Synthetic Data Generation: Enhancing Digital Marketing Predictions Using Flowmatching Models

Milla Nielsen, Kevin Shiah, Daniel Jenkins, Callen MacPhee

March 13, 2025

Contents

Abstract

This study addresses the challenge of click-through rate (CTR) prediction in the digital marketing space. Through the use of predictive and generative A.I. models, we analyze a database of user behavior to optimize ad performance. Knowing the limitations of privacy concerns and biases, we wish to transform our real-world data into high-quality, secure synthetic data. Synthetic data has shown to be useful for obscuring real, sensitive information from users that would otherwise be at risk of exposure and would violate data protection regulations. Through a flow-based generative approach, we can produce a dataset with robust statistical properties and safeguarded information. Training our model on this should offer improved utility while preserving data privacy, balancing performance and integrity.

1 Introduction

1.1 CTR Prediction

When it comes to digital advertising, user data is collected in order to observe patterns and trends in user behavior. Marketers use this data to tailor their advertisements to what appears to work best in garnering a response from the consumers they are reaching out to. This response is measured by Click-Through Rate, the ratio of users who click on an advertisement to the total number of users who see the ad. Marketing specialists wish to maximize CTR, as it is directly correlated with their possible sales. Increasing ad traffic space is a direct way for businesses to promote their product, and in doing so they will see more successful transactions and higher revenue. That's why predicting the accuracy of CTR is so important, as it indicates what types of ads, the demographic the ads are targeted towards, and the strategies used for implementing the ads work best. This includes information such as the city the consumer lives in, the browser type they use, and the creative ID of the advertisement. So the goal of these statistical models is to find out which factors in the observed user behavior collected data have the greatest impact in determining CTR prediction accuracy.

1.2 Data Scarcity and Concerns

Handling real-world data can be particularly challenging when it comes to the amount and security of the data. It is often difficult to gather a proper quantity of data that is high in quality. This is because data may not be easily available, not publicly accessible, or comes at too great of a cost. With a lack of sufficient data, it hinders the effectiveness of training and validating models, not being able to clearly identify relationships based on limited observations. And even if there is plenty of data, there comes the more pressing concern of ensuring the data's privacy. When manipulating this kind of data, it is imperative to guarantee the user's privacy so that their information is not publicly leaked. This information can be harmful in the wrong hands, as it can be used maliciously to actively harm the users. With CTR relying on behavior logs and user demographic information, these issues become even more pronounced. If large-scale datasets are unavailable, training models for CTR prediction becomes challenging, as the models may not generalize the data well or may miss important interactions between predictors. And if users feel as if their personal info is being mismanaged and distributed without their consent, they will likely be unwilling to use these services. This would further reduce the number of observations for analysis.

1.3 Synthetic Data's Role

Synthetic data is able to address these issues by artificially generating large amounts of high-quality data that is capable of mimicking the statistical properties of the real dataset. By being trained on even a small sample of real data, synthetic data can be replicated thousands of times over so models can be accurately trained on it. Additionally, synthetic data offers a solution for privacy concerns as the information is altered to create data that doesn't contain any real information. This new data may be similar in the way that the real

data trends, but it removes the direct correlation of the behavior to the user. This makes synthetic data both scalable and secure, whilst saving those who train models on it time and resources.

1.4 Problem Statement

This situation proposes the need to acquire synthetic data that our statistical models can properly train on. Not only does this data need to be high-quality, it also needs to maintain a balance of utility and privacy. If the synthetic data is only concerned about producing a highly accurate relationship between the predictors and the response, it may be fine-tuning the data too much, and could be sacrificing some of the privacy. Higher values of utility are going to mean that the synthetic data matches well with the real data, but it becomes concerning when that match is too correlated. The same can be said if the privacy levels are too high, as although it may produce extremely secure, anonymized data, it could drop in quality for the metric of utility, as the predictors and response may become unassociated. So the goal is to find a way to take the real data set and synthesize it so that it maintains, and potentially improves upon, the utility of the data, whilst introducing a significantly impactful level of privacy.

1.5 Contributions

A solution to this trouble is the use of flow-based models. Flow-based models excel at comprehending the intricacies and complexities of real-world data, especially on the multifaceted dataset of CTR predictions. What makes flow-based models so desirable is their ability to capture the fine details of user behavior patterns and ad interactions, maintaining a high level of fidelity for our data. This is useful for when it comes to training the model, as the synthetic data will contain the necessary statistical properties needed for analysis. It also has the tools necessary for privacy-preserving data generation, using differential privacy and adversarial training. Differential privacy ensures that through an analysis of the synthetic data, noise is added so as to conceal personalized information. And adversarial training combines training a dataset on a model with detecting privacy risks and synthetic to real data similarities. That means this synthesized data can improve in utility, while maintaining its fidelity, and overcoming the issue of a lack of privacy. This can be checked through a series of feature similarities and predictive performance comparisons that will show how well synthetic data can act as the replacement for real data.

