Stock Price Prediction with LSTM

Prof. Kuan-Ting Lai

2021/4/17

Stock Trading System

Trading Strategy

ENTER

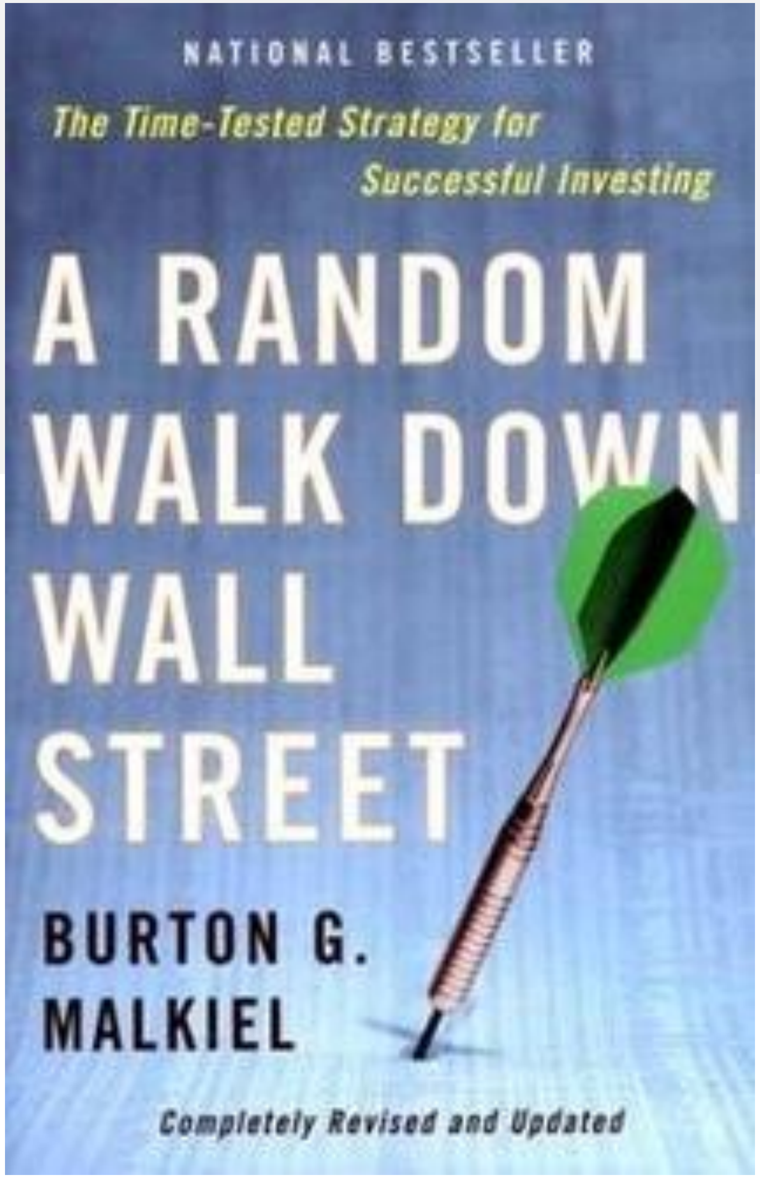
EXIT

STOP

MAX DRAW DOWN (MDD)

A Random Walk Down Wall Street

- A blindfolded monkey throwing darts at a newspaper's financial pages could select a portfolio that would do just as well as one carefully selected by experts.



