

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

Sequential_Data_Prediction_LSTM_Attention.py

```
1 #-*- coding: utf-8 -*-
2 """
3 Updated on Wed Aug 28 16:45:26 2019
4 @author: Yesser H. Nasser
5 Collect the data, pre-process the data to handle nan values
6 """
7
8 import re
9 import time
10 import bs4 as bs
11 import pickle
12 import requests
13 import datetime as dt
14 import pandas as pd
15 import random
16 import numpy as np
17 import pandas_datareader.data as web
18 import os
19 import matplotlib.pyplot as plt
20 import matplotlib.dates as mdates
21 import matplotlib.ticker as mticker
22
23 from sklearn import preprocessing
24 from mpl_finance import candlestick_ohlc
25 from matplotlib import style
26 from alpha_vantage.timeseries import TimeSeries
27 from sklearn import preprocessing
28 from collections import defaultdict
29 from datetime import datetime
30
31 style.use('f5')
32
33 start = time.time()
34
35 # =====
36 SEQ_LEN = 100
37 TARGET_LEN = 100
38 PERIOD = 'D'
39 c = 0.01
40
41 # =====
42 weeks = 10, # weeks of data collections
43 atts = ['close', 'volume', 'high', 'low'], # accumulate indicators
44 # =====
45 def save_tickers(tickers, f):
46     re = os.path.join('tickers', f)
47     with open(re, 'wb') as f:
48         pickle.dump(tickers, f)
49     print(tickers)
50     return tickers
51
52 # =====
53 # and ticker != 'GL' and ticker != 'IEX'
54
55 with open('tickers.pkl', 'wb') as f:
56     pickle.dump(tickers, f)
57     print(tickers)
58     return tickers
59
60 tickers = save_tickers(tickers, 'tickers.pkl')
```

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 ...

Variable explorer History log

IPython console

Console 1/A

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

IPython console File explorer Help

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

Sequential_Data_Prediction_LSTM_Attention.py

```
57 return tickers
58 tickers = save_sp500_tickers()
59
60 # append additional tickers
61 symbols_1 = ['SPY', 'IWM', 'DIA', 'IEF', 'TLT', 'GLD', 'SLV', 'USD',]
62 for symbol in symbols_1:
63     if symbol not in tickers:
64         tickers.append(symbol)
65
66 def preprocess_price_volume (df,Keep,week):
67     df = df.interpolate()
68     if Keep == 'volume':
69         df = df.round(0)
70         df = df.fillna(0)
71     else:
72         for col in df.columns:
73             df[col] = df[col].interpolate()
74             df[col] = df[col].round(0)
75             df[col] = df[col].fillna(0)
76
77     df.columns = df.columns + '_inter'
78     print(df)
79     df.to_csv('data_inter.csv')
80
81 # =====
82 # =====
83 def compile_data(df, week):
84     with open('data_inter.csv', 'r') as f:
85         df = pd.read_csv(f)
86
87     # store data in a list
88     for i in range(0, df.shape[0], week):
89         df = df[i:i+week]
90
91         # create a list of DataFrames
92         df_list = []
93         for i in range(0, df.shape[0], week):
94             df_list.append(df[i:i+week])
95
96         # create a list of sequences
97         df_list = df_list[0:df_list.shape[0]-week]
98
99         # create a list of sequences
100         df_list = df_list[0:df_list.shape[0]-week]
101
102         # create a list of sequences
103         df_list = df_list[0:df_list.shape[0]-week]
104
105         # create a list of sequences
106         df_list = df_list[0:df_list.shape[0]-week]
107
108         # create a list of sequences
109         df_list = df_list[0:df_list.shape[0]-week]
110
111         # create a list of sequences
112         df_list = df_list[0:df_list.shape[0]-week]
113
114         # create a list of sequences
115         df_list = df_list[0:df_list.shape[0]-week]
```

10/11/2019

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 ...

Variable explorer

History log

IPython console

Console 1/A

460
470
480
490
500

date MMM_low ABT_low ABBV_low ... GLD_low SLV_low USD_low
...
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

