PREDICTING STOCK PRICES USING RECURRENT NEURAL NETWORKS

Adrian P. Bustamante, Ph.D.
adrianpebus@gmail.com

OUTLINE

Objective

About the data

Data preparation

RNN models

Summary of Results

Conclusion and next steps.

OBJECTIVE

We consider prices/volume of IBM stocks in a period of around 1000 days, between January 2020 and April 2024.

The objective is to use Recurrent Neural Networks (RNN) to predict the stock prices and volume for 90 days (about 10% of the data).

We consider a few simple RNN architectures, including LSTM and GRU.

We also use keras tuner to optimize hyperparameters (Dropout, units, optimizer, and learning rate).

A Jupyter Notebook with the full implementation can be found at:

https://github.com/adrian-pbustamante/Predicting-stock-prices-with-Recurrent-Neural-Networks/blob/main/RNN-stock-prices.ipynb

ABOUT THE DATA

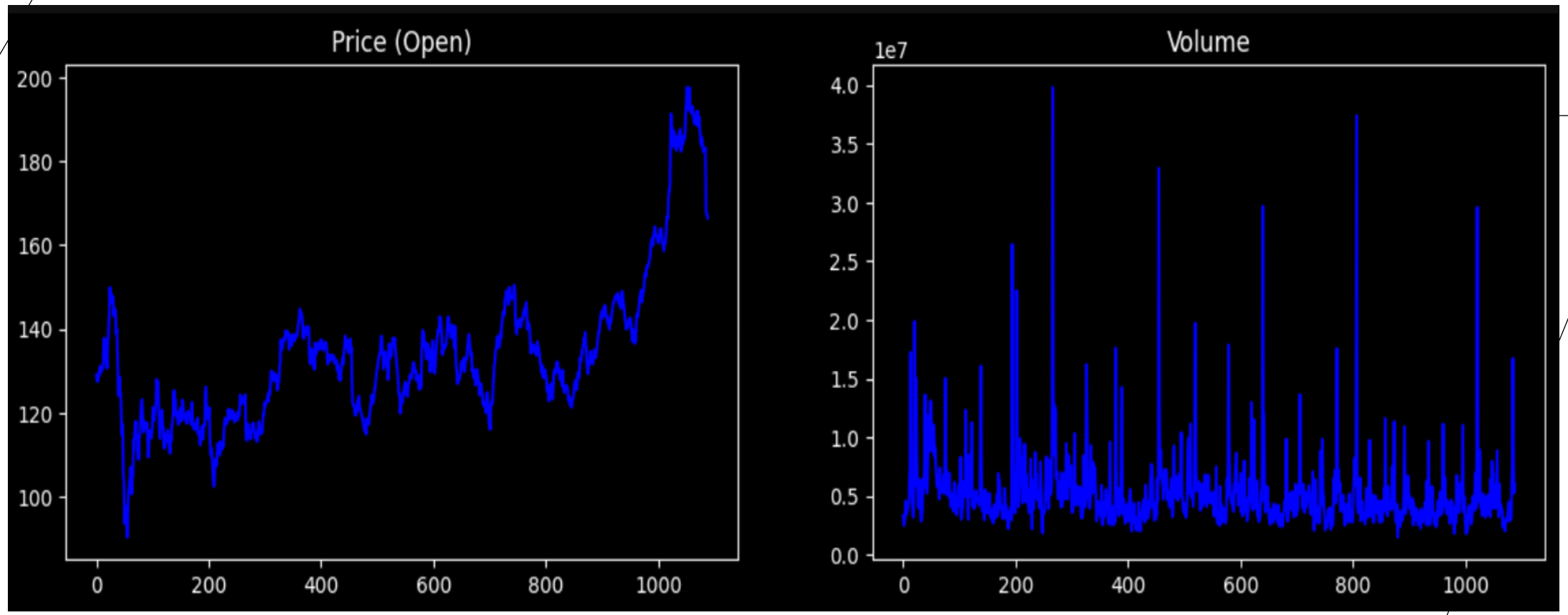
International Business Machines Corporation, together with its subsidiaries, provides integrated solutions and services worldwide. The company operates through Software, Consulting, Infrastructure, and Financing segments. The Software segment offers a hybrid cloud and AI platforms that allows clients to realize their digital and AI transformations across the applications, data, and environments in which they operate. The Consulting segment focuses on skills integration for strategy, experience, technology, and operations by domain and industry. The Infrastructure segment provides on-premises and cloud based server, and storage solutions, as well as life-cycle services for hybrid cloud infrastructure deployment. The Financing segment offers client and commercial financing, facilitates IBM clients' acquisition of hardware, software, and services. International Business Machines Corporation was incorporated in 1911 and is headquartered in Armonk, New York.

This dataset contains historical stock price data for International Business Machines Corporation (IBM) from [Jan/01/2020] to [May/01/2024]. The dataset includes daily closing prices, adjusted closing prices, and other relevant information. Features:

