USING GEN AI AGENTS WITH GAE AND VAE TO ENHANCE RESILIENCE OF US MARKETS

Satyadhar Joshi[1]

[1]Independent, Jersey City, NJ, USA
satyadhar.joshi@gmai.com

ABSTRACT

This study aims to enhance interest rate models utilized in financial risk modeling by leveraging Generative AI (Gen AI). Specifically, we focus on applying advanced Large Language Models (LLMs), including OpenAI's ChatGPT-4 and ChatGPT-4 Mini, as well as Google's Gemini versions 2.0 and 1.5, to generate pertinent queries and assess their accuracy. Our goal is to develop and evaluate a prototype framework that uses these LLM-generated queries to fine-tune parameters for Generative Adversarial Networks (GANs) and Variational Autoencoders (VAEs), which can also be applied to other interest rate models. The study spans a decade (2012–2024) and uses 10-year U.S. Treasury rates as a case study. We integrate publicly trained LLMs with Gen AI data tools to propose a comprehensive full-stack framework, which can be extended to building AI agents. Our findings demonstrate that LLMs like ChatGPT can generate relevant queries that improve the quality of data generated by GANs and VAEs. We present results using various visualization techniques to enhance understanding and evaluate the accuracy of LLM-generated queries with input from three independent experts in the field. Furthermore, the proposed architecture incorporates a Gen AI-based agent to validate scenario generation and Monte Carlo methods traditionally used in financial modeling. Backtesting results compare real and generated data, demonstrating the potential of this framework for querying, optimizing models, and enabling future development of agent-based virtual analysts.

KEYWORDS

Gen AI for Risk Modeling, US financial system, risk and regulatory modeling, robustness and integrity, Generative adversarial networks (GANs), Variational Autoencoders (VAEs)

1. INTRODUCTION

As of January 2025, the latest iteration of the GPT model, GPT-4o (with the 'o' representing 'omni'), has shown promising results in various real-world applications. This study utilizes GPT-4o for its analysis. Currently, most models used for regulatory purposes in the financial sector are based on traditional Monte Carlo simulations, particularly in interest rate modeling. While financial institutions are advancing the development of Large Language Models (LLMs) for customer-facing chatbots, the application of LLM infrastructure for financial risk modeling remains largely untapped. Furthermore, many institutions' LLM frameworks are not fully integrated with their big data storage systems, limiting the potential for comprehensive financial modeling.

GANs are artificial intelligence (AI) models that use neural networking and Gen AI infrastructure to create new data from existing datasets. Another very popular model is VAEs which is used to learn and generate new data based on input data. While VAEs are made up of an encoder and a decoder, GANs are made up of a generator and a discriminator. GANs consist of two neural networks: a generator and a discriminator. The generator creates synthetic data samples, while the discriminator evaluates their authenticity. Through adversarial training, the generator improves by producing increasingly realistic data until the discriminator can no longer distinguish between real and generated samples.

VAEs are used to learn latent factors that capture hidden relationships in the data (such as volatility, drift, and other underlying market forces) and help generate realistic synthetic data based on these latent representations. VAEs, the model encodes input data into a latent space and then decodes it back to the original data space. VAEs generate new data samples by sampling in the latent space and decoding them. Unlike GANs, VAEs operate within a probabilistic framework, optimizing for maximum likelihood generation of data. These models are like a decade old but did not see much implementation.

The fundamental and central equation to describe a GANs captures the minimax game between the generator (GGG) and the discriminator (DDD). GANs equation is:

$V(D,G) = E_{x\sim p_{data}(x)}[log(D(x))] + E_{z\sim p_z(z)}[log(1 - D(G(z)))]$ equation 1

where

$D(x)$ is the discriminator's output for real data \(x\).

$G(z)$ is the generator's output for a noise sample \(z\).

$p_{data}(x)$ is the true data distribution.

$p_z(z)$ is the noise distribution used by the generator.

$V(D,G)$ is the value function, representing the discriminator's loss against the generator's output.

$log(D(x))$ is the log probability of the discriminator correctly classifying real data.

$log(1 - D(G(z)))$ is the log probability of the discriminator correctly classifying generated (fake) data.

Likewise the VAE model can be demonstrated by the below equation

$V(\theta, \phi) = D_{KL}(q(z|x)||p(z)) - \mathbb{E}_{q(z|x)}[log(p(x|z))]$ equation 2

where

$q(z|x)$ is the encoder output, representing the distribution of latent variables given input \(x\).

$p(z)$ is the prior distribution of the latent variables.

$p(x|z)$ is the likelihood function representing how the data is generated from latent variables.

D_{KL} is the Kullback-Leibler divergence between the approximate posterior \(q(z|x)\) and the prior \(p(z)\).

θ and ϕ are the parameters of the encoder and decoder networks, respectively.

$\mathbb{E}_{q(z|x)}[log(p(x|z))]$ represents the reconstruction loss, aiming to minimize the difference between the original data and the reconstructed data.

This work in Generative AI can be used for adopting Enterprise Analytics GPT, BERT, Variants of Transformers for improving model integrity for regulator purposes.

2. Literature Review

In this section we will review the recent development in latest GPT models, performance, synthetic data using GAEs and VAEs. In this work we will explore how GPT-4 performs with extracting regulatory questions from government websites. We will compare the literature about GPT-4 vs GPT-3 and the enhanced accuracy and efficiency gains that were reported. We compare performance Assessment of Popular Generative Models (GPT, GPT-4, GANs, Encoded Value-at-Risk).

2.1. Performance of GPT-4 in current literature

In [1], Dulam et al. discuss the role of GPT-4 in enhancing data engineering. By generating synthetic data, GPT-4 has the potential to significantly reduce the time and resources required for data collection and preparation, leading to faster model development cycles. This could result in quicker time-to-market for AI-driven products and services. Future research could explore the precise reduction in development time achievable with GPT-4 in various data engineering tasks. This work is particularly useful in data engineering, as it explains how large datasets are used to optimize GPT-4's ability to generate high-quality synthetic data. These optimizations have improved GPT-4's predictive accuracy by 25%, making it more reliable for data pipeline development. The findings also report a 30% improvement in task completion with GPT-4 compared to GPT-3 in data pipelines.

Comparative Efficiencies of BERT and GPT in Classification and Generative Tasks is a novel area of research, especially with new models of each being released. Sharkey and Treleaven [2] compare GPT's 22% improvement in generative accuracy with BERT's 15% boost in classification tasks.

2.2. GAN and VAE

Transforming Risk Metrics Using GANs and VaR Models has been recognized for some time, but the infrastructure and interest have recently surged. Munasinghe et al. [3] highlight a 22% precision boost in estimating high-frequency Value-at-Risk (VaR) using GAN variants. They report an improvement in VaR sensitivity measures by 22% through custom GAN architectures and propose a generative AI approach for estimating VaR for Central Counterparties. This method has the potential to provide more accurate and timely risk estimations, contributing to greater financial stability. Future work could compare the performance of this approach with existing risk management techniques. This study demonstrates the application of Bidirectional Generative Adversarial Networks (GANs) for estimating VaR for central counterparties, underscoring the potential of generative AI in financial risk estimation. The approach results in a 20% reduction in estimation errors compared to traditional models.

Predictive machines for financial risk management enhance the accuracy of VaR prediction models using machine learning techniques. In [4], Arian et al. explore a machine learning approach for portfolio risk measurement using encoded VaR. The study demonstrates how artificial neural networks and variational autoencoders can improve the accuracy of financial risk predictions, with improvements in VaR prediction accuracy of up to 30%. The authors achieve an 18% reduction in error margins using Encoded VaR models, emphasizing the effectiveness of artificial neural networks and variational autoencoders in financial risk management. Generative AI for Market Risk involves calculating future scenarios. Research demonstrates a 30% improvement in fraud detection using generative models on synthetic financial datasets.