IPython console

File explorer

Help

- Compiling data

- Building Dataframes

- Sequences

all the no-alphabet i

ker name

Permissions: RW

End-of-lines: CRLF

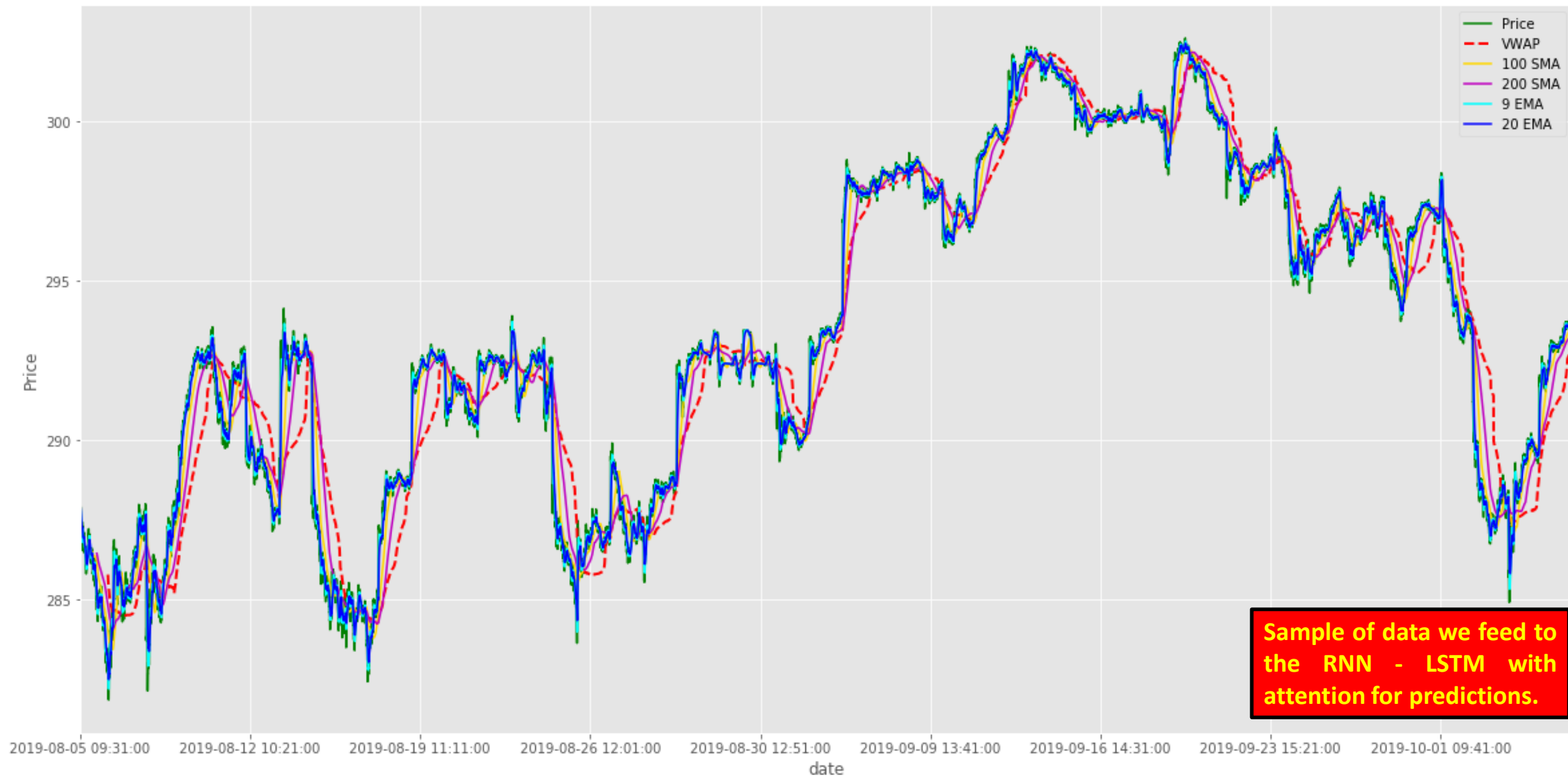