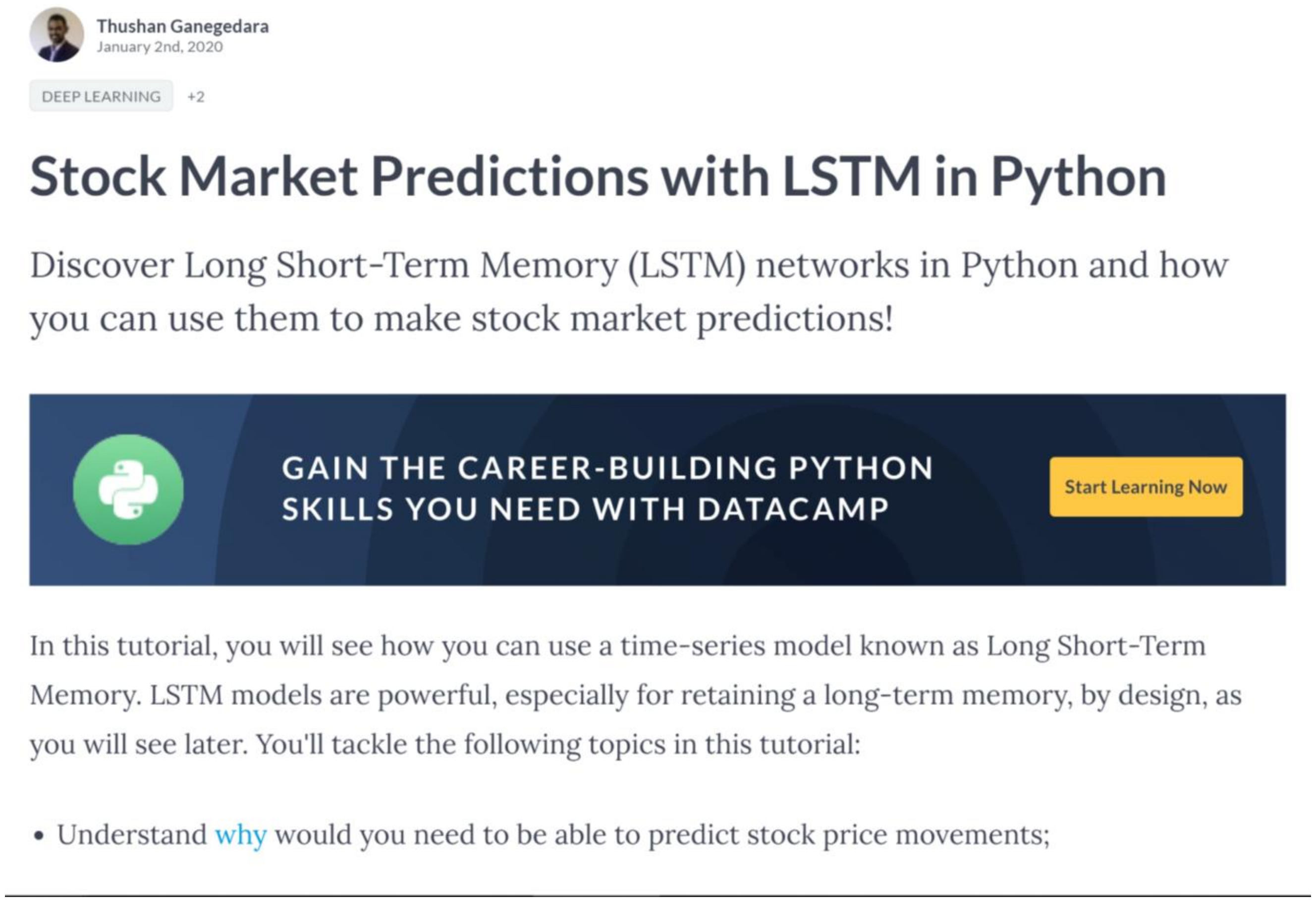
Cathie Wood

Flash Crash

https://en.wikipedia.org/wiki/2010_flash_crash

DataCamp Tutorial: Stock Market Predictions with LSTM in Python

- https://www.datacamp.com/community/tutorials/lstm-python-stock-market


Thushan Ganegedara
January 2nd, 2020

DEEP LEARNING +2

Stock Market Predictions with LSTM in Python

Discover Long Short-Term Memory (LSTM) networks in Python and how you can use them to make stock market predictions!