2 Related Works

2.1 CTR Prediction Models

To evaluate the performance of our synthetic data and flow based model, we need a reference point to make comparisons to. The first way of doing this is through traditional predictive modeling approaches like Logistic Regression and Random Foresting. Logistic Regression is a simple and interpretable baseline model that aids in finding the important predictors for feature selection. Random Foresting uses a bagging ensemble approach which offers more robust performance and feature importance rankings by reducing the impurity of the data. Another method is to use deep learning approaches, like Multilayer Perceptron (MLP) and a Transformer Architecture. MLP is a fully connected neural network which allows for nonlinear relationships between features, and can reveal the complex dependencies of the variables. A transformer is a refined model capable of capturing contextual dependencies found within the data, enhancing the MLP. However, the best method we decided to use is the machine learning algorithm of XGBoost. This scalable approach is based on gradient-boosted decision trees, applicable for both classification and regression tasks. With its high computational speeds, superior predictive performance, and ability to reduce overfitting through regularization, we decided this would be the best model to compare to our flow matching one.

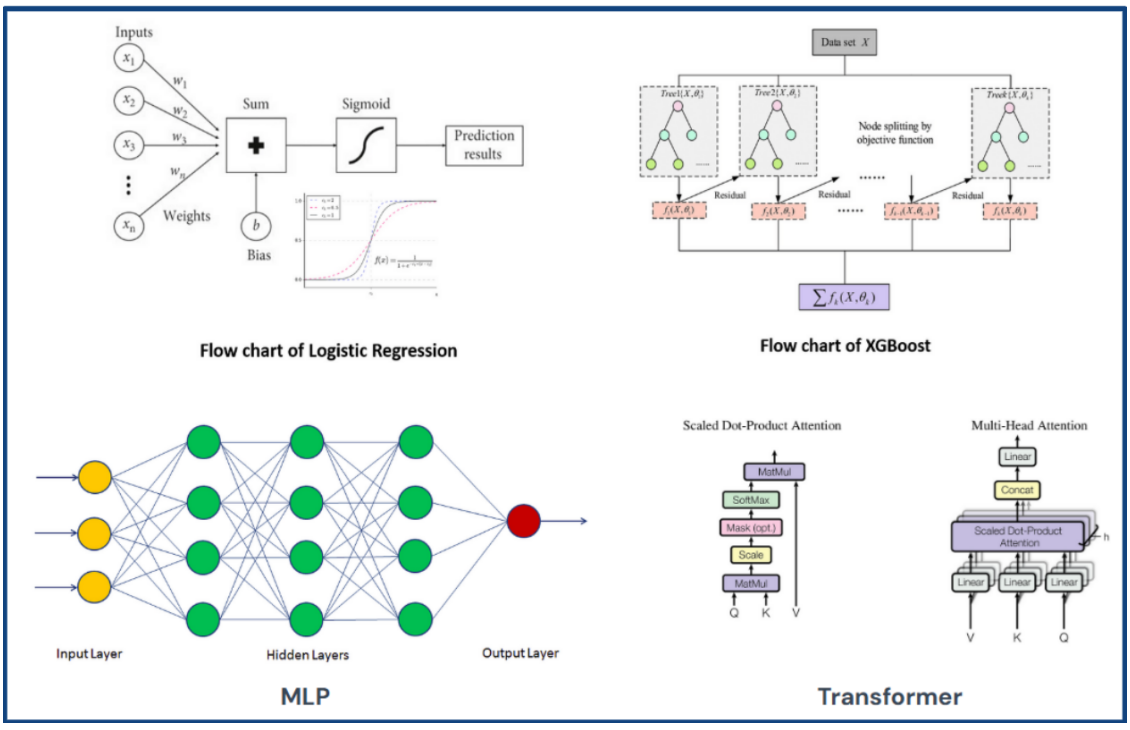

Figure 1: Basic Prediction Models previously used for CTR predictions

2.2 Theoretical Foundation

2.2.1 Flow-based Generative Models

Flow-based generative models rest on the principle of transforming a simple base distribution into a complex target distribution through a series of invertible mappings [6]. Let $\mathbf{x} \in \mathbb{R}^d$ be a random variable with unknown target density $p(\mathbf{x})$, and $\mathbf{z} \in \mathbb{R}^d$ be a random variable with a simple base density $\pi(\mathbf{z})$, typically a standard normal distribution. A normalizing flow defines an invertible transformation $f : \mathbb{R}^d \to \mathbb{R}^d$ such that $\mathbf{x} = f(\mathbf{z})$. The change of variables formula provides the mathematical foundation for computing probability densities under such transformations. For a given point $\mathbf{x}$, the density $p(\mathbf{x})$ relates to $\pi(\mathbf{z})$ through:

$$p(\mathbf{x}) = \pi\left(f^{-1}(\mathbf{x})\right) \left|\det \nabla f^{-1}(\mathbf{x})\right|$$

where $\nabla f^{-1}(\mathbf{x})$ is the Jacobian matrix of the inverse transformation. The log-likelihood of a data point $\mathbf{x}$ can thus be expressed as:

$$\log p(\mathbf{x}) = \log \pi(f^{-1}(\mathbf{x})) + \log \left|\det \nabla f^{-1}(\mathbf{x})\right|$$

This exact likelihood computation distinguishes flow-based models from other deep generative approaches. Variational Autoencoders (VAEs) maximize a lower bound on the log-likelihood through variational inference [4], while Generative Adversarial Networks (GANs) forgo explicit density estimation entirely in favor of implicit sampling through adversarial training [3].

The practical implementation of normalizing flows requires careful design of the transformation f to ensure both invertibility and tractable Jacobian determinant computation. Common architectures include coupling layers, autoregressive flows, and residual flows, each making different trade-offs between expressiveness and computational efficiency [6].

