FOREIGN TRADE UNIVERSITY – HCM CAMPUS

FINAL ASSIGNMENT

COURSE: **AI IN THE ERA OF DIGITAL TRANSFORMATION**

Date: 02/01/2024 – Class Code: ML306 – Course Code: **AIDE300**

No.	Full Name	ID	Peer Evaluation (0%-100%)
1	Trần Lưu Thanh Thảo	2011156096	100%
2	Trần Anh Thư	2011156099	100%
3	Dương Ngọc Minh Thư	2011156102	100%

Grade (in numbers)	**Grade (in words)**
Examiner 1 **Dr. Lê Trung Thành**	**Examiner 2** **Dr. Nguyễn Thị Hoàng Anh**

TABLE OF CONTENTS

1. INTRODUCTION

This artificial intelligence (AI) project implemented in the Jupyter Notebook is designed to perform stock price prediction, specifically targeting the SSI Securities Corporation (SSI). The primary aim of this endeavor is to harness the power of cutting-edge technologies and advanced predictive models to forecast SSI's stock prices accurately. Predicting stock prices within the financial industry, particularly for companies directly involved in stock trading and investment services, holds immense significance due to the sector's inherent volatility and interdependence. Forecasting stock prices of financial firms offers critical insights into market trends, regulatory changes, and economic shifts that can profoundly impact not just the company itself but the broader financial ecosystem. Accurate predictions empower investors, analysts, and financial institutions to anticipate market movements, optimize investment strategies, and manage risks effectively. Additionally, as the financial industry is highly sensitive to global economic fluctuations and policy changes, predicting stock prices within this sector becomes a vital tool for stakeholders seeking to navigate the intricate landscape of investments, regulatory compliance, and the evolving nature of financial markets.

This project embarks on a comprehensive journey that encompasses data exploratory analysis, leveraging traditional econometric methods like the AutoRegressive Integrated Moving Average (ARIMA) model, and harnessing the prowess of state-of-the-art deep learning models including Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU), and Transformer architectures. By amalgamating these diverse methodologies, the project strives not only to forecast SSI's stock prices but also to provide a comparative analysis of the efficacy and performance of these varied models.

The overarching goal is to unlock a deeper understanding of the intricate patterns and trends within the stock market, offering valuable insights into the potential trajectories of SSI's stock prices. Through meticulous data exploration, rigorous econometric analysis, and the deployment of sophisticated deep learning architectures, this project endeavors to push the boundaries of predictive accuracy, thereby empowering stakeholders with actionable and reliable predictions in the volatile landscape of stock trading.

2. RESULTS

2.1. Descriptive Statistics

2.1.1. Data description

The historical price of SSI stock is derived from the Vietnam Stock market through 'vnstock' package in Python. After the data-cleaning process, the dataset now contains 2944 rows and 6 columns about the 'time', 'open', 'high', 'low', 'close', and 'volume' of the SSI stock price from 20 March 2012 to 28 December 2023.

	time	open	high	low	close	volume
0	2012-03-20	5170	5290	5110	5290	2018790
1	2012-03-21	5350	5560	5320	5380	2955280
2	2012-03-22	5270	5460	5240	5270	2388550
3	2012-03-23	5270	5500	5270	5460	2609850
4	2012-03-26	5500	5500	5380	5380	2511360
...	...	...	...	...	...	...
2939	2023-12-22	32350	32950	32200	32350	17457600
2940	2023-12-25	32350	32750	32100	32549	13953300
2941	2023-12-26	32600	32900	32500	32750	14005700
2942	2023-12-27	32950	33150	32800	32800	16101500
2943	2023-12-28	32850	33150	32650	33000	10895900

2944 rows × 6 columns

Then we created two new columns including 'daily_lag' – one day lag shifted of close price, and 'daily_return' – the percentage change between the lagged value and the current day's closing price.

	time	open	high	low	close	volume	daily_lag	daily_return
0	2012-03-20	5170	5290	5110	5290	2018790	NaN	NaN
1	2012-03-21	5350	5560	5320	5380	2955280	5290.0	-0.016729
2	2012-03-22	5270	5460	5240	5270	2388550	5380.0	0.020873
3	2012-03-23	5270	5500	5270	5460	2609850	5270.0	-0.034799
4	2012-03-26	5500	5500	5380	5380	2511360	5460.0	0.014870

The data description and rounding its values up to two decimal places as follow:

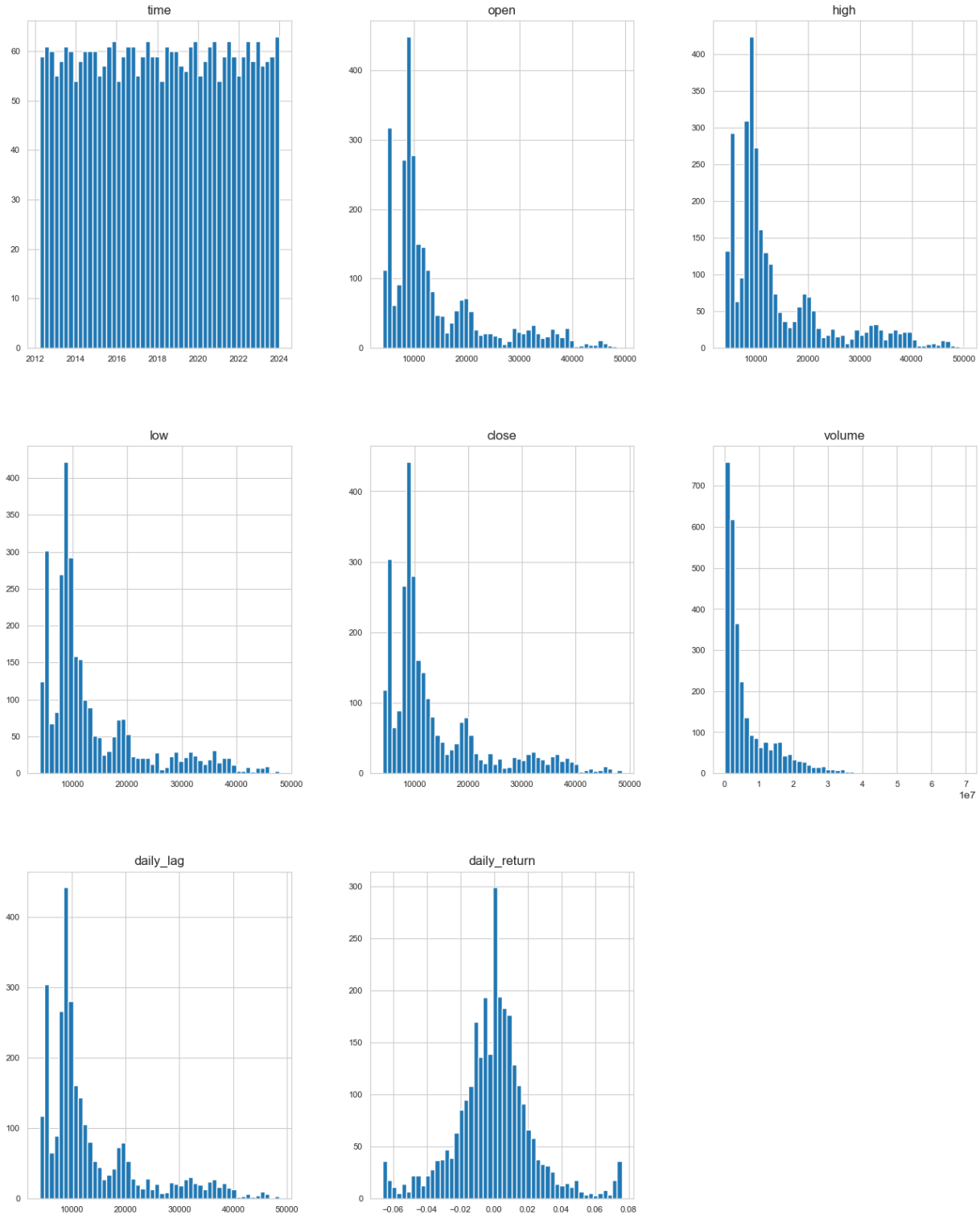
	time	open	high	low	close	volume	daily_lag	daily_return
count	2944	2944.00	2944.00	2944.00	2944.00	2944.00	2943.00	2943.00
mean	2018-02-11 04:50:03.260869632	13661.85	13880.10	13447.16	13652.51	6520083.20	13645.94	-0.00
min	2012-03-20 00:00:00	3930.00	4000.00	3930.00	3950.00	137750.00	3950.00	-0.07
25%	2015-03-08 06:00:00	8184.25	8290.00	8050.00	8189.00	1493690.00	8189.00	-0.01
50%	2018-02-08 12:00:00	9910.00	10010.00	9780.00	9890.00	3219395.00	9890.00	0.00
75%	2021-01-19 06:00:00	17197.50	17510.00	16842.50	17190.00	8636900.00	17160.00	0.01
max	2023-12-28 00:00:00	49380.00	50160.00	47810.00	48590.00	69559104.00	48590.00	0.08
std	NaN	9324.83	9505.34	9141.07	9313.78	7761242.43	9308.53	0.02

