# Snapshot of the algorithm and data collected for the RNN-LSTM with attention

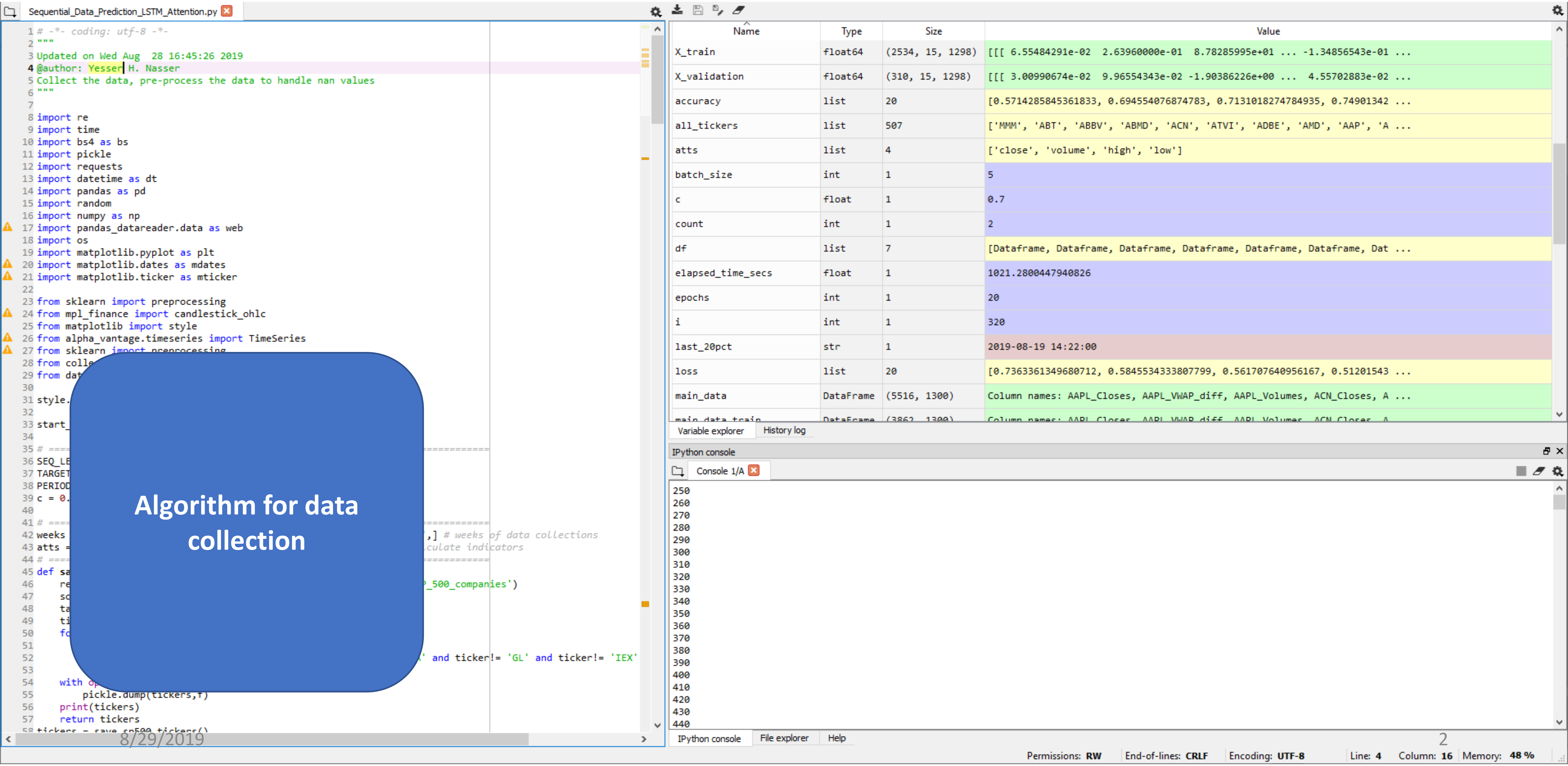
Ongoing Application: High Frequency Quant Trading
strategies using machine learning.

Author: Yesser Nasser, PhD

Quantitative Data Scientist – Machine Learning Engineer

Yesser.Nasser@icloud.com

# Algorithm Part 1: data collection



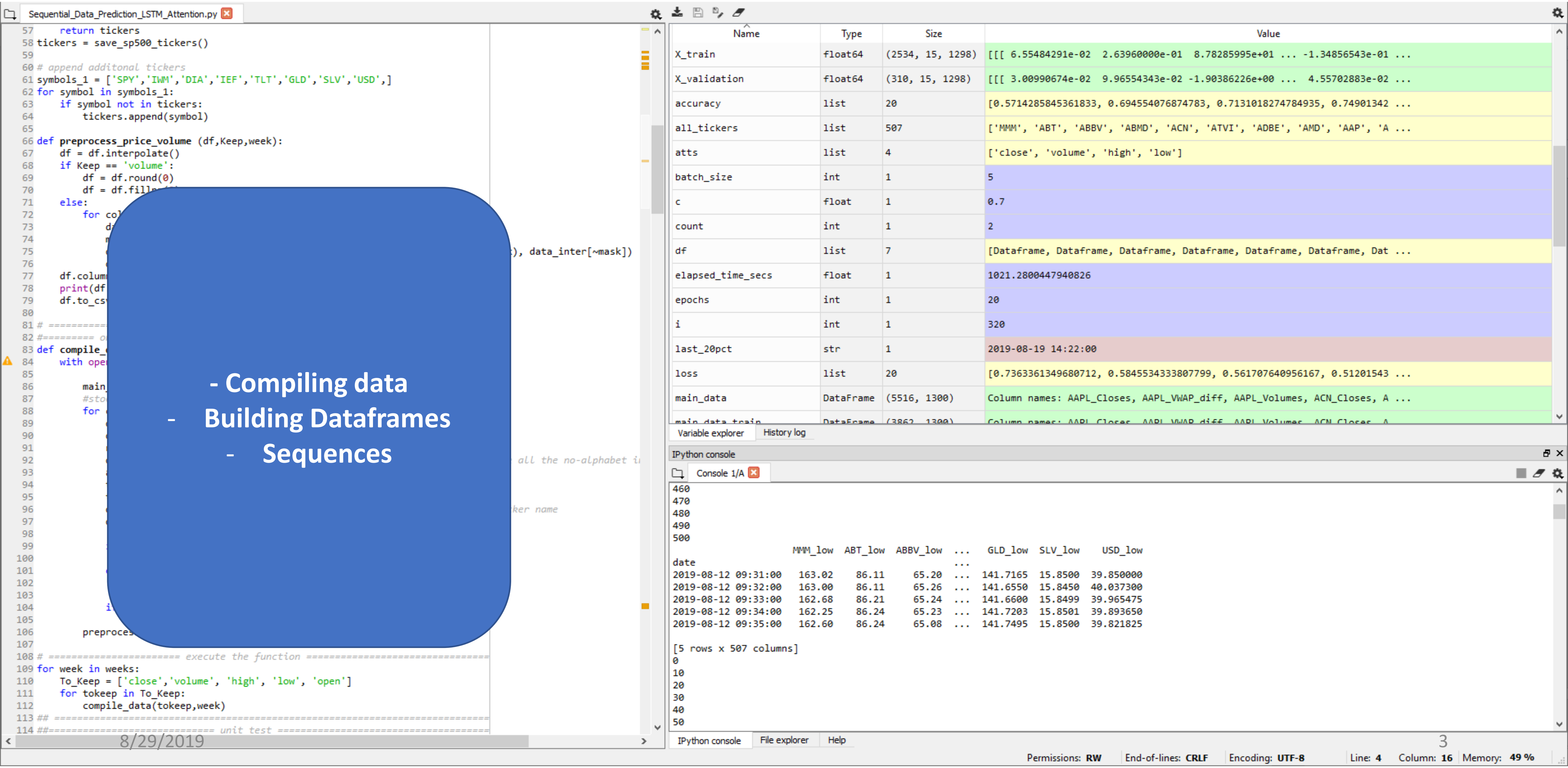
```python
# -*- coding: utf-8 -*-
"""
Updated on Wed Aug  28 16:45:26 2019
@author: Yesser H. Nasser
Collect the data, pre-process the data to handle nan values
"""

import re
import time
import bs4 as bs
import pickle
import requests
import datetime as dt
import pandas as pd
import random
import numpy as np
import pandas_datareader.data as web
import os
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
import matplotlib.ticker as mticker

from sklearn import preprocessing
from mpl_finance import candlestick_ohlc
from matplotlib import style
from alpha_vantage.timeseries import TimeSeries
from sklearn import preprocessing
from colle
from dat

style.

start_

# ====                                             ============
SEQ_LE
TARGET
PERIOD
c = 0.

# ====                                             ============
weeks                                               ,] # weeks of data collections
atts =                                               culate indicators
# ====                                             ============
def sa
    re                                              _500_companies')
    sc
    ta
    ti
    fo

                                                   ' and ticker!= 'GL' and ticker!= 'IEX'

    with op
        pickle.dump(tickers,f)
    print(tickers)
    return tickers
tickers = save_sp500_tickers()
```

Algorithm for data collection

| Name | Type | Size | Value |
|---|---|---|---|
| X_train | float64 | (2534, 15, 1298) | [[[ 6.55484291e-02  2.63960000e-01  8.78285995e+01 ... -1.34856543e-01 ... |
| X_validation | float64 | (310, 15, 1298) | [[[ 3.00990674e-02  9.96554343e-02 -1.90386226e+00 ...  4.55702883e-02 ... |
| accuracy | list | 20 | [0.5714285845361833, 0.694554076874783, 0.7131018274784935, 0.74901342 ... |
| all_tickers | list | 507 | ['MMM', 'ABT', 'ABBV', 'ABMD', 'ACN', 'ATVI', 'ADBE', 'AMD', 'AAP', 'A ... |
| atts | list | 4 | ['close', 'volume', 'high', 'low'] |
| batch_size | int | 1 | 5 |
| c | float | 1 | 0.7 |
| count | int | 1 | 2 |
| df | list | 7 | [Dataframe, Dataframe, Dataframe, Dataframe, Dataframe, Dataframe, Dat ... |
| elapsed_time_secs | float | 1 | 1021.2800447940826 |
| epochs | int | 1 | 20 |
| i | int | 1 | 320 |
| last_20pct | str | 1 | 2019-08-19 14:22:00 |
| loss | list | 20 | [0.7363361349680712, 0.5845534333807799, 0.561707640956167, 0.51201543 ... |
| main_data | DataFrame | (5516, 1300) | Column names: AAPL_Closes, AAPL_VWAP_diff, AAPL_Volumes, ACN_Closes, A ... |
| main_data_train | DataFrame | (3862, 1300) | Column names: AAPL_Closes, AAPL_VWAP_diff, AAPL_Volumes, ACN_Closes, A |

```
250
260
270
280
290
300
310
320
330
340
350
360
370
380
390
400
410
420
430
440
```

