Instructor: Dr. Svitlana Vyetrenko

IMPROVING TIME SERIES FORECASTING WITH GANS

MACHINE LEARNING FOR FINANCIAL MARKETS PROJECT

Group composition: Isabella Cappiello, Diego Fracassi, Francesca Vasta

Introduction and motivation

Traditional prediction models for time series

Traditional financial prediction models, such as **ARIMA**, struggle to accurately capture complex patterns and nonlinear trends in stock market data. While ARIMA can identify general trends, it lacks precision in forecasting dynamic market behaviors.

Introduction and motivation

Best prediction models for time series

Advanced machine learning models, such as **Long Short-Term Memory (LSTM)** networks and **eXtreme Gradient Boosting (XGBoost)**, have demonstrated superior performance in financial time-series forecasting. According to the study "Deep Learning for Stock Market Prediction" by Nabipour et al. (2020), these models outperform traditional statistical approaches, like AutoRegressive Integrated Moving Average (ARIMA), by capturing complex dependencies, non-linear patterns, and market fluctuations more effectively.

Time-GANs to create financial synthetic data

Furthermore, **Time-series Generative Adversarial Networks (TimeGAN)** have emerged as a powerful method for generating high-quality synthetic time-series data. As described in the paper by Yoon, Jarrett, and van der Schaar (2019), presented at NeurIPS, TimeGAN uniquely combines adversarial training with supervised learning to preserve the temporal correlations and latent structure of real-world time-series data. This capability makes it particularly valuable in financial modeling, where data scarcity, missing values, and overfitting are common challenges.

Research question

Can TIME-GANS improve LSTM and XGBOOST predictions of financial data?

Can the integration of TimeGAN with predictive models such as LSTM and XGBoost **enhance forecasting accuracy** by augmenting training datasets with realistic synthetic data?
Furthermore, does this approach help mitigate the limitations of small or incomplete datasets while improving model generalization for more robust and reliable financial predictions?

Dataset

Data type

Time-series

Description

Microsoft's stocks data from 02/20/2015 to 02/19/2025 covering an horizontal period of 10 years.

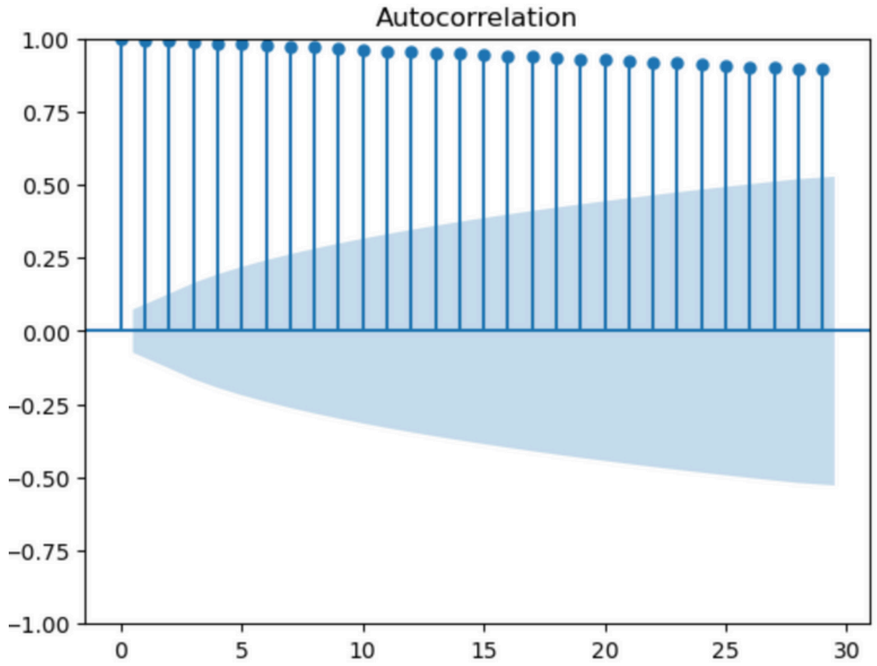
	Date	Close/Last	Volume	Open	High	Low
0	02/19/2025	$414.77	24114200	$407.88	$415.49	$407.65
1	02/18/2025	$409.64	21423050	$408.00	$410.597	$406.50
2	02/14/2025	$408.43	22758460	$407.79	$408.91	$405.88
3	02/13/2025	$410.54	23891730	$407.00	$411.00	$406.36
4	02/12/2025	$409.04	19121730	$407.21	$410.75	$404.3673
...	...	...	...	...	...	...
2510	02/26/2015	$44.055	26524300	$43.99	$44.23	$43.89
2511	02/25/2015	$43.99	29749090	$43.95	$44.09	$43.80
2512	02/24/2015	$44.09	25262080	$44.15	$44.30	$43.92
2513	02/23/2015	$44.15	32510550	$43.70	$44.19	$43.65
2514	02/20/2015	$43.855	29710140	$43.51	$43.88	$43.29

2515 rows × 6 columns

Stationarity check

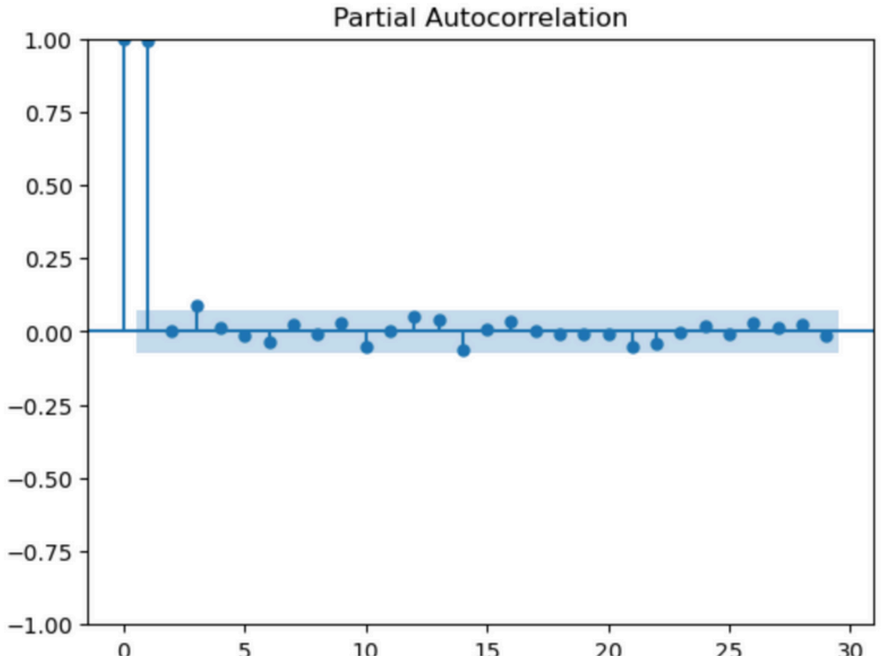
ACF PLOT
(Autocorrelation Function)

ACF is the correlation of the time series with its lags



PACF plot
(Partial Autocorrelation Function)

is the partial correlation of the time series with its lags, after removing the effects of lower-order-lags between them



From non-stationary to stationary : DIFFERENCING

What is Differencing?

