SynthRank: Synthetic Data Generation of Individual's Financial Transactions Through Learning to Ranking

Waylon Li[1], Mengyu Wang[1], Carsten Maple[2,3], Tiejun Ma[1,3]

[1]University of Edinburgh
[2]University of Warwick
[3]The Alan Turing Institute

1. Background
2. Dataset & Task
3. Methodology: SynthRank
4. Experiments
5. Conclusions

1 Background

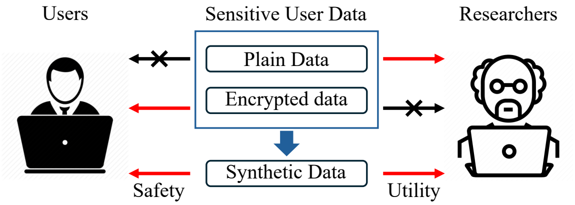
2 Dataset & Task

3 Methodology: SynthRank

4 Experiments

5 Conclusions

Why Synthetic Data For Financial Transactions?

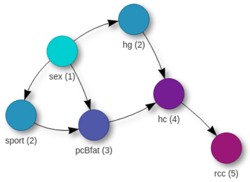


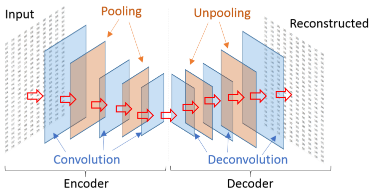
In the finance sector, significant constraints exist in accessing data both within and across organisations, since the data usually contains sensitive information including personal detail, which preclude it being shared.

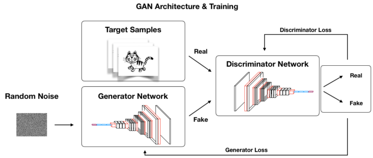
Previous Work

Three predominant generative models for tabular data:

- Bayesian Networks
- Auto-Encoders
- Generative Adversarial Networks (GANs)







However...

However, some recent generation methods struggled to achieve the balance or lack a comprehensive assessment of privacy, utility, and fidelity [4].

Utility, Privacy, Fidelity

- **Utility:** How helpful it is for a specific task [3].
- **Privacy:** How much it reveals about the actual data it's based on [3].
- **Fidelity:** How closely the synthetic data matches the real data in terms of patterns [3].

Target: prioritise utility and privacy.

1. Background
2. **Dataset & Task**
3. Methodology: SynthRank
4. Experiments
5. Conclusions

Dataset

A unique dataset from leading UK-based trading sectors, containing individual trading transactions.

- 13,607,120 trading records from November 2003 to July 2014.
- 20,514 active traders.
- Binary Classification (risky / non-risky)

Dataset

The following table shows the features in the trasactions dataset:

Feature	Description	Feature	Description
	Continuous Features		
PerFTSE20	Share of trades placed in the FTSE100	ProfitxDur20	Interaction of ProfitRate20 and DurationRate20
AVGPTS3_20	P&L in Points $\geq$ 3 during the last 20 trades	PassAvgReturn	Avg. return
ProfitRate20	Average profit rate of the trader in the past 20 trades	AvgShortSales20	Share of short positions in the past 20 trades
WinTradeRate20	Trader's average winning rate in the past 20 trades	DurationRate20	Average time trader leaves winning vs losing position open
SharpeRatio20	Mean/st.dev. of returns in the past 20 trades	AvgOpen20	Average of the P&L among trader's past 20 trades
DurationRatio20	Mean trade duration (mins) / std.dev. duration of past 20 trades	OrderCloseRate20	% of trades closed by an order in the last 20 trades
TradFQ20	The number of trades on average that a typical trader poses daily	NumTrades	Accumulated until the last 20 trades
NextTotalPL_GBP20	P&L for the next 20 trades in the future	NextTotalPL_GBP	P&L for the next 100 trades in the future
WinningRate	Winning rate of the next 100 trades in the future	SharpNext100	Mean/st.dev. of the returns of trader's next 100 trades
TotalTrades	Total trades made by a trader	Period	Buckets of 20 trades per account
	Discrete Features		
Age1	Indicate which age group the trader belongs to	MarketCluster5	Indicate which market cluster the stock belongs to
Age2	Indicate which age group the trader belongs to	MarketCluster6	Indicate which market cluster the stock belongs to
Age3	Indicate which age group the trader belongs to	MarketCluster7	Indicate which market cluster the stock belongs to
Age4	Indicate which age group the trader belongs to	MarketCluster8	Indicate which market cluster the stock belongs to
Age5	Indicate which age group the trader belongs to	Segment1	Categories of past 20 trades' average return
Mobile	Indicate what device the trade is made from	Segment2	Categories of past 20 trades' average return
MarketCluster0	Indicate which market cluster the stock belongs to	Segment3	Categories of past 20 trades' average return
MarketCluster2	Indicate which market cluster the stock belongs to	accountid	The account ID of the trader
MarketCluster3	Indicate which market cluster the stock belongs to		

