Quantum Computing Companies Stock Price Prediction

Xiaoyan Hua

Mingqian Chen

Why We Want to Learn Quantum Computing Companies Stock Prices?

- 1. Current state of quantum computing market
- 2. Predict the quantum computing market trend
- 3. Which company to invest
- 4. The future of market and impact on academia

Outline

- Theoretical Background:
  - Linear regression
  - Recurrent neural network (RNN)
  - Long short-term memory (LSTM)
- Data Analysis:
  - Data preparation
  - Comparison of the model performance
    - linear regression with time series, improved linear regression with SMA
    - LSTM, improved LSTM
- Conclusion

Companies to Predict

- Companies to predict:
  - ACN IBM MSFT QNC.V INTC BIDU NOK MIELY
- Dataset:
  - Kaggle Project (quantumstock1.csv)
  - Quantum Computing Company Report

Methods to Solve Stock Predictions

- Linear Regression

$$y_t = \beta_0 + \beta_1 X_t$$

- Recurrent Neural Network(RNN)

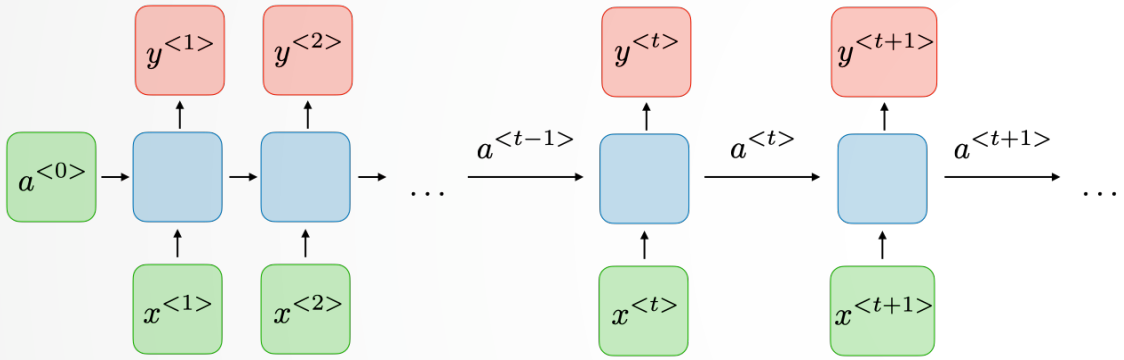
$$h_t = \sigma_h(W_h x_t + U_h h_{t-1} + b_h)$$
$$y_t = \sigma_y(W_y h_t + b_y)$$

- Long Short-Term Memory (LSTM)

$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f)$$
$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i)$$
$$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o)$$
$$\tilde{c}_t = \sigma_c(W_c x_t + U_c h_{t-1} + b_c)$$

$$c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t$$
$$h_t = o_t \circ \sigma_h(c_t)$$

Naive Recurrent Neural Network (RNN)