The summary statistics (min, max, percentile, standard deviation) provide a glimpse into the distribution and variability of the numerical columns. Notably, the mean of 'daily_return' is close to zero (-0.00), suggesting an almost neutral average daily return. The standard deviation of 'daily_return' indicates moderate volatility in the daily returns, considering its magnitude (0.02). The range between the minimum and maximum values of 'daily_return' (-0.07 to 0.08) illustrates the variability of daily returns over the period covered by the dataset.

2.1.2. Distribution

For more visualization, there is a histogram grid for each numerical column in the dataset. Looking at the attributes related to price (open, high, low, close) and volume, it shows that their distribution is right skewed. The frequency of prices around VND10,000 is higher than other prices. Prices at higher thresholds such as above VND35,000 have low frequency

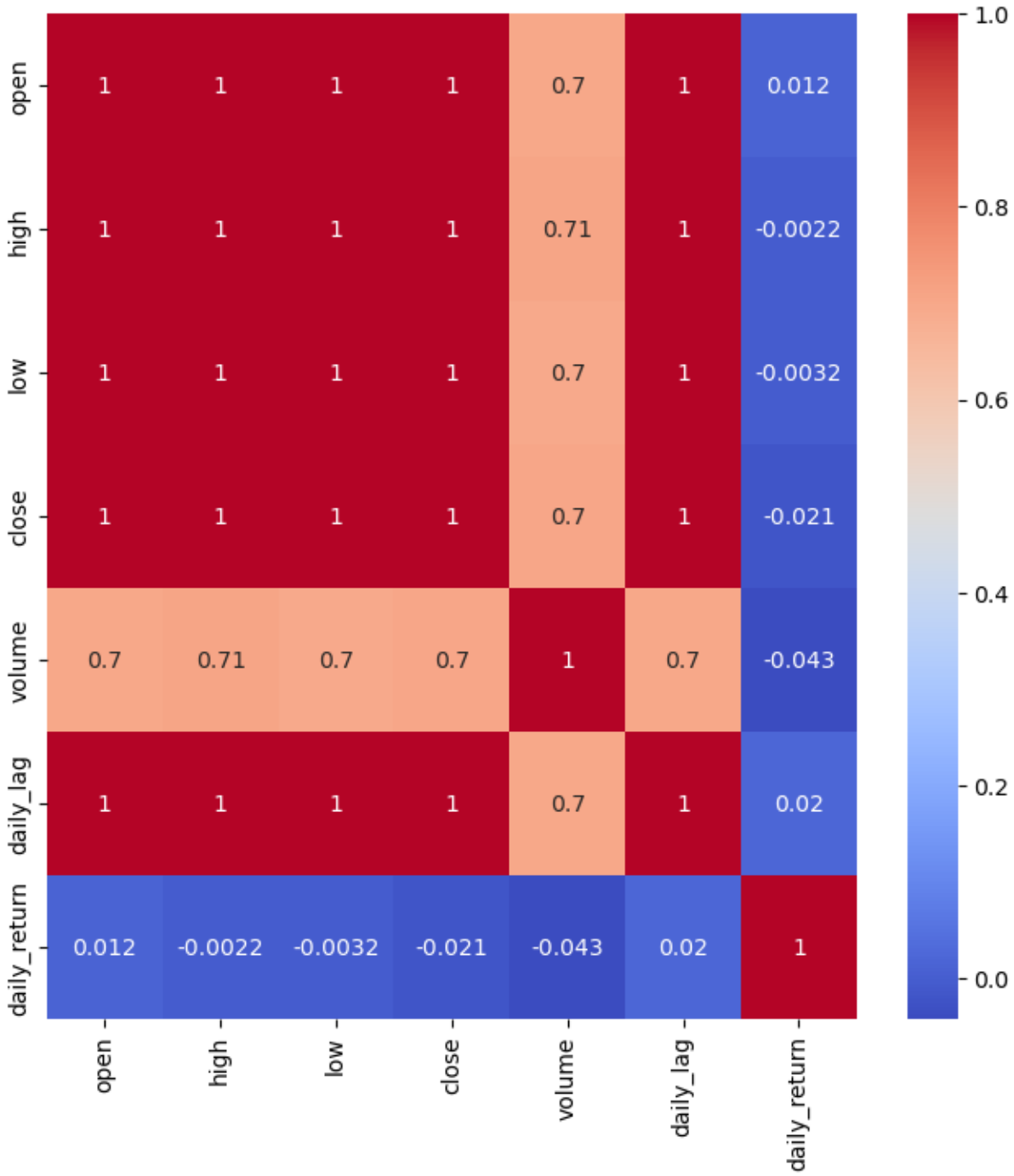
and the graph also shows that the closing price of SSI stock has never exceeded the VND50,000 mark.



One remark worth mentioning is that the distribution of 'daily return' closely resembles the normal distribution. The distribution of ‘daily_return’ fluctuates in the range from -0.7 to

0.8, which is consistent with the variation range of the ceiling price and floor price regulated on the Vietnamese stock exchange. The two tails of the distribution chart are suddenly unusually higher than neighboring values, indicating that SSI stock not only excitedly touched the ceiling price many times but also fell to the floor price within a day.

2.1.3. Correlation



The provided correlation matrix indicates the correlation coefficients between different columns (features) in a dataset. The values in the matrix range between -1 and 1. A value of 1 signifies a perfect positive correlation, meaning the two variables move in perfect harmony in the same direction. A value of -1 represents a perfect negative correlation, indicating a perfect inverse relationship between the variables. A value close to 0 suggests a weak or no linear relationship between the variables.

Strong Positive Correlations: The 'open', 'high', 'low', and 'close' columns have strong positive correlations among themselves, indicating that these variables are highly positively correlated. This suggests that as one of these variables increases, the others tend to increase as well. This is expected as they are different aspects of stock prices.

