Universal Tabular Data Generator with Large Language Models

Stanford CS224N Custom Project

Julian Chu
Department of Computer Science
Stanford University
juliantc@stanford.edu

Abstract

The generation of synthetic data has become increasingly important due to privacy concerns, data insufficiency, and the need for more diverse datasets for training machine learning models. This study demonstrates the capability of large language models to generate realistic tabular data and even exceed the performance of state-of-the-art methods such as CTGAN. Additionally, large language models trained on multiple tabular datasets can generate synthetic data not only on datasets it had been trained on but also on those it had not been trained on. Future work is recommended to study the effects of increasing the training data size and model size, This study suggests the feasibility of creating a general synthetic data generation model that can generate any synthetic data on command.

1 Key Information to include

- Mentor: None
- External Collaborators (if you have any): None
- Sharing project: None

2 Introduction

The generation of synthetic data has become increasingly important in recent years due to a variety of reasons, including privacy concerns, data insufficiency, and the need for more diverse datasets for training machine learning models. Synthetic data generation can help to mitigate these issues by providing additional data that can be used to train machine learning models without compromising privacy or other sensitive information.

One important area where synthetic data generation is particularly relevant is in the domain of tabular data. Tabular data is widely used in various industries such as finance, business, and healthcare, and it often contains sensitive information about individuals or organizations [1]. However, generating synthetic tabular data that accurately captures the statistical properties of real data is a challenging task.

While there has been a lot of research on synthetic data generation, existing methods still have some limitations. First, existing methods require some real data to start with and need to be retrained for each new dataset, which can be time-consuming and costly. Furthermore, most existing methods fail to incorporate information across different datasets, which could be useful information for generating synthetic data (e.g. incorporating information about Florida housing prices and the cost of living in Florida and California to generate synthetic data on California housing prices).

The goal of this work is to harness the power of large language models (LLMs) to develop a novel approach for generating synthetic tabular data. Specifically, we aim to build a model that can generate synthetic tabular data from a natural language dataset description and column metadata (column name), and is able to generate synthetic data on datasets it has never seen before

3 Related Work

3.1 Synthetic Data

Synthetic data generation has been a topic of research for several years. The goal is to generate realistic data that can be used to train machine learning models without revealing sensitive information. One common technique is to use generative models such as Generative Adversarial Networks. Generative Adversarial Networks (GANs) [2] are a class of deep learning models that have gained significant attention in recent years for their ability to generate synthetic data that is almost indistinguishable from real data. GANs were first introduced by Goodfellow et al. in 2014, and since then, they have been applied in various fields such as computer vision, natural language processing, and robotics[3][4].

The fundamental idea behind GANs is to train two neural networks simultaneously: a generator network and a discriminator network. The generator network takes random noise as input and generates synthetic data that is intended to mimic the real data. The discriminator network, on the other hand, takes as input both real data and the synthetic data generated by the generator and tries to distinguish between them.

3.2 Tabular Synthetic Data

Tabular data is a widely used format in various domains such as finance, business, and healthcare. Generating synthetic tabular data has been the focus of recent research. Classical methods for modeling tabular data include Bayesian networks and statistical tools such as copulas. Conditional Tabular Generative Adversarial Networks (CTGAN) [5] is the state of the art method for generating synthetic tabular data using GANs. The method was introduced by Xu et al. in 2019 and has shown promising results in generating realistic tabular data that preserves the statistical characteristics of the original data. CTGAN utilizes a modified GAN architecture that incorporates conditional generative modeling, feature normalization, and gradient penalty techniques to address the challenges of generating high-dimensional and complex tabular data.

3.3 Large Language Model

Large language models are, by definition, a very large model, typically a transformer [6], that has been trained on vast amounts of textual data to predict the next work given the previous words. These model are shown to be able to perform a wide range of language-related tasks, including language translation, question-answering, and text generation [7]. Large language models have grown in popularity in recent years, with the development of models such as GPT-3, which contains over 175 billion parameters, making it one of the largest language models to date. Some large language models are also demonstrated to be able to perform zero-shot, one-shot, and few shot learning [8]. There are two main ways to teach a large language model to perform specific task: few shot learning through prompting and finetuning.