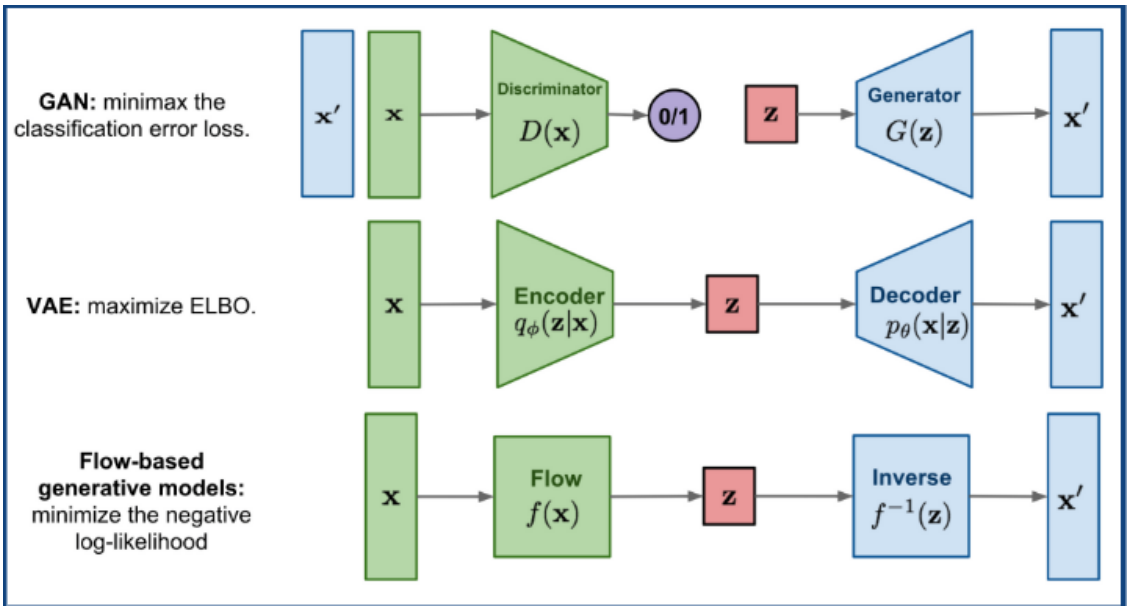

Figure 2: Comparison of different generative AI models

2.2.2 Continuous Normalizing Flows (CNF)

Continuous normalizing flows extend the discrete framework by defining transformations through continuous-time dynamics using Neural Ordinary Differential Equations (Neural ODEs) [8]. Instead of specifying a finite sequence of transformations, CNFs define a vector field that continuously transforms the probability density. Let $\mathbf{z}(t) \in \mathbb{R}^d$ represent the state at time $t \in [0, T]$, where $\mathbf{z}(0)$ follows the base distribution and $\mathbf{z}(T)$ represents the target sample. The evolution of $\mathbf{z}(t)$ is governed by an ODE:

$$\frac{d\mathbf{z}(t)}{dt} = v(\mathbf{z}(t), t; \theta)$$

where $v : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$ is a neural network parameterized by θ that defines the instantaneous change. The transformation from base to target distribution occurs by integrating this ODE from $t = 0$ to T. The corresponding evolution of the probability density follows the instantaneous change of variables formula, expressed through the continuity equation:

$$\frac{\partial p(\mathbf{z}, t)}{\partial t} = -\nabla \cdot (p(\mathbf{z}, t) v(\mathbf{z}(t), t; \theta))$$

where $\nabla\cdot$ denotes divergence. This yields an expression for the log-likelihood:

$$\log p(\mathbf{z}(T)) = \log p(\mathbf{z}(0)) - \int_0^T \operatorname{tr}\left(\frac{\partial v(\mathbf{z}(t), t; \theta)}{\partial \mathbf{z}(t)}\right) dt$$

CNFs offer several advantages over discrete flows. The memory requirement remains constant with respect to the complexity of the transformation, as only the vector field needs to be stored rather than a sequence of transformations [9]. The continuous formulation also allows for more natural modeling of temporal dynamics and provides a theoretical connection to physical transport processes. Furthermore, adaptive numerical integration methods can automatically trade off computation time and precision during both training and inference.

However, the primary challenge in training CNFs lies in the computational cost of evaluating the trace term in the log-likelihood computation, which requires $\mathcal{O}(d)$ forward passes through the network for dimension d. Various techniques have been proposed to address this limitation, including stochastic trace estimation and specialized architectural designs [8].

2.2.3 Flow Matching

Flow Matching represents a significant advance in training continuous normalizing flows by reformulating the problem through an optimal transport perspective [1]. This approach circumvents the computational challenges associated with traditional maximum likelihood training of CNFs while maintaining their theoretical advantages. At its core, Flow Matching defines probability transport through vector fields that describe instantaneous transformations between distributions. Given a source distribution p_0 and target distribution p_1, the method constructs a time-dependent vector field $v(\mathbf{x}, t)$ that generates a continuous path of probability distributions $\{p_t\}_{0 \le t \le 1}$ connecting p_0 to p_1.