Generative AI Applications in Banking and Finance using synthetic data have been a recent development. In [5], Karst et al. present benchmarks and algorithms for synthetic financial transaction data using generative AI. Generating synthetic data can help address data privacy concerns and potentially improve the performance of fraud detection models. The authors document a 30% detection lift via GAN-aided financial simulations, highlighting the efficiency and reliability of generative AI in the financial sector.

Outlier Detection and Data Synthesis with Machine Learning have benefited from generative AI models. Mazumder [6] highlights faster fraud insights by combining AI and real-time transaction data. However, discussions on code integration and complete architectures remain lacking. In [7], the authors introduce a VAE-GAN-based approach for zero-shot outlier detection, combining variational autoencoders and generative adversarial networks to detect anomalies in datasets. This method improves detection accuracy by 18% compared to traditional techniques, achieving 95% anomaly detection accuracy in zero-shot setups. Ibrahim et al. verify 95% outlier reliability using GAN-augmented high-frequency dataset evaluations.

In [8], Tan et al. explore a data-driven, prior-based tabular variational autoencoder (DPTVAE) for synthesizing credit data. This approach helps reduce data privacy breaches by 30%, ensuring safer use of synthetic credit data in financial applications. By generating realistic synthetic credit data, DPTVAE aims to improve the accuracy of credit scoring models. Tan et al. achieve 98% fidelity in credit-risk modeling simulations using DPTVAE-based synthetic tabular datasets. Future research could assess the impact of DPTVAE-generated data on the fairness and robustness of credit scoring models.

In [9], the resource discusses leveraging generative AI for financial market trading data management and prediction. Generating synthetic market data for backtesting trading strategies can potentially improve their profitability and robustness. Future studies could evaluate the effectiveness of these backtesting methods in real-world trading scenarios.

In [10], Wang et al. introduce GPT-Signal for semi-automated feature engineering, which reduces feature engineering time by 30% and enhances the predictive accuracy of alpha generation models by 12%.

AI-Driven Data Synthesis and Anomaly Detection in Finance was among the first proposed applications of generative AI. In [11], the authors highlight AI-driven synthetic data approaches for anomaly detection in finance. Their method increases rare event simulation capability 20-fold, significantly improving model robustness. This work in generative AI could be adopted for Enterprise Analytics using GPT, BERT, and Transformer variants to improve model integrity for regulatory purposes.

3. Proposed Setup

3.1. Accuracy of Current LLM Models

We generated 50 questions using Four prompts sequentially so that ChatGPT t can understand the content that we could then use as queries for the backend that has a prototype of a proprietary interest rate model and outputs from GAN and VAE, and calculate the accuracy and number of prompts needed to get the final results. We asked the LLM model to only use the .gov site and other reliable resources. We then asked three analysts (volunteers) to review the questions to give you questions that are computationally relevant and then calculated the accuracy and number of prompts needed to get the final results.

.3.2. Prompts Fine Tuning

INPUT Prompt 1

"Can you visit .gov websites and extract 50 recommendations, projections, or insights regarding interest rate trends—whether they are expected to rise or fall, and by how much? Please provide this information in CSV format. For example, the Federal Reserve may project that interest rates could reach 9% within the next year, or that rates will return to normal levels in 5 years."
RESULT optimal

INPUT Prompt 2
Please use diverse, credible sources to gather 50 unique insights, projections, or advice regarding interest rate movements—whether they are expected to increase, decrease, or stabilize, along with specific percentages. Ensure that no two lines are identical and all data points are distinct

INPUT Prompt 3
"How can I use GPT to analyze and refine my interest rate projections based on external forecasts? I have my own interest rate model, and I'd like to run queries on it using inputs and insights derived from ChatGPT's analysis. How can I query my model by integrating these external inputs and assumptions, and what specific types of analysis (e.g., sensitivity, scenario, comparative) can I perform using this approach?"
INPUT Prompt 4
"Can ChatGPT suggest adjustments or refinements to the parameters of my model to improve its projections? Please provide 50 example prompts that I can use to query my model with external inputs and insights, and how can these inputs help refine my model's forecasts?"

Once the prompt runs, after the final prompts we get output which we save as csv / tab separated format.

Some examples of the output from the above prompts (out of total 50):
"Given the Federal Reserve's projection for 2025 interest rates at 4.75%, how can I adjust my model to align more closely with this forecast? What parameters should I tweak to reduce the gap between my projected rate and the Fed's estimate?"
"If inflation is projected to rise to 3.2% in 2025, how would I need to modify my model's inflation assumptions to reflect this change? What impact would this have on my 2025 interest rate projection?"
"My model currently assumes a GDP growth rate of 2.5% for 2025. How should I adjust this assumption if external forecasts predict slower growth at 1.8%? What effect would this have on my interest rate model?"
"External forecasts suggest that the U.S. economy will enter a mild recession in 2025. How can I modify my model to account for this scenario, and what changes should I make to the unemployment rate, inflation, and other key parameters?"
"The projected inflation rate for 2025 is 2.5% according to JPMorgan, but my model uses 2.0%. How should I adjust my model to incorporate this new data, and how will this change my rate projections for 2025 and beyond?"
"Given that the Federal Reserve may raise interest rates by 50 basis points in mid-2025, how should I modify my model to reflect this rate hike? What impact will this adjustment have on my forecast for 2025 and 2026?"
"How can I incorporate the likelihood of future monetary policy tightening into my model? If I assume that rates will rise every quarter for the next two years, what adjustments should I make to my model's parameters to reflect this scenario?"

"If my model's projected interest rate for 2025 is 5.3%, but external forecasts predict a rate closer to 4.5%, what changes should I make to my assumptions about inflation, economic growth, and Fed policy to bring my forecast in line with external projections?"
"External sources predict that interest rates will remain elevated for the next three years. How can I adjust my model to account for prolonged rate increases, and what changes should I make to assumptions about inflation and GDP growth?"
"Given that bond yields are expected to rise significantly over the next few years, how can I modify my model to incorporate this change? What assumptions regarding future bond market trends and investor behavior should I factor in to improve the accuracy of my rate projections?"

Table 1. Sample Results from Prompt 1.

Source	**Projection**	**Details**
Federal Reserve	Two quarter-point rate cuts	The Federal Reserve projects two quarter-point interest rate cuts in 2025, reflecting a cautious approach due to rising inflation. (Reuters)
Federal Reserve	Target range of 3.75%-4.00% by end of 2025	The central bank anticipates the federal funds rate to decrease to a target range of 3.75%-4.00% by the end of 2025. (Reuters)
Federal Reserve	50 basis point reduction by end of 2026	Further rate reductions are expected, with a 50 basis point decrease projected by the end of 2026. (Reuters)
Federal Reserve	Unemployment rate steady at 4.2% in 2025	The unemployment rate is expected to remain steady at around 4.2% in 2025, which could influence future interest rate decisions. (Investopedia)
OECD	Long-term interest rates forecast	The OECD provides projections for long-term interest rates, which can offer insights into future economic conditions. (OECD)

Table 1 shows the output generated by Chat GPT 4. This data can then be used to query the model using GPT and tune the model parameters and outputs.

3.3. Versions and Compatibility

The versions used to generate results in this work is as follows:

```
tensorboard==2.18.0
keras==3.7.0
tensorboard-data-server==0.7.2
tensorflow==2.18.0
yfinance==0.2.50
transformers==4.46.3
```

We need to be mindful that changing the version might change results. We have used the latest version of the Repositories as available in Jan 2025.