`Date, Open, High, Low, Close, Adj Close, Volume.`

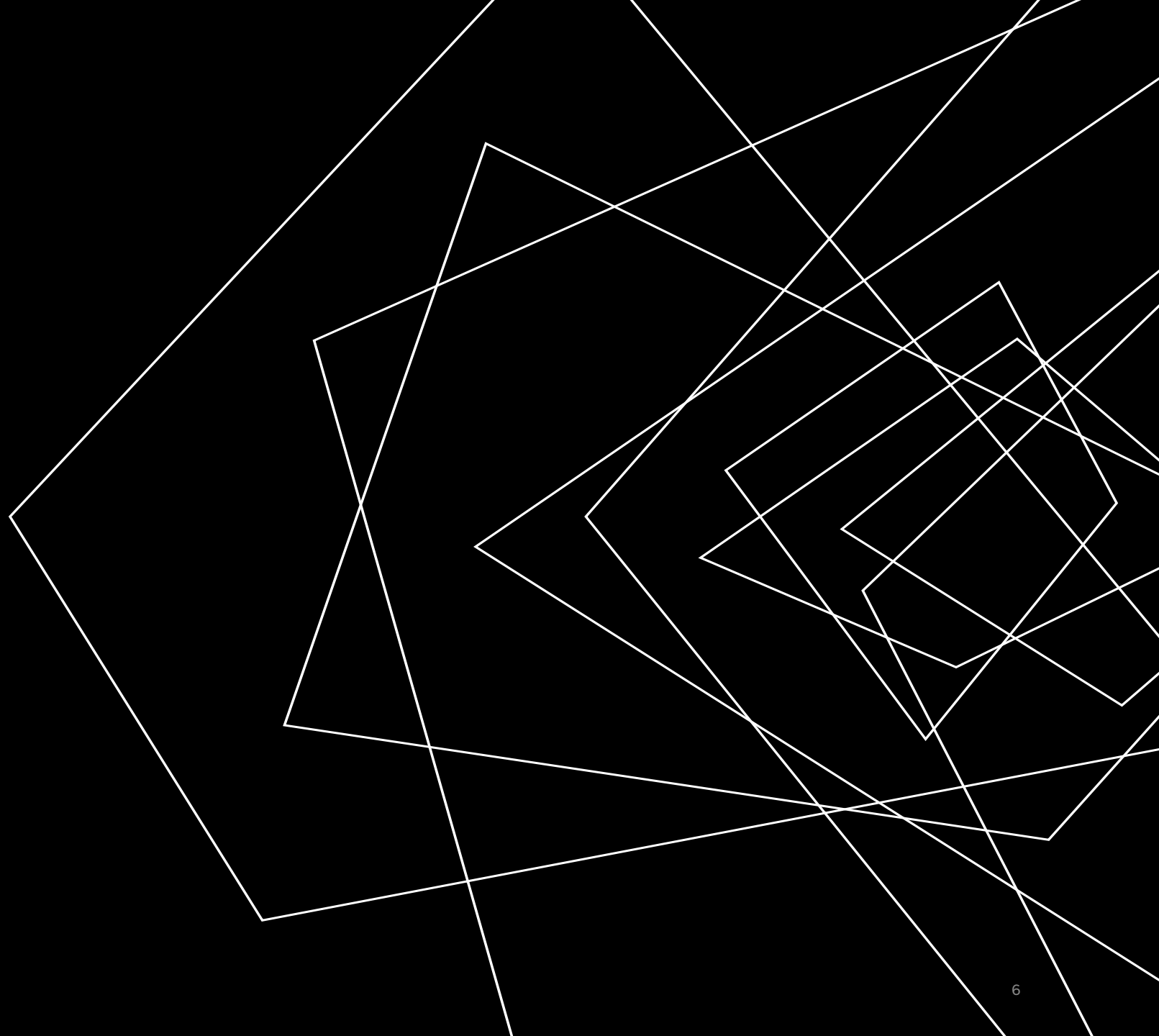
Source: https://www.kaggle.com/datasets/innocentmfa/ibm-stock-prices-dataset

TARGET FEATURES

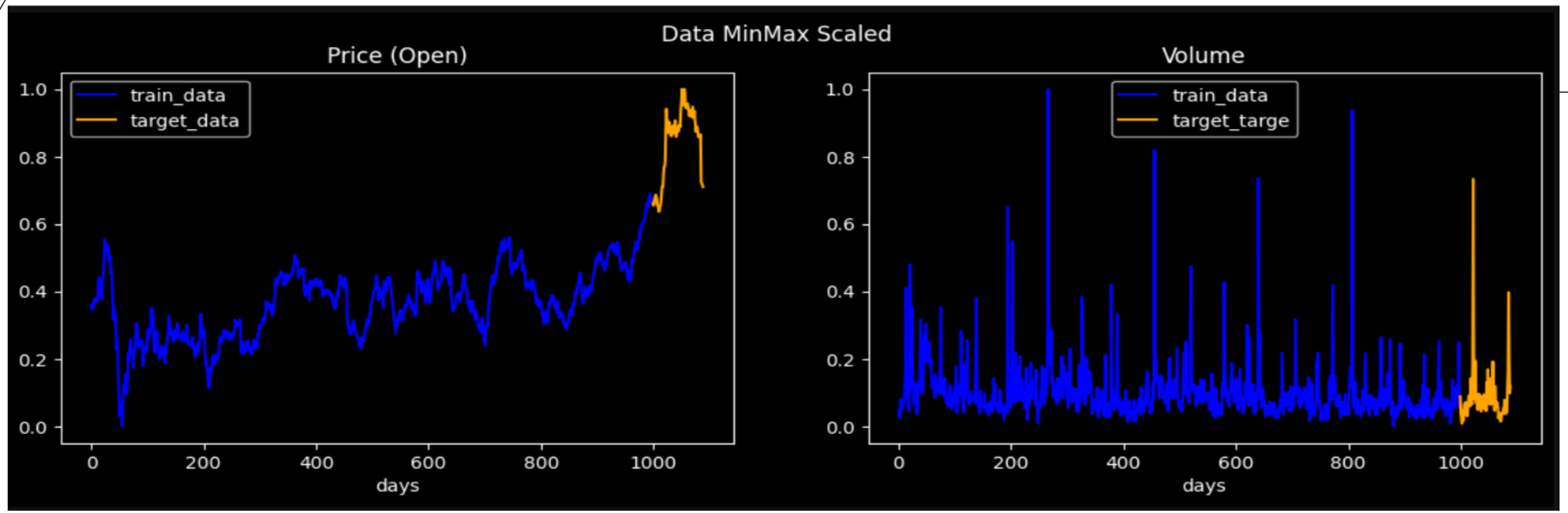


The objective is to predict the last 90 days of the Price and Volume of the Stocks.

