Time-Series Mastery

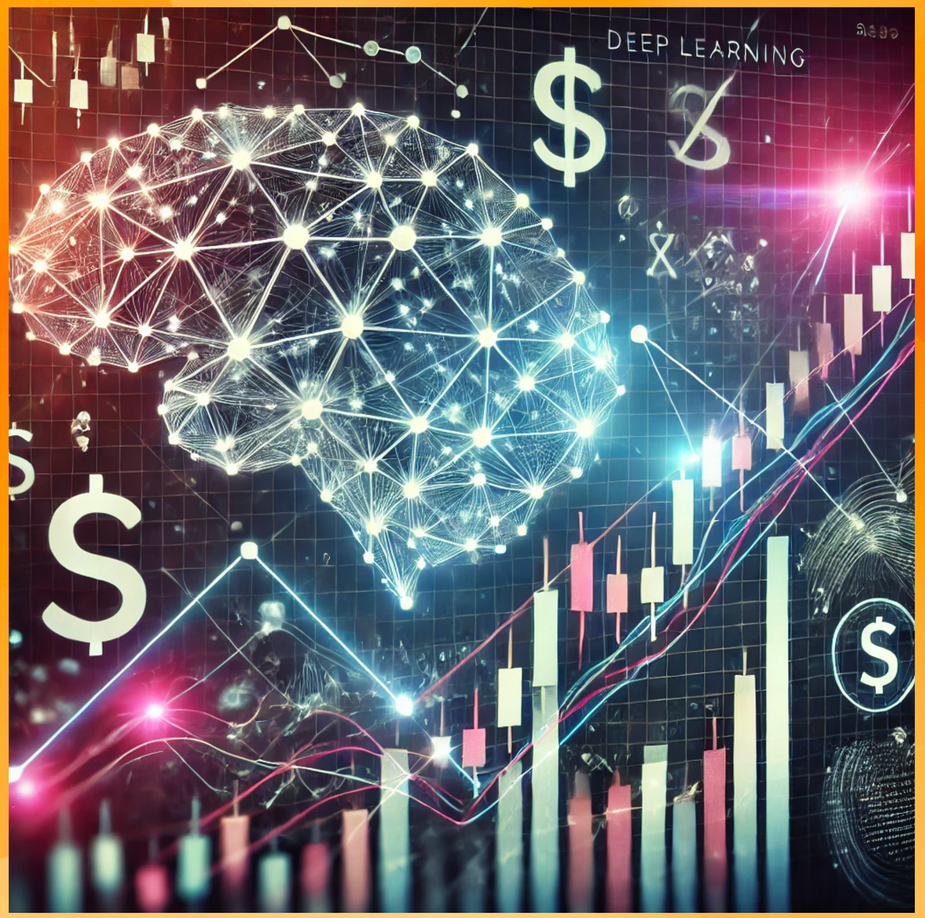
Deep Learning-Driven Stock Forecasting

Anay Abhijit Joshi

December 1, 2024

Cracking The Code!

Stock Prediction using Deep Learning….



Challenge: Navigating Market Volatility

Financial Markets are tempestuous, with wild price swings dictated by myriad forces. Therefore, high accuracy and high precision in forecasting can be the GOLDEN ticket to consistent profits.

Opportunity: Turing Data into Decisions

Harnessing predictive power transforms data into actionable insights, enabling traders to execute decisions with surgical precision for peak returns.

HFT Advantage: Next Move at the Speed of Light

TIME is the ultimate CURRENCY. Cutting-edge models empower traders to react faster than the blink of an eye, turning volatility into an opportunity.

Quant Finance Impact: Powering Portfolios and Redefining Strategies

Stock prediction is the cornerstone of pioneering financial strategies, intelligent risk management, and avant-garde portfolio optimization.

Real-World Edge: From Models to Money

This isn’t just theory - it’s a competitive arsenal. Robust forecasting translates directly into smarter trades, higher alpha, and minimized downside risks.

From Concepts to Code

Crafting Custom Solutions from the Ground Up….

Strategy

- **Sequential Acumen**

  Mastering Sequential Dependencies to Uncover Long-Term Trends.

- **Pattern Recognition**

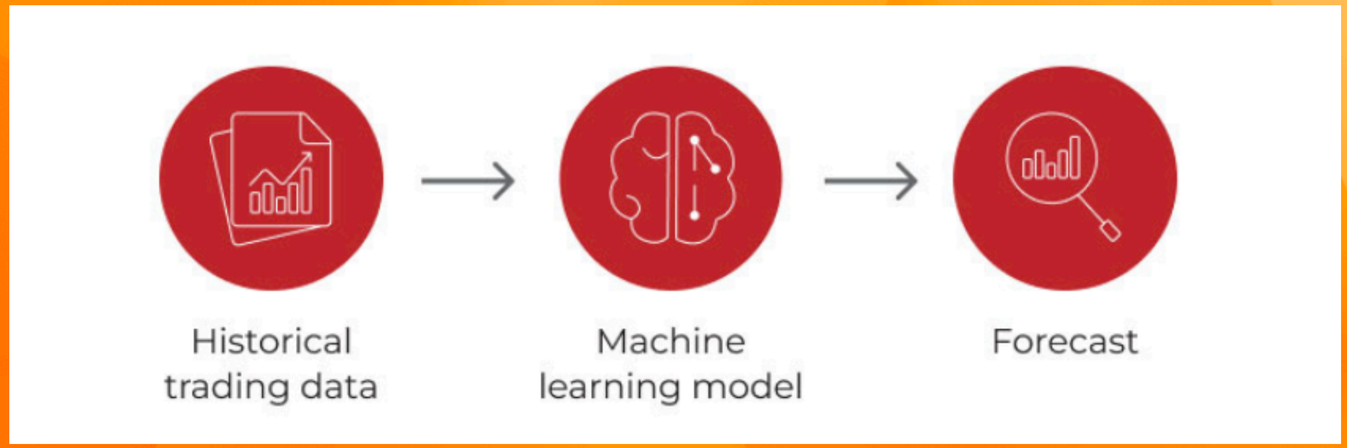
  Extracting Intricate Signals for Sharper Insights.

- **Time-Series Analysis**

  Leveraging Temporal Data to Refine Predictive Accuracy, and Other Metrics.

- **Dynamic Adaptability**

  Employing Architectures which Respond to Evolving Market Patterns.



Build from Scratch, Unconstrained from Research Papers

- **Unleashing True and Hands-On Innovation**

  By developing models independently, I cultivate genuine-original thinking, using Python libraries and modules to adapt seamlessly to volatile market data in real-time.

- **Independent Exploration**

  This approach nurtures a more profound, hands-on understanding, free from the constraints of the existing frameworks and/or methodologies, allowing for greater flexibility and creativity.