The Flow Matching objective is formulated as:

$$\mathcal{L} = \mathbb{E}_{t\sim\mathcal{U}[0,1],\mathbf{x}\sim p_t}\left[\|v(\mathbf{x},t) - u(\mathbf{x},t)\|^2\right]$$

where v represents the learnable vector field and $u(\mathbf{x},t)$ is the target vector field that generates the desired probability path [2]. This regression objective eliminates the need for expensive ODE integration during training. A key innovation comes from the connection to optimal transport theory. By constructing conditional probability paths using optimal transport maps, the resulting vector fields become simpler and more efficient to learn [5]. The conditional formulation takes the form:

$$p_t(\mathbf{x} \mid \mathbf{z}) = \mathcal{N}(\mathbf{x} \mid \mu_t(\mathbf{z}), \sigma_t^2(\mathbf{z})\mathbf{I})$$

where $\mathbf{z}$ represents the conditioning variable, and μ_t and σ_t define the mean and standard deviation of the Gaussian conditional distributions. This framework establishes an important connection to score-based models, as the target vector field $u(\mathbf{x},t)$ can be interpreted as a score function scaled by the diffusion coefficient. However, Flow Matching offers greater flexibility in choosing probability paths and typically requires fewer function evaluations for sampling compared to score-based approaches [2].

The method's theoretical guarantees stem from the fact that when the vector field perfectly matches the target field, the resulting flow generates exactly the desired probability path [1]. This provides a principled approach to training continuous normalizing flows while avoiding the computational overhead of traditional likelihood-based methods.

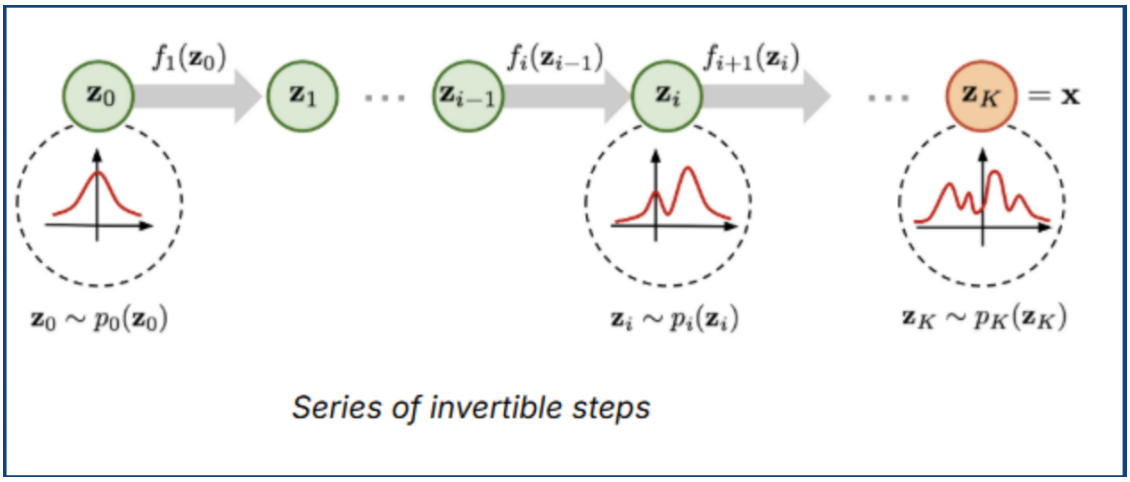

Figure 3: Visualization of steps of flow-matching model

3 Methodology

3.1 Dataset

The dataset used in this study was sourced from the 2022 DIGIX CTR dataset on Kaggle. The dataset is specifically designed for Click-Through Rate (CTR) prediction in the context of advertising. The dataset from the target domain includes user behavior logs, user profiles, and advertisement material information, while the source domain provides user behavior data and metadata about news feed items. By combining these two domains, the data supports the creation of user interest models that improve CTR prediction accuracy.

Our training dataset includes user demographic details (e.g., `age`, `gender`, `residence`), device metadata (e.g., `device_size`, `series_dev`, `series_group`), and ad-specific features (e.g., `creat_type_cd`, `adv_id`, `slot_id`). For feature importance analysis, we utilized XGBoost to identify the most significant features contributing to CTR prediction within the original dataset. XGBoost's capability to model complex, nonlinear relationships in high-dimensional tabular data made it particularly well-suited for this task.

The analysis of feature importance revealed that key predictors such as `creat_type_cd`, representing the creative type of advertisements and its impact on user engagement; `u_refreshTimes`, a measure of user refresh frequency that correlates strongly with interaction intensity; and `u_feedLifeCycle`, which captures the temporal relevance of advertisements, play a critical role in CTR prediction. These findings underscore the value of the original dataset in capturing key behavioral patterns. As discussed later in this paper, the same feature importance structure is maintained when analyzing the synthetic datasets, further demonstrating the fidelity of our data generation approach.

To prepare the dataset for modeling, continuous features were normalized to a range of $[-1, 1]$ using Min-Max scaling. The dataset was split into training and testing subsets, with 80% allocated for training and 20% for testing. To address class imbalance, undersampling was applied to both classes in the training set, ensuring a balanced representation of positive and negative labels. Additionally, Gaussian noise was added to create diverse inputs for synthetic data generation, and the processed features were duplicated multiple times for further modeling steps. This preprocessing pipeline establishes a robust foundation for building user interest models and improving CTR prediction accuracy.