3.4 Transfer learning

Transfer learning [9] is a technique in machine learning where knowledge gained from one task is leveraged to improve performance on a different but related task. In other words, a pre-trained model that has learned useful patterns and features from a large dataset in one domain can be re-used and adapted to a new task in a different but related domain, without the need for retraining

the model from scratch. Transfer learning has been widely used in various domains such as natural language processing, computer vision, and speech recognition.

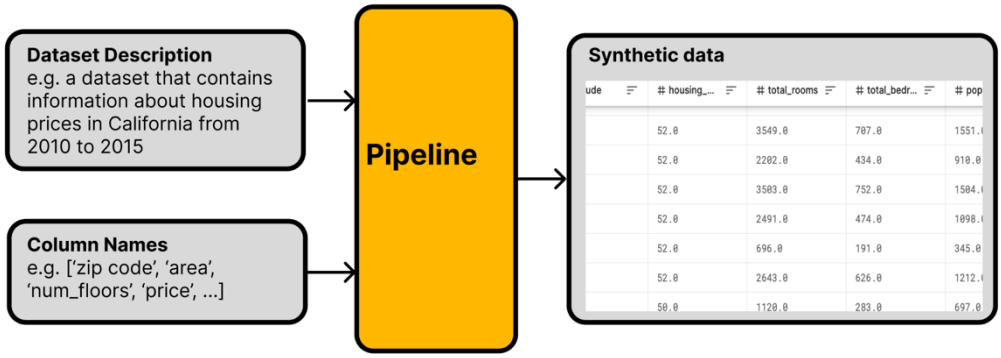
3.5 Synthetic tabular data with LLM

There is only handful of work trying to use large language models to generate synthetic tabular data. One notable work is GReaT (Generation of Realistic Tabular data) [10], which exploits an auto-regressive generative LLM to sample synthetic and highly realistic tabular data. They accomplish this by translating tabular data into textual data (for example from 94305, 100m^2, $10M to "zip code is 94305, area is 100m^2, price is $10M), with additional techniques like Random feature order permutation to enhance performance, then finetuning the model with the data. In their paper, the researchers demonstrated that GReaT can generate realistic synthetic tabular data. Their model achieved results surpassing that of CTGAN. However, they didn't try to use LLM to generate unseen datasets, i.e. datasets that it had not been explicitly trained on before. The paper is still under review.

4 Approach

4.1 Pipeline

We want to build a pipeline such that given dataset description, column names, and number of rows as inputs, it will output a realistic dataset with the respective columns and the required number or rows.



4.1.1 training

We first have to convert the problem into a prompt completion problem that we can train the large language model to learn. The prompt consist of the dataset description, column names, and previous row values (except for the first few rows, which doesn't have previous row values). Examples of the data are in the appendix.

We perform the transformation specified above to multiple tabular datasets, then finetune the model with the prompt completion dataset we created. We theorize that training the model on more than one tabular dataset will further increase the performance on both a seen dataset (a dataset that is included in its training data) and an unseen dataset (a dataset the model is not trained on). We will test this by also training a model that is trained only data created from one tabular dataset.

4.1.3 Recurrent generation

At the generation stage, we recurrently call the model we finetuned and extract the new rows generated until we reached the required number of rows.

4.3 Models

The baseline model we are using is Condition Tabular Generative Adversarial Network (CTGAN), which is the state-of-the-art synthetic tabular data generation model. It learns the joint distribution of tabular data and generate new synthetic samples. The key idea behind CTGAN is to condition the GAN on the values of some of the variables in the dataset, thereby allowing it to generate new samples that respect the conditional dependencies in the original dataset. The mathematical formula for CTGAN involves the objective function that the GAN aims to optimize during training, which includes a discriminator and a generator network. The discriminator network aims to distinguish between real and synthetic data, while the generator network aims to generate synthetic samples that fool the discriminator. The objective function for CTGAN is given by:

$$min_G max_D E_{x \sim P_{data}}[\log(D(x))] + E_{z \sim P_z}[log(1 - D(G(z)))]$$

where G and D denote the generator and discriminator networks, respectively, and P_data and P_z are the distributions of real and synthetic data, respectively. The objective function is iteratively optimized until the generator produces samples that are indistinguishable from real data by the discriminator.