In this tutorial, you will see how you can use a time-series model known as Long Short-Term Memory. LSTM models are powerful, especially for retaining a long-term memory, by design, as you will see later. You'll tackle the following topics in this tutorial:

- Understand why would you need to be able to predict stock price movements;

Getting Your Data

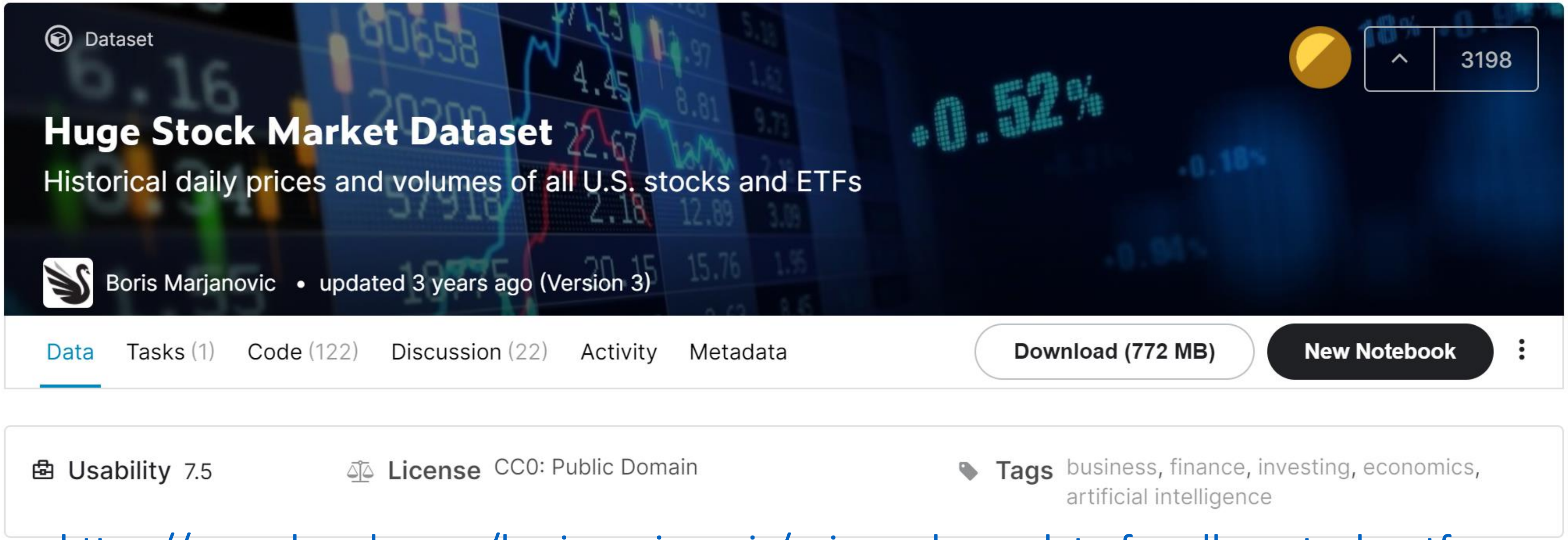
- Alpha Vantage Stock API
  - Obtain for free API Key here.