DATA PREPARATION

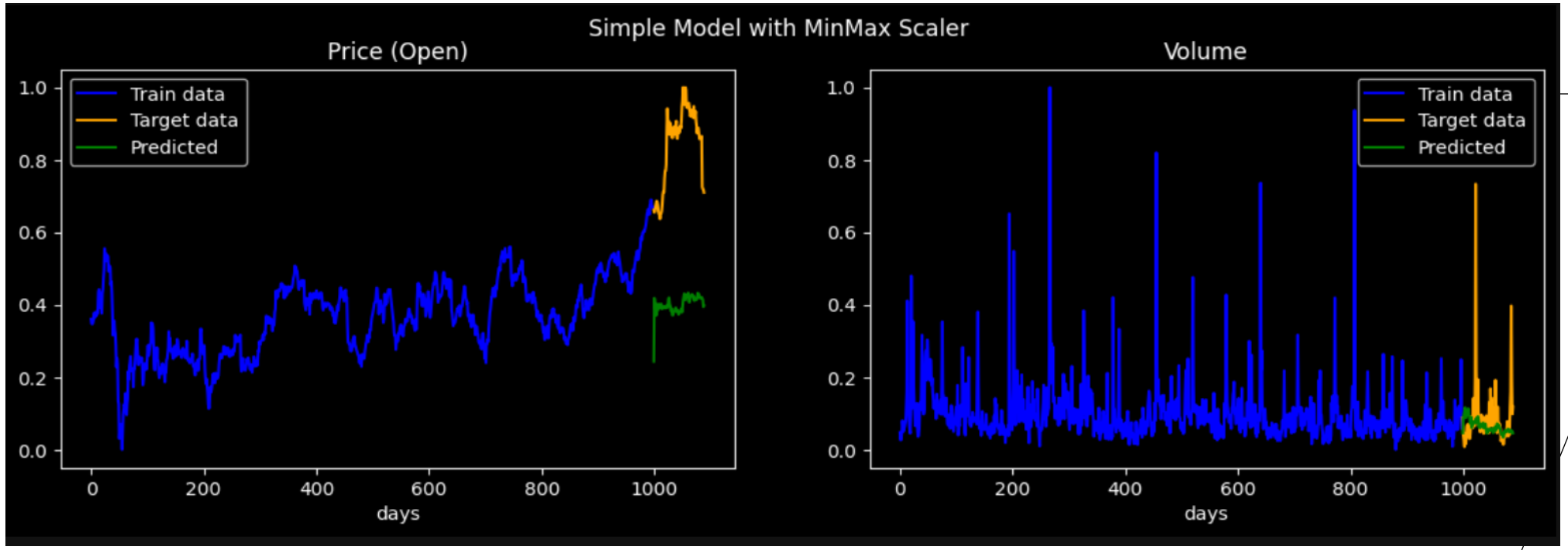


PROBLEMS WITH MINMAX SCALER



The Open Price of the stocks has an increasing trend. This implies that most of the values in the target set are out of the scale of the train set, which brings about the problem that the Recurrent Networks we will implement need to predict some numbers that have not been seen before. We can see this by training a simple RNN:

SIMPLE RNN WITH MINMAX SCALED DATA



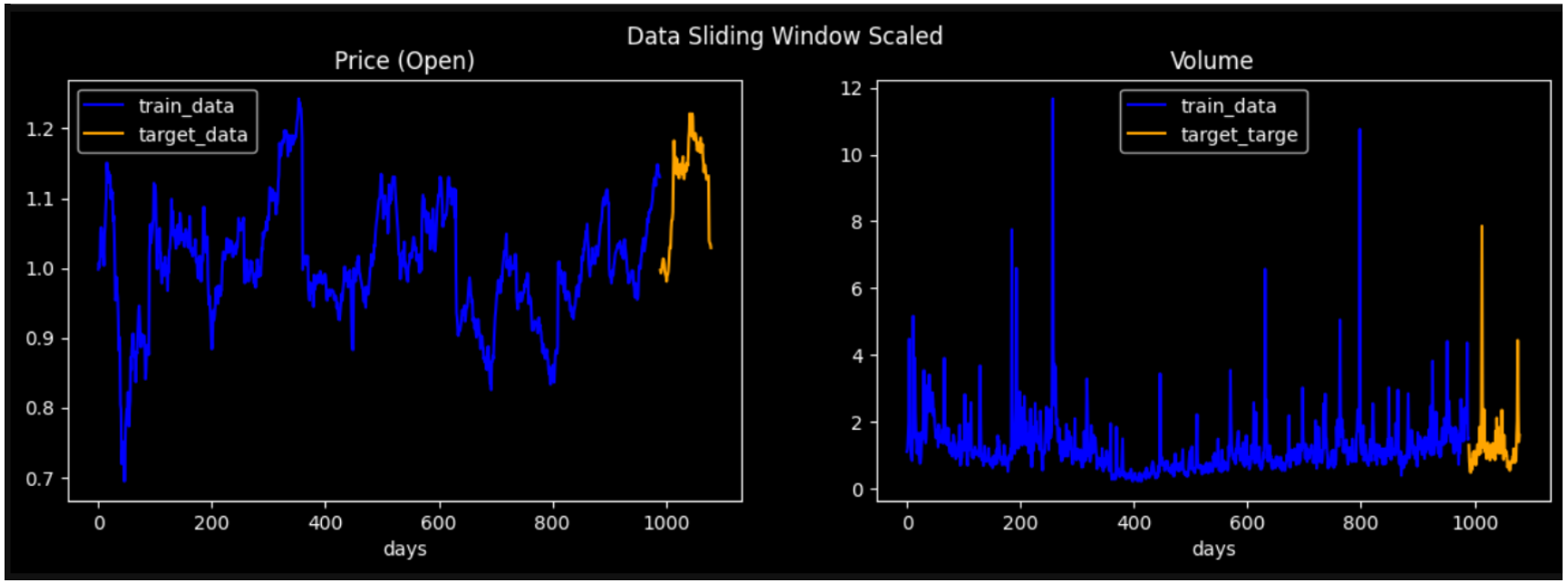
We can see in the figure that the predicted values, for the Price, are outside the range of the target set (and inside the range of the train set).