3.2 Baseline Models

Building upon the dataset described in the previous section, we evaluated the performance of various baseline models to establish a reference for CTR prediction. This furnishes insights into the challenges of CTR prediction and the potential benefits of synthetic data generation. The models evaluated include Logistic Regression (L1), XGBoost, Random Forest, Multi-Layer Perceptron (MLP), and a Transformer-based model. Among these, Random Forest achieved the highest Accuracy (0.986) and ROC AUC Score (0.789), indicating its strong overall predictive ability. XGBoost also performed well with an Accuracy of 0.984 and a Precision of 0.474, demonstrating its ability to handle tabular data effectively.

MLP and Transformer models, which are more complex architectures, showed lower performance, with the Transformer achieving the highest Recall (0.60) but lower Accuracy

(0.7316). Logistic Regression exhibited reasonable Accuracy (0.975) but struggled with Precision and Recall.

3.2.1 Conditional Flow Matching (CFM)

Our model architecture for generating CTR synthetic data leverages Conditional Flow Matching (CFM) to create a transport map between source and target probability distributions. The framework extends traditional flow-based generative models by incorporating conditional information from both CTR labels and associated features. The mathematical foundation rests on defining a conditional probability path $p_t(\mathbf{x} \mid y, \mathbf{c})$, where $\mathbf{x}$ represents the input features, y represents CTR labels, and $\mathbf{c}$ represents conditional feature information. This path is parameterized using a Gaussian distribution:

$$p_t(\mathbf{x} \mid y, \mathbf{c}) = \mathcal{N}(\mathbf{x} \mid \mu_t(y, \mathbf{c}), \sigma_t^2(y, \mathbf{c})\mathbf{I})$$

where μ_t and σ_t are learned functions that parameterize the mean and standard deviation of the Gaussian distribution at time t. The learning objective follows the CFM formulation:

$$\mathcal{L} = \mathbb{E}_{t,p_t,y,\mathbf{c}} \left\| v(\mathbf{x}, t, y, \mathbf{c}) - u(y, \mathbf{c}) \right\|^2$$

For implementation, we employ the TorchCFM library to handle the core flow matching operations. The process begins with normalizing input features using MinMaxScaler to the range $[-1, 1]$. We then construct conditional paths by duplicating each real data sample K times ($K = 100$ in our implementation) to enable multiple noise samples per real data point. This approach enhances the diversity of the generated samples while maintaining the conditional relationships. The time discretization uses $n_t = 50$ equally spaced steps, providing sufficient granularity for the flow evolution while remaining computationally tractable. For each time step, we compute both the interpolated position ($\mathbf{x}_t$) and the target vector field ($\mathbf{u}_t$) using Independent Conditional Flow Matching (I-CFM).

Our approach differs from traditional neural network-based implementations by employing XGBoost models to learn the vector field, with one model per feature dimension at each time step. This decision leverages the superior performance of tree-based methods on tabular data while maintaining the theoretical guarantees of flow matching. The XGBoost models are configured with a maximum depth of 7 and 100 estimators, balancing model complexity with computational efficiency. To handle the categorical nature of CTR labels, we maintain separate models for each label class, allowing the flows to capture class-specific feature distributions. The integration process uses a simple Euler solver with adaptive step sizing, providing a robust balance between accuracy and computational cost. This implementation enables efficient generation of synthetic CTR data while preserving the complex relationships between features and maintaining the statistical properties of the original dataset.

3.2.2 XGBoost Integration

Our implementation integrates XGBoost as the primary vector field approximator within the flow matching framework, specifically tailored for CTR prediction data. The

vector field approximation is structured as a collection of XGBoost models, organized across three dimensions: time steps (n_t), feature dimensions ($\mathbf{c}$), and class labels. Each XGBoost regressor is configured with specific hyperparameters optimized for our use case, including a maximum depth of 7, 100 estimators, a learning rate (η) of 0.3, the histogram-based tree method (`"hist"`), and no regularization.

3.2.3 Training Process

The training process for our ForestFlow model combines flow matching principles with gradient boosting optimization. Duplicate samples ($K = 100$) are used per real data point to enhance stability and improve the learned vector field's quality. Learning occurs across $n_t = 50$ time steps, with parallel processing employed for efficiency.

3.2.4 Data Generation

The data generation process encompasses three main stages: initial sampling, flow integration, and post-processing. Sampling begins with drawing from a standard normal distribution as the noise source, followed by integration of learned vector fields using an Euler solver. Post-processing ensures adherence to constraints and statistical validity, maintaining high-quality synthetic CTR prediction data suitable for model training and evaluation.

4 Experiment Setup

4.1 Implementation Details

The primary model employed for Click-Through Rate (CTR) prediction was XGBoost. It was configured with a learning rate of 0.3, a maximum tree depth of 7, and 100 estimators. The tree method was set to hist for computational efficiency, while regularization parameters, including lambda and alpha, were both set to 0.0. A subsample of 1.0 was applied. For synthetic data generation, the Conditional Flow Matcher (CFM) operated with 50 time steps and 100 noise duplications, creating diverse inputs for training.The implementation leveraged Python, with libraries such as XGBoost, PyTorch, and scikit-learn forming the computational backbone. This setup provided the necessary performance to handle both the real and synthetic datasets effectively. The dataset was split into training and testing subsets with an 80-20 split. The training set was used to fit the model parameters, while the testing set evaluated final performance. To address class imbalance in the training data, under-sampling was applied, ensuring balanced representation of positive and negative labels.