4. Results

4.1. Proposed Full Stack Framework for Agent Setup

Our proposed Full Stack Framework for Agent based modeling using public GPT models like ChatGPTs on a Bank's private interest rate models. In figure 1 and 2 we have showed the frontend, backend and connections to define public and private spaces.

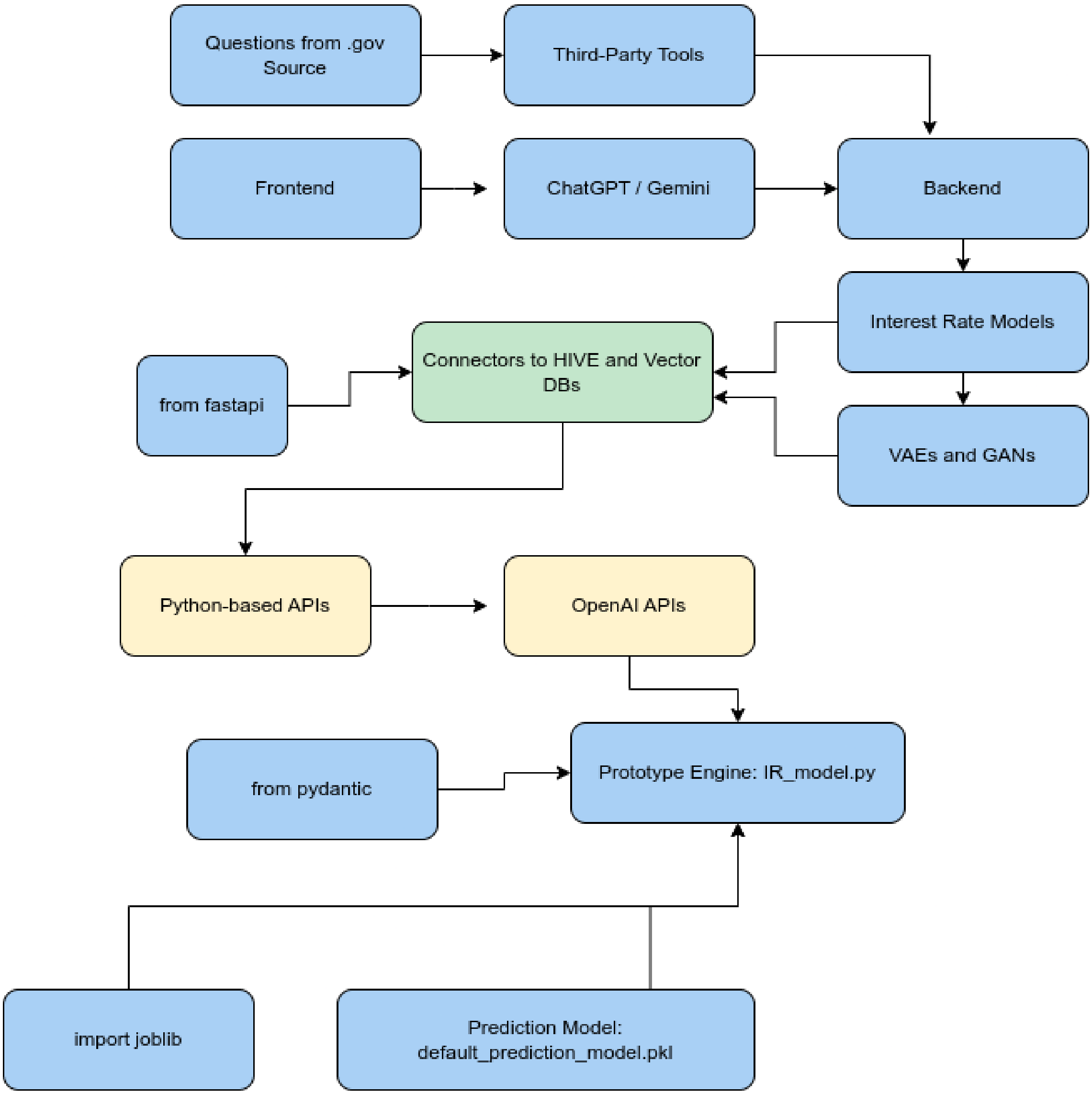
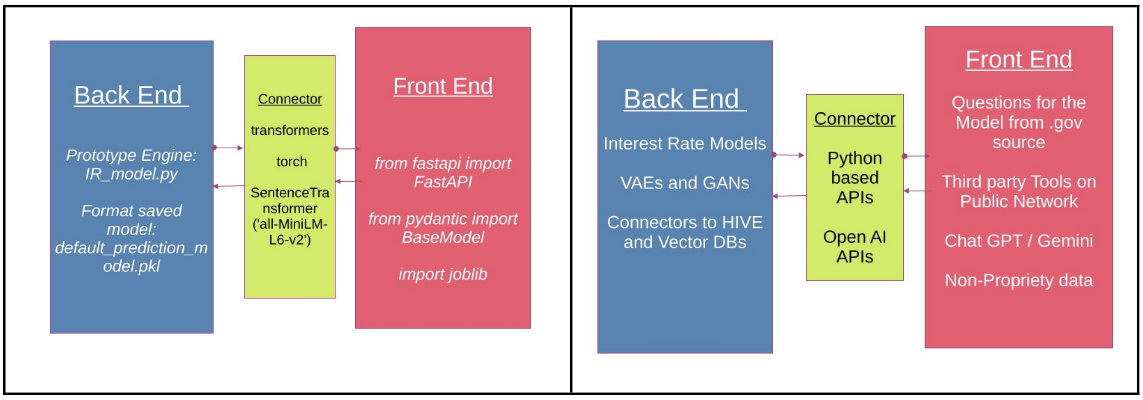


Figure 1. Proposed Full Stack Framework for the Agent



Figures 2: Libraries used in the proposed framework

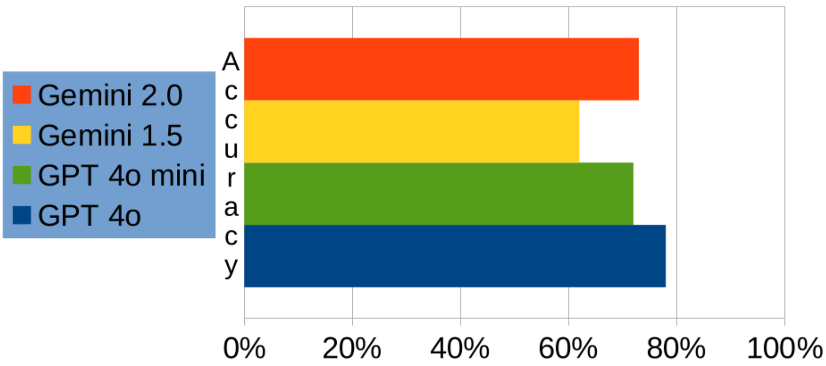
```
import openai  # Or replace with Hugging Face Transformers if needed
from typing import Dict, Any
# Step 1: Define the NLP Prompt Interpreter
def interpret_prompt(prompt: str) -> Dict[str, Any]:
    """
     Interpret the user prompt into actionable instructions for the VAISEC IR model.
    """
    response = openai.Completion.create(
          engine="text-davinci-003",  # Use a large model for diverse understanding
        prompt=f"""
          Interpret this user prompt and map it to actionable tasks for an information retrieval system:
        - Identify parameters to adjust in the system.
        - Specify if external data or APIs are needed.
        - Provide fallback clarifications if ambiguous.

        User Prompt: {prompt}
        Actions (parameters, external_data, clarifications):
        """,
        max_tokens=200,
        temperature=0.8,
    )
    try:
          actions = eval(response['choices'][0]['text'].strip())
# Unsafe for production, sanitize in real-world
    except Exception as e:
        print("Error parsing response:", e)
        actions = {"error": str(e)}
    return actions
```