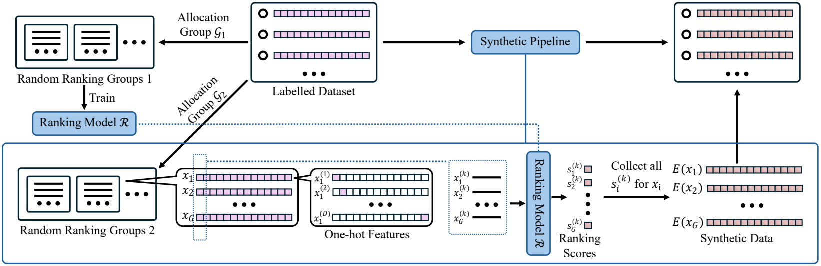
1. Background
2. Dataset & Task
3. Methodology: SynthRank
4. Experiments
5. Conclusions

LEarning-TO-Rank (LETOR) for Generation

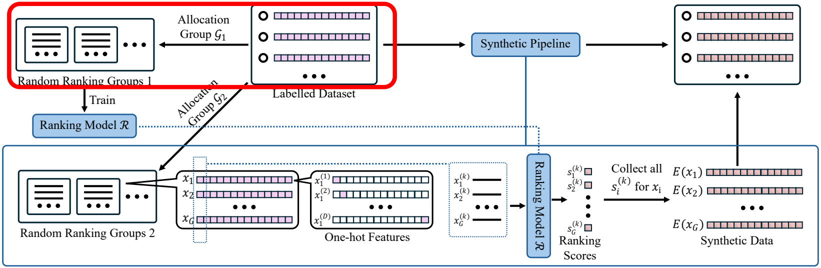
Two things needed for ranking models:

- Ranking group (known as *qid* in information retrieval) is crucial as items are ranked within their group. This is also our motivation for introducing ranking algorithm for the problem.
- A ranking label is required for each item. We can directly use the risky trader label as a binary relevance label, similar to those in retrieval datasets.

LEarning-TO-Rank (LETOR) for Generation

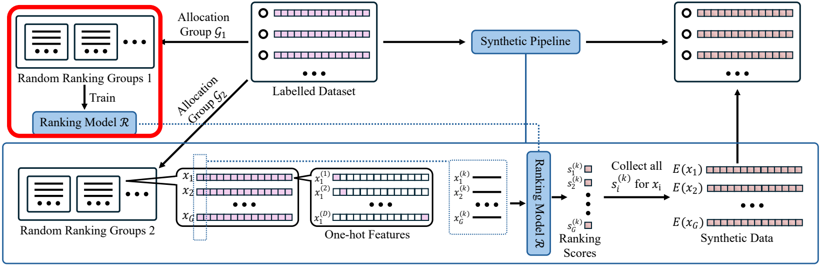


LEarning-TO-Rank (LETOR) for Generation



Step 1: Allocate the trading items using a ranking group allocation strategy $\mathcal{G}_1$, where the trading items are allocated into a branch of buckets with size G.

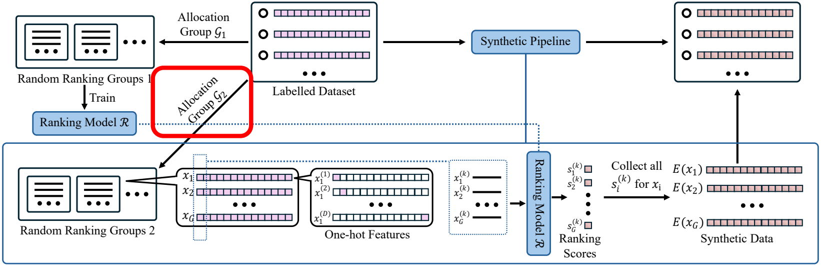
LEarning-TO-Rank (LETOR) for Generation



Step 2: Train a ranking model $\mathcal{R}$ on a grouped and labelled dataset. During training, the model predicts ranking scores $(s_1, \ldots, s_G)$ for a group of items $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_G\}$, where each item is a d-dimensional vector representing d features. Here, G is the group size and s_i is a real-valued score.