1. Allow previous outputs to be used as inputs while having hidden states

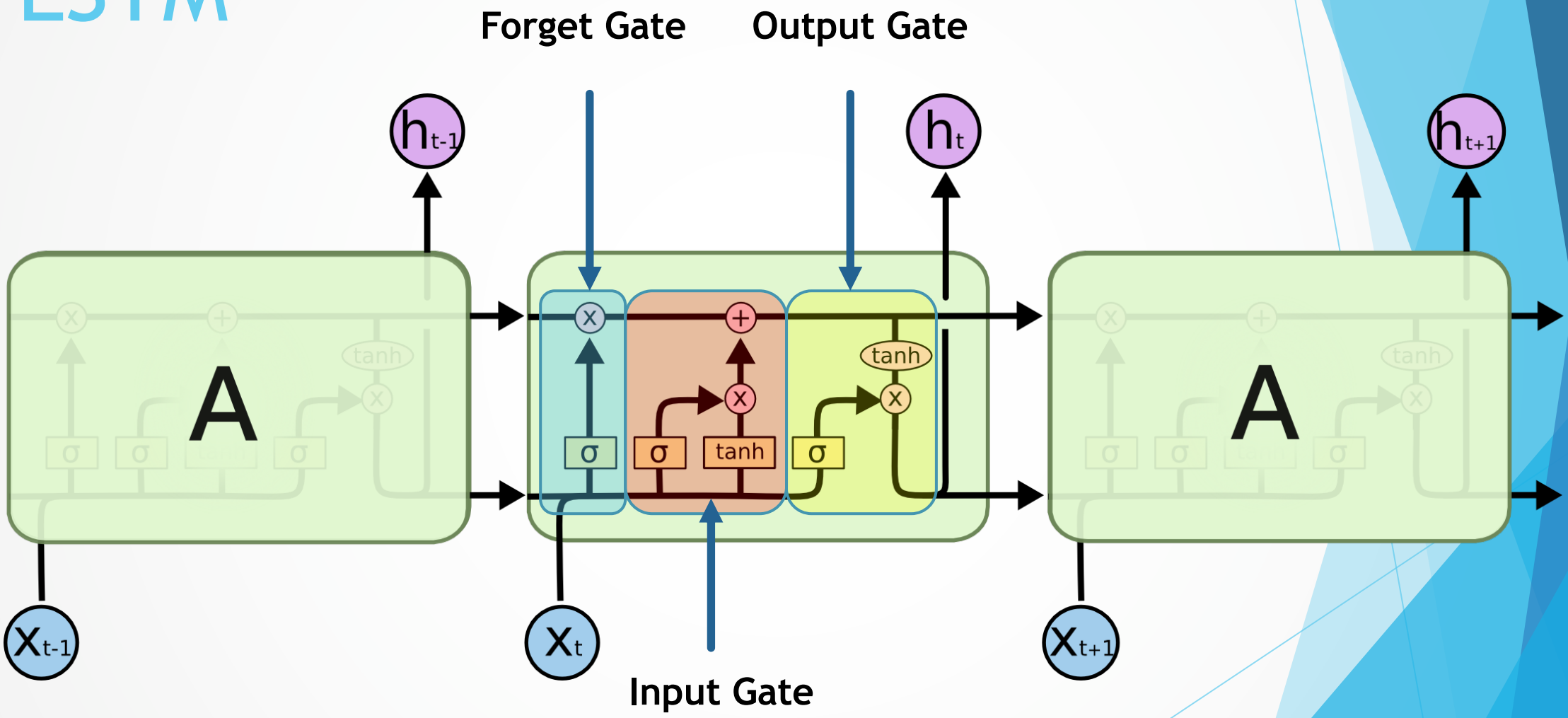
2. Gradient vanishing/exploding

Gradient Vanishing

- The optimizer of RNN gets the first-order derivative of the loss function to search for the optimal values. Because RNN is recursive, the first-order derivation process will make a number smaller and smaller, then eventually vanish.
- This certain mathematic process makes RNN not a good choice to retain the past memories.

We need a recursive structure that the information does not vanish quickly.

LSTM


Forget Gate
Output Gate
h_{t-1}
h_t
h_{t+1}
A
A
σ
σ
tanh
σ
tanh
X_{t-1}
X_t
X_{t+1}
Input Gate

Long Short-Term Memory (LSTM)

- Uses the short-term memory processes to create longer memory
- Cell:
  - Remembers values over arbitrary time intervals
- Input gate, output and forget gates:
  - Regulate the flow of information into and out of the cell.
  - Sigmoid functions with output in [0,1] to pass limited information or all information. A value of zero means filter out the information completely, while a value of one means passing the information completely.

Forget Gate

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

- h_{t-1} : output of the memory cell at t-1
- It discover the details to be discarded from the block.
- It looks at the previous state h_{t-1} and the content input x_t and
- Use a sigmoid function to decide it.
- Output (for each number in the cell state C_{t-1}):
  - 0 (omit)
  - 1 (keep)

Input Gate

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

- It takes the previous output and current input and applies both a sigmoid and tanh activation.
- Sigmoid function
  - decides what must be kept from the input. 0 discard, 1 keep.
- $tanh$ function
  - normalizes the values into the range [-1, +1], stabilizing the training process.

