AIGT: AI Generative Table Based on Prompt

Mingming Zhang[1*] **, Zhiqing Xiao**[1*]**, Guoshan Lu**[1*]**, Sai Wu**[1]**,**
Weiqiang Wang[2]**, Xing Fu**[2]**, Can Yi**[2]**, Junbo Zhao**[1†]
[1] Zhejiang University, Hangzhou, China
[2] Ant Group, Hangzhou, China
{mmz,zhiqing.xiao,luguoshan,wusai,j.zhao}@zju.edu.cn
{weiqiang.wwq,zicai.fx,yican.yc}@antgroup.com

Abstract

Tabular data, which accounts for over 80% of enterprise data assets, is vital in various fields. With growing concerns about privacy protection and data-sharing restrictions, generating high-quality synthetic tabular data has become essential. Recent advancements show that large language models (LLMs) can effectively generate realistic tabular data by leveraging semantic information and overcoming the challenges of high-dimensional data that arise from one-hot encoding. However, current methods do not fully utilize the rich information available in tables. To address this, we introduce **AI** **G**enerative **T**able (AIGT) based on prompt enhancement, a novel approach that utilizes metadata information, such as table descriptions and schemas, as prompts to generate ultra-high-quality synthetic data. To overcome the token limit constraints of LLMs, we propose long-token partitioning algorithms that enable AIGT to model tables of any scale. AIGT achieves state-of-the-art performance on 14 out of 20 public datasets and two real industry datasets within the Alipay risk control system.

1 Introduction

Given the prevalence of tabular data in practical applications, synthesizing high-quality tabular data is an essential task. It ensures privacy protection, enhances the generalization capabilities of machine learning models, and boosts performance in scenarios with limited samples. However, this task is fraught with unique challenges, including specificity, impurities, class imbalances, and privacy concerns.

Traditional approaches to tabular data synthesis, such as generative models and statistical methods, often lose textual information and struggle with capturing complex feature relationships (Choi et al., 2017; Park et al., 2018; Xu et al., 2019a; Borisov et al., 2022). Recent efforts have explored the use of language models to incorporate feature names for better contextual learning. For instance, GReaT (Borisov et al., 2023) made the pioneering attempt to generate tabular data using large language models (LLMs), achieving satisfactory synthetic data. TapTap (Zhang et al., 2023) further enhanced the quality of synthetic data through table pre-training and applied it to data augmentation in tabular data for improved performance in prediction tasks, achieving state-of-the-art (SOTA) results. However, these methods are constrained by token limits, limiting their ability to handle arbitrarily wide tables. Additionally, they fail to fully utilize crucial tabular elements such as headers and column names, resulting in incomplete information integration.

Recently, Artificial Intelligence Generation and Creation (AIGC) technologies, including large-scale language models and automatic image generation techniques, are driving progress in AI's capabilities for content creation and data generation. Prompt Learning, as an innovative approach, aligns the pre-training of language models with specific downstream tasks. By using well-designed prompts, which include task descriptions, input data, background information and output indicators, create a template that directs the model's attention and thus improves the accuracy and relevance of the generated context.

AIGC is more broadly applied to the generation of multimedia content, such as images, videos, and audio. Combined with Prompt Learning, AIGC systems can generate content more accurately according to user instructions or needs. However, The application of AIGC technology in the generation of tabular data is currently relatively limited. Although work on table generation based on large-scale language models has begun to explore, these efforts have mostly utilized only the cell values in the tables, without fully leveraging

[*] Indicates equal contribution.
[†] Corresponding author.

Proceedings of the 31st International Conference on Computational Linguistics, pages 9926–9938
January 19–24, 2025.

the more comprehensive information contained in the tabular data. We recognize that table metadata, such as headers and column names, provides valuable context often overlooked in analysis. Inspired by AIGC and Prompt learning, we propose a prompt-enhanced model that leverages this metadata to improve performance. Therefore, we propose **AI G**enerative **T**able (AIGT) technique based on prompt enhancement, which leverages metadata to enhance the quality and relevance of table generation.

Methods based on language models inherently face limitations in generating long sequences of tokens, which is why previous approaches like GReaT and TapTap struggled to model tables with a large number of columns—something very common in real-world industry data. To address this, we developed a long-token partitioning algorithm specifically adapted for AIGT. This approach allows AIGT to effectively handle data synthesis tasks for tables of any size, overcoming the token length constraints. We further demonstrate AIGT's effectiveness in real-world scenarios such as Alipay's risk control system for commercial credit and merchant fraud detection.

Our key contributions are as follows: (1) **A Prompt-Enhanced LM for Tabular Data Synthesis:** We utilize the metadata of tables to construct prompt-enhanced language model and design a series of training techniques to enhance the capability of table generation. (2) **Scalability:** Our proposed partitioning algorithm enables LM-based generation methods to be scaled to tables of any size, overcoming the token limit constraints of LM methods. (3) **Superior Performance:** We achieve state-of-the-art results on 14 out of 20 academic datasets and two real-world industry datasets from Alipay.

2 Related Works

In this section, we provide a brief background on prompt engineering and tabular data synthesis approaches.

2.1 Prompt Engineering

Prompt engineering has recently garnered significant attention as a technique for enhancing the performance and usability of AI models, especially in natural language processing (NLP). As described by (Lester et al., 2021), a "prompt" involves providing natural language instructions or commands to guide an AI model in task completion. This approach offers several benefits, including increased model effectiveness, reduced training time and costs, and improved interpretability and controllability. Notable works in this area include those by (Jiang et al., 2020; Liu et al., 2023; Brown et al., 2020).

2.2 Tabular Data Synthesis

Existing approaches for tabular data synthesis can be categorized into four main groups:

Probabilistic Models. These models leverage probabilistic techniques to synthesize data. For instance, Gaussian copula models (Patki et al., 2016) are effective for continuous variables but not for categorical ones. Conversely, Bayesian networks (Zhang et al., 2017; Aviñó et al., 2018) are adept at handling categorical data but struggle with continuous variables.

Generative Adversarial Networks. Generative Adversarial Networks (GANs) have been widely used to generate tabular data. MedGAN (Armanious et al., 2020) and RGAN (Esteban et al., 2017) produce healthcare records but face challenges with mixed data types. TableGAN (Park et al., 2018) employs Convolutional Neural Networks (CNNs), demonstrating that synthetic data can perform comparably to real data. CTGAN and TVAE (Xu et al., 2019a) address multimodality with column-specific preprocessing and Variational Gaussian Mixture (VGM) models. Other notable works include (Xu et al., 2019b; Marti, 2020; Jordon et al., 2018; Che et al., 2017).

Diffusion Models. These approaches utilize diffusion models for data synthesis. TabDDPM (Kotelnikov et al., 2023) models both categorical and continuous values but encounters difficulties with correlations. SOS (Kim et al., 2022) uses Score-based Generative Models (SGMs) to handle imbalanced data, though it lacks the ability to condition on both data types.