- Kaggle: Huge Stock Market Dataset

- 永豐金證券

- XQ全球贏家

- Multicharts https://www.multicharts.com/


Dataset

3198

Huge Stock Market Dataset

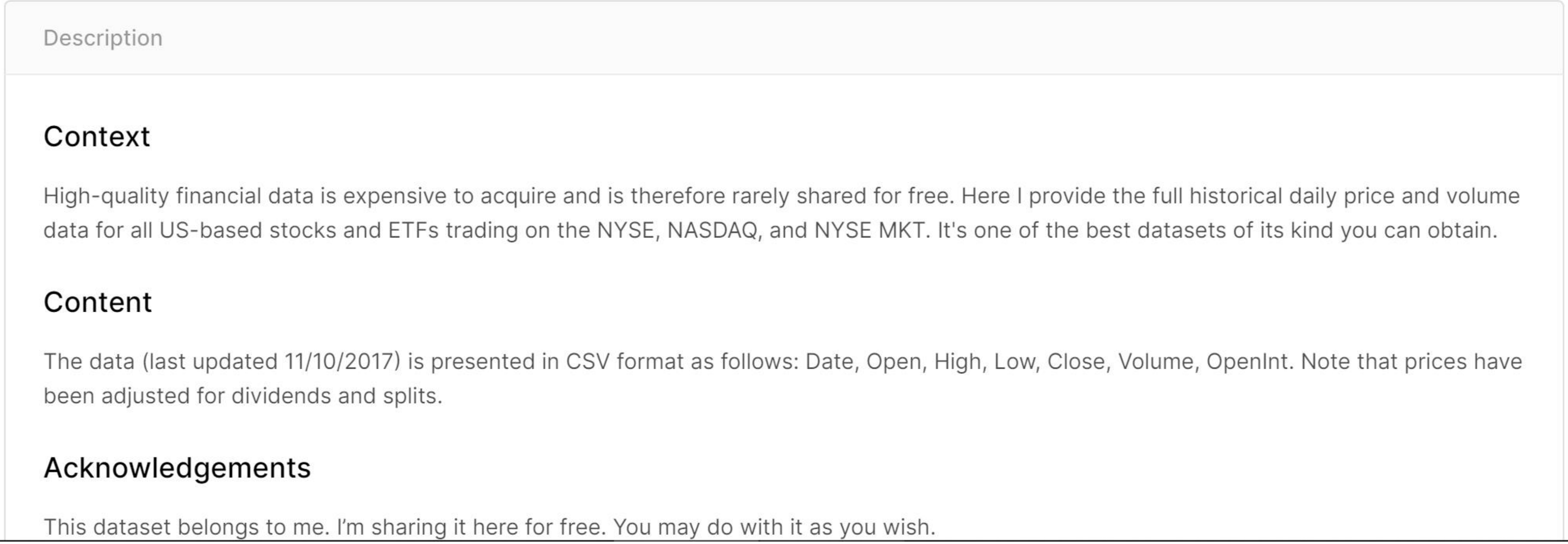
Historical daily prices and volumes of all U.S. stocks and ETFs

Boris Marjanovic • updated 3 years ago (Version 3)

Data Tasks (1) Code (122) Discussion (22) Activity Metadata Download (772 MB) New Notebook

Usability 7.5 License CC0: Public Domain Tags business, finance, investing, economics, artificial intelligence

- https://www.kaggle.com/borismarjanovic/price-volume-data-for-all-us-stocks-etfs


Description

Context

High-quality financial data is expensive to acquire and is therefore rarely shared for free. Here I provide the full historical daily price and volume data for all US-based stocks and ETFs trading on the NYSE, NASDAQ, and NYSE MKT. It's one of the best datasets of its kind you can obtain.

Content

The data (last updated 11/10/2017) is presented in CSV format as follows: Date, Open, High, Low, Close, Volume, OpenInt. Note that prices have been adjusted for dividends and splits.

Acknowledgements

This dataset belongs to me. I'm sharing it here for free. You may do with it as you wish.

Stock Price Prediction with LSTM in TensorFlow 1.x

- https://colab.research.google.com/drive/14XaxRKS-5c5awkkNlfnLsvHCBpRMevdd?usp=sharing

Print Your Data

stock_predict_LSTM_tf.ipynb

檔案 編輯 檢視畫面 插入 執行階段 工具 說明 儲存中...