Permissions: RW   End-of-lines: CRLF   Encoding: UTF-8   Line: 4   Column: 16   Memory: 48 %



```
    return tickers
tickers = save_sp500_tickers()

# append additonal tickers
symbols_1 = ['SPY','IWM','DIA','IEF','TLT','GLD','SLV','USD',]
for symbol in symbols_1:
    if symbol not in tickers:
        tickers.append(symbol)

def preprocess_price_volume (df,Keep,week):
    df = df.interpolate()
    if Keep == 'volume':
        df = df.round(0)
        df = df.fill
    else:
        for co
                                                  )), data_inter[~mask])

    df.colum
    print(df
    df.to_cs

# =========
#========= o
def compile_
    with ope

        main_
        #sto
        for
                                                  all the no-alphabet i

                                                  ker name

            i

        preproce

# ====================== execute the function ==============================
for week in weeks:
    To_Keep = ['close','volume', 'high', 'low', 'open']
    for tokeep in To_Keep:
        compile_data(tokeep,week)
## =========================================================================
##========================== unit test =====================================
```

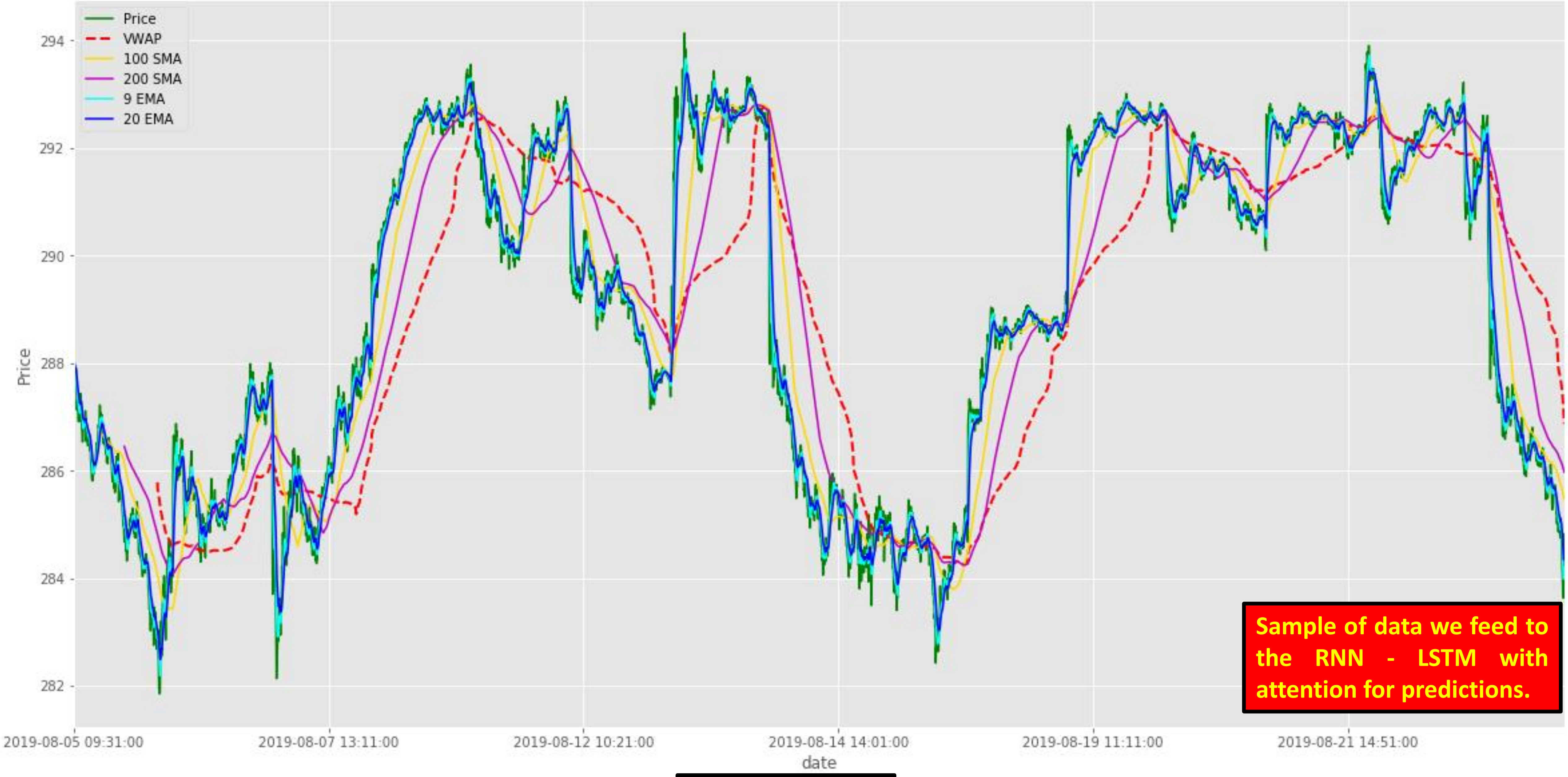
| Name | Type | Size | Value |
|---|---|---|---|
| X_train | float64 | (2534, 15, 1298) | [[[ 6.55484291e-02  2.63960000e-01  8.78285995e+01 ... -1.34856543e-01 ... |
| X_validation | float64 | (310, 15, 1298) | [[[ 3.00990674e-02  9.96554343e-02 -1.90386226e+00 ...  4.55702883e-02 ... |
| accuracy | list | 20 | [0.5714285845361833, 0.694554076874783, 0.7131018274784935, 0.74901342 ... |
| all_tickers | list | 507 | ['MMM', 'ABT', 'ABBV', 'ABMD', 'ACN', 'ATVI', 'ADBE', 'AMD', 'AAP', 'A ... |
| atts | list | 4 | ['close', 'volume', 'high', 'low'] |
| batch_size | int | 1 | 5 |
| c | float | 1 | 0.7 |
| count | int | 1 | 2 |
| df | list | 7 | [Dataframe, Dataframe, Dataframe, Dataframe, Dataframe, Dataframe, Dat ... |
| elapsed_time_secs | float | 1 | 1021.2800447940826 |
| epochs | int | 1 | 20 |
| i | int | 1 | 320 |
| last_20pct | str | 1 | 2019-08-19 14:22:00 |
| loss | list | 20 | [0.7363361349680712, 0.5845534333807799, 0.561707640956167, 0.51201543 ... |
| main_data | DataFrame | (5516, 1300) | Column names: AAPL_Closes, AAPL_VWAP_diff, AAPL_Volumes, ACN_Closes, A ... |
| main_data_train | DataFrame | (3862, 1300) | Column names: AAPL_Closes, AAPL_VWAP_diff, AAPL_Volumes, ACN_Closes, A |

```
460
470
480
490
500
                     MMM_low  ABT_low  ABBV_low  ...   GLD_low  SLV_low    USD_low
date                                             ...
2019-08-12 09:31:00   163.02    86.11     65.20  ...  141.7165  15.8500  39.850000
2019-08-12 09:32:00   163.00    86.11     65.26  ...  141.6550  15.8450  40.037300
2019-08-12 09:33:00   162.68    86.21     65.24  ...  141.6600  15.8499  39.965475
2019-08-12 09:34:00   162.25    86.24     65.23  ...  141.7203  15.8501  39.893650
2019-08-12 09:35:00   162.60    86.24     65.08  ...  141.7495  15.8500  39.821825

[5 rows x 507 columns]
0
10
20
30
40
50
```

Permissions: RW   End-of-lines: CRLF   Encoding: UTF-8   Line: 4   Column: 16   Memory: 49 %