Language Models. Self-attention models, which have revolutionized NLP (Vaswani et al., 2017), have also been adapted for tabular data synthesis. These include encoding models (Lan et al., 2020; Devlin et al., 2018), sequence-to-sequence models (Raffel et al., 2020), and auto-regressive models (Radford et al., 2019). Transformers have been applied to table classification (Gorishniy et al., 2021) and joint table-text representations (Tang et al., 2020; Gong et al., 2020). GReaT (Borisov et al., 2023) and TapTap (Zhang et al., 2023) gener-

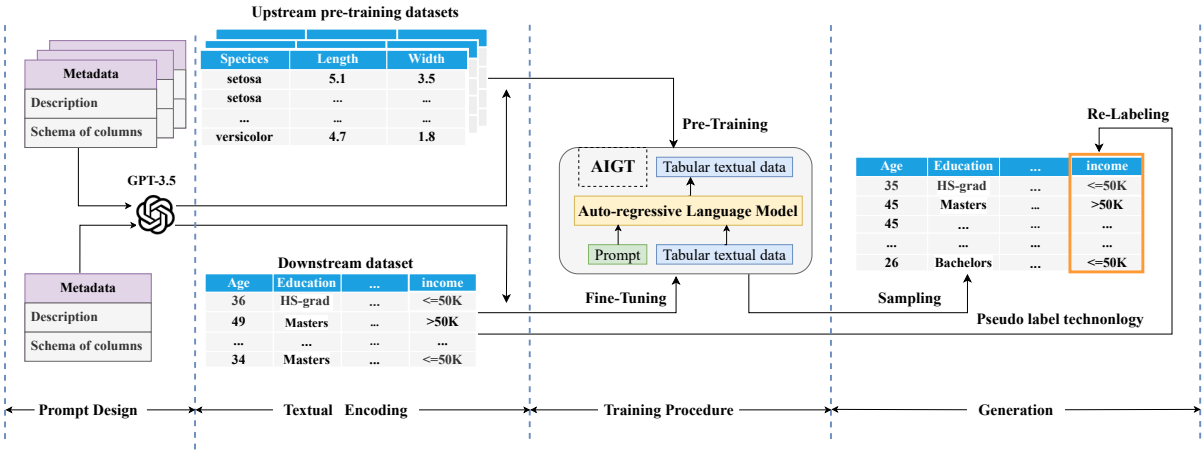

Figure 1: The architecture of the proposed AIGT. Firstly, AIGT utilizes the collected pre-trained corpus for pre-training; Then perform fine-tuning training on each downstream table to learn the complex relationships between features; Finally, based on the trained language model model, sample rows can be composed.

ate synthetic tables but encounter limitations with wide tables.

3 The Task of Tabular Data Synthesis

To find a data synthesizer G learnt from a table D and using G to generate a synthetic table D_{syn}. The objective of table generation is to produce data that is similar in distribution to the original data. We evaluate the generator G from multiple perspectives: (1) Machine Learning Efficiency: When we train a classifier or regressor on the generated dataset, can it achieve the accuracy achieved by training on the original dataset? (2) Data Augmentation: When we add synthetic data to the original data, will it enhance the classification/regression task? (3) The difference between the synthetic data and the original data. We hope that the synthetic data is not a copy of the original data.

4 Methods

This section introduces the AIGT method for generating tabular data using prompt-based enhancement. As illustrated in Figure 1, AIGT comprises five main stages: (1) **Prompt Design:** Construct a prompt based on the table's caption information and column names. (2) **Textual Encoding:** Convert table features and their values into sentences, concatenate these into prompts, and construct data suitable for model input. (3) **Training Procedure:** Utilize a pre-trained Large Language Model (LLM) on an extensive corpus, and then conduct specific fine-tuning for downstream tables. (4) **Generation:** Generate samples using the fine-tuned auto-regressive language model. To support tables of any size, unrestricted by the number of features, we proposed a **Partitioning Algorithm for Long Tokens** to enhance the scalability of our model.

4.1 Prompt Design

To enhance the understanding of table structures and semantics in language models, we utilize the metadata of tables to construct prompts. In our framework, the metadata of the table comprises information from two parts.

- Caption information: the description of the dataset, including the purpose of the dataset, background information.
- Feature information: the names of features, the target column for prediction, and the meanings associated with the features, especially the meanings of certain abbreviations.

We define the function **prompt** that can process metadata, which is implemented by calling the GPT3.5[1]. The corresponding code is available in the appendix B.

4.2 Textual Encoding

Feature Serialization. The standard large language model expects text as input. Thus, AIGT transforms each row from the dataset into a text format. We follow the previous work (Borisov et al., 2023) by serializing each sample into a text sequence. By concatenating the feature names and values of the table into sentences, that is, "[Feature] is [Value]". Considering that tabular data follows

[1]OpenAI API: gpt-3.5-turbo

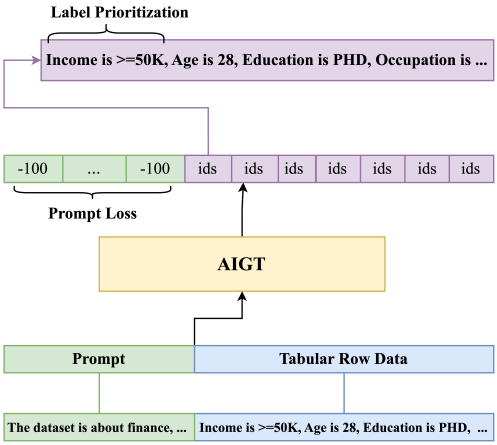

Figure 2: Training Strategies for AIGT. Here, Prompt's losses are not calculated, the label feature was fixed in the first place, the other features were pre-mutated randomly.

the invariance of feature arrangement, we apply an arrangement function P to randomly shuffle the order of features when encoding a table. Formally, given a table $D = \{(x_i, y_i)\}$, Let x_{ij} be the j-th feature value of the i-th sample and F_i represents the feature name in the i-th column. The textual encoding is to transform the i-th sample x_i into a splice of sentences separated by commas $T_i = (T_{i,k_1}, T_{i,k_2}, ..., T_{i,k_m})$, where $[k_1, k_2, ..., k_m] = P([1, 2, ..., m])$ and $T_{i,j} = (F_j \ is \ x_{ij})$ and m is the total number of columns.

Label Prioritization. We believe that the meaning of features and labels in tables is different, and labels are a summary of all features. It is crucial to prioritize the label column during the serialization process. By placing the label column at the forefront, we ensure that the model can immediately recognize the target variable, which can enhance the overall understanding and performance of the model. Therefore, given a table $D = \{(x_i, y_i)\}$ along with a prompt, which is generated based on the table metadata (refer to Section 4.1), where F_y represents the name of the label, the encoded sentence becomes $T_i = (\textbf{prompt} \ F_y \ is \ y_i, T_{i,k_1}, T_{i,k_2}, ..., T_{i,k_m})$, where $[k_1, k_2, ..., k_m] = P([1, 2, ..., m])$.