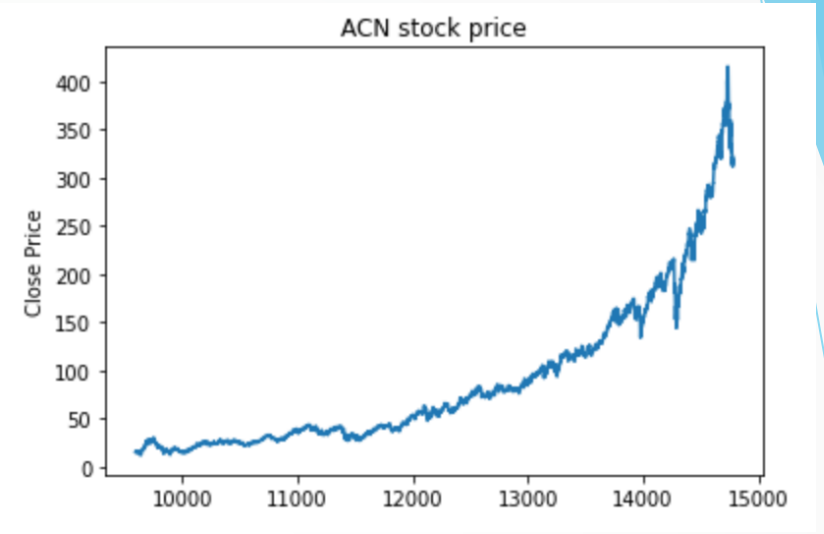
Output Gate

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * \tanh(C_t)$$

- It takes a normalized value for memory through tanh and a sigmoid activated value for the previous output and current input
- Decides what must be predicted for the current input value
- Output value, pass value and memory to the next cell.

Prepare Dataset



- Take ACN as an example
  - data shape: (5190, 8)
  - date period: 2001-07-19 to 2021-03-03
  - features: Open, High, Low, Close, Adjclose, Volume

	Date	open	high	low	close	adjclose	volume	ticker
9592	2001-07-19	15.10	15.29	15.00	15.17	11.223610	34994300.0	ACN
9593	2001-07-20	15.05	15.05	14.80	15.01	11.105230	9238500.0	ACN
9594	2001-07-23	15.00	15.01	14.55	15.00	11.097830	7501000.0	ACN
9595	2001-07-24	14.95	14.97	14.70	14.86	10.994254	3537300.0	ACN
9596	2001-07-25	14.70	14.95	14.65	14.95	11.060840	4208100.0	ACN

Stock Features

- Open:

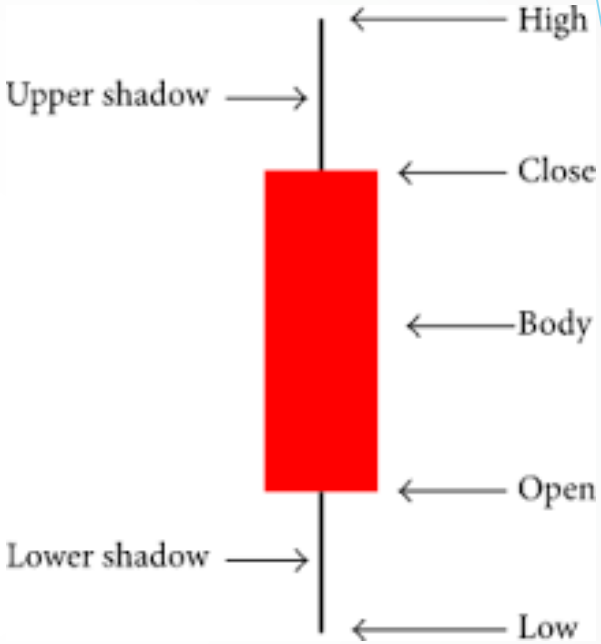
  The start price of a stock on a trading day
- Low / High:

  The lowest/highest price on a trading day
- Close:

  The end price of a stock on a trading day
- Adjust Close:

  Amends a stock's closing price to reflect that stock's real value
- Volume:

  Number of shares of a security traded between daily open and close



Prepare Dataset

- Input and Output:
  - Output: predicting Close aims to determine the future movement of the stock value of a financial exchange
  - Input: Different models require different input features and forms
- Train and Test:
  - Considering the amount of data, we divide dataset into a training set (90%) and a test set(10%)

Prepare Dataset: Normalization

	Date	open	high	low	close	adjclose	volume	ticker
9592	2001-07-19	15.10	15.29	15.00	15.17	11.223610	34994300.0	ACN
9593	2001-07-20	15.05	15.05	14.80	15.01	11.105230	9238500.0	ACN
9594	2001-07-23	15.00	15.01	14.55	15.00	11.097830	7501000.0	ACN
9595	2001-07-24	14.95	14.97	14.70	14.86	10.994254	3537300.0	ACN
9596	2001-07-25	14.70	14.95	14.65	14.95	11.060840	4208100.0	ACN

```
scaler = MinMaxScaler()
```

	open	high	low	close	adjclose	volume
0	0.009139	0.007479	0.009195	0.008227	0.006056	0.389084
1	0.009015	0.006887	0.008698	0.007830	0.005764	0.101229
2	0.008892	0.006788	0.008077	0.007805	0.005746	0.081810
3	0.008768	0.006690	0.008450	0.007458	0.005491	0.037510
4	0.008151	0.006640	0.008325	0.007681	0.005655	0.045007

Accuracy

- Root Mean Square Error (RMSE):

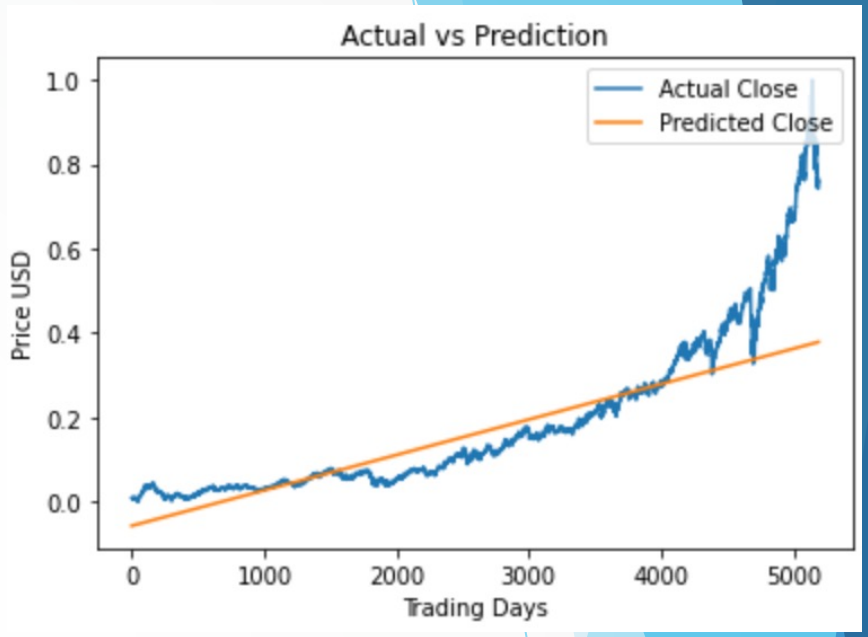
$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}}$$

- Mean Absolute Percentage Error (MAPE):