- Compiling data
- Building Dataframes
- Sequences

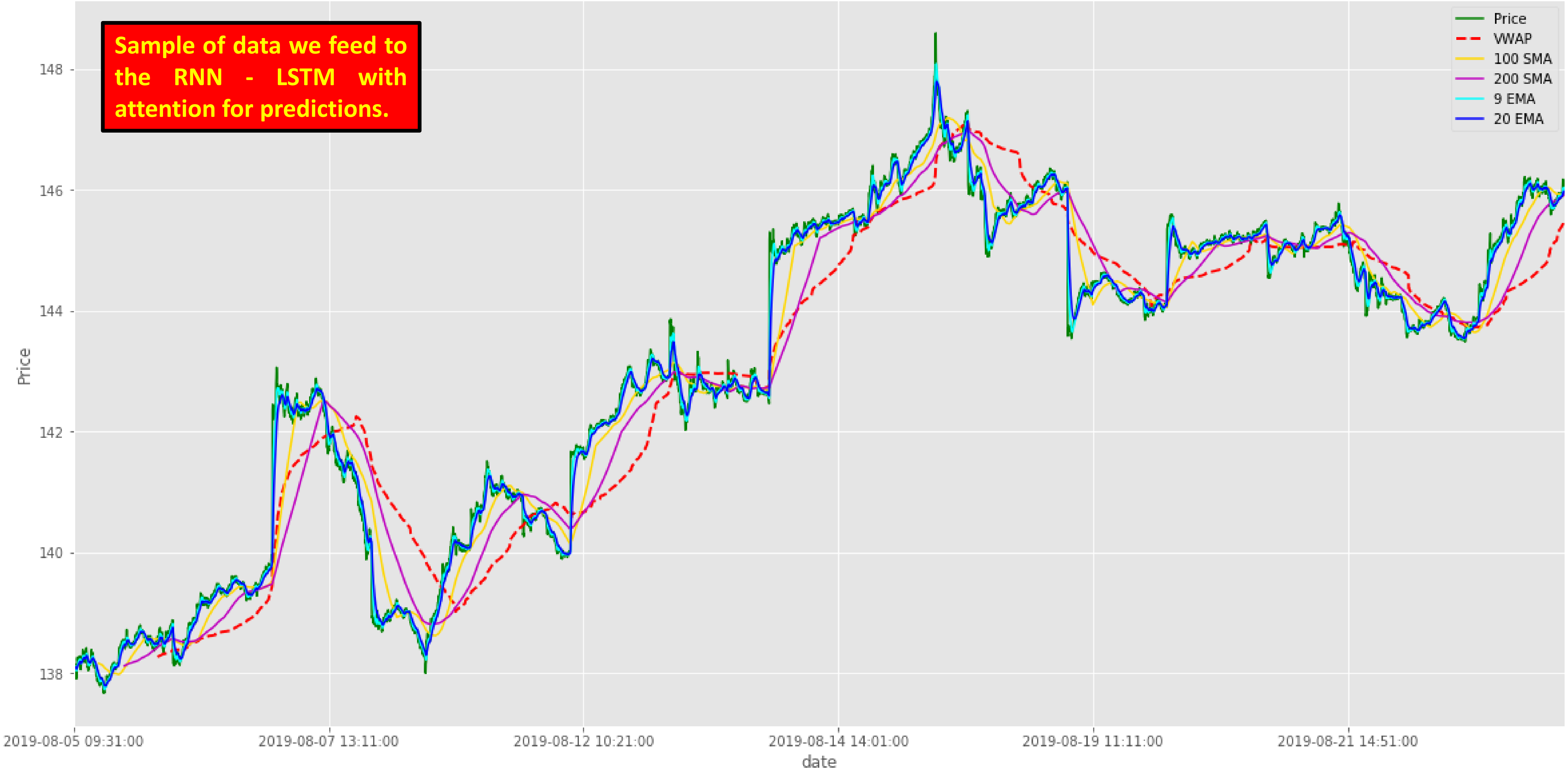
# SPY



Sample of data we feed to the RNN - LSTM with attention for predictions.

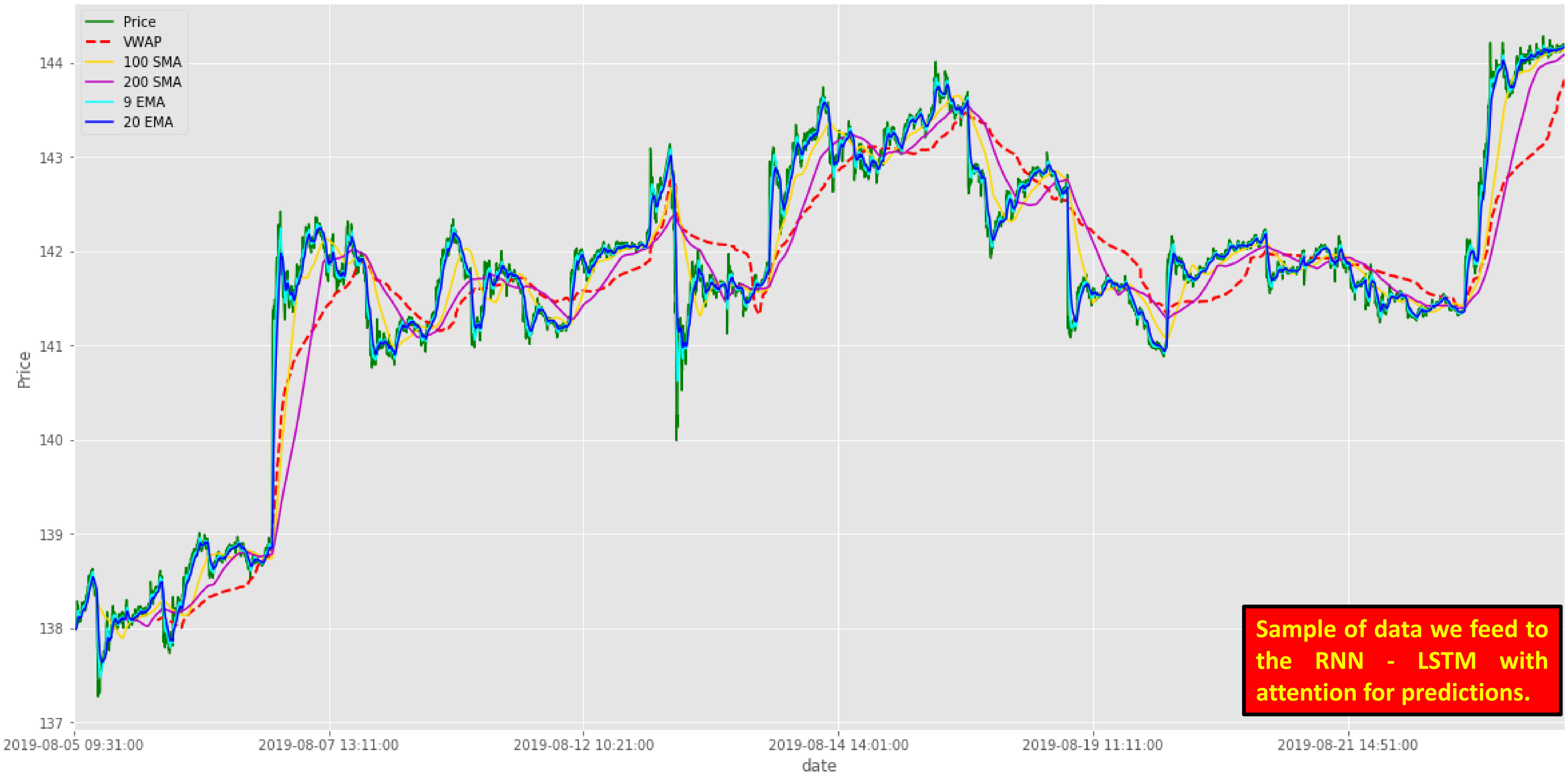
Very resent data!

# TLT



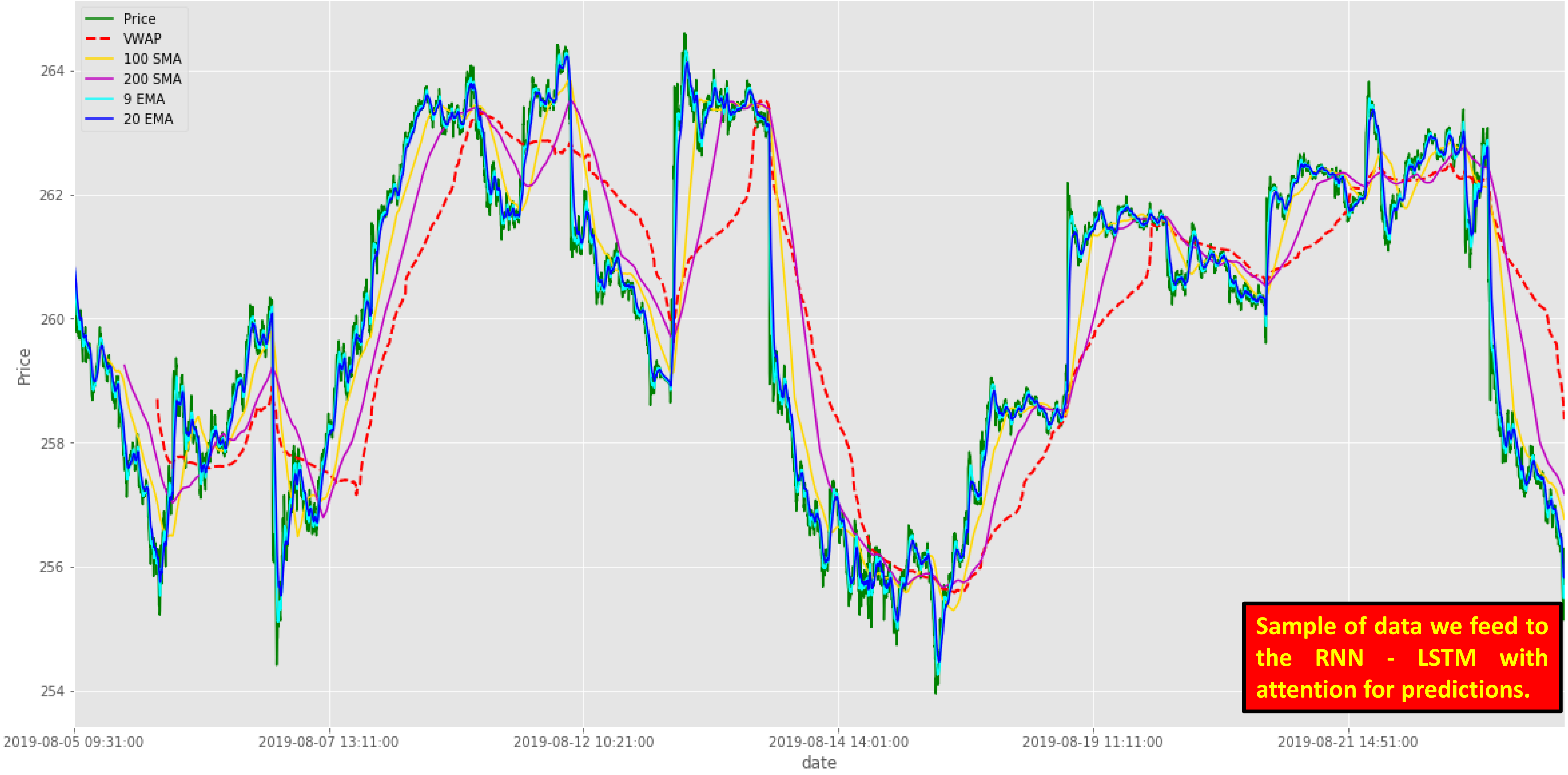
**Sample of data we feed to the RNN - LSTM with attention for predictions.**

