Report for Simulation of Credit Default Swap Index Transaction Data

Nan Yang

GANs are a type of generative model that learns high-dimensional continuous distributions, which is, in most cases, image data. In this report, we present an example showing that GANs have great potential in dealing with financial data as well. In finance, since the Credit Default Swap (CDS) index trade data is rare to find and complex in structure, our goal is to build a simulator that can artificially generate more CDS index trade data that are similar to existing trading records. We train a special type of Generative Adversarial Network, the Sequential GAN (SeqGAN) to achieve this.

After constructing the synthetic dataset, we then explore various ways and metrics to validate the effectiveness of the synthetic dataset. For example, we examine the marginal distributions and the correlations among variables. We also test the relative effectiveness of popular machine learning algorithms on both the original and the synthetic dataset.

1 Introduction

Credit Default Swaps are a type of financial credit derivative, designed for buyers to transfer some of the risk exposure on the underlying product, usually corporate bonds, mortgate-backed securities (MBS) or municipal bonds, to the sellers. In the most basic case, the buyers pay the sellers a predetermined price, or spread, on a regular basis. Whenever a special credit event that is specialized in the contract happens, for example, when the underlying bond defaults, the buyer receives payment from the sellers. In essence, the sellers are protecting the buyers from the credit risks by issuing insurance.

CDSs are usually highly customized and traded not on exchanges, but over-the-counter. CDS index, on the other hand, is a basket of CDSs, which is standardized and traded on the exchange. There are two major types of corporate CDS indices: CDX, which contains companies from North America and emerging markets, and iTraxx. Multiple types of CDX are available on the market with focus on different risk levels, for example, High Yield (HY) and Investment Grade (IG). The composition of these CDXs are recalibrated and changed regularly (usually every six months). Their prices generally reflect the credit risk evaluated by the market.

1.1 Related Works

GANs for data augmentation. Generative neural networks (GANs) are typically used for generating images that are similar to existing ones in the dataset. Since it learns the underlying distribution of observations in an non-parametric way, people have been exploring the uses of GANs in data augmentation, and the datasets are sometimes not restricted to image data. Data Augmenting GAN (DAGAN) ([1]) is a new structure of GAN that can improve the accuracy of classifiers by generating additional samples in the low-data regime. [3] augments the performance of brain segmentation by introducing a Progressive Growing of GANs (PGGAN) network. [6] uses a convolutional neural network (CNN) to output synthetic computed tomography (CT) image data in order to improve liver lesion classification. Please refer to [14], [18], [16] and [20] for more applications.

Simulating CDS. For simulating CDS prices, people have been considering probabilistic techniques in the literature. Most of them use credit risk models to simulate default events and generate spreads or prices. To name a few, see [13], [11], [10] and [17].

Deep learning in Finance. The uses of deep learning in finance, on the other hand, is a relatively new field. [9] uses deep learning to calibrate stochastic models for volatility surfaces, for example, the rough Bergomi model, the Heston model and the Bergomi model. The inputs of the network are parameters in the stochastic models, that is, the coefficients of the Brownian motion and the drift terms. The outputs are the volatility at different maturity and strikes. Their method is two-step: 1. Simulate by Monte-Carlo, using the stochastic models and known parameters, thousands of Brownian motion paths as the training set. Then the neural network is optimized by minimizing the mean square error at each grid point of the volatility surface. 2. Simulate another test set. Then try to recover the model parameters using the trained neural network in step 1 (learn the inputs of the networks). In summary, their neural network approximates the mapping between model parameters and the paths of Brownian motions.

[4] adapts the conditional no-arbitrage moment condition to an min-max optimization problem with adversarial loss. The Stochastic Discount Factor weights try to minimize the loss, while the conditioning functions (of macroeconomic and firm-specific characteristics) try to maximize it. The SDF and the conditioning functions can each be approximated by a neural network with LSTM units. This is similar to the GAN structure (two networks with adversarial loss) hence they apply the iterative training procedure from GAN to optimize everything. By definition, the SDF weights are the weights of the conditional mean-variance efficient portfolio, which maximizes the Sharpe ratio.