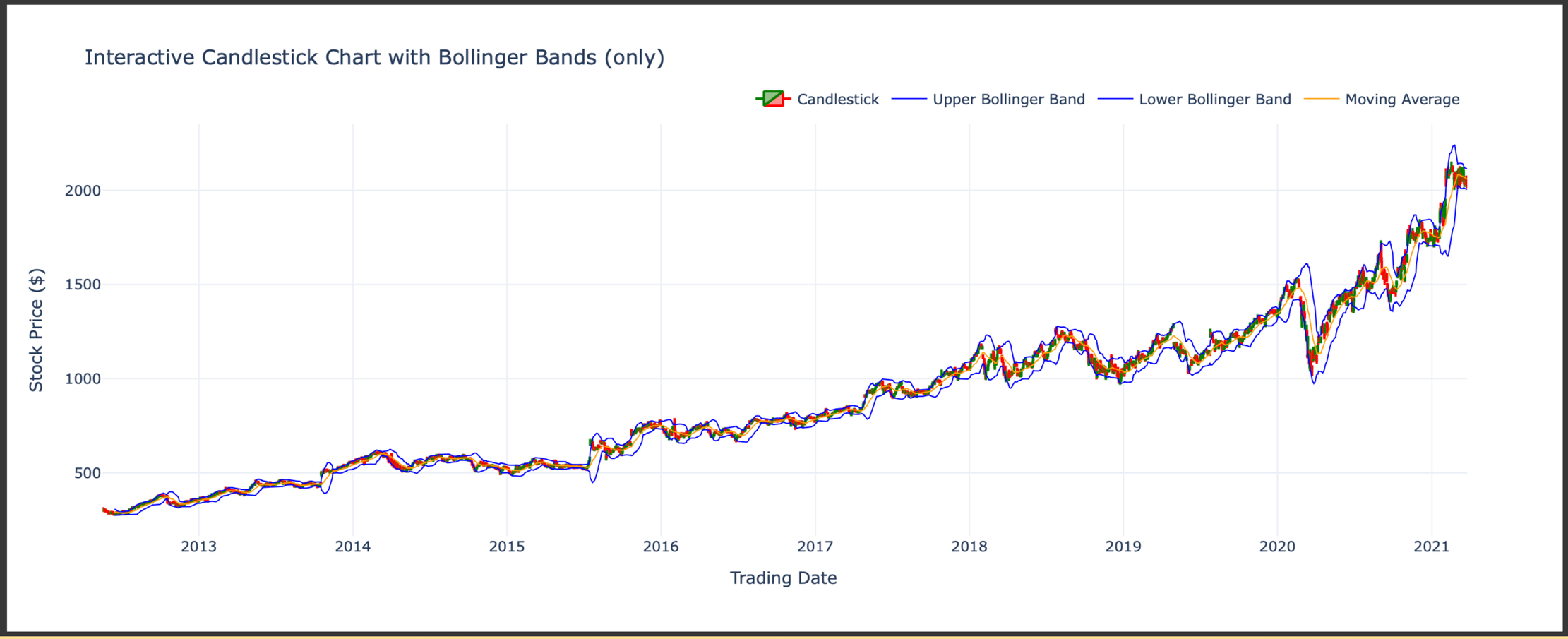
- **Tailored Solutions**

  Custom models are meticulously designed to align with the specific market dynamics, yielding results that generic, research-based models often fall short of - some of the top traits that are highly sought after by the leading quantitative finance firms.

Quant Alpha

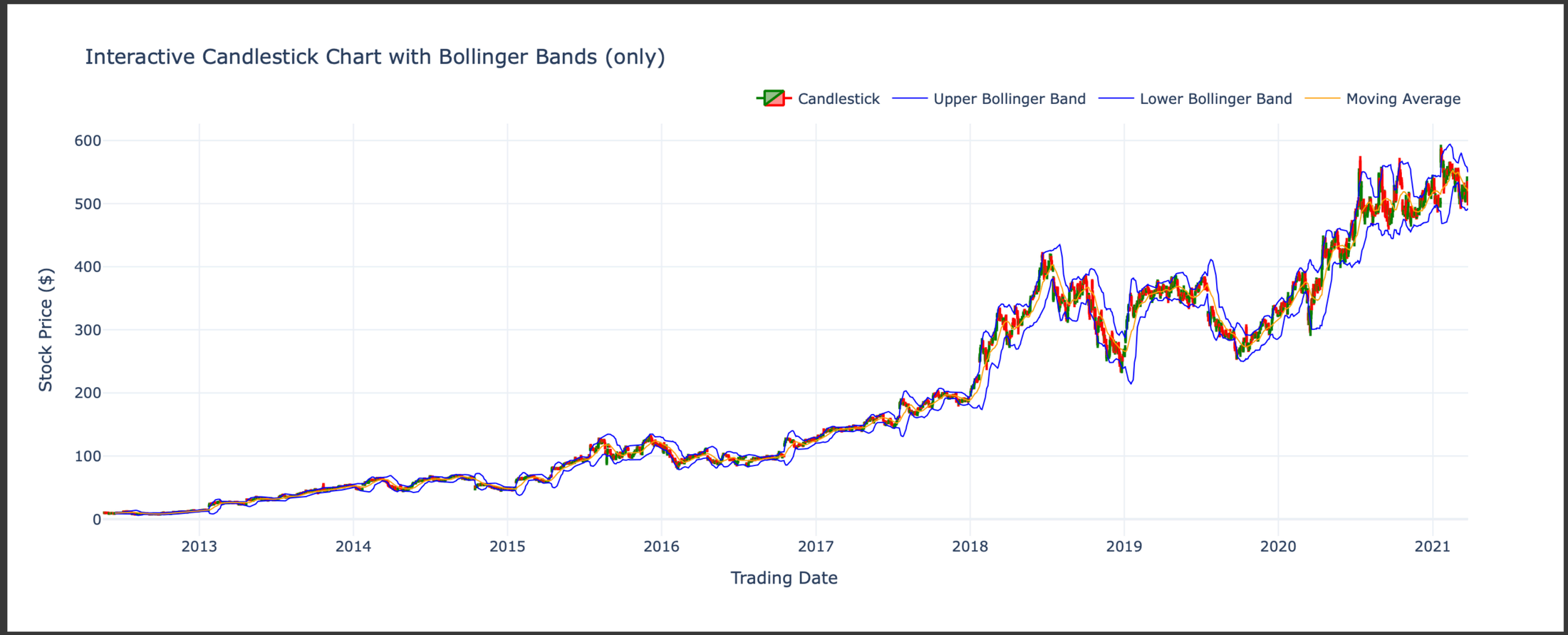
Google

```
[14] # Interactive Candlestick Chart with Bollinger Bands (only) - Google
     company_data = pd.read_csv("Google.csv")
     company_data['Date'] = pd.to_datetime(company_data['Date'])
     candlestick_with_bollinger(company_data)
```

