PERFGEN: A Synthesis and Evaluation Framework for Performance Data using Generative AI

Banooqa H. Banday
Texas State University
banooqa@txstate.edu

Tanzima Z. Islam
Texas State University
tanzima@txstate.edu

Aniruddha Marathe
Lawrence Livermore National Laboratory
marathe1@llnl.gov

***Abstract*—Collecting data in High-Performance Computing (HPC) is a laborious task, demanding that application scientists execute the application multiple times with different configurations. Due to the essential nature of performance modeling and root cause analysis as initial phases of performance enhancement, the data collection phase prolongs the optimization process. Motivated by this observation, we investigate the feasibility of leveraging the recent advancement in the field of generative Artificial Intelligence (AI) to synthesize performance samples. However, generating synthetic performance data introduces an additional hurdle: the absence of ground truths to assess the quality of the synthetic data. This work takes a step toward bridging this gap where we propose a framework—PERFGEN—for generating performance data and evaluating its quality using a novel metric called `Dissimilarity`. Our experiments with three performance and five machine learning datasets (including three classification and two regression datasets), confirm that our proposed `Dissimilarity` correlates with model accuracy better than three of the state-of-the-art metrics—SD quality, Kullback-Leibler Divergence (KL), and TabSyndex, demonstrating that the `Dissimilarity` metric strongly correlates with the quality of generated scientific data. We evaluate the quality by measuring how well the generated data enables a downstream Machine Learning (ML) task to generalize. Since performance data is a special case of scientific data—typically stored in tabular format and consisting of numerical, categorical, and ordinal features—our methodologies and metrics apply to scientific data from other domains as well.**

***Index Terms*—Large Language Model, Generative Modeling, Evaluation, Scientific Data**

I. INTRODUCTION

Data collection in HPC is time-consuming; an application scientist must run the application with various configurations many times. Not only is the process time-consuming, it could potentially lead to system failures or overloads on the shared cluster causing an inconvenience to all the users. The collected data is pivotal in root cause analysis, leading to better and optimal processes. As performance modeling and root cause analysis are necessary for performance optimization, the data collection process can delay it.

Motivated by these observations, this work aims to answer the following research question: *Can we leverage deep generative models to synthesize new performance samples to reduce the data collection overhead?* However, synthetic performance data generation has an additional challenge: there is no ground truth to compare and assess the quality of the synthetic data [1]. A characteristic of an appropriate evaluation metric is that it should corroborate with the quality of the downstream analysis tasks, e.g., a synthesized dataset with high score should achieve similar prediction accuracy as the real dataset for the same test samples. To compute the quality of synthesized data, most works on generative networks such as [2] visualize it against the actual data, which is only possible when the data is of low dimensionality. Furthermore, other works have used different metrics to assess the quality of synthesized data. For instance, Conditional Tabular Generative Adversarial Networks (CTGAN) [3] uses the likelihood fitness metric and machine learning efficacy; [4] and [5] use statistical similarity as a measure in addition to machine learning efficacy; [6] uses measures such as KL-divergence and log-cluster. The disparity between the metrics used for the same goal calls for a unified, effective metric that succinctly represents the similarity between any two datasets.

To address this gap, we propose a new evaluation metric that jointly considers the subspace and distribution similarity of the real and synthesized samples, and our extensive evaluation (Section V) demonstrates that our proposed metric correlates well with the accuracy of a downstream analysis task. The rationale is that synthesized samples can be used to augment real measurements when they make the downstream analysis task perform similar to how it would have if real measurements were done instead of synthesis. Since the main use of data augmentation is to improve machine learning models' performance, whether it's for prediction or generation, once we achieve high-quality synthetic samples, these samples can be reliably used to train new models for solving varied sets of problems. This is why we base the quality of synthetic samples as well as the efficiency of the metric on machine learning efficiency (MLE).

Another unique challenge of generating performance data is that it requires conditioning on a range of continuous values (e.g., execution time), different from images and text. This problem is also more challenging as there is no one label to match against. Additionally, the features of performance samples are often discrete and bounded. For instance, the number of threads on an HPC system can be 4, 8, 12, 16, 32, etc. It is improbable that the number of threads will be set to 27. Therefore, the generative models need to learn to represent features from user-defined acceptable subsets.

To address the above-mentioned gaps and motivated by the advancement in generative modeling, we propose a general-

izable framework—PERFGEN—of synthesis, evaluation, and explanation for performance data. In this work, we implement both a Large Language Model (LLM) and a Generative Adversarial Networks (GAN) to synthesize performance data. Further, we assess the quality of the generated data using our proposed `Dissimilarity` score and compare it with several evaluation metrics proposed in the literature. Our evaluations demonstrate that our proposed metric correlates with the efficacy of downstream analysis tasks (measured using MLE) better than the state-of-the-art metrics.

To summarize, our contributions are to:

1) Build a generalizable API for leveraging deep generative models including LLMs for synthesizing tabular data conditioned on continuous feature with a mixture of data types as described in III under use case II;
2) Propose a new evaluation metric for quantifying the quality of the unconditionally generated synthesized data for the purpose of augmentation as described in III under use case I;
3) Compare the proposed evaluation metric with other state-of-the-art ones by correlating their scores with the performance of downstream ML tasks;
4) Validate the synthesized data's quality using a visualization-based method.

We demonstrate the generalizability of our work by synthesizing data using eight HPC applications' performance datasets and five ML benchmark datasets. Extensive evaluations in Section V demonstrate that the PERFGEN framework is capable of synthesizing data with ranges as conditions, numerical and categorical features, and equipped with a superior evaluation metric—`Dissimilarity` score, that can indicate how effective the synthesized data is in a downstream analytics task.