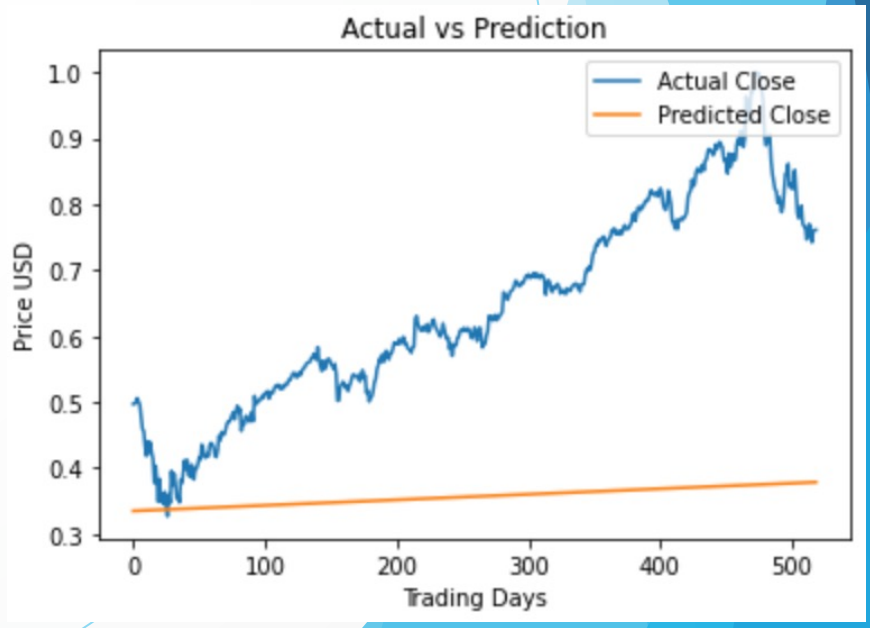
$$MAPE = \frac{100\%}{n} \sum_{i=1}^{n} \left| \frac{(\hat{y}_i - y_i)^2}{y_i} \right|$$

Linear Regression 1: Time Series

- Input Set:
  - Trading date
- Output Set:
  - Close price
- Pros and cons:
  - Easy to implement
  - Fail to capture nonlinear changes in price
  - Unable to reflect other features
- Accuracy:
  - RMSE:0.3180
  - MAPE:0.4124