Prompt-Enhanced Loss. Similar to the loss design of most prompt engineering methods (Lester et al., 2021), AIGT's training strategy only calculates non prompt loss calculations and ignores the loss calculations of the prompt itself, to achieve better table generation quality. We show these strategies in Figure 2.

4.3 Training Procedure

Pre-Training. We perform pre-training on a large-scale dataset upstream. Specifically, following the first two steps, we convert each row of table data into text to form the pre-training corpus $\mathcal{T}$. Each sentence $t \in \mathcal{T}$ can be encoded into a sequence of tokens using $tokenize(t) = (w_1, ..., w_N)$. In general, AIGT factorizes the probability of generating t in an auto-regressive manner as $p(t) = \prod_{i=1}^{N} p(w_i | w_1, ..., w_{i-1})$. During pre-training, AIGT is optimized towards maximizing the probability $\prod_{i=1}^{\|\mathcal{T}\|} p(i)$ on the entire pre-training corpus. The pre-training process can initiate with an auto-regressive language model, thereby capitalizing on the extensive knowledge that these models have already acquired.

Fine-tuning. The fine-tuning of AIGT on downstream tables follows a similar process as pre-training. The only difference is that fine-tuning aims to target specific downstream tables.

4.4 Generation

Sampling. We have trained an auto-regressive model $\mathbf{q}$ through fine-tuning on the text training dataset. This model predicts the potential subsequent labels $w_1, ..., w_{k-1}$ for the classification output distribution $z = \mathbf{q}(w_1, ...w_{k-1})$. Multiple sampling strategies can be utilized in this scenario. Typically, the next token w is selected through weighted sampling from the output z of the LLM, guided by a temperature parameter $T > 0$,

$$p(w | w_1, ..., w_{k-1}) = \frac{e^{(z_w / T)}}{\sum_{w' \in \mathcal{W}} e^{(z_{w'} / T)}}. \quad (1)$$

Following GReaT (Borisov et al., 2023), the model is initialized with specific conditions and LLM is tasked with sampling the remaining tokens to complete the feature vector in its textual representation. Although we fix the position of the label in the first place during training, it is possible to generate data according to a specific feature column; one simply needs to prepend the corresponding label. For example, the provided condition can be "**prompt** $[Label]$ is $[Value]$". Here, the prompt is generated from the semantic information of the table metadata as Section 4.1.

Re-Labeling. We observe that in TapTap (Zhang et al., 2023), the re-labeling through the table prediction model effectively enhances the ability to

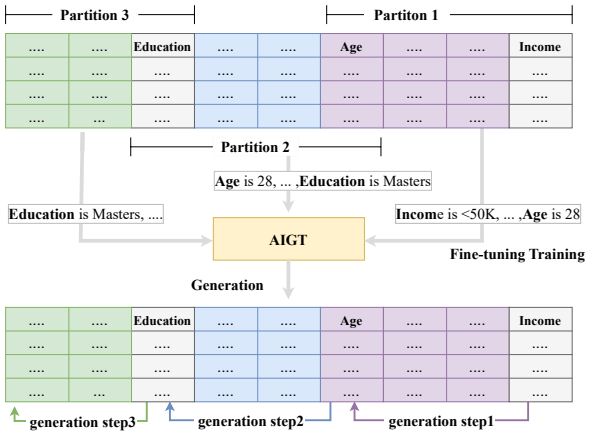

Figure 3: The Process of Long Token Partition Algorithm in AIGT. Divide the table into sub-tables based on columns, with overlaps between the columns of the sub-tables. Use the data from the sub-tables to train and generate the model.

predict labels of synthesized data. Given the original train datasets $D = \{(x_i, y_i)\}$, similar to labeling technique, we fine-tune AIGT on it and generate synthesized data $D' = \{(x_i', y_i')\}$. Then, a tabular predictor P, *e.g.*, LightGBM, is trained to fit D, and then the synthesis label y_i' can be replaced by $y_i' = P(x_i')$.

4.5 Partitioning Algorithm for Long Tokens

In practical industrial scenarios, datasets are large with numerous features, leading to the number of tokens to exceed LLM input limits. To address this scalability challenge for LLM-based methods, we propose a long-token partitioning algorithm that integrates seamlessly with existing training and generation methods.

Partition Training. As shown in the Figure 3, first, we partition the features to ensure that there are some overlapping columns between each region, and then perform mixed training to enable the large language model to learn the feature associations of different partitions. The function of the cover column is to alleviate to some extent the missing feature associations between different regions. In our industrial scenario experiment, the number of overlapping columns is set to 1, and due to the small number of features on academic datasets, partitioning algorithms are not required.

Partition Generation. When generating, it is generated from the back to the front. After each partition is generated, the remaining partitions are generated using the cover part of the generated partition as the starting distribution column, and finally merge the partitions.

5 Experiments

In Section 5.1, we outline the datasets, the baseline methods utilized in our experiments. In Section 5.2, we conducted extensive experiments, encompassing the machine learning efficiency of generated data, Distance to Closest Record (DCR) distance, data augmentation, and the application of our partitioning algorithm across both public and industrial datasets. To further show the excellence of our approach, we include additional analysis in the appendix C, such as discriminative metrics and feature statistical similarity between generated and original data.

5.1 Experimental Setup

Datasets. The experimental dataset consists of three parts as following:

(1) **Upstream Large-scale Cross-table Pretraining Dataset.** We collected approximately 1000 tables with their metadata, sourced primarily from OpenML [2], referred to as STABS. We have made STABS open-source in the hope of contributing to and advancing the community focused on pre-training tabular data. More information about STABS can be seen in the appendix A.1.

(2) **Downstream Open Public Dataset.** For downstream tabular tasks, we utilize a diverse set of 20 public benchmark tabular datasets to test the efficacy of our model. These datasets, sourced from OpenML, UCI repository and Kaggle, contain both binary, multi-class classification and Regression tasks. Information regarding each dataset is provided in the Table 1 .

(3) **Ailpay Dataset.** Datasets in real-world industrial settings generally comprise a multitude of column names. To evaluate the efficiency and performance of long token partitioning, we utilized two industrial datasets from Alipay's risk control system. We show the details in Table 2.