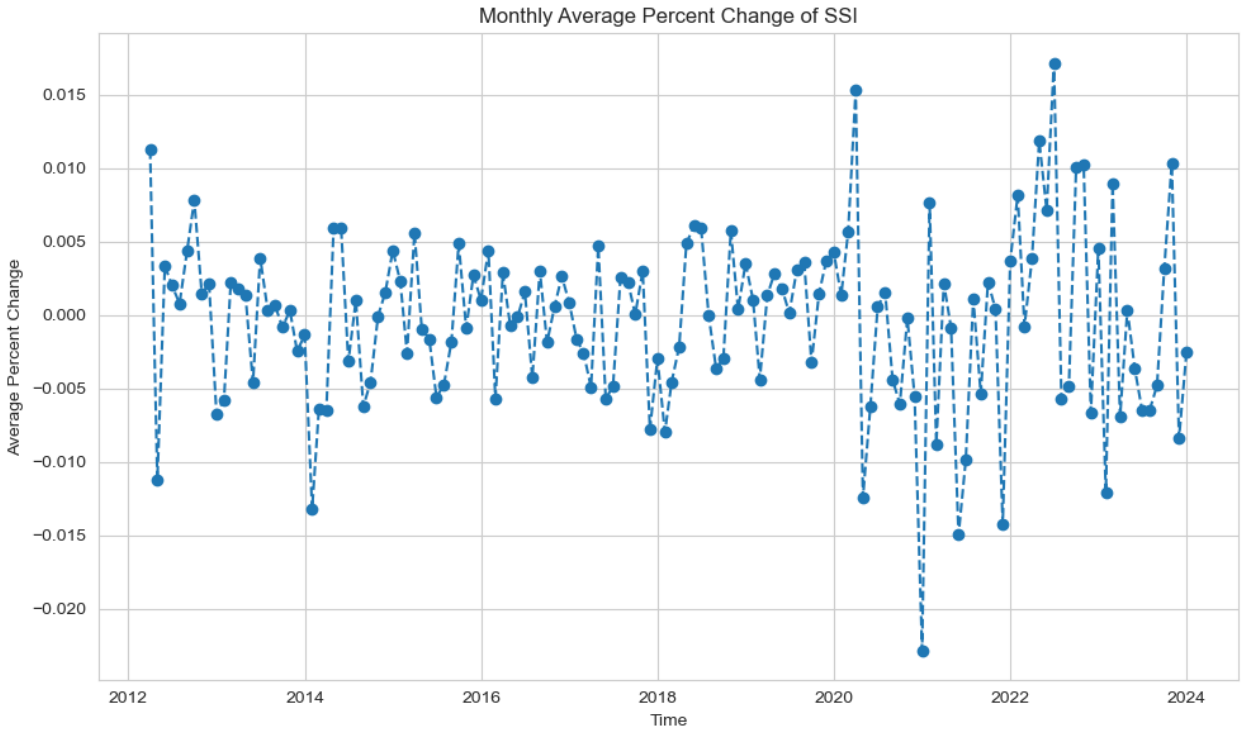
Volume Correlations: The 'volume' column has moderate positive correlations with 'open', 'high', 'low', 'close', and 'daily_lag'. This suggests some level of positive association between trading volume and stock prices or previous day's stock prices; indicating some association between trading volume and price movements.

Daily Return Correlations: The 'daily_return' column has very weak correlations with other columns ('open', 'high', 'low', 'close', 'volume', 'daily_lag', 'time'), mostly close to zero or negative, except for a slight positive correlation with 'daily_lag'.

Time Correlations: The 'time' column seems to have moderate positive correlations with the price-related columns ('open', 'high', 'low', 'close'), indicating some level of temporal correlation between time and stock prices.

2.1.4. Monthly average percentage change

From 'daily_return', it can calculate the monthly average percentage change of SSI stock price. It is clear that before 2020, price fluctuations in a month were quite stable, mostly around -5% to 5%. However, from the beginning of 2020 onwards, price volatility in one month has become more abusive, with the highest profit level being above 15% while the record loss can reach more than 20%.

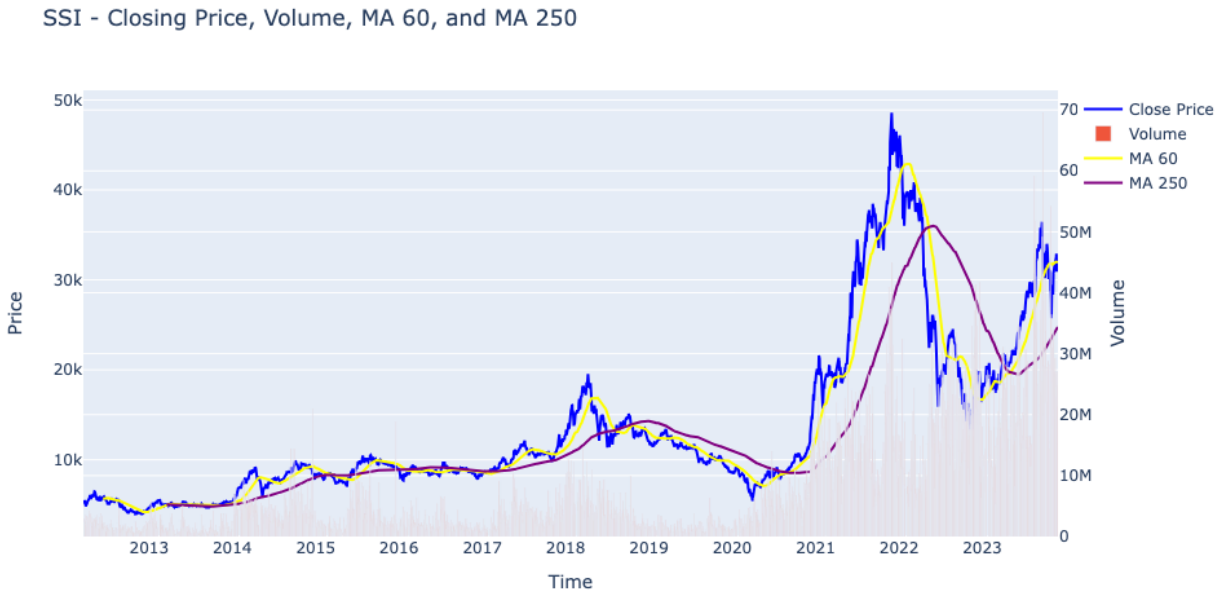


2.1.5. Closing price over time

The chart showing SSI stock prices over time once again confirms the strong fluctuations of the stock market since 2020. SSI stocks have witnessed a strong increase reaching a peak within two years from 2020 to 2022 Although 2020 was the year of the Covid-19 pandemic and full of difficulties, the stock market witnessed a strong increase. When the economy shows signs of recovery in 2022, stock prices will decrease. This shows that capital flow circulating in the economy does not go hand in hand but precedes economic development. Especially with stocks in the financial sector like SSI, this phenomenon is even more evident.

A commonly used sign in determining buying and selling points in technical analysis is the moving average (MA). The moving average is a simple technical analysis tool that smooths out price data by creating a constantly updated average price. The average is taken over a specific period of time, like 10 days, 20 minutes, 30 weeks, or any time period the trader chooses. Here the author chooses moving average for 60 days (MA60) and 250 days (MA250) to determine medium and long-term trading signals. Simply put, if the price cuts above the MA60 or MA250 line, it represents a buy signal, and if the price cuts below the

MA60 line, it is a sell signal. However, signals often appear very rarely, which causes many investors to miss out on chances.