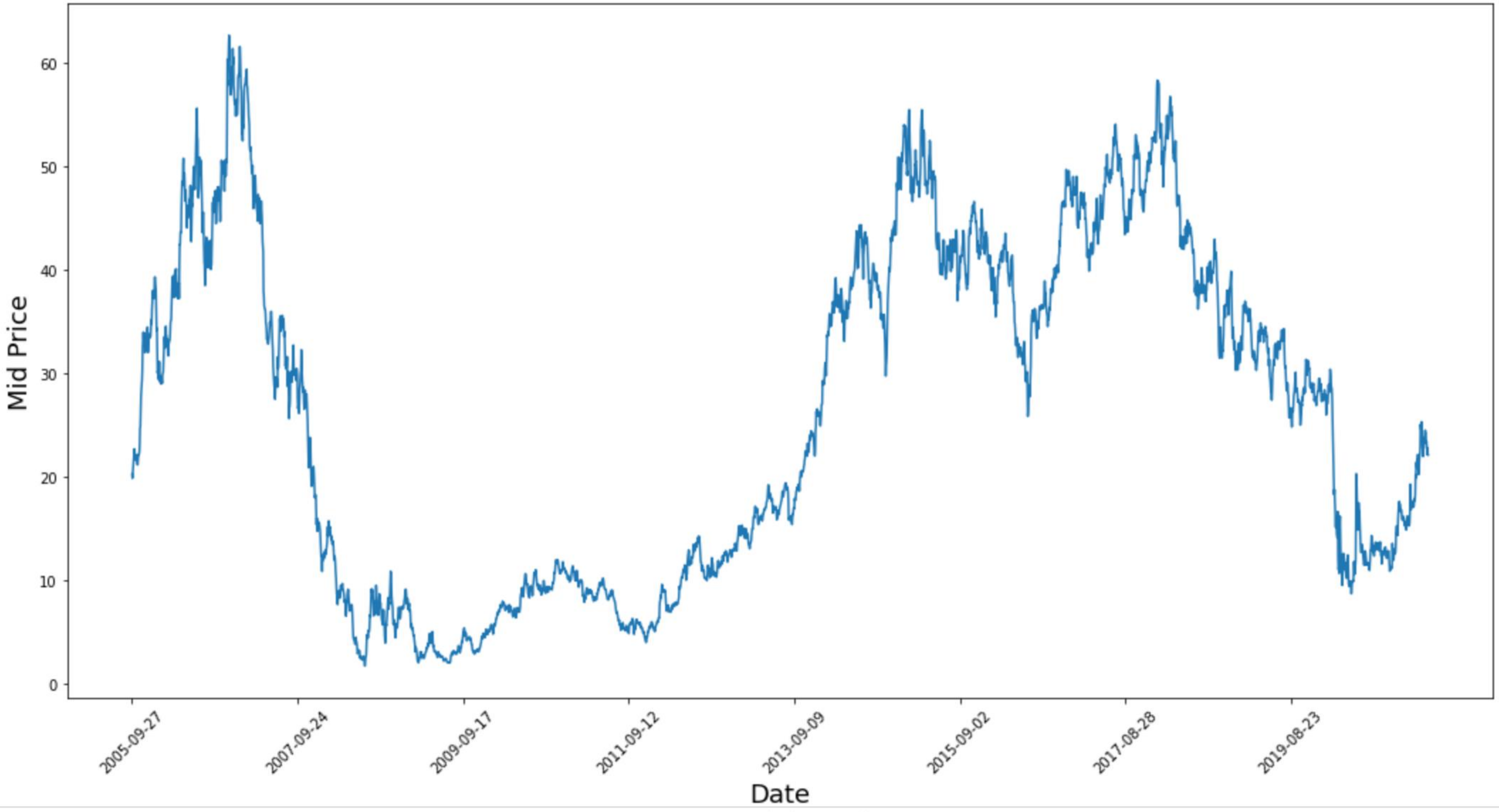
+ 程式碼 + 文字

[3]
```
# Sort DataFrame by date
df = df.sort_values('Date')
df.head()
```

	Unnamed: 0	Date	Low	High	Close	Open
3914	0	2005-09-27	19.10	21.40	19.30	21.05
3913	1	2005-09-28	19.20	20.53	20.50	19.30
3912	2	2005-09-29	20.10	20.58	20.21	20.40
3911	3	2005-09-30	20.18	21.05	21.01	20.26
3910	4	2005-10-03	20.90	21.75	21.50	20.90