Baseline Methods. We evaluate AIGT alongside five other SOTA tabular data synthesis algorithms: CTGAN, TVAE, TabDDPM, GReaT and TapTap. CTGAN (Xu et al., 2019b) based on generative adversarial networks (Creswell et al., 2018) for tabular data, allowing the generation process to be conditional only on a single discrete feature. The same author proposed TVAE (Xu et al., 2019b),

[2]https://www.openml.org/

Abbr	Name	#Samples	# Num	# Cat	Task type
BW	Breast-w	699	9	0	Binclass
CG	Credit-g	1000	7	13	Binclass
DI	Diabetes	768	8	0	Binclass
EL	Electricity	45312	8	0	Binclass
SI	Sick	3772	7	22	Binclass
AD	Adult	48842	15	6	Binclass
WI	Wilt	4839	6	5	Binclass
CM	Cmc	1473	9	0	Multiclass
VH	Vehicle	846	18	0	Multiclass
SA	Satimage	6430	36	0	Multiclass
AF	Analcatdata_dmft	661	2	2	Multiclass
CR	Car	1728	0	6	Multiclass
SE	Segment	2310	15	0	Multiclass
EU	Eucalyptus	736	14	5	Multiclass
LO	Loan	5000	12	0	Binclass
HE	Heloc	10460	22	1	Binclass
CA	California Housing	20640	8	0	Regression
AG	Crab Age	3894	7	1	Regression
IN	Insurance	1338	3	3	Regression
KI	King	21613	18	1	Regression

Table 1: Information of Downstream Tabular Datasets. Dataset abbreviations are in parentheses. # Num and # Cat represent the number of continuous and categorical columns.

Dataset Name	# columns	# train	# valid	# test	PR
SYH	251	466320	51814	29187	0.06
NonBD	45	6148904	1537226	960766	0.05

Table 2: Statistical Information of Alipay Datasets. The two tables come from Alipay's financial risk control system, and their goals are binary, PR means the proportion of instances that are labeled as positive.

a variational autoencoder (VAE) for tabular data. TabDDPM (Kotelnikov et al., 2023) adapts diffusion models to tabular data. GReaT and TapTap, which utilize language-based approaches, adopt a pre-trained DistilGPT-2 model. This DistilGPT-2 framework also serves as the foundational model for AIGT. To illustrate the potential of larger models, our approach incorporates Llama3.1-8B as a foundational model, denoted as AIGT-L.

Baseline Implementation. For CTGAN and TVAE, we set the training epochs to 300, except for those datasets that have less than 5k data. Due to their small sample size, we will set a larger number of training epochs to 500, to ensure better training results on these small datasets. For the diffsion method TabDDPM, we employ default settings. For GReaT, TapTap and AIGT, we use the distilGPT-2 as framework. We pretrained AIGT for 10w steps with the learning rate 1×10^{-4} . We finetune the GReaT, TapTap and AIGT for 100 epoch. The batch size is 32 for all datasets. We use the AdamW optimizer for the proposed generative models, with the learning rate 5×10^{-5}. For AIGT-L, We use the AdamW optimizer with the learning rate 1×10^{-5}.

Environment Experiments run on a machine equipped with 4 NVIDIA A100-SXM4-80GB GPU and 100 GB RAM , Intel(R) Xeon(R) Platinum 8369B CPU @ 2.90GHz CPU under Ubuntu 20.04 with 64 cores.

5.2 Overall Performance

In this section, we will demonstrate the performance of the proposed AIGT method through multiple experiments. Additionally, we also present the effects of the partitioning algorithm on both public and industrial datasets.

Machine Learning Efficiency (MLE). In this section, we compare AIGT to alternative generative models in terms of machine learning efficiency. Each dataset was split into two parts: 80% for training purposes and 20% reserved for testing. Initially, each generative algorithm is trained on the training data. Subsequently, the trained model is utilized to generate synthetic data of equivalent size. This synthetic data is then used to train a classification/regression model, which is then evaluated using the real test set. We expect that for high-quality synthetic data, models trained on this data will perform comparably to those trained on real data. To assess the effectiveness of the machine learning models, we apply the LightGBM model, a leading GBDT method, to evaluate their efficiency. We adopt the AUC score as the evaluation metric for classification tasks and employ R^2 score for regression tasks. For a fair comparison we use the the standard hyper-parameter tuning budget of 50 trials. Our full search space is provided in the Appendix D and all the experimental results are averaged over 10 different random seeds. The result was showed in Table 3, Note that we match or exceed state-of-the-art on 14 out of 20 datasets.

Distance to closest records histogram To verify that the generated data is similar to the original sample, rather than an exact replica, this metric calculates the distance from the nearest record in the original training dataset D_{train}. For each synthesized record s, it is given by DCR (s)=$\min\{distance(s, s_i)|s_i \in D_{train}\}$. As a distance measure, we use the L1 norm for numerical features. For categorical features, we set the difference to 0, otherwise it is set to 1. Note that models such as CTGAN and TabDDPM have a fixed

Dataset	BW	CM	CG	DI	VH	EL	SA	EU	SI	AF
Real	99.2±0.1	74.9±0.1	76.9±1.8	82.7±1.7	93.9±0.0	97.6±0.1	99.1±0.0	89.1±0.0	96.3±0.5	52.5±1.1
CTGAN	97.5±2.6	50.1±3.1	52.4±4.0	75.0±2.6	58.8±2.4	82.4±0.4	95.4±0.2	53.1±3.5	65.8±0.8	53.8±2.4
TVAE	98.9±0.3	59.3±0.9	74.8±2.1	80.3±0.9	86.6±0.4	84.0±1.1	97.3±1.0	84.9±0.8	93.2±1.5	53.0±2.3
TabDDPM	**99.0±0.2**	71.7±0.8	71.0±1.2	78.5±2.2	59.8±0.7	89.1±0.1	78.8±4.2	69.5±0.7	96.5±0.7	52.5±0.6
GReaT	98.2±0.4	56.4±3.2	53.9±3.1	59.2±0.3	64.3±4.5	92.3±0.3	95.9±0.3	81.1±2.1	96.2±1.2	53.1±2.2
TapTap	**99.0±0.2**	71.5±0.4	76.3±1.6	79.7±0.2	90.8±1.6	91.1±0.2	97.8±0.2	86.8±0.7	**98.4±0.8**	53.2±0.7
AIGT	98.5±0.1	71.6±0.6	**78.1±2.3**	81.7±0.5	**91.5±1.1**	**94.3±0.2**	97.8±0.1	**86.9±1.0**	97.9±1.6	53.8±0.9
AIGT-L	98.2±0.3	**72.1±0.6**	76.5±1.3	**82.2±0.1**	**91.5±0.1**	93.3±0.5	**97.9±0.4**	86.6±0.5	97.8±1.4	**54.4±1.0**