# GLD



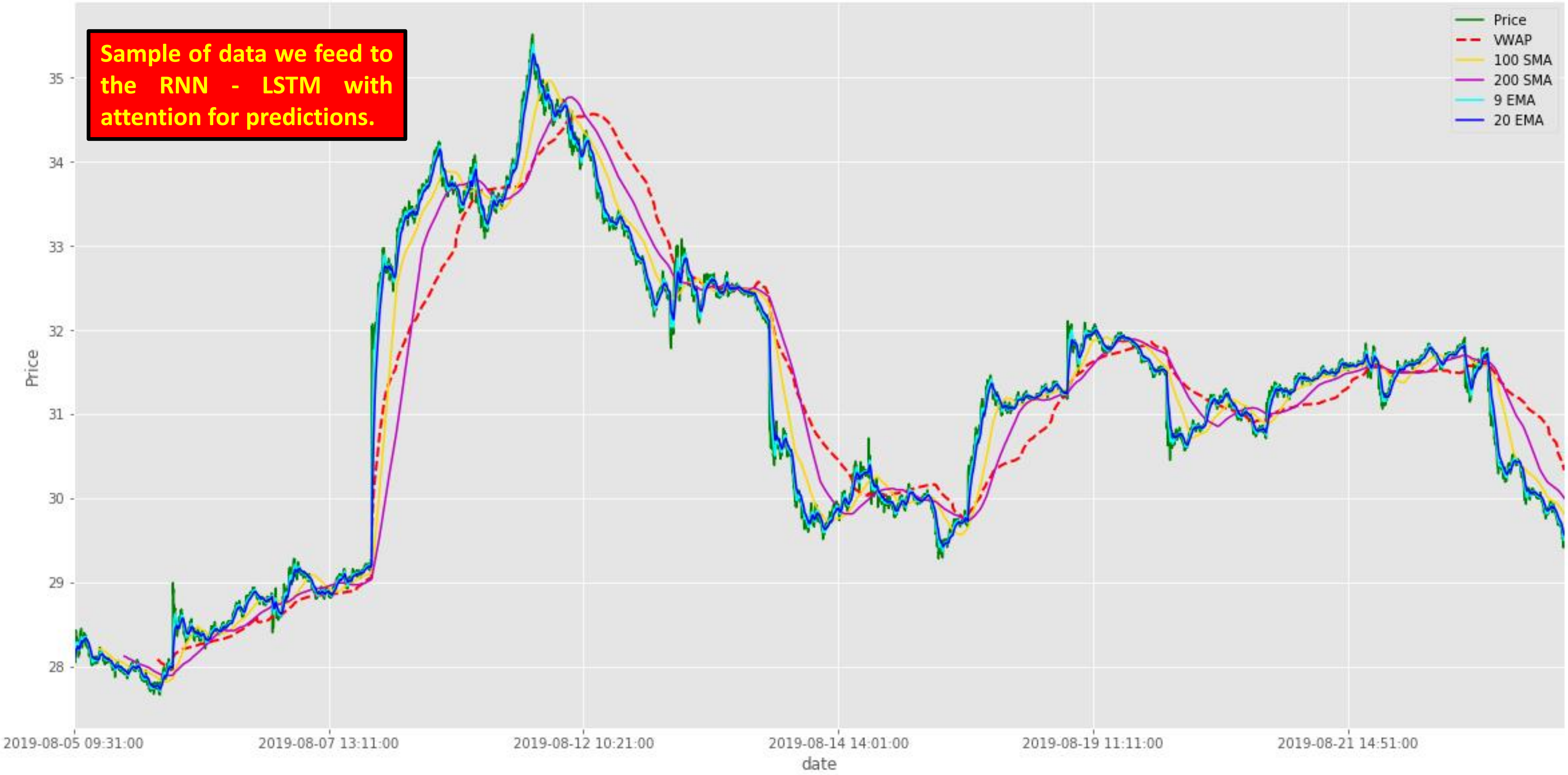
Sample of data we feed to the RNN - LSTM with attention for predictions.

# DIA



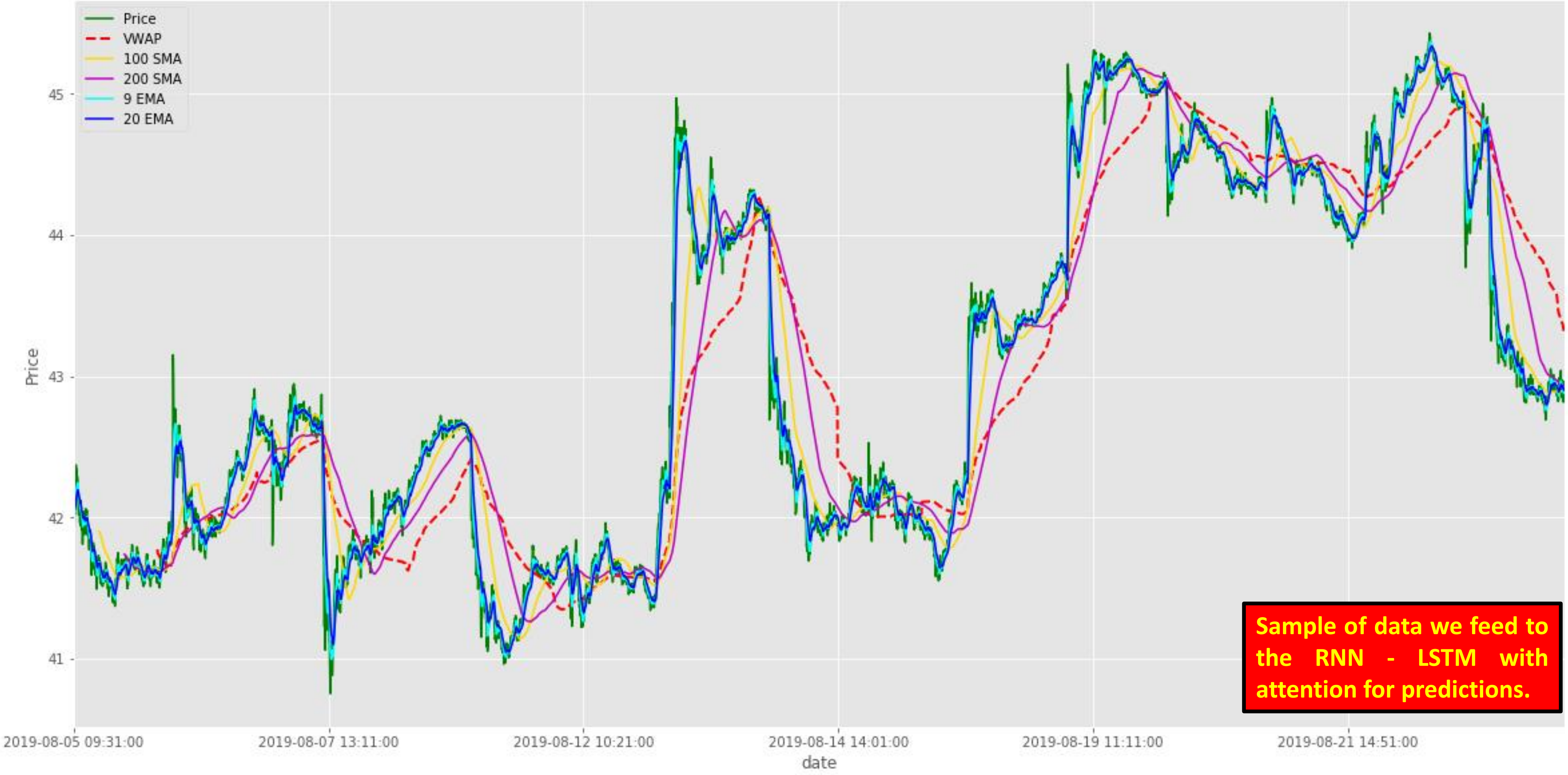
**Sample of data we feed to the RNN - LSTM with attention for predictions.**

# Stock Price - AMD



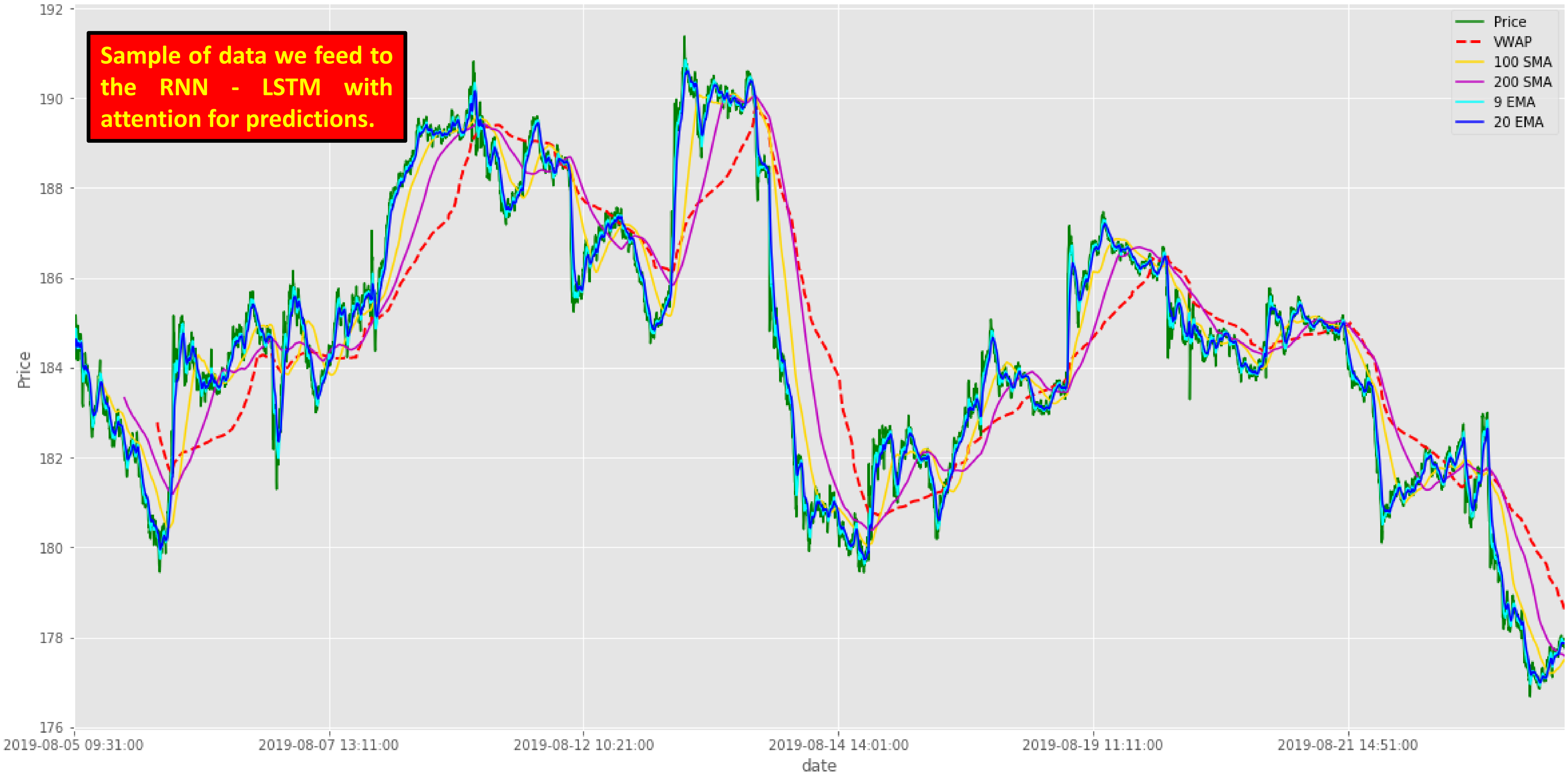
Sample of data we feed to the RNN - LSTM with attention for predictions.