For the large language models we are using to test our approach, we are finetuning the GPT-3 babbage model. Even though we would expected far better result using better models such as the davinci mode, the babbage model costs 50 times less and is more feasible for the scope of this project. The babbage model contains only 1.3 Billion parameters, (while davinci has 175 Billion parameters). The model has a stacked transformer architecture.

5 Experiments

5.1 Data

We used 4 datasets in total. The datasets consist of tabular data on bank customer churn, telecom customer churn, housing prices, and travel customer churn (see appendix). The travel customer churn data is used as an unseen dataset to see if the model can generalize its learnings and generate tabular data with descriptions and column names it had never seen before. The tabular data are transformed according to the method described in the approach section before being used to train on the model

5.2 Evaluation method

5.2.1 Kolmogorov–Smirnov test

We used the Kolmogorov–Smirnov test to measure the quality of the synthetic data on continuous columns. The Kolmogorov-Smirnov test is a statistical hypothesis test that is used to determine whether two sets of data come from the same probability distribution or not. The test is based on the maximum difference between the empirical cumulative distribution functions (ECDFs) of the two datasets. The ECDF of a dataset is a step function that gives the proportion of data points that are less than or equal to a given value. The test statistic, denoted by D, is the maximum absolute difference between the two ECDFs:

$$D = \max|F\_1(x) - F\_2(x)|$$

where F_1 and F_2 are the ECDFs of the two datasets, and x is the value at which the difference is evaluated. The null hypothesis of the test is that the two datasets come from the same distribution, and the alternative hypothesis is that they come from different distributions. The p-value of the test is computed as the probability of observing a test statistic as extreme as D under the null hypothesis. If the p-value is smaller than a chosen significance level, typically 0.05 or 0.01, then the null hypothesis is rejected in favor of the alternative hypothesis.

5.2.2 Total Variation Distance

We calculated the Total Variation Distance to determine quality of the synthetic data on categorical columns. This test computes the Total Variation Distance (TVD) between the real and synthetic columns. To do this, it first computes the frequency of each category value and expresses it as a probability. The TVD statistic compares the differences in probabilities, as shown in the formula below:

$$\delta(R, S) = \frac{1}{2} \sum_{\omega \in \Omega} |R_\omega - S_\omega|$$

Here, ω describes all the possible categories in a column, Ω. Meanwhile, R and S refer to the real and synthetic frequencies for those categories. The TVComplement returns 1-TVD so that a higher score means higher quality.

$$score = 1 - \delta(R, S)$$

5.2.4 Similarity of correlation between columns

We also compare similarity of the correlation between different columns of the synthetic data to that of the real data. This is important because realistic synthetic data doesn't just capture the distribution of each individual columns, but the joint distribution of all the columns. For example, a datapoint describing a person with an age of 9 and a salary of $100k would be unrealistic, even though both values is realistic individually (it is realistic that someone has an age of 9 and that another person has a salary of $100k, but it is unrealistic for that to be the same person). Therefore, we calculate the similarity of the correlation between columns.

The formula for calculating the correlation coefficient is:

r = (nΣxy - ΣxΣy) / sqrt([nΣx² - (Σx)²][nΣy² - (Σy)²])

Where x and y are the two variables, Σ is the sum of, and n is the number of observations. The numerator of the formula represents the covariance between the two variables, while the denominator represents the product of their standard deviations.

