} = \begin{cases} q(x_t|x_0, t)_i & \text{if } x_{0,i} \neq \texttt{NaN} \\ p_\theta(x_t|x_{t+1}, t+1)_i & \text{if } x_{0,i} = \texttt{NaN} \end{cases} \tag{1}$$

Since only $p_\theta(x_{t-1}|x_t, t)$ is given by our learned diffusion model, we still have to define $q(x_t|x_0, t)$. For numerical features $x_t^n$, we used a function `gaussian_q_sample` implemented by the TabDDPM authors which samples

$$x_t^n \sim \mathcal{N}\left(x_0^n\sqrt{\prod_{i=1}^{t}(1-\beta_i)}, 1-\prod_{i=1}^{t}(1-\beta_i)\right).$$

In our experiments we found no noticeable difference in imputation quality when changing variance to zero, i.e., when sampling

$$x_t^n \sim \mathcal{N}\left(x_0^n\sqrt{\prod_{i=1}^{t} 1-\beta_i x_0^n}, 0\right)$$

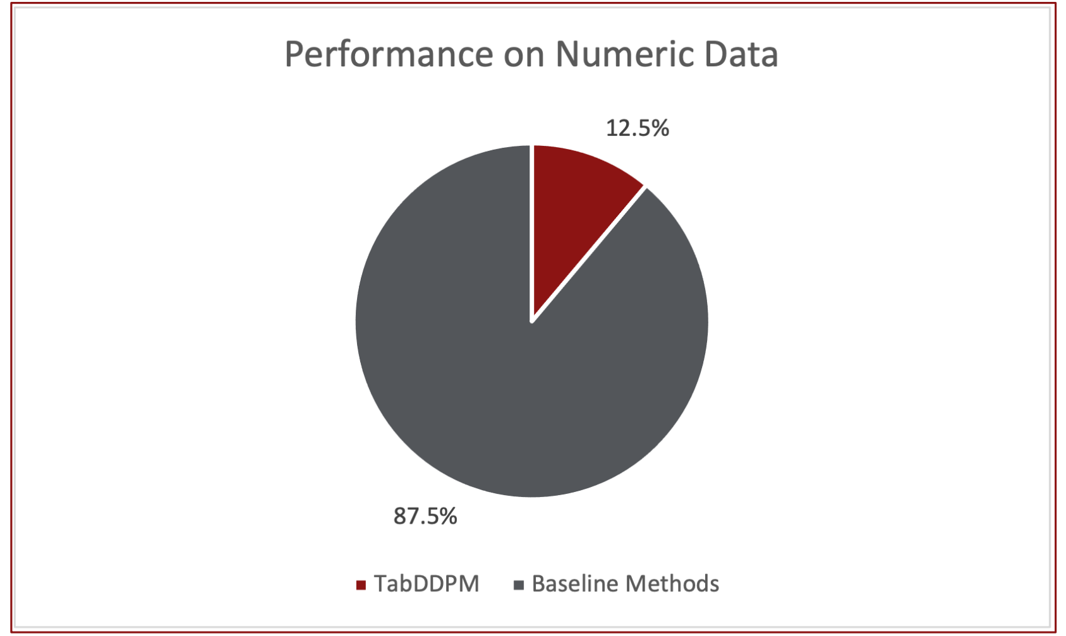
or in other words setting $x_t^n := x_0^n\sqrt{\prod_{i=1}^{t} 1-\beta_i}$ Given that both approaches yielded similar results, we used zero variance, since adding noise to the known variables takes away information from the diffusion model. For categorical features we tried using a pre-implemented function `q_sample`, which sets