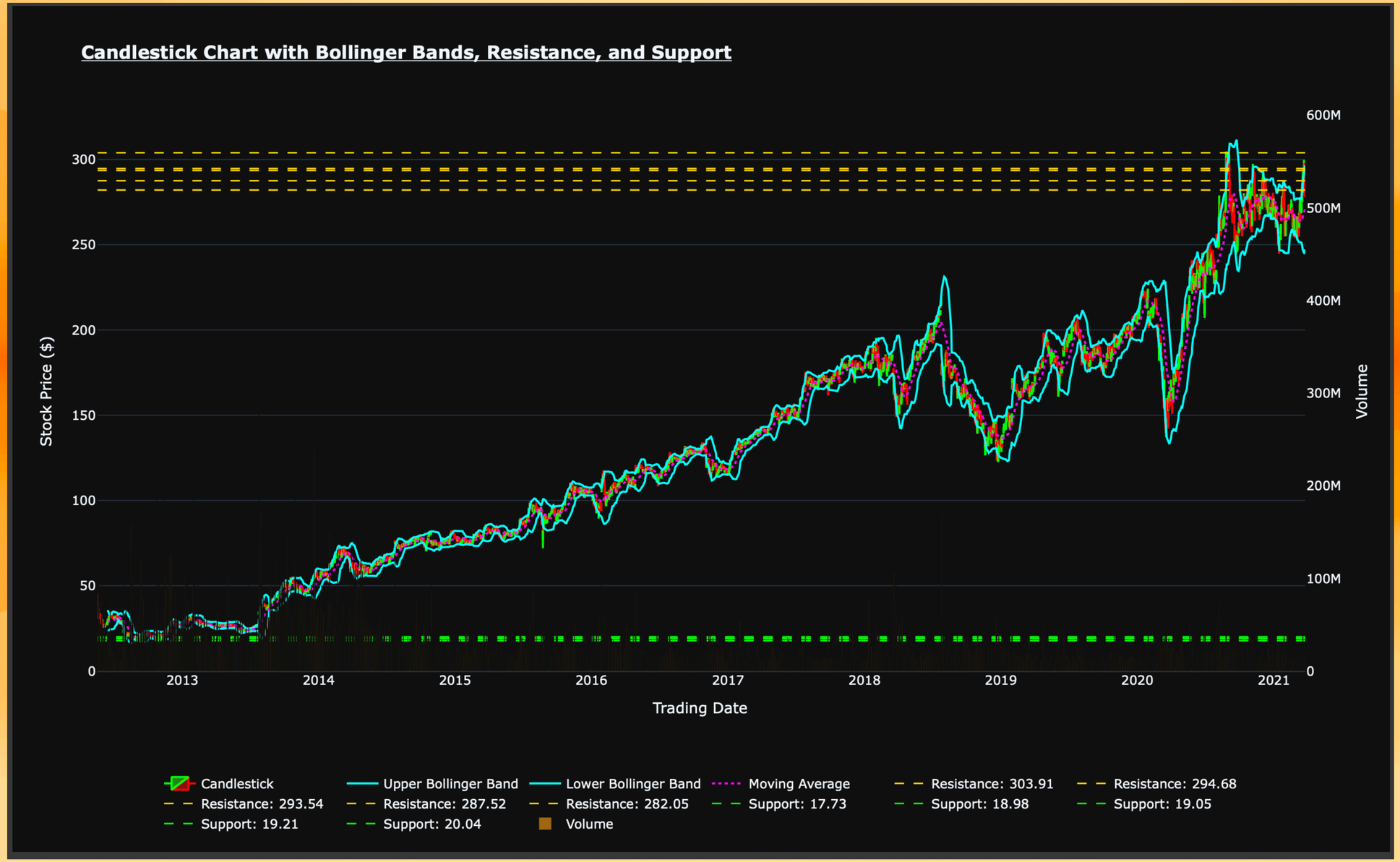


Netflix

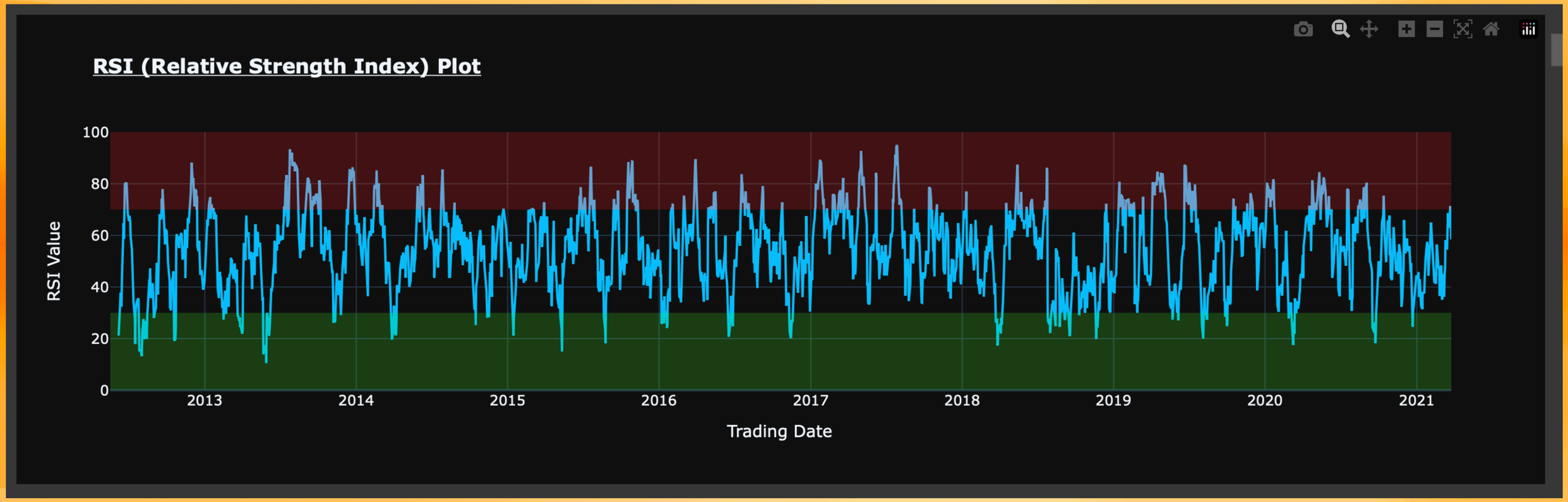
```
# Interactive Candlestick Chart with Bollinger Bands (only) - Netflix
company_data = pd.read_csv("Netflix.csv")
company_data['Date'] = pd.to_datetime(company_data['Date'])
candlestick_with_bollinger(company_data)
```