Using For a pair of columns, A and B, this test computes a correlation coefficient on the real and synthetic data, R and S. This yields two separate correlation values. The test normalizes and returns a similarity score using the formula below.

$$score = 1 - \frac{|S_{A,B} - R_{A,B}|}{2}$$

5.2.5 Overall Quality Score

We boil down the KS score and TV score into one metric by taking the average across the scores in all columns. We call this column shape similarity. We also take the average of the correlation of all the numeric column pairs as the Correlation Similarity score.

5.4 Results

After training all three models, we tasked each model with generating 1000 rows of data for both the tabular datasets that it had been trained on with tabular data that it had never been trained on before.

	Column Shape Similarity		Correlation similarity	
	seen	unseen	seen	unseen
CTGAN	0.8488	0 (NA)	0.8095	0 (NA)
Single dataset LLM	**0.8630**	0 (failed)	**0.8293**	0 (failed)
Multi datasets LLM	0.8601	**0.2830**	0.8248	**0.2233**

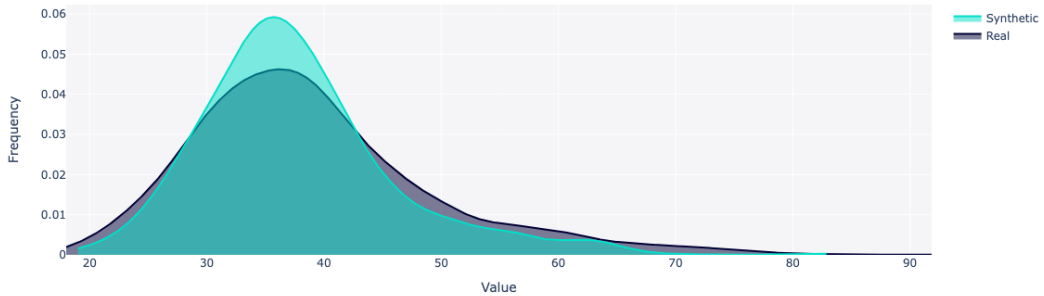
* NA: model is not capable of doing so by definition
* failed: model failed to generate anything

The results shows that for seen data, both of our models perform significantly better than CTGAN. The model trained on multiple datasets seems to be performing slightly worse on seen data. This can potentially be due to the fact that the same model had been trained to generate many datasets and as a result sacrificed performance on generating each individual dataset.

CTGAN is by definition not capable of generating unseen data, as it can only generate data if we first train it on some data. The LLM we trained on a single dataset only is also not able to generate unseen data. It simply generated nothing. The LLM trained on multiple datasets (3 datasets) is the only model capable of generating unseen data. Even though the performance is not high, it is performing significantly better than random, showing that the overall approach of generalizing synthetic data generation from multiple datasets is working.