Differencing is a technique used to make a non-stationary time series stationary.
This operation removes linear trends and reduces autocorrelation, eliminating the integrated component of the series.

It involves calculating the **difference between consecutive observations** in the series.

$$Y_t' = Y_t - Y_{t-1}$$

In other words, each value in the series is replaced by the difference between itself and the previous value.

LSTM (Long Short-Term Memory) and **XGBOOST don’t require stationary time series**, so the non-stationarity of the time serie is not an issue.

Our Methodology

Part 1: Forecasting model trained on Real Data

- ***Training Phase***: Train a forecasting model (LSTM NN or XGBOOST) using only the original (real) training dataset.
- ***Prediction:*** Apply the trained model to unseen test data to make predictions.
- ***Evaluate the model's performance*** using metrics such as Mean Squared Error (MSE) and Mean Absolute Error (MAE).

Part 2: Forecasting model trained on augmented Data

- ***Synthetic Data Generation:*** Build and train a TimeGAN Combine the synthetic data with the original training dataset to create an augmented training dataset.
- **Training Phase:** Train a new model on the augmented training dataset.
- **Prediction and Evaluation:** Apply the new model model to the same unseen test data to make predictions. Evaluate its performance using the same metrics (MSE and MAE).

Part 3: Comparison and Final Considerations

Compare the performance of the two forecasting models:
Measure whether the inclusion of synthetic data leads to an improvement in performance.

Dataset preparation - The approach

	Target Date	Target - 5	Target - 4	Target - 3	Target - 2	Target - 1	Target
0	2015-02-27	43.855	44.150	44.090	43.990	44.055	43.850
1	2015-03-02	44.150	44.090	43.990	44.055	43.850	43.880
2	2015-03-03	44.090	43.990	44.055	43.850	43.880	43.280
3	2015-03-04	43.990	44.055	43.850	43.880	43.280	43.055
4	2015-03-05	44.055	43.850	43.880	43.280	43.055	43.110
...	...	...	...	...	...	...	...
2505	2025-02-12	413.290	415.820	409.750	412.220	411.440	409.040
2506	2025-02-13	415.820	409.750	412.220	411.440	409.040	410.540
2507	2025-02-14	409.750	412.220	411.440	409.040	410.540	408.430
2508	2025-02-18	412.220	411.440	409.040	410.540	408.430	409.640
2509	2025-02-19	411.440	409.040	410.540	408.430	409.640	414.770

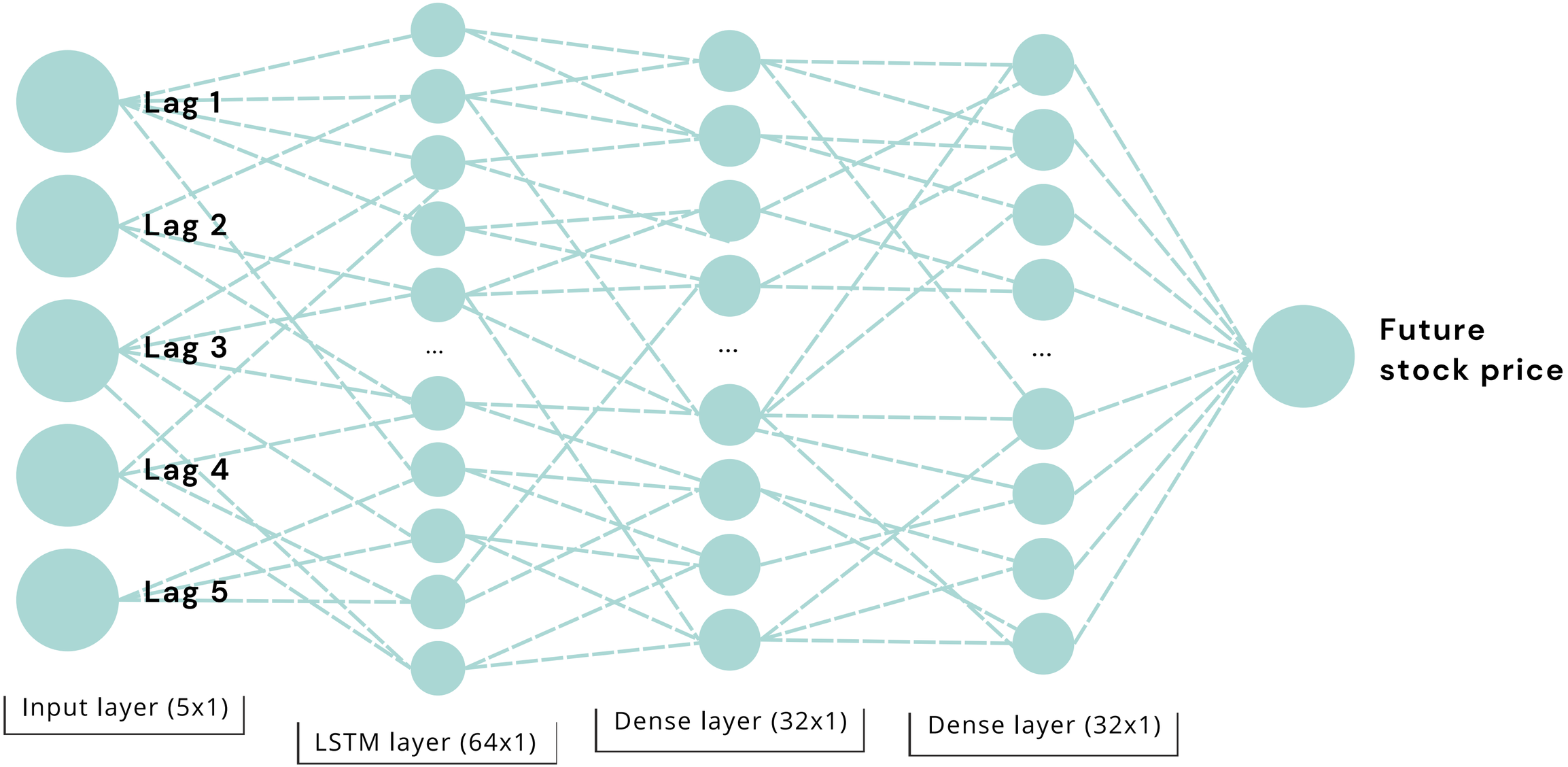
2510 rows × 7 columns

To predict the target value on a given day t, the dataset utilizes data from the previous five days (t-1 to t-5). These past values serve as features, helping to capture trends and patterns in the time series.