Dataset	AD	WI	CR	SE	LO	HE	IN	AG	CA	KI
Real	92.7±1.3	99.1±0.0	100.0±0.0	99.3±0.0	99.8±0.0	80.9±0.1	88.0±0.0	53.2±0.3	85.3±0.2	87.8±0.1
CTGAN	89.4±0.2	53.2±3.6	56.7±1.9	86.6±2.0	49.0±1.4	40.6±2.0	37.7±6.4	27.6±1.6	50.1±0.1	55.3±1.8
TVAE	87.4±0.4	80.6±1.8	89.1±2.3	97.7±0.2	97.2±0.4	61.0±4.0	61.0±3.2	20.5±1.0	69.0±0.3	72.3±2.5
TabDDPM	90.6±0.1	**98.9±0.3**	99.3±0.2	98.3±0.1	98.6±0.2	74.6±0.5	81.0±0.1	45.8±0.3	80.0±0.2	75.0±3.0
GReaT	**90.8±0.3**	96.1±1.1	91.4±2.0	98.1±2.0	97.8±0.2	75.3±0.4	57.9±5.1	41.2±0.8	75.6±0.3	61.8±2.1
TapTap	90.0±0.4	97.7±0.9	99.4±0.2	**98.6±0.1**	**99.5±0.1**	79.8±0.3	**88.0±0.1**	51.4±0.3	82.6±0.1	87.7±0.3
AIGT	89.8±0.2	97.4±0.8	99.8±0.1	**98.6±0.1**	99.4±0.2	**80.3±0.1**	87.9±0.2	**53.3±0.2**	82.4±0.1	**88.0±0.2**
AIGT-L	86.7±1.4	98.8±0.2	**99.9±0.7**	98.4±0.1	99.4±0.2	79.0±0.3	87.4±0.5	53.1±0.5	**83.2±0.1**	87.5±0.4

Table 3: ML efficiency experiment. We used 20 real-world datasets, for classification tasks, auc score is reported. For regression datasets, R^2 is reported. The values of machine learning efficiency computed with regards to the state-of-the-art tuned LightGBM model. The best results are marked in bold, while the second best results are marked with underscores.

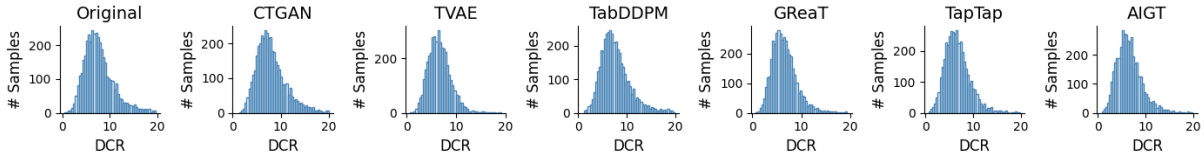

Figure 4: Distance to closest record (DCR) distribution for the California Housing dataset. "Original" denotes the DCR of the original test set with respect to the original train set. The experimental results illustrate that each method does not copy samples from the train set.

DCR score. However, generative algorithms based on language models can produce more novel samples by adjusting the temperature coefficient. For the sampling step of GReaT, TapTap and AIGT, we set the temperature parameter T to 0.7 for all datasets. We compare the distribution of the minimal distances of the generated samples to the training data set. The visualization of the minimum distance distribution can be found in Figure 4.

Data Augmentation. When dealing with datasets that are relatively small in sample size, our approach is to boost the performance of machine learning models by supplementing these datasets with synthetic data. We integrate the synthetic data with the original training data to create an augmented training set. Following this, we conduct model training on this enlarged training set and assess its performance on the test set. The results are presented in Table 4, which show that AIGT is able to perform better than all baseline methods on most datasets.

Effectiveness of Partition Generation Algorithm. To validate our partitioning algorithm, we selected datasets with more than 20 columns from public datasets and two Real-world industrial Ailpay datasets. We divided the academic datasets and the NonBD table into two partitions, and partitioned the SYH table into five segments. Utilizing the synthetic data generated through our partitioning algorithm, we evaluated the ML efficiency of the synthetic datasets and compared it with the generation methods that do not employ language models. The results of the partitioning algorithm are shown in Table 5. It can be seen that even with the partitioning algorithm, our method still outperforms models not using language models in efficiency.

Dataset	BW	CM	CG	DI	VH	EL	SA	EU	SI	AF
Real	99.2±0.1	74.9±0.1	76.9±1.8	82.7±1.7	93.9±0.0	97.6±0.1	99.1±0.0	89.1±0.0	96.3±0.5	52.5±1.1
CTGAN	99.0±0.1	72.8±0.7	72.6±0.4	80.2±0.7	92.4±0.0	95.5±0.1	99.0±0.0	88.1±0.0	99.8±0.5	53.3±1.1
TVAE	99.1±0.1	73.2±0.7	75.6±1.6	**83.0±0.7**	92.0±0.6	95.4±0.1	99.0±0.0	89.1±0.8	99.7±0.1	53.1±1.1
TabDDPM	99.1±0.2	73.3±1.6	74.7±1.2	82.2±1.2	92.5±0.0	95.5±0.1	**99.1±0.0**	89.1±0.5	99.7±0.1	53.0±0.5
GReaT	99.1±0.2	71.8±0.8	72.6±1.9	79.5±2.0	91.9±0.3	96.8±0.0	98.6±0.1	88.8±0.7	99.8±0.0	53.1±0.2
TapTap	**99.2±0.2**	**73.8±0.4**	75.7±1.4	80.9±1.0	93.2±0.6	97.1±0.2	**99.1±0.4**	**89.2±2.0**	99.8±2.4	53.1±0.8
AIGT	**99.2±1.1**	73.5±0.9	**79.5±2.6**	81.7±1.7	**93.6±0.4**	**97.9±0.2**	99.0±0.6	**89.2±2.0**	**99.9±0.2**	**53.8±0.5**
AIGT-L	**99.2±0.1**	74.2±0.6	76.6±0.4	82.3±0.1	93.0±0.4	97.1±0.1	99.0±0.6	88.6±0.4	**99.9±0.2**	**53.8±1.0**

Dataset	AD	WI	CR	SE	LO	HE	IN	AG	CA	KI
Real	92.7±1.3	99.1±0.0	100.0±0.0	99.3±0.0	99.8±0.0	80.9±0.1	88.0±0.0	53.2±0.3	85.3±0.2	87.8±0.1
CTGAN	92.4±0.1	97.5±0.6	96.4±1.1	99.3±0.1	98.9±0.4	75.4±0.6	59.3±6.0	49.8±0.6	81.4±0.5	82.3±1.1
TVAE	92.4±0.1	97.2±0.8	98.0±0.6	99.3±0.1	**99.8±0.5**	77.3±0.3	82.4±3.2	49.3±0.4	81.5±0.1	82.4±0.3
TabDDPM	92.3±0.1	**99.5±0.2**	99.8±0.1	99.4±0.0	**99.8±0.1**	80.2±0.3	86.5±0.6	53.0±0.5	84.1±0.2	82.4±1.0
GReaT	**92.7±0.1**	99.1±0.3	99.8±0.2	99.3±0.1	**99.8±0.1**	80.4±0.1	81.9±4.3	53.0±0.5	82.3±0.2	78.0±1.8
TapTap	91.9±0.2	98.9±0.7	99.9±0.6	**99.8±0.0**	**99.8±0.4**	81.0±0.4	88.1±0.2	53.0±0.6	**85.2±2.0**	87.6±0.5
AIGT	91.7±0.4	99.3±0.2	**100.0±0.0**	99.7±0.8	99.7±0.4	**81.1±0.1**	**88.2±0.1**	**54.2±0.2**	**85.2±0.1**	87.3±3.0
AIGT-L	**92.2±0.7**	99.3±0.4	**100.0±0.4**	99.3±0.4	**99.8±0.4**	79.2±0.2	88.1±0.8	**54.2±0.1**	**85.2±0.1**	**88.7±1.6**

Table 4: Data Augmentation. "Real" means training with the original data. Each generative methods means training with the original data plus the synthetic data.