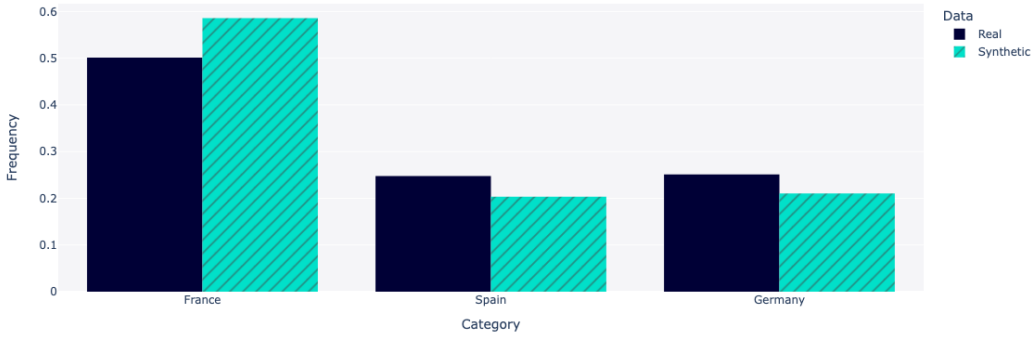
6 Analysis

Real vs synthetic data for column Age – multi-dataset LLM

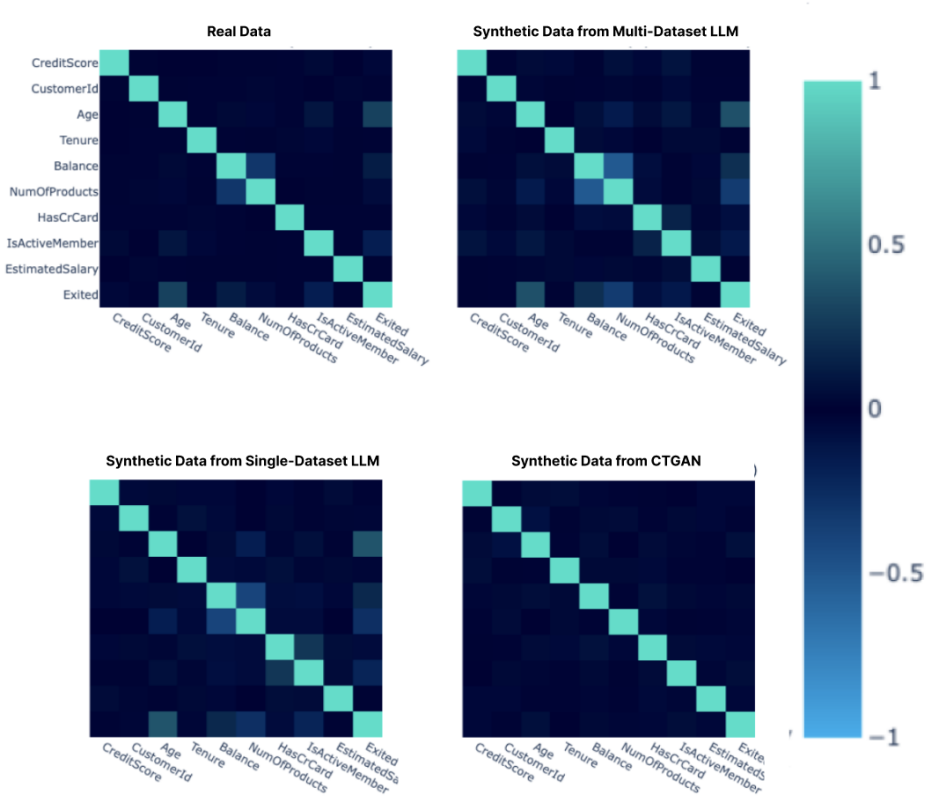


Above is a figure plotting the distribution of the continuous numeric column ‘Age’ from synthetic data generated by the multi-dataset model and the distribution of that column in the real data. We can see that the model has learned the shape of the underlying distribution for numeric data while generating numeric data.

Real vs synthetic data for column ‘Geography’ – single-dataset LLM



Above is a figure plotting the distribution of the discrete column ‘Geography’ in synthetic data generated by single-dataset LLM and the distribution of that in the real data. It shows that the real data is able to learn the distribution of the discrete data as well.



From the correlation matrices above, we can see that the correlation matrices of the synthetic data from the two LLM models look similar to the real data. On the other hand, the correlation between columns in CTGAN’s synthetic data is mostly 0 and doesn’t resemble the real data at all.

7 Conclusion

We showed that large language models are capable of generating realistic tabular data and even exceed the performance of state-of-the-art methods like CTGAN. Moreover, large language models trained on generating multiple tabular datasets are capable of generating synthetic data not only on datasets it had been trained on, but also datasets it had not been trained on.

Further work should be done to study whether the performance of generating unseen data will increase if we increase the training data size by increasing the number of datasets it is trained on, and by increasing the model size, such as by using models like GPT-3 davinci. Further work should also be done on improving the current pipeline. For example, we can test whether including column descriptions and column metadata would improve the performance of the model, since the column names are sometimes not self-explanatory. We can also try the same approach on other modalities, such as using diffusion models as universal image data generation models

Overall, This work proved the feasibility of creating a general synthetic data generation model that is capable of generating any synthetic data on command.