II. Background and Related Work

A. Generative Models

Generative modeling refers to deep learning models that generate new, unseen samples with the same visual appearance and statistical properties as the already existing data. These models exist in various different architectures such as Variational Auto-encoders (VAEs) [7] or the Generative Adversarial Networks (GAN) [8]. Many frameworks have been developed based on these architectures for addressing various data types, including images, audio, and text. StyleGAN [9] is an example of a GAN based model used for image synthesis, whereas [10] is an example of a model used for speech synthesis. MedGAN [11] is another framework that leverages GANs for medical image-to-image translation. There are frameworks for generating tabular scientific data with mixed data types that consist of categorical and numerical features. Generative modeling has also been used to generate I/O traces [12]; however, this work is only specific to I/O logs generated by a library called Darshan [12]. Using GAN architecture, the CTGAN framework Conditional Tabular Generative Adversarial Networks (CTGAN) conditionally synthesizes tabular data with mixed data types. More recently, a lot of work has been leveraging the knowledge of Large Language Models (LLMs) to reason with and generate tabular data. LLMs have been used for translation [13], for virtual assistance [14] among others. Generate Realistic Tabular Data (GReaT) [15] is another such framework which in-turn leverages the HuggingFace [16] framework for fine-tuning pre-trained LLMs.

B. Evaluation metrics:

With data generation comes the need for evaluating the quality of the data. In the case of images, one of the simple ways to assess the quality of generated data is by visual inspection. However, the same method does not apply to assessing scientific data. Another way is to compare the real and fake data distributions; however, comparing just the distributions is not enough since it compares two specific instances of the population. Here, we present several metrics to assess the quality of the generated data:

Data Quality by SDMetrics (SDQuality) [17] is a percentage measure calculated on the basis of comparing marginal distribution and correlation distributions in two datasets. Both of these are measures of statistical similarities, however, the former compares individual columns shapes or frequencies whereas the latter compares the relationship between column pairs. Furthermore, the marginal distribution or the column shapes are computed as $1-$ Kolmogorov-Smirnov statistic [18] (or KS Complement) for numerical columns and as $1-$ Total Variation Distance[19] (or TV Complement) for categorical or boolean columns. For calculating the correlation between the column pairs, the metric uses Pearson [20] and Spearman [21] rank coefficients for pair trends between two numerical columns, otherwise computes a normalized contingency table [22].

TabSynDex [23] is a measure with value between 0 and 1 and is calculated by taking a mean of the basic statistical measures (mean, standard deviation, and median), log-transformed correlation coefficients [24], propensity Mean Squared Error (pMSE) [25], a support coverage score based histogram comparison, and an average of ML efficacy measures. Equation 1 defined TabSynDex as:

$$TabSynDex = \frac{BS + CORR + pMSE + SC + MLE}{5} \tag{1}$$

where the five measures in the numerator are the five different metrics mentioned above.

Mean KL [26] refers to the mean of KL diverges calculated between the real and synthesized datasets. Since KL is not a symmetric measure, i.e., the value calculated from dataset A to dataset B is not equal to the value calculated from dataset B to dataset A, we take the mean KL, which is the mean of KL from real dataset to synthesized dataset and KL from synthesized dataset to real one. If $KL_{real \to syn}$ represents the kl-divergence calculated from the real dataset to the synthesized dataset and $KL_{syn \to real}$ represents the kl-divergence calculated from the synthetic dataset to the real dataset, the mean KL can be given by:

$$meanKL = \frac{KL_{real \to syn} + KL_{syn \to real}}{2} \tag{2}$$

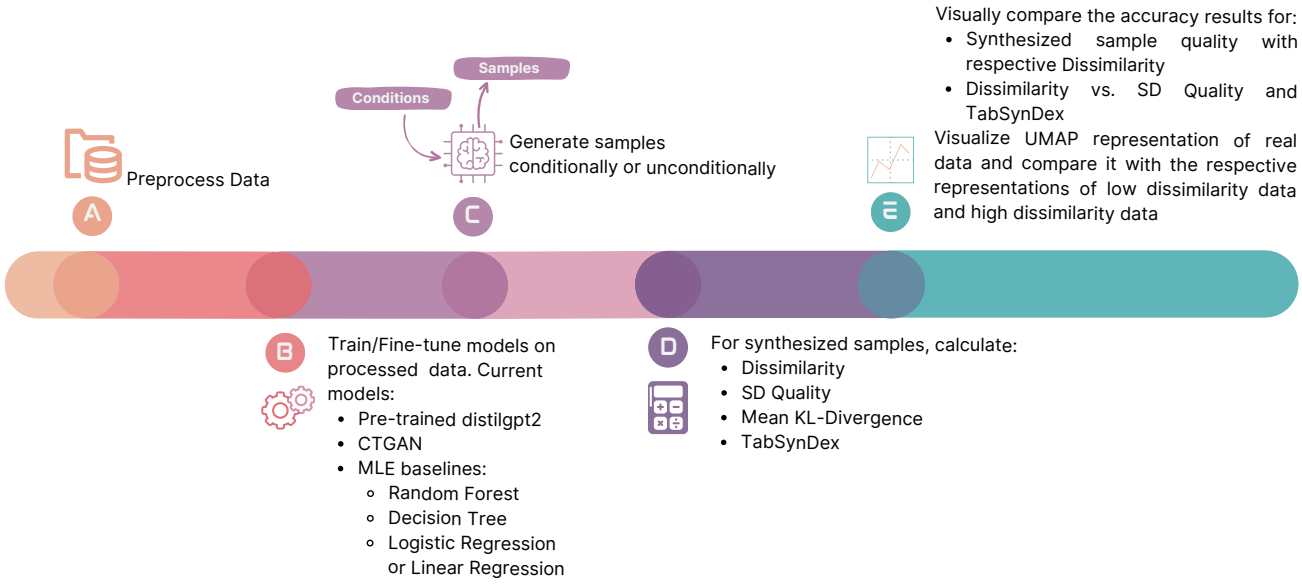

Fig. 1: Overview of the PERFGEN framework.

In contrast, our work differs from the previous work as we measure similarity between both the high-dimensional subspaces in addition to distributions that help compare the salient structures present within the data to approximate the dissimilarity between the datasets better.

III. Design and Implementation

Figure 1 shows the overall workflow of the PERFGEN framework. We train a GAN based architecture using CTGAN and fine-tune an LLM which is a transformer based architecture using GReaT.

A. Pre-processing

We list the steps for pre-processing the data below, and describe how they are implemented in PERFGEN.