# Stock Price - MU



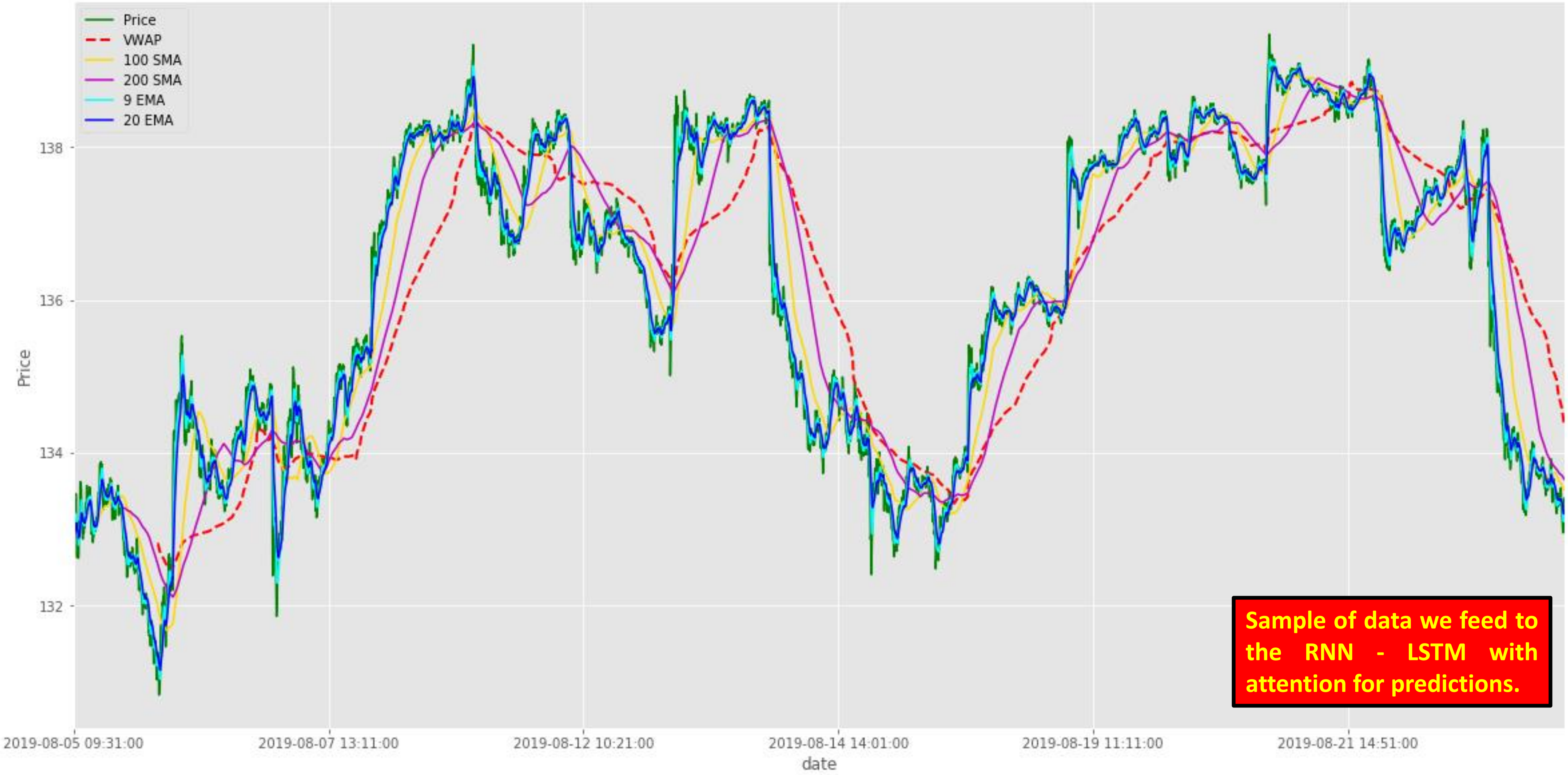
Sample of data we feed to the RNN - LSTM with attention for predictions.

# Stock Price - FB



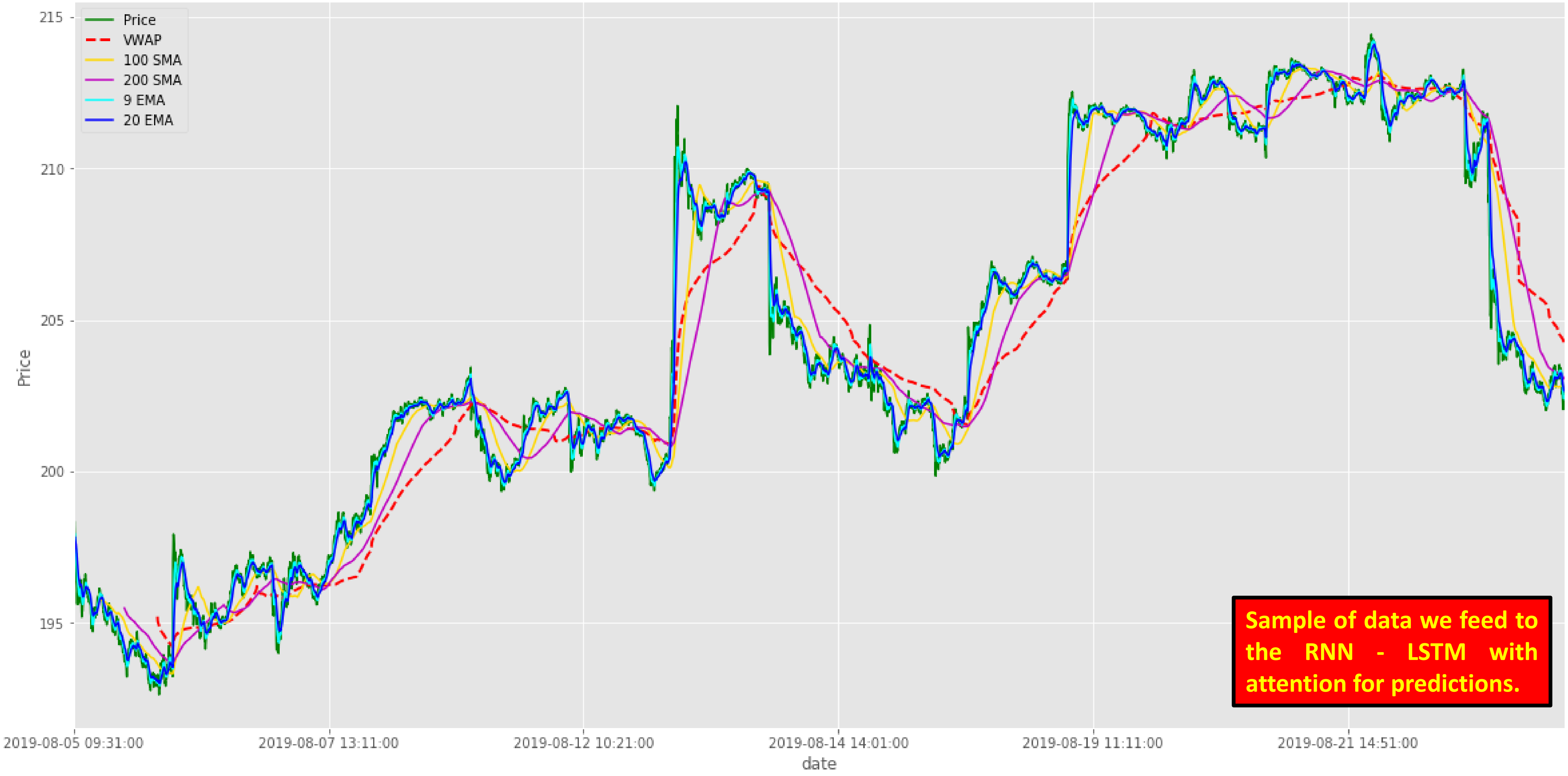
Sample of data we feed to the RNN - LSTM with attention for predictions.

# Stock Price - MSFT



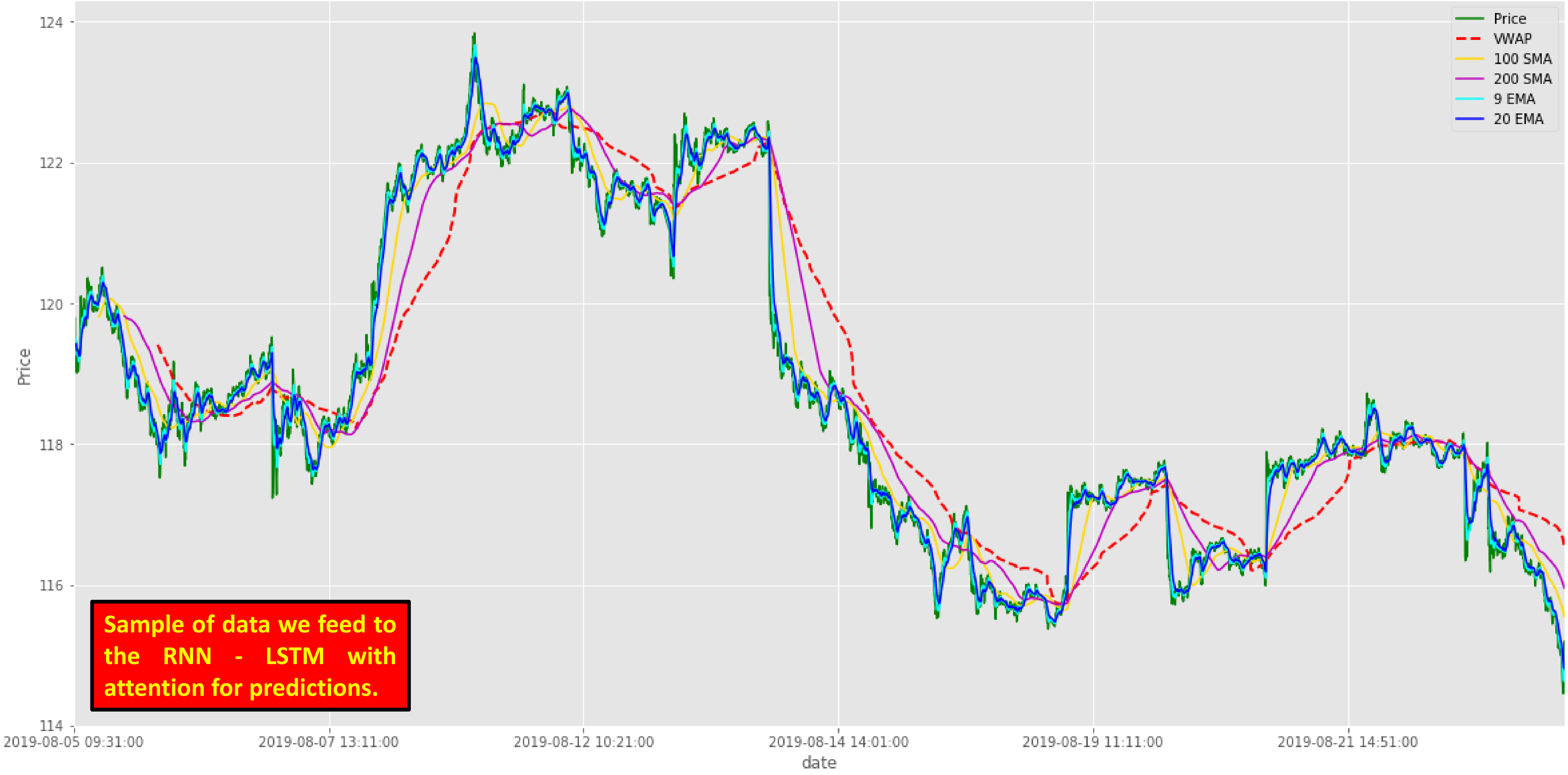
**Sample of data we feed to the RNN - LSTM with attention for predictions.**

