Scenario Generation and Data Augmentation

Raymond Luo, Ziheng Chen, Xiaobin Ou, Elaine Huang

August 16, 2021

1 Introduction

There is a need to generate additional financial data while keeping the statistical features of original data for the following reasons:

1) With the development of deep learning, the need for adequate data is unprecedented. More data is needed to get the deep learning algorithms full trained and to tackle the data imbalance problem.

2) For the purpose of risk assessment, we need to generate severe but plausible scenarios. However, traditional risk assessment like VAR cannot capture the whole scenario and wasted the majority of information contained in original data.

3) Data anonymization and preserving of data privacy. For data have privacy concerns, such as illness information. We would like to extract information out of these data and keep patients' private information at the same time.

Hence, we propose a new method to generate synthetic financial data while keeping statistical features of original data by combing Hidden Markov Models and generative models. We use HMM model [5] to identify different market regimes and trained generative algorithms conditioned on different regimes. GAN and VAE are both used to generate synthetic data. The degree of "how real" of these synthetic data was greatly improved, compared with training without considering market regimes. Evaluation was formed as a classification problem. KNN and Random Forest were used as classifiers and trained on real financial assets return series. The output probability of using classifiers to predict synthetic data was used as the degree of "how real" these synthetic data was.

2 Methodology

We proceed our investigation in the following manner. We first recreate the results given by two major toolsets for scenario generation: the variational autoencoder (VAE) and the time series generative adversarial network (timeGAN) models. This procedure involves identifying the code syntax used by the respective authors of the models as well as updating the code for our own usage - some of the machine learning and tensorflow packages have deprecated due to the rapid development of deep learning packages. We then extrapolate the usage of the code onto our own choice of initial data. As equity indices are both abundant and diverse, we proceed with a selection of 10 years of various daily and weekly equity index data. For theoretical considerations

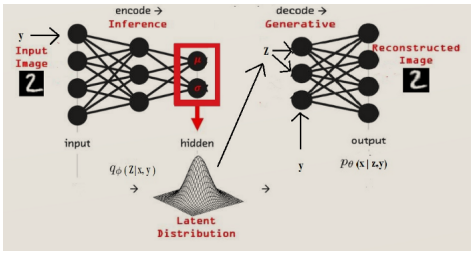
of the financial behavior of stock returns under the efficient market hypothesis and the crucial rate of convergence for our deep learning models, we focus on normalized stock returns in data processing for the entirety of our investigation.

After establishing baseline models, we proceed in modifications in two ways. Firstly, we combine the two models in sequence to varying degrees in order to see whether our results are for better or worse. Secondly, we modify parameters in the VAE and timeGAN models to see if we have increased success in our results. These modifications include but are not limited to adding discrete jump processes to the baseline GAN data as noise or observing regime shift periods in the underlying data to add market shift imitations. For VAE data this could include adding variation to the path-signature of generated data paths to add noise.

3 Variational Autoeoncoders

3.1 Base Model Results

Variational Autoencoders (VAEs) is a deep learning generative model involving two sequences: an encoder and a decoder. The encoder projects the sample space onto a latent space (or encoded space) through a process of reparametrization. More specifically, the encoder sequence produces an output of a mean μ and variance σ used with a random gaussian sampling ϵ to create a latent representation of $z = \mu + \epsilon * exp(\sigma/2) \sim N(0, 1)$. This latent representation is then decoded back to the initial observed space. The optimization of this generative model involves a reconstruction loss from the decoder and loss from the initial encoding (the optimization of the combination of these objective functions ultimately determine the success of the model). The diagram below demonstrates the pipeline process for the VAE model:



VAEs have historically found less success in comparison to GAN models due to the limited structure they impose on the generated data. One paper that has particularly shown great promise proposes the use of Conditional Variational Autoencoders (cVAE) on path-signatures [3]. cVAEs are VAE models where the input data is coupled with labels, so that the latent and realized distributions are conditioned on labelling.

The idea for Path-Signatures follows from Stochastic Analysis and Algebraic Topology - We consider iterated coordinate integrals from the free nilpotent group embedded in the truncated free tensor algebra as a natural "linear" basis for continuous paths. The signatures for the paper are defined by:

Definition 3.1 (Signature)**.** Let $X : [0, T] \to^d$ d be a continuous path of bounded variation. Then the signature of X is defined by the sequence of iterated integrals given by:

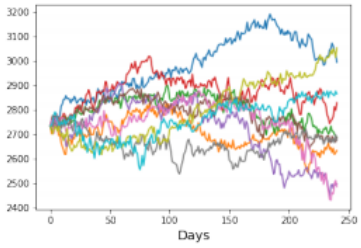
$$\mathbf{X}_T^{<\infty} := (1, \mathbf{X}_T^1, \cdots, \mathbf{X}_T^n, \cdots)$$

where

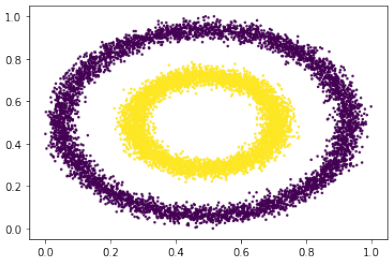
$$\mathbf{X}_T^n := \int_{0<u_1<\cdots<u_k<T} dX_{u_1} \otimes \cdots \otimes dX_{u_k} \in (^d)^{\otimes n}$$

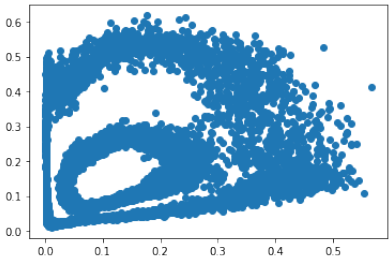
.

This definition is a generalization of smooth paths from stochastic analysis with $\otimes$ being the tensor product. The key to this approach is to consider the signature a feature for the training of the cVAE model and the integral of the tensor product has a *postulated* value of $\mathbf{X}_T^n$.



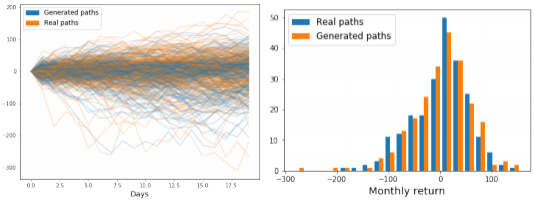
Our first step aims to replicate the model proposed in the paper using an updated version of Tensorflow (the paper uses deprecated code from Tensor v.1.0) and to extrapolate the mathematical aspects to extend to our own data sets. While we adopted the same parametrizations as the original paper - as 0.02 ratio between reconstruction loss and encoding loss, 10000 epochs, 0.03 learning rate etc... - there's been a slight difference in results. We see this from our attempt at a test set to recreate circle data:



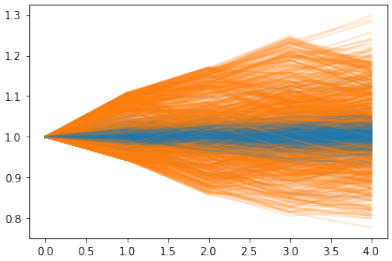


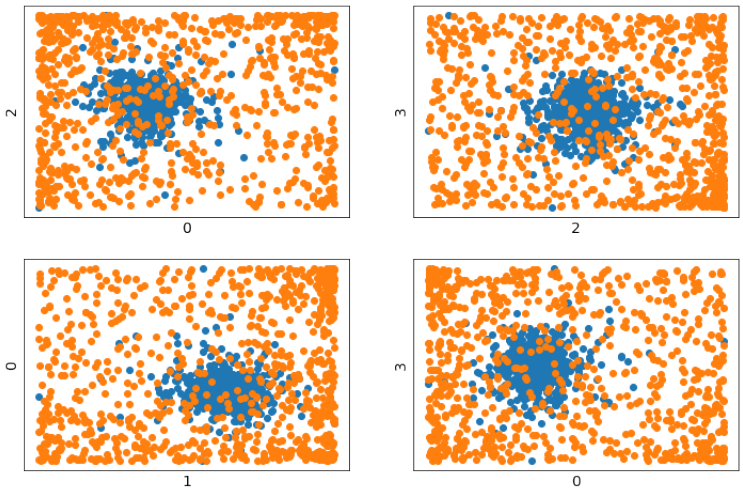
We can see that there's a deformation in our reconstruction which isn't found in the original paper. This could be looked into more in the future with some hyperparameter tuning.