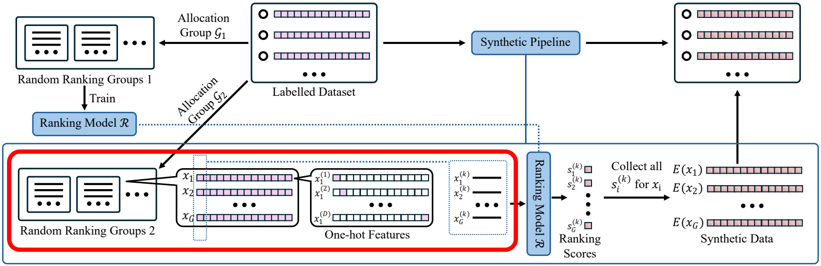
$$\mathcal{R}(\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_G\} \mid \mathcal{G}_1) \to [s_1, \ldots, s_G] \tag{1}$$

LEarning-TO-Rank (LETOR) for Generation



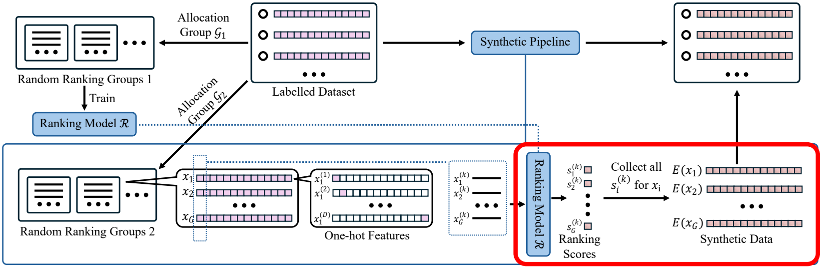
Step 3: During generation, given a subset or the complete training set, we allocate the trading items using the same strategy used for training, but with different random seeds, following the group size G. The original dataset is split into ranking groups with size G using the group allocation $\mathcal{G}_2$.

LEarning-TO-Rank (LETOR) for Generation



Step 4: A trading item $\boldsymbol{x}_i$ is broken down into D input items $\{\boldsymbol{x}_i^{(1)}, \boldsymbol{x}_i^{(2)}, \cdots, \boldsymbol{x}_i^{(D)}\}$, where each $\boldsymbol{x}_i^{(j)}$ represents the vector with all other $d-1$ features except the j-th feature being masked by 0.

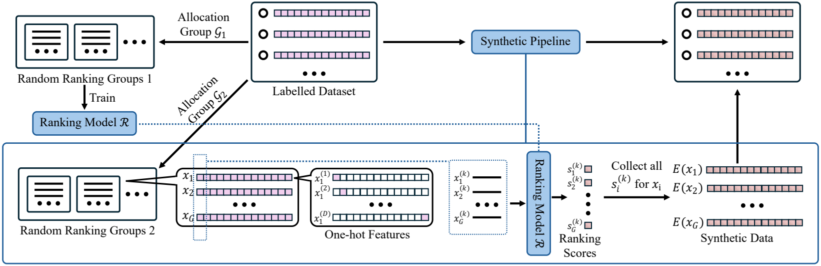
LEarning-TO-Rank (LETOR) for Generation



Step 5: We run the ranking model for D times, denoting each separate run as $\mathcal{R}_j$ to obtain feature-specific ranking scores:

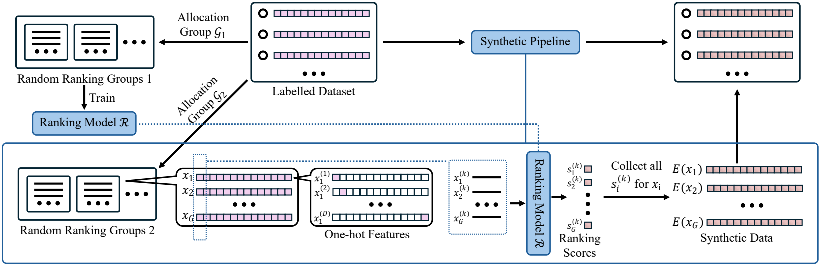
$$\mathcal{R}_j(\{\boldsymbol{x}_1^{(j)}, \ldots, \boldsymbol{x}_G^{(j)}\} \mid \mathcal{G}_2) = (s_{1j}, \ldots, s_{Gj}) \tag{2}$$

LEarning-TO-Rank (LETOR) for Generation



- **Utility:** the ranking score embedding directly helps with the prediction task. It also offers flexibility where different ranking group allocation can lead to variations in ranking outputs.