4.2 Evaluation Metrics

Utility performance will be evaluated in the Model Performance section using multiple metrics, including Accuracy, ROC-AUC, Precision, Recall, and F1-Score. Accuracy measures the proportion of correctly classified predictions among all samples, offering a general assessment of the model's performance. ROC-AUC examines the trade-off between true positive and false positive rates across various thresholds, highlighting the model's ability to discriminate between positive and negative classes. Precision focuses on the proportion of correctly predicted positive instances out of all predicted positives, while Recall emphasizes the proportion of actual positive instances that were correctly identified. F1-Score provides a harmonic mean of Precision and Recall, balancing these two aspects to give a single performance measure. These metrics collectively provide a comprehensive evaluation of the model's ability to predict CTR accurately and reliably. To evaluate privacy, we utilized multiple metrics to ensure the synthetic data maintained an appropriate balance between utility and privacy. One key metric was the Nearest Neighbor Distance Ratio (NNDR), which measures the similarity between the nearest neighbors in the synthetic and real datasets. NNDR compares the distance to the nearest neighbor in the synthetic dataset with the distance to the nearest neighbor in the real dataset. A ratio closer to one indicates a reasonable balance between privacy preservation and the utility of the synthetic data. Additionally, Distance to Closest Record (DCR) was used to quantify the minimum distance between synthetic and real data points. Larger DCR values indicate reduced similarity between synthetic and real data, thus enhancing privacy. Privacy loss was another critical aspect evaluated using a holdout dataset. By comparing the NNDR of synthetic and real datasets on unseen data, we assessed the extent to which privacy is preserved in synthetic samples. This methodology ensures that synthetic data maintains statistical properties of the real data without compromising sensitive information. Finally, the Hitting Rate metric was applied to measure how frequently synthetic records matched exactly with real records, with lower values ensuring better privacy protection1.

Table 1: Privacy and Fidelity Metrics for Synthetic Data

Metric	**Value**	**Interpretation**
Nearest Neighbor Distance Ratio (~ 1)	1.664	Higher is better
Distance to Closest Record (> 1)	0.798	Higher is better
Privacy Loss (~ 0)	0.0429	Smaller is better
Hitting Rate (~ 0)	0.0	Smaller is better
KL Divergence (~ 0)	1.814	Smaller is better
Earth Mover's Distance (~ 0)	1.391	Smaller is better
Hellinger Distance (~ 0)	0.313	Smaller is better

To evaluate fidelity, metrics such as Kullback-Leibler (KL) Divergence and Earth Mover's Distance (EMD) were used to assess how well the synthetic data replicates the statistical distribution of the real data. KL Divergence quantifies the difference between the probability distributions of the real and synthetic datasets, with lower values indicating higher fidelity1. EMD measures the cost of transforming the synthetic data distribution to align with the real data distribution, where smaller EMD values signify better alignment. Additionally, Hellinger Distance was employed to compare categorical distributions, offering another perspective on fidelity. Lower Hellinger Distance values indicate that the synthetic and real data distributions are closely matched1.

5 Results and Analysis

5.1 Synthetic Data Quality

KDE plots can be used to assess feature similarity between real and synthetic data by visualizing the distribution of features for both datasets. Our KDE plots focus on the top five most important features, as determined by the XGBoost model described in the feature importance analysis in the following section. Overall, our KDE plots4 display highly overlapping curves, indicating that the distribution of features in the synthetic data closely mirrors that of the real data, thereby preserving general feature relationships. Ideally, these curves would be identical; however, due to random variations in the data, some minor spikes may appear in the feature values. For instance, the 'u_feedLifeCycle' feature, which describes user engagement on news feeds, slightly over-represents values in the interval between 16 and 18 in the synthetic data compared to the real data. Despite these minor variations in the density distributions, there are no clear outliers or unusual values in the synthetic data to suggest that the model failed to generate values consistent with the real data distributions. Overall, these plots demonstrate high fidelity in the synthetic data.

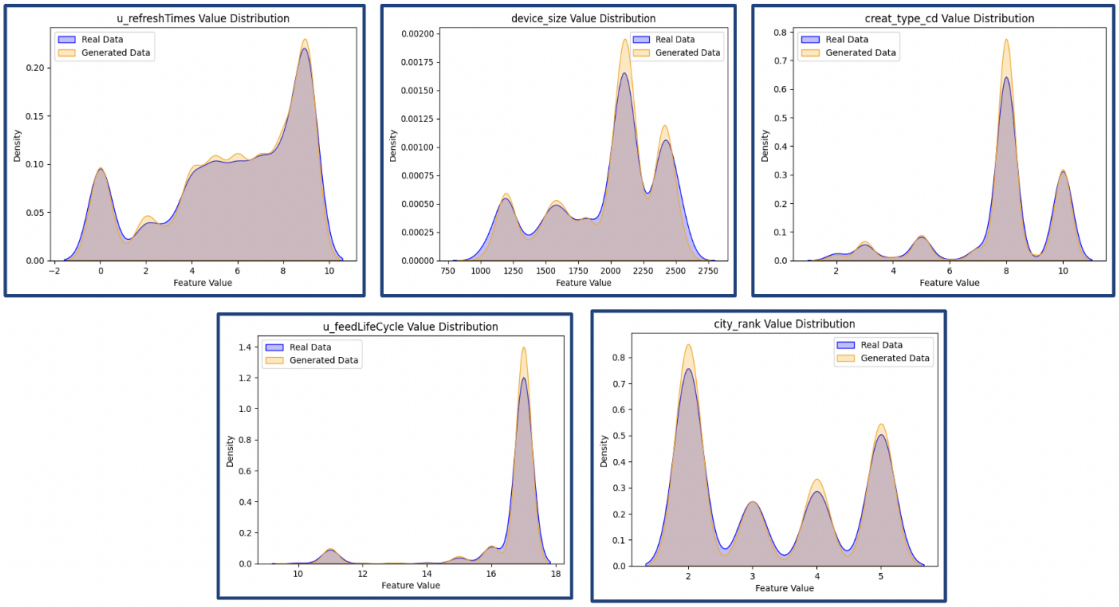

Figure 4: KDE plots visualizing the top five most important feature distributions