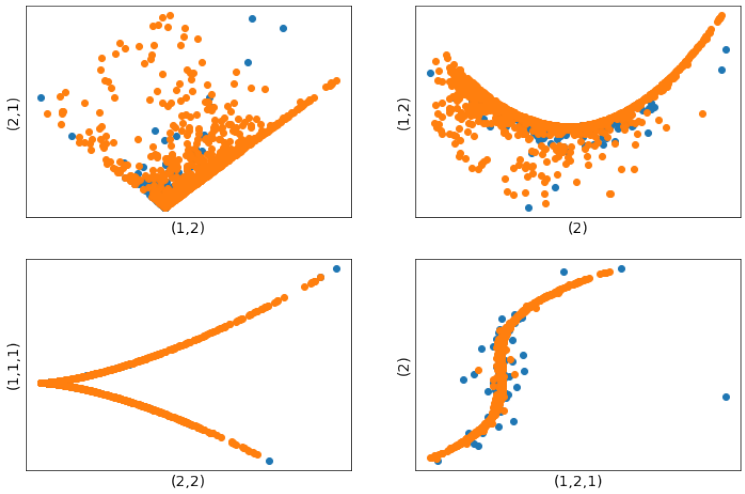
With our have recreated the results from the paper as follows:



These results didn't make much sense as returns could not reasonably go to -300 with non-negative prices. Following conventional theory that stock prices following a geometric brownian motion, we thought it would make more sense to apply the analysis to the log-return of asset prices. We get the following results:



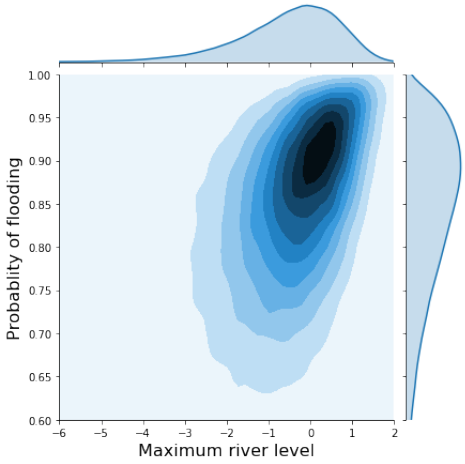




4 Gaussian Copula

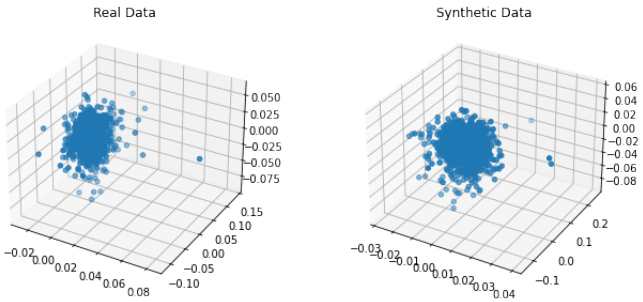
A concern with our cVAE model is that it generates an individual possible asset path meaning it fails to consider cross-asset correlations. On the other hand, VAE models prove to be more useful in extreme scenarios when compared to a very straightforward approach of random sampling from the historical empirical distribution of the data. An approach that attempts to reconcile both ideas is using gaussian copula to capture the multivariate-joint distribution of cross-asset pairs copula.

In probability theory, a copula is a multivariate cumulative distribution function over the hypercube - meaning the marginal probability distribution of each component variable is uniform over [0, 1]. Copulas are extremely useful in describing the dependence (inter-correlation) between random variables and it follows that they provide useful sampling that captures the relation between variables[6].

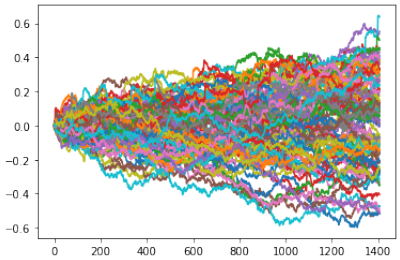


We use the copula model as a base model that captures asset correlation very well and compare it to the results we get from the VAE model and GAN model. To demonstrate, we have the

following generative results for the gaussian copula across cross-asset portfolio composing of WTI crude oil futures, GBPUSD, and AAPL:



We simulate the 1000 path for GBP using the copula to get:



5 Generative Adversarial Networks

The generative adversarial network was originally and mainly used for image generation. It is an generative modeling using deep learning methods, such as convolutional neural networks, recurrent neural networks or etc. Generative modeling is an unsupervised learning task in machine learning that involves automatically discovering and learning the patterns from the input data in such a way that the model can be used to output new examples that plausibly could have been drawn from the original dataset. GAN is a clever way of training a generative model by framing the problem as a supervised learning problem with two sub-models: the generator model and the discriminator model. The generator model is used to generate new examples, and the discriminator model tries to classify examples as either real (from the domain) or fake (generated). The two models are trained together in the way that one side's gain is equivalent to another's loss, until the discriminator model is fooled about half the time, meaning the generator model is generating plausible examples.

Since the stock return data has time dependency, we apply it to the time-series domain, here choose Time-series Generative Adversarial Networks.

We implement the Time-series Generative Adversarial Networks (TimeGAN) [2], which takes temporal dynamics of sequences into consideration, combining the unsupervised GAN framework with supervised training via adding an embedding space. In other words, in addition to minimizing the unsupervised loss, we also want to capture the stepwise dynamics in the data and minimize the corresponding supervised loss. Suppose we have static features ($\mathbf{S}$) and temporal features ($\mathbf{X}$) in our data set. And let $\mathcal{S}, \mathcal{X}$ be the vector spaces of static features and temporal features ($\mathbf{S} \in \mathcal{S}, \mathbf{X} \in \mathcal{X}$), respectively. We aim to approximate the joint distribution of $(\mathbf{S}, \mathbf{X}_{1:T})$, $p(\mathbf{S}, \mathbf{X}_{1:T})$, which may

be difficult under the standard GAN framework. Therefore, we break down the objective into a sequence of stepwise objectives, and approximate $p(\mathbf{X}_t|\mathbf{S}, \mathbf{X}_{1:t-1})$ for each time t by

$$p(\mathbf{S}, \mathbf{X}_{1:T}) = p(\mathbf{S}) \prod p(\mathbf{X}_t|\mathbf{S}, \mathbf{X}_{1:t-1})$$

Thus we transform our goal from global,

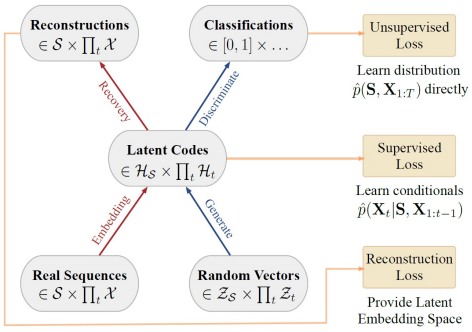
$$\min_{\hat{p}} D(p(\mathbf{S}, \mathbf{X}_{1:T})||\hat{p}(\mathbf{S}, \mathbf{X}_{1:T}))$$

to local,

$$\min_{\hat{p}} D(p(\mathbf{X}_t|\mathbf{S}, \mathbf{X}_{1:t-1})||\hat{p}(\mathbf{X}_t|\mathbf{S}, \mathbf{X}_{1:t-1}))$$

where $\hat{p}(\cdot)$ is the learned density and D is a distance measure. The local minimization can be better achieved since it only depends on ground-truth sequences.

The TimeGAN network consists of four steps: embedding function, recovery function, sequence generator and sequence discriminator. Embedding and recovery functions are the first two mappings between feature and latent space, where the embedding function takes static features ($\mathcal{S}$) and temporal features ($\prod_t \mathcal{X}$) to their latent spaces $\mathcal{H}_\mathcal{S}$ and $\prod_t \mathcal{H}_\mathcal{X}$, respectively, and the recovery function, on the contrary, takes static and temporal codes from the spaces back to features. Sequence generator and discriminator are the latter two networks operating within the latent space, where the generator takes static and temporal random vectors from vector spaces $\mathcal{Z}_\mathcal{S}, \mathcal{Z}_\mathcal{X}$ to the latent space, and the discriminator returns classifications (either real or synthetic data) by receiving static and temporal codes from latent space. The bellow diagram shows the interactions and objectives of the TimeGAN functions.