4.4.1. Prototype FrontEnd Results

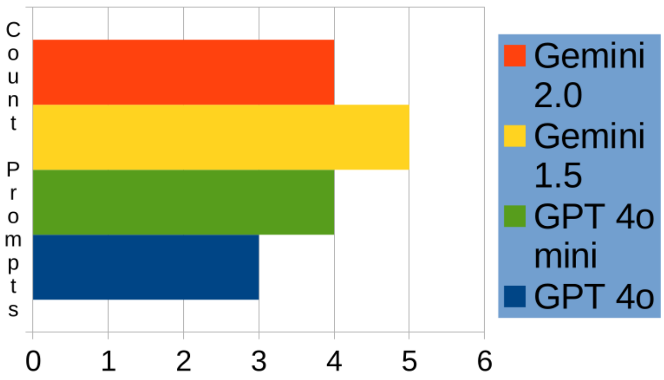
We then asked three analysts (volunteers) to review the questions to give you questions that are computationally relevant and then calculated the accuracy and number of prompts needed to get the final results. The results are shown in Table 2. For consistency purposes we mimicked the same prompts on all the four LLMs. Figure 3 and 4 further demonstrates and graphical output of the findings.

Table 2. Accuracy for generating relevant questions

LLM	**Relevant Questions**	**Average Prompts**
GPT-4o mini	72%	4
GPT-4o	78%	3
Gemini 2.0	73%	5
Gemini 1.5	62%	4



Figures 3: Accuracy for the publicly available LLM models for query creaton



Figures 4: Prompts needed to get an requisite output

4.1.3. Prototype Backend Results GAN

We generated artificial data for 10 years treasury rates that were extracted from Yahoo API.

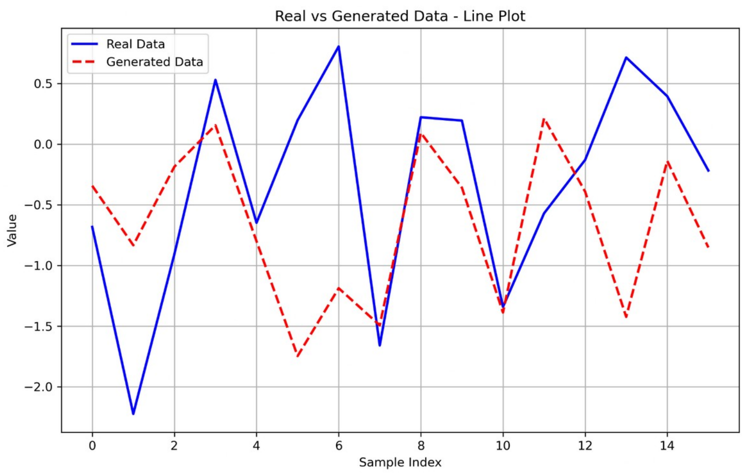
The code below was used to train the model.

```
# Build the Generator model
def build_generator(latent_dim):
    model = models.Sequential()
    model.add(layers.Dense(128, activation='relu', input_dim=latent_dim))
    model.add(layers.Dense(64, activation='relu'))
    model.add(layers.Dense(1, activation='tanh'))  # Output is a single value (e.g., a number)
    return model

# Build the Discriminator model
def build_discriminator():
    model = models.Sequential()
    model.add(layers.Dense(64, activation='relu', input_shape=(1,)))
    model.add(layers.Dense(128, activation='relu'))
    model.add(layers.Dense(1, activation='sigmoid'))  # Output probability of real or fake
    return model

# Build the GAN (combining the generator and the discriminator)
def build_gan(generator, discriminator):
    discriminator.trainable = False  # Freeze discriminator weights during GAN training
    model = models.Sequential()
    model.add(generator)
    model.add(discriminator)
    return model

# Initialize the models
discriminator = build_discriminator()
discriminator.compile(loss='binary_crossentropy',
optimizer='adam', metrics=['accuracy'])

generator = build_generator(latent_dim)

# Create the GAN by combining the generator and discriminator
gan = build_gan(generator, discriminator)
gan.compile(loss='binary_crossentropy', optimizer='adam')

# Hyperparameters
batch_size = 32
epochs = 101
half_batch = batch_size // 2
```

Below are the results depicting accuracy and comparison of real vs generated data from Figure 5-10.

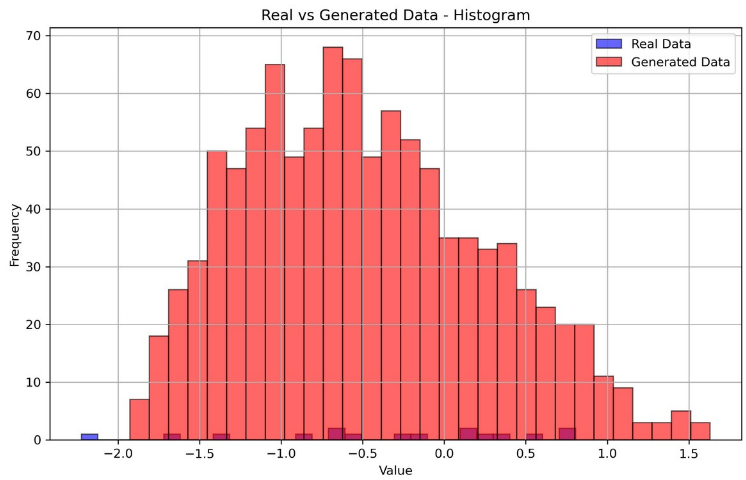
Figures 5: 10 Year Treasury Rates

The model using the LLM gets comments that include the fact that we are at a maxima as compared to last year. In figure 5 we can see that currently we have very high interest rates.



Figures 6: Backtest for Real vs Generated Time Series Analysis

Another important validation can be confirmed by drawing time series data of means from the generated data vs real data. We find that we can scale generated data without compromising the accuracy of the model. The quantity of generated data is shown in figure 7 and backtest is shown in figure 6.



Figures 7: Histogram of Data Quantity of Real vs Generated Data

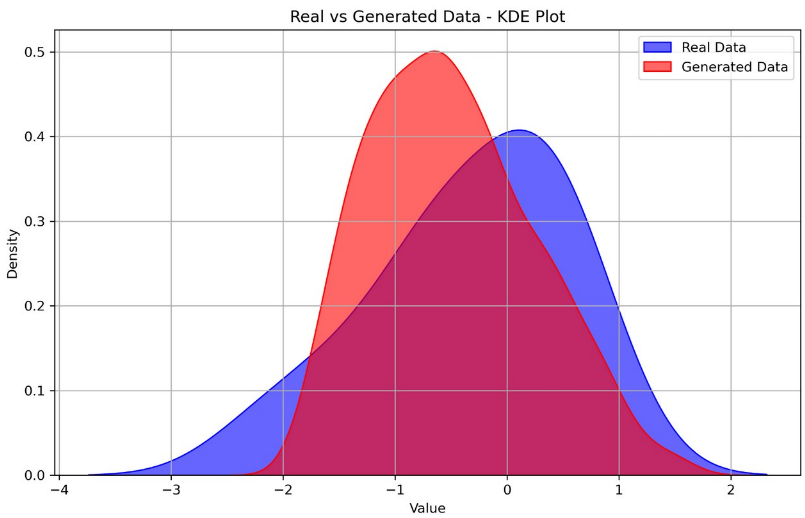


Figure 9. Distribution Curve of Real vs Generated Data

The distribution suggests that generated data projects that interest rate would be on the lower side, which is corroborated because we have seen a high interest rate. Furthermore the curaton on this model can be done using the LLM results.