[5] trains a neural network, which applys a regression-like model that uses exposures to macroeconomic and firm-wise characteristics in a non-linear fashion, to predict excess asset returns. [8] compares deep neural nets with other machine learning models on the performance of measuring asset risk premia.

[15] also uses neural networks to predict asset returns, while [2] predicts bond returns. [7] uses autoencoder plus the no-arbitrage assumption, to look for nonlinear latent factors that drive asset prices.

2 Dataset

The dataset is from the public Swap Data Repository (SDR), which is hosted by the Depository Trust & Clearing Corporation (DTCC). According to the U.S. Commodity Futures Trading Commission, the swap dealers are required by the Dodd-Frank Act to report all CDS index (CDX) transactions to registered SDRs. The real-time reporting contains swap transaction and pricing data, available on a daily basis.

We downloaded the credit swap index message data from 01-01-2014 to 02-28-2019, a total of 1885 days, from DTCC's website, among which files of 30 days are missing. After combining the files, the dataset contains a total of 1103307 records with 44 features. Some of the features are categorical variables, for example, the `taxonomy` variable indicates whether the CDX is high-yield (high credit risk) or investment-grade, the `action` variable, like what we usually observe in a limit order book, tells us whether this record is for posting new orders to the market (new), correcting an existing order on the market (correct), or just cancel an outstanding order (cancel).

Some variables are continuous in nature. For example, the `price` variable tells us the price at which the order is posted or traded. However, the price is truncated if it is above certain thresholds that depend on each individual records.

The features are also highly unbalanced. For example, for the `action` feature among all records, 904440 (82%) of them are new orders, 124583 (11%) are order cancellations, and the rest (less than 7%) are corrections of existing orders. The large fraction of the orders being new is very different from what we usually observe on the equity market, where most orders should be cancellations.

The quality of records are relatively low. 10 out of 44 features are completely unpopulated. Other features contain NaN values, or incorrect values that do not match the descriptions of that column. Preprocessing is hence an essential step in this example.

2.1 Preprocessing

In this section, we describe the preprocessing done on the dataset. We perform a minimum amount of manipulation on the records to ensure that even without much guidance or domain knowledge, the neural network would still be able to capture the essence of the structure of our dataset. The preprocessing therefore will focus on removing redundant or uninformative rows and columns.

For this example, instead of including all 44 variables, of which most of them are irrelevant or of low data quality, we decide to use 7 of the most relavant features: `action`, `cleared`, `indication_of_collateralization`, `taxonomy`,

price_forming, price and notional_amount. The following list summarizes their possible values:

- action: new, correct, cancel.
- taxonomy: investment grade (IG), high yield (HY).
- price_forming: trade, novation, partial termination, amendment.
- cleared: cleared, uncleared.
- indication_of_collateralization: collateralized, not collateralized.
- price: continuous.
- notional_amount: continuous.

We proceed through the following steps to preprocess the dataset and prepare for training:

1. Select only orders whose notional_currency is in U.S. dollars. This reduces the number of rows from 1103307 to 823555.
2. Select only orders whose taxonomy is a high-yield or investment-grade credit default swap index (CDX). This reduces the number of orders to 300112.
3. Select only orders that has a duration of around 5 years (4.5 to 5.5 years). The duration is calculated as the difference of end_date and effective_date. Most of the orders have an duration in this range. The number of rows is now 285134.
4. To avoid duplication of orders, we create a list of order IDs that appear multiple times in the dataset. We then drop new orders whose dissemination_ID are in the list, cancel orders whose original_dissemination_ID are in the list, and correct orders whose dissemination_ID are in the list. This way we would only consider the most-updated orders. We end up with 245104 orders.
5. We only keep orders whose price_notation_type is in basis points or in percentage, leaving us 158632 orders.
6. We keep orders whose price_notation is not 0. This reduces the number of orders to 158131.
7. To deal with the inconsistency in price notations. We normalize the price notations in the following way:
    - For prices quoted in percentage, some of the orders are actually quoted in basis points, which are easily detected if the prices are in the range of [1000, 100000]. We convert them by dividing the price by 10000.