LEarning-TO-Rank (LETOR) for Generation



- **Utility:** the ranking score embedding directly helps with the prediction task. It also offers flexibility where different ranking group allocation can lead to variations in ranking outputs.
- **Privacy:** two elements of knowledge are intrinsic to our generation process, the ranking group allocation and the ranking model, which are concealed from attackers.

1 Background

2 Dataset & Task

3 Methodology: SynthRank

4 Experiments

5 Conclusions

Benchmark Models

Auto Encoder:

- TVAE (2019) [6]

GANs:

- CTGAN (2019) [6]
- CopulaGAN (2023) [2]

GPT:

- REaLTabFormer (2023) [5]

Ours:

- SynthRank (with LambdaMART [1])

Evaluation Design

Utility:

- Predictive power test

Privacy:

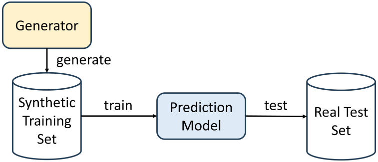
- Attribute Inference Attack (AIA)

Fidelity:

- Kolmogorov-Smirnov (KS) test statistics
- Kernel Density Estimate (KDE) plots
- Quantile-Quantile (Q-Q) plots

Utility Results

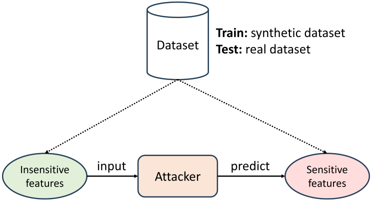
Data Source	F_1	P&L	AUC
Results with RF Classifier			
Original	0.511	127.738	0.837
Baseline Synthetic Data Generators			
TVAE	0.498	127.060	0.732
CTGAN	0.510	127.833	0.781
CopulaGAN	0.499	127.203	0.775
RTF	0.499	127.098	0.824
Our SynthRank Pipelines			
LMART(XGB:pair)	0.548	128.185	**0.863**
LMART(XGB:ndcg)	0.558	128.985	0.849
LMART(XGB:map)	**0.567**	129.160	0.858
LMART(LGBM)	0.566	**131.269**	0.857
Results with MLP Classifier			
Original	0.522	127.968	0.838
Baseline Synthetic Data Generators			
TVAE	0.509	127.539	0.648
CTGAN	0.531	124.721	0.742
CopulaGAN	0.517	125.907	0.729
RTF	0.498	127.131	0.805
Our SynthRank Pipelines			
LMART(XGB:pair)	0.543	131.207	**0.862**
LMART(XGB:ndcg)	0.536	129.858	0.848
LMART(XGB:map)	0.560	128.451	0.845
LMART(LGBM)	**0.567**	**131.547**	0.856



- REaLTabFormer, while a more recent model, falls short in F1 performance.
- SynthRank shows consistent improvement with different implementations of LambdaMART.

Privacy Results

Attacker	TVAE	CTGAN	CopulaGAN	RTF	SynthRank
RF	0.4510	0.4374	0.4288	0.3973	**0.9453**
KNN	0.5155	0.5021	0.4850	0.1442	**0.9453**
MLP	0.4414	0.4311	0.4293	0.4324	**0.9117**



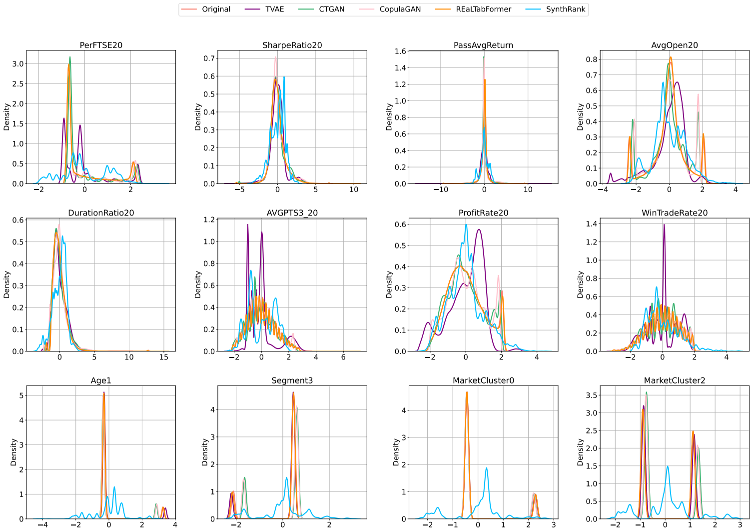
- The PAI score is derived from (1 - Acc), where Acc is the accuracy of an attacker in inferring true sensitive information.