train set and test set

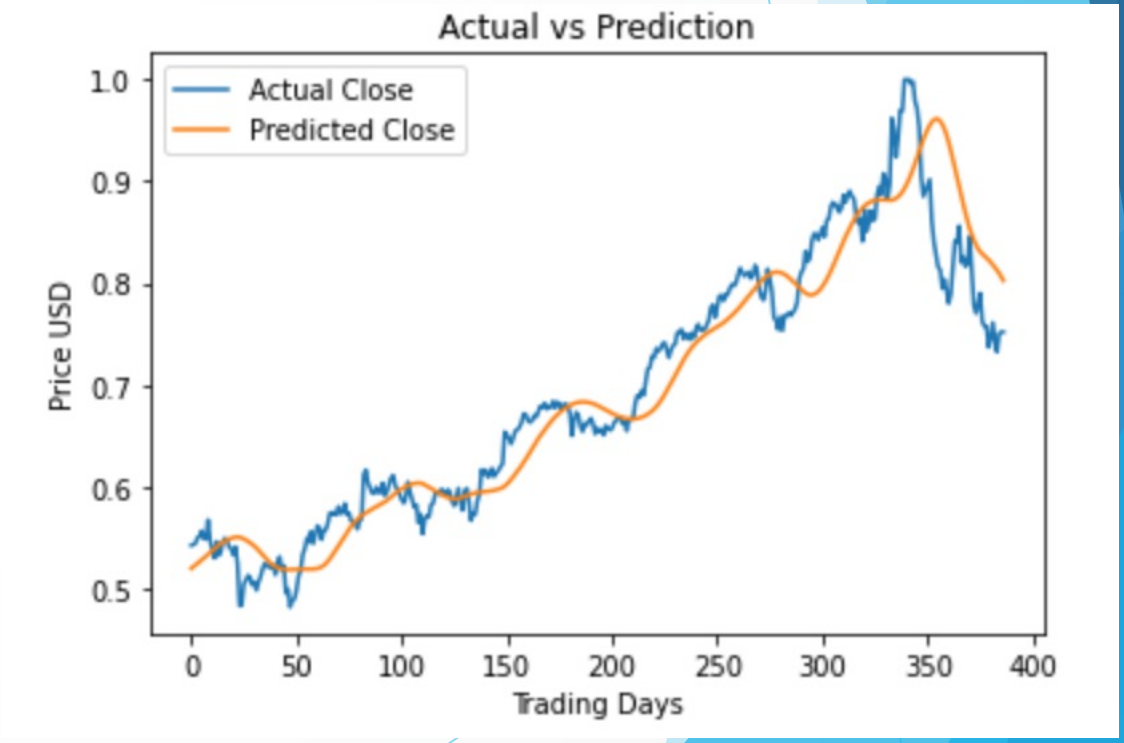


only test set

Linear Regression 2: SMA

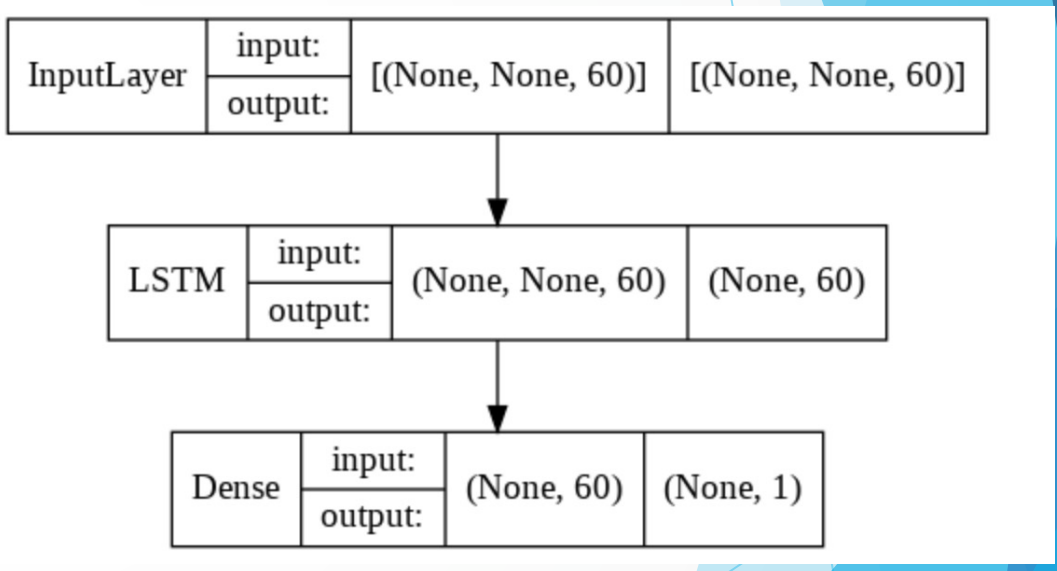
- Input Set:
  - Simple moving average(SMA) of Open from 30 days before the predicted day
- Output Set:
  - Close price
- Pros and cons:
  - Better fitness than before
  - Unable to reflect the impact of volume
  - Unable to fit mutations in time
- Accuracy:
  - RMSE: 0.0401
  - MASE: 0.0494