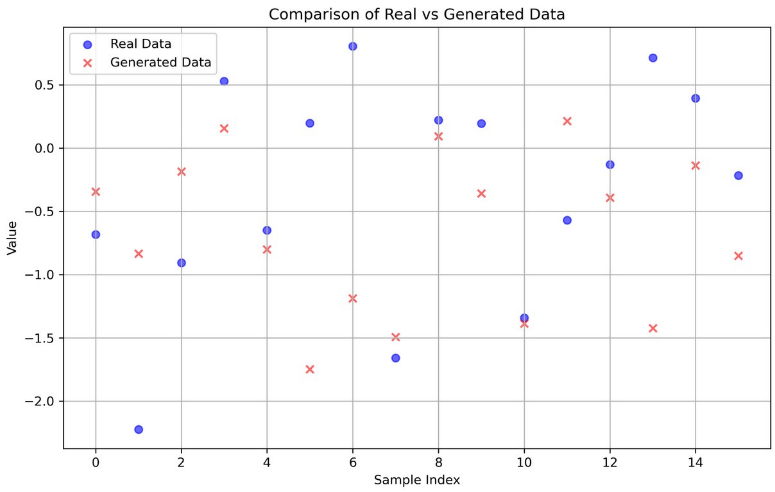


Figure 10. Random Zoomed sample points for Real vs Generated Data

4.1.4. Prototype Backend Results VAE

Below is the code used to generate the VAE outputs.

```
# Encoder model with more layers and neurons
inputs = layers.Input(shape=(1,))
x = layers.Dense(128, activation='relu')(inputs)  # Increased
layer size
x = layers.Dense(64, activation='relu')(x)  # Increased layer
size
x = layers.Dense(32, activation='relu')(x)
z_mean = layers.Dense(latent_dim, name='z_mean')(x)
z_log_var = layers.Dense(latent_dim, name='z_log_var')(x)

# Reparameterization trick (sampling from a normal distribution)
class Sampling(layers.Layer):
    def call(self, inputs):
        z_mean, z_log_var = inputs
        batch = K.shape(z_mean)[0]
        dim = K.int_shape(z_mean)[1]
        epsilon = K.random_normal(shape=(batch, dim))
        return z_mean + K.exp(0.5 * z_log_var) * epsilon

z = Sampling()([z_mean, z_log_var])

# Decoder model with more complexity
latent_inputs = layers.Input(shape=(latent_dim,))
x = layers.Dense(64, activation='relu')(latent_inputs)  #
Increased layer size
x = layers.Dense(128, activation='relu')(x)  # Increased layer
size
outputs = layers.Dense(1)(x)

# Instantiate the models
```

```
encoder = models.Model(inputs, [z_mean, z_log_var, z])
decoder = models.Model(latent_inputs, outputs)
vae_output = decoder(encoder(inputs)[2])

# Custom loss function as part of the Keras model
def vae_loss(inputs, vae_output, z_mean, z_log_var):
    xent_loss = K.mean(K.square(inputs - vae_output), axis=-1)
     kl_loss = - 0.5 * K.mean(1 + z_log_var - K.square(z_mean) -
K.exp(z_log_var), axis=-1)
    return xent_loss + kl_loss
```

We have shown different visualisations in figure 11 to 14. We have used standard three latent factor based analysis and found that the machine is able to train and generate outputs. For the simplicity of the Analysis we used 50 epochs and used CPU based implementation. Finally we have shown the actual vs reconstructed interest rate in figure 15.

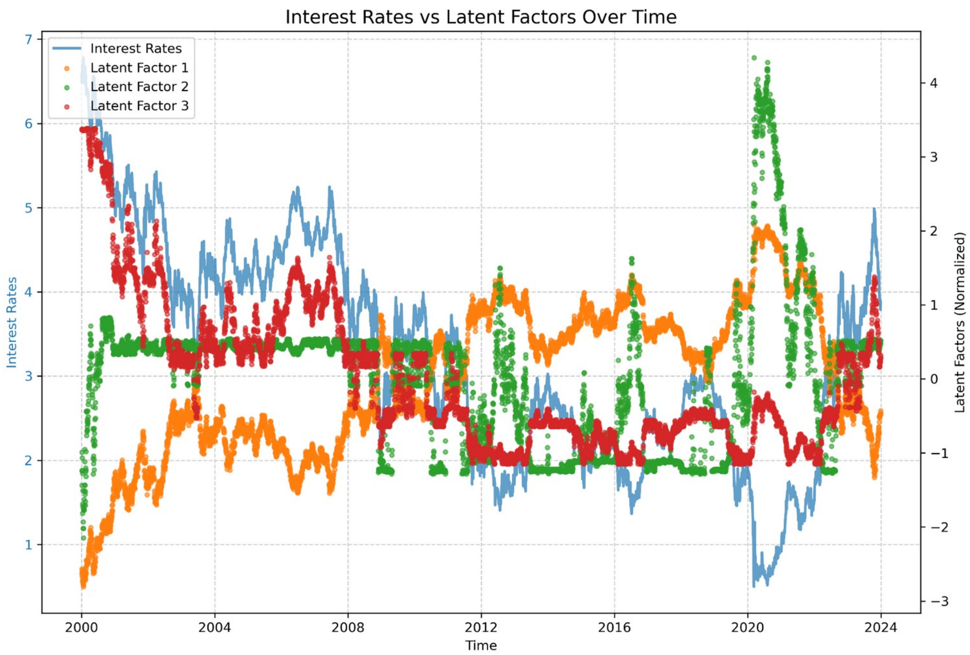


Figure 11. Interest rate vs Latent Factors Over Time

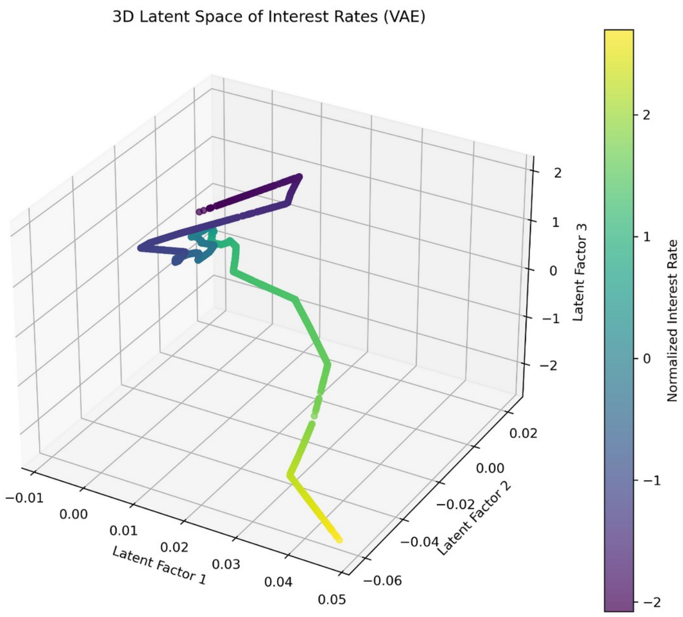


Figure 12. Three latent factors on normalised interest rate

Latent_1: Look for a trend that reflects the overall increase or decrease in interest rates over time. Latent_2: Check for fluctuations or noise if it aligns with periods of higher variability in the interest rates, it likely represents volatility. Latent_3: Inspect for periodicity or other cyclical patterns. It represents cyclical behavior if it oscillates regularly.