- For prices quoted in basis points, we convert them to percentage by dividing the price by 100.

8. In the current version of the model, we drop all rows that has an NaN in them. This reduces the number of rows to 30857.

9. To transform the continuous features to discrete ones, we construct bins for the `price`s and the `notional_amount`s.

10. We combine each 30 contiguous records into a sequence. We then add a token `<GO>` to the head of each sequence and append a token `<END>` to the end of each sequence.

3 Model Setup

In this example, we are dealing with a mixture of continuous and discrete type data, that is, some features are discrete (categorical) and some features are continuous (positive real numbers). In addition, the data is actually a time series: contiguous samples are often posted in a clustered fashion, and therefore their features and prices are correlated. We apply Sequential GAN (SeqGAN) ([19]) to address these two problems through policy gradients. The following section describes how SeqGAN works with a discrete and sequencial underlying distribution.

3.1 SeqGAN

In SeqGAN, the objective is to train a generative neural network G_θ parameterized by θ to generate a random sequence $Y_{1:T} = (y_1, \ldots, y_t, \ldots, y_T)$, where $y_t \in \mathcal{Y}$ and $\mathcal{Y}$ is our action space, also called vocabulary in this setting. At the same time, a discriminator D_ϕ, parameterized by ϕ, is trained so that the output on the real data $D_\phi(Y^n_{1:T})$ is close to 1, and the output on the generated data $D_\phi(Y_{1:T})$ is close to 0. The objective function on the discriminator side is hence similar to a vanilla GAN,

$$\min_\phi -\mathbb{E}_{Y\sim p_{\text{data}}}[\log D_\phi(Y)] - \mathbb{E}_{Y\sim G_\theta}[\log)1 - D_\phi(Y)]. \tag{1}$$

The structure of SeqGAN is visualized in Figure 1.

The generator network, G_θ, is trained to maximize the expected end reward over all possible actions y_1 given the current state s_0,

$$J(\theta) = \mathbb{E}[R_T|s_0, \theta] = \sum_{y_1\in\mathcal{Y}} G_\theta(y_1|s_0)\cdot Q^{G_\theta}_{D_\phi}(s_0, y_1). \tag{2}$$

Since the discriminator network D_ϕ only gives an output when the whole sequence is finished, SeqGAN applies Monte Carlo search that samples future possible actions from the vocabulary. A rollout policy is used for generating the remaining time periods. In this example, the rollout policy is the current best generator.