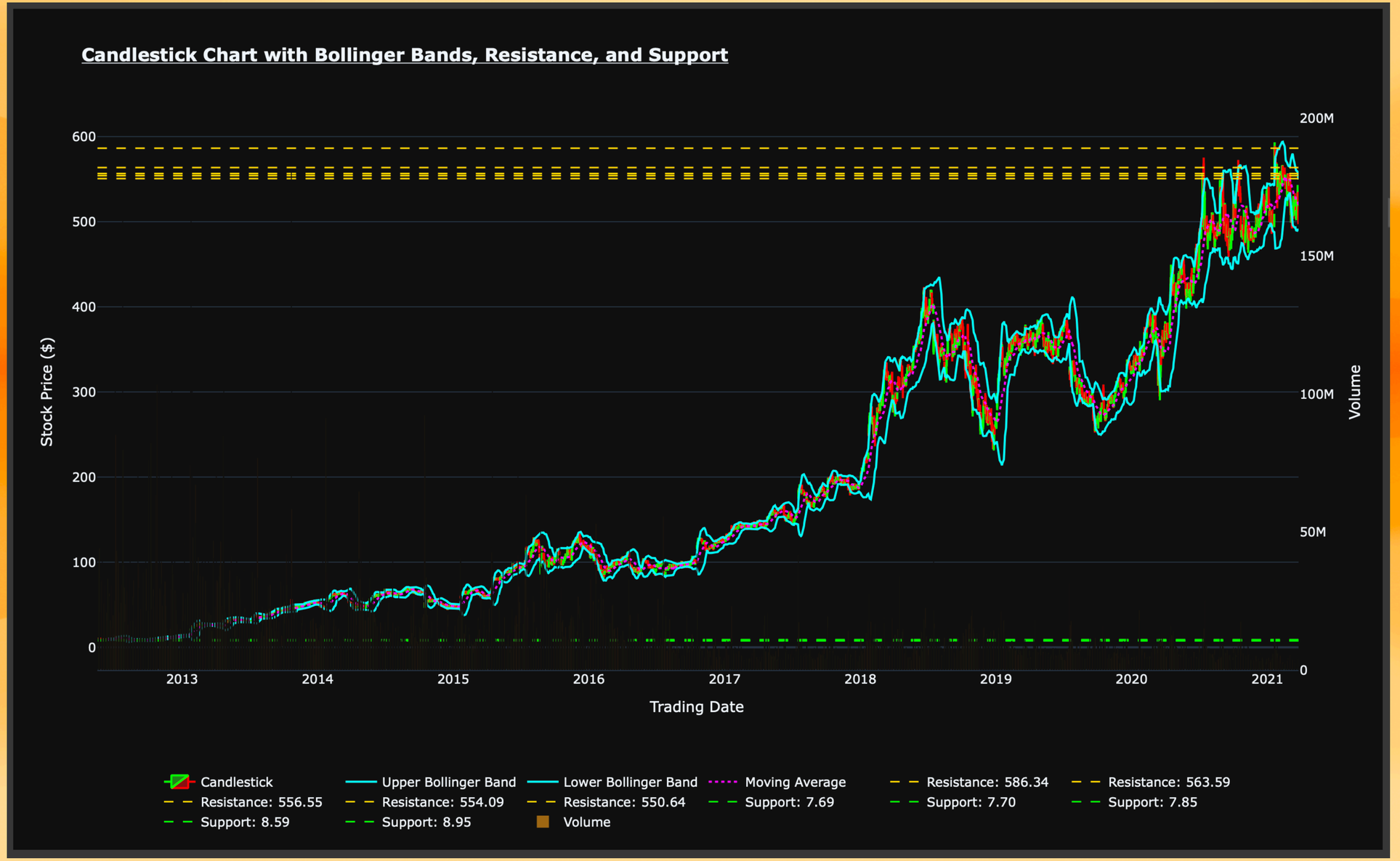
Facebook


Candlestick Chart with Bollinger Bands, Resistance, and Support
Stock Price ($)
Volume
Trading Date
0
50
100
150
200
250
300
0
100M
200M
300M
400M
500M
600M
2013
2014
2015
2016
2017
2018
2019
2020
2021
Candlestick
Upper Bollinger Band
Lower Bollinger Band
Moving Average
Resistance: 303.91
Resistance: 294.68
Resistance: 293.54
Resistance: 287.52
Resistance: 282.05
Support: 17.73
Support: 18.98
Support: 19.05
Support: 19.21
Support: 20.04
Volume

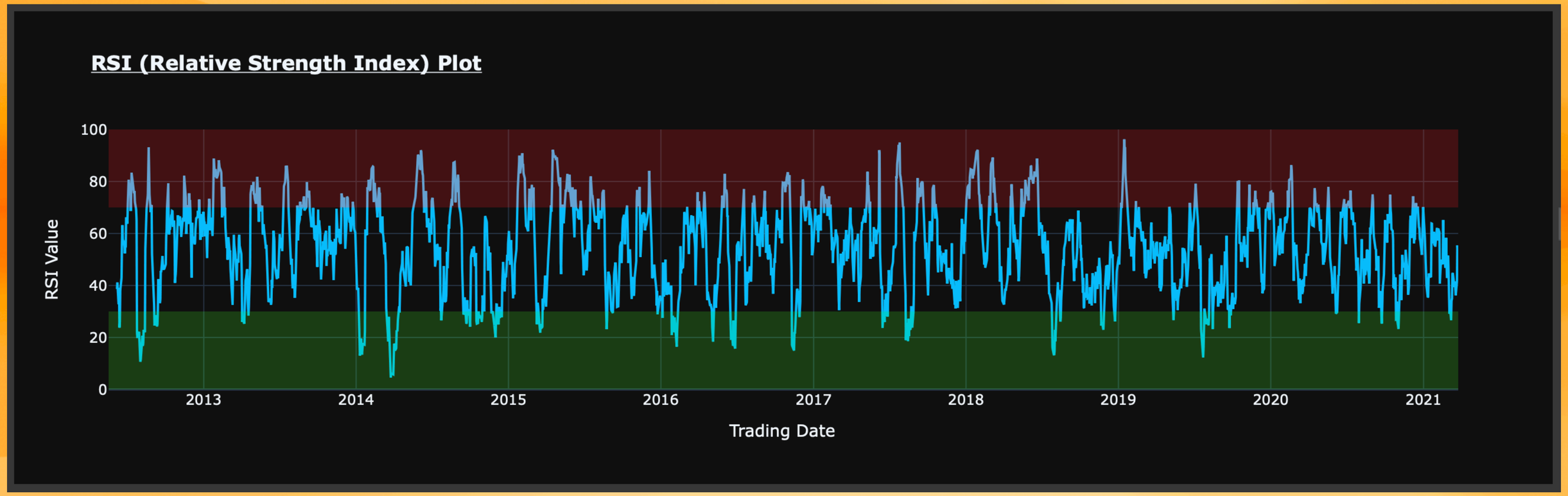
Facebook RSI (Relative Strength Index)



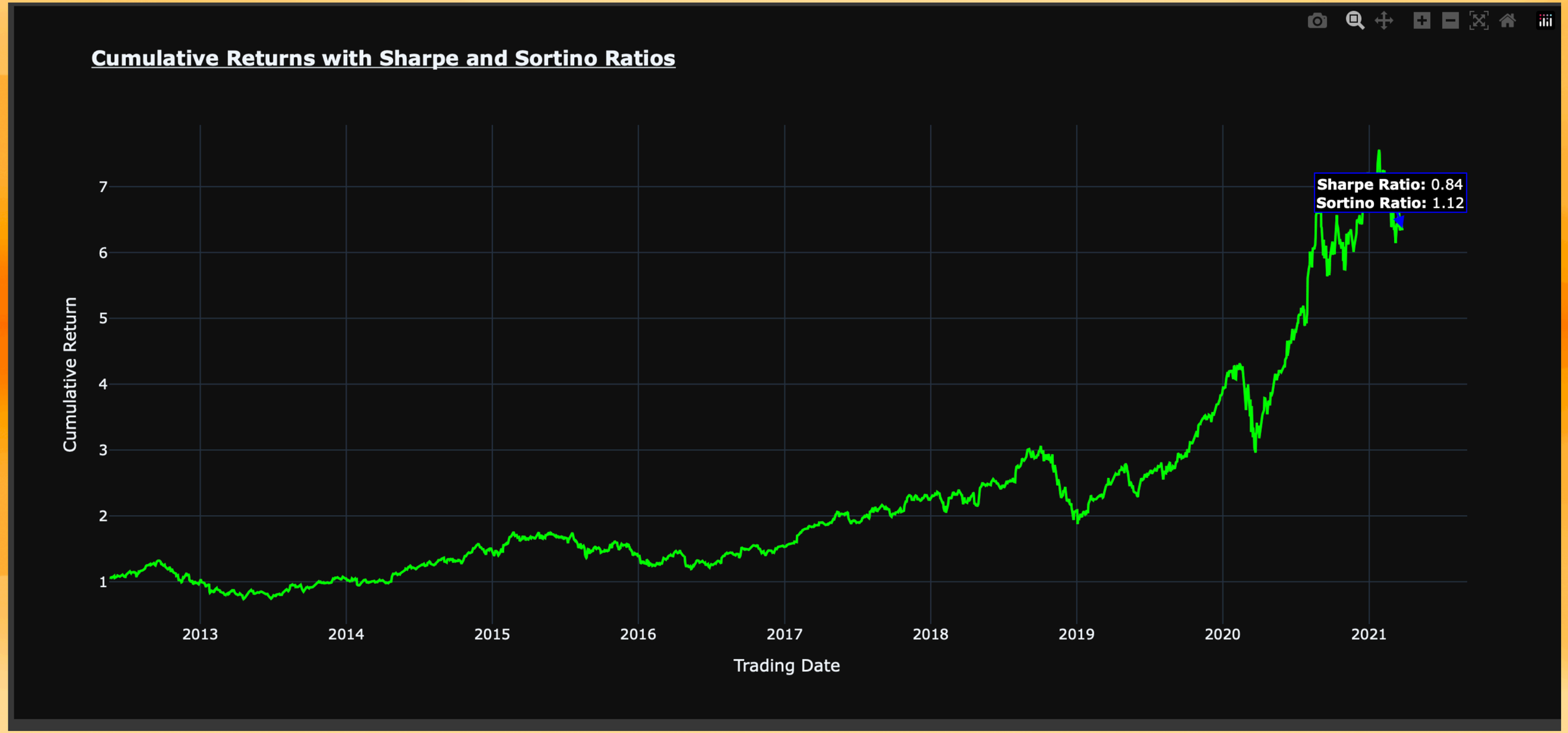
Netflix


Candlestick Chart with Bollinger Bands, Resistance, and Support
Stock Price ($)
600
500
400
300
200
100
0
200M
150M
100M
50M
0
Volume
2013
2014
2015
2016
2017
2018
2019
2020
2021
Trading Date
Candlestick
Upper Bollinger Band
Lower Bollinger Band
Moving Average
Resistance: 586.34
Resistance: 563.59
Resistance: 556.55
Resistance: 554.09
Resistance: 550.64
Support: 7.69
Support: 7.70
Support: 7.85
Support: 8.59
Support: 8.95
Volume