Data Visualization

```
[4]  plt.figure(figsize = (18,9))
     plt.plot(range(df.shape[0]),(df['Low']+df['High'])/2.0)
     plt.xticks(range(0,df.shape[0],500),df['Date'].loc[::500],rotation=45)
     plt.xlabel('Date',fontsize=18)
     plt.ylabel('Mid Price',fontsize=18)
     plt.show()
```



Data Normalization

- MinMaxScaler

```
# Scale the data to be between 0 and 1
# When scaling remember! You normalize both test and train data with respect to training data
# Because you are not supposed to have access to test data
scaler = MinMaxScaler()
train_data = train_data.reshape(-1,1)
test_data = test_data.reshape(-1,1)
```

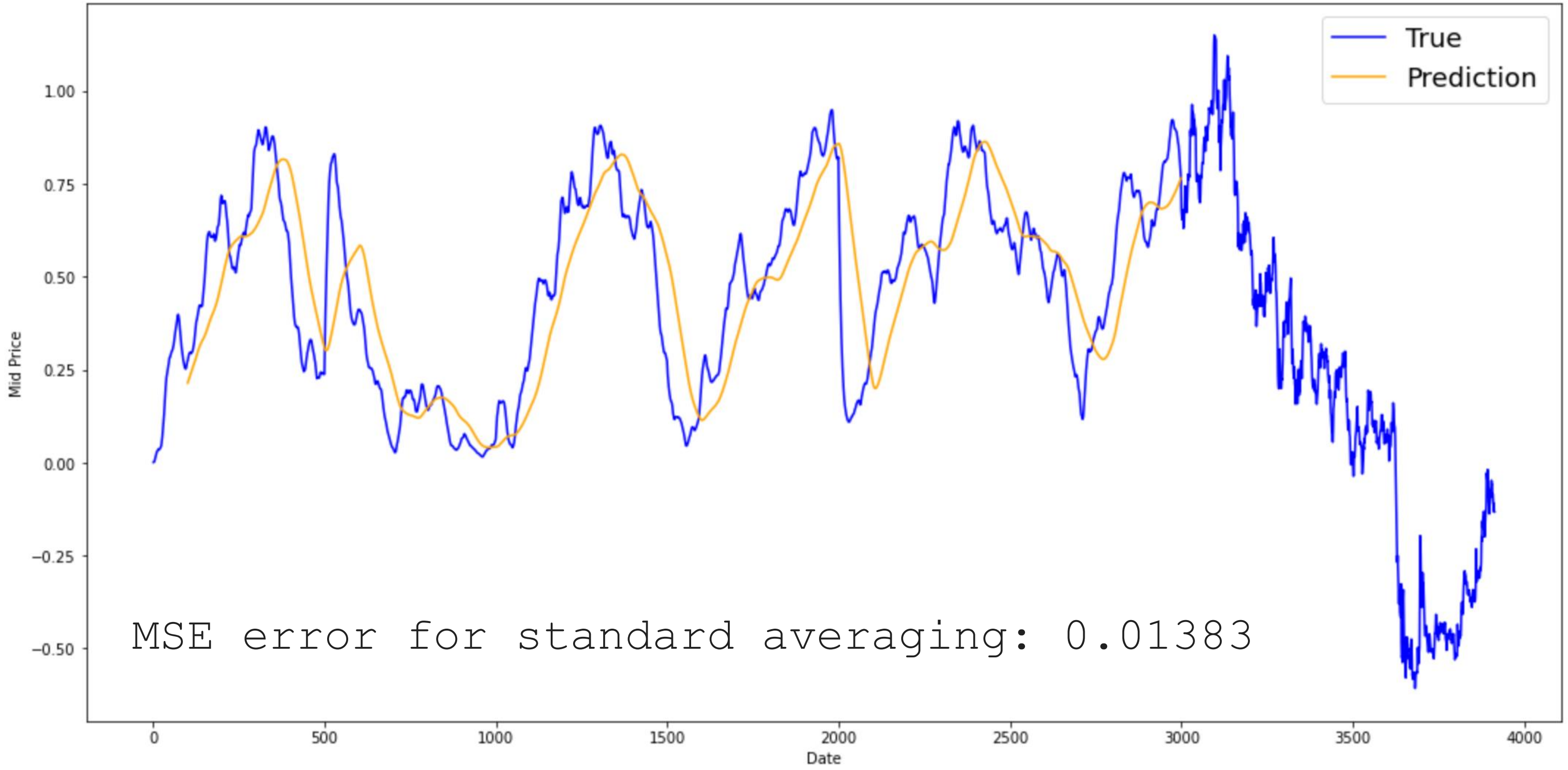
```
[ ] train_data.shape
```

```
(3000, 1)
```