Comparing feature importance between the real and synthetic data helps determine whether the synthetic data identifies the same key features as important. This comparison serves as an indicator of fidelity, as it reveals whether underlying patterns are consistent across both datasets. The two feature importance plots below5 display the ranking of features most determinative of the outcome in an XGBoost classification model. This model is a simple classifier that predicts Click Through Rate (a binary target variable, 0 or 1) using the other features in the dataset. Both models were trained and evaluated on their respective datasets. As shown in the plots, the top three most important features are consistent be-

tween the two datasets, with slight variations in their rankings. These variations could arise for a couple of reasons. First, random fluctuations in the data may affect the model's feature rankings. Second, differences in feature distributions could contribute to these variations. From the KDE plots above4, we observe that the 'u_feedLifeCycle' and 'creat_type_id' features deviate slightly from the real data distribution, over representing their most common values. On the other hand, the 'u_refreshTimes' feature, ranked as the most important for the synthetic data, closely follows the real data distribution, preserving more of the original data structure. This may explain why the synthetic model ranked 'u_refreshTimes' as the most important feature, while the real data model ranked it second. Overall, the top features ranked as most important were largely consistent across both datasets. This comparison of feature importance indicates high fidelity in the synthetic data.

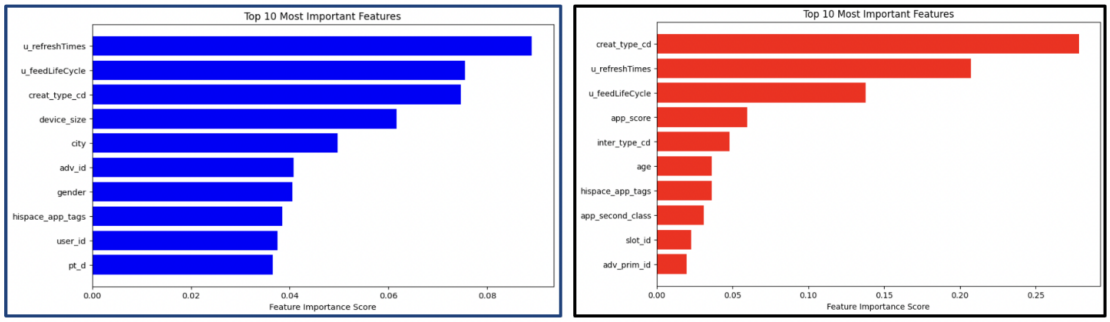

Figure 5: Feature importance for synthetic model (left) and real data model (right)

To visually represent the data distribution between the real and synthetic datasets as a whole, a T-SNE plot was created. T-SNE is a dimensionality reduction technique that visualizes high-dimensional data in a 2-dimensional plot, helping to explain the distribution of ad performance features. It helps evaluate how closely the generated (synthetic) data resembles the real data. The plot6 shows a significant overlap between the generated and real data, indicating high fidelity between the two datasets, as their dimensional reductions are similar. Furthermore, there are no clear separations, suggesting that user behavior in the synthetic data aligns well with real data.

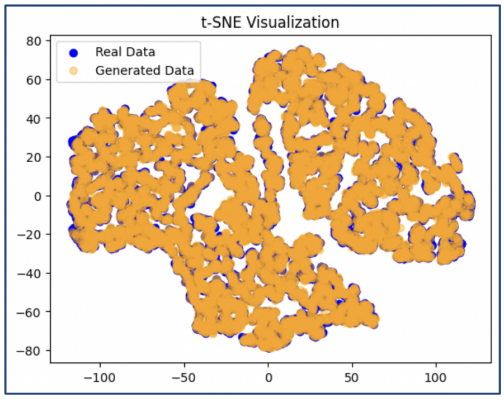

Figure 6: T-SNE plot comparing real and generated data

5.2 Privacy and Fidelity Analysis

Privacy and fidelity metrics demonstrated a solid balance between preserving privacy and maintaining data utility. NNDR and DCR indicated strong privacy preservation, while fidelity metrics such as KL Divergence suggested moderate alignment with real data distributions.

5.3 Model Performance

To compare the utility of our synthetic dataset we compare the performance of a simple XGBoost model trained on original data versus a combination of original and synthetic data. Both models were evaluated on a testing dataset containing only real data to ensure consistent results across all of the metrics. The original dataset consisted of 7 million observations, while the synthetic dataset added 1 million more, representing a 14% increase in data observations. Notably, the F1-Score, a crucial metric for evaluating model performance on highly class imbalanced datasets like this one, saw a minor improvement when synthetic data was included2. This is likely due to the additional positive observations in the training data, which provided the model with more positively labeled observations to better make predictions regarding those observations.