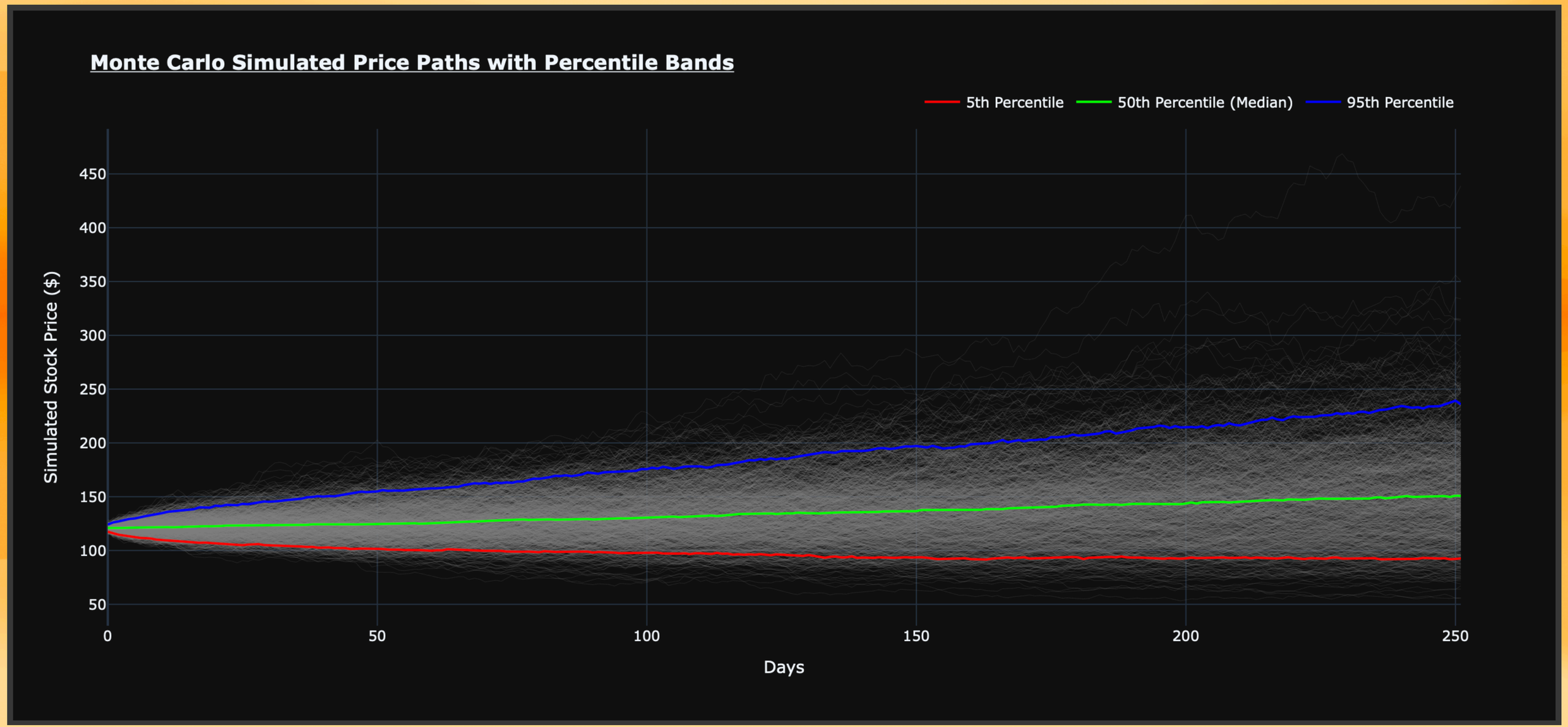
Netflix RSI (Relative Strength Index)