$$SMA = \frac{P1 + P2 + ... + Pn}{N}$$



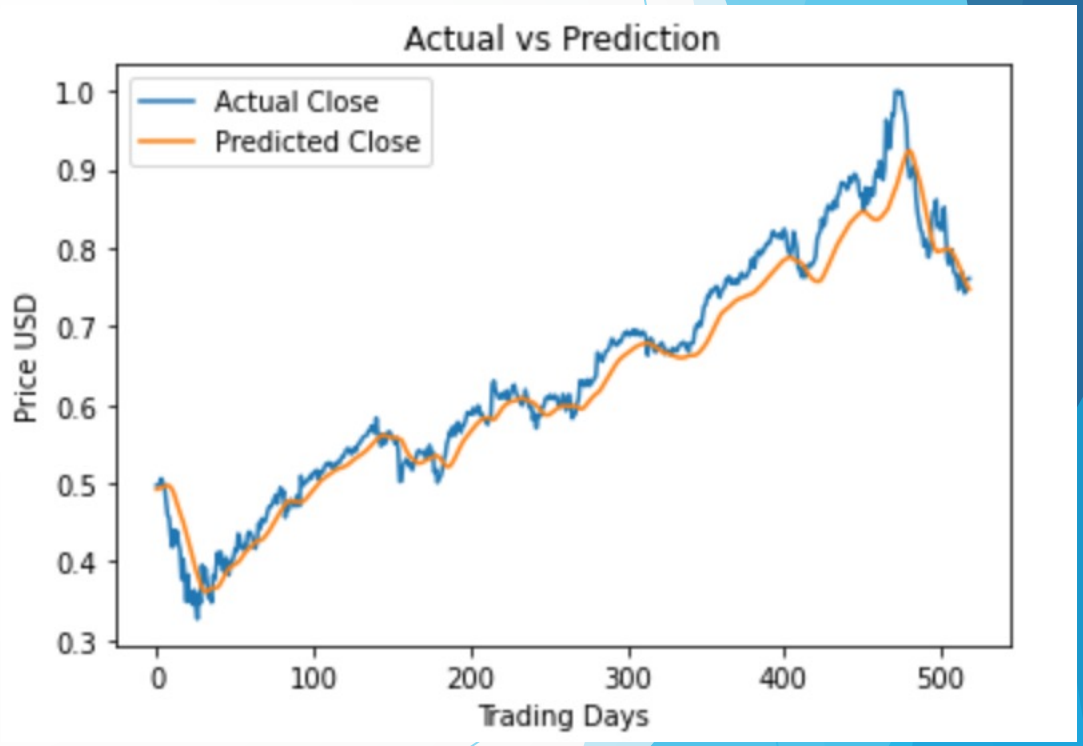
Basic LSTM Model

- Input:
  - Open: If we have higher Close than the Open that we have some profit otherwise we saw losses
  - Volume: indicates market strength, as rising markets on increasing volume viewed strong and vice versa
  - Data of first 60 days as an input unit
- Output:
  - Close of the predicted day



Basic LSTM Model

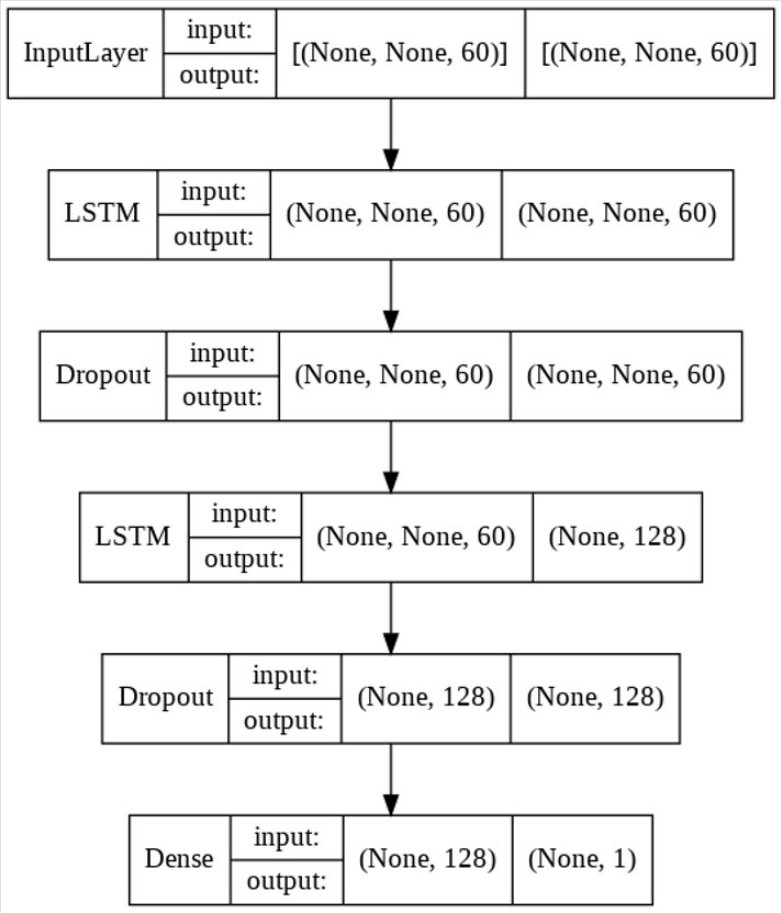
- Pros and Cons:
  - Model chronological sequences and their long-range dependencies precisely
  - Solve problem of vanishing gradients of RNN
  - Require more resources and time to get trained
  - Affected by random weight initialization
  - Prone to overfitting
- Accuracy:
  - RMSE:**0.0336**
  - MAPE:**0.0406**