References

1. Samuel A. Assefa, Danial Dervovic, Mahmoud Mahfouz, Robert E. Tillman, Prashant Reddy, and Manuela Veloso. 2021. Generating synthetic data in finance: opportunities, challenges and pitfalls. In Proceedings of the First ACM International Conference on AI in Finance (ICAIF '20). Association for Computing Machinery, New York, NY, USA, Article 44, 1–8. https://doi.org/10.1145/3383455.3422554

2. Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio. 2014.
   Generative Adversarial Networks

3. Y. Lin, C. Tang, F. -J. Chu and P. A. Vela, "Using Synthetic Data and Deep Networks to Recognize Primitive Shapes for Object Grasping," 2020 IEEE International Conference on Robotics and Automation (ICRA), Paris, France, 2020, pp. 10494-10501, doi: 10.1109/ICRA40945.2020.9197256.

4. Swami Sankaranarayanan, Yogesh Balaji, Arpit Jain, Ser Nam Lim, Rama Chellappa; Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 3752-3761

5. Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, Kalyan Veeramachaneni. 2019.
   Modeling Tabular Data using Conditional GAN

6. Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin. 2017.
   Attention Is All You Need

7. 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. J. Mach. Learn. Res. 21, 1, Article 140 (January 2020), 67 pages.

8. 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, Dario Amodei. 2019.
   Language Models are Few-Shot Learners

9. Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, Qing He. 2019.
   A Comprehensive Survey on Transfer Learning

10. Vadim Borisov, Kathrin Seßler, Tobias Leemann, Martin Pawelczyk, Gjergji Kasneci. 2022.
    Language Models are Realistic Tabular Data Generators

A Appendix (optional)

A.1. Examples of training data:

Example 1: row 1-3:

prompt :

A dataset which contain some customers who are withdrawing their account from the bank due to some loss and other issues with the help this data we try to analyse and maintain accuracy. The following data is in a csv format. The columns of this dataset includes
RowNumber,CustomerId,Surname,CreditScore,Geography,Gender,Age,Tenure,Balance,NumOfProducts,HasCrCard,IsActiveMember,EstimatedSalary,Exited

---
RowNumber,CustomerId,Surname,CreditScore,Geography,Gender,Age,Tenure,Balance,NumOfProducts,HasCrCard,IsActiveMember,EstimatedSalary,Exited

completion :

1,15634602,Hargrave,619,France,Female,42,2,0,1,1,1,101348.88,1
2,15647311,Hill,608,Spain,Female,41,1,83807.86,1,0,1,112542.58,0
3,15619304,Onio,502,France,Female,42,8,159660.8,3,1,0,113931.57,1

Example 2: row 4 onwards:

prompt :

A dataset which contain some customers who are withdrawing their account from the bank due to some loss and other issues with the help this data we try to analyse and maintain accuracy. The following data is in a csv format. The columns of this dataset includes
RowNumber,CustomerId,Surname,CreditScore,Geography,Gender,Age,Tenure,Balance,NumOfProducts,HasCrCard,IsActiveMember,EstimatedSalary,Exited

---
RowNumber,CustomerId,Surname,CreditScore,Geography,Gender,Age,Tenure,Balance,NumOfProducts,HasCrCard,IsActiveMember,EstimatedSalary,Exited

