COMP 5313- ARTIFICIAL INTELLIGENCE

PROJECT 1

Stock Prices Prediction Using AI and Machine Learning

Methodology:

I created one .py file in order to create stock prices prediction using AI and Machine Learning.

The dataset that I used to train the models is the Samsung dataset from finance.yahoo.com.

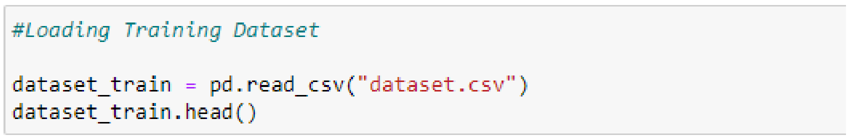
I used two models to predict the stock prediction like LSTM and RNN.

Dataset:

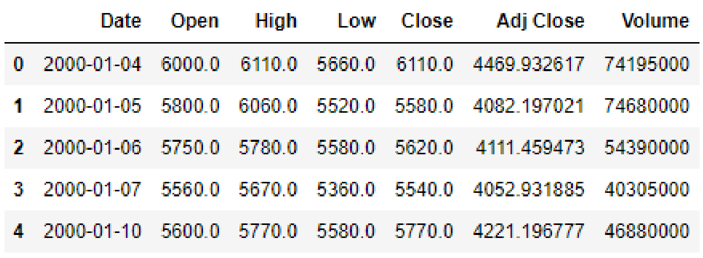
I downloaded Samsung dataset from finance.yahoo.com. I got the historical data for the past 24 years from January 02, 2000 to February 14, 2024. The dataset contained seven columns. They are

- Date
- Open
- High
- Low
- Close
- Adj Close
- Volume

From this columns I used Open column for prediction.



```
#Loading Training Dataset

dataset_train = pd.read_csv("dataset.csv")
dataset_train.head()
```



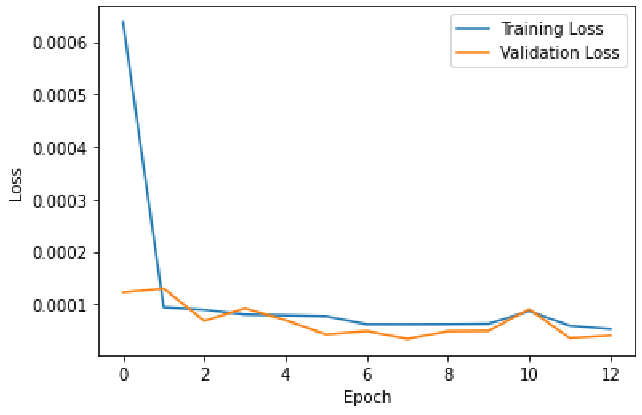
	Date	Open	High	Low	Close	Adj Close	Volume
0	2000-01-04	6000.0	6110.0	5660.0	6110.0	4469.932617	74195000
1	2000-01-05	5800.0	6060.0	5520.0	5580.0	4082.197021	74680000
2	2000-01-06	5750.0	5780.0	5580.0	5620.0	4111.459473	54390000
3	2000-01-07	5560.0	5670.0	5360.0	5540.0	4052.931885	40305000
4	2000-01-10	5600.0	5770.0	5580.0	5770.0	4221.196777	46880000

LSTM- Long Short-Term Memory Network

Long short-term memory (LSTM) network is a recurrent neural network (RNN), aimed to deal with the vanishing gradient problem present in traditional RNNs.

In this I added four LSTM layers sequentially and each LSTM layer has 64 units.

```
# Plotting training and validation loss

plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.show()
```



As you can see above, I plotted the training and validation losses.

```
# Calculate Mean Squared Error (MSE) and Mean Absolute Error (MAE) for testing data

test_mse = np.mean(np.square(predicted_stock_price - actual_stock_price))
test_mae = np.mean(np.abs(predicted_stock_price - actual_stock_price))
print("Test MSE:", test_mse)
print("Test MAE:", test_mae)
```

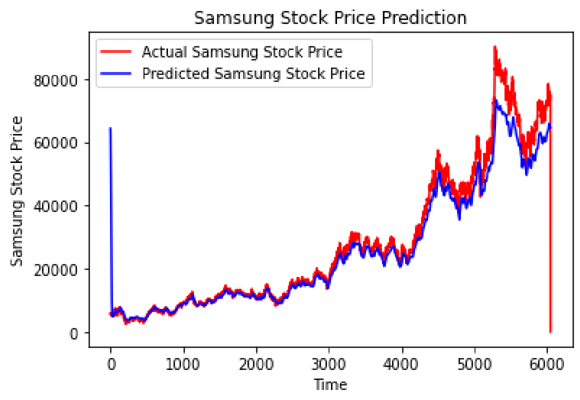
```
Test MSE: 22859537.500968315
Test MAE: 2782.497258939092
```

```
# Printing the Mean Squared Error (MSE)

train_mse = history.history['loss'][-1]
print("Train MSE:", train_mse)
```

```
Train MSE: 5.279506876831874e-05
```