# Stock Price - AAPL



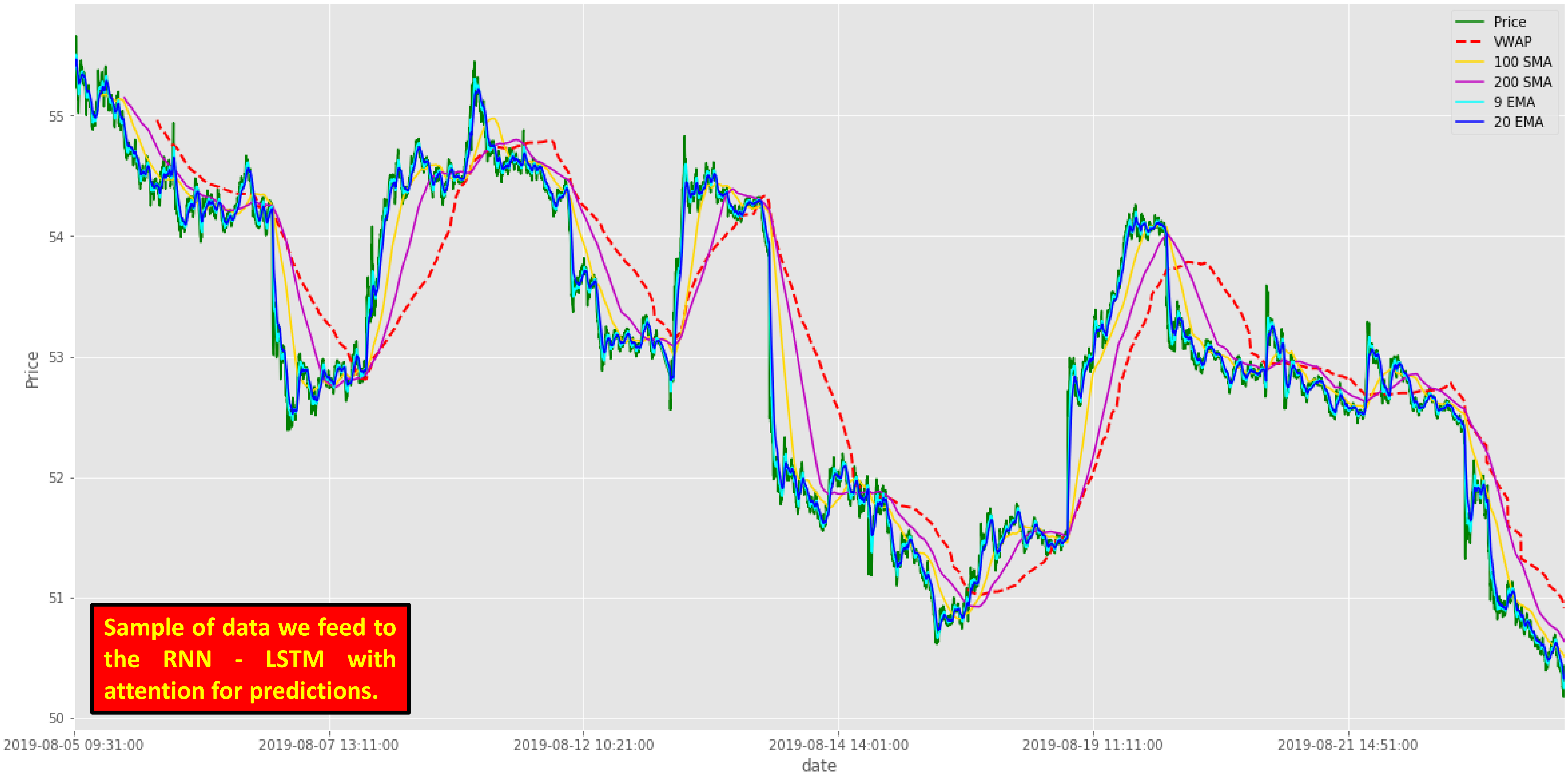
Sample of data we feed to the RNN - LSTM with attention for predictions.

# Stock Price - CVX



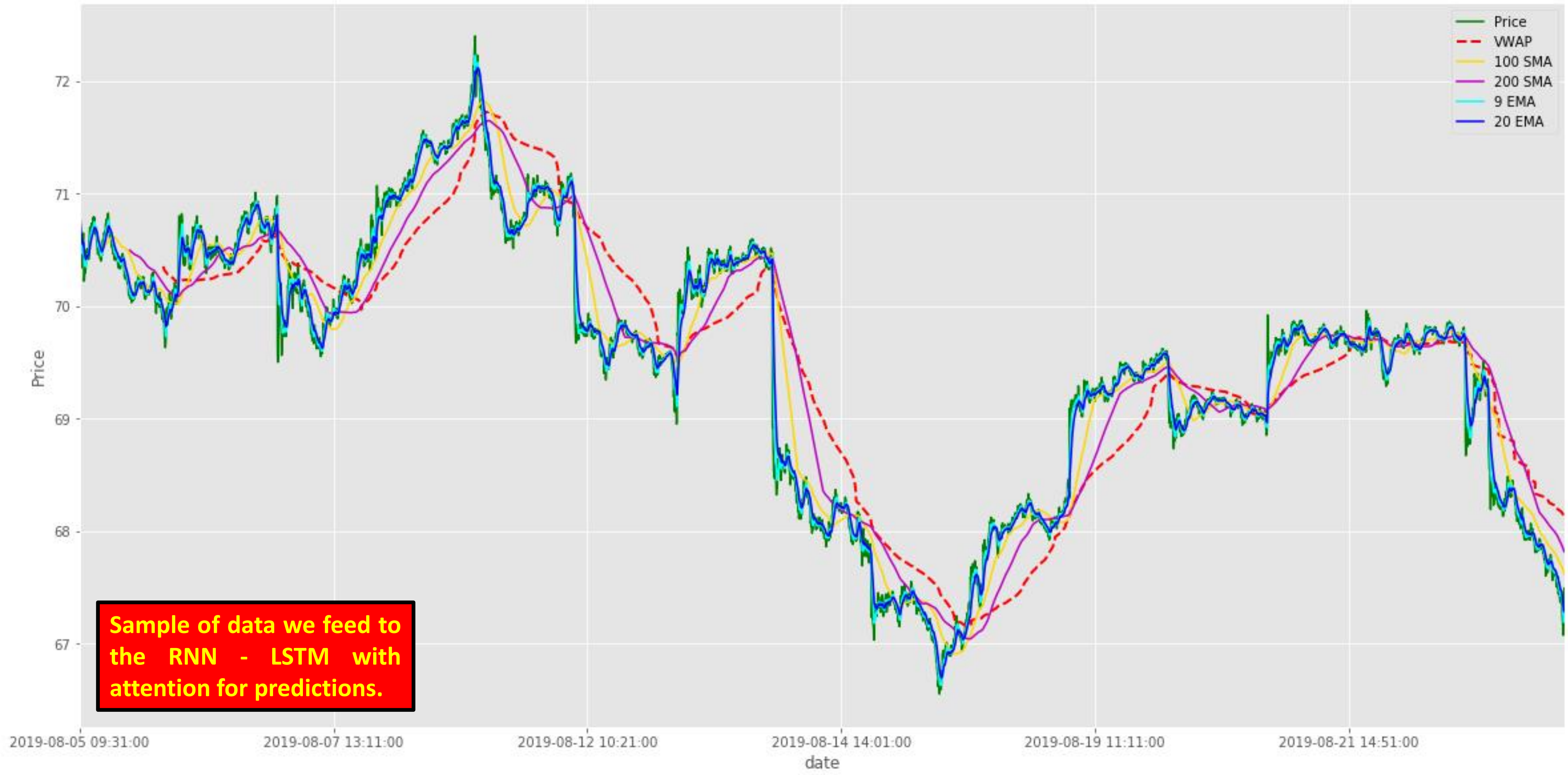
Sample of data we feed to the RNN - LSTM with attention for predictions.