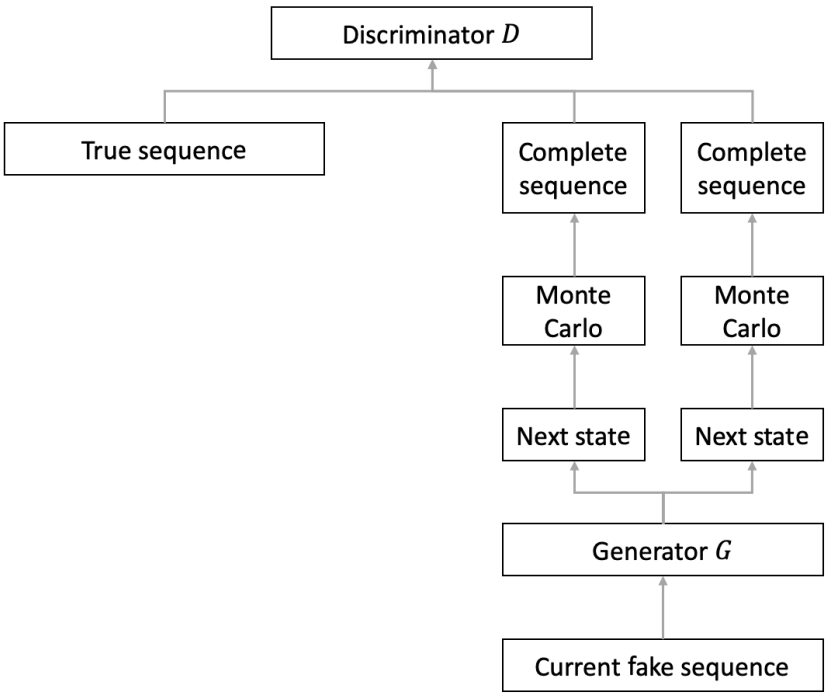

Figure 1: Structure of SeqGAN

3.2 Network Structure

The generator network is a recurrent neural network (RNN) that uses Long-Short Memory Units (LSTMs) as building blocks. In addition to all the possible combinations of features, three special tokens are added to the dictionary as possible actions to take by the generator: `<GO>`, `<PAD>` and `<END>`. The input goes through several hidden layers before being fed into a softmax layer to output the probabilities of taking each possible action. A maximum sequence length of 50 is allowed before an `<END>` is appended to the end of each sequence.

The discriminator network is a convolutional neural network (CNN). For each sequence, we apply a series of convolutional kernels with different sizes to it, and then take the max-pooling over all outputs as the feature. The final layer is a fully connected layer with sigmoid as the activation function.

Hyperparameter Settings For this example, we use mostly the default settings in the original SeqGAN paper. On the generator side, the embedding dimension is 32, the hidden dimension is 100 and the maximum sequence length is 50. The batchsize is 64. On the discriminator side, the embedding dimension is 32, the kernels' sizes are [1,2,3,4,5,6,7,8,9,10,15], the number of kernels are

[100, 200, 200, 200, 200, 100, 100, 100, 100, 100, 160], the dropout rate is 0.25, the L^2 regularization parameter is 0.2, the batchsize is 64.

4 Evaluation of results

The similarity between generated records and the historical true records is evaluated using the following criteria,

1. Marginal distribution of each individual features;
2. Dependency of each pair of features, measured by the correlation coefficients;
3. Stability of accuracy of machine learning models on the generated and the true datasets.

Marginal distributions We check the marginal distributions of each variables. Since they are all categorical, we plot out the pie charts that show the percentage of every possible category. Below are shown some samples:

Figure 2 shows the marginal distribution of the `action` variable.

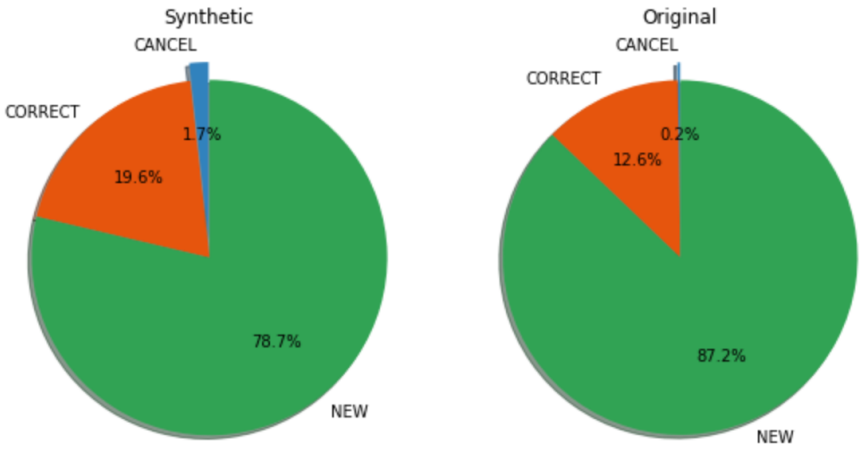

Figure 2: Pie chart of the action variable on the generated and the original datasets

Figure 3 shows the marginal distribution of the `indication_of_collateralization` variable.