Dataset	CG	SA	EU	SI	HE	KI	SYH	NonBD
CTGAN	52.4±4.0	95.4±0.2	53.1±3.5	65.8±0.8	40.6±2.0	55.3±1.8	45.7±0.8	78.6±0.6
TVAE	74.8±2.1	97.3±1.0	84.9±0.9	93.2±1.5	61.0±4.0	72.3±2.5	54.6±0.5	86.3±1.2
TabDDPM	71.0±1.2	78.8±4.2	69.5±0.7	96.5±0.7	74.6±0.5	75.0±3.0	56.1±0.7	79.6±1.8
AIGT	78.1±2.3	97.8±0.1	86.9±1.0	97.9±1.6	80.3±0.1	88.0±0.2	✗	✗
AIGT-part	74.3±1.1	97.8±0.2	83.9±0.8	96.4±0.4	78.6±0.1	86.6±0.8	57.2±1.6	88.2±0.2

Table 5: ML efficiency experiment for partition generation algorithm. The public datasets and the NonBD table were devided into two partitions, and partitioned the SYH table into five segments. Below the backbone model is LightGBM. ✗ indicates that the calculation cannot be performed due to the excessive columns in the table.

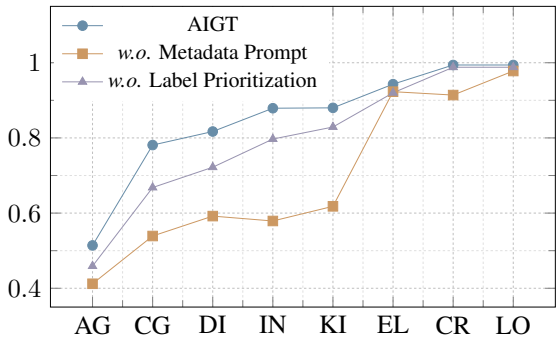

Figure 5: Ablation experiments related to training strategies. The y-axis is the average metric values across all datasets with LightGBM.

5.3 Ablation Analysis

Effects Analysis of Training Strategy. We selected 8 datasets and tested ablation experiments under the following different conditions: (1) ***w.o.* Metadata Prompt**. This refers to AIGT without the metadata from tables serving as fixed prompts. (2) ***w.o.* Label Prioritization**. We consider labels as features, shuffling them with other features for use in the training and generation processes, rather than moving the label column to the first position during the serialization process. The experimental results are shown in Figure 5, we can see that the prompt-enhanced method indeed has a good gain, demonstrating the effectiveness of our method.

Performance Analysis of Partition Algorithm. We evaluate the impact of the number of partitions in our partitioning algorithm on the synthesis of data. Four public datasets are selected for this purpose. The results are shown in Figure 6.

6 Conclusion

In this paper, we introduce a novel data-synthesis method for language models called AIGT, enhanced with prompts. This method utilizes the metadata of tables, guiding the language model to

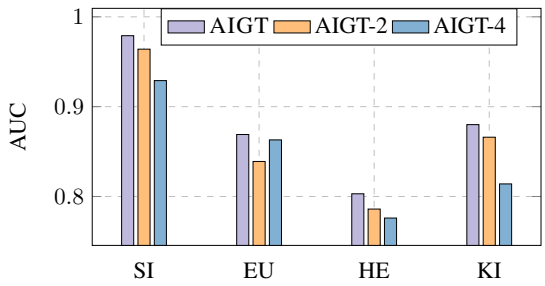

Figure 6: Analysis of the number of partitions for partition generation algorithm. "AIGT" signifies no partitioning, while "AIGT-2" denotes partitioning into two sections,"AIGT-4" denotes partitioning into four sections.

generate data more effectively.

Existing table generation methods based on language models are unable to tackle the issue of long tokens. To overcome this limitation, we've designed a long-token partitioning algorithm, can support the generation of tabular data at any scale. Our approach is more flexible, capable of handling tabular data with a larger number of feature columns.

Acknowledgments

This work is supported by the Fundamental Research Funds for the Central Universities (226-2024-00049), the NSFC Grants (No.62206247) and the Pioneer R&D Program of Zhejiang (No.2024C01035). The authors from Ant Group are supported by the Leading Innovative and Entrepreneur Team Introduction Program of Hangzhou (Grant No.TD2022005).

Limitations

A primary limitation of our method is its processing speed. While we leverage the powerful capabilities of LLMs, this also results in increased running time and GPU memory consumption compared to more lightweight approaches such as GANs. Detailed comparisons are provided in appendix C.3. Furthermore, our current method for handling numerical values treats numbers as characters, applying tokenization and transformation without additional processing. This approach is unable to capture the magnitude relationships of numerical values, focusing solely on semantic similarity during tokenization. Future work will aim to develop more advanced encoding techniques for numerical values to address this limitation.

Ethics Statement

In this paper, the STABS dataset we constructed is derived from publicly available datasets that have been openly shared on well-known machine learning platforms such as OpenML, ensuring that no private information is included. For the risk control data from Alipay that we used, since it is internal company data and has not been anonymized, we only present the algorithm's performance without disclosing the dataset to protect user privacy and comply with data security laws and regulations. Additionally, our use of the GPT-3.5 API is conducted through compliant intermediaries and utilized by the enterprise in accordance with regulatory requirements.

References

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, page 2623–2631, New York, NY, USA. Association for Computing Machinery.

Karim Armanious, Chenming Jiang, Marc Fischer, Thomas Küstner, Tobias Hepp, Konstantin Nikolaou, Sergios Gatidis, and Bin Yang. 2020. Medgan: Medical image translation using gans. *Computerized medical imaging and graphics*, 79:101684.

Laura Aviñó, Matteo Ruffini, and Ricard Gavaldà. 2018. Generating synthetic but plausible healthcare record datasets. *arXiv preprint arXiv:1807.01514*.

Vadim Borisov, Tobias Leemann, Kathrin Seßler, Johannes Haug, Martin Pawelczyk, and Gjergji Kasneci. 2022. Deep neural networks and tabular data: A survey. *IEEE Transactions on Neural Networks and Learning Systems*.