Prompt Generation: LLMs are pre-trained on massive amounts of data, usually in text. To leverage these LLMs for tabular data, we must transform the tabular data in a way that can be used to fine-tune the LLM. For this purpose, PERFGEN pre-processes each sample into a sentence wherein feature values are represented in the form `feature_name` "is" `feature_value`, each feature separated by a comma.

Imputation: For most machine learning models, a standard step before feeding the model with data is to ensure no missing values exist. If the number of samples with missing values is insignificant, we remove the samples. Otherwise, we use the `Simpler Imputer` function provided by Sklearn [27]. The `Simpler Imputer` function finds the missing values and replaces all missing values in the column based on a selected strategy. We used the feature column's mean to replace any missing values.

B. Modeling

Generative models are Neural networks capable of synthesizing data samples akin to the existing ones. There are various types of generative models in the literature such as CTGAN, and Variational Autoencoders (VAE) [28]. In this work, we chose to work with a GAN based architecture and a transformer-based LLM for both the unconditional generation of synthetic data and generating synthetic samples conditionally. Figure 2 shows how we can generate data conditionally or unconditionally using PERFGEN.

C. Implementation

Generating using GAN: To generate data using GAN, PERFGEN uses the CTGAN which consists of two fully connected layers for both the generator and the discriminator where both minimize two different loss functions given by:

$$\mathcal{L}_{\mathcal{D}} = \frac{1}{m}\sum_{i=1}^{m}[D(x'^{(i)}) - D(x^{(i)})] \tag{3}$$

$$\mathcal{L}_{\mathcal{G}} = \frac{1}{m}\sum_{i=1}^{m}[D(x'^{(i)})] + H \tag{4}$$

Here, $\mathcal{L}_{\mathcal{D}}$ represents the loss function for the discriminator, and $\mathcal{L}_{\mathcal{G}}$ represents the loss function for the generator. $x^{(i)}$ represents the i^{th} sample of the real data and $x'^{(i)}$ represents the i^{th} sample of the synthesized data. $D(x)$ is the discriminator's output on the real data, $D(x)$ is the discriminator's output on the synthesized data and H is the cross entropy loss. When the discriminator's output on the synthesized data $D(x)$ matches that of the real data $D(x)$, the discriminator loss $\mathcal{L}_{\mathcal{D}}$ becomes zero. H is the cross entropy loss that penalizes the generator when a large error occurs so that the next time synthesized data can be closer to the real data.

Generating using LLMs: The GReaT framework uses pre-trained LLMs, which are based on transformers. The framework consists of three modules: (a) textual encoder that

encodes the tabular data into text or sentences for the LLM to consume and also additionally permutes or shuffles the feature sequence, (b) the fine-tuning module wherein the selected pre-trained LLM is fine-tuned using the encoded data, and (c) sampling module wherein we sample synthetic data from the fine-tuned model.

D. Conditioning:

CTGAN doesn't support conditioning on numerical features. However, our framework provides an API for specifying the conditions on numerical features for CTGAN. These conditions include a range for the target variable, e.g., "generate configurations to finish running an application in **less than 10 seconds**", or "generate configurations with the number of threads in the list of [4, 8, 16, 32, 64] to finish running an application between 5 and 10 seconds". The GReaT framework, on the other hand, supports arbitrary conditioning as a result of introducing permutations while encoding the tabular data.

E. Evaluation:

One of the core contributions of this work is a new metric for comparing the quality of the synthesized data quantitatively, and it is defined as:

$$\texttt{Dissimilarity} = \exp \sum_i \theta_i \lambda_i \tag{5}$$

Here, $d*$ is the optimal dimension of the Principal Component Analysis (PCA) [29] of the datasets. θ_i calculates the angle between the two low-dimensional subspaces of the original and synthesized data, representing the geodesic distance of the Grassmanian manifold, denoting how aligned the two subspaces are. The higher the value, the less aligned they are. To strengthen the quality of this metric, we also compare the distributions of the original and synthesized data samples by computing the KL divergence between the two datasets. The rationale for considering the subspaces in addition to distributions is that a low-dimensional projection often emphasizes the salient structure in data while rejecting noise and outlier data. In contrast, a distribution comparison only focuses on the sample at hand.

F. Explanation:

We leverage the Uniform Manifold Approximation and Projection (UMAP) [30] method for projecting the high-dimensional features into low dimensions and visualize them to identify information clusters or patterns.

G. Use Cases

In this work, we demonstrate the applicability of the PERFGEN framework in generating tabular data using six datasets for two use cases: (1) unconditional generation of features and targets for data augmentation and (2) conditional generation of features given targets for prediction and decision-making. Although our goal is to generate performance data, by using ML datasets, we can demonstrate that the data generation and evaluation capability of the PERFGEN framework applies to any scientific data.

Use Case I: Unconditional generation of features and targets for data augmentation

We unconditionally generate all the features of a dataset from a generative model trained on the whole of the real dataset available to augment the dataset. Such a use case is important for HPC as a high-fidelity generation process can reduce the need for extensive performance measurement experiments, reducing the overhead from days and weeks to seconds. Although we use four ML datasets, namely "HELOC", "Magic Telescope", "Parkinsons", and "Handwritten Digits", from the datasets described in the section IV for this use case, the same methodology can be applied to HPC datasets as well.

We train/fine-tune the models on all the samples and use the generated samples to (1) assess the quality of the trained model by comparing the Machine Learning Efficiency (MLE) achieved by the synthesized samples to the MLE achieved by the real samples on the same test dataset, (2) validate that `Dissimilarity` does corroborate with the MLE efficiencies achieved by the synthesized samples better than the SOTA metrics described in section II. The MLE evaluation metric is described in Section IV-D.

Use Case II: Conditional generation of features given targets for prediction and decision-making

We apply PERFGEN to synthesize the features given a specific value or range for the target column. This the problem can be mapped to that of decision-making, where a user, for instance, would like to know which configurations to use (e.g., number of threads, power cap, thread binding algorithm) that will enable the application to finish under a certain time limit. This use case is a practical one for HPC as it guides scientists and facilities in achieving good performance without evaluating hundreds or thousands of configuration options, thus saving developer time.