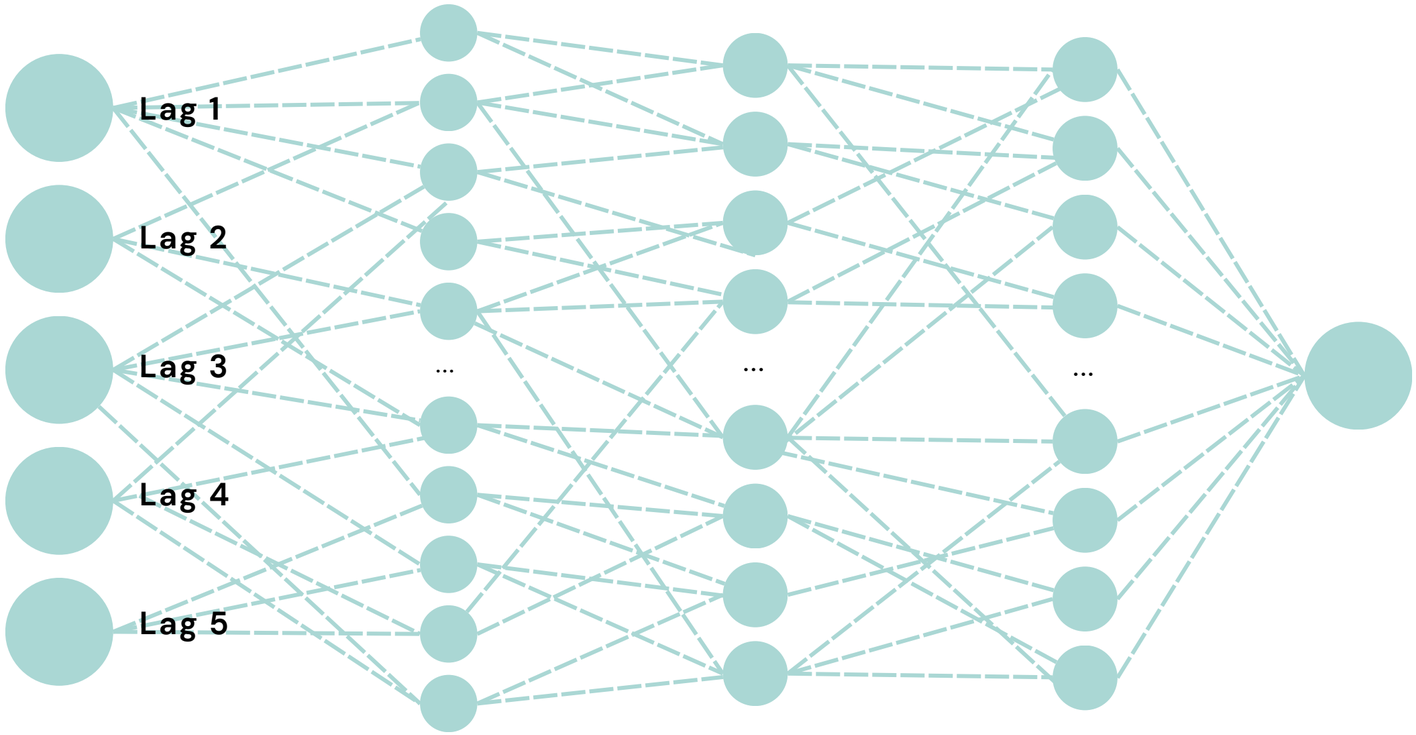
We also take into account the volume of transactions and the previous day's highs and lows values.

Neural network for LSTM

NN Structure



Neural network for LSTM



Number of epochs

The model is trained for 100 iterations over the dataset.

Evaluation metrics:

Mean absolute error.

Loss function:

Huber Loss that's a mix between MSE and MAE. It is robust to outliers than MSE.

Optimizer:

Adam with a learning rate of 0.0001 for stable learning.

Performance assessment :

MSE:
Mean Squared Error

- Calculates the average of the squared differences between predicted and actual values
- The MSE **penalizes larger errors more heavily** than smaller ones (due to squaring)

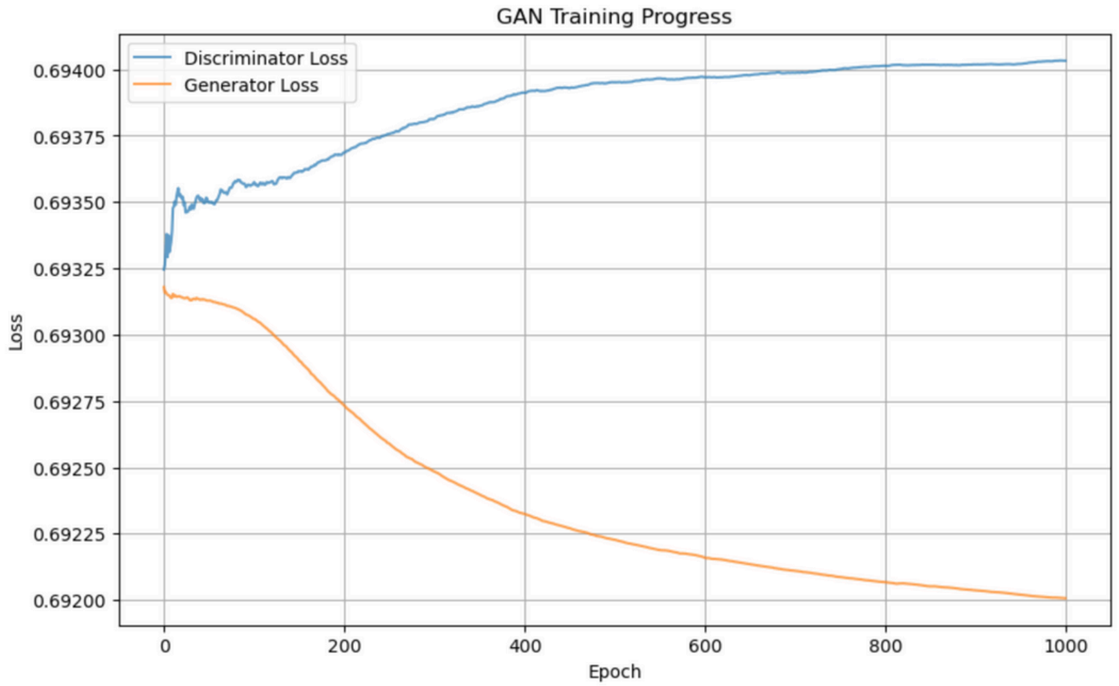
MAE:
Mean Absolute Error

- Calculates the average of the absolute differences between predicted and actual values
- **Treats all errors equally** regardless of their magnitude
- More robust to outliers compared to MSE