2.2. Traditional Econometric Analysis

This project employed the ARIMA (AutoRegressive Integrated Moving Average) models to perform a traditional econometric method to predict stock price. This model is a popular choice in forecasting stock prices owing to its efficacy in handling time series data patterns.

ARIMA is specified by three order parameters (p,d,q):

- **AR (p) Autoregression:** An autoregressive (AR(p)) component refers to the use of past values in the regression equation for the time series.
- **I (d) Integration** uses differencing of observations (subtracting an observation with an observation from the previous step) in order to make the time series stationary.
- **MA (q) Moving Average:** A moving average component depicts the error of the model as a combination of previous error terms. The order q represents the number of terms to be included in the model.

2.2.1. Testing the stationary of the series

This project uses the Augmented Dickey-Fuller (ADF) Test to verify if the series is stationary or not. The null and alternate hypotheses for the ADF Test are defined as:

- **Null hypothesis:** The Time Series is non-stationary.
- **Alternative hypothesis:** The Time Series is stationary.

As the result, ADF statistics is bigger than critical values, meaning that we cannot reject the null-hypothesis stating that there is the presence of unit root. So our time-series is unstationary. We should make it stationary first.

```
df_close = arima['close']

# Augmented Dickey-Fuller test for staionarity
result = adfuller(df_close)
print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])
print('Critical Values:')
for key, value in result[4].items():
    print('\t%s: %.3f' % (key, value))

# the series is not stationary
```

```
ADF Statistic: -0.977165
p-value: 0.761501
Critical Values:
        1%: -3.433
        5%: -2.863
        10%: -2.567
```

2.2.2. Making the series stationary

We can use some of the following methods to convert a non-stationary series into a stationary one by log transformation or by differencing the series (lagged series). In this project, we use a difference order of 1 to make the time series stationary.

```
# get the differences of the observations for stationarity
df_close1 = df_close.diff()
df_close1 # first row should be dropped due to the NaN value
```

```
# drop the NaN row
df_close1.dropna(inplace=True)
df_close1 # now the series one less --> 2943
```

```
time
2012-03-21     90.0
2012-03-22   -110.0
2012-03-23    190.0
2012-03-26    -80.0
2012-03-27   -270.0
              ...
2023-12-22    150.0
2023-12-25    199.0
2023-12-26    201.0
2023-12-27     50.0
2023-12-28    200.0
Name: close, Length: 2943, dtype: float64
```

```
# check again to see if the series get stationary
result = adfuller(df_close1, regression='ctt')
print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])
print('Critical Values:')
for key, value in result[4].items():
    print('\t%s: %.3f' % (key, value))
```

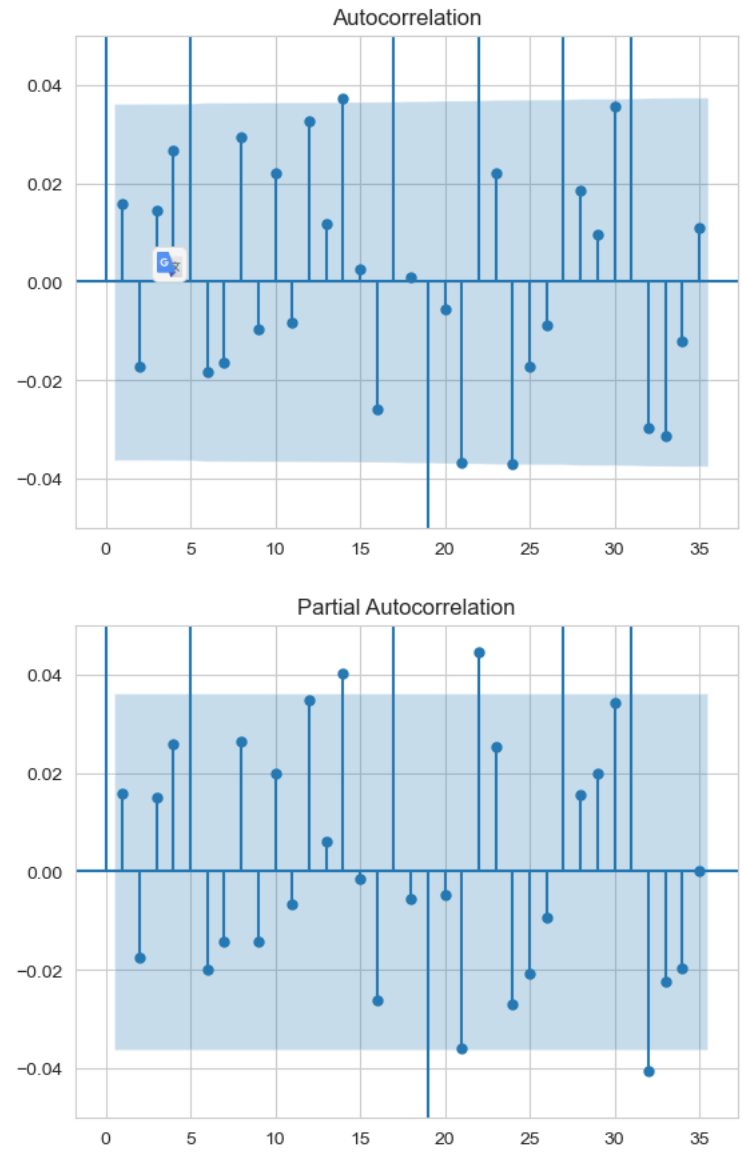
```
ADF Statistic: -9.742292
p-value: 0.000000
Critical Values:
        1%: -4.375
        5%: -3.834
        10%: -3.555
```

This time our series got stationary in every confidence level (smaller than all critical values and p-value suggest the rejection of the null-hypothesis).

2.2.3. Selecting parameters

First, we will plot the ACF and PACF plots to get the values of p and q. From the above PACF plot, we can see that the highest lag at which the plot extends beyond the statistically significant boundary is lag 1. This indicates that an AR Model of lag 1 (p=1) should be sufficient to fit the data. Similarly, from the ACF plot, we can infer that q=1.

```
(-0.05, 0.05)
```



Then this project use Auto-ARIMA to determine the values of p,d,q. As the result, the best model is ARIMA (1, 0, 1).

```
from pmdarima.arima import auto_arima

# we will set (p,d,q)start = 0 and (p,d,q)max = 2
arima_model = auto_arima(train_data, start_p = 0, d=0, start_q = 0,
                         max_p = 3, max_d=2, max_q = 3,
                         seasonal = False, # there is no seasonality
                         trace = True,
                         test = 'adf', # this will find optimal 'd'
                         error_action ='ignore',
                         suppress_warnings = True,
                         stepwise = True, n_fits=50)
```

```
Performing stepwise search to minimize aic
 ARIMA(0,0,0)(0,0,0)[0]             : AIC=50634.463, Time=0.07 sec
 ARIMA(1,0,0)(0,0,0)[0]             : AIC=inf, Time=0.66 sec
 ARIMA(0,0,1)(0,0,0)[0]             : AIC=inf, Time=0.93 sec
 ARIMA(1,0,1)(0,0,0)[0]             : AIC=33438.883, Time=3.06 sec
 ARIMA(2,0,1)(0,0,0)[0]             : AIC=33439.192, Time=3.00 sec
 ARIMA(1,0,2)(0,0,0)[0]             : AIC=33440.102, Time=2.25 sec
 ARIMA(0,0,2)(0,0,0)[0]             : AIC=inf, Time=5.81 sec
 ARIMA(2,0,0)(0,0,0)[0]             : AIC=inf, Time=0.66 sec
 ARIMA(2,0,2)(0,0,0)[0]             : AIC=33441.217, Time=1.26 sec
 ARIMA(1,0,1)(0,0,0)[0] intercept   : AIC=33440.595, Time=2.06 sec

Best model:  ARIMA(1,0,1)(0,0,0)[0]
Total fit time: 19.809 seconds
```

2.2.4. Preparing and Training data

We use 80% of the dataset to train the model ARIMA (1, 0, 1) and the remaining 20% to test the prediction performance.