Encoding: UTF-8

Line: 4

Column: 16

Memory: 49 %

SPY



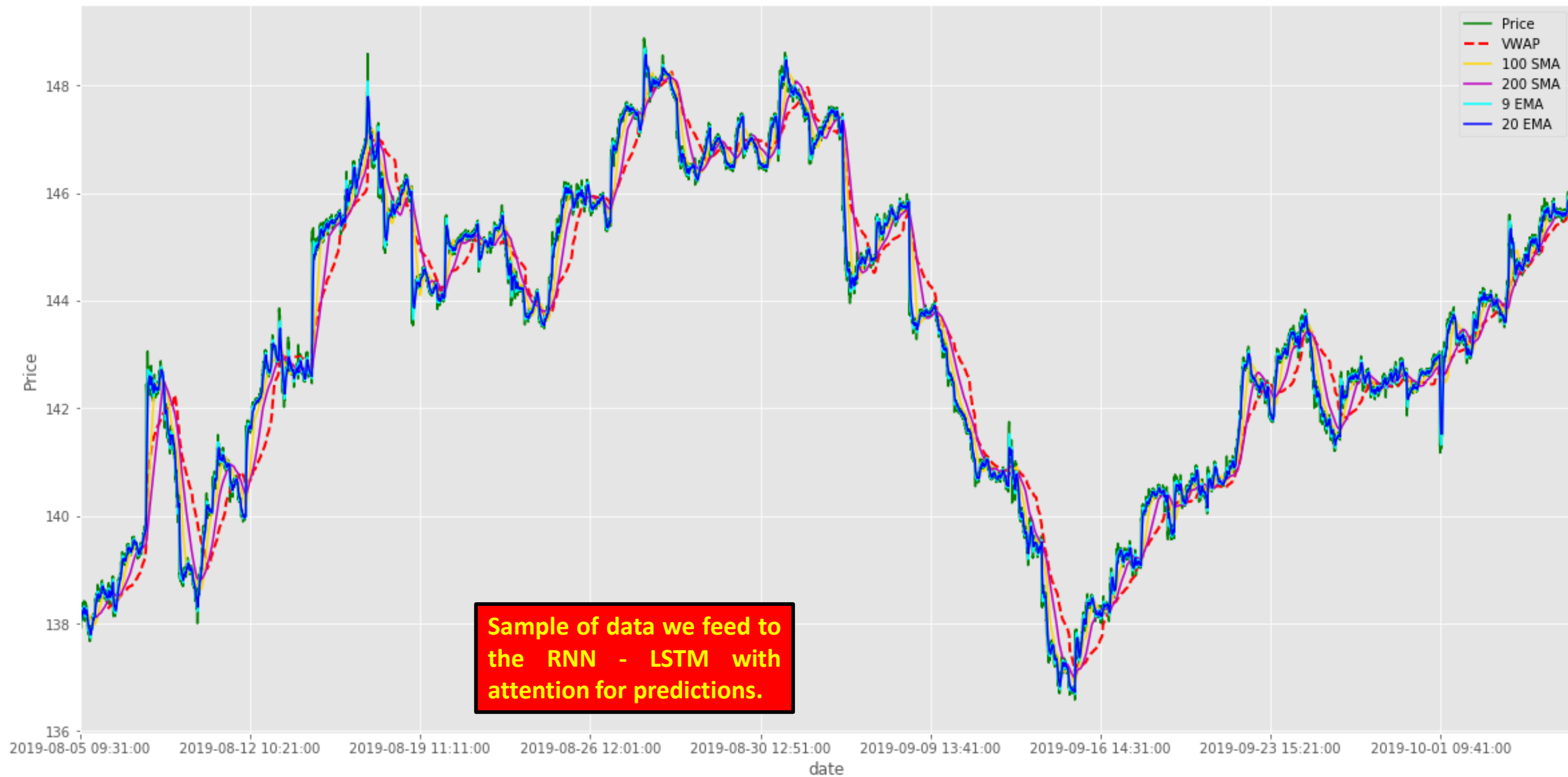
Sample of data we feed to the RNN - LSTM with attention for predictions.