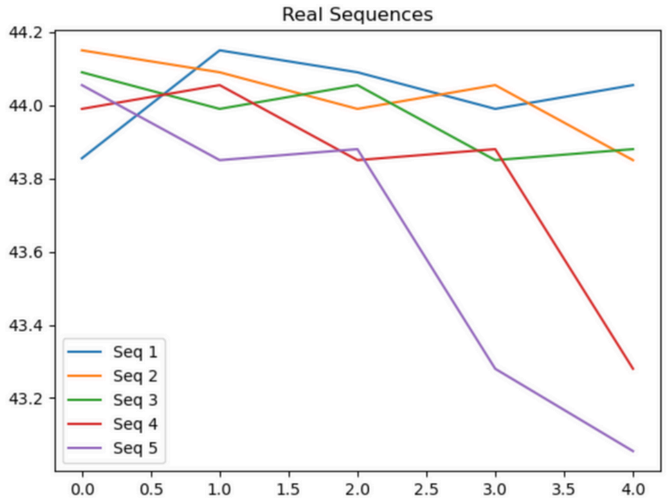
RMSE:
Root Mean Squared Error

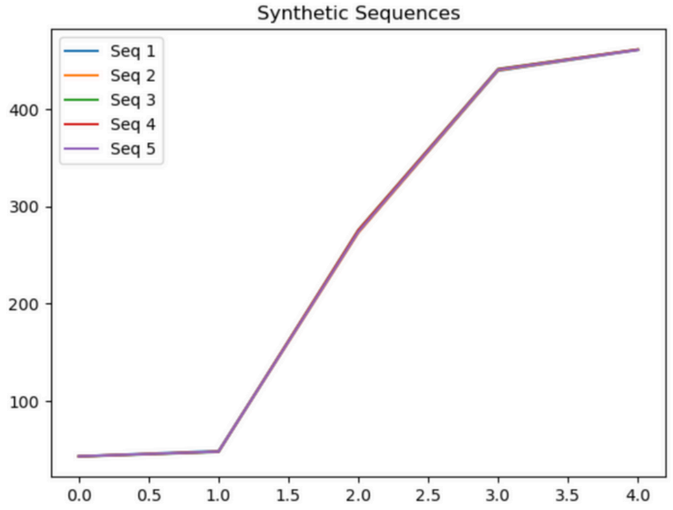
- Maintains the same properties as MSE but converts the result back to the same unit as the target variable
- **More interpretable than MSE** since it uses the same scale as the original variable

LSTM (univariate) + TimeGAN



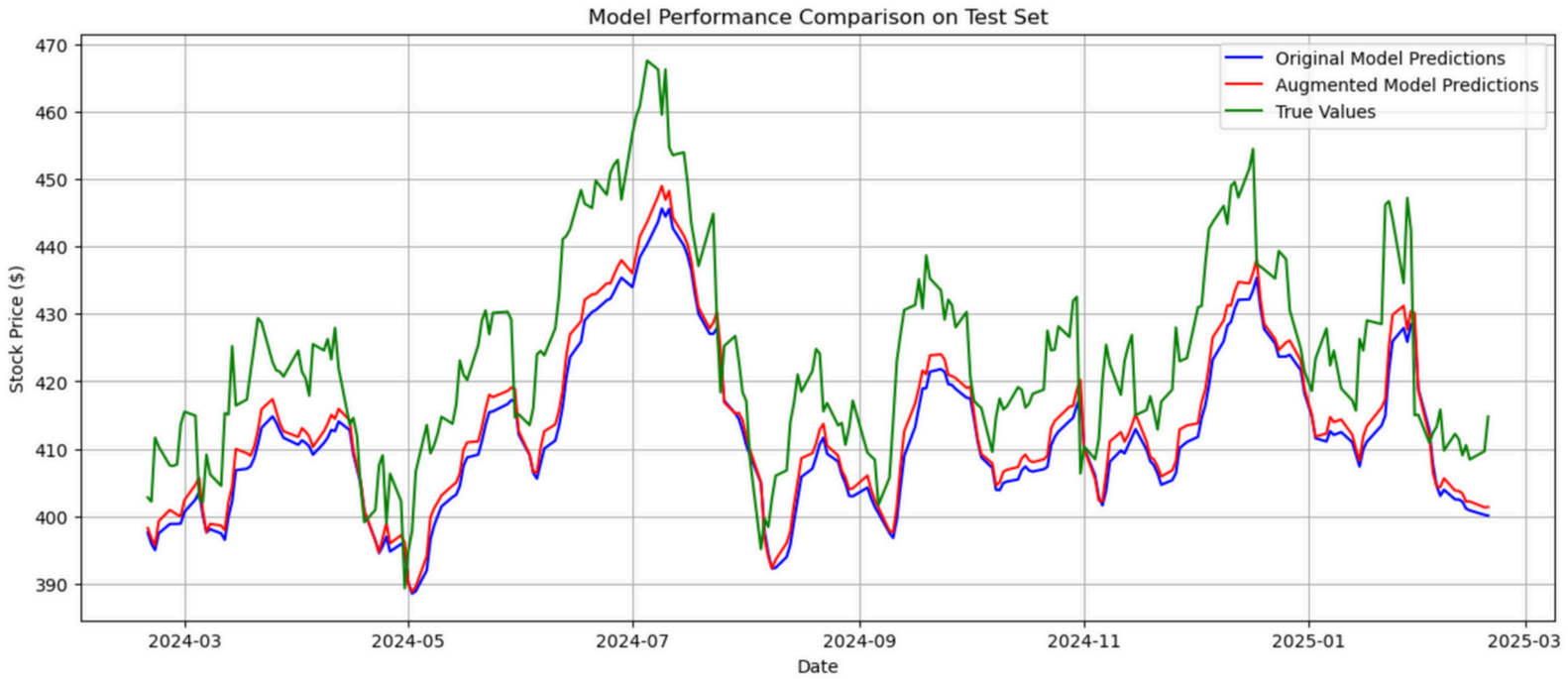

```
num_synthetic = int(X_train_orig.shape[0] * 0.5)  # 50% of synthetic data
```