To implement decision-making, we train/fine-tune the models on 90% of the "California Housing", and "Perfvar" datasets. We then use the rest of the samples to conditionally generate different columns based on the kind of decision to be made. For instance, we use the features "longitude" and "latitude" along with "house value" to predict the average number of rooms, answering the question: **given a certain area and the budget, what is the number of rooms or square footage of the house one can expect to purchase?**

For prediction use-case, we train/fine-tune the models with 90% of the "California Housing", "Kripke", "Hypre", and "Perfvar" data samples. We then give the models all the features from the test set and calculate the Mean Square Error (MSE) for the target column based on the prediction made by the model and the ground truth. For this use case, a user needs to provide PERFGEN with a data file containing the desired values for the conditional columns and "nan" for all the other feature values.

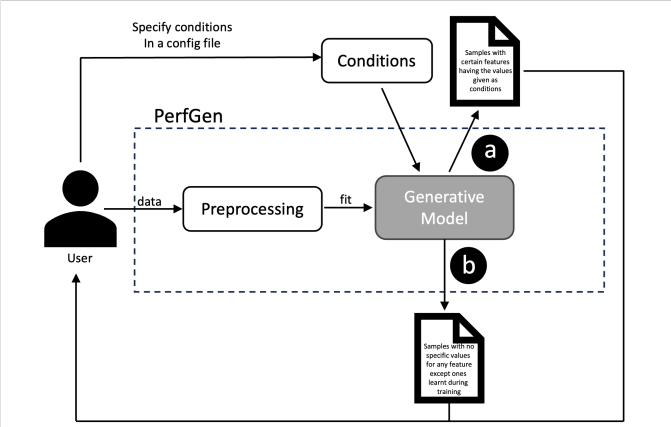

Fig. 2: Ways of generating data from a model - (a) Conditional Generation: where the user gives certain values for selected features to a trained model, and the model imputes the rest of the column values. (b) Unconditional Generation: where the user doesn't put any condition on any of the features so the trained model generates values based on the values it has seen during training

IV. Experimental Setup

A. Baselines

In this section, we describe the baselines and evaluation metrics for judging the quality of the synthetic samples.

ML Baselines:

1) ***Random Forest***

[31] is a popular machine learning algorithm known for its robustness and effectiveness in various tasks. It operates by constructing multiple decision trees during training and outputs the mode or mean prediction of the individual trees for classification or regression, respectively.

2) ***Decision Tree***

[32] is a versatile and interpretable machine learning model used for both classification and regression tasks. It works by recursively partitioning the feature space into regions based on feature values, aiming to minimize impurity or maximize information gain at each split. Each leaf node represents a decision or prediction, making it easy to interpret and explain.

3) ***Linear Regression***

[33] is another fundamental statistical method used for modeling the relationship between a dependent variable and one or more independent variables. The goal of linear regression is to find the best-fitting line (or hyperplane in higher dimensions) that minimizes the sum of squared residuals between the observed and predicted values. Linear regression is widely used for predictive modeling for regression tasks.

4) ***Logistic Regression***

[34] models the probability of the occurrence of the event of interest by fitting a logistic curve to the data. Despite its name, logistic regression is a classification algorithm. It estimates the probability that a given input belongs to a particular class.

We implemented the above mentioned baselines using Sklearn library in python. Also, Section II describes the state-of-the-art evaluation metrics we compare with our proposed `Dissimilarity` score.

B. Datasets

Table I provides a summary of the datasets used. Additional information about all the datasets is given below.

1) ***Perfvar***

[35] is an HPC dataset collected from six different applications by running them on a supercomputer at a national laboratory. Since all the applications had the same performance counters, it seemed ideal to combine them all in one dataset such that a single model could be trained and used for synthesis of samples for all the six applications. The features in this dataset include application and algorithm, which constitute the names of the algorithm and application in use for a particular sample. Bandwidth level, task count, power cap, and thread count are the other features that represent the system configuration. The application, algorithm, and bandwidth level features are categorical, whereas the task count, threads, power cap, and runtime features are numerical. The ML task in this dataset is to predict the runtime of an application, which is a continuous value. It is a regression dataset with $3,990$ samples.

2) ***Kripke***

[36] is another HPC datasets that was collected in Argonne National Laboratory. It consists of performance counters for the application named kripke such as the number of nodes used, the nesting method, number of threads used, package power and core power. The target for this dataset is the execution time, which is continuous, making this a regression dataset. The total number of samples is 6930 and the number of features is nine.

3) ***Hypre***

[37] was collected in Argonne National Laboratory. This dataset is similar to the ones mentioned above in the way that this also refers to performance counters collected by running an application on supercomputers. However, the main difference between Kripke and Hypre is that the performance counters apart from the runtime collected for the two are different. As Table I shows, this is a biggest dataset we have, with close to $25k$ samples. The total number of features are ten.

4) ***California Housing***

[38] is an ML benchmark dataset based on census data from 1990 pertaining to California. The features in this dataset include the longitude and latitude that represent a specific area, house age, number of rooms in the house, total number of bedrooms, median income of the household, and number of people in the household. The median house value serves as the target for this dataset which is a continuous value. Table I shows that the total number of samples in this dataset is $20,640$.

5) ***Parkinsons***

[39] is an ML benchmark dataset consisting of voice measurements of people in their early stages of Parkinson's

TABLE I
SUMMARY OF THE DATASETS

Dataset	# samples	# features	Target Type
PerfVar	3,991	7	Continuous
Hypre	24,965	10	Continuous
Kripke	6,930	9	Continuous
California Housing	20,640	8	Continuous
Parkinsons	5,875	19	Continuous
Magic Telescope	19,020	10	Categorical
Handwritten Digits	3,832	64	Categorical
HELOC	10,459	23	Categorical

disease recorded in their respective homes. There are nineteen features in this dataset wherein the five features with the prefix "Jitter" refer to measurements related to frequency, the six with the prefix "Shimmer" refer to measurements related to amplitude, and two other features correspond to the ratio of noise to tonal components in the voice recorded in the sample. The rest of the features pertain to the patient's age, gender, time in number of days since the signing up. The target for this dataset is the clinician's score based on the rest of the features. This is a regression dataset. Table I shows that the dataset has a total of 5875 samples in this dataset.