SLIDING WINDOW SCALER

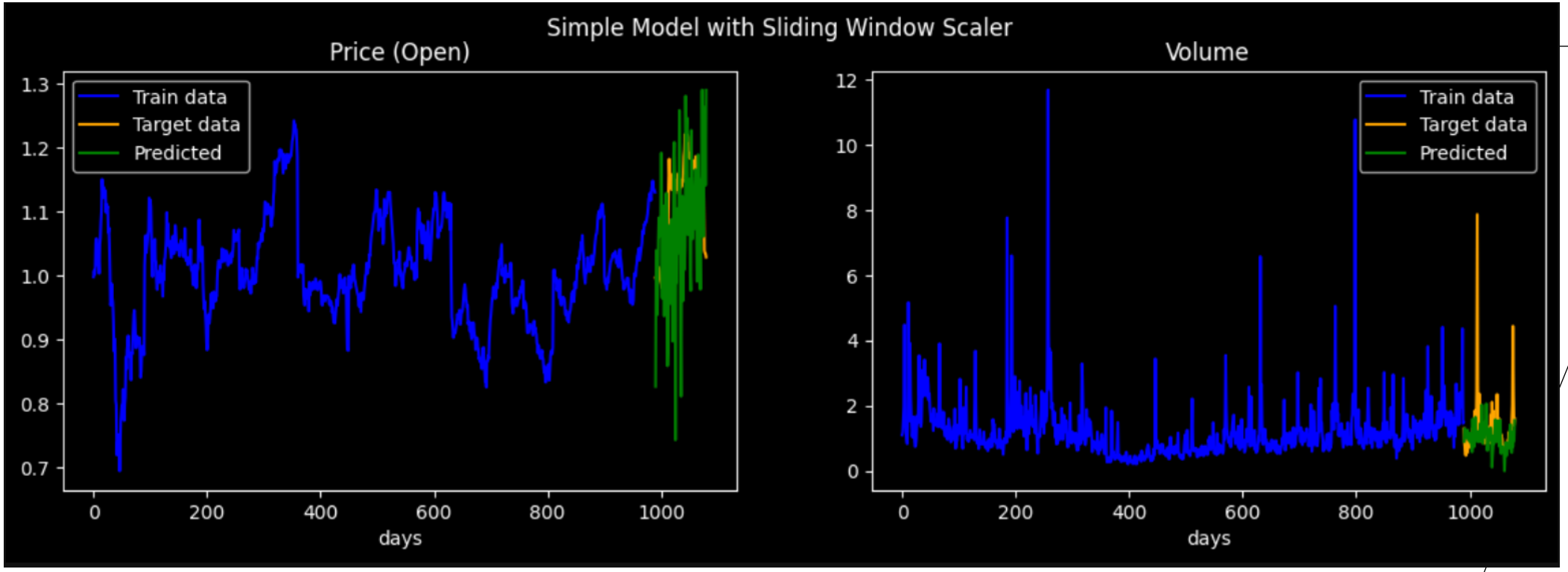
To solve this issue, we normalize the data in windows of size ℓ as follows:

$$
\begin{aligned}
&\vdots \\
W_t &= \left(\frac{p_{t\ell}}{p_{t\ell-1}}, \frac{p_{t\ell+1}}{p_{t\ell-1}}, \ldots, \frac{p_{(t+1)\ell-1}}{p_{t\ell-1}} \right) \\
W_{t+1} &= \left(\frac{p_{(t+1)\ell}}{p_{(t+1)\ell-1}}, \frac{p_{(t+1)\ell+1}}{p_{(t+1)\ell-1}}, \ldots, \frac{p_{(2t+1)\ell-1}}{p_{(t+1)\ell-1}} \right) \\
&\vdots
\end{aligned}
$$

We can see below that, scaling the data using sliding windows, now the target set is within the range of the train set.



SIMPLE RNN WITH SLIDING WINDOW SCALER



We can see that with the sliding window normalization the predicted data, using a simple RNN, is in the range of the target set.

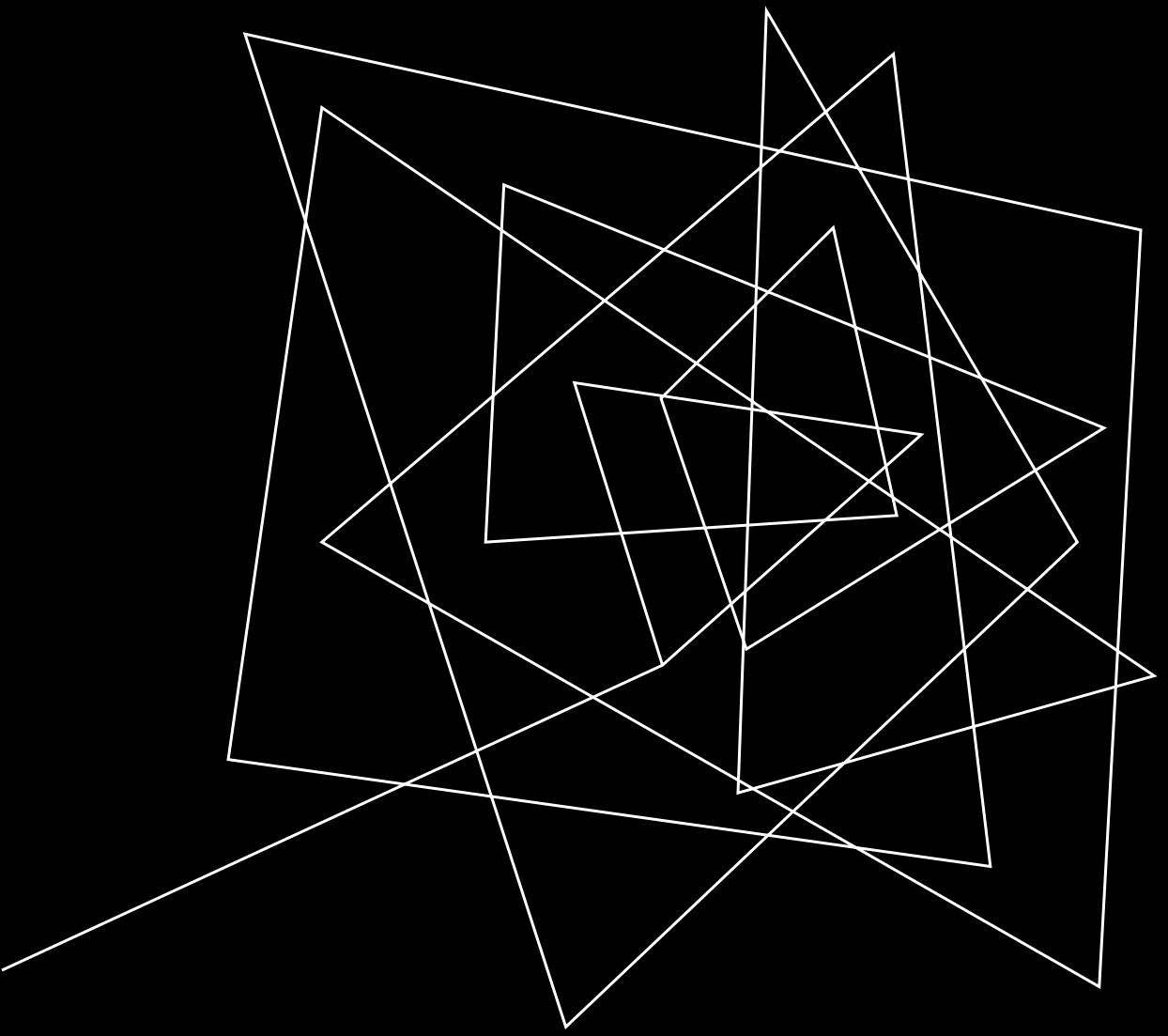
WRITING THE DATA AS SEQUENCES

Stock prices/volumes are time series of length N, i.e., $p_0, p_1, p_2, \ldots, p_N$, where p_i =(price, volume) on the day i, $0 \leq i \leq N$. RNN are mainly used to deal with sequential data. The models that we are about to construct use values in sequential form, i.e., we construct sequences of the form