Fidelity Results - KS Test

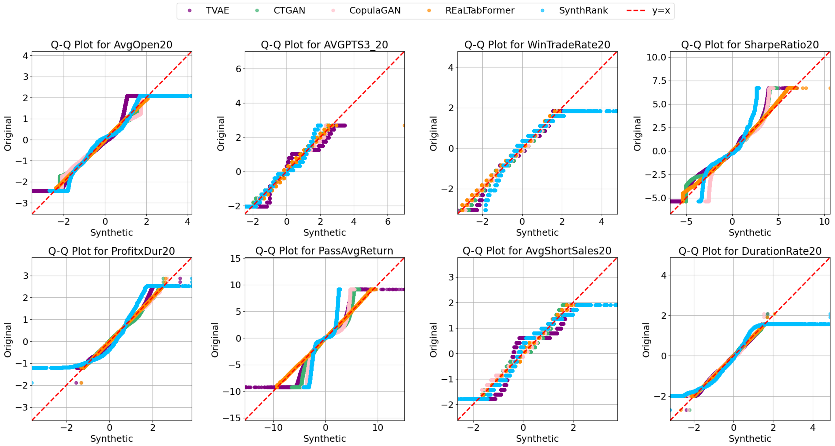
Feature Type	TVAE	CTGAN	CoGAN	RTF	SynthRank
Continuous	0.188	0.147	0.152	**0.129**	0.153
Discrete	0.082	0.097	0.145	**0.003**	0.460
All	0.128	0.119	0.148	**0.058**	0.326

- REaLTabFormer is the strongest performer in replicating the feature distributions.
- SynthRank achieves similar performance as CopulaGAN and CTGAN on continuous features, but fall shorts on discrete features due to the transformation into continuous forms.

Fidelity Results - KDE Plot



Fidelity Results - QQ Plot



1 Background

2 Dataset & Task

3 Methodology: SynthRank

4 Experiments

5 Conclusions

Conclusions

- The first attempt towards using learning-to-rank algorithms for the generation of synthetic financial transaction data.

Conclusions

- The first attempt towards using learning-to-rank algorithms for the generation of synthetic financial transaction data.
- We introduce SynthRank, a task-oriented pipeline for synthetic financial transaction data generation.

Conclusions

- The first attempt towards using learning-to-rank algorithms for the generation of synthetic financial transaction data.
- We introduce SynthRank, a task-oriented pipeline for synthetic financial transaction data generation.
- Our findings shed light on an interesting observation: while not exactly mirror the original distribution, SynthRank remarkably preserves pivotal characteristics.

Conclusions

- The first attempt towards using learning-to-rank algorithms for the generation of synthetic financial transaction data.
- We introduce SynthRank, a task-oriented pipeline for synthetic financial transaction data generation.
- Our findings shed light on an interesting observation: while not exactly mirror the original distribution, SynthRank remarkably preserves pivotal characteristics.
- In the context of financial transaction data, SynthRank provides accurate predictions while securing sensitive information, making it as a preferred choice in this scenario.

References

[1] Burges, C.
From ranknet to lambdarank to lambdamart: An overview.
Learning 11 (01 2010).

[2] Espinosa, E., and Figueira, A.
On the quality of synthetic generated tabular data.
Mathematics 11, 15 (2023).

[3] Jordon, J., Szpruch, L., Houssiau, F., Bottarelli, M., Cherubin, G., Maple, C., Cohen, S. N., and Weller, A.
Synthetic data – what, why and how?, 2022.

[4] Ram Mohan Rao, P., Murali Krishna, S., and Siva Kumar, A. P.
Privacy preservation techniques in big data analytics: a survey.
Journal of Big Data 5, 1 (Sep 2018), 33.

[5] Solatorio, A. V., and Dupriez, O.
Realtabformer: Generating realistic relational and tabular data using transformers, 2023.

[6] Xu, L., Skoularidou, M., Cuesta-Infante, A., and Veeramachaneni, K.
Modeling Tabular Data Using Conditional GAN.
Curran Associates Inc., Red Hook, NY, USA, 2019.

Thanks!

Contact:

- Waylon Li (waylon.li@ed.ac.uk)
- Mengyu Wang (M.Wang-100@sms.ed.ac.uk)
- Prof. Carsten Maple (cm@warwick.ac.uk)
- Prof. Tiejun Ma (tiejun.ma@ed.ac.uk)