10/11/2019

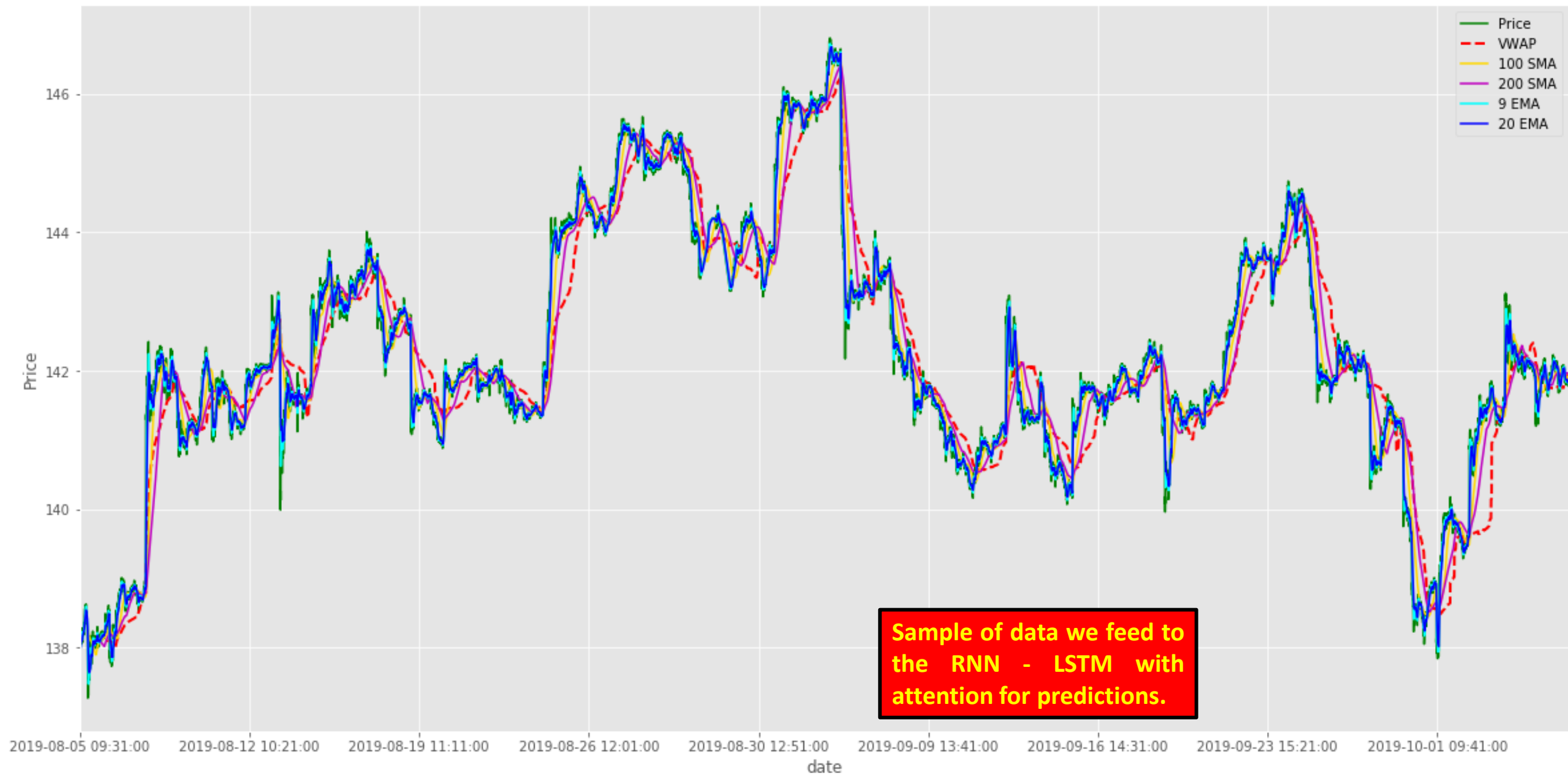
N.B: Time frame : 1 min

Very resent data!