$$\begin{aligned}
X_0 &= [p_0, p_1, \ldots, p_{\ell-1}], & y_0 &= [p_\ell, p_{\ell+1}, \ldots, p_{2\ell-1}] \\
X_1 &= [p_1, p_2, \ldots, p_\ell], & y_1 &= [p_{\ell+1}, p_{\ell+2}, \ldots, p_{2\ell}] \\
X_2 &= [p_2, p_2, \ldots, p_{\ell+1}], & y_2 &= [p_{\ell+2}, p_{\ell+3}, \ldots, p_{2\ell+1}] \\
&\vdots & &\vdots \\
X_{N-2\ell+1} &= [p_{N-2\ell+1}, p_{N-2\ell+2}, \ldots, p_{N-\ell}], & y_{N-2\ell+1} &= [p_{N-\ell+1}, p_{\ell+2}, \ldots, p_N]
\end{aligned}$$

where X_i are the inputs to the network, y_i are the respective targets, and ℓ is the number of days that we want to predict. With this notation, our objective is to use RNNs to predict prices/volume of the last $\ell(= length)$ days, that is, we want to predict $y_{N-2\ell+1} = [p_{N-\ell+1}, p_{\ell+2}, \ldots, p_N]$.

Note that the train set will be of the form $\{(X_i, y_i)\}_{i=0}^{m}$ with $m \leq N - 3\ell$. In this way, none of the values on the training set include any of the values we want to predict, i.e., $\{p_{N-\ell+1}, \ldots, p_N\}$.



RNN MODELS AND HYPERPARAMETER TUNING

MODELS CONSIDER

We consider three common types of recurrent Neural Networks:
-Simple RNN
-LSTM (Long Short-Term Memory)
-GRU (Gated Recurrent Units)

For each type of Neural Network we consider two Sequential architectures:
- Input/Hidden layer/Dropout/Output
- Input/Hidden layer/Hidden layer/Dropout/Output

We use Keras Tuner to optimize the hyperparameters:
-Number of Units per hidden layer
-Dropout rate
-Optimizer
-Learning rate

HYPERPARAMETER RANDOM SEARCH FOR SIMPLE AND DEEP RNN

For the simple RNN we RandomSearch hyperparameters as follows:
-Units on SimpleRNN hidden layer : [50, 75, 100, 125, 150]
-Dropout Rate: [0.0, 0.1, 0.2, 0.3]
-learning_rate: [0.001, 0.01, 0.1]
-optimizer: ['RMSprop', 'Adam']

For the deep RNN we RandomSearch hyperparameters as follows:
-Units on SimpleRNN first hidden layer : [50, 75, 100, 125, 150]
-Units on SimpleRNN second hidden layer : [50, 75, 100, 125, 150]
-Dropout Rate: [0.0, 0.1, 0.2, 0.3]
-learning_rate: [0.001, 0.01, 0.1]
-optimizer: ['RMSprop', 'Adam']

HYPERPARAMETER RANDOM SEARCH FOR SIMPLE AND DEEP LSTM

For the simple LSTM we RandomSearch hyperparameters as follows:

-Units on LSTM hidden layer : [50, 75, 100, 125, 150]
-Dropout Rate: [0.0, 0.1, 0.2, 0.3]
-learning_rate: [0.001, 0.01, 0.1]
-optimizer: ['RMSprop', 'Adam']

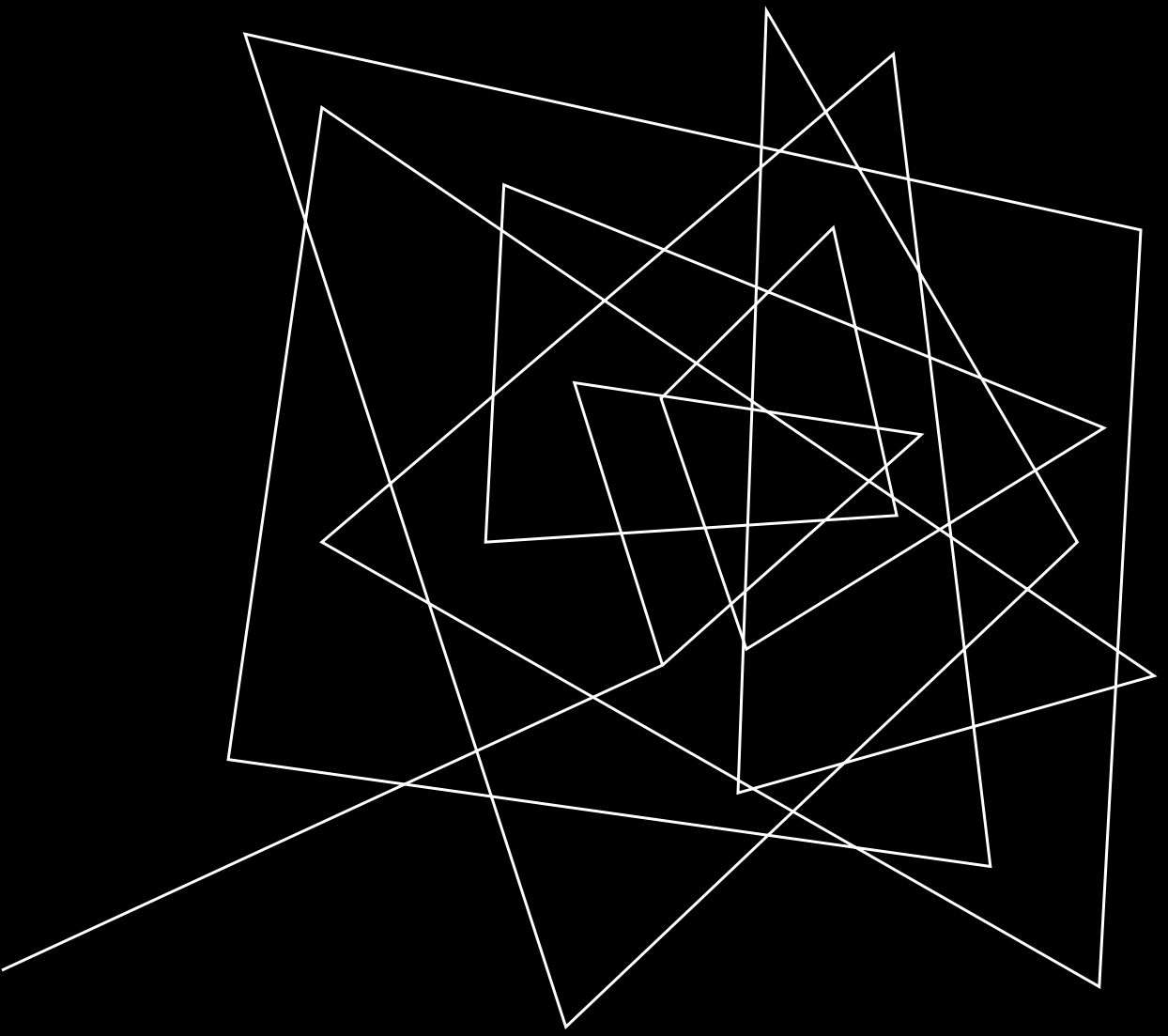
For the deep LSTM we RandomSearch hyperparameters as follows:

-Units on LSTM first hidden layer : [50, 75, 100, 125, 150]
-Units on LSTM second hidden layer : [50, 75, 100, 125, 150]
-Dropout Rate: [0.0, 0.1, 0.2, 0.3]
-learning_rate: [0.001, 0.01, 0.1]
-optimizer: ['RMSprop', 'Adam']

HYPERPARAMETER RANDOM SEARCH FOR SIMPLE AND DEEP GRU

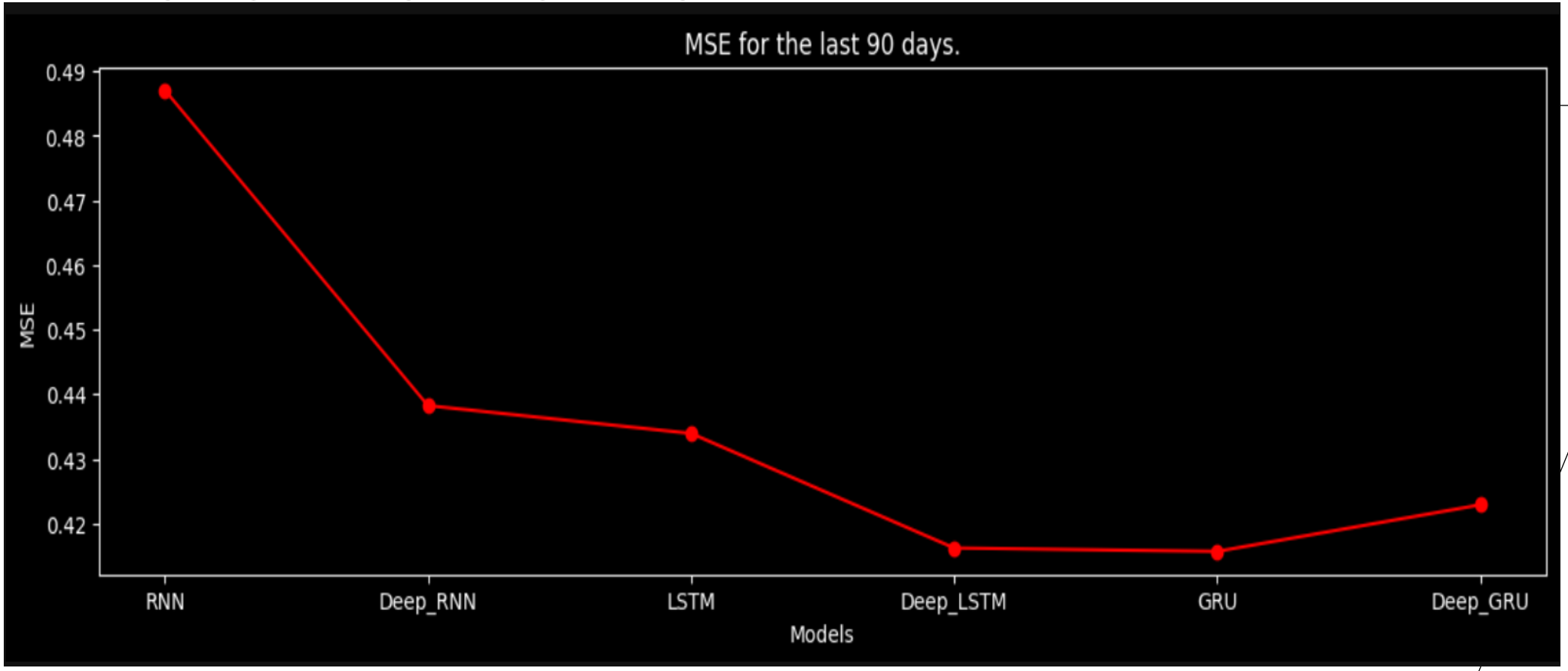
For the simple GRU we RandomSearch hyperparameters as follows:
-Units on GRU hidden layer : [50, 75, 100, 125, 150]
-Dropout Rate: [0.0, 0.1, 0.2, 0.3]
-learning_rate: [0.001, 0.01, 0.1]
-optimizer: ['RMSprop', 'Adam']

For the deep GRU we RandomSearch hyperparameters as follows:
-Units on GRU first hidden layer : [50, 75, 100, 125, 150]
-Units on GRU second hidden layer : [50, 75, 100, 125, 150]
-Dropout Rate: [0.0, 0.1, 0.2, 0.3]
-learning_rate: [0.001, 0.01, 0.1]
-optimizer: ['RMSprop', 'Adam']