The Generator loss decreases, while the Discriminator loss increases

LSTM (univariate) + TimeGAN

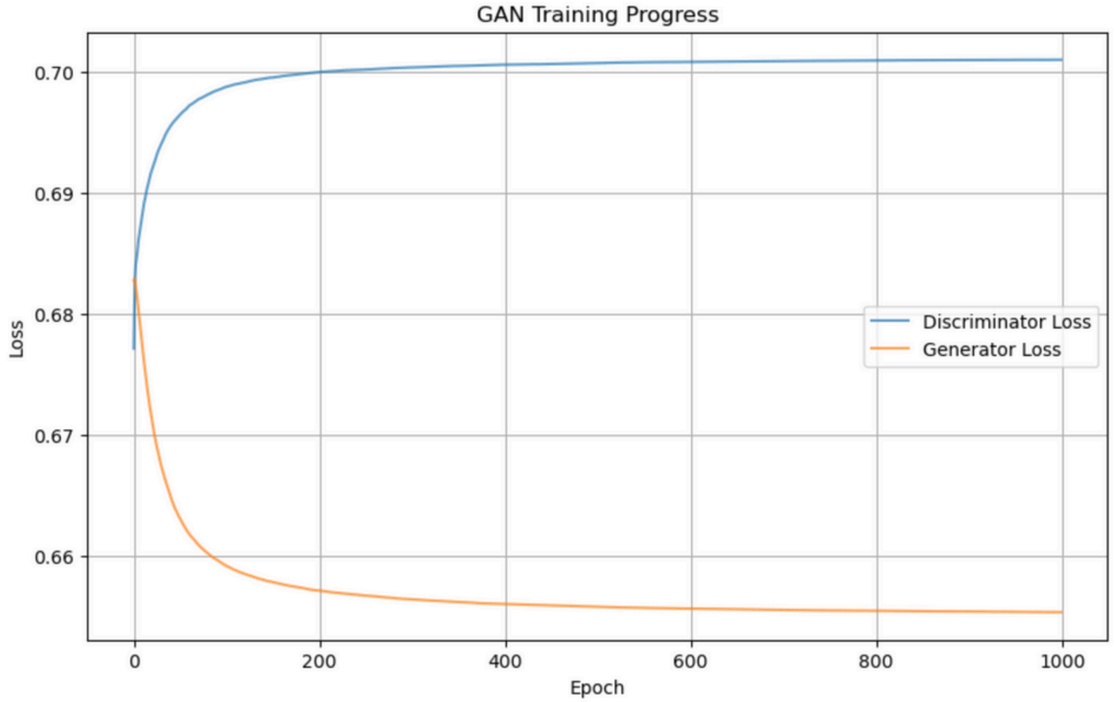



```
Test Set Performance Metrics:

Original Model:
MAE: $12.10
RMSE: $13.50

Augmented Model:
MAE: $10.55
RMSE: $11.85

Improvement with Data Augmentation:
MAE Improvement: 12.79%
RMSE Improvement: 12.21%
```

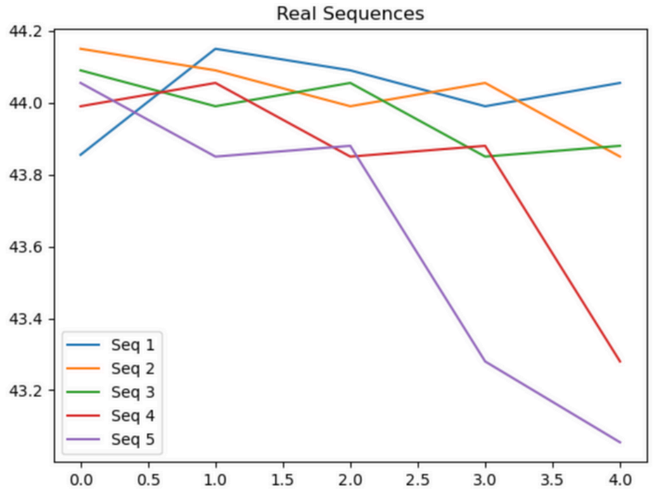
The **augmented model** (red line) is closer to the true data

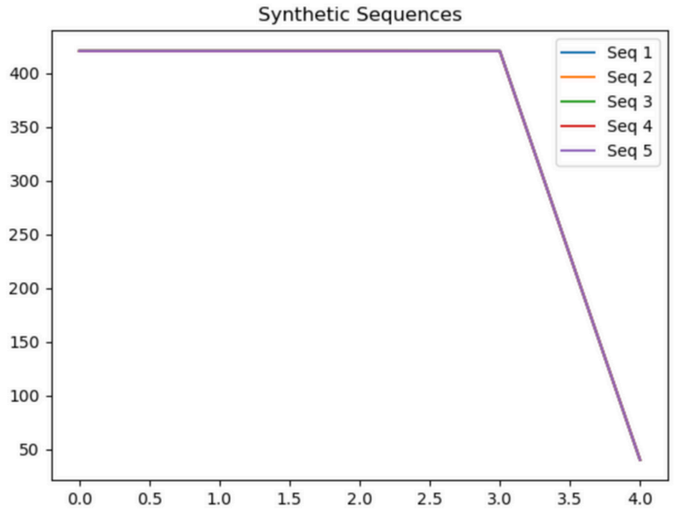
The **overall performance** of the augmented model is **higher** in terms of as MSE or RMSE

XGBOOST (univariate) + TimeGAN




```
num_synthetic = int(X_train_orig.shape[0] * 0.5)  # 50% of synthetic data
```