TLT



GLD



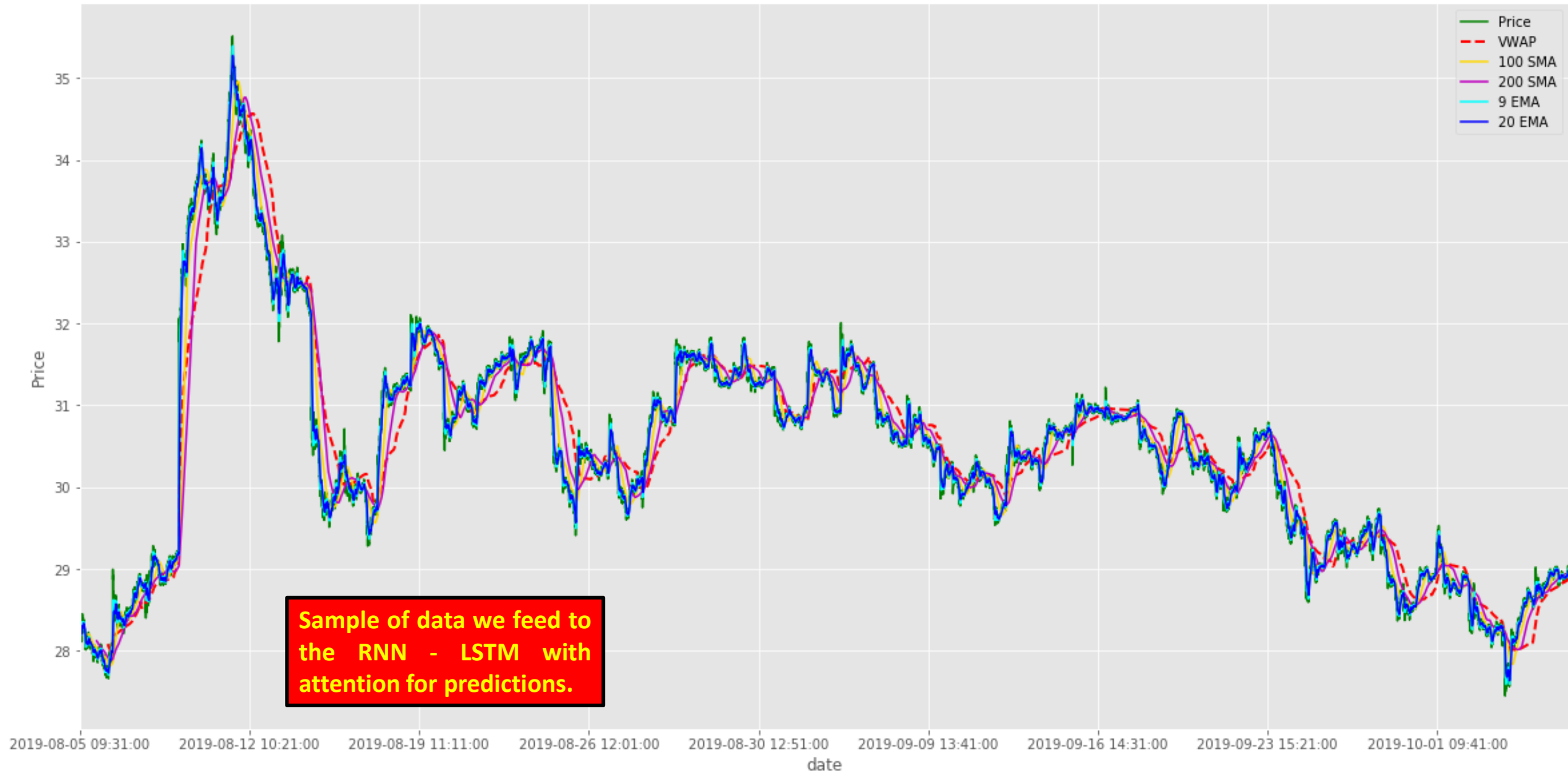
10/11/2019

N.B: Time frame : 1 min

DIA



Stock Price - AMD



10/11/2019

N.B: Time frame : 1 min