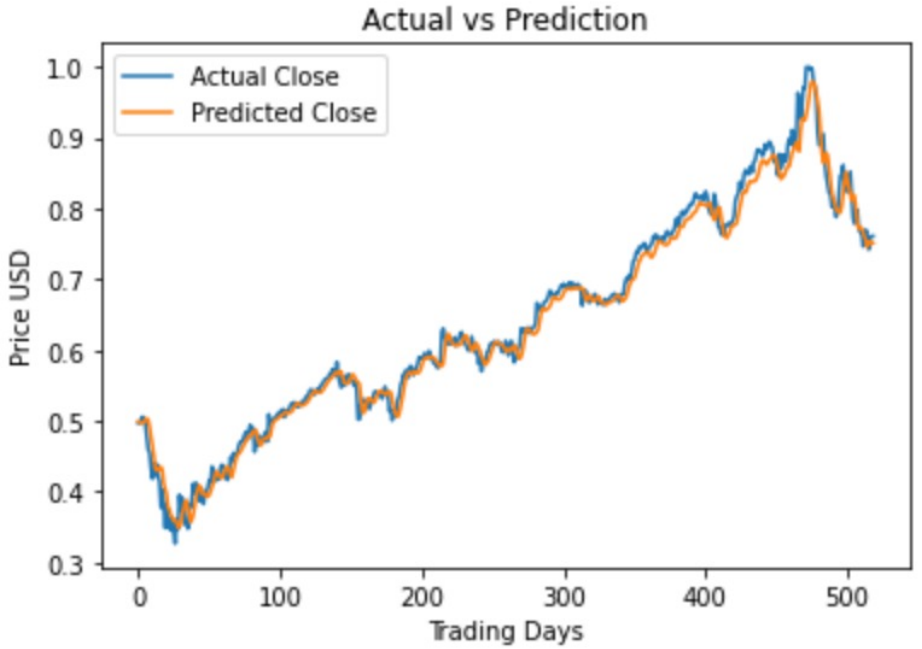


Some ways to improve LSTM Model

- Add hidden layer:
  - Adding stacked multi-layers is for extracting more abstract information.
  - In this case, we changed layers from 1 into 3
- Add Dropout:
  - A regular method to reduce overfitting and improving model performance
  - In this case, we added dropout of 0.2 at each layer of LSTM
- Increase epochs:
  - The times that the learning algorithm will work through the entire training dataset
  - In this case, we Increased epochs from 10 to 100

Improved LSTM model





- Input and output:
  - The same as basic model
- Accuracy:
  - RMSE:**0.0185**
  - MAPE:**0.0224**

Evaluating Prediction Performance for Stock Price Prediction

	Linear regression		LSTM	
Corporation	RMSE	MAPE	RMSE	MAPE
ACN	0.0401	0.0494	**0.0185**	**0.0224**
IBM	0.0351	0.0419	**0.0152**	**0.0174**
MSFT	**0.0285**	**0.0356**	0.0343	0.0416
QNC.V	0.0474	0.1724	**0.0260**	**0.0933**
INTC	0.0470	0.0501	**0.0214**	**0.0214**
BIDU	0.0648	0.0762	**0.0297**	**0.0381**
NOK	0.0069	0.1070	**0.0034**	**0.0438**
MIELY	0.0377	0.0420	**0.0181**	**0.0207**

Linear regression model here refers to linear regression with SMA
LSTM model refers to improved LSTM

Conclusion

- In this project, we implement two models to predict stock price: linear regression and LSTM
- According to our comparison, LSTM is more precise than linear regression in most cases
- Due to its complicated cell, LSTM requires far more resources and time to get trained
- We may try another commonly used method next: Gated Recurrent Unit(GRU) which has fewer training parameter and executes faster

Thank you!