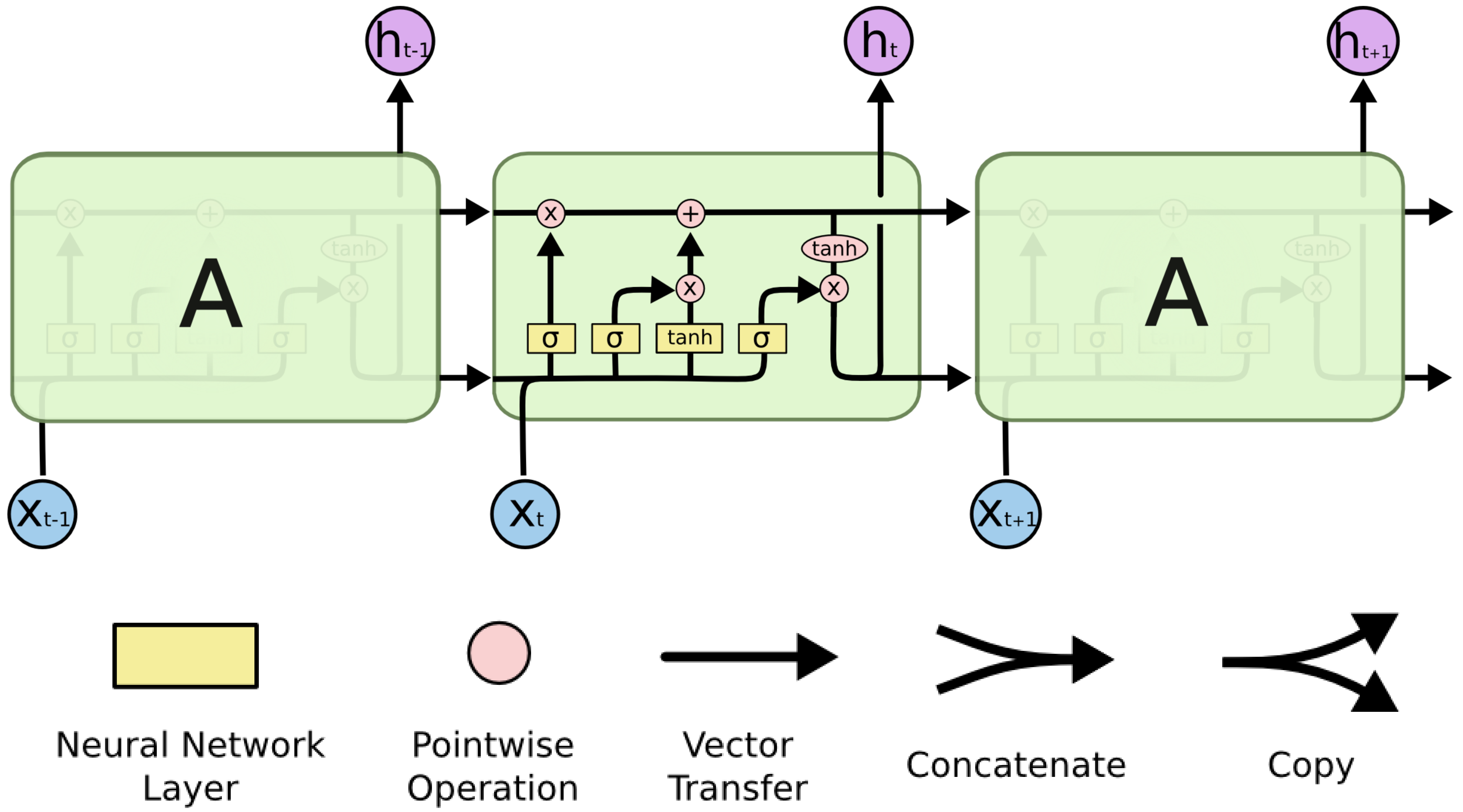
```
[ ] # Train the Scaler with training data and smooth data
smoothing_window_size = 500
for di in range(0,NUM_TRAIN_DATA,smoothing_window_size):
    scaler.fit(train_data[di:di+smoothing_window_size,:])
    train_data[di:di+smoothing_window_size,:] = scaler.transform(train_data[di:di+smoothing_window_size,:])
```