Figure 4 shows the marginal distribution of the `price_forming` variable.

Figure 5 compares the marginal distribution of the `rounded_notional_amount` variable.

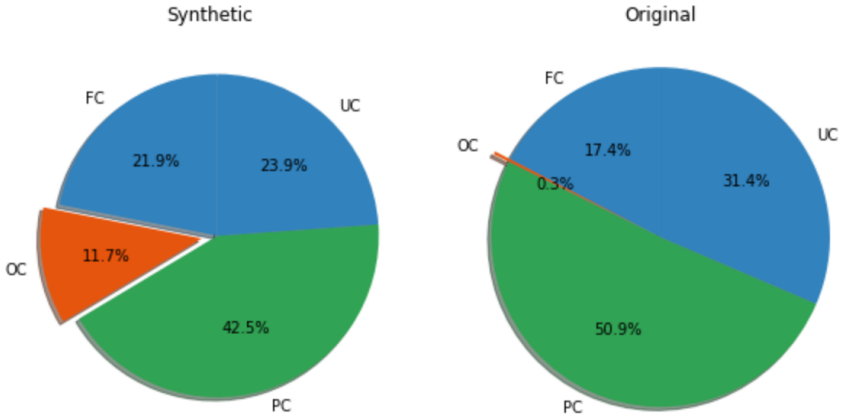

Figure 3: Pie chart of the collateral variable on the generated and the original datasets

Pairwise correlations In addition to the marginal distributions, we also check the pairwise relationship between variables, that is, if the simulated dataset could reconstruct the correlations between variables. In Figure 6, red cells denote positive correlations between variables, while blue cells denote negative correlations. We can see that SeqGAN is able to reconstruct the directions of 15 out of 21 pairwise correlations.

Stability of machine learning algorithms In reality, after the new dataset is generated, we train various machine learning models. The generated new dataset is the training set, and the real dataset becomes the test set. Ideally, we would observe that the accuracies of the models trained on the synthetic data should remain roughly the same on the real dataset. In the model selection setting, however, a weaker sense of stability is required, that is, we would expect that the better performing model on the generated dataset should still perform better in the original dataset. To measure this kind of stability and to check whether the simulated dataset recovers the joint distribution of all variables, we introduce the idea of Synthetic Ranking Agreement (SRA) from [12].

Formally, let $\mathcal{A}_1, \ldots, \mathcal{A}_k$ be the k models considered for selection. For each dataset $\mathcal{D}$, we split it into training set $\mathcal{D}_1$ and testing set $\mathcal{D}_2$. Let m be a performance metric, e.g., accuracy, and let $\mathcal{D}^G$ be the synthetic dataset generated by the generator G. Then we would expect the following relationship: if for two models $\mathcal{A}_i$ and $\mathcal{A}_j$, $1 \leq i, j \leq k$, $\mathcal{A}_i$ performs better than $\mathcal{A}_j$ on the synthetic dataset, that is,

$$m(\mathcal{A}_i(\mathcal{D}_1^G), \mathcal{D}_2^G) < m(\mathcal{A}_j(\mathcal{D}_1^G), \mathcal{D}_2^G), \tag{3}$$

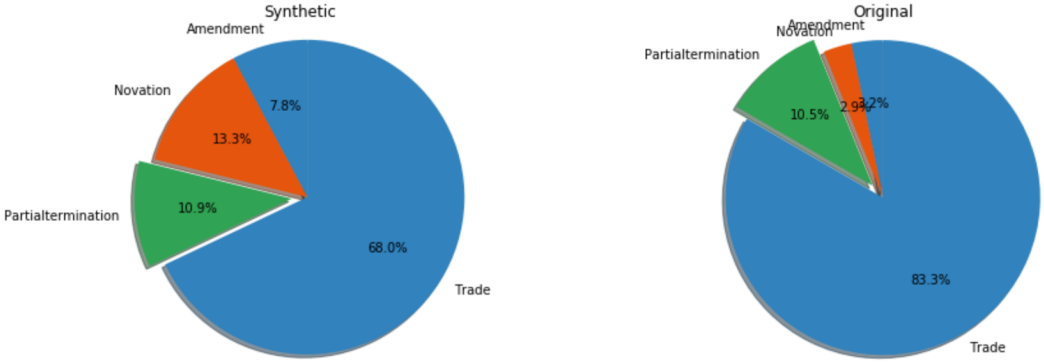

Figure 4: Pie chart of the price forming continuation variable on the generated and the original datasets