# Stock Price - COP



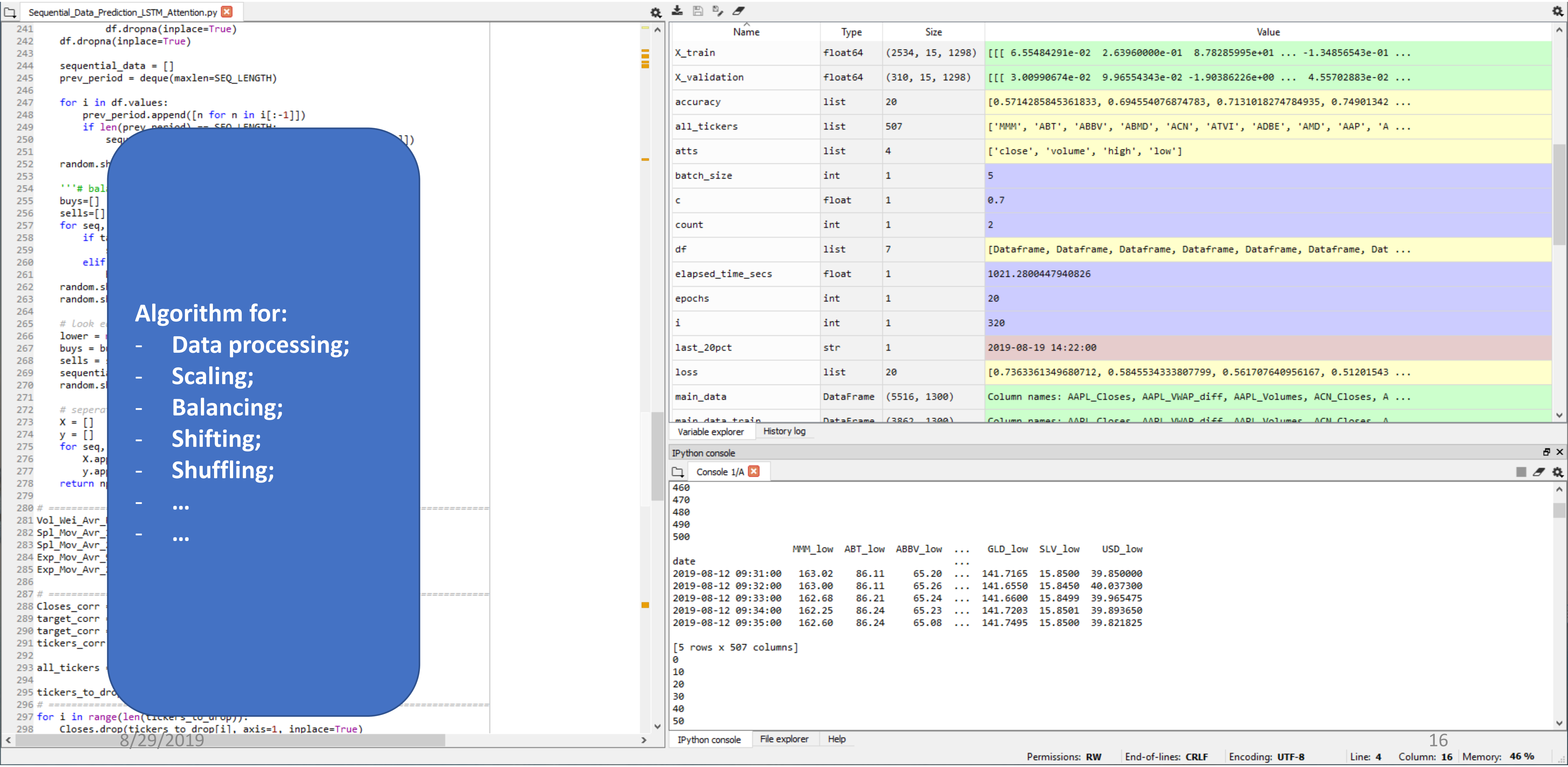
Sample of data we feed to the RNN - LSTM with attention for predictions.

# Stock Price - XOM



**Sample of data we feed to the RNN - LSTM with attention for predictions.**

# Algorithm Part 2: Data preprocessing



```python
            df.dropna(inplace=True)
    df.dropna(inplace=True)

    sequential_data = []
    prev_period = deque(maxlen=SEQ_LENGTH)

    for i in df.values:
        prev_period.append([n for n in i[:-1]])
        if len(prev_period) == SEQ_LENGTH:
            seq                                              ]])

    random.sh

    '''# bal
    buys=[]
    sells=[]
    for seq,
        if t

        elif

    random.s
    random.s

    # Look e
    lower =
    buys = b
    sells =
    sequenti
    random.s

    # sepera
    X = []
    y = []
    for seq,
        X.ap
        y.ap
    return n

# =========                                                  =============
Vol_Wei_Avr_
Spl_Mov_Avr_
Spl_Mov_Avr_
Exp_Mov_Avr_
Exp_Mov_Avr_

# =========                                                  =============
Closes_corr
target_corr
target_corr
tickers_corr

all_tickers

tickers_to_dro
# ==========                                                 =============
for i in range(len(tickers_to_drop)):
    Closes.drop(tickers_to_drop[i], axis=1, inplace=True)
```

| Name | Type | Size | Value |
|---|---|---|---|
| X_train | float64 | (2534, 15, 1298) | [[[ 6.55484291e-02  2.63960000e-01  8.78285995e+01 ... -1.34856543e-01 ... |
| X_validation | float64 | (310, 15, 1298) | [[[ 3.00990674e-02  9.96554343e-02 -1.90386226e+00 ...  4.55702883e-02 ... |
| accuracy | list | 20 | [0.5714285845361833, 0.694554076874783, 0.7131018274784935, 0.74901342 ... |
| all_tickers | list | 507 | ['MMM', 'ABT', 'ABBV', 'ABMD', 'ACN', 'ATVI', 'ADBE', 'AMD', 'AAP', 'A ... |
| atts | list | 4 | ['close', 'volume', 'high', 'low'] |
| batch_size | int | 1 | 5 |
| c | float | 1 | 0.7 |
| count | int | 1 | 2 |
| df | list | 7 | [Dataframe, Dataframe, Dataframe, Dataframe, Dataframe, Dataframe, Dat ... |
| elapsed_time_secs | float | 1 | 1021.2800447940826 |
| epochs | int | 1 | 20 |
| i | int | 1 | 320 |
| last_20pct | str | 1 | 2019-08-19 14:22:00 |
| loss | list | 20 | [0.7363361349680712, 0.5845534333807799, 0.561707640956167, 0.51201543 ... |
| main_data | DataFrame | (5516, 1300) | Column names: AAPL_Closes, AAPL_VWAP_diff, AAPL_Volumes, ACN_Closes, A ... |
| main_data_train | DataFrame | (3862, 1300) | Column names: AAPL_Closes, AAPL_VWAP_diff, AAPL_Volumes, ACN_Closes, A |

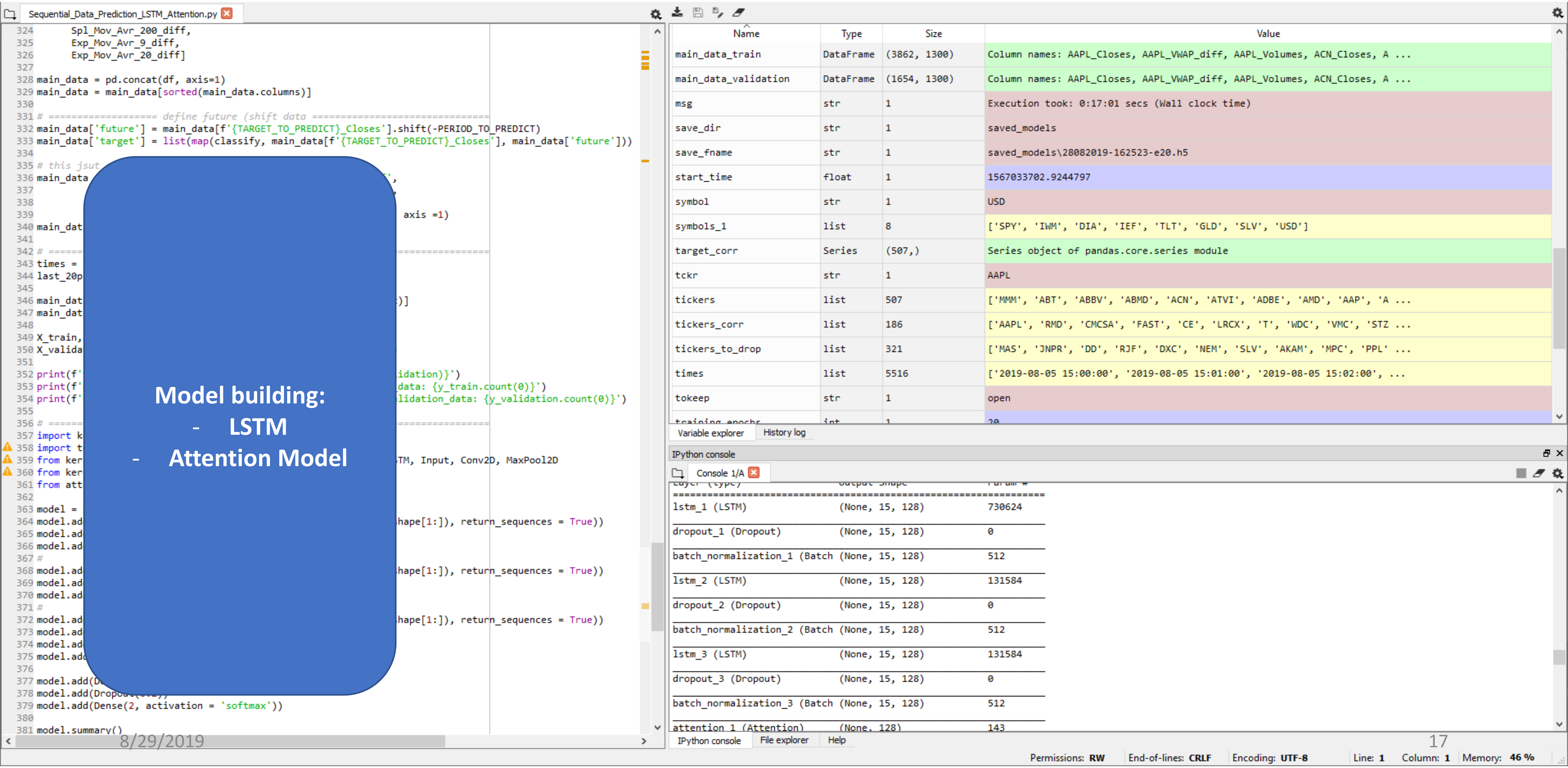
```
460
470
480
490
500
                     MMM_low  ABT_low  ABBV_low  ...   GLD_low  SLV_low    USD_low
date                                             ...
2019-08-12 09:31:00   163.02    86.11     65.20  ...  141.7165  15.8500  39.850000
2019-08-12 09:32:00   163.00    86.11     65.26  ...  141.6550  15.8450  40.037300
2019-08-12 09:33:00   162.68    86.21     65.24  ...  141.6600  15.8499  39.965475
2019-08-12 09:34:00   162.25    86.24     65.23  ...  141.7203  15.8501  39.893650
2019-08-12 09:35:00   162.60    86.24     65.08  ...  141.7495  15.8500  39.821825

[5 rows x 507 columns]
0
10
20
30
40
50
```