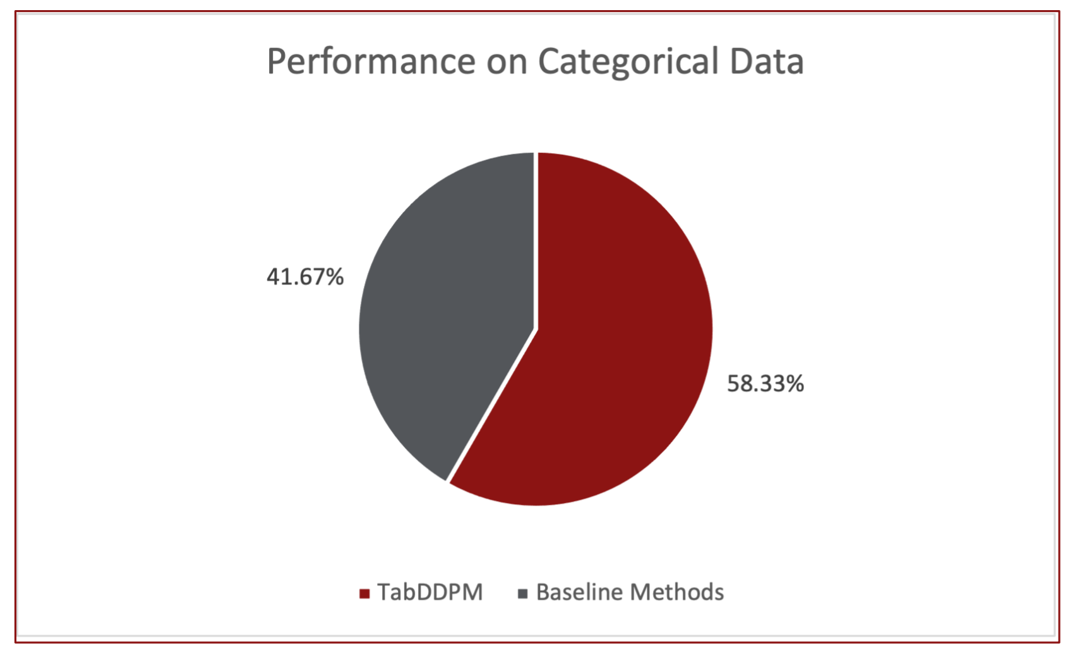
$$x_t^c \sim Cat\left(x_t^c; \left(\prod_{i=1}^{t}(1-\beta_i)\right)x_0^c + \left(1-\left(\prod_{i=1}^{t}(1-\beta_i)\right)\right)/K\right)$$

However, we saw similar performance by using a simpler method, setting $x_t^c := x_0^c$.

## Discussion:

Given that the TabDDPM model was able to outperform the mean/mode method of imputation in many experiments, we have demonstrated that our method is feasible for data imputation. However, one clear conclusion from our results is that our method should not be used on numeric features. Tuning the hyperparameters of the model on a new dataset took ten to twenty hours. The random forest baseline, on the other hand, could impute values on a newly provided dataset in less than a minute and with a higher degree of accuracy. The evidence is more promising for categorical data, where TabDDPM was able to outperform the simpler baseline methods more consistently.
Our results align with results of the TabDDPM model for synthetic data generation, which performed comparably to the simpler SMOTE technique. Unfortunately, the advantages for privacy using TabDDPM offers on synthetic data do not apply to imputation problems. The positive results seen on certain categorical features are encouraging but are not consistent enough to justify the length of time necessary to implement this method.





| Dataset | Abalone | | | Adult | | | Buddy | | |
|---|---|---|---|---|---|---|---|---|---|
| | MCAR | MAR | MNAR | MCAR | MAR | MNAR | MCAR | MAR | MNAR |
| Mean/Mode | 0.1818 | 0.2727 | **1.0000** | 12.0485 | 11.3982 | 14.3676 | 0.1923 | 0.2007 | 0.1837 |
| Random Forest | **0.7190** | 0.3429 | 0.3846 | **10.3463** | **10.5234** | **12.1608** | 0.5487 | 0.4719 | 0.5606 |
| TabDDPM | 0.1905 | **0.3636** | 0.1937 | 14.2834 | 14.9314 | 15.5247 | **0.7513** | **0.6898** | **0.7473** |