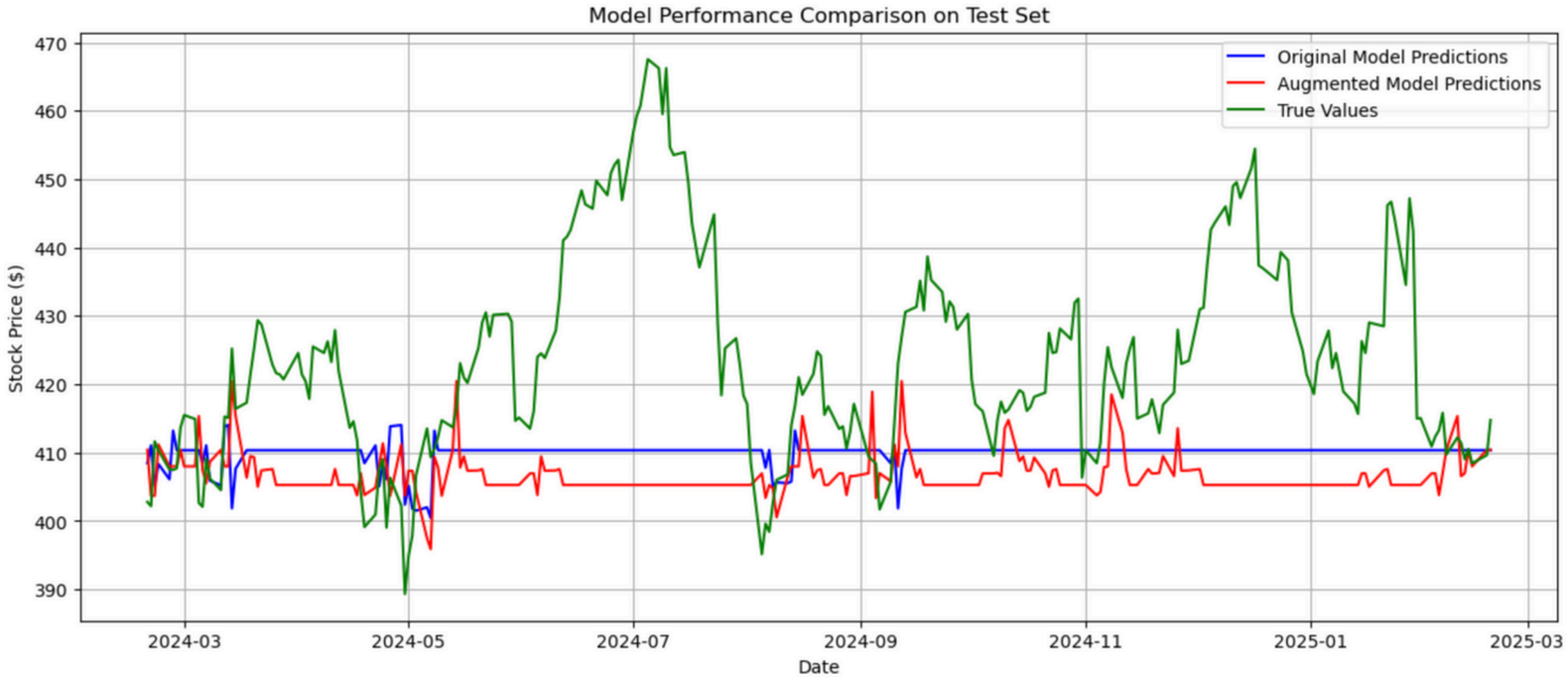




The Generator loss decreases, while the Discriminator loss increases

```
Epoch 0: D Loss: 0.6771, G Loss: 0.6827
Epoch 100: D Loss: 0.6987, G Loss: 0.6592
Epoch 200: D Loss: 0.6999, G Loss: 0.6571
Epoch 300: D Loss: 0.7003, G Loss: 0.6564
Epoch 400: D Loss: 0.7005, G Loss: 0.6560
Epoch 500: D Loss: 0.7007, G Loss: 0.6558
Epoch 600: D Loss: 0.7008, G Loss: 0.6556
Epoch 700: D Loss: 0.7008, G Loss: 0.6555
Epoch 800: D Loss: 0.7009, G Loss: 0.6555
Epoch 900: D Loss: 0.7009, G Loss: 0.6554
```

XGBOOST (univariate) + TimeGAN



Test Set Performance Metrics:

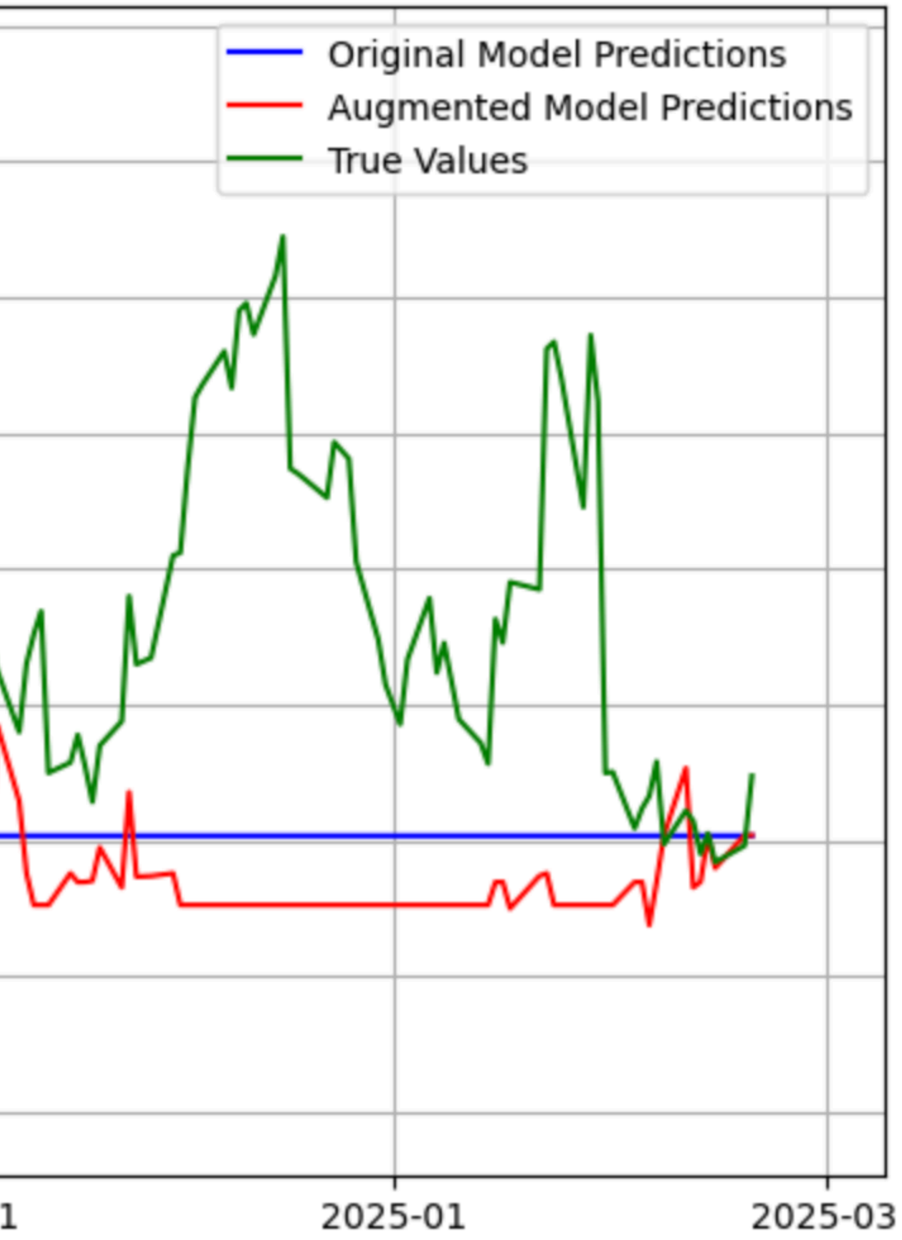
Original Model:
MAE: $15.19
RMSE: $19.95