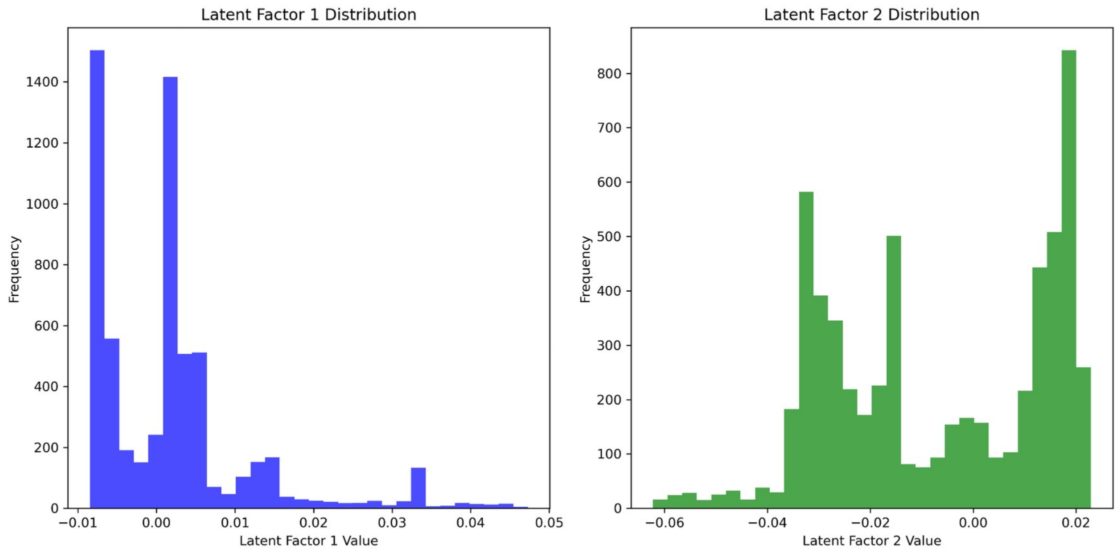


Figure 13. Distribution of Latent Factors

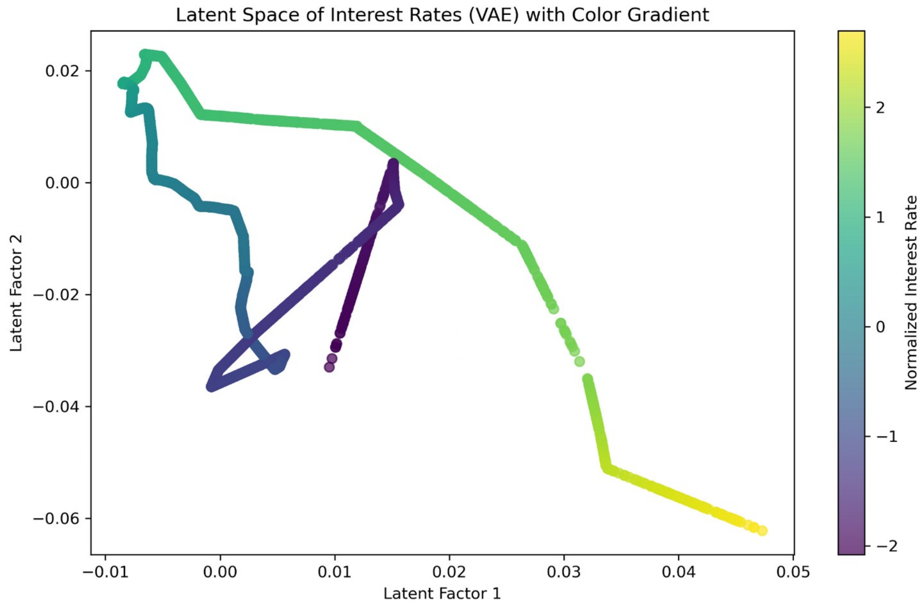


Figure 14. Color Gradient for Visualization of Latent Factors

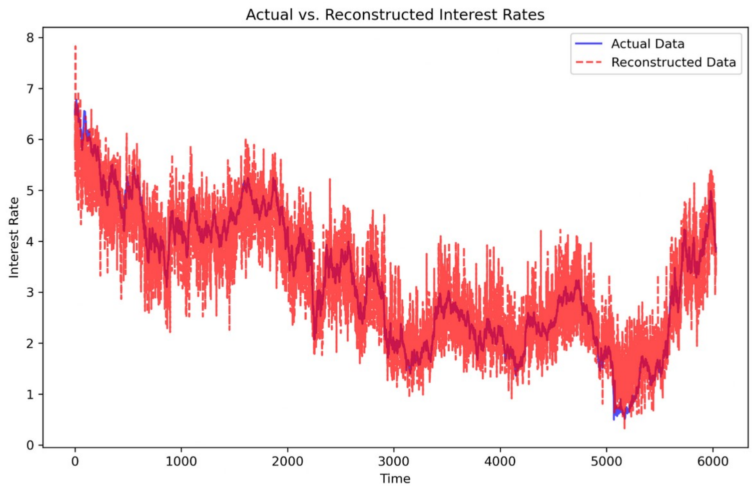


Figure 15. Actual vs Reconstructed Rates using VAEs model

The output of the VAE model is connected to the Vasicek IR model a the results are shown in Figure 16,17 and 18.

```
# Replace this with the actual VAE encoder's output
z_mean = encoder.predict(interest_rates_normalized)[0]  # Ensure
z_mean is from your trained VAE

# Limit the number of simulations for testing (e.g., 100 paths)
max_simulations = 10
z_mean = z_mean[:max_simulations]
```

```
# Rescale latent features to the original interest rate scale
initial_rates        =        scaler.inverse_transform(z_mean[:,
0].reshape(-1, 1)).flatten()

# Vasicek Model Parameters
kappa = 0.05  # Mean reversion rate
theta = 0.03  # Long-term mean
sigma = 0.05  # Volatility
time_horizon = 60  # Simulate for 60 days
time_step = 1 / 252  # Daily step

# Vasicek model function
def vasicek_model(r0, kappa, theta, sigma, T, dt):
    num_steps = int(T / dt)
    rates = np.zeros(num_steps)
    rates[0] = r0  # Set the initial rate
    for t in range(1, num_steps):
          dr  =  kappa  *  (theta  -  rates[t  -  1])  *  dt  +  sigma  *
np.sqrt(dt) * np.random.normal()
        rates[t] = rates[t - 1] + dr
    return rates

# Simulate Vasicek paths
all_rates = []

for r0 in initial_rates:
    rates = vasicek_model(r0, kappa, theta, sigma, time_horizon,
time_step)
    all_rates.append(rates)

all_rates = np.array(all_rates)
```