Stock Price - MU

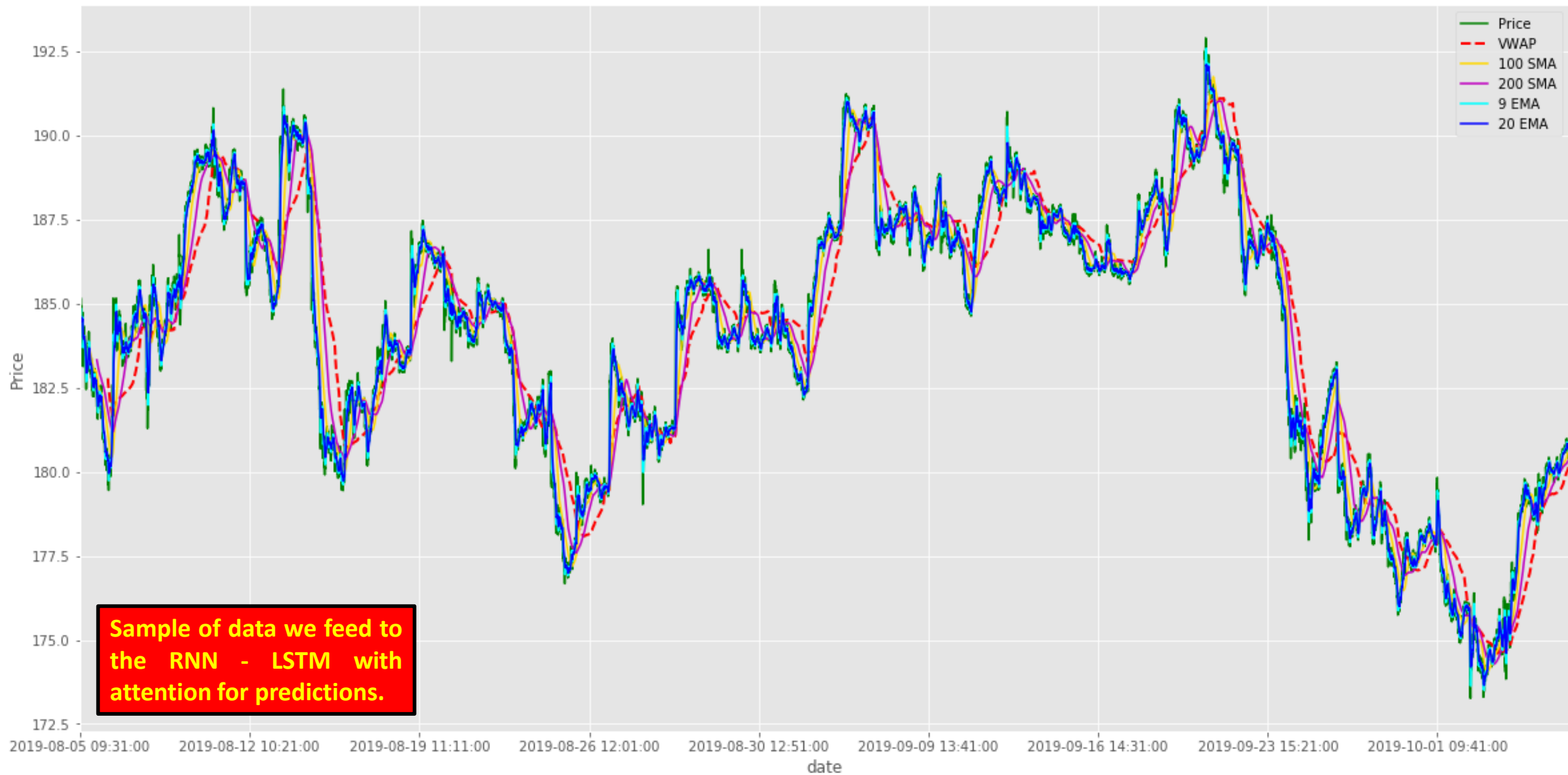


Sample of data we feed to the RNN - LSTM with attention for predictions.