While the accuracy and ROC AUC only had minor improvements, the precision and F1-Score showed more noticeable increases in the model trained with combined data2. These metrics are particularly important for this use case, as they minimize irrelevant ads shown to users and ensure a balanced trade-off between identifying the majority of ad clicks and avoiding false positives which was one of the main issues with our original model. Overall, the results demonstrate that augmenting the training data with high-quality synthetic observations can enhance predictive performance without compromising interpretability.

Metric	**Original Data**	**Original + Synthetic Data**
Accuracy	0.970	0.973
ROC AUC	0.803	0.804
Precision	0.198	0.251
Recall	0.242	0.225
F1-Score	0.218	0.237

Table 2: Performance comparison between models trained on original data and combined data.

6 Discussion

The flow-based model demonstrated its effectiveness in generating synthetic data that closely resembles the real data. This is shown by not only the improved utility metrics, such as F1-Score, precision, and accuracy, when synthetic data was incorporated into the model but also the high utility of our synthetic data. While this approach is relatively uncommon for tabular dataset, more commonly used models such as GANs or VAEs are used, the results suggest that this type of model can be used for enhancing model performance. Additionally, synthetic data introduces a critical balance in the privacy-utility trade-off by somewhat effectively masking sensitive information while still maintaining high data fidelity, mitigating privacy risks without sacrificing predictive power. However, limitations such as the need for increasing the volume of the generated synthetic data and exploring its impacts on datasets with different class distributions remain a challenge and could be further explored through more research.

7 Conclusion and Future Work

7.1 Conclusion

This model was able to handle all of the goals we set out for it to accomplish. Our flow-based model created a synthetic dataset that statistically represented our real dataset, while the accuracy of that analyzed data increased. And on top of this, the real dataset became encrypted, so as to obscure all real user information. This means our model met our expectations, as it improved all metrics of utility, privacy, and fidelity. It proceeded to then exceed our expectations by being able to be trained in a short, limited time. The model was able to generalize well across the unseen CTR data, highlighting its practical significance. This would mean that it can seamlessly be applied to real-world advertising platforms, optimizing a company's campaign without compromising user privacy. And so, this would enable businesses to make data-driven decisions while still adhering to security regulations. By improving both ad targeting accuracy and fostering trust between users and advertisers, this model is invaluable for its versatility in advancing CTR prediction solutions.

7.2 Future Directions

Although our model performed well, there are still ways we can work on improving it. First, fine-tuning the flow-based model to further enhance its performance against other generative techniques, like GANS and VAES. Since GANs excel at generating highly realistic data, and VAEs offer strong probabilistic foundations, comparing these models would be critical for further improving our metrics. It would also be useful to scale the model to generate larger amounts of synthetic data, as this should directly increase utility and thus evaluate the CTR predictions better. Outside of CTR predictions, this model could be applied to other fields, like user engagement, fraud detection, or recommendation systems, where privacy is vital. Our findings could also help us dive into other research opportunities, like exploring new evaluation frameworks for generative models investigating hybrid approaches with GANS and VAEs. This project could lead to more innovations in the world of synthetic data and its applications to real life.

8 Acknowledgments

The code used to generate and evaluate the synthetic vs. real data can be found in this repository: https://github.com/millaenielsen/CTR-prediction

References

[1] Lipman, Y., Voleti, R., Baque, D., Bronstein, M.M., Cohen-Or, D., Flow matching for generative modeling, *arXiv preprint arXiv:2210.02747*, 2022.

[2] Tong, A., Malkin, N., Huguet, G., Zhang, Y., Rector-Brooks, J., Fatras, K., Wolf, G., Bengio, Y., Conditional flow matching: Simulation-free dynamic optimal transport, *arXiv preprint arXiv:2302.00482*, 2023.

[3] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y., Generative adversarial networks, *Communications of the ACM*, 63(11):139–144, 2020.

[4] Cinelli, L.P., Marins, M.A., Silva, H.M., Petraglia, A., Variational autoencoder, *Variational Methods for Machine Learning with Applications to Deep Networks*, Springer International Publishing, 111–149, 2021.

[5] Generale, A.P., Robertson, A.E., Kalidindi, S.R., Conditional Variable Flow Matching: Transforming Conditional Densities with Amortized Conditional Optimal Transport, *arXiv preprint arXiv:2411.08314*, 2024.

[6] Winkler, C., Worrall, D., Hoogeboom, E., Welling, M., Learning likelihoods with conditional normalizing flows, *arXiv preprint arXiv:1912.00042*, 2019.

[7] Jolicoeur-Martineau, A., Fatras, K., Kachman, T., Generating and imputing tabular data via diffusion and flow-based gradient-boosted trees, *International Conference on Artificial Intelligence and Statistics*, PMLR, 2024.

[8] Tong, A., Malkin, N., Huguet, G., Zhang, Y., Rector-Brooks, J., Fatras, K., Wolf, G., Bengio, Y., Improving and Generalizing Flow-Based Generative Models with Minibatch Optimal Transport, 2023.

[9] Tong, A., Malkin, N., Fatras, K., Atanackovic, L., Zhang, Y., Huguet, G., Wolf, G., Bengio, Y., Simulation-Free Schrödinger Bridges via Score and Flow Matching, 2023.