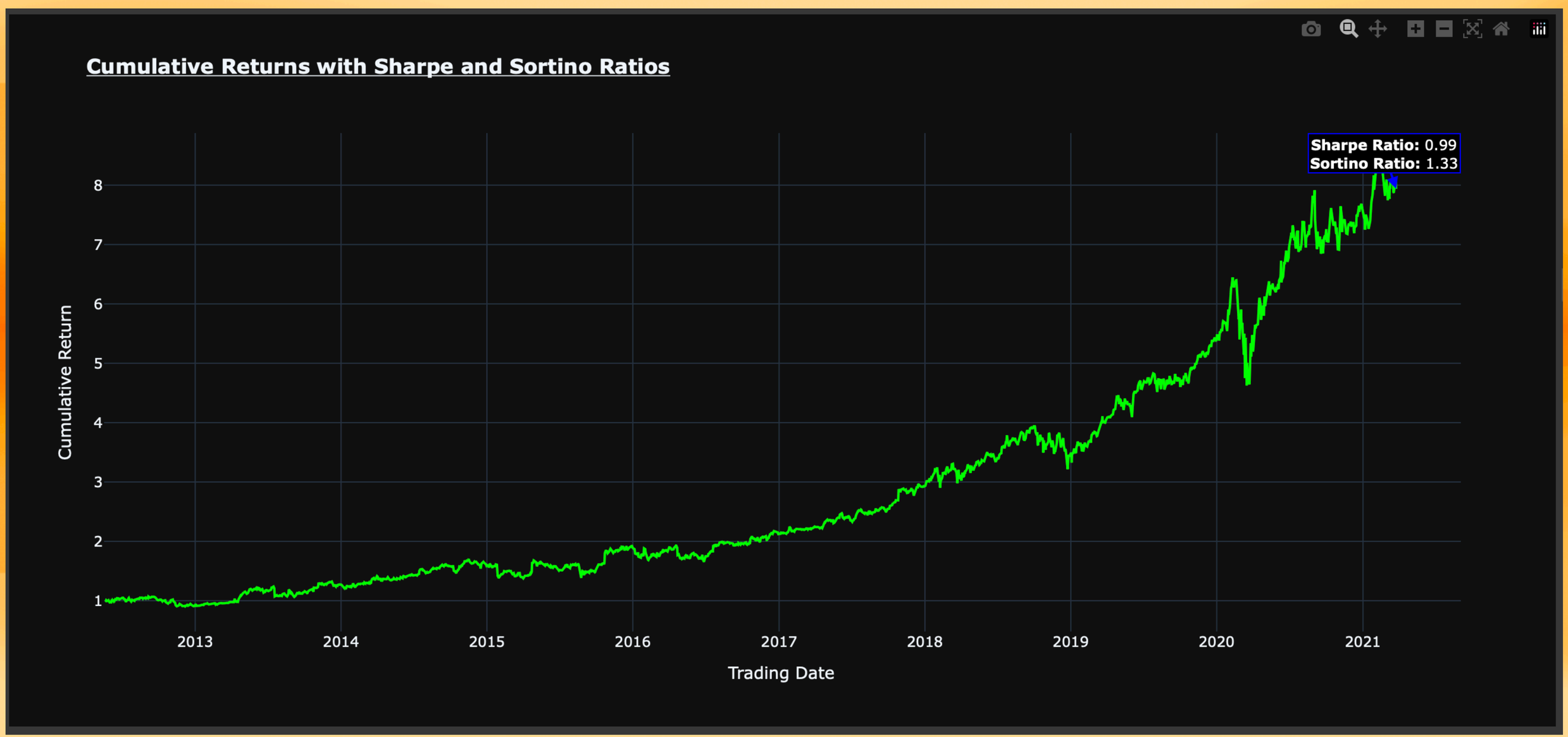
Apple Cumulative Returns



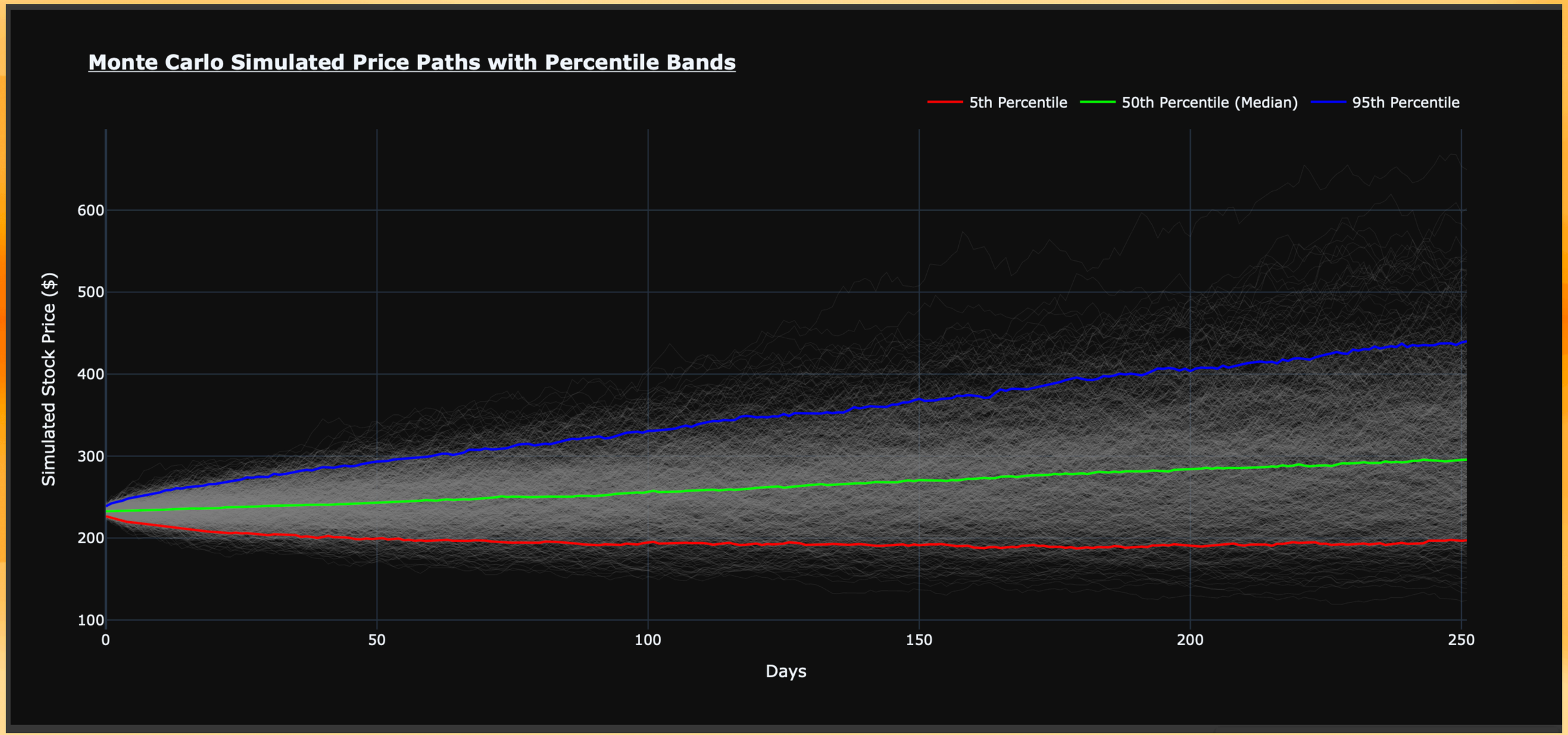
Apple Monte Carlo Simulated Price Path(s)



Microsoft Cumulative Returns



Microsoft Monte Carlo Simulated Price Path(s)



Closing Prices: “FAANG + M”

Overview

Data Split Ratio

1. Training - 80 %
2. Validation - 10 %
3. Testing - 10 %

Normalization

MinMaxScaler

Quant Alpha's
Deep Neural Architectures

1. ANN (Artificial Neural Network)
2. LSTM (Long Short-Term Memory)
3. Autoencoder
4. RNN (Recurrent Neural Network)
5. MLP (Multi-Layer Perceptron)

Implementation and Results

ANN *(Artificial Neural Network)*

Architecture

- Input Layer : 128 Neurons
- Activation Function : ReLU *(Rectified Linear Unit)*
- Hidden Layer : 64 Neurons
- Dropout : 20% *(to prevent Overfitting)*
- Output Layer : 1 Neuron for Regression

Optimizer

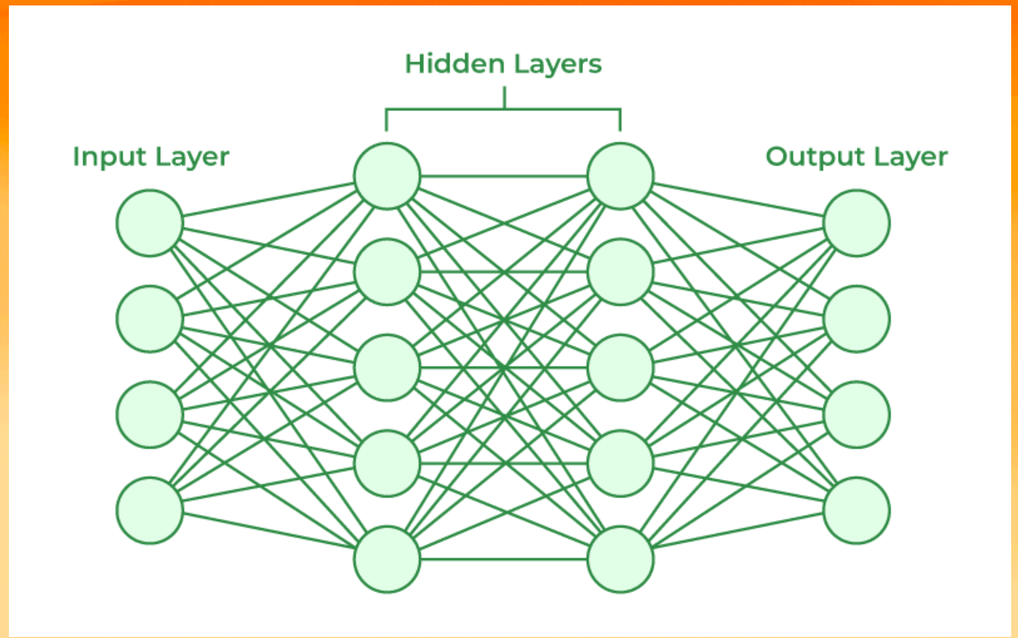
Adam

Loss Function

Mean Squared Error (MSE)