Vadim Borisov, Kathrin Seßler, Tobias Leemann, Martin Pawelczyk, and Gjergji Kasneci. 2023. Language models are realistic tabular data generators. *Preprint*, arXiv:2210.06280.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Zhengping Che, Yu Cheng, Shuangfei Zhai, Zhaonan Sun, and Yan Liu. 2017. Boosting deep learning risk prediction with generative adversarial networks for electronic health records. In *2017 IEEE International Conference on Data Mining (ICDM)*, pages 787–792. IEEE.

Edward Choi, Siddharth Biswal, Bradley Malin, Jon Duke, Walter F Stewart, and Jimeng Sun. 2017. Generating multi-label discrete patient records using generative adversarial networks. In *Machine learning for healthcare conference*, pages 286–305. PMLR.

Antonia Creswell, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil A Bharath. 2018. Generative adversarial networks: An overview. *IEEE signal processing magazine*, 35(1):53–65.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Cristóbal Esteban, Stephanie L Hyland, and Gunnar Rätsch. 2017. Real-valued (medical) time series generation with recurrent conditional gans. *arXiv preprint arXiv:1706.02633*.

Heng Gong, Yawei Sun, Xiaocheng Feng, Bing Qin, Wei Bi, Xiaojiang Liu, and Ting Liu. 2020. Tablegpt: Few-shot table-to-text generation with table structure reconstruction and content matching. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 1978–1988.

Yury Gorishniy, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. 2021. Revisiting deep learning models for tabular data. *Advances in Neural Information Processing Systems*, 34:18932–18943.

Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. 2020. How can we know what language models know? *Transactions of the Association for Computational Linguistics*, 8:423–438.

James Jordon, Jinsung Yoon, and Mihaela Van Der Schaar. 2018. Pate-gan: Generating synthetic data with differential privacy guarantees. In *International conference on learning representations*.

Jayoung Kim, Chaejeong Lee, Yehjin Shin, Sewon Park, Minjung Kim, Noseong Park, and Jihoon Cho. 2022. Sos: Score-based oversampling for tabular data. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 762–772.

Akim Kotelnikov, Dmitry Baranchuk, Ivan Rubachev, and Artem Babenko. 2023. Tabddpm: modelling tabular data with diffusion models. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. Albert: A lite bert for self-supervised learning of language representations. *Preprint*, arXiv:1909.11942.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Comput. Surv.*, 55(9).

Gautier Marti. 2020. Corrgan: Sampling realistic financial correlation matrices using generative adversarial networks. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8459–8463. IEEE.

Noseong Park, Mahmoud Mohammadi, Kshitij Gorde, Sushil Jajodia, Hongkyu Park, and Youngmin Kim. 2018. Data synthesis based on generative adversarial networks. *arXiv preprint arXiv:1806.03384*.

Neha Patki, Roy Wedge, and Kalyan Veeramachaneni. 2016. The synthetic data vault. In *2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pages 399–410. IEEE.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551.

Nan Tang, Ju Fan, Fangyi Li, Jianhong Tu, Xiaoyong Du, Guoliang Li, Sam Madden, and Mourad Ouzzani. 2020. Rpt: relational pre-trained transformer is almost all you need towards democratizing data preparation. *arXiv preprint arXiv:2012.02469*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. 2019a. Modeling tabular data using conditional gan. *Advances in neural information processing systems*, 32.

Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. 2019b. Modeling tabular data using conditional gan. *Advances in neural information processing systems*, 32.

Jun Zhang, Graham Cormode, Cecilia M Procopiuc, Divesh Srivastava, and Xiaokui Xiao. 2017. Privbayes: Private data release via bayesian networks. *ACM*

Transactions on Database Systems (TODS), 42(4):1–41.

Tianping Zhang, Shaowen Wang, Shuicheng Yan, Jian Li, and Qian Liu. 2023. Generative table pre-training empowers models for tabular prediction. *Preprint*, arXiv:2305.09696.

A Dataset

A.1 More Information about STABS

In this paper, we collected and filtered out 978 publicly available tabular datasets to construct the pre-training corpus for AIGT. Significant effort has been invested in conducting thorough data filtering and cleaning procedures to uphold dataset quality. For each table, our data cleaning protocols include, but are not limited to:
(1) *Check the semantic degree of the column names.* For example, the column names {*user_age*, weight, *monthly_income*} have high semantic information, while the column names {*f1*, *f2*, *xyz*} have low. We calculate a cumulative semantic relevance score for each table and, as part of our protocol, exclude tables which less than 50% of the semantic score.
(2) *Check the missing values.* The datasets with more than 40% missing values are discarded. Because too many missing values can easily lead to biased or inaccurate results in the pre-training phase.
(3) *Data Preprocessing.* For categorical features in the tables, we restore them to their original textual values whenever possible. As for numerical features, we employ normalization to mitigate the impact of inconsistent measurement units across different tables (e.g., kilograms vs. grams).

A.2 More Details about Downstream Dataset

We provide the urls of the public datasets in Table 6.

Abbr	Link
BW	https://www.openml.org/d/15
CG	https://www.openml.org/d/45058
DI	https://www.openml.org/d/42608
EL	https://www.openml.org/d/151
SI	https://www.openml.org/d/41946
AD	https://www.openml.org/d/1590
WI	https://www.openml.org/d/40983
CM	https://www.openml.org/d/45054
VH	https://www.openml.org/d/42863
SA	https://www.openml.org/d/42858
AF	https://www.openml.org/d/469
CR	https://www.openml.org/d/40975
SE	https://www.openml.org/d/42860
EU	https://www.openml.org/d/43925
LO	https://www.kaggle.com/datasets/burak3ergun/loan-data-set
HE	https://www.kaggle.com/datasets/averkiyoliabev/home-equity-line-of-creditheloc
CA	https://www.kaggle.com/datasets/camnugent/california-housing-prices
AG	https://www.kaggle.com/datasets/sidhus/crab-age-prediction
IN	https://www.kaggle.com/datasets/mirichoi0218/insurance
KI	https://www.kaggle.com/harlfoxem/housesalesprediction

Table 6: The urls of Downstream Tabular Datasets.

B Prompt Templates

In our implementation, we use the gpt-3.5-turbo model via the OpenAI-API to construct the function **prompt**. Here we presented the code to call the GPT API in Listing 1:

Our prompt input to gpt-3.5 is shown as Listing 2. We utilize the table's metadata **M** to construct prompt input. In our framework, we actively collect metadata from two sources:

- The caption information (**C**): the description of the dataset, including the purpose of the dataset, background information.
- Feature information(**F**): the name of columns and the meaning of columns.