5.1 Data Processing

In the current stage, we constructed our basic portfolio using different types of assets and then calculated the daily return as our dataset.

Stock: MSFT,AMZN,DIS,JNJ,JPM,AAPL,
WBA,XOM,HES,UNH,PFE

Corporate Bond: CSCO,GS,JNJ

Currency: GBPUSD, CNYJPY

Treasury Bond: TNX,FVX,TVX

Commodity:HG1,CL1,W1,GC1

We used data between the period 2001/01/01 and 2021/06/18. After obtaining returns, we used MinMaxScalar to normalize the returns, and then cut the data by sequence length. Then we mixed data via random permutation to make it similar to independent and identically distributed random variables, then it is ready for model training.

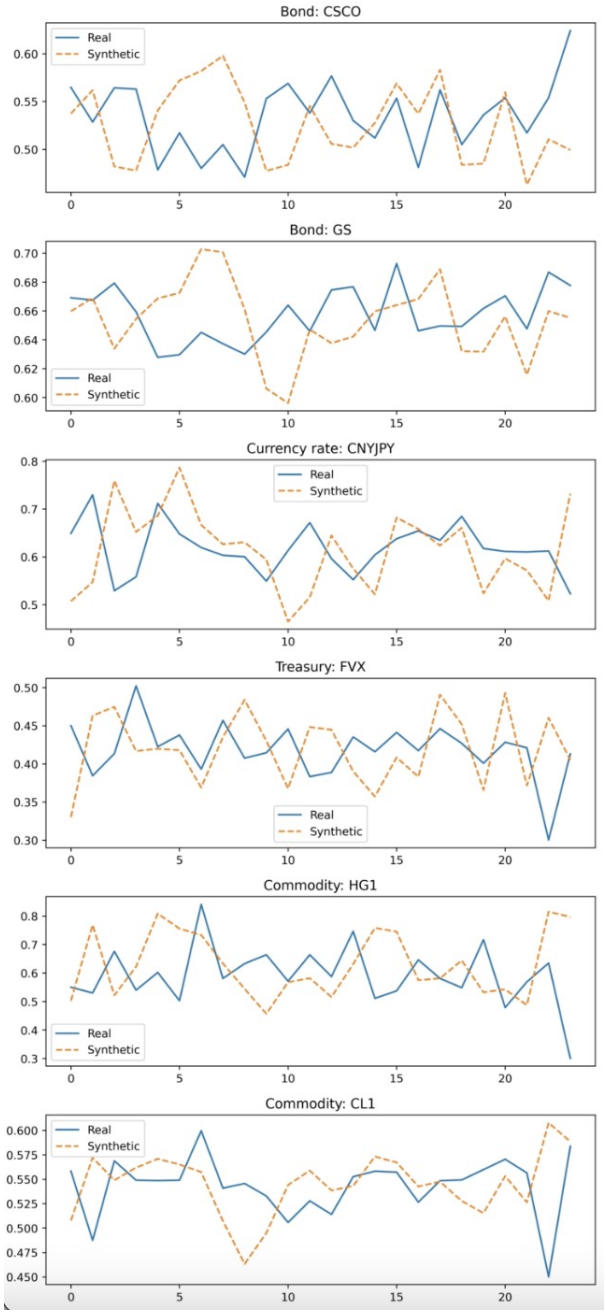
5.2 GAN Model Building

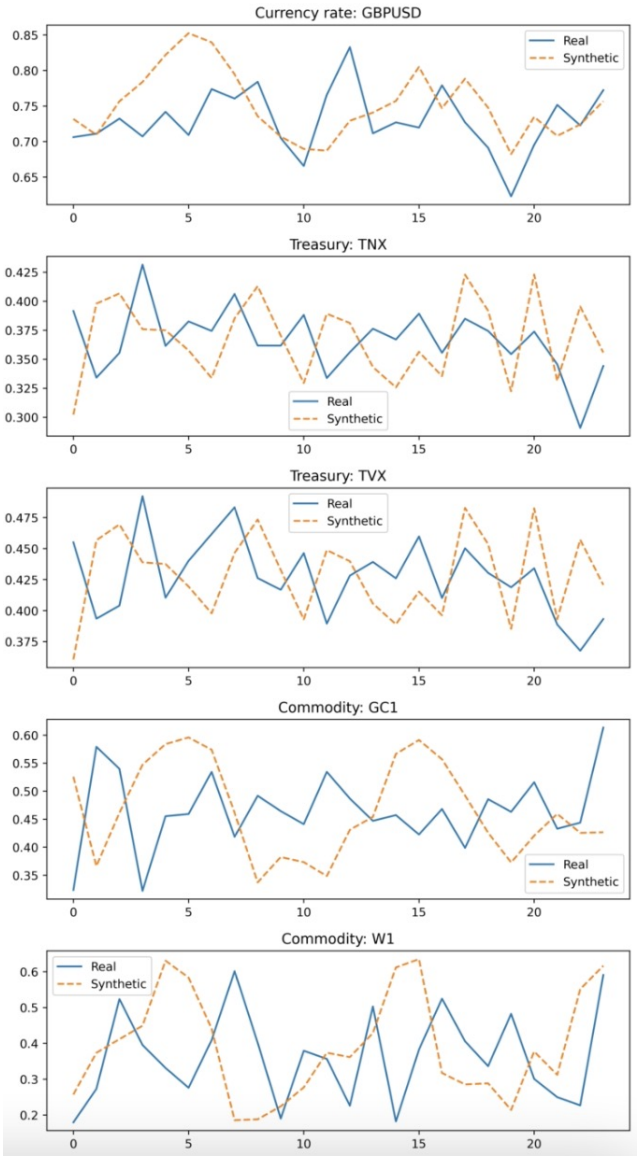
We use TGAN to generate synthetic time-series data, which is based on RNN networks. It is consisted of four elements which are generator, discriminator, embedder and recovery network. In the model, the architectures of each element would be optimized and tailored to the dataset. Here we defined sequence length, number of sequence, hidden dimension, gamma, noise dimension, batch size, log step, and learning rate. After finishing defining model hyperparameters, we trained the timeGAN synthetizer using 50000 train steps. It took around two hours to finish training in the current stage.

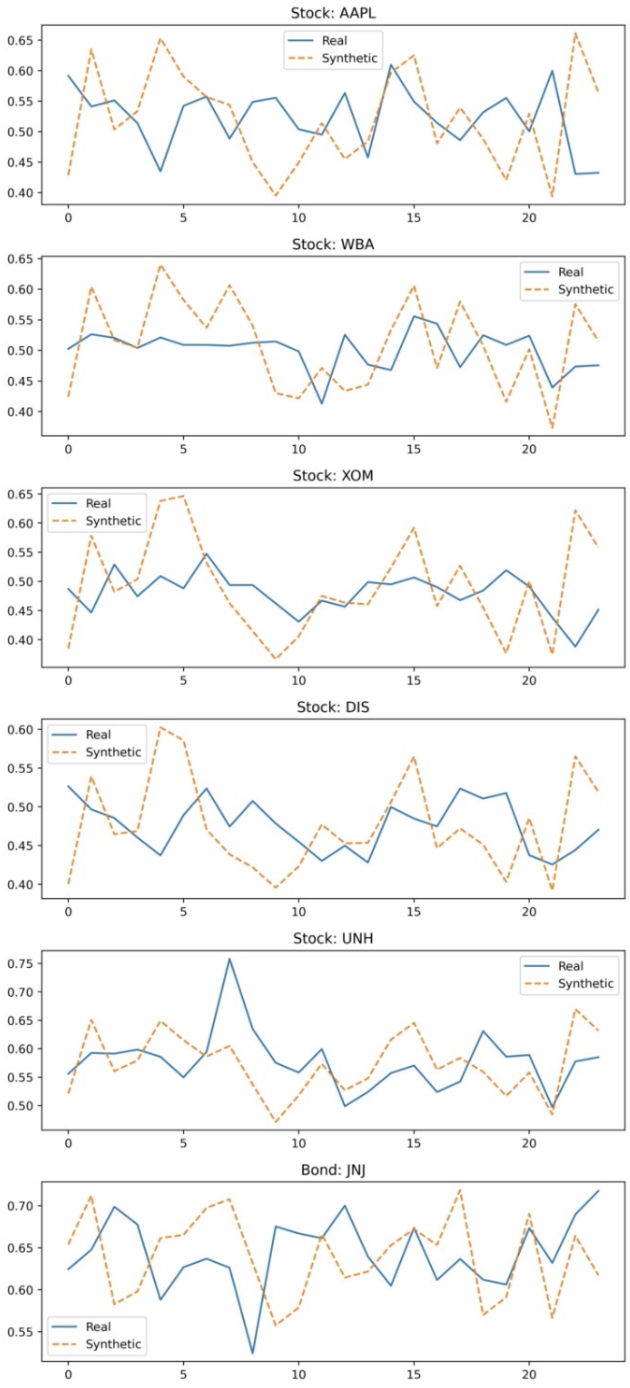
Hyperparameters	value
sequence length	24
number of sequence	6
hidden dimension	24
noise dimension	32
batch size	128
log step	100
gamma	1
learning rate	$5e^{-4}$