Prediction using Standard Average

```
plt.figure(figsize = (18,9))
plt.plot(range(df.shape[0]),all_mid_data,color='b',label='True')
plt.plot(range(window_size,N),std_avg_predictions,color='orange',label='Prediction')
#plt.xticks(range(0,df.shape[0],50),df['Date'].loc[::50],rotation=45)
plt.xlabel('Date')
plt.ylabel('Mid Price')
plt.legend(fontsize=18)
plt.show()
```



Long Short-Term Memory (LSTM)



Implement LSTM using TensorFlow

```
[ ] D = 1 # Dimensionality of the data. Since your data is 1-D this would be 1
    num_unrollings = 10 # Number of time steps you look into the future.
    batch_size = 500 # Number of samples in a batch
    num_nodes = [200,200,150] # Number of hidden nodes in each layer of the deep LSTM stack we're using
    n_layers = len(num_nodes) # number of layers
    dropout = 0.2 # dropout amount
```

```
[ ] tf.reset_default_graph() # This is important in case you run this multiple times
```

```
[ ] # Input data.
    train_inputs, train_outputs = [],[]

    # You unroll the input over time defining placeholders for each time step
    for ui in range(num_unrollings):
            train_inputs.append(tf.placeholder(tf.float32, shape=[batch_size,D],name='train_inputs_%d'%ui))
            train_outputs.append(tf.placeholder(tf.float32, shape=[batch_size,1], name = 'train_outputs_%d'%ui))
```

```
[ ] lstm_cells = [
            tf.contrib.rnn.LSTMCell(num_units=num_nodes[li],
                                    state_is_tuple=True,
                                    initializer= tf.contrib.layers.xavier_initializer())
      for li in range(n_layers)]

    drop_lstm_cells = [tf.contrib.rnn.DropoutWrapper(
            lstm, input_keep_prob=1.0,output_keep_prob=1.0-dropout, state_keep_prob=1.0-dropout
    ) for lstm in lstm_cells]
    drop_multi_cell = tf.contrib.rnn.MultiRNNCell(drop_lstm_cells)
    multi_cell = tf.contrib.rnn.MultiRNNCell(lstm_cells)

    w = tf.get_variable('w',shape=[num_nodes[-1], 1], initializer=tf.contrib.layers.xavier_initializer())
    b = tf.get_variable('b',initializer=tf.random_uniform([1],-0.1,0.1))
```

Prediction Results

- MSE is around 0.04