SUMMARY OF RESULTS

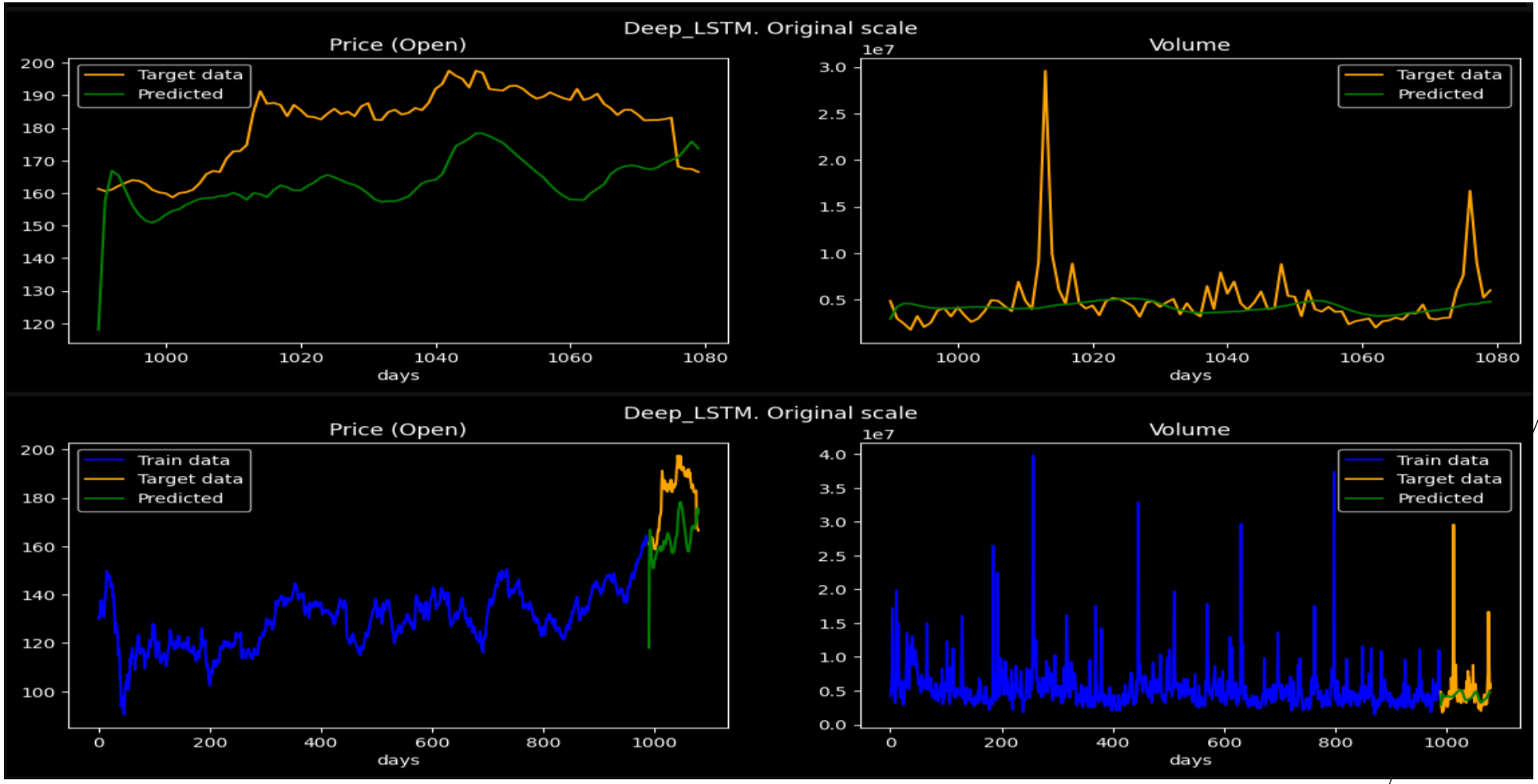
MSE OF BEST MODELS



Mean Squared Error (MSE) of predicted data for the last 90 days. This are the results of training the models using the best hyperparameters found.

DEEP LSTM

MSE for the last 90 days predicted: 0.41620735890886734
Best hyperparameters found for model Deep_LSTM: {'units_1': 150, 'units_2': 75, 'rate': 0.2, 'learning_rate': 0.001, 'optimizer': 'RMSprop'}