*6) **Magic Telescope***

[40] is an ML benchmark dataset consisting of ten features, and one class label. The purpose of this dataset is to help simulate the registration of gamma particles in the Cherenkov gamma telescope through imaging, thereby statistically differentiating between the showers caused by gamma rays (class g) from the ones caused by cosmic rays in the upper atmosphere (class h). The class column referring to whether a sample belongs to class 'h' or class 'g' serves as the target. Table I shows that the total number of samples in this dataset is $19,020$.

*7) **Handwritten Digits***

[41] dataset is a the tabular representation of 8x8 images of handwritten digits. As a result, there are 64 features, all corresponding to the different pixel values of an image. All features have values between zero and sixteen, whereas the class column (target) has values between zero and nine. This dataset is divided into train and test sets and the train set consists of 3832 samples, the number of samples we have trained our models on.

*8) **HELOC***

[42] or Home Equity Line Of Credit is an ML benchmark dataset consisting of a total of $10,459$ samples and 23 features. The features in this dataset consist of details about applicants, and the class tells whether those applicants will be able to repay their loan within two years.

C. Hyperparameter Tuning

We leverage wandb [43] for hyperparameter tuning and experiment tracking for CTGAN and we tune the following hyperparameters to find models with varying performances:

`batch size`: the number of samples fed into the model simultaneously. The batch sizes used are between $100 - 500$.

`epochs`: the number of times the model loops through the dataset. The number of epochs was varied between $600-1500$.

`discriminator steps`: the number of times the discriminator should be trained before returning to generator training. This number is varied between 1 and 3.

For the LLMs, we varied the number of epochs while batch size, and learning rate were kept the same.

D. Compare Quality of the Evaluation Metrics

We use the same samples generated for the previous use case and the results achieved in the ML baselines. Then, we compare these with the evaluation metric baselines and `Dissimilarity`. We want to see which of these measures consistently corroborate with the ML results and so can then be used as a standard for gauging the quality of synthetic data.

The results achieved from the machine learning models are referred to as MLE. It is a measure of how accurately a machine learning model can predict or classify compared to the ground truth or human-labeled data. The metrics used by the models depend upon the task at hand. For a classification problem, accuracy is used whereas for regression problems, MSE is calculated.

V. RESULTS

This section presents the detailed results of generating data for the eight datasets from two use cases discussed in Section III. Specifically, we address the following research questions:

- **RQ1:** Which generative AI architecture works well for synthesizing performance datasets?
- **RQ2:** Does our proposed `Dissimilarity` score correlate with the outcome of the downstream tasks?
- **RQ3:** How does our proposed `Dissimilarity` metric compare against the state-of-the-art (SOTA) evaluation metrics?
- **RQ4:** Provide insight into why our proposed `Dissimilarity` score correlates well with the data quality.

A. *RQ1: Comparison between GAN and LLMs for synthesizing performance datasets*

The object of this experiment is to evaluate the capability of the two different architectures in generating data conditionally in two ways: (1) When all the features are given and the model needs to predict the target, and (2) When only a few features are given and the model imputes all the other values.

Prediction: To test the prediction capability of a generative model as mentioned in use case II, we reserve 10% of the real samples while use the rest of the 90% samples for training. During testing, we provide the model with all features but no value for the target column. The model predicts the target values and we calculate the mean MSE from the predicted values and the ground truth. While LLMs are able to predict the target columns conditioned upon any feature, the GAN architecture is unable to handle conditions on numeric columns. Table III gives the details of the LLM training for each dataset along with the best MSE achieved from the baseline models.

TABLE II

THE SCORE FOR DIFFERENT METRICS ALONG WITH THE MACHINE LEARNING EFFICIENCY () ACHIEVED FROM ONE OF MACHINE LEARNING BASELINES - LOGISTIC REGRESSION(LOR)/LINEAR REGRESSION(LIR) OR DECISION TREE(DT) OR RANDOM FOREST(RF), WHICHEVER PERFORMED BEST. THE DATASET WITH AN ASTERISK(*) IS REGRESSION DATASET SO THE MLE FOR THAT REPRESENTS MEAN SQUARED ERROR (MSE) WHILE FOR THE OTHER DATASETS IT REPRESENTS ACCURACY. THE ARROWS NEXT TO THE METRIC NAMES INDICATE WHETHER THE METRIC IS SUPPOSED TO INCREASE OR DECREASE WITH THE INCREASE IN MLE.

Dataset	Machine Learning Efficiency			Dissimilarity (↓)	SDQuality (↑)	Tabsyndex (↑)	Mean KL Divergence (↓)
	Original	Synthesized	Model				
Handwritten Digits	97	21	LoR	85.63	99.99	-	0.03
HELOC	69.69	69.6	LoR	2.3	100	0.46	0.02
Magic Telescope	79.36	77.21	LoR	1.3	100	0.41	0.01
Parkinson's*	0.09	10.2	RF	464.05	99.96	-	1.19

TABLE III

MSE FOR LLM PREDICTION: THIS TABLE DETAILS THE NUMBER OF SAMPLES AND EPOCHS USED FOR FINE-TUNING THE LLM; THE MSE USING LLM, AND THE BEST BASELINE MODEL FOR EACH OF THE FOUR DATASETS.

Dataset	**# train**	**# epochs**	**MSE**	**BL-MSE**
PerfVar	3,592	600	22.02	4.76
Hypre	22,469	300	9.454	1.23
Kripke	6,237	300	1298.55	110
California Housing	18,577	300	0.51	0.25

Decision-making: For decision-making in use case II, we use the "California" dataset to enquire about what the house age and average bedrooms would be for a given longitude, latitude, and income. On average, the LLM recommends house age that is off by 9 years and average number of bedrooms off by 0.078. For the "Perfvar" dataset, we ask the generative models to recommend the number of threads to use given the application name, thread binding algorithm, and a runtime constraint. For this dataset, the LLM recommends the correct thread count 80% of the time. On the other hand, the GAN based generative model recommends a thread count off by 4 during all trials.