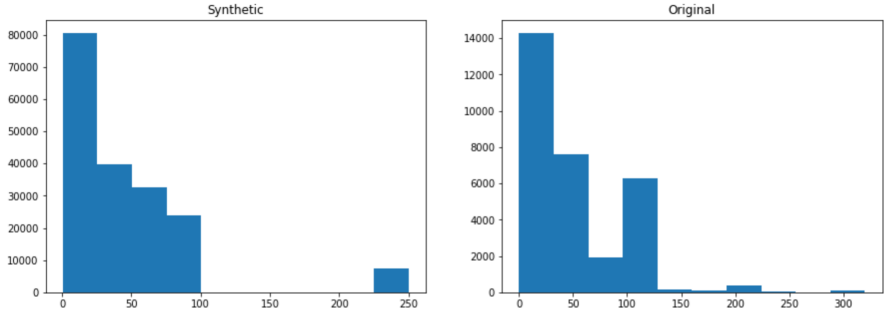

Figure 5: Histograms of the rounded notional amount variable on the generated and the original datasets

then it also performs better on the original dataset, that is

$$m(\mathcal{A}_i(\mathcal{D}_1), \mathcal{D}_2) < m(\mathcal{A}_j(\mathcal{D}_1), \mathcal{D}_2). \tag{4}$$

To come up with a metric for a generator G, the synthetic ranking agreement score is defined as

$$\mathrm{SRA}(G) = \frac{1}{k(k-1)} \sum_{i=1}^{k} \sum_{j \neq i} 1_{((R_i - R_j)(S_i - S_j) > 0)}. \tag{5}$$

where $R_i = m(\mathcal{A}_i(\mathcal{D}_1, \mathcal{D}_2)$, $S_i = m(\mathcal{A}_i(\mathcal{D}_1^G, \mathcal{D}_2^G)$.

In this example, we consider the following 5 machine learning models, which are all very popular in the data science community:

1. Random Forest,
2. Logistic Regression,

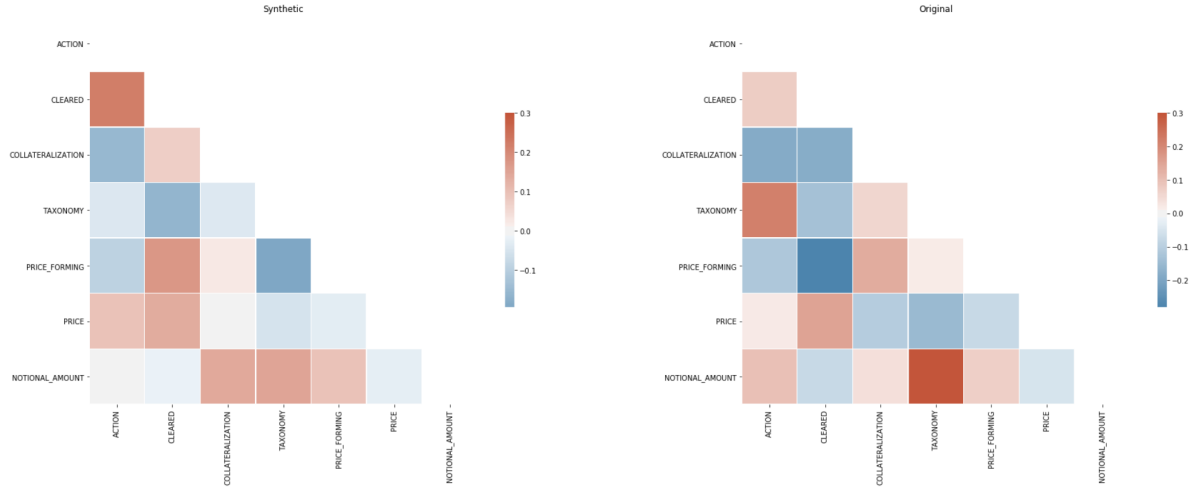

Figure 6: Heatmaps of the pairwise correlation coefficients on the generated and the original datasets

3. Linear Support Vector Machine (SVM),
4. Multinomial Naive Bayes,
5. Gaussian Naive Bayes.

The models are all trained directly out-of-the-box from the Scikit-learn package, with default hyperparameters and configurations. For each variable in the dataset, we use it as the label and the rest of the variables as features. We split the samples into 80% for training and 20% for testing. Table 1 shows the model ranks for the `rank` variable. The SRA score is 0.90. Random Forest is the best model. The rest are not as good but the performances on both datasets are similar.

	Random Forest	Logistic Reg.	Linear SVM	Multinomial NB	Gaussian NB
Original	0.872	0.693	0.728	0.653	0.664
Synthetic	0.968	0.766	0.752	0.553	0.682

Table 1: Accuracy of models predicting the cleared variable on the original and the synthetic datasets

When predicting the `price_formulation` variable using other variables, the results are shown in Table 2. The SRA score is 0.90. Random forest remains the best.

When we predict the price of CDS using other variables, the models yield the following results in Table 3. The SRA score is 0.90.

	Random Forest	Logistic Reg.	Linear SVM	Multinomial NB	Gaussian NB
Original	0.901	0.857	0.823	0.784	0.653
Synthetic	0.916	0.752	0.753	0.586	0.512

Table 2: Accuracy of models predicting the price_formulation variable on the original and the synthetic datasets

	Random Forest	Logistic Reg.	Linear SVM	Multinomial NB	Gaussian NB
Original	0.941	0.929	0.929	0.789	0.867
Synthetic	0.958	0.697	0.623	0.545	0.591

Table 3: Accuracy of models predicting the price variable on the original and the synthetic datasets

From the results above, we conclude that the best performing model among all 5, the random forest, has similar performance on the original dataset and the synthetic one. The other 4 models, on the other hand, could have different predictive powers on both datasets, but their relative rankings stay the same. This shows that our synthetic dataset is capable of running model selection tasks.

5 Limitations and Future Works

In this example, we only consider fixed lengths of order sequences. In fact, it could be better if the length of order sequences are dynamically determined, based on factors such as densities of incoming orders or volatilities of the market.

To ensure a finite action space, we divide the range of the continuous features into bins. Although this is understandable since the variables themselves are truncated at the beginning, the number and sizes of bins are arbitrary for now.

Finally, the rollout policy for the discriminator is chosen to be the current best generator, whose value is slow to evaluate. To improve training speed, the rollout policy could be chosen as a separate model with less complexity.

6 Conclusion

In this report, we show the following three facts:

- The Sequential GAN model is capable to learn the underlying distribution of the CDX dataset, and simulate new order records based on these information;
- The synthetic dataset, the marginal distributions and pairwise correlations among features are well-kept;
- Besides, machine learning model selections done on the new dataset are consistent with the original one.

References

[1] Antreas Antoniou, Amos Storkey, and Harrison Edwards. "Data augmentation generative adversarial networks". In: *arXiv preprint arXiv:1711.04340* (2017).

[2] Daniele Bianchi, Matthias Büchner, and Andrea Tamoni. "Bond risk premia with machine learning". In: *USC-INET Research Paper* 19-11 (2019).