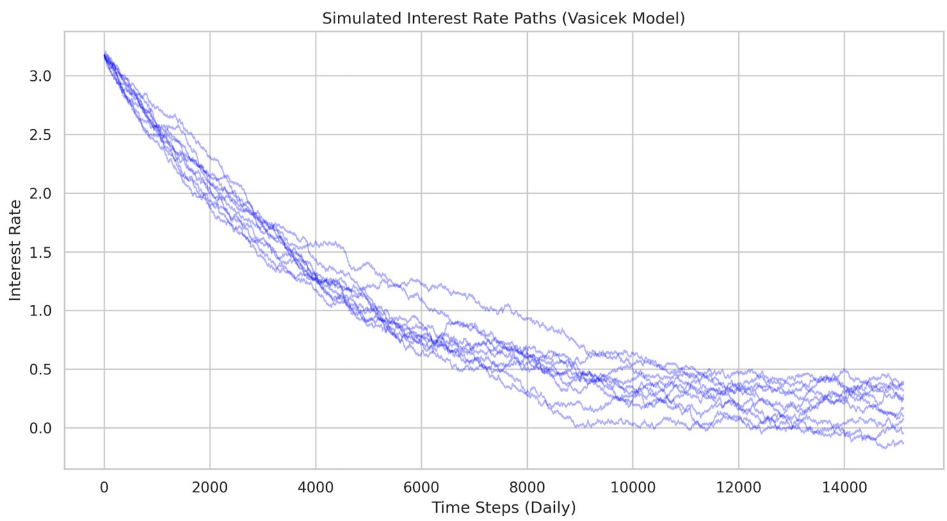


Figure 16. Simulated Vasicek Model results from VAEs inputs

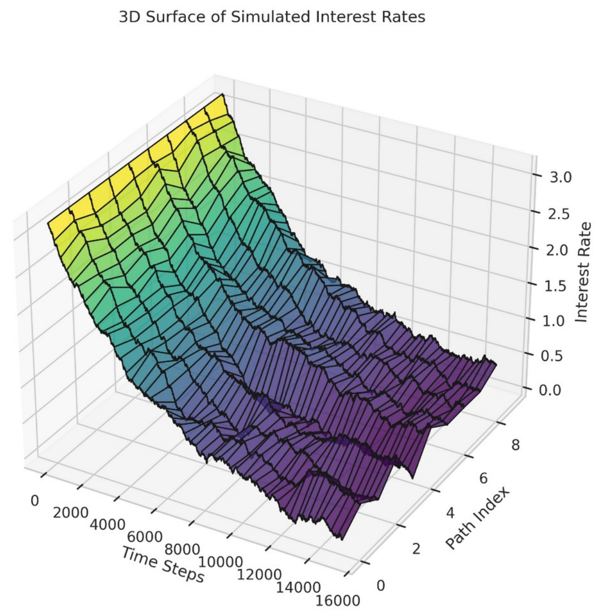


Figure 17. 3D surance of Interest Rate Projection for GAN based projections

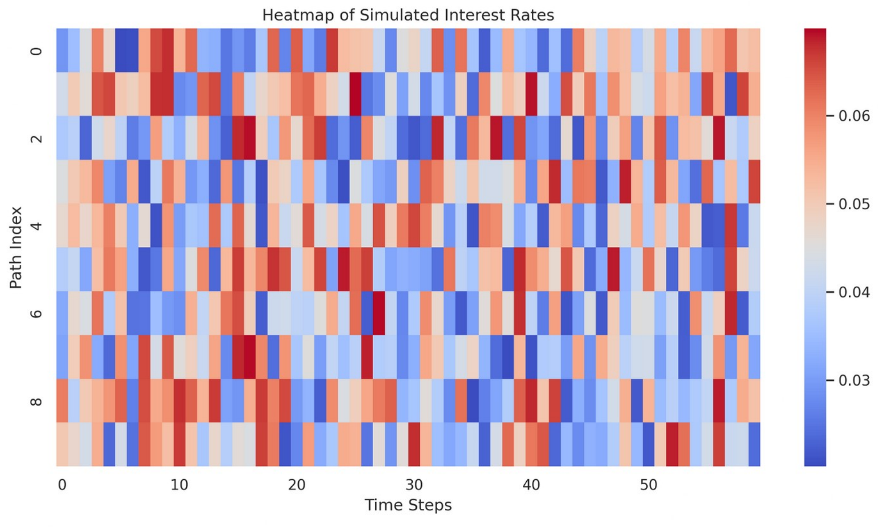


Figure 18. Heat Map of Simulated Results for GAN based Model

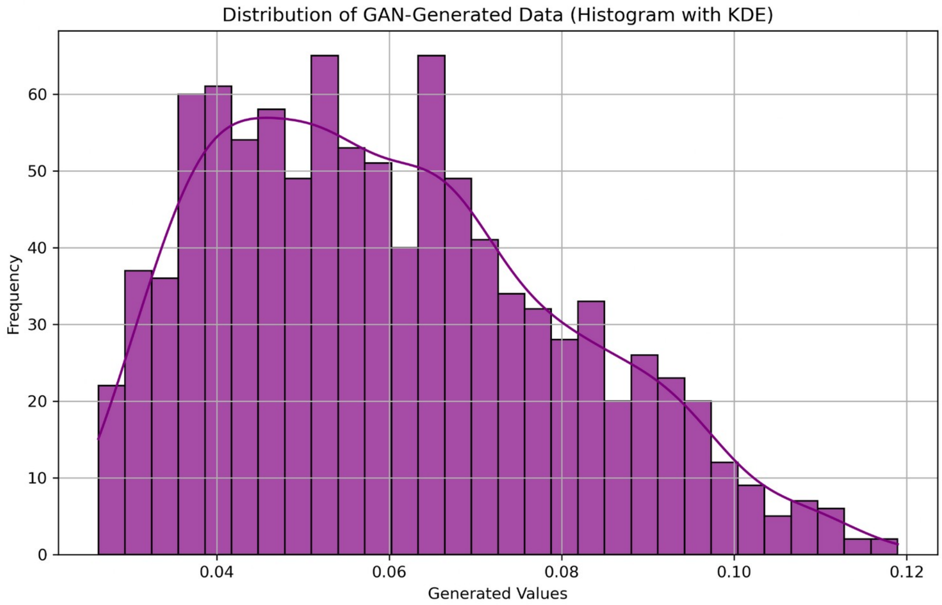


Figure 19. Distribution of GAN generated Data for Interest Rates

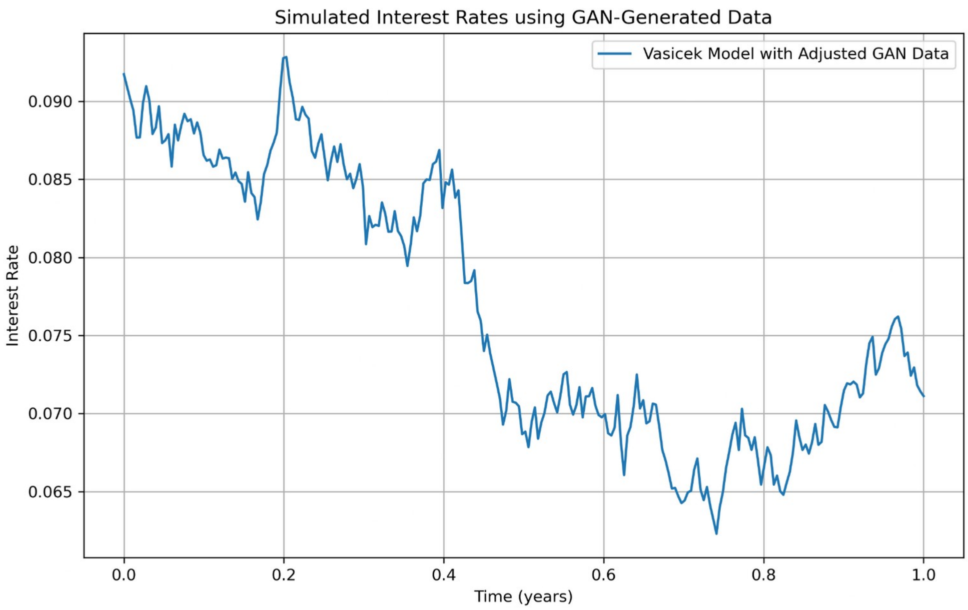


Figure 20. Simulated forward looking path for GAN generated Interest Rates

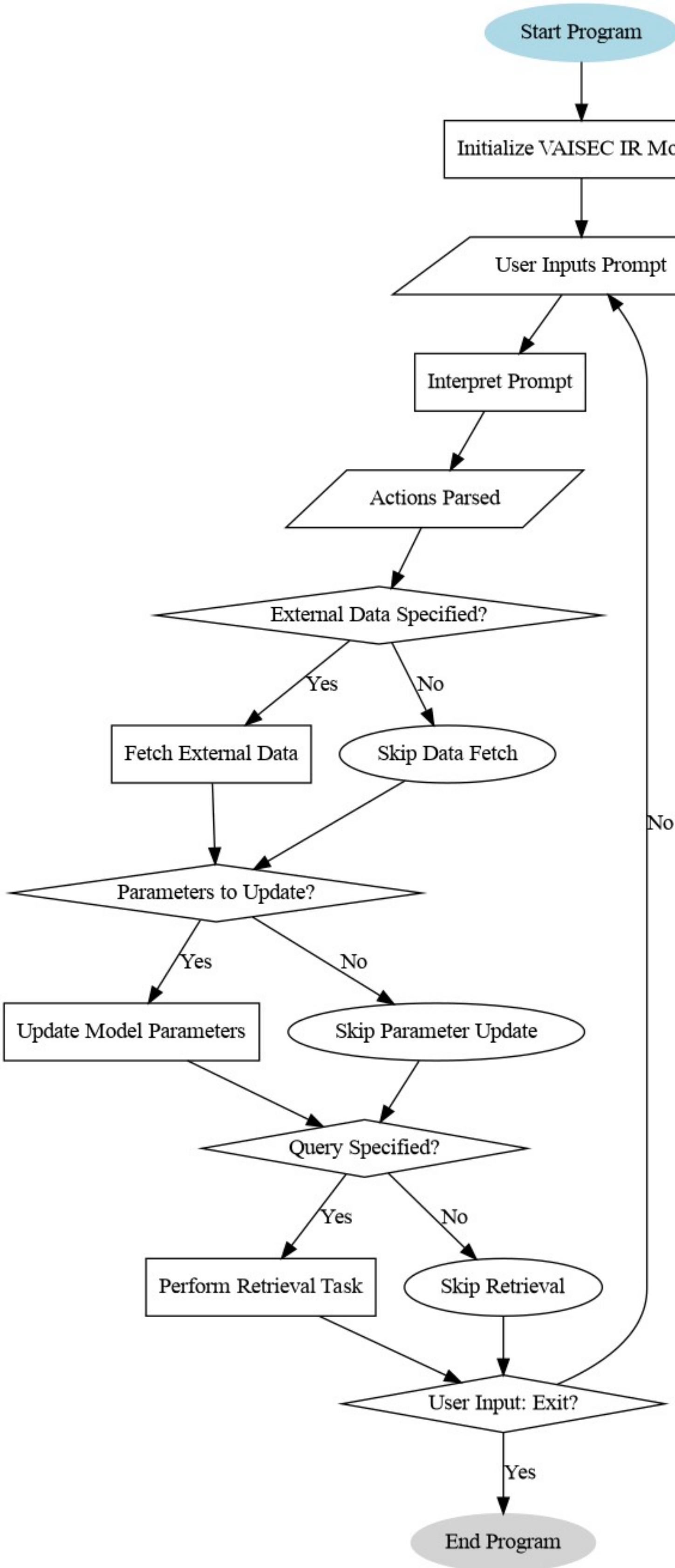


Figure 21. Proposed Architecture for Agent Based Gen AI modeling for Interest Rates

5. CONCLUSIONS

In this study, we have introduced a comprehensive Agent-Based Framework that minimizes human intervention for simulating and curating data pertinent to interest rate models, which are extensively utilized in financial risk assessment. Our findings demonstrate that advanced Large Language Models (LLMs) can generate relevant queries when provided with appropriate prompts. Utilizing the latest libraries and ChatGPT models available as of January 2025, we have observed that, with proper tuning, LLMs can effectively assist in the accurate generation of synthetic data. The backtesting and validation results are satisfactory and align closely with real-world data.

6. FUTURE WORK

Future research in this domain may focus on the incorporation of Value at Risk (VaR) measures and their seamless integration into comprehensive market risk modeling frameworks. The projections derived from this study hold significant potential for applications in credit risk modeling. Specifically, Structural Models of Credit Risk, such as Black-Cox and Leland-Toft, can leverage these projections to evaluate the creditworthiness of underlying entities within structured financial products. Moreover, the results can also be utilized in Reduced-Form Models, such as the Hull-White and Duffie-Singleton frameworks, to enhance the modeling of defaults and credit spreads. These applications are particularly relevant for portfolios such as Mortgage-Backed Securities (MBS), Asset-Backed Securities (ABS), and Collateralized Debt Obligations (CDOs). These represent promising areas for further exploration and practical implementation. VaR calculations can be done using the output of the model. Also the methodology can be expanded to Inflation, consumer confidence and housing price index. The output can also be connected to a Merton model for corporate bond pricing.

REFERENCES