2.2.5. Testing data

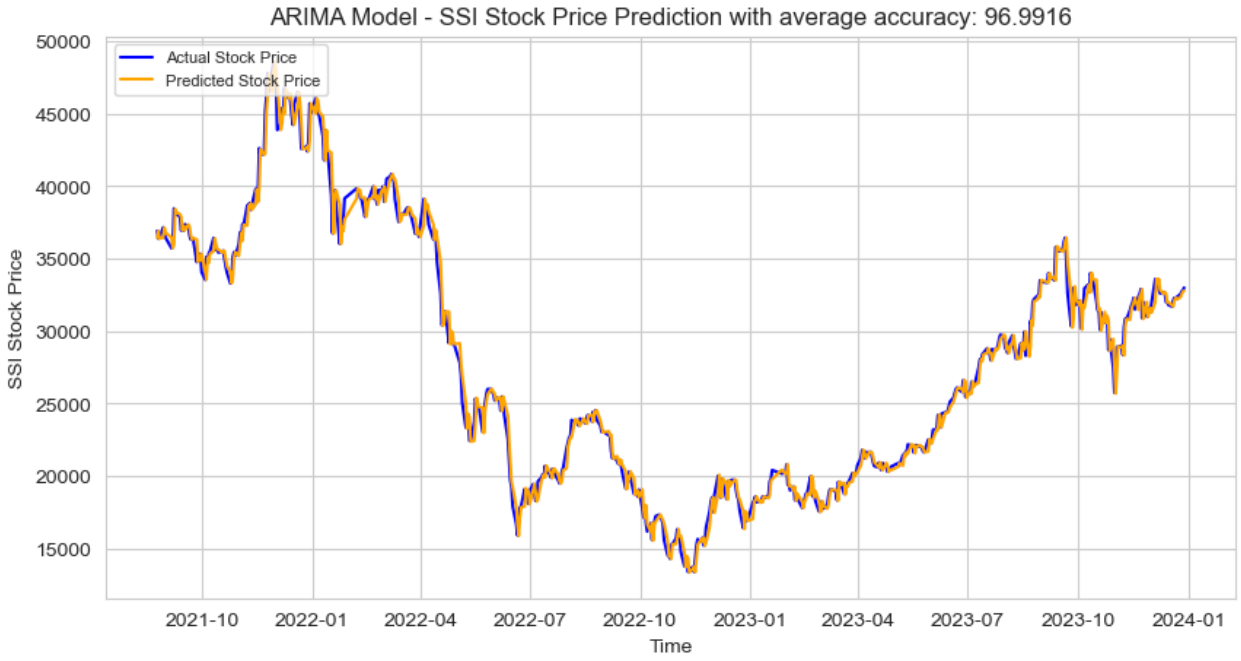
For testing data, this project performs a walk-forward validation using an ARIMA model for time series forecasting. Walk-forward validation is the process that repeats for each time

step in the test data, gradually incorporating actual observations from the test set into the historical data used for subsequent predictions. This approach simulates real-world scenarios by retraining the model at each step with updated historical data and evaluating its predictions.

```
# evaluate an ARIMA model using a walk-forward validation
import time

# Record start time of the execution
start = time.time()

# Initialize history with training data
history = [x for x in train_data]
predictions = list()

# Iterate through each data point in the test data
for t in range(len(test_data)):
    # Create ARIMA model using historical data in history
    model = ARIMA(history, order=(1, 0, 1))

    # Fit ARIMA model to historical data
    model_fit = model.fit()

    # Forecast the next value
    output = model_fit.forecast()

    # Retrieve the forecasted value
    yhat = output[0]

    # Append the predicted value to predictions list
    predictions.append(yhat)

    # Get the actual observed value from test_data
    obs = test_data[t]

    # Update history with the observed value for the next iteration
    history.append(obs)

    # Print predicted and expected values for each iteration
    print('Predicted=%f, Expected=%f ' % (yhat, obs))

# Record end time of the execution
end = time.time()

# Calculate and print total execution time
print("Total time:", end - start, "seconds")
```

2.2.6. Evaluating model performance

As we can see ARIMA gives a very good prediction of the real stock price with the accuracy being 96.99%.



	MSE	RMSE	MAE	MAPE	R-square
ARIMA	667859.13	817.27	592.65	2.22	0.99

Accuracy: The accuracy metric assesses how well predicted values align with actual values, factoring in their relative differences as a percentage of the actual values themselves. The closer the percentage is to 100%, the better the predictions align with the actual values.

Mean Squared Error (MSE): The MSE of 667859.13 indicates the average squared difference between the predicted and actual values. A lower MSE suggests better accuracy, though the interpretation might vary based on the scale of the data.

Root Mean Squared Error (RMSE): With an RMSE of 817.27, this metric represents the square root of MSE, providing an interpretable measure in the same units as the target variable. Lower RMSE values indicate better predictive performance.

Mean Absolute Error (MAE): The MAE of 592.65 measures the average absolute differences between predicted and actual values. It's less sensitive to outliers compared to MSE and RMSE.

Mean Absolute Percentage Error (MAPE): The MAPE of 2.22% represents the average percentage difference between predicted and actual values. It's a relative measure and can be useful for understanding the model's performance in terms of percentage accuracy.

R-squared: The R-squared value of 0.99 indicates the proportion of variance in the target variable explained by the model. A value closer to 1 suggests that the model captures a high percentage of the variance in the data, indicating a strong fit.

Overall, the ARIMA model demonstrates strong performance across multiple metrics. The low values for MSE, RMSE, MAE, and MAPE, along with the high R-squared value, indicate that the model provides accurate predictions and effectively captures patterns in the data.

3. AI MODELS ANALYSIS

In this project, we employ 3 AI models, which are long short-term memory (LSTM), Gated Recurrent Unit (GRU) and transformer model. In which, both GRU and LSTM belong to traditional recurrent neural networks suitable for sequential data like stock prices. For Transformer, although originally designed for natural language processing, it is also applicable to sequential data and efficiently captures dependencies across the entire sequence length due to its self-attention mechanism to understand long-term patterns. Also, the parallel processing capability can facilitate for faster training, especially with large datasets.

3.1. Steps in prediction stock prices using AI models

1. Importing packages and data
2. Preparing Training (80%) and Testing (20%) set
3. Choosing the timesteps: It is worth mentioning that the choice of the number of timesteps is important in both recurrent neutral networks or transformers. For recurrent neutral networks, the optimum is usually between 40-90 timesteps. but for the transformers the optimum choice is often much lower, with usually under 10. In

this project, we create a data structure with 60 timesteps for LSTM and GRU, and 8 timesteps for the transformer model.