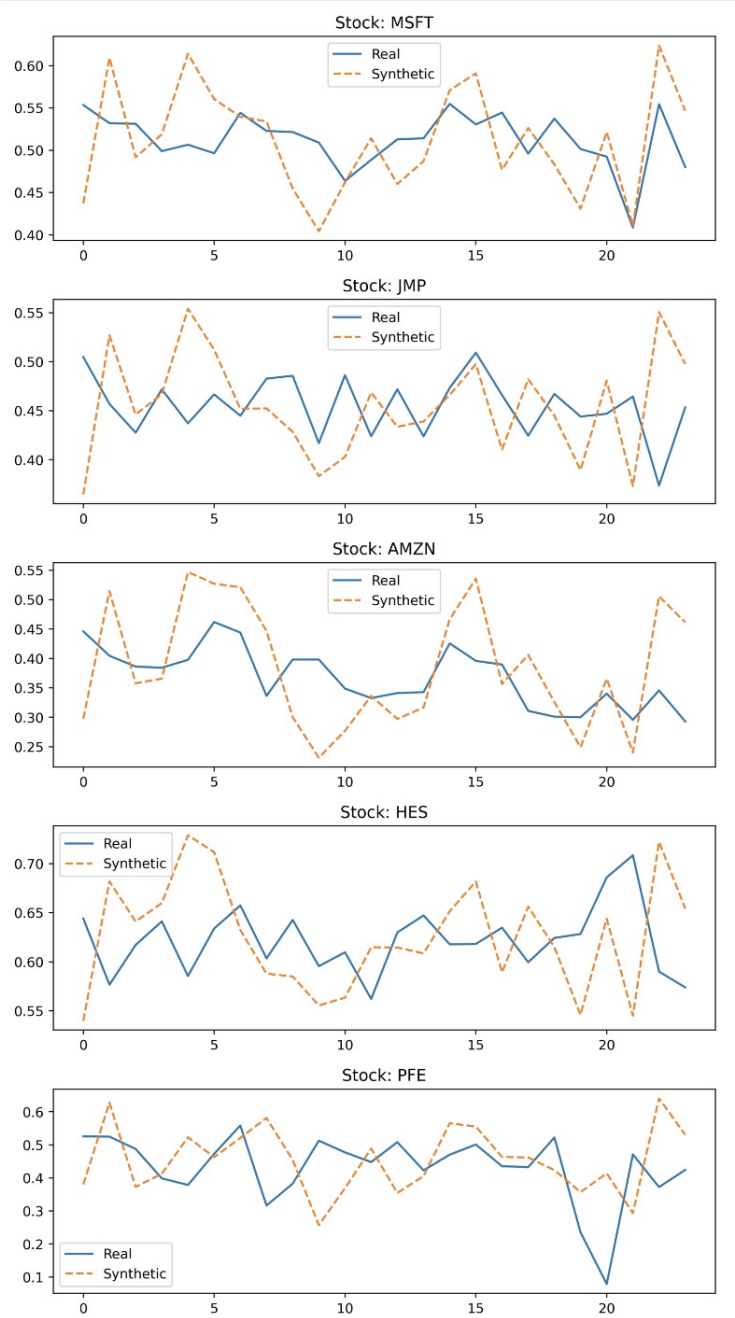
5.3 GAN Model Results

After model training was done, we plotted some generated samples, which included both synthetic and original data standardized with values between [0,1]. The results are shown below.




Currency rate: GBPUSD
Real
Synthetic
Treasury: TNX
Real
Synthetic
Treasury: TVX
Real
Synthetic
Commodity: GC1
Real
Synthetic
Commodity: W1
Real
Synthetic

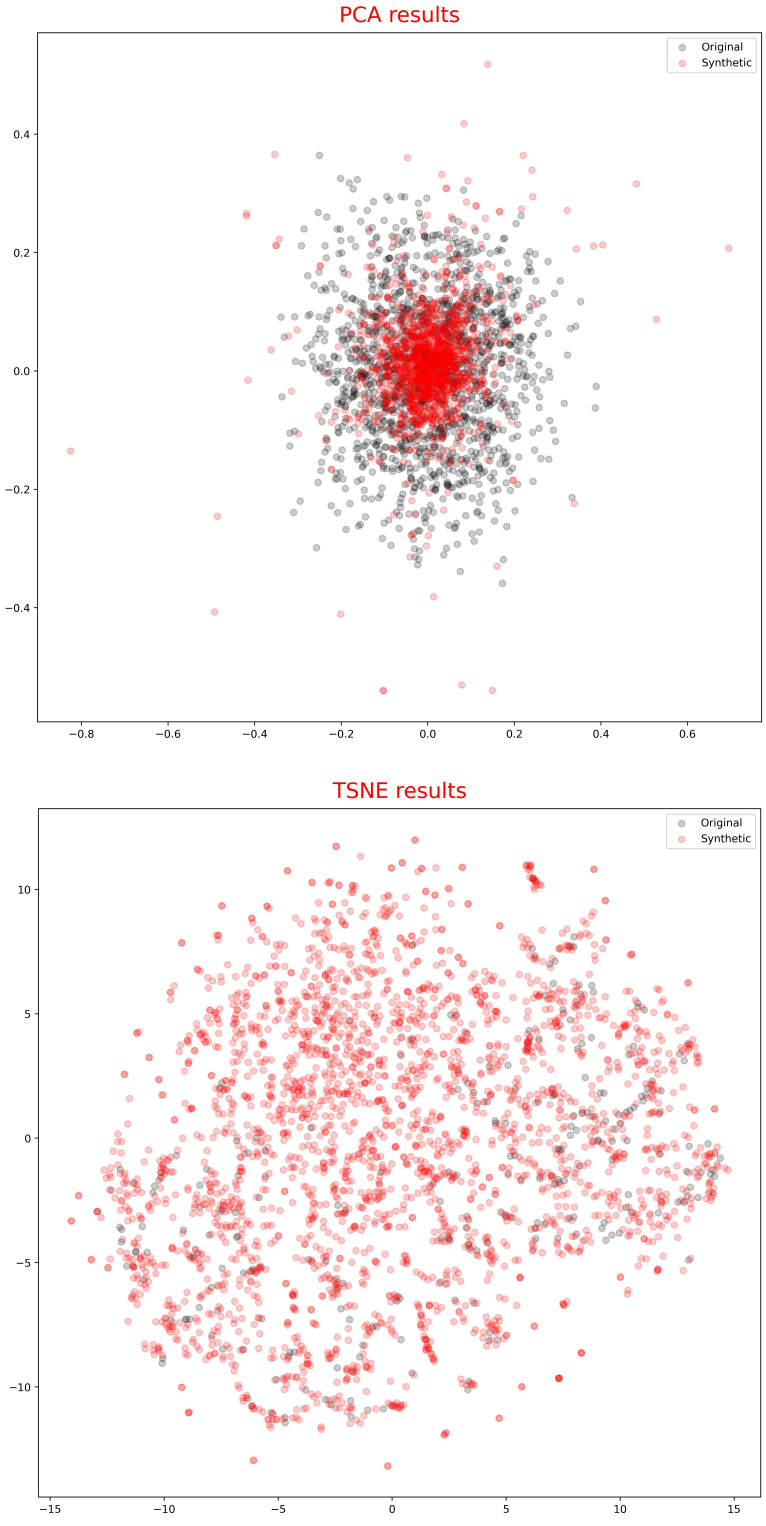

Stock: AAPL
Real
Synthetic
Stock: WBA
Real
Synthetic
Stock: XOM
Real
Synthetic
Stock: DIS
Real
Synthetic
Stock: UNH
Real
Synthetic
Bond: JNJ
Real
Synthetic



5.4 GAN Model Evaluation

To evaluate the generated synthetic data, we used PCA and TSNE. For the sake of comparing them visually, we need the data to be 2-Dimensional. For that reason we used only two components

for both the PCA and TSNE. The results are shown below.