Augmented Model:
MAE: $18.12
RMSE: $23.10

Improvement with Data Augmentation:
MAE Improvement: -19.30%
RMSE Improvement: -15.76%

The **augmented model** (red line) actually shows **more variability** than the original model (blue line)

The **overall performance** of the augmented model is **lower** in terms of as MSE or RMSE



Final considerations

Although the absolute forecast error may be higher, the higher variability of the augmented model is a significant advantage:

- The more frequent fluctuations reflect more closely the **volatile nature** of the real stock price (green line).
- **Reduces the risk of overfitting** on **oversimplified patterns**
- Demonstrates **greater responsiveness to market changes**, even if it does not fully capture the magnitude of actual fluctuations
- **Avoids the constant (or near-constant) forecasting problem** that characterizes the original model

With more in-depth fine-tuning, it is likely that the augmented model also significantly improve in terms of predictive performance.

Optimization of TimeGAN hyperparameters data

More sophisticated feature engineering, including technical trading indicators and time series-derived features

More complex XGBOOST models or hybrid models combining XGBoost with RNN

LSTM VS XGBOOST

LSTM

VS

XGBOOST

```
Improvement with Data Augmentation:
MAE Improvement: 12.79%
RMSE Improvement: 12.21%
```

```
Improvement with Data Augmentation:
MAE Improvement: -19.30%
RMSE Improvement: -15.76%
```

The addition of synthetic data helped the recurrent network to generalize better.
On the other hand the **decision tree-based model did not benefit** from the synthetic data.

Intrigued by the results we try to fit and test a **MULTIVARIATE LSTM**, i.e., a model that can consider more input variables to make predictions.
The **TimeGAN** will now cr**eate synthetic data for 5 different variables**, expanding the information context of the model.

Our goal is to see if the implementation of more features further improves the performance of the LSTM.

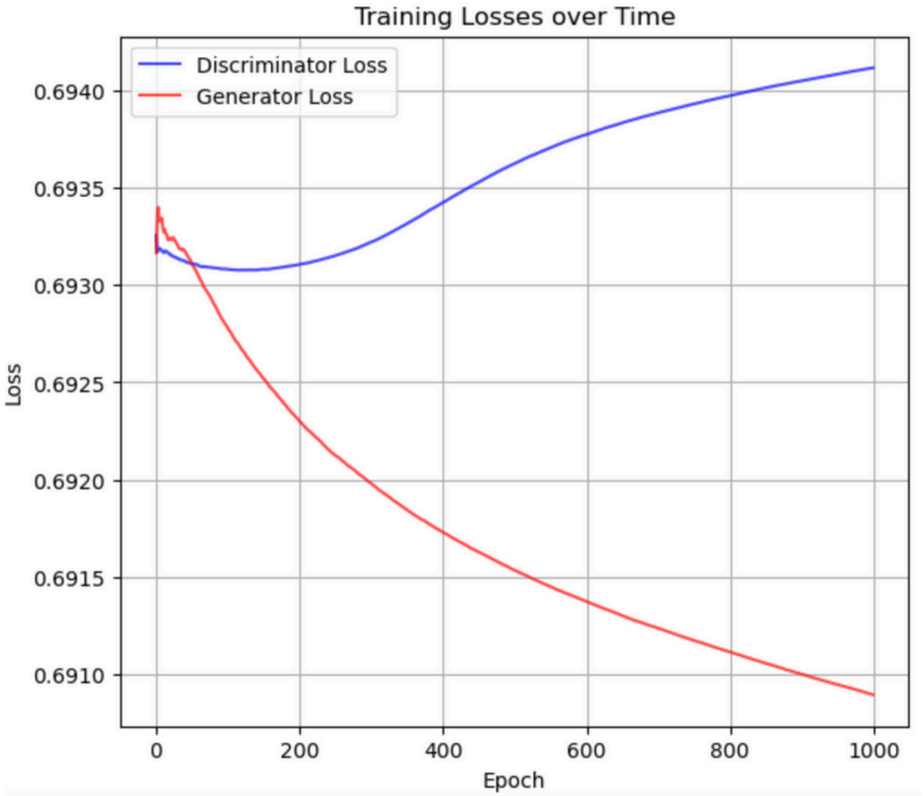
LSTM (multivariate) + TimeGAN
The approach:

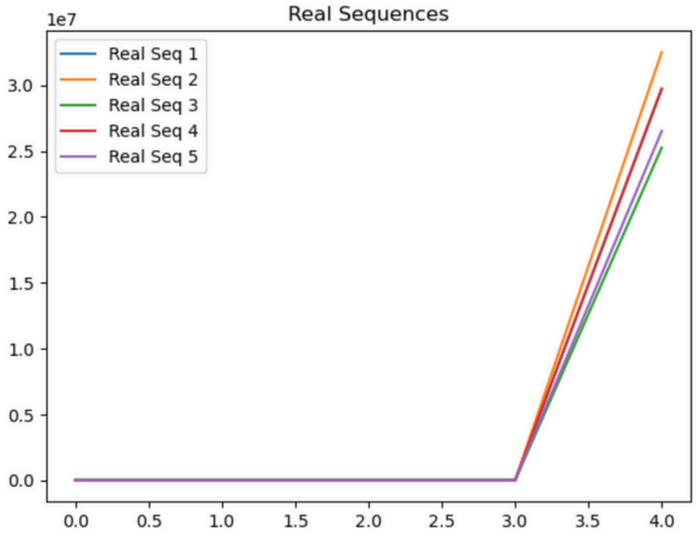
	Target Date	Open-1	High-1	Low-1	Close-1	Volume-1	Target
0	2015-02-23	43.51	43.880	43.2900	43.855	29710140.0	44.150
1	2015-02-24	43.70	44.190	43.6500	44.150	32510550.0	44.090
2	2015-02-25	44.15	44.300	43.9200	44.090	25262080.0	43.990
3	2015-02-26	43.95	44.090	43.8000	43.990	29749090.0	44.055
4	2015-02-27	43.99	44.230	43.8900	44.055	26524300.0	43.850
...	...	...	...	...	...	...	...
2509	2025-02-12	409.64	412.490	409.3000	411.440	18140590.0	409.040
2510	2025-02-13	407.21	410.750	404.3673	409.040	19121730.0	410.540
2511	2025-02-14	407.00	411.000	406.3600	410.540	23891730.0	408.430
2512	2025-02-18	407.79	408.910	405.8800	408.430	22758460.0	409.640
2513	2025-02-19	408.00	410.597	406.5000	409.640	21423050.0	414.770