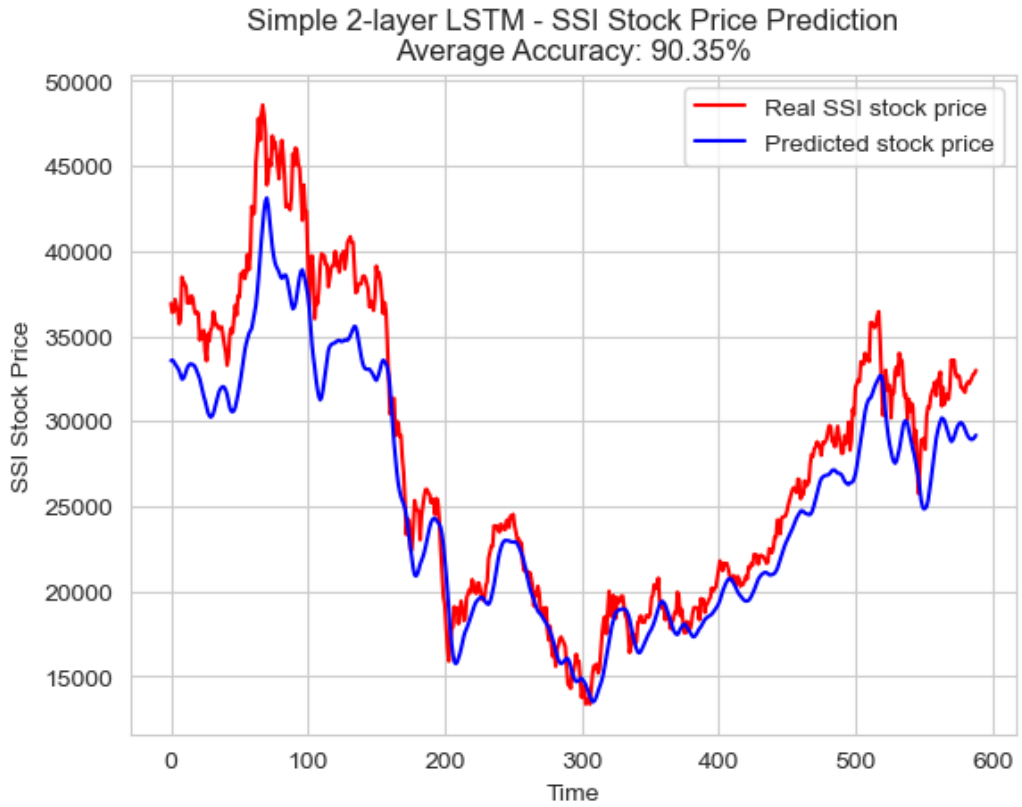
4. Building and Training model: To avoid overfitting, our project have a training strategy which is setting the Learning Rate Scheduler. This learning rate scheduler has a warm-up phase followed by a linear decay phase. This helps in stabilizing the training process at the beginning and gradually reducing the learning rate to refine the model. Second, we use EarlyStopping callbacks for stopping training when the model's performance on a validation set is not improved.
5. Testing and Evaluating model performance: Metrics used are Accuracy, MSE, RMSE, MAE, MAPE and R-squared.

3.2. Results

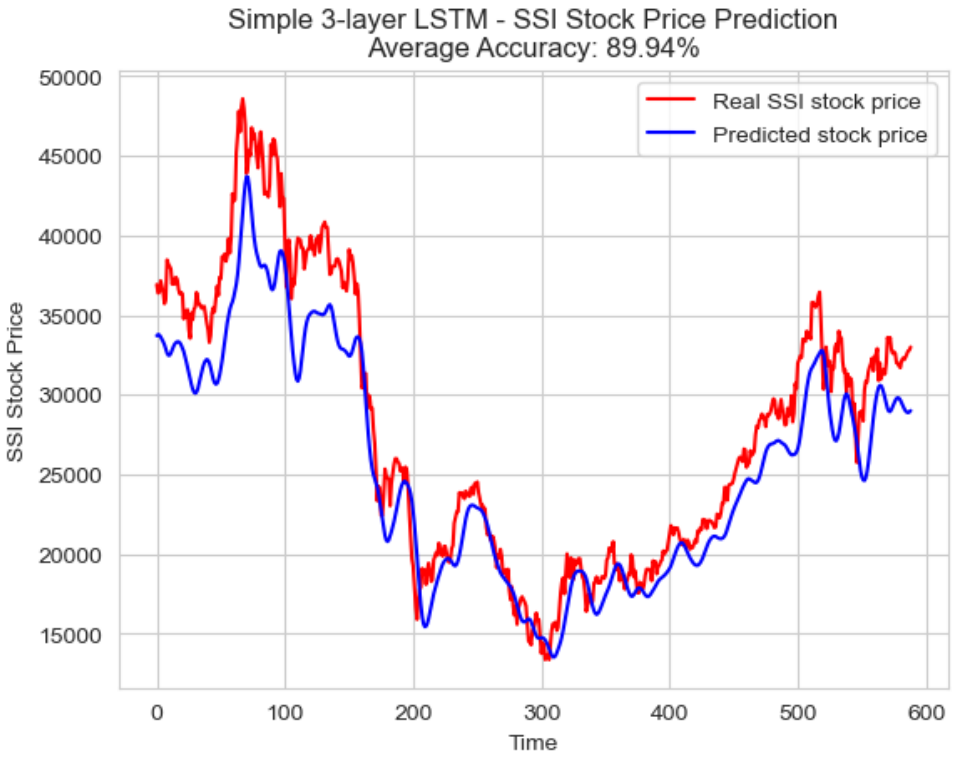
3.2.1. LSTM model

3.2.1.1. Same Architectured Layers

The 2-same-layer LSTM



The 3-same-layer LSTM



Observations:

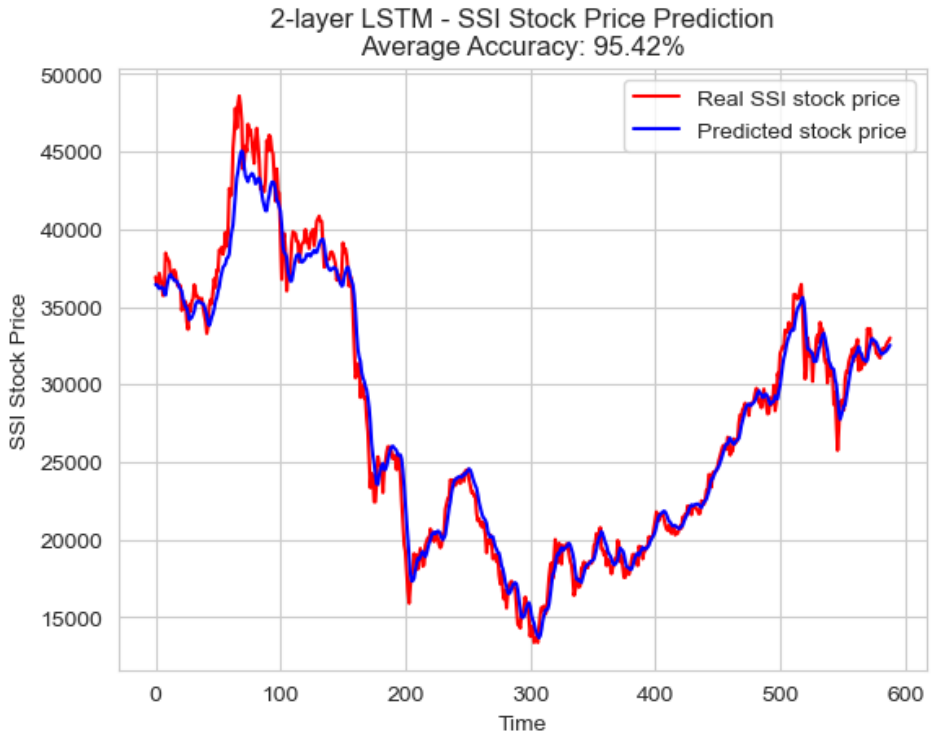
- The result of 2-layer and 3-layer LSTM shows that the increase in layer of this this same achitectured layer LSTM model does not improve the prediction accuracy.