Training

Batch Size : 32
Number of Epochs : 10



ANN ➡ Apple

```
[38] test_model(file_name="Apple.csv", model_name="ANN")
```

```
Deep Learning Model:     ANN (Artificial Neural Network)
Epoch 1/10
61/61 [==============================] - 1s 2ms/step - loss: 0.0055
Epoch 2/10
61/61 [==============================] - 0s 2ms/step - loss: 0.0013
Epoch 3/10
61/61 [==============================] - 0s 2ms/step - loss: 0.0011
Epoch 4/10
61/61 [==============================] - 0s 2ms/step - loss: 7.9440e-04
Epoch 5/10
61/61 [==============================] - 0s 2ms/step - loss: 7.2865e-04
Epoch 6/10
61/61 [==============================] - 0s 2ms/step - loss: 7.4494e-04
Epoch 7/10
61/61 [==============================] - 0s 2ms/step - loss: 7.3680e-04
Epoch 8/10
61/61 [==============================] - 0s 2ms/step - loss: 8.1169e-04
Epoch 9/10
61/61 [==============================] - 0s 2ms/step - loss: 6.7256e-04
Epoch 10/10
61/61 [==============================] - 0s 2ms/step - loss: 6.7970e-04
7/7 [==============================] - 0s 1ms/step

                                                R² Score = 0.9295907492695888
                          Mean Squared Logarithmic Error (MSLE) = 0.0016238158325390716
```

ANN ➡ Apple *(Predictions vs Observed Values)*

LSTM *(Long-Short Term Memory)*

Architecture

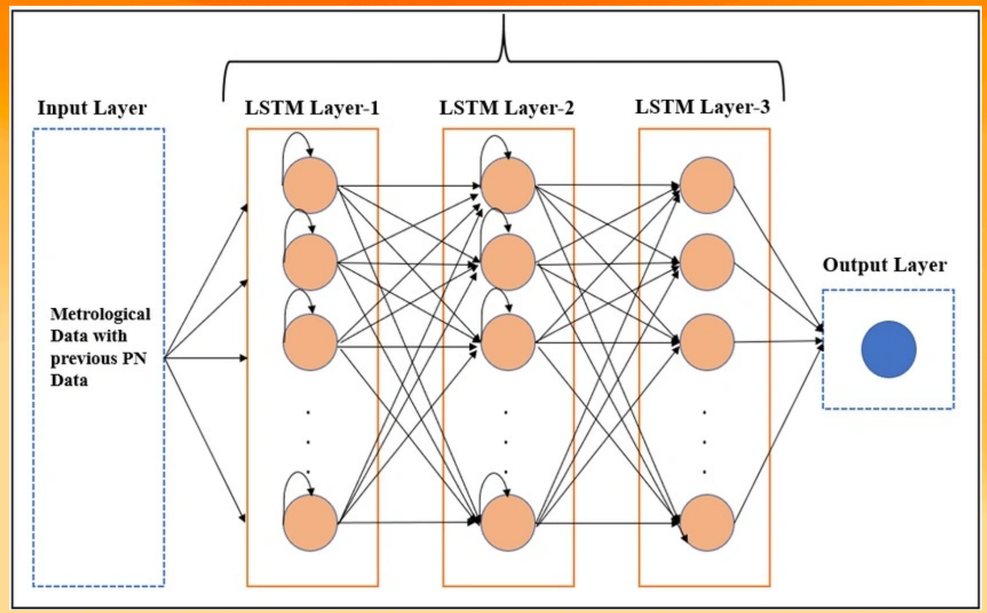
- LSTM Layer 1 : 128 Neurons
- LSTM Layer 2 : 50 Neurons
- Dense Layer : 32 Neurons
- Dropout : 20% *(to prevent Overfitting)*
- Output Layer : 1 Neuron for Regression

Optimizer

Adam

Loss Function

Mean Squared Error (MSE)



LSTM ➡ Google *(Predictions vs Observed Values)*

```
test_model(file_name="Google.csv", model_name="LSTM")
```

```
Deep Learning Model:     LSTM (Long Short-Term Memory)
1945/1945 [==============================] - 53s 26ms/step - loss: 0.0030
7/7 [==============================] - 1s 25ms/step

                                                    R² Score = 0.9393513217434637
                                Mean Squared Logarithmic Error (MSLE) = 0.0010283612592525348
```


```
Timestamp: 2024-12-03 03:18:19.675664
```

Autoencoder

Architecture

- Encoder
  - Dense Layers: 32 & 16 Neurons
- Decoder
  - Dense Layers: 16 & 1 Neuron(s)
- L1 Regularization

Optimizer

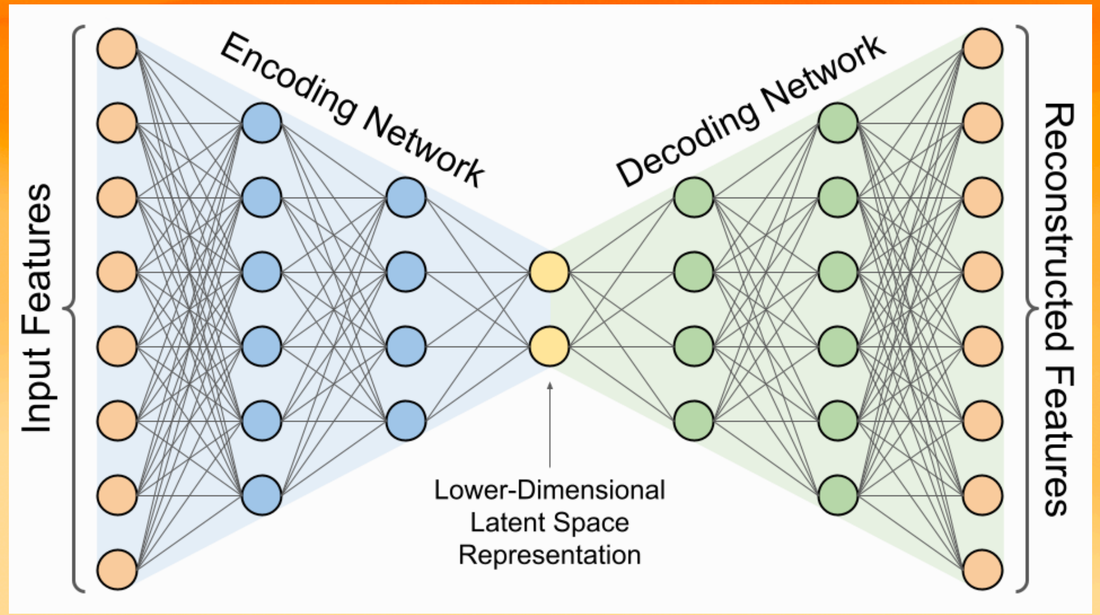
Adam

Loss Function

Mean Squared Error (MSE)