Then we trained on synthetic datset and tested on real dataset to further evaluate the model. We first implemented a simple RNN model for prediction. To prepare the dataset for the regression model, we splited data on train dataset and test dataset, and then we defined s and y for both the

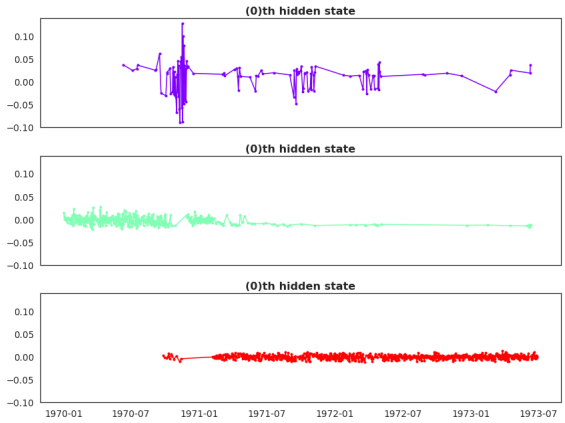
synthetic and real datasets. Finally, we trained the model with both the synthetic data and the real train data with parameters epochs 200, batch size 128, and callbacks to be early stopping. We summarize the metrics and the result is shown below, which includes R-squared score, mean absolute error and mean squared log error.

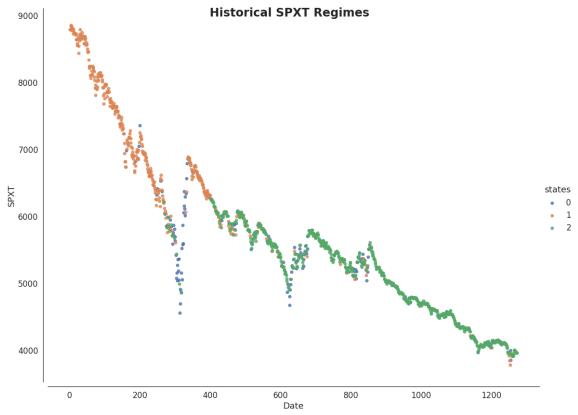
	r2	MAE	MRLE
Real	0.906701	0.012914	0.000629
Synthetic	0.913467	0.018099	0.000715

6 Evaluation of Real Time Series

An initial data analysis that we can provide that is an analysis of the market regimes that the real data goes through. This allows us to understand some interesting perspectives: what is a reasonable length of a market regime, what are the statistical qualities that shift during a market regime, and etc. We attempt to observe these market regimes tentatively with use of Hidden Markov Models (HMMs) with a varying parameter on the number of regimes to segment our historical data into various data regimes. We will continue to conduct a similar analysis on the generated data to see how our synthetic data segments into various regimes. The idea here is that a synthetic data that has few and sparse regime shifts is less representative of realistic data and the possible scenarios in reality.

Alternatively, we can also use the metrics in our market regimes to determine the qualities that our various asset groups (from various styles of equities to other asset classes) that are shared within the groups. This allows us to identify key parameters to hold for future data-augmentation models and which qualities are to be tested for our evaluation. The diagrams below display an basic analysis of one of the training sets in a 20 year period for 3 market regimes.





7 Market Regimes Detection

The behavior of financial markets often modifies abruptly, due to changing periods of government policy, regulatory environment and other macroeconomic effects. Such periods are known as "market regimes". We want to detect such regimes that are impacting the observed asset returns and use GAN model to generate synthetic data for each regime. If the means and variances of time series are stable though a regime, the data generated via dividing regimes should be more realistic than data generated via un-dividing regimes.

The regime detection methodology we use is the Hidden Markov Models (HMMs) [5]. We use the sklearn's GaussianMixture to fit a model that estimates the regimes. Mixture models implement a closely related, unsupervised form of density estimation that utilize the expectation-maximization algorithm to estimate the means and covariances of hidden states (regimes). In this section, we choose one asset from each of the six asset classes (see the table as below) and fit six mixture models. The observable variables we use are daily close price and daily asset returns. We choose 3 states for AAPL stock, TNX treasury and HG1 commodity, and 2 states for CSCO bond and GBPUSD currency (because the time scale of a third state for these two assets is too small for GAN model to train and generate data).

Class	stock	corporate bond	treasury	commodity	currency
Asset	AAPL	CSCO(5 1/2 01/15/2040 Corp)	TNX	HG1(copper)	GBPUSD
States Number	3	2	3	3	2

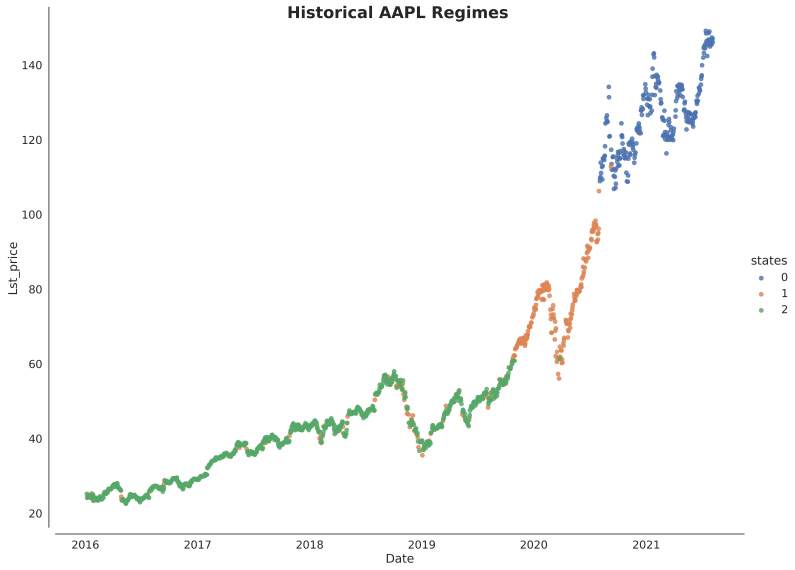
7.1 Stock: AAPL

We classify 3 regimes as High, Neutral and Low Volatility with the goal to first minimize variance of return and then maximize mean return. From the below summary table of means and variances for AAPL hidden states, it appears the 2th hidden state is the low volatility regime with the second largest mean return 0.13% and the smallest variance 0.011%. The 0th hidden state is neutral volatility regime with the largest mean return 0.16% and the second largest variance 0.039%. The 1th hidden state is the high volatility regime with the smallest mean return 0.093%

and the largest variance 0.096%.

		close price	daily return
0th hidden state	mean	126.4121	0.1619%
	var	103.1365	0.03887%
1th hidden state	mean	62.3145	0.09257%
	var	414.4906	0.09610%
2th hidden state	mean	39.2150	0.1282%
	var	105.7603	0.01120%

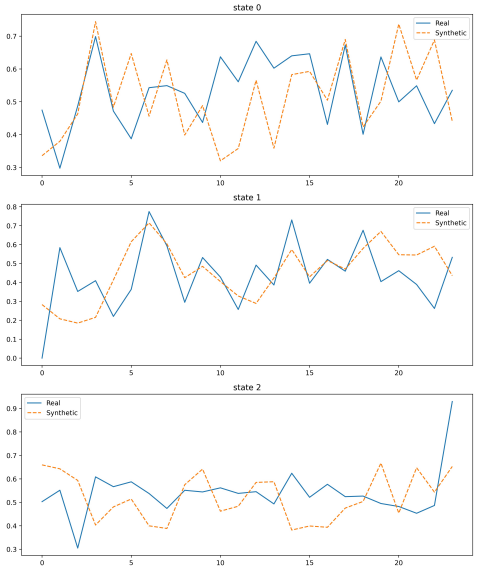
From 2018 to 2019, the regime of AAPL switches from low volatility to high volatility, as we can see state 1 points and state 2 points overlaid in the below figure of historical AAPL regimes, with prices plummeting 9.96% in 2018 to the worst low Apple has seen since 2017. A great cause is the trade tensions between the united states and China, sales numbers decreased mostly in China region with price increasing and cheaper battery replacements.