10/11/2019

N.B: Time frame : 1 min

Stock Price - FB

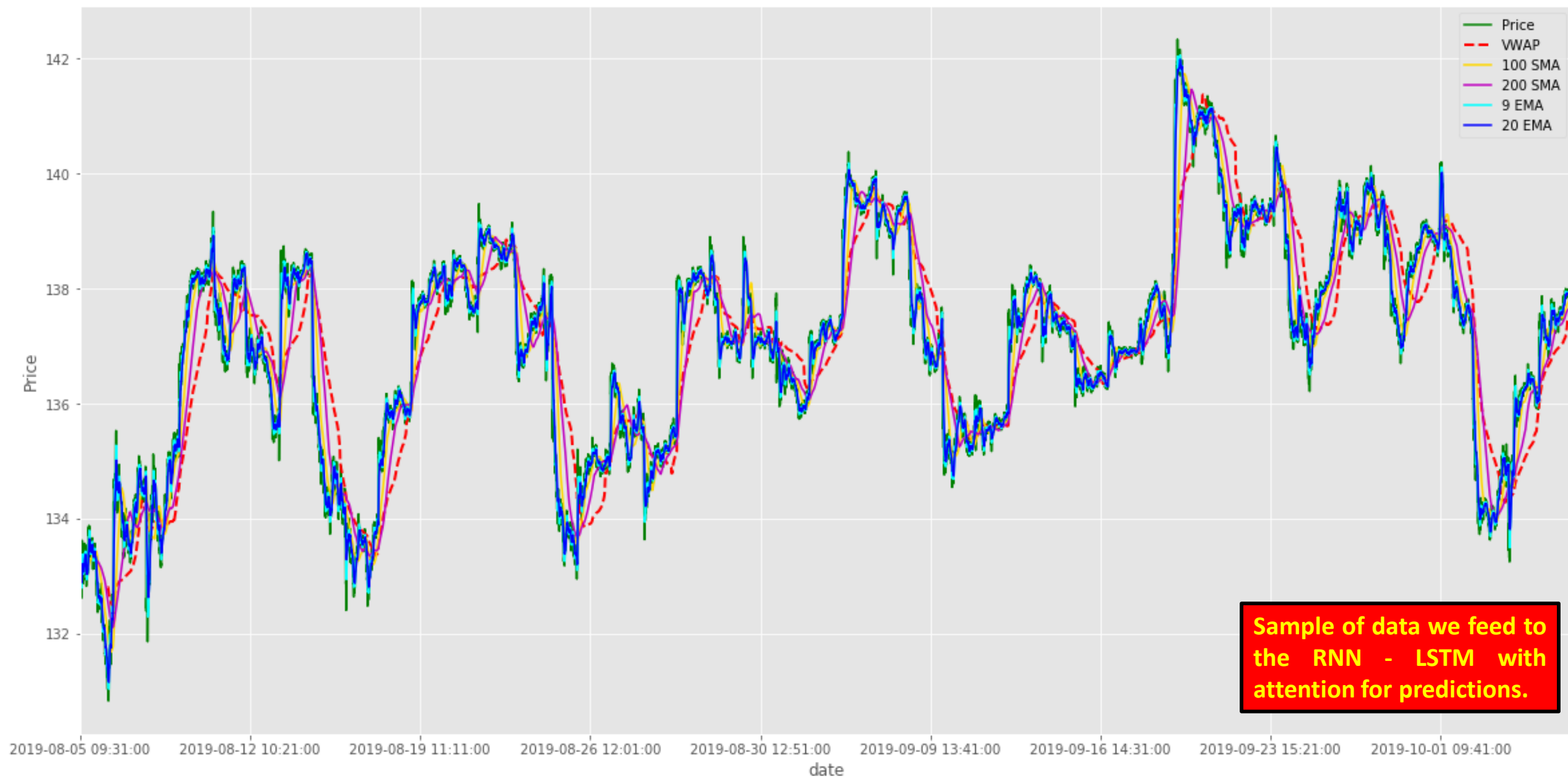


10/11/2019

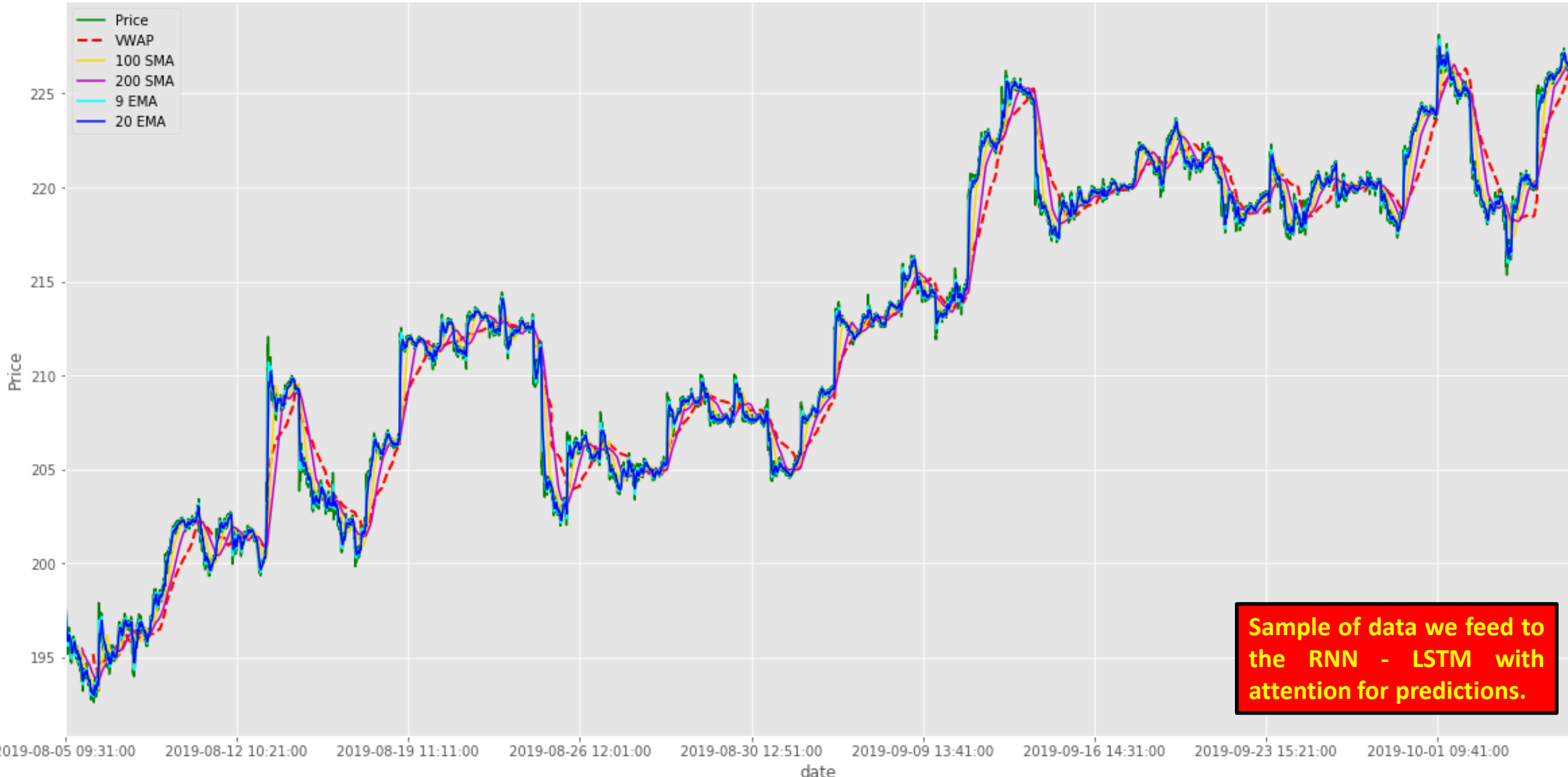
10

N.B: Time frame : 1 min