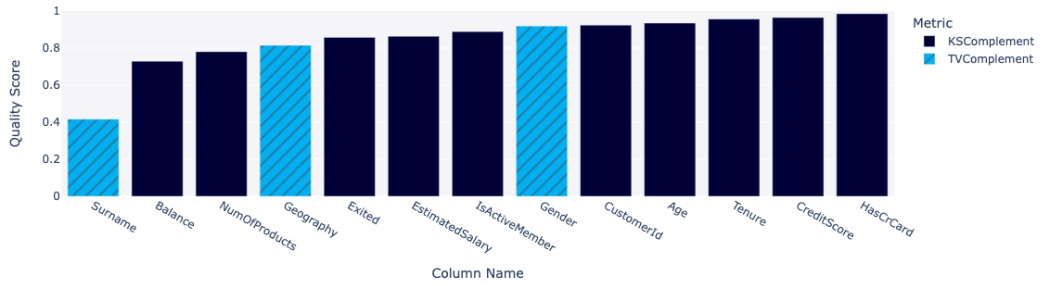
1,15634602,Hargrave,619,France,Female,42,2,0,1,1,1,101348.88,1
2,15647311,Hill,608,Spain,Female,41,1,83807.86,1,0,1,112542.58,0
3,15619304,Onio,502,France,Female,42,8,159660.8,3,1,0,113931.57,1

completion :

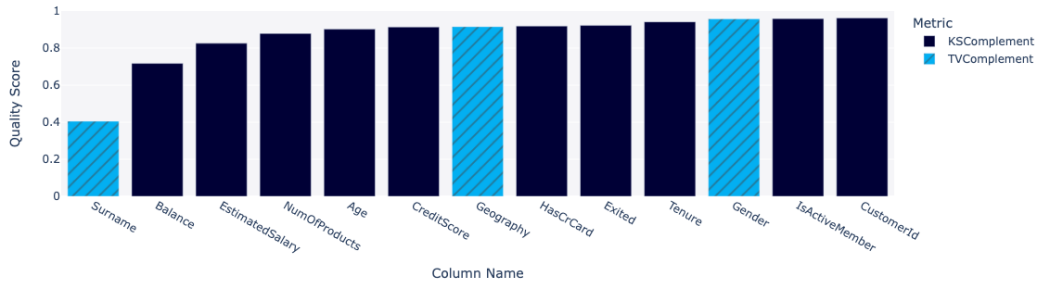
4,15701354,Boni,699,France,Female,39,1,0,2,0,0,93826.63,0
5,15737888,Mitchell,850,Spain,Female,43,2,125510.82,1,1,1,79084.1,0
6,15574012,Chu,645,Spain,Male,44,8,113755.78,2,1,0,149756.71,1

A.2. KS score and TV scores of different models

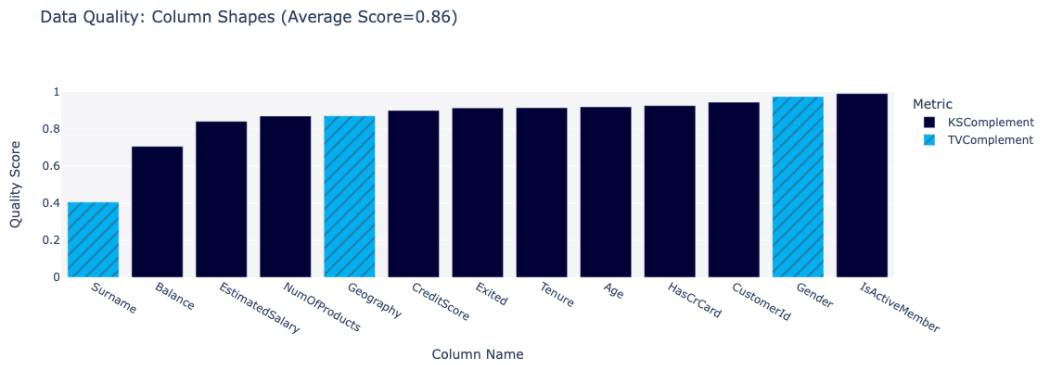
CTGAN



Single-dataset LLM



Multi-dataset LLM



A.3 Dataset Sources

Bank Churn Dataset: https://www.kaggle.com/datasets/santoshd3/bank-customers

Telecom Churn Dataset: https://www.kaggle.com/datasets/blastchar/telco-customer-churn
Travel Churn Dataset:
https://www.kaggle.com/datasets/tejashvi14/tour-travels-customer-churn-prediction
Housing Price Datset: https://www.kaggle.com/datasets/yasserh/housing-prices-dataset