[3] Christopher Bowles, Liang Chen, Ricardo Guerrero, Paul Bentley, Roger Gunn, Alexander Hammers, David Alexander Dickie, Maria Valdés Hernández, Joanna Wardlaw, and Daniel Rueckert. "GAN augmentation: augmenting training data using generative adversarial networks". In: *arXiv preprint arXiv:1810.10863* (2018).

[4] Luyang Chen, Markus Pelger, and Jason Zhu. "Deep learning in asset pricing". In: *Available at SSRN 3350138* (2019).

[5] Guanhao Feng, Jingyu He, and Nicholas G Polson. "Deep learning for predicting asset returns". In: *arXiv preprint arXiv:1804.09314* (2018).

[6] Maayan Frid-Adar, Eyal Klang, Michal Amitai, Jacob Goldberger, and Hayit Greenspan. "Synthetic data augmentation using GAN for improved liver lesion classification". In: *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*. IEEE. 2018, pp. 289–293.

[7] Shihao Gu, Bryan T Kelly, and Dacheng Xiu. "Autoencoder Asset Pricing Models". In: *Available at SSRN* (2019).

[8] Shihao Gu, Bryan Kelly, and Dacheng Xiu. *Empirical asset pricing via machine learning*. Tech. rep. National Bureau of Economic Research, 2018.

[9] Blanka Horvath, Aitor Muguruza, and Mehdi Tomas. "Deep Learning Volatility". In: *Available at SSRN 3322085* (2019).

[10] John C Hull and Alan D White. "Valuation of a CDO and an n-th to default CDS without Monte Carlo simulation". In: *The Journal of Derivatives* 12.2 (2004), pp. 8–23.

[11] John C Hull and Alan D White. "Valuing credit default swaps I: No counterparty default risk". In: *The Journal of Derivatives* 8.1 (2000), pp. 29–40.

[12] James Jordon, Jinsung Yoon, and Mihaela van der Schaar. "Measuring the quality of Synthetic data for use in competitions". In: *arXiv preprint arXiv:1806.11345* (2018).

[13] Tarja Joro, Anne R Niu, and Paul Na. "A simulation-based First-to-Default (FTD) Credit Default Swap (CDS) pricing approach under jump-diffusion". In: *Proceedings of the 36th Conference on Winter Simulation*. Winter Simulation Conference. 2004, pp. 1632–1636.

[14] Giovanni Mariani, Florian Scheidegger, Roxana Istrate, Costas Bekas, and Cristiano Malossi. "Bagan: Data augmentation with balancing gan". In: *arXiv preprint arXiv:1803.09655* (2018).

[15] Marcial Messmer. “Deep learning and the cross-section of expected returns”. In: *Available at SSRN 3081555* (2017).

[16] Dong Nie, Roger Trullo, Jun Lian, Caroline Petitjean, Su Ruan, Qian Wang, and Dinggang Shen. “Medical image synthesis with context-aware generative adversarial networks”. In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2017, pp. 417–425.

[17] Madhvi Sethi, Parthiv Thakkar, and Zahid M Jamal. “A Simulation Model for Pricing the Spread in a Credit Default Swap: Application and Analysis”. In: *SDMIMD Journal of Management* 9.2 (2018), pp. 9–18.

[18] Hoo-Chang Shin, Neil A Tenenholtz, Jameson K Rogers, Christopher G Schwarz, Matthew L Senjem, Jeffrey L Gunter, Katherine P Andriole, and Mark Michalski. “Medical image synthesis for data augmentation and anonymization using generative adversarial networks”. In: *International Workshop on Simulation and Synthesis in Medical Imaging*. Springer. 2018, pp. 1–11.

[19] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. “Seqgan: Sequence generative adversarial nets with policy gradient”. In: *Thirty-First AAAI Conference on Artificial Intelligence*. 2017.

[20] Xinyue Zhu, Yifan Liu, Zengchang Qin, and Jiahong Li. “Data augmentation in emotion classification using generative adversarial networks”. In: *arXiv preprint arXiv:1711.00648* (2017).