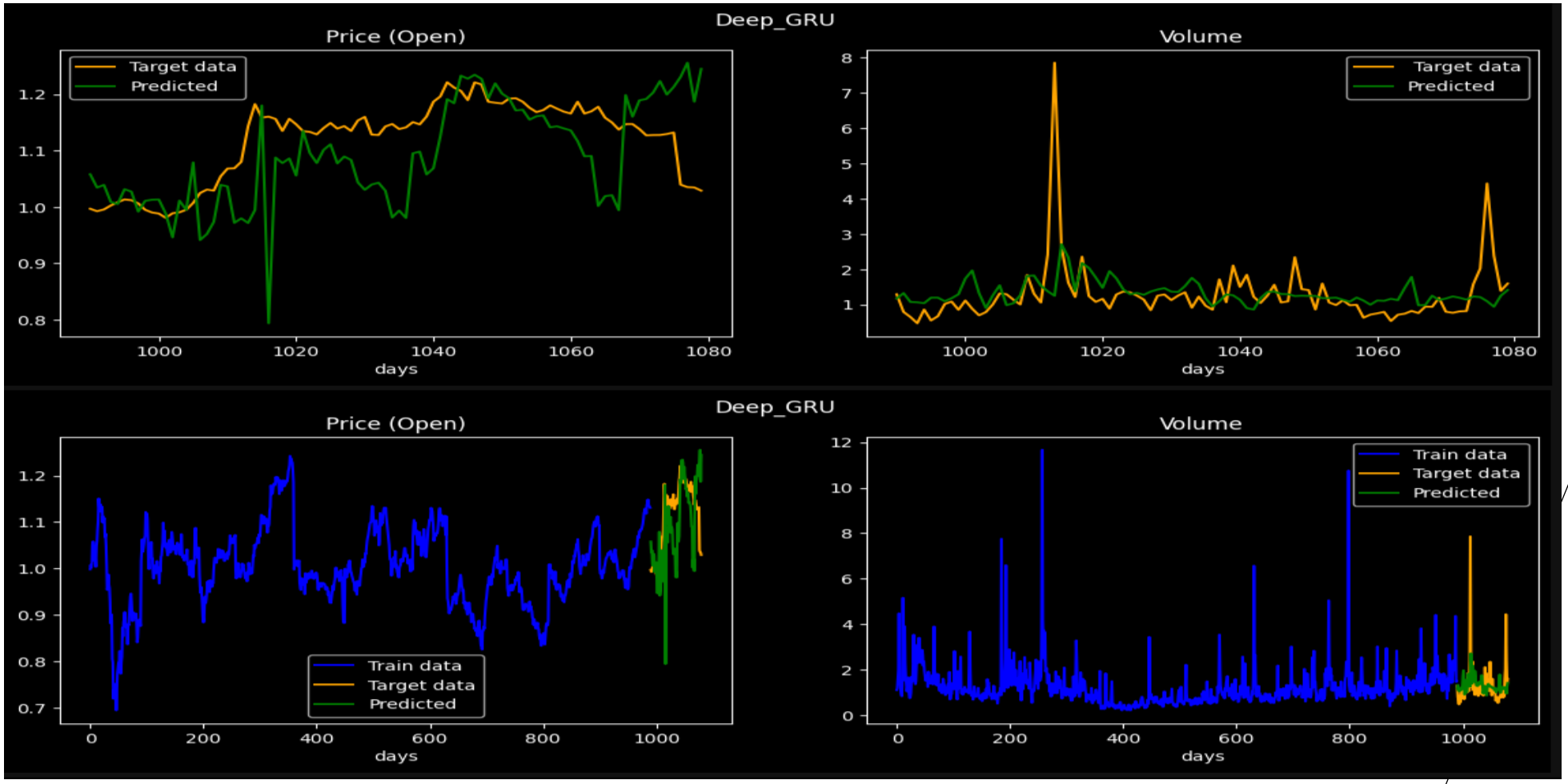


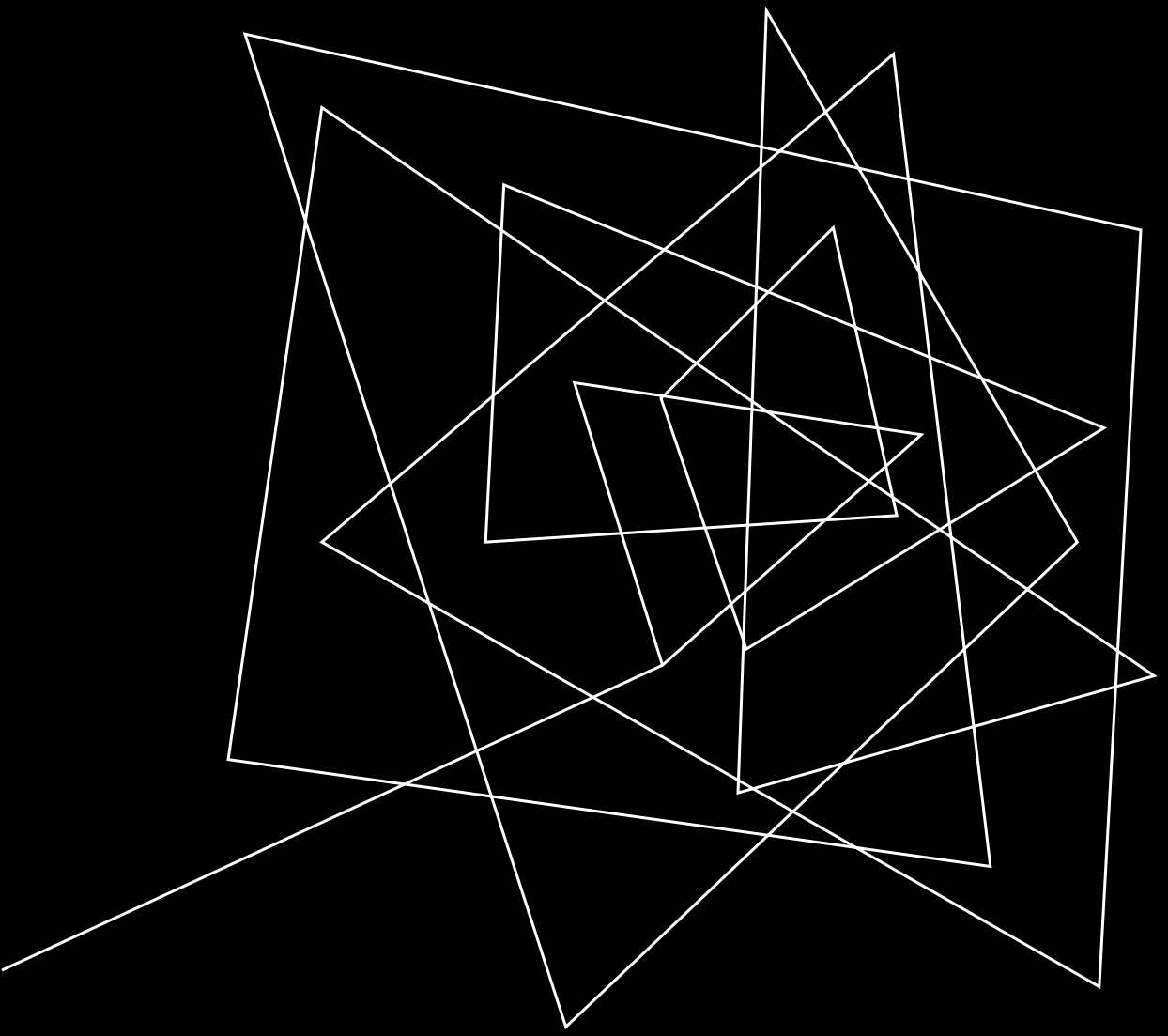
GRU

MSE for the last 90 days predicted: 0.41568419949564644
Best hyperparameters found for model GRU: {'units': 75, 'rate': 0.3,
'learning_rate': 0.001, 'optimizer': 'RMSprop'}

DEEP GRU

MSE for the last 90 days predicted: 0.42289310914661954
Best hyperparameters found for model Deep_GRU: {'units_1': 50, 'units_2': 150, 'rate': 0.1, 'learning_rate': 0.001, 'optimizer': 'Adam'}





CONCLUSION AND NEXT STEPS

CONCLUSION

We considered a dataset containing stock Prices and Volumes for about 1000 days.

We tuned hyperparameters (number of units, dropout rate, optimizer, learning_rate) and train some RNN models to predict the stock price/volume for the last 90 days of the dataset.

After RandomSearch hyperparameter tuning, and using MSE as error metric, we found the Deep_LSTM (with two hidden layers) and the GRU models as the ones with the lowest error metric.

NEXT STEPS

The models considered were very simple and there could be many ways in which they can be improved. For example,

-Using different scaling techniques

-Consider a custom metric for data series

-Consider data from other companies with the same characteristics

-Explore different model architectures

-Perform a thorough hyperparameter search.

In general, predicting stock prices is not an easy task.

THANK YOU

Adrian P. Bustamante, Ph.D.

adrianpebus@gmail.com