Training

Batch Size : 32
Number of Epochs : 10



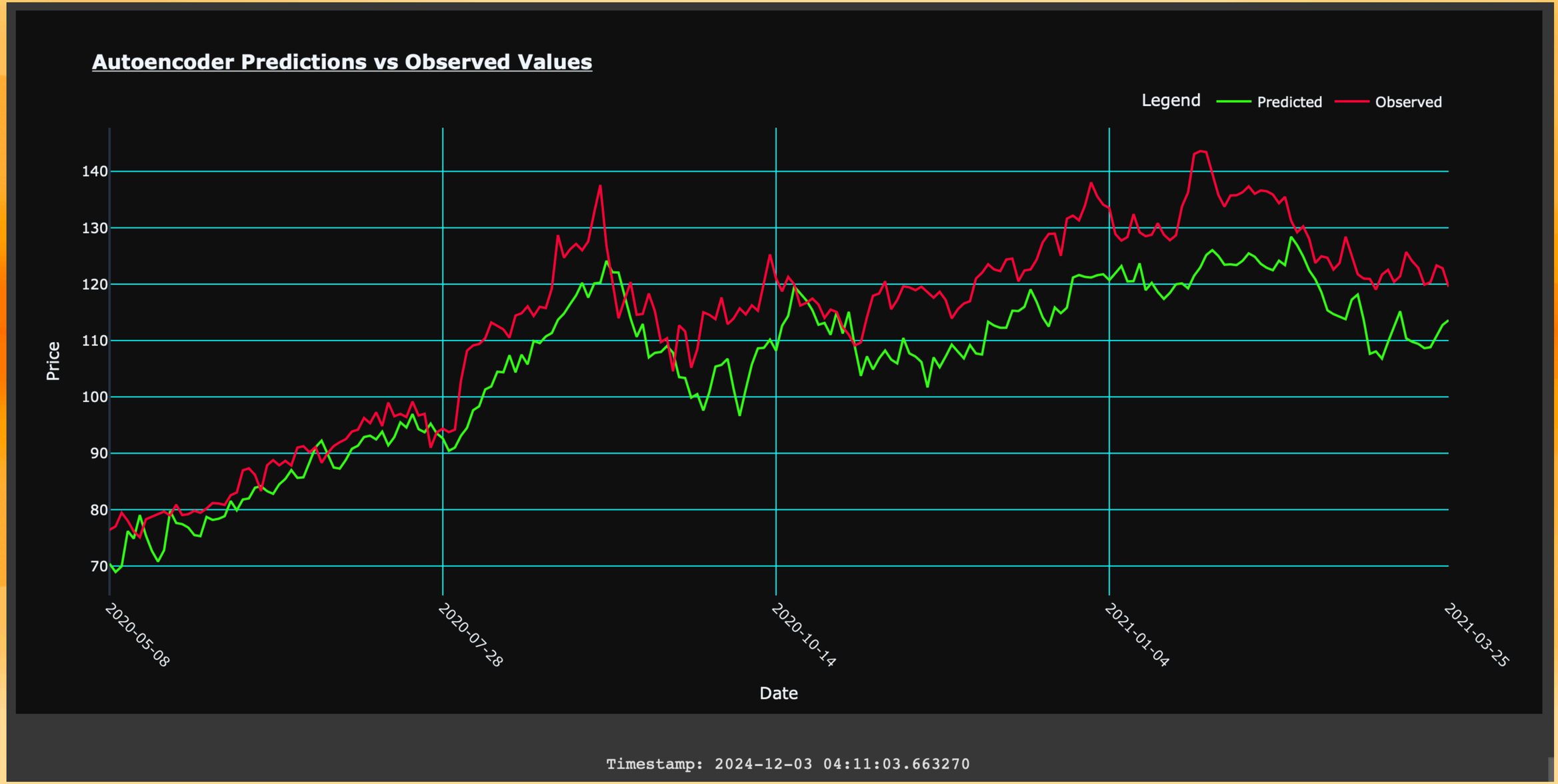
Autoencoder ➡ Apple

```
test_model(file_name="Apple.csv", model_name="Autoencoder")
```

```
Deep Learning Model:     Autoencoder
Epoch 1/10
61/61 [==============================] - 1s 2ms/step - loss: 0.0260
Epoch 2/10
61/61 [==============================] - 0s 2ms/step - loss: 3.4927e-04
Epoch 3/10
61/61 [==============================] - 0s 2ms/step - loss: 1.8713e-04
Epoch 4/10
61/61 [==============================] - 0s 2ms/step - loss: 1.4644e-04
Epoch 5/10
61/61 [==============================] - 0s 2ms/step - loss: 1.6925e-04
Epoch 6/10
61/61 [==============================] - 0s 2ms/step - loss: 1.2428e-04
Epoch 7/10
61/61 [==============================] - 0s 2ms/step - loss: 1.2699e-04
Epoch 8/10
61/61 [==============================] - 0s 2ms/step - loss: 1.2053e-04
Epoch 9/10
61/61 [==============================] - 0s 2ms/step - loss: 1.1428e-04
Epoch 10/10
61/61 [==============================] - 0s 2ms/step - loss: 1.0424e-04
7/7 [==============================] - 0s 2ms/step

                                                R² Score = 0.7215058479435453
                              Mean Squared Logarithmic Error (MSLE) = 0.006261768906263677
```

Autoencoder ➡ Apple *(Predictions vs Observed Values)*



RNN *(Recurrent Neural Network)*

Architecture

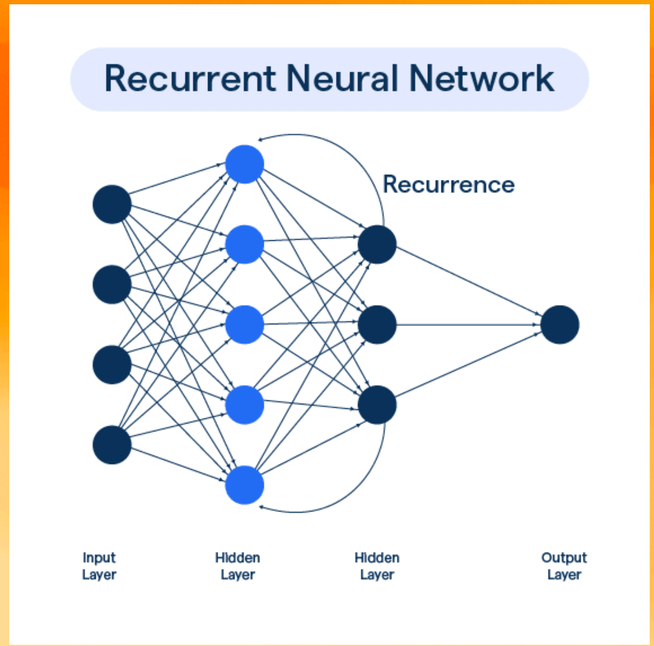
- RNN Layer : 4 Neurons
- Dense Output Layer: 1 Neuron

Optimizer

Adam

Loss Function

Mean Squared Error (MSE)



RNN ➡ Google

```
test_model(file_name="Google.csv", model_name="RNN")
```

```
Deep Learning Model:     RNN (Recurrent Neural Network)
Epoch 1/10
1945/1945 [==============================] - 10s 5ms/step - loss: 2.9084e-04
Epoch 2/10
1945/1945 [==============================] - 9s 5ms/step - loss: 1.2320e-04
Epoch 3/10
1945/1945 [==============================] - 9s 5ms/step - loss: 1.1009e-04
Epoch 4/10
1945/1945 [==============================] - 9s 5ms/step - loss: 1.0045e-04
Epoch 5/10
1945/1945 [==============================] - 9s 5ms/step - loss: 9.5497e-05
Epoch 6/10
1945/1945 [==============================] - 9s 4ms/step - loss: 8.8141e-05
Epoch 7/10
1945/1945 [==============================] - 9s 5ms/step - loss: 8.7044e-05
Epoch 8/10
1945/1945 [==============================] - 9s 5ms/step - loss: 8.8595e-05
Epoch 9/10
1945/1945 [==============================] - 9s 5ms/step - loss: 8.6456e-05
Epoch 10/10
1945/1945 [==============================] - 9s 5ms/step - loss: 8.6732e-05
7/7 [==============================] - 0s 4ms/step

                                                R² Score = 0.9030484500218665
                          Mean Squared Logarithmic Error (MSLE) = 0.001520258280332099
```

RNN ➡ Google *(Predictions vs Observed Values)*

Timestamp: 2024-12-03 03:39:58.382727

MLP *(Multi-Layer Perceptron)*

Architecture

- **Hidden Layer :** **100 Neurons**
- **Activation Function :** **ReLU** ***(Rectified Linear Unit)***

Optimizer

LBFGS (Limited-memory Broyden-Fletcher-Goldfarb-Shanno)
- A Quasi-Newton method for faster convergence

Regularization

Alpha parameter **set to** 0.01 **to prevent overfitting.**

MLP ➡ Google *(Predictions vs Observed Values)*

```
75] test_model(file_name="Google.csv", model_name="MLP")

Deep Learning Model:     MLP (Multi-Layer Perceptron)

                                        R² Score = 0.959946719750091
                Mean Squared Logarithmic Error (MSLE) = 0.0006909881585625005
```


```
Timestamp: 2024-12-03 04:14:21.704472
```

Thank You !!!