3.2.1.2. Different Architectured Layers

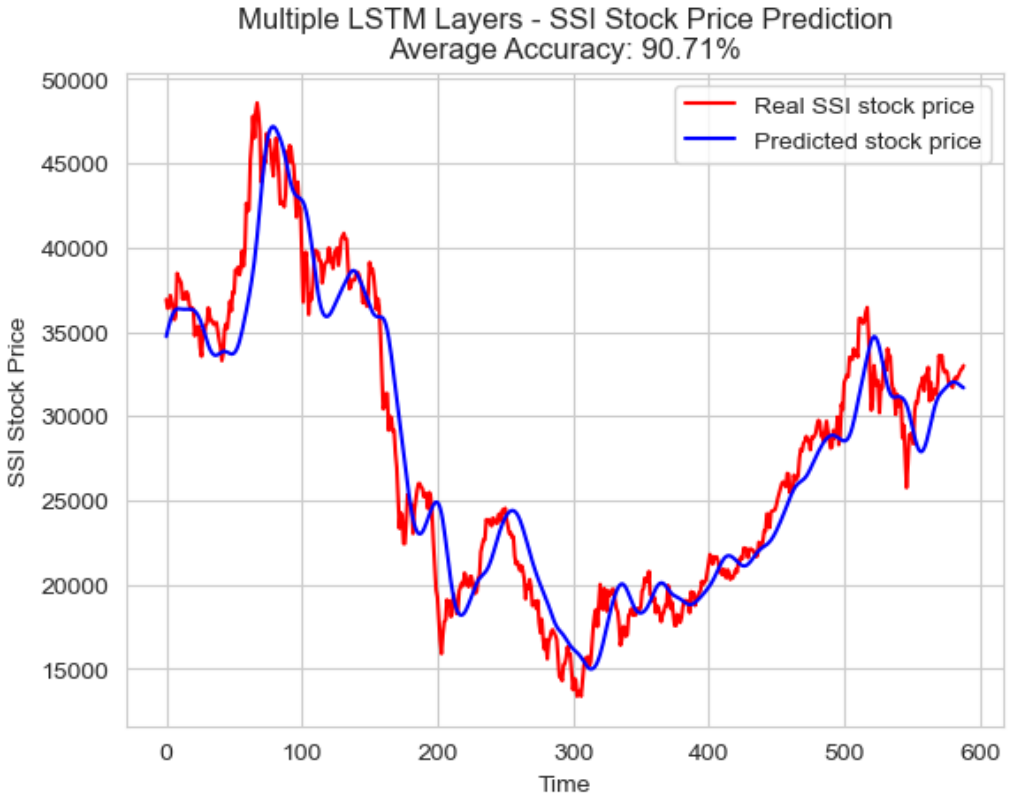
Observations:

- An increase in layers do not lead to increase in model performance.
- Different architectured layer LSTM shows better prediction accuracy and the same architectured layer LSTM.

The 2-different-layer LSTM

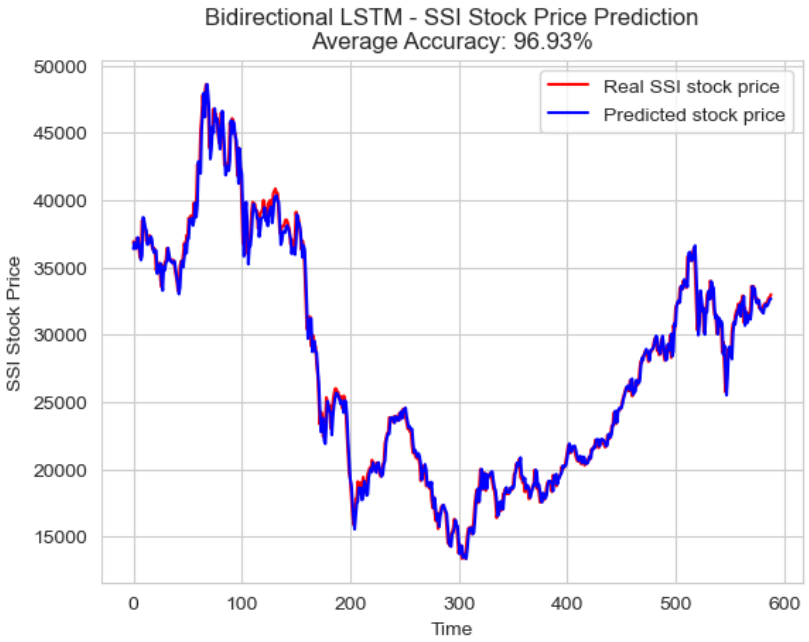


The 4-different-layer LSTM

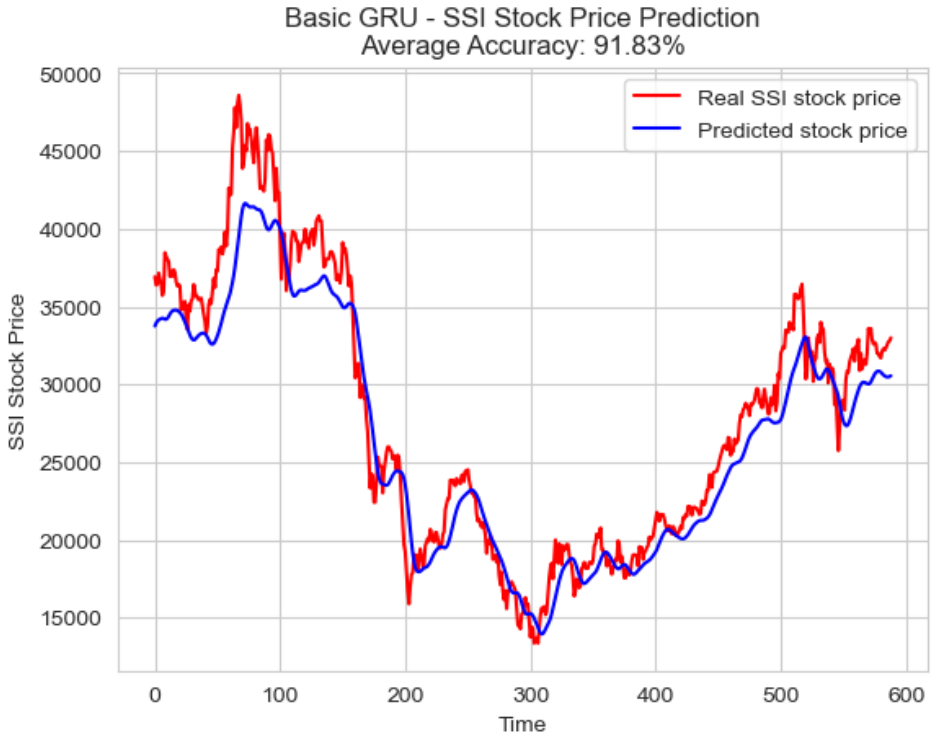


3.2.1.3. Bidirectional LSTM

The bidirectional LSTM shows more predicting power than the previous LSTM models with nearly 97%.