From the above figure of historical AAPL regimes we can see there's again a big drop in May 2019, as stock is increasingly hit by worsening US-China trade relations. Another cause of high volatility in state 2 is the tech industry feeling the affects of COVID-19 in 2020. Since 2021, high volatility has flattened to neutral volatility, with Apple gradually climbing back up and hitting an all-time high as stores reopen and high demand for new devices and new generations to adapt to work-at-home conditions.

Next we train the GAN model for each market regime. we see from the figure of $(b) AAPL: Real\ vs\ Synthetic\ data\ for\ states$ that in sate 1 and 2, there are bigger gaps between the real and synthetic data than that in state 0.

(a) AAPL: Hidden states



(b) AAPL: Real vs Synthetic data for states

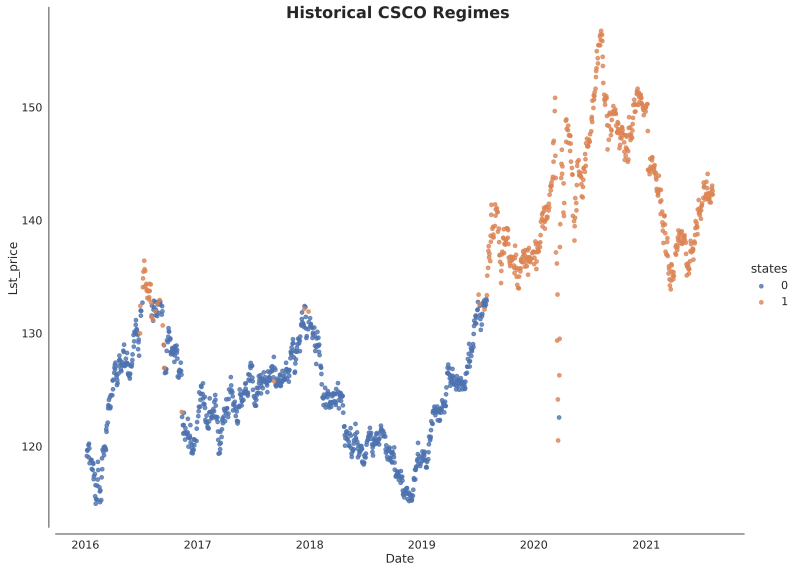
7.2 Corporate bond: CSCO

The observable bond we choose is Cisco's $51/201/15/2040$ $Corp$. We classify 2 regimes as high volatility and low volatility. The 0th hidden state is the low volatility with larger mean return 0.019% and lower variance 0.0029%, and the 1th hidden state is the high volatility with smaller mean return 0.0032% and higher variance 0.0085%.

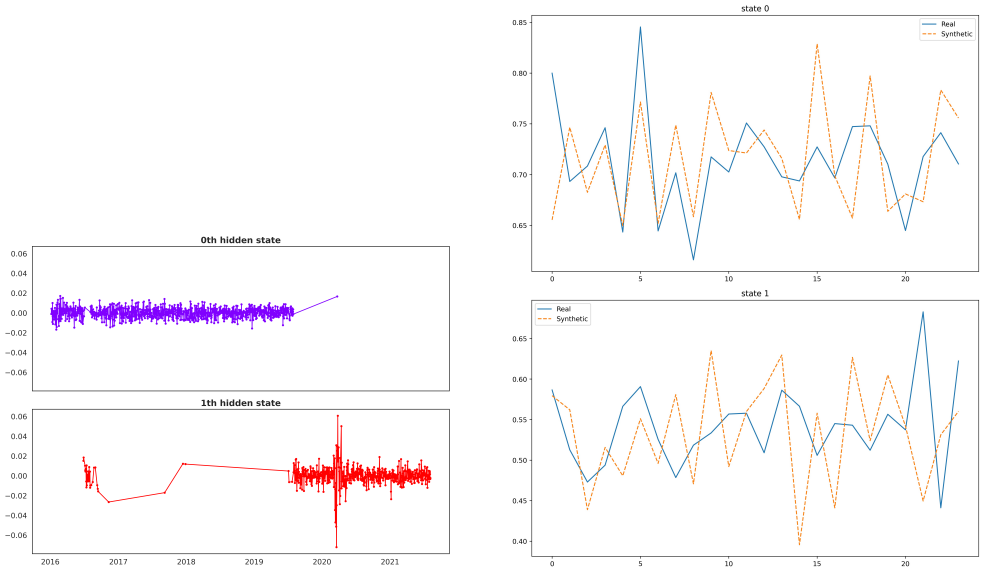
		close price	daily return
0th hidden state	mean	124.0847	0.01922%
	var	19.5088	0.002851%
1th hidden state	mean	141.0764	0.003169%
	var	44.9253	0.008541%

As we known, the U.S. bond market rallies impressively during much of 2019 as the Federal Reserve cut interest rates, but lower interest rates forecast lower returns for bonds in the future.

This might be the meaning of 1th hidden state from 2020 to the near future.



After training the GAN model for the two market regimes, we get the clear picture of real vs synthetic data which shows better accuracy in the low volatility.



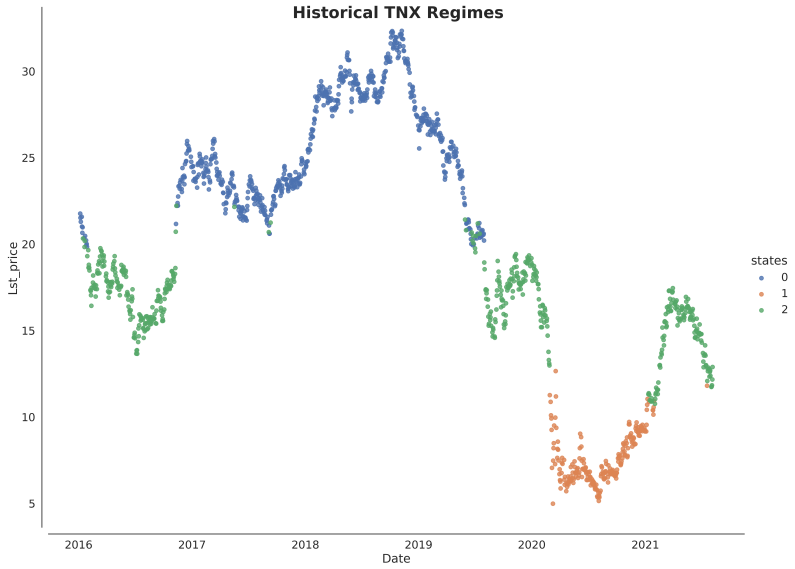
(a) Hidden states (b) Real vs Synthetic data for states

7.3 Treasury: TNX

The summary table of TNX index's hidden states shows all three regimes having negative mean returns. The 1th hidden state is the high volatility with the highest variance 0.59%, the 2th hidden state is the neutral volatility with the second lowest variance 0.79% and the 0th state is the

low volatility with the lowest variance 0.019%.

		close price	daily return
0th hidden state	mean	25.5262	-0.008252%
	var	1.1521	0.01855%
1th hidden state	mean	7.6735	-0.1032%
	var	2.3384	0.05903%
2th hidden state	mean	1.6674	-0.05560%
	var	4.9881	0.07887%



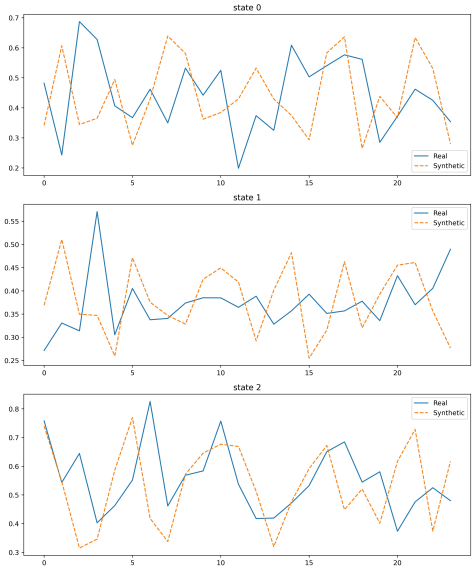
The historical TNX regimes and the Real vs Synthetic data visualization (figure (b) on the right below) show that none of the regimes can be observed to generate good synthetic data. Whether it is reasonable for treasury to generate synthetic data by dividing regimes needed to be further evaluated.