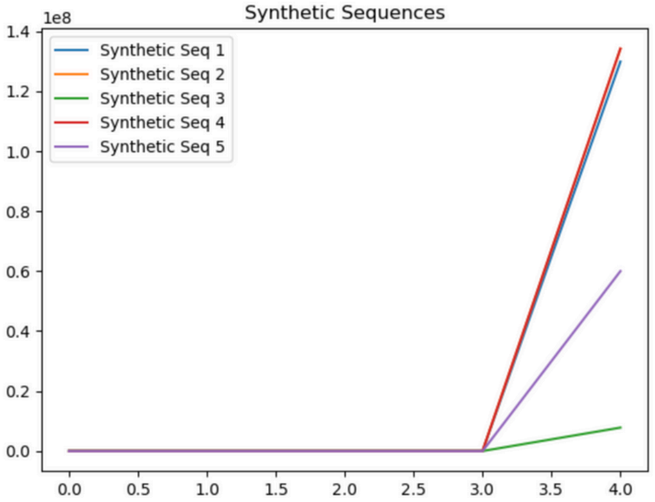
2514 rows × 7 columns

To predict the target value on a given day t, the dataset utilizes data from the previous five days (t-1 to t-5). These past values serve as features, helping to capture trends and patterns in the time series.

LSTM (multivariate) + TimeGAN

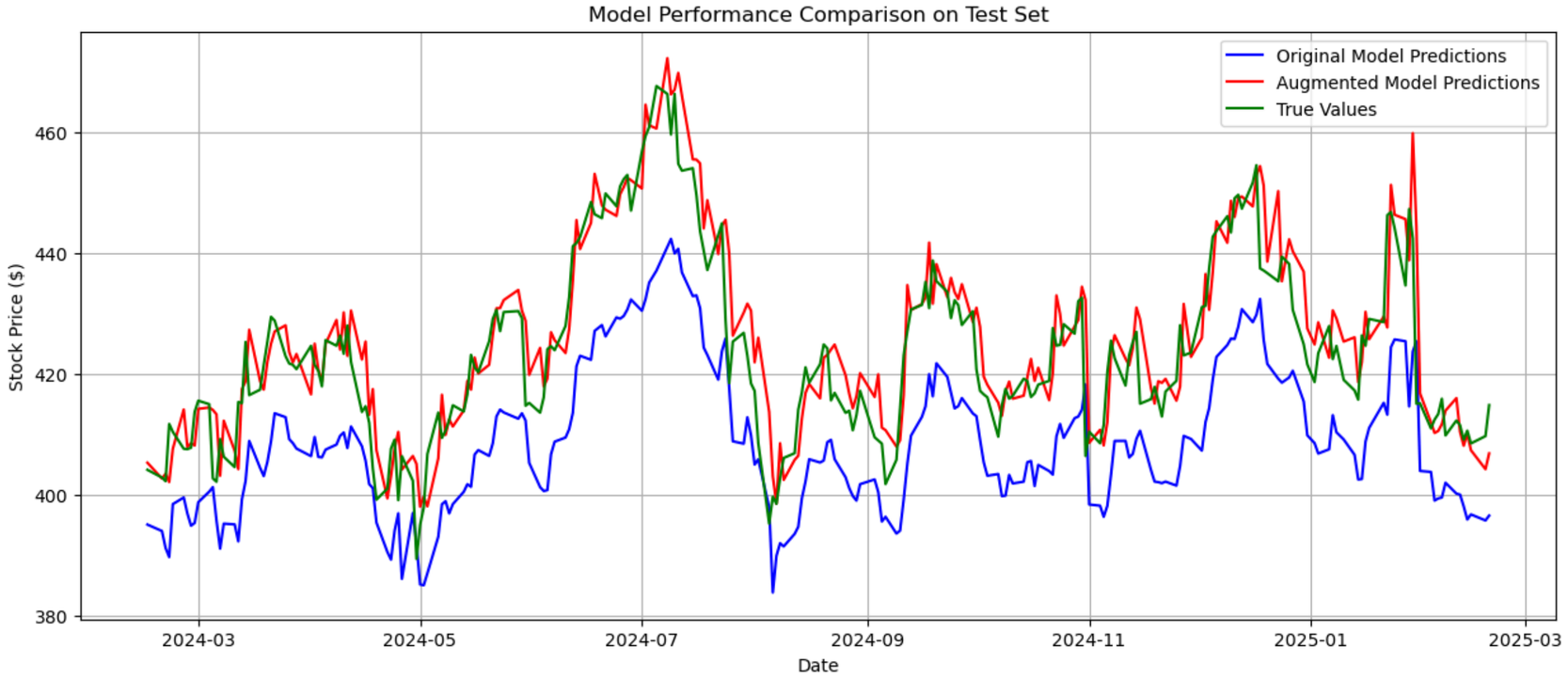






The Generator loss decreases, while the Discriminator loss increases

LSTM (multivariate) + TimeGAN



Test Set Performance Metrics:

Original Model:
MAE: $15.18
RMSE: $16.24

Augmented Model:
MAE: $5.06
RMSE: $6.82

Improvement with Data Augmentation:
MAE Improvement: 66.70%
RMSE Improvement: 58.02%

References:

- Yoon, J., Jarrett, D., & van der Schaar, M. (2019). *Time-series Generative Adversarial Networks (TimeGAN).* 33rd Conference on Neural Information Processing Systems (NeurIPS), Vancouver, Canada.

- M. Nabipour, P. Nayyeri, H. Jabani, A. Mosavi, E. Salwana e Shahab S. (2020). *Deep Learning for Stock Market Prediction* . Entropy nel 2020, volume 22, page 840.

Thank you very much!