B. RQ2: Correlation between the `Dissimilarity` score and the outcome of the downstream tasks:

This experiment aims to study the correlation between our proposed `Dissimilarity` score with the outcome of the downstream tasks. We use the Parkinsons, Magic Telescope, Handwritten Digits, and HELOC datasets (Table I) for this experiment. We train/fine-tune models on each dataset separately and then sample data unconditionally, in context of the use case I, from these respective models. The quality of the synthesized data is then compared by training several ML baselines presented in Section III on the real and synthesized datasets separately and testing both the models on the same test set taken from the real dataset. Since the test samples from the actual dataset contain ground truth labels, we can quantify how well a model using synthesized data can approximate the samples from the real distribution. The model producing data that scores accuracy or mean squared error (MSE) closest to the results achieved by the real data would be considered the best model.

In Figure 3, we see the metrics with the MLE achieved on both the real and synthesized samples. We see two datasets, namely HELOC and Magic Telescope, wherein synthesized samples achieve MLE very close to the MLE achieved by the respective real samples. We see for both of these datasets, `Dissimilarity` score is very close to zero. However, for Handwritten Digits, we see the accuracy drops from 97% on the real samples to barely 21% on synthesized samples. Here, we see a considerable rise in the `Dissimilarity` score, reaching 85.63. Similarly, for another dataset for which the synthesized samples perform poorly with respect to MLE is the Parkinsons dataset (See Table II). We see the rise in Mean Squared Error (MSE) from 0.09 to 10.2, which is almost $100x$. `Dissimilarity` score reflects the same with a very high number of 464.

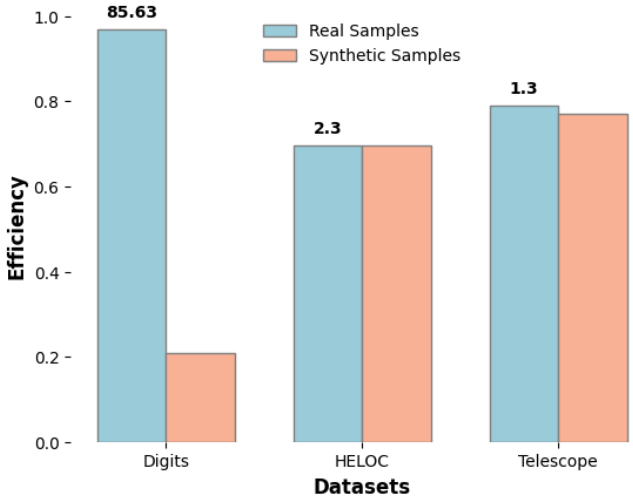

Fig. 3: The figure shows the MLE achieved on the classification datasets for both the real and synthesized samples. The number above each bar represents the `Dissimilarity` score for that dataset. We see a clear correlation between accuracy of the downstream machine learning tasks and `Dissimilarity`.

Therefore, we conclude the positive correlation holds between the `Dissimilarity` score and accuracy of the downstream machine learning tasks across all datasets, and all downstream machine learning tasks prove the viability of `Dissimilarity` as an effective evaluation metric for scientific data generation.

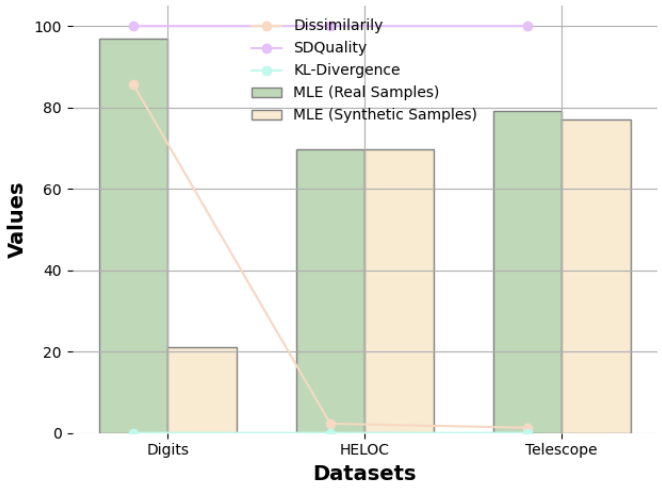

Fig. 4: The figure shows the MLE achieved on the classification datasets for both the real and synthesized samples along with the three metric scores (`Dissimilarity`, SDQuality, and KL-Divergence). We see no visible difference in SDQuality score and the KL-Divergence for the datasets even though the MLE difference for Digits dataset is evident. `Dissimilarity` score, however, increases with decrease in accuracy, which is the anticipated behavior.

C. *RQ3: Compare `Dissimilarity` against the SOTA evaluation metrics:*

This experiment aims to compare the sensitivity of `Dissimilarity` score to the quality of the unconditionally synthesized samples (use case I). Specifically, as the baseline, we select three commonly used evaluation metrics for scientific data—KL-Divergence, SDQuality, and TabSynDex (described in Section II). Note that Table II shows accuracy for the three datasets, namely Handwritten Digits, Magic Telescope, and HELOC, with classification labels and MSE for the Parkinson's dataset with continuous target. As explained in the previous section, `Dissimilarity` corroborates well with the results achieved from the MLE tasks.

Table II shows that the mean kl-divergence between the real and synthesized samples is only 0.03 for the Handwritten Digits dataset when the downstream accuracy achieved on the synthesized samples is way lower than the accuracy achieved on the real samples. On the other hand, SDQuality gives a score of 99% or above even though the datasets perform poorly based on the MLE.

Similarly, Table II shows that the Tabsyndex score is low at almost 0.4 although both the datasets achieve a close MLE to the the real dataset. Further investigation shows that the MLE score within the TabSyndex metric is low (0.11), which do not match with our calculation of MLE. Due to this discrepancy, we do not use the TabSyndex metric for the rest of the datasets.

In Figure 4, we observe that the SD quality and Kl-Divergence metrics do not correlate with accuracy; that is, the value of SDQuality is high even for Handwritten Digits, which shows poor performance for the downstream models, whereas KL-Divergence shows small divergence when it should be considerably large. In contrast, the `Dissimilarity` score gives a variation that correlates with the downstream analysis task's performance showing that it is sensitive to the quality of the data and the `Dissimilarity` value reflects the expected accuracy.