| Dataset | California | | | Cardio | | | Churn2 | | |
|---|---|---|---|---|---|---|---|---|---|
| | MCAR | MAR | MNAR | MCAR | MAR | MNAR | MCAR | MAR | MNAR |
| Mean/Mode | 0.4042 | 0.1129 | 0.0755 | 0.3890 | 0.3873 | 0.3880 | 0.3377 | 0.3443 | 1.0000 |
| Random Forest | **0.1108** | **0.0718** | **0.0458** | **0.7373** | **0.7257** | **0.7693** | 0.4479 | 0.5886 | 0.4048 |
| TabDDPM | 0.3790 | 0.1144 | 0.0652 | 0.6214 | 0.6040 | 0.6997 | **1.0000** | **1.0000** | **1.0000** |

| Dataset | Diabetes | | | FB Comments | | | Gesture | | |
|---|---|---|---|---|---|---|---|---|---|
| | MCAR | MAR | MNAR | MCAR | MAR | MNAR | MCAR | MAR | MNAR |
| Mean/Mode | 10.3491 | **6.8101** | **3.4343** | 117.8655 | 109.3840 | 30.3612 | 0.0008 | 0.0018 | 0.0013 |
| Random Forest | 9.2168 | 7.2049 | 4.9026 | **2.8292** | **3.0761** | **0.9898** | **0.0006** | **0.0010** | 0.0010 |
| TabDDPM | **8.8205** | 10.0074 | 11.2128 | 35.4298 | 32.3356 | 5.4699 | 0.0006 | 0.0014 | **0.0010** |

| Dataset | Higgs Small | | | House | | | Insurance | | |
|---|---|---|---|---|---|---|---|---|---|
| | MCAR | MAR | MNAR | MCAR | MAR | MNAR | MCAR | MAR | MNAR |
| Mean/Mode | 0.5225 | 0.4699 | 0.4480 | 0.3200 | 0.2890 | 0.3877 | 0.1061 | 0.0667 | 0.0000 |
| Random Forest | **0.3858** | **0.3431** | **0.3100** | **0.3014** | **0.2620** | **0.3265** | 0.4083 | 0.3330 | 0.2690 |
| TabDDPM | 0.5496 | 0.5039 | 0.4229 | 0.3767 | 0.3288 | 0.4431 | **0.5954** | **0.5868** | **0.3697** |

| Dataset | King | | | Miniboone | | | Wilt | | |
|---|---|---|---|---|---|---|---|---|---|
| | MCAR | MAR | MNAR | MCAR | MAR | MNAR | MCAR | MAR | MNAR |
| Mean/Mode | 0.4983 | 0.4983 | 1.0000 | 55.3293 | 3.4683 | 3.4492 | 33.5430 | 84.9879 | 23.3350 |
| Random Forest | **0.7488** | **0.6983** | **1.0000** | **0.0555** | **0.0538** | **0.0355** | **8.1695** | **15.4178** | **5.3480** |
| TabDDPM | 0.4977 | 0.6620 | 0.4988 | 36.7834 | 0.0866 | 0.0568 | 12.3297 | 48.2451 | 7.4212 |

| Dataset | Japanese Vowels | | |
|---|---|---|---|
| | MCAR | MAR | MNAR |
| Mean/Mode | 0.3172 | 0.2678 | 0.1300 |
| Random Forest | **0.0916** | **0.0797** | **0.0812** |
| TabDDPM | 0.1016 | 0.0959 | 0.0943 |

Table 1: Performance with 10% missing data. Abalone, Buddy, Cardio, Churn2, Insurance, and King show macro F1 scores. The rest show RMSE.

## Future Work :

If the tuning time of the TabDDPM model were to be shortened drastically, new avenues of investigation might be worthwhile. In particular, further testing on categorical features could be conducted to determine if the method actually does represent an improvement in that area. Results would need to be compared to a wider selection of generative and discriminative imputation techniques to draw such a conclusion. Also, it might be worth studying the performance of our method after the TabDDPM model is trained on incomplete training data. We originally wanted to test this capability, but had to prioritize other tasks due to the chokehold that tuning time placed on our work.