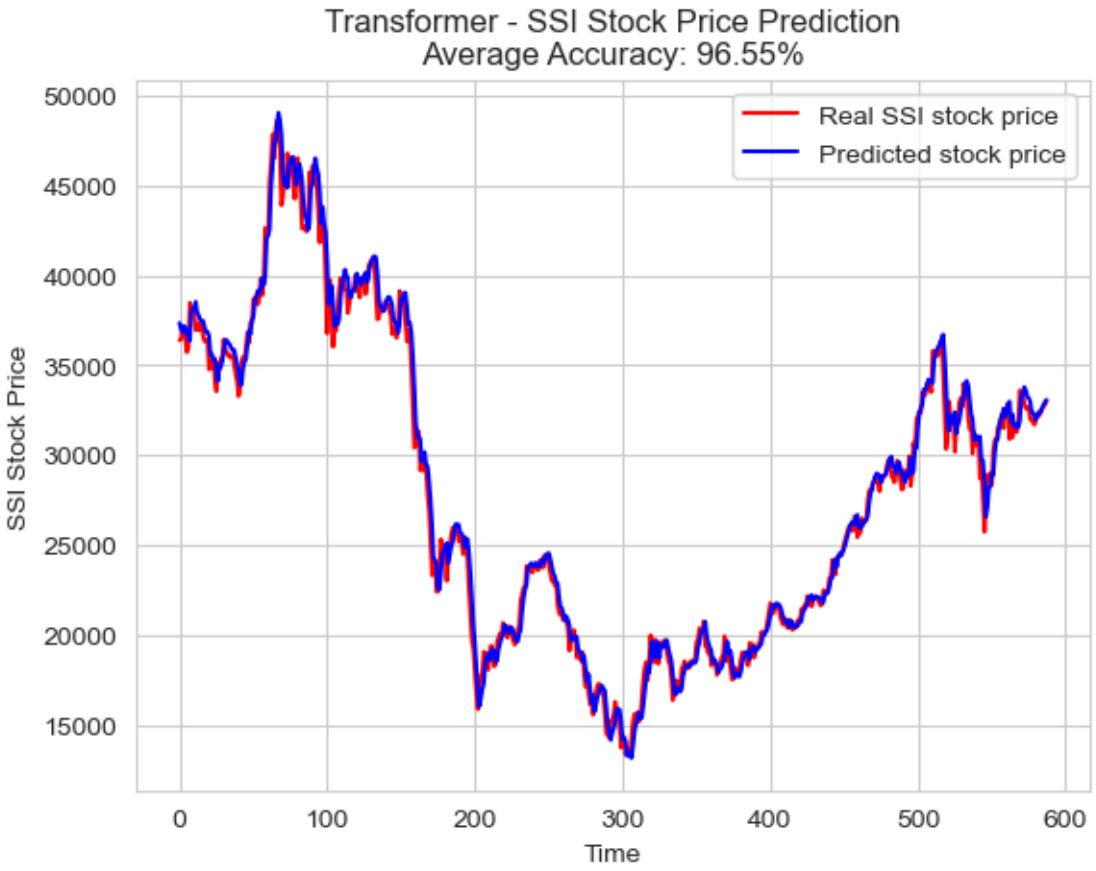
3.2.2. GRU Model



The result shows that the 4-same-layer GRU with more than 91% has better prediction than same-layer LSTM architectures although this accuracy of GRU is not as high as the other complex LSTM model.

3.2.3. Transformer Model

The accuracy of predictions using transformer, which is more than 96%, just lower than bidirectional LSTM model and ARIMA model.



4. CONCLUSION

Model	Accuracy	MSE	RMSE	MAE	MAPE	R-Square
ARIMA	96.99	667859.13	817.27	592.65	2.22	0.99
LSTM (2-same-layer)	90.35	10394602.69	3224.07	2589.03	8.47	0.86
LSTM (3-same-layer)	89.94	10789005.30	3284.66	2637.78	8.75	0.86
LSTM (2-different layer)	95.42	1628088.71	1275.97	904.04	3.30	0.98
LSTM (4-different-layer)	90.71	5716696.28	2390.96	1826	6.93	0.92
LSTM (Bidirectional)	96.93	710032.77	842.63	620.58	2.30	0.99
GRU (4-same-layer)	91.83	91.83	2499.91	1975.22	6.83	0.92
Transformer	96.55	858497.79	926.55	658.19	2.48	0.99

In conclusion, when comparing both traditional echometric and deep learning models, surprisingly the traditional one gives the highest results in average accuracy, the second place is the bidirectional LSTM model, and the Transformer holds the third place. These three models all have the very high R-Square (0.99) suggests a good fit to the data. Models like LSTM (2-different-layer), Bidirectional LSTM, and Transformer demonstrate

promising performance with high accuracy and low error metrics, making them competitive alternatives to ARIMA for stock price prediction.

There are several contrasts between the ARIMA model and these AI models:

- Interpretability: ARIMA models are relatively simpler and easier to interpret compared to complex neural network architectures.
- Training Time: AI models often require more computational resources and training time compared to ARIMA, especially with deep learning models like LSTM, GRU, and Transformer.
- Performance: While ARIMA performs admirably, some AI models demonstrate comparable accuracy and superior performance in specific cases, particularly when handling nonlinear relationships or capturing complex patterns in data.

ARIMA outperforms most neural network-based models in terms of accuracy and error metrics, except for the LSTM (2-different-layer) and Transformer models. The first reason is that the length of the sequence being predicted is short, just one output. It is believe that as longer predicted length of output the AI models will outperform. Also, this may because the ARIMA model we have use automation to choose the best parameter while these deep learning model can vary significantly on hyperparameters. The choice of architecture and the number of layers significantly affect the performance, as seen in variations across LSTM models. Adjusting these factors might further improve model accuracy.

In essence, while ARIMA shows strong performance and interpretability, AI models present competitive accuracy and predictive capabilities. The choice between ARIMA and AI models often depends on the specific context, computational resources, interpretability needs, and the nature of the dataset. AI models may offer advantages in capturing intricate patterns but may require more resources and expertise in model tuning.

5. REFERENCES

(n.d.). Retrieved from https://github.com/034adarsh/Stock-Price-Prediction-Using-LSTM/blob/main/LSTM_Improved_model(diff_dataset).ipynb

(n.d.). Retrieved from https://www.kaggle.com/code/faressayah/stock-market-analysis-prediction-using-lstm

(n.d.). Retrieved from https://github.com/hungchun-lin/Stock-price-prediction-using-GAN/tree/master/Code/NLP

(n.d.). Retrieved from https://www.kaggle.com/code/bryanb/stock-prices-forecasting-with-lstm

(n.d.). Retrieved from https://www.kaggle.com/code/fredblair/transformers-for-stocks/notebook

(n.d.). Retrieved from https://github.com/swapnilin/Stock-Prediction-using-LSTMs/blob/main/Stock%20Market%20Models-ARIMA%2C%20Feature%20Extraction%2C%20LSTM.ipynb

(n.d.). Retrieved from https://github.com/PabloAguirreSolana/Forecasting-Stock-Prices/blob/main/Time_Series.ipynb

(n.d.). Retrieved from https://github.com/DivyaJagarlapoodi/Amazon-Stock-Price-Prediction/blob/master/Amazon%20Stock%20Price%20prediction.ipynb

(n.d.). Retrieved from https://www.kaggle.com/code/dpamgautam/stock-price-prediction-lstm-gru-rnn

6. APPENDIX

JUPYTER NOTEBOOK

https://drive.google.com/file/d/1uq_Hj54Nck6bwfNA9h-wM_MekI_GwFcf/view?usp=sharing