D. *RQ4: Validate the quality of synthesized samples using the `Dissimilarity` score*

In this experiment, we visualize the relationships between real and the unconditionally synthesized samples, in context of use case I, as a function of the `Dissimilarity` score. Specifically, using the Handwritten Digits dataset, we visualize the 2-D representations of the real dataset, a set synthesized from a model that does poorly in terms of MLE, and a set from a model that does well in terms of to investigate the local structures of the datasets. We also observe similar patterns in other datasets; however, we omit including them due to the page limit. To plot the 2-D representations, we leverage the Uniform Manifold Approximation and Projection (UMAP) [30] dimension reduction method. UMAP is known for it's capability of capturing complex non-linear relationships and preservation of local structure present within the data. We use this capability to our advantage and look at the local structures present within the datasets and then see the correlation between the similarities in structure and the `Dissimilarity` score.

In Figure 5, we observe similar clusters across all three datasets for the nine classes. However, in Figure 5a, we observe the clusters to be homogeneous which corroborates with the dataset achieving a high accuracy of 97%. On the other hand, 5b shows clusters of various colored dots mixed in. This set of samples achieved an 85 `Dissimilarity` score, which we clearly see correlated with higher dissimilarity with 5a. Additionally, these samples achieved an accuracy of 21%. In 5c, although we observe some noises, each cluster predominantly contains dots of the same color. Here, the `Dissimilarity` score is 4 and the MLE is 91%.

It's worth noting that the SDQuality scores for both the synthetic sets (5b and 5c) are 99.99%, which do not correlate with the appearance of the local clusters. This visualization validates our hypothesis that leveraging subspace distance and distribution similarity together creates a stronger distance measure that can indicate whether the synthesized data will enable a downstream analysis technique to perform as well as the real data.

E. *Discussions*

The SDQuality is not able to corroborate effectively with the MLE metric. This is because it primarily focuses on the variability of data within a dataset but may not capture other aspects of data quality, such as accuracy, completeness, relevance, or consistency. As a result, relying solely on SDQuality may overlook important data quality dimensions crucial for many applications. Additionally, SDQuality is sensitive to outliers or extreme values in the dataset, which can disproportionately influence its calculation and provide misleading

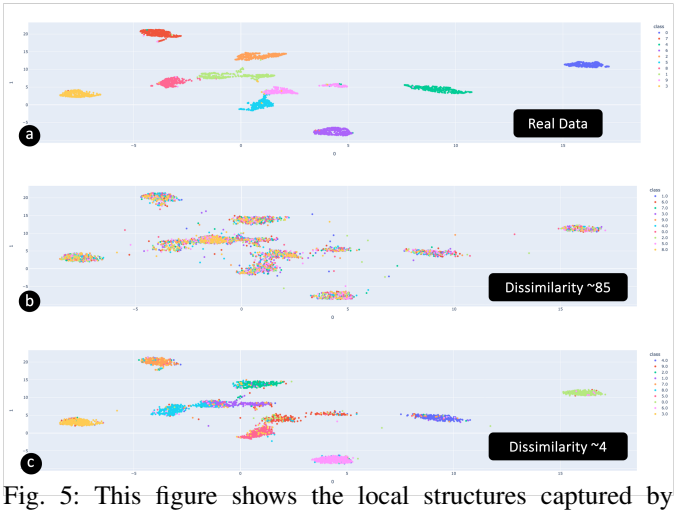

Fig. 5: This figure shows the local structures captured by UMAP for (a) Real Data where we see almost homogeneous clusters each color representing a digit from 0-9, (b) Synthesized Data with `Dissimilarity` score of 85, (c) Synthesized Data with `Dissimilarity` score of 4. We see the data with lower `Dissimilarity` has cleaner and more homogeneous clusters whereas the one with higher `Dissimilarity` has pretty random clusters showing that `Dissimilarity` reflects the similarities within the structure of the datasets.

assessments of overall data quality. Similarly, KL-Divergence only looks at the relationship between individual samples, and so if a synthesized dataset has similar statistical properties, it will result in a low divergence, however, holistic similarity in the dataset might not exist.

This is where considering the low-dimensional representation that reduces noise and looking at the sub-spaces gives a much better understanding of the differences between the datasets. Such a measure is very valuable as a large percentage of all the data available to us is in tabular format, and people are generating synthesized tabular data samples for various applications in HPC, finance, medicine, and education, among others. **Our thorough evaluations demonstrate that leveraging LLMs for decision making for HPC datasets is a promising route.**

VI. Conclusion

This work addresses the two challenges of (1) performance data generation, (2) quality of synthetic data evaluation in a quantitative manner. Our experiments with three HPC and five ML dataset demonstrate that `Dissimilarity` measure correlates with model accuracy better than the state-of-the-art metrics, implying that `Dissimilarity` is a better evaluation metric for quantifying the quality of generated data. For performance data generation from the perspective of both prediction and decision making, an LLM based model seems to be a better choice.

Acknowledgement

This material is based upon work supported by the U.S. Department of Energy, Office of Science under Award Number DE-SC0022843.

References

[1] Eyal Betzalel, Coby Penso, Aviv Navon, and Ethan Fetaya. A study on the evaluation of generative models, 2022.

[2] Giovanni S. Alberti, Matteo Santacesaria, and Silvia Sciutto. Continuous generative neural networks, 2023.

[3] Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. Modeling tabular data using conditional gan, 2019.

[4] Aoting Hu, Renjie Xie, Zhigang Lu, Aiqun Hu, and Minhui Xue. Tablegan-mca: Evaluating membership collisions of gan-synthesized tabular data releasing, 2021.

[5] Zilong Zhao, Aditya Kunar, Hiek Van der Scheer, Robert Birke, and Lydia Y. Chen. Ctab-gan: Effective table data synthesizing, 2021.

[6] Bingyang Wen, Yupeng Cao, Fan Yang, Koduvayur Subbalakshmi, and Rajarathnam Chandramouli. Causal-TGAN: Modeling tabular data using causally-aware GAN. In *ICLR Workshop on Deep Generative Models for Highly Structured Data*, 2022. URL https://openreview.net/forum?id=BEhxCh4dvW5.