Permissions: RW   End-of-lines: CRLF   Encoding: UTF-8   Line: 4   Column: 16   Memory: 46 %

Algorithm for:

- Data processing;
- Scaling;
- Balancing;
- Shifting;
- Shuffling;
- …
- …

# Algorithm Part 2: Model building



```
      Spl_Mov_Avr_200_diff,
      Exp_Mov_Avr_9_diff,
      Exp_Mov_Avr_20_diff]

main_data = pd.concat(df, axis=1)
main_data = main_data[sorted(main_data.columns)]

# =================== define future (shift data ===============================
main_data['future'] = main_data[f'{TARGET_TO_PREDICT}_Closes'].shift(-PERIOD_TO_PREDICT)
main_data['target'] = list(map(classify, main_data[f'{TARGET_TO_PREDICT}_Closes'], main_data['future']))

# this jsut
main_data                                                  ',
                                                           ,

                                                            axis =1)
main_dat

# ======                                                   ================
times =
last_20p

main_dat                                                   )]
main_dat

X_train,
X_valida

print(f'                                                   idation)}')
print(f'                                                   data: {y_train.count(0)}')
print(f'                                                   lidation_data: {y_validation.count(0)}')

# ======                                                   ================
import k
import t
from ker                                                   TM, Input, Conv2D, MaxPool2D
from ker
from att

model =
model.ad                                                   hape[1:]), return_sequences = True))
model.ad
model.ad
#
model.ad                                                   hape[1:]), return_sequences = True))
model.ad
model.ad
#
model.ad                                                   hape[1:]), return_sequences = True))
model.ad
model.ad
model.ad

model.add(D
model.add(Dropo
model.add(Dense(2, activation = 'softmax'))

model.summary()
```

Model building:

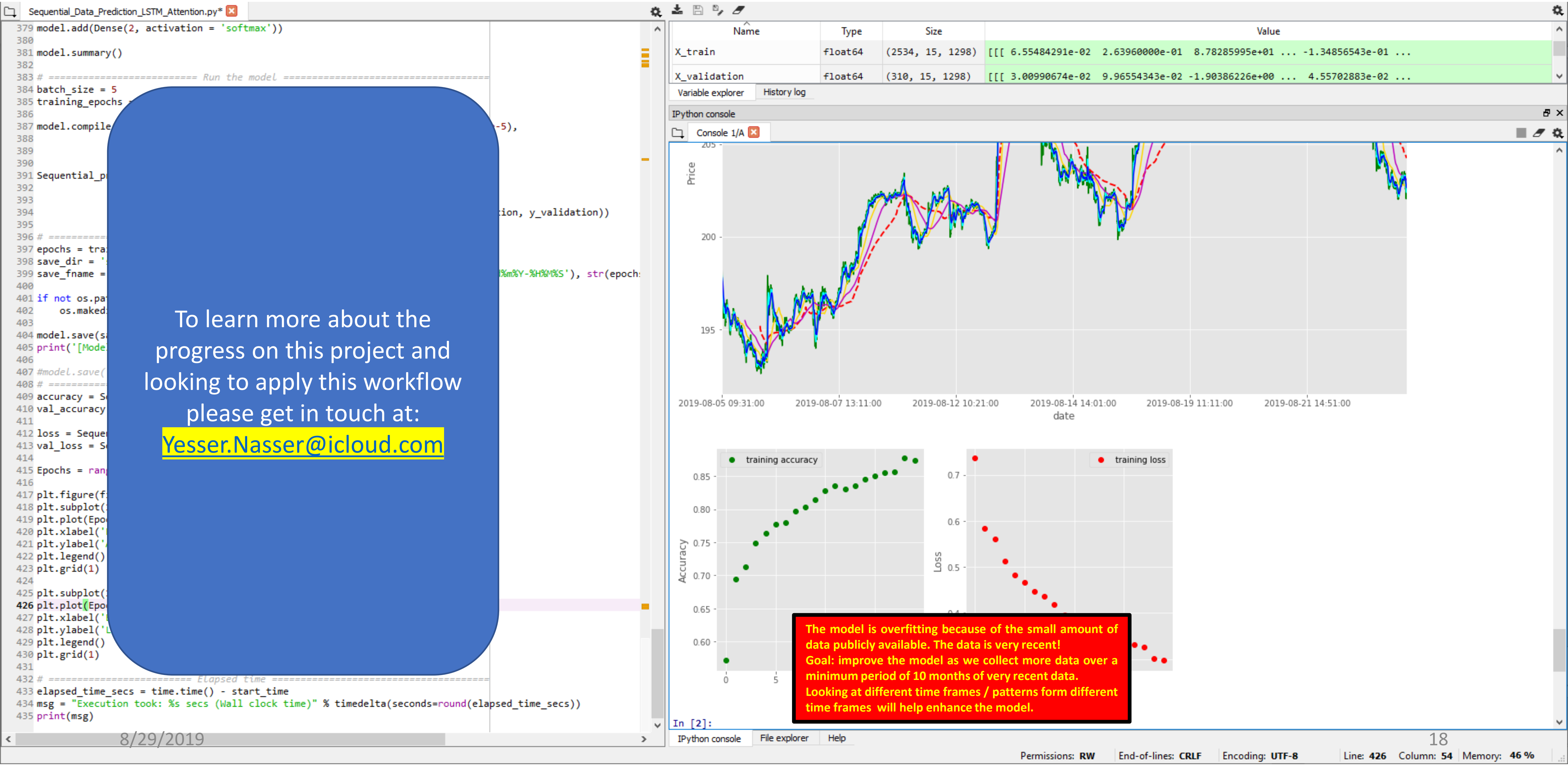
- LSTM
- Attention Model

| Name | Type | Size | Value |
|---|---|---|---|
| main_data_train | DataFrame | (3862, 1300) | Column names: AAPL_Closes, AAPL_VWAP_diff, AAPL_Volumes, ACN_Closes, A ... |
| main_data_validation | DataFrame | (1654, 1300) | Column names: AAPL_Closes, AAPL_VWAP_diff, AAPL_Volumes, ACN_Closes, A ... |
| msg | str | 1 | Execution took: 0:17:01 secs (Wall clock time) |
| save_dir | str | 1 | saved_models |
| save_fname | str | 1 | saved_models\28082019-162523-e20.h5 |
| start_time | float | 1 | 1567033702.9244797 |
| symbol | str | 1 | USD |
| symbols_1 | list | 8 | ['SPY', 'IWM', 'DIA', 'IEF', 'TLT', 'GLD', 'SLV', 'USD'] |
| target_corr | Series | (507,) | Series object of pandas.core.series module |
| tckr | str | 1 | AAPL |
| tickers | list | 507 | ['MMM', 'ABT', 'ABBV', 'ABMD', 'ACN', 'ATVI', 'ADBE', 'AMD', 'AAP', 'A ... |
| tickers_corr | list | 186 | ['AAPL', 'RMD', 'CMCSA', 'FAST', 'CE', 'LRCX', 'T', 'WDC', 'VMC', 'STZ ... |
| tickers_to_drop | list | 321 | ['MAS', 'JNPR', 'DD', 'RJF', 'DXC', 'NEM', 'SLV', 'AKAM', 'MPC', 'PPL' ... |
| times | list | 5516 | ['2019-08-05 15:00:00', '2019-08-05 15:01:00', '2019-08-05 15:02:00', ... |
| tokeep | str | 1 | open |
| training_epochs | int | 1 | 20 |

```
Layer (type)                 Output Shape              Param #
=================================================================
lstm_1 (LSTM)                (None, 15, 128)           730624
_________________________________________________________________
dropout_1 (Dropout)          (None, 15, 128)           0
_________________________________________________________________
batch_normalization_1 (Batch (None, 15, 128)           512
_________________________________________________________________
lstm_2 (LSTM)                (None, 15, 128)           131584
_________________________________________________________________
dropout_2 (Dropout)          (None, 15, 128)           0
_________________________________________________________________
batch_normalization_2 (Batch (None, 15, 128)           512
_________________________________________________________________
lstm_3 (LSTM)                (None, 15, 128)           131584
_________________________________________________________________
dropout_3 (Dropout)          (None, 15, 128)           0
_________________________________________________________________
batch_normalization_3 (Batch (None, 15, 128)           512
_________________________________________________________________
attention_1 (Attention)      (None, 128)               143
```

Permissions: RW  End-of-lines: CRLF  Encoding: UTF-8  Line: 1  Column: 1  Memory: 46 %

# Algorithm Part 3: Running Model – preliminary results



```
model.add(Dense(2, activation = 'softmax'))

model.summary()

# ========================= Run the model ===================================
batch_size = 5
training_epochs 

model.compile                                                              -5),



Sequential_p



                                                                           ion, y_validation))

# =========
epochs = tra
save_dir = '
save_fname =                                                              %m%Y-%H%M%S'), str(epoch

if not os.pa
    os.maked

model.save(s
print('[Mode

#model.save(
# =========
accuracy = S
val_accuracy

loss = Seque
val_loss = S

Epochs = ran

plt.figure(f
plt.subplot(
plt.plot(Epo
plt.xlabel('
plt.ylabel('
plt.legend()
plt.grid(1)

plt.subplot(
plt.plot(Epo
plt.xlabel('
plt.ylabel('L
plt.legend()
plt.grid(1)

# ======================== Elapsed time ========================================
elapsed_time_secs = time.time() - start_time
msg = "Execution took: %s secs (Wall clock time)" % timedelta(seconds=round(elapsed_time_secs))
print(msg)
```

| Name | Type | Size | Value |
|---|---|---|---|
| X_train | float64 | (2534, 15, 1298) | [[[ 6.55484291e-02 2.63960000e-01 8.78285995e+01 ... -1.34856543e-01 ... |
| X_validation | float64 | (310, 15, 1298) | [[[ 3.00990674e-02 9.96554343e-02 -1.90386226e+00 ... 4.55702883e-02 ... |

```
In [2]:
```

To learn more about the progress on this project and looking to apply this workflow please get in touch at: Yesser.Nasser@icloud.com

The model is overfitting because of the small amount of data publicly available. The data is very recent!
Goal: improve the model as we collect more data over a minimum period of 10 months of very recent data.
Looking at different time frames / patterns form different time frames will help enhance the model.