7.4 Commodity: Copper(HG1)

From the below table, the 1th hidden state has the lowest mean return 0.039% and the smallest variance 0.00015 at the same time. But it has extremely high volatility of price 0.056 due to market uncertainties created by US economic and foreign policies, including the trade war between the US and China, and the COVID -19 and its impact on the global economy.

The 0th and 2th state have the neutral volatility at 0.0002 level, while the mean return in 2th state (0.16%) is much larger than that in 0th state (0.03%). Because the time scale of 0th state, mainly year 2016, was a recovering year for copper affected by the world stock markets dropping substantially in 2015. While from 2020 to 2021, expectations for copper drive price higher and higher: optimism around the trade deal with countries reaffirm their commitment to an agreement;

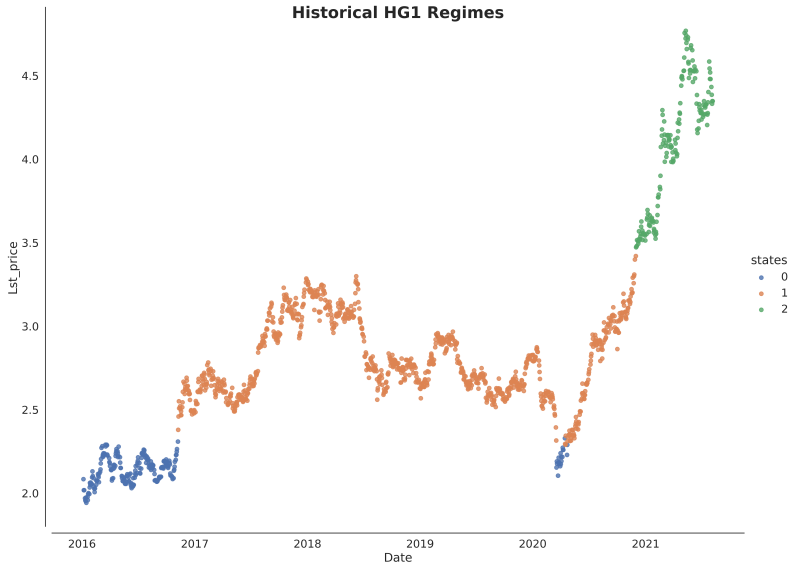
(a) TNX: Hidden states



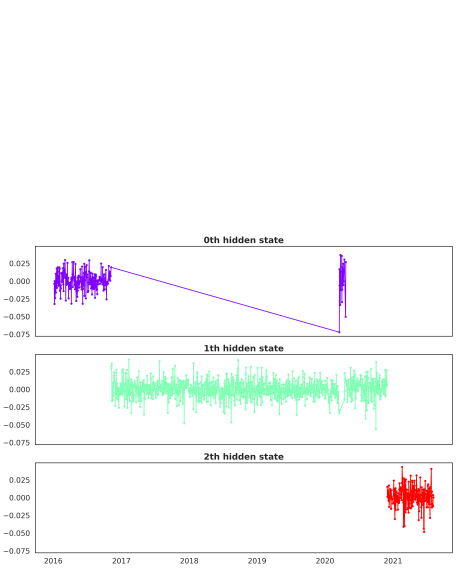
(b) TNX: Real vs Synthetic data for states

Chinese stimulus to support copper demand in combination with an expected global economic recovery; demand for copper in green energy.

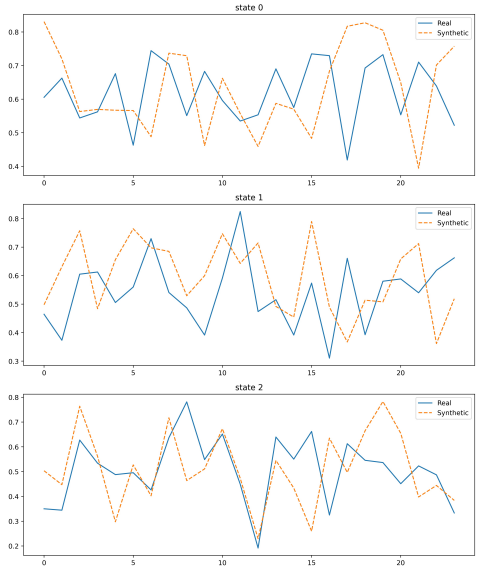
		close price	daily return
0th hidden state	mean	2.1443	0.03332%
	var	0.006427	0.0207%
1th hidden state	mean	2.7957	0.03903%
	var	0.05607	0.01491%
2th hidden state	mean	4.0767	0.1585%
	var	0.1690	0.0229%



Next we train the GAN model for each market regime. we see from the lower right figure that real and synthetic data are the most similar in state 2, which is the period of positive expectations for 2021.



(a) HG1: Hidden states

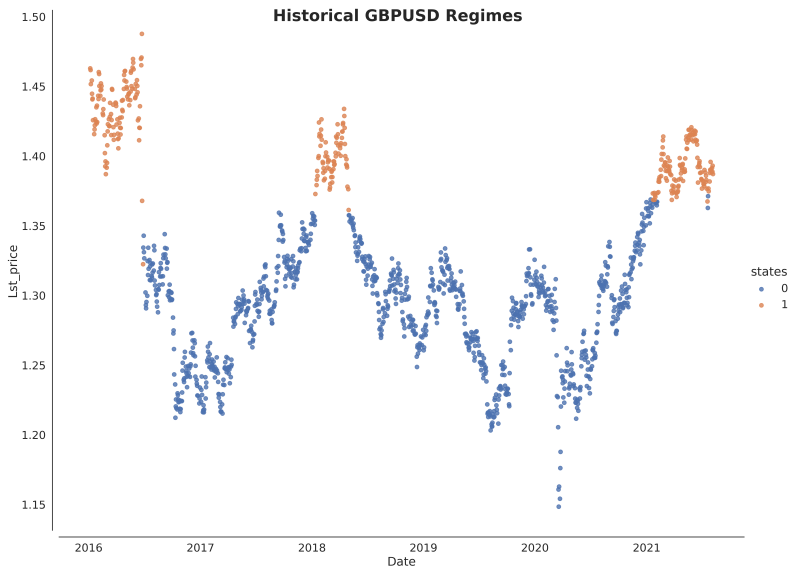


(b) HG1: Real vs Synthetic data for states

7.5 Currency: GBPUSD

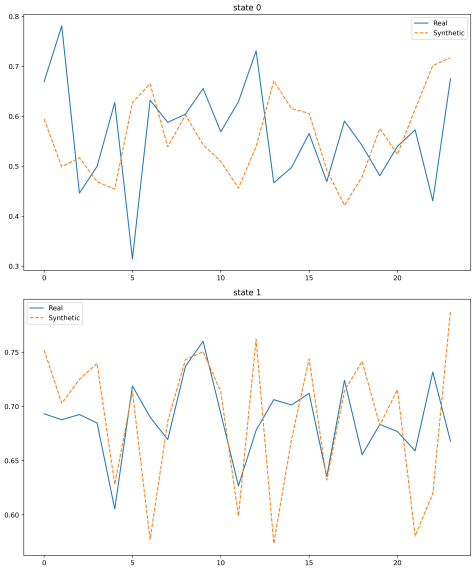
For GBPUSD currency, there exists two obvious hidden states. the 0th state is the low volatility with larger mean return 0.0032% and lower variance 0.0037%, and the 1th state is the high volatility with negative mean return -0.029% and higher variance 0.0058%.

		close price	daily return
0th hidden state	mean	1.2894	0.003181%
	var	0.001614	0.003664%
1th hidden state	mean	1.4078	-0.02876%
	var	0.0008762	0.005821%



From the historical GBPUSD regimes figure, there seems to be periodicity between states. There is a gap of 2 to 3 years between two adjacent 1th state. The intuition behind the states is the US president's fiscal plan combined with Federal Reserve tools. According to President Joe Biden's fiscal stimulus plan for the second half of 2021 and Federal Reserve's tapering monetary stimulus measures, if the regimes estimates are accurate, these signs indicate that the trend of GBP/USD will lower.

(a) GBPUSD: Hidden states



(b) GBPUSD: Real vs Synthetic data for states

GAN model results show that high volatility (1th state) has better generative accuracy than low volatility (0th state) in GBPUSD case.