C Further Experiments

C.1 Statistical similarity

To accurately assess the dependencies among columns in synthetic data, we calculate pair-wise correlation matrices separately for both real and synthetic datasets. We use the Pearson correlation coefficient to analyze continuous variables, which produces results within the range of [-1, +1]. For categorical features, we use the uncertainty coefficient for evaluation, yielding values within the range of [0, 1]. Additionally, we use the correlation ratio to investigate the relationship between categorical and continuous variables, which also yields values within [0, 1]. Subsequently, we compute the Frobenius norm between the pairwise correlation matrices of the real and synthetic datasets and refer to it as the Correlation Distance. It is worth noting that a lower Correlation Distance value signifies a higher quality of data synthesis. The mean Correlation distance of 20 datasets was presented in figure 7.

C.2 Discriminator Measure.

In order to verify whether the data we generated can be easily distinguished from the original data, we trained a LightGBM discriminator (with hyperparameter tuning) on a combination of the generated training set (with a label of 0) and the original training set (with a label of 1). Following this, we reported the test accuracy on a test data set, which comprises equal portions of samples from both the generated test set and the real test set. The scores, which are displayed in Table 7, demonstrate the superior performance of AIGT.

```
import requests
api_key = 'openai_api_key'
url = 'https://api.openai.com/v1/engines/davinci-codex/completions'
headers = {
    'Content-Type': 'application/json',
    'Authorization': f'Bearer {api_key}'
}
data = {
    'prompt': '',
    'max_tokens': 200
}
response = requests.post(url, headers=headers, json=data)
if response.status_code == 200:
    result = response.json()
    print(result['choices'][0]['text'].strip())
else:
    print(f"Error: {response.status_code}")
    print(response.json())
```

Listing 1: Code for calling GPT3.5

```
prompt=
"Following is a description of a dataset, a profile of an object from the dataset,
    and a target description. The objective is to predict the target based on the
    information provided about the object.\n
Dataset description: \{table caption\},\n
feature name: \{columns\}.\n
Target: \{target column\}.\n
the meaning of columns:\{the meaning of features\}
Information to be returned includes: 1) a brief summary of the table description (
    such as field and background); 2) the target columns; 3) the features and their
    explanations. Here is an example output: The dataset is about economics, the
    target is income, and the features along with their explanations are as follows:
     ID represents a unique identifier for each user; Age denotes the age of each
    user. Make it brief but informative. Try to limit it to 200 words."
```

Listing 2: prompt construction

Dataset	CTGAN	TVAE	TabDDPM	GReaT	TapTap	AIGT
BW	99.9±0.2	99.8±0.3	**57.9±2.8**	69.7±1.1	67.2±2.6	71.9±2.5
CM	78.6±2.3	89.9±0.9	55.7±1.4	95.5±0.4	95.7±0.5	**54.0±2.4**
CG	90.0±1.4	99.6±0.2	75.3±5.8	99.3±0.6	99.5±0.5	**75.0±4.5**
DI	97.7±1.2	98.9±0.8	88.6±2.3	76.1±2.7	**59.8±3.5**	73.5±2.7
VH	99.4±0.6	86.9±2.9	100.0±0.0	88.4±1.0	78.8±1.7	**76.4±3.0**
EL	99.7±0.0	99.5±0.1	83.1±0.4	**68.7±0.4**	100.0±0.0	69.9±0.3
SA	99.9±0.1	100.0±0.0	100.0±0.0	77.5±0.1	100.0±0.0	**78.7±0.8**
EU	99.8±0.4	99.9±0.2	100.0±0.0	100.0±0.0	100.0±0.0	**95.6±0.2**
SI	98.7±0.4	99.3±0.3	91.9±0.9	73.8±0.6	**69.1±0.1**	72.8±0.7
AF	67.9±3.0	86.0±2.2	53.0±1.5	71.4±3.1	68.2±4.0	**51.4±2.5**
AD	100.0±0.0	100.0±0.0	**71.5±0.4**	99.9±0.0	99.9±0.1	99.4±0.0
WI	84.1±1.0	71.1±1.4	**51.8±0.2**	70.0±0.4	61.3±0.7	69.8±0.1
CR	69.2±1.7	59.7±1.5	53.6±1.8	50.9±0.2	52.6±0.5	**50.7±0.2**
SE	100.0±0.1	100.0±0.1	91.9±0.6	81.3±0.2	96.0±0.1	**72.2±0.1**
IN	89.7±1.3	80.6±1.1	67.9±0.8	82.6±0.2	80.3±0.3	**67.8±0.3**
HE	98.5±0.2	96.1±0.2	85.7±0.6	96.9±0.4	96.6±0.4	**78.8±0.6**
CR	99.3±0.2	98.6±0.4	87.8±0.9	99.2±0.3	99.2±0.4	**59.2±0.7**
CA	91.5±0.3	88.7±0.4	**65.0±0.1**	76.5±0.6	77.5±0.6	75.4±0.6
LO	98.9±0.3	98.2±0.4	71.9±0.5	92.2±0.7	90.4±0.7	**71.3±0.9**
KI	100.0±0.0	99.9±0.1	95.9±0.2	81.9±0.7	**79.9±0.6**	81.3±0.7

Table 7: Discriminator measure. A lower accuracy rate suggests a difficulty for the discriminator in discerning between artificial records and original samples. A completely indistinguishable dataset would yield an accuracy rate of 0.5. The superior results are highlighted in bold, while the results of the next best are underscored. Best results are bold, second-best results are underlined. Results are averages over ten trials with different random seeds.

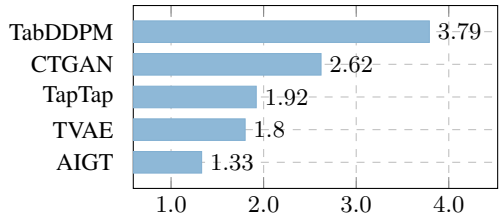

Figure 7: Correlation distance mean value for 20 datasets.

	Training Time	Sampling Time
CTGAN	873	9
TVAE	360	3
TabDDPM	856	158
GReaT	960	895
Taptap	910	506
AIGT	906	456

Table 8: The running time in seconds on the Adult Income dataset of different methods in the privacy protection setting. The number of fine-tuning steps for GReaT, TapTap and AIGT was 10k. A total of 36k samples were generated.

C.3 Running Time

We analyze the running time of AIGT and baseline methods. The results of the Adult Income dataset are in Table 8. As can be seen from the results, the generation method based on language models requires more computational resources and time.

D Hyperparameters Optimization

We employ Optuna (Akiba et al., 2019) for hyperparameter tuning of LightGBM. For each specific dataset and model, we initially tune the model's hyperparameters using the original data. The determined set of hyperparameters is then consistently applied across all experiments on the dataset for all methods, ensuring a fair comparison.

Models	Parameter	Values
LightGBM	learning_rate	Uniform[0.01, 0.1]
	num_leaves	Uniform[10, 100]
	subsample	Uniform[0.5, 1.0]
	colsample_bytree	Uniform[0.5, 1.0]
	min_child_samples	UniformInt[2, 100]
	#Iterations	UniformInt[100, 1000]

Table 9: Hyperparameter Space of LightGBM.