Then I calculated Mean Squared Error (MSE) for training and testing data then Mean Absolute Error for testing data.

```
# Plotting

plt.plot(actual_stock_price, color='red', label='Actual Samsung Stock Price')
plt.plot(predicted_stock_price, color='blue', label='Predicted Samsung Stock Price')
plt.title('Samsung Stock Price Prediction')
plt.xlabel('Time')
plt.ylabel('Samsung Stock Price')
plt.legend()
plt.show()
```



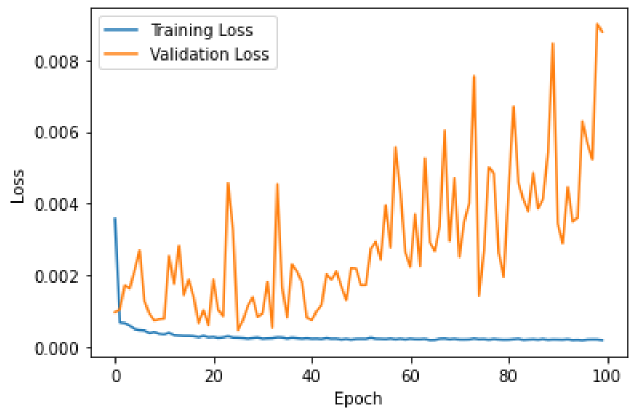
This graph represents the Actual Samsung stock price in red and Predicted Samsung stock price in blue.

RNN- Recurrent Neural Network

A recurrent neural network (RNN) is a deep learning model that is trained to process and convert a sequential data input into a specific sequential data output.

In this I added three LSTM layers sequentially and each LSTM layer has 50 units.

```
# Plot training and validation loss

plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.show()
```



As you can see above, I plotted the training and validation losses.

```
# Calculate Mean Squared Error (MSE) and Mean Absolute Error (MAE) for testing data

test_mse = np.mean(np.square(predictions - Y_test))
test_mae = np.mean(np.abs(predictions - Y_test))
print("Test MSE:", test_mse)
print("Test MAE:", test_mae)
```

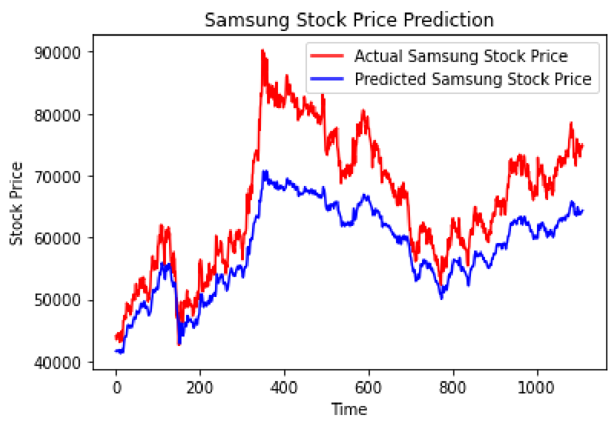
```
Test MSE: 71643911.31610885
Test MAE: 7524.791362573264
```

```
# Print Mean Squared Error (MSE) for training data

train_mse = history.history['mean_squared_error'][-1]
print("Train MSE:", train_mse)
```

```
Train MSE: 0.00017887516878545284
```

Then I calculated Mean Squared Error (MSE) for training and testing data then Mean Absolute Error for testing data.

```
plt.plot(Y_test, color='red', label='Actual Samsung Stock Price')
plt.plot(predictions, color='blue', label='Predicted Samsung Stock Price')
plt.title('Samsung Stock Price Prediction')
plt.xlabel('Time')
plt.ylabel('Stock Price')
plt.legend()
plt.show()
```



This graph represents the Actual Samsung stock price in red and Predicted Samsung stock price in blue.

Conclusion:

As you can see from the above results the LSTM has a Train MSE: 5.28, Test MSE: 22859537.50 and Test MAE: 2782.4972 then RNN has Train MSE: 0.00017887516878545284, Test MSE: 71643911.31610885 and Test MAE: 7524.791362573264. From this we can see LSTM has lower MSE and MAE for both training and testing data compared to RNN model. RNN model has higher errors indicating poorer performance. Considering these points and the graph we can say LSTM model appears to be best.

References:

[1] https://aws.amazon.com/what-is/recurrent-neural-network/#:~:text=A%20recurrent%20neural%20network%20(RNN,a%20specific%20sequential%20data%20output.

[2] https://en.wikipedia.org/wiki/Long_short-term_memory#:~:text=Long%20short%2Dterm%20memory%20(LSTM,and%20other%20sequence%20learning%20methods.