[7] Geoffrey E. Hinton and Richard S. Zemel. Autoencoders, minimum description length and helmholtz free energy. In *Proceedings of the 6th International Conference on Neural Information Processing Systems*, NIPS'93, page 3–10, San Francisco, CA, USA, 1993. Morgan Kaufmann Publishers Inc.

[8] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[9] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[10] Shan Yang, Lei Xie, Xiao Chen, Xiaoyan Lou, Xuan Zhu, Dongyan Huang, and Haizhou Li. Statistical parametric speech synthesis using generative adversarial networks under a multi-task learning framework. In *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 685–691, 2017. doi: 10.1109/ASRU.2017.8269003.

[11] Karim Armanious, Chenming Jiang, Marc Fischer, Thomas Küstner, Tobias Hepp, Konstantin Nikolaou, Sergios Gatidis, and Bin Yang. MedGAN: Medical image translation using GANs. *Computerized Medical Imaging and Graphics*, 79:101684, jan 2020. doi: 10.1016/j.compmedimag.2019.101684. URL https://doi.org/10.1016%2Fj.compmedimag.2019.101684.

[12] Arnab Paul, Jong Youl Choi, Ahmad Maroof Karimi NLN, and Feiyi Wang. Machine learning assisted hpc workload trace generation for leadership scale storage systems. 6 2022.

[13] Wenxiang Jiao, Wenxuan Wang, Jen tse Huang, Xing Wang, Shuming Shi, and Zhaopeng Tu. Is chatgpt a good translator? yes with gpt-4 as the engine, 2023.

[14] Minh Duc Vu, Han Wang, Zhuang Li, Jieshan Chen, Shengdong Zhao, Zhenchang Xing, and Chunyang Chen. Gptvoicetasker: Llm-powered virtual assistant for smartphone, 2024.

[15] 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*, 2023. URL https://openreview.net/forum?id=cEygmQNOeI.

[16] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Huggingface's transformers: State-of-the-art natural language processing, 2020.

[17] *Synthetic Data Metrics*. DataCebo, Inc., 4 2023. URL https://docs.sdv.dev/sdmetrics/. Version 0.9.3.

[18] Frank J. Massey Jr. The kolmogorov-smirnov test for goodness of fit. *Journal of the American Statistical Association*, 46(253):68–78, 1951. doi: 10.1080/01621459.1951.10500769. URL https://www.tandfonline.com/doi/abs/10.1080/01621459.1951.10500769.

[19] Arnab Bhattacharyya, Sutanu Gayen, Kuldeep S. Meel, Dimitrios Myrisiotis, A. Pavan, and N. V. Vinodchandran. On approximating total variation distance, 2022.

[20] Philip Sedgwick. Pearson's correlation coefficient. *Bmj*, 345, 2012.

[21] Jerrold H Zar. Spearman rank correlation. *Encyclopedia of biostatistics*, 7, 2005.

[22] Carol E Hicks, James E Rush, and Suzanne M Strong. Contingency table analysis. *Encyclopedia of Computer Science and Technology: Volume 6-Computer Selection Criteria to Curriculum Committee on Computer Science*, 1977.

[23] Vikram S Chundawat, Ayush K Tarun, Murari Mandal, Mukund Lahoti, and Pratik Narang. Tabsyndex: A universal metric for robust evaluation of synthetic tabular data, 2022.

[24] Agustin Garcia Asuero, Ana Sayago, and AG González. The correlation coefficient: An overview. *Critical reviews in analytical chemistry*, 36(1):41–59, 2006.

[25] Joshua Snoke, Gillian Raab, Beata Nowok, Chris Dibben, and Aleksandra Slavkovic. General and specific utility measures for synthetic data, 2017.

[26] Jonathon Shlens. Notes on kullback-leibler divergence and likelihood, 2014.

[27] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, R. Prettenhofer, P.and Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[28] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2022.

[29] R Bro and AK Smilde. Principal component analysis. 2014.

[30] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction, 2020.

[31] Tin Kam Ho. Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition*, volume 1, pages 278–282. IEEE, 1995.

[32] Xindong Wu, Vipin Kumar, J Ross Quinlan, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, Geoffrey J McLachlan, Angus Ng, Bing Liu, S Yu Philip, et al. Top 10 algorithms in data mining. *Knowledge and information systems*, 14(1):1–37, 2008.

[33] Jürgen Groß. *Linear regression*, volume 175. Springer Science & Business Media, 2003.

[34] David R Cox. The regression analysis of binary sequences. *Journal of the Royal Statistical Society: Series B (Methodological)*, 20(2):215–232, 1958.

[35] Tapasya Patki, Jayaraman J Thiagarajan, Alexis Ayala, and Tanzima Z Islam. Performance optimality or reproducibility: That is the question. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–30, 2019.

[36] A Marathe, R Anirudh, N Jain, A Bhatele, J Thiagarajan, B Kailkhura, J S Yeom, B Rountree, and T Gamblin. Performance modeling under resource constraints using deep transfer learning. 8 2017. doi: 10.1145/3126908.3126969. URL https://www.osti.gov/biblio/1410070.

[37] Aniruddha Marathe, Rushil Anirudh, Nikhil Jain, Abhinav Bhatele, Jayaraman Thiagarajan, Bhavya Kailkhura, Jae-Seung Yeom, Barry Rountree, and Todd Gamblin. Performance modeling under resource constraints using deep transfer learning. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–12, 2017.

[38] Kaggle. California housing prices, . URL https://www.kaggle.com/datasets/camnugent/california-housing-prices.

[39] Athanasios Tsanas and Max Little. Parkinsons Telemonitoring. UCI Machine Learning Repository, 2009. DOI: https://doi.org/10.24432/C5ZS3N.

[40] R. Bock. MAGIC Gamma Telescope. UCI Machine Learning Repository, 2007. DOI: https://doi.org/10.24432/C52C8B.

[41] E. Alpaydin and C. Kaynak. Optical Recognition of Handwritten Digits. UCI Machine Learning Repository, 1998. DOI: https://doi.org/10.24432/C50P49.

[42] Kaggle. Home equity line of credit(heloc), . URL https://www.kaggle.com/datasets/averkiyoliabev/home-equity-line-of-creditheloc.

[43] Lukas Biewald. Experiment tracking with weights and biases, 2020. URL https://www.wandb.com/.