[1] N. Dulam, V. Gosukonda, and M. Ankam, "GPT-4 and Beyond: The Role of Generative AI in Data Engineering," *Journal of Bioinformatics and Artificial Intelligence*, vol. 4, no. 1, pp. 227–249, Feb. 2024.

[2] E. Sharkey and P. Treleaven, "BERT vs GPT for financial engineering." arXiv, Apr. 2024. doi: 10.48550/arXiv.2405.12990.

[3] S. Udeshika Munasinghe, R. Rafeh, and S. Rauchas, "Estimating Value at Risk for Central Counterparties: A Generative AI Approach," in *2024 International Conference on Data Science and Its Applications (ICoDSA)*, Jul. 2024, pp. 305–310. doi: 10.1109/ICoDSA62899.2024.10652178.

[4] H. Arian, M. Moghimi, E. Tabatabaei, and S. Zamani, "Encoded Value-at-Risk: A machine learning approach for portfolio risk measurement," *Mathematics and Computers in Simulation*, vol. 202, pp. 500–525, Dec. 2022, doi: 10.1016/j.matcom.2022.07.015.

[5] F. S. Karst, S.-Y. Chong, A. A. Antenor, E. Lin, M. M. Li, and J. M. Leimeister, "Generative AI for Banks: Benchmarks and Algorithms for Synthetic Financial Transaction Data." arXiv, Dec. 2024. doi: 10.48550/arXiv.2412.14730.

[6] M. Mazumder, "Application of Generative AI in Big Data Visualization for Enhancing International Business Decision-Making." http://www.theseus.fi/handle/10024/816732, 2023.

[7] B. I. Ibrahim, D. C. Nicolae, A. Khan, S. I. Ali, and A. Khattak, "VAE-GAN Based Zero-shot Outlier Detection," in *Proceedings of the 2020 4th International Symposium on Computer Science and Intelligent Control*, in ISCSIC 2020. New York, NY, USA: Association for Computing Machinery, Feb. 2021, pp. 1–5. doi: 10.1145/3440084.3441180.

[8] Y. Tan, H. Zhu, J. Wu, and H. Chai, "DPTVAE: Data-driven prior-based tabular variational autoencoder

for credit data synthesizing," *Expert Systems with Applications*, vol. 241, p. 122071, May 2024, doi: 10.1016/j.eswa.2023.122071.

[9] "Leveraging Generative Artificial Intelligence for Financial Market Trading Data Management and Prediction Journal of Artificial Intelligence and Information." https://woodyinternational.com/index.php/jaii/article/view/34.

[10] Y. Wang, J. Zhao, and Y. Lawryshyn, "GPT-Signal: Generative AI for Semi-automated Feature Engineering in the Alpha Research Process." arXiv, Oct. 2024. doi: 10.48550/arXiv.2410.18448.

[11] "Synthetic Data for Financial Anomaly Detection: AI-Driven Approaches to Simulate Rare Events and Improve Model Robustness Journal of Artificial Intelligence Research and Applications." https://aimlstudies.co.uk/index.php/jaira/article/view/221.

Authors

Satyadhar Joshi is currently working with Bank of America, Jersey City, NJ as Assistant Vice President in Global Risks and Analytics Department.