8 Evaluation of Synthetic Data

8.1 Evaluation Process

One of the foremost challenges to generating synthetic data is the evaluation of "how real" synthetic data is.[**?**] Using the statistics which can describe the above features, combined with some other statistical features, such as first, second and third moments. We can use a set of statistics to describe time series. Using these real data and data with different statistical features, such as white noise, we train a classifier which can tell whether a time series is real. After that, we let the classifier to tell whether synthetic data looks real. If the classifier always classify synthetic data as real, we think our synthetic data show similar stylized features as real financial times series data.

8.2 Classification Algorithm

We train two classification algorithms,K-NN and Random Forest independently. After taking average of their output probabilities, we use this as the degree of "how real" synthetic data is.

8.3 Feature Choice

We try to evaluate whether a time series from the aspect that whether synthetic time series exhibit similar stylized features as real data. These are some financial time series stylized features:

1) Non-stationery. Future returns can behave quite different from history. We could not expect financial return series is a statistical stationary series.

2) Fat-tails. Financial returns are not in the form of normal distribution. They always have fatter tails, which means there are more extreme events happening.

3) Volatility clustering. Financial time series tend to have clustering volatility, which means another high volatility period is likely to happen after a previous volatility period.

4) Market Regime. Financial time series could behave totally different in different market regimes. In crashing regimes, for example, S&P index drop over 10 percent in month, the correlations among stocks could be really high, while in booming regimes, there are nearly no correlations among these stocks.

8.4 Model Training Process

Using statistics features that can reflect information of original data, including moments, auto-regression coefficients, heteroscedasticity of return time series, we trained classification algorithms to classify whether a time series looks like a stock return series. Train set was made up of real stock return data and time series that look quite like stock return but was generated from a mixture of normal distribution and normal distribution.

8.5 Evaluation Result

First, we do not take in consideration of market regimes. We use all of our data to train our generative models and let it generate synthetic data. The classifiers are also trained using all the data, then we use our trained model to analyze the synthetic data generated by Time GAN and

VAE. The result was as following. About one third of synthetic data was classified as not having the same statistics features as stock returns, while two thirds of synthetic data was classified as having the same statistics features as stock return series. We think this could be attributed to the huge differences among different market regimes.

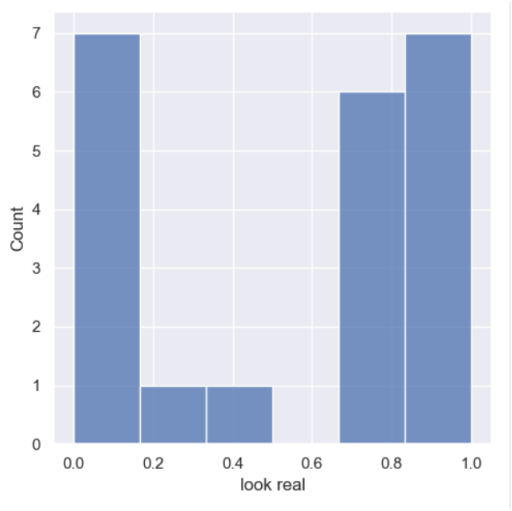


Figure 6.1 Evaluation result of Time GAN without considering regimes; data source="AAPL"

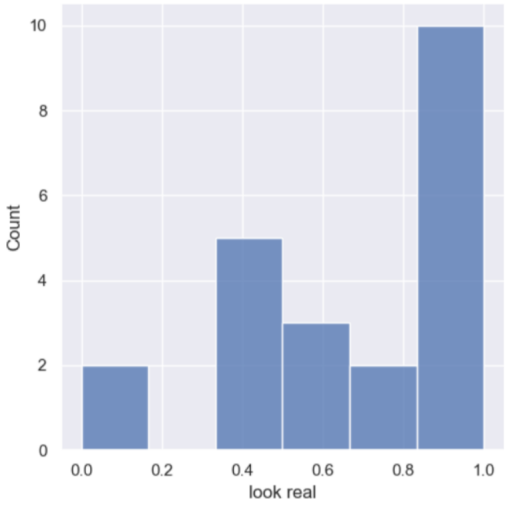


Figure 6.2 Evaluation result of VAE without considering regimes; data source="AAPL"

Then, we took in consideration of different market regimes. Using HMM model we previously developed, the who return series is classified into 3 market regimes. This time, we train our generative models conditioned on these regimes, and we also train our classifier conditioned on these regimes. We can see from the result, the degree of "how real" is greatly improved.

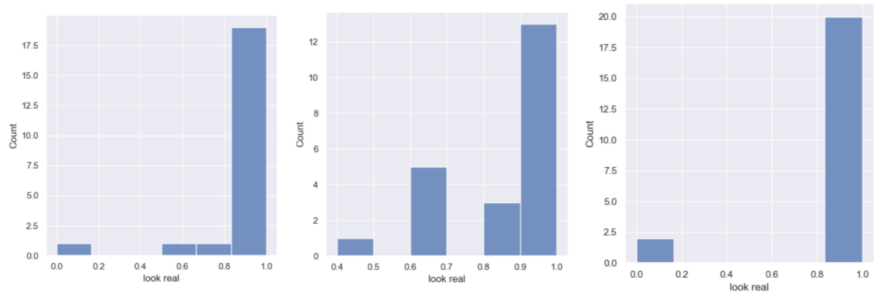


Figure 6.3 Evaluation result of Time GAN conditioned on regime 1,2,3; data source="AAPL"

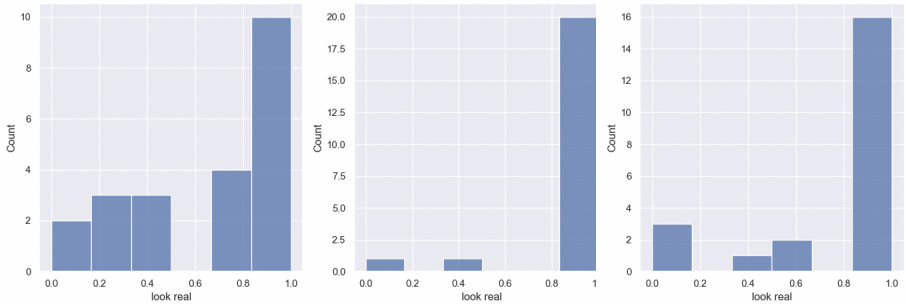


Figure 6.4 Evaluation result of VAE conditioned on regime 1,2,3; data source="AAPL"

9 Conclusion

We can clearly find that after taking in consideration market regimes, the efficiency of generative model are both improved. So, we believe it is useful to train generative models conditioned on different market regimes identified using HMM.

10 Future Plan

We will construct a more diverse portfolio including stocks of different industries, bonds, and other derivatives. We will further improve GAN model via using parameters containing additional information.

11 Contribution

Raymond Luo: VAE model building, evaluation of real time series, HMM Regime coding, Gaussian Copula Generation coding

Ziheng Chen: VAE model building, evaluation of synthetic time series

Elaine Huang: TimeGAN model building, Evaluation of synthetic data using PCA, TSNE and TSTR

Xiaobin Ou: data download and processing, cooperate in TimeGAN model building, model training and data generating

References

[1] Van der Schaar Lab, *Machine Learning and Artificial Intelligence for Medicine.*, https://github.com/vanderschaarlab/mlforhealthlabpub

[2] J. Yoon, D. Jarrett, M. der Schaar, *Codebase for Time-series Generative Adversarial Networks (TimeGAN) - NeurIPS 2019*, https://github.com/jsyoon0823/TimeGAN

[3] H. Buhler, B. Horvath, T. Lyons, I. Perez Arribas and B. Wood. *Generating financial markets with signatures*. SSRN 3657366, 2020.

[4] Z. Li, Y. Zhao, J. Fu, *SYNC: A Copula based Framework for Generating Synthetic Data from Aggregated Sources*, 2021

[5] Wang M, Lin Y-H, Mikhelson I. *Regime-Switching Factor Investing with Hidden Markov Models*. Journal of Risk and Financial Management. 2020; 13(12):311. https://doi.org/10.3390/jrfm13120311

[6] Thomas Wiecki, "An Intuitive, Visual Guide to Copulas", https://twiecki.io/blog/2018/05/03/copulas/, May, 2018

[7] Javier Franco-Pedroso1, "The ETS challenges: a machine learning approach to the evaluation of simulated financial time series for improving generation processes", Nov, 2018