Stock Price - MSFT



Stock Price - AAPL



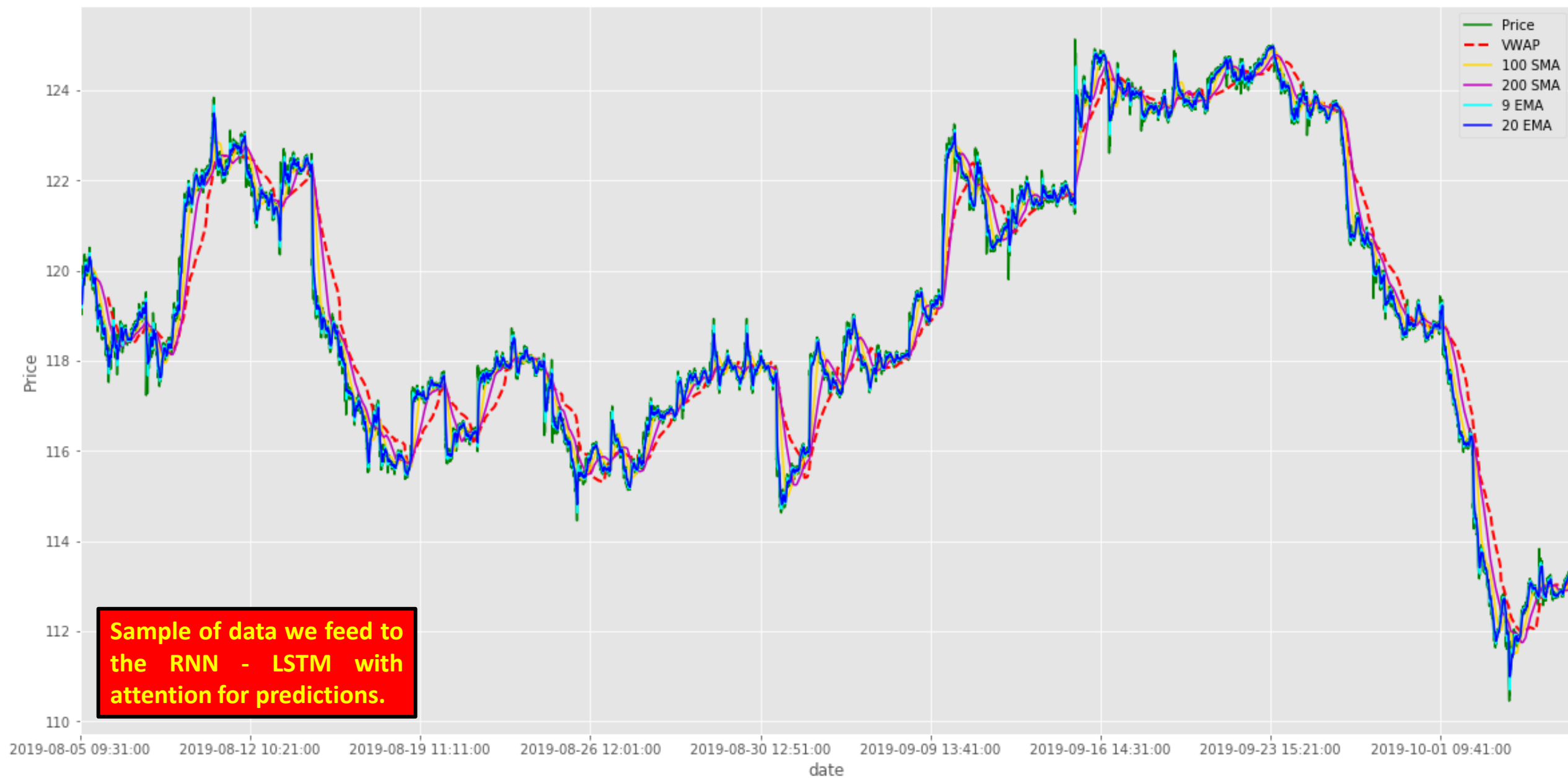
Sample of data we feed to the RNN - LSTM with attention for predictions.

10/11/2019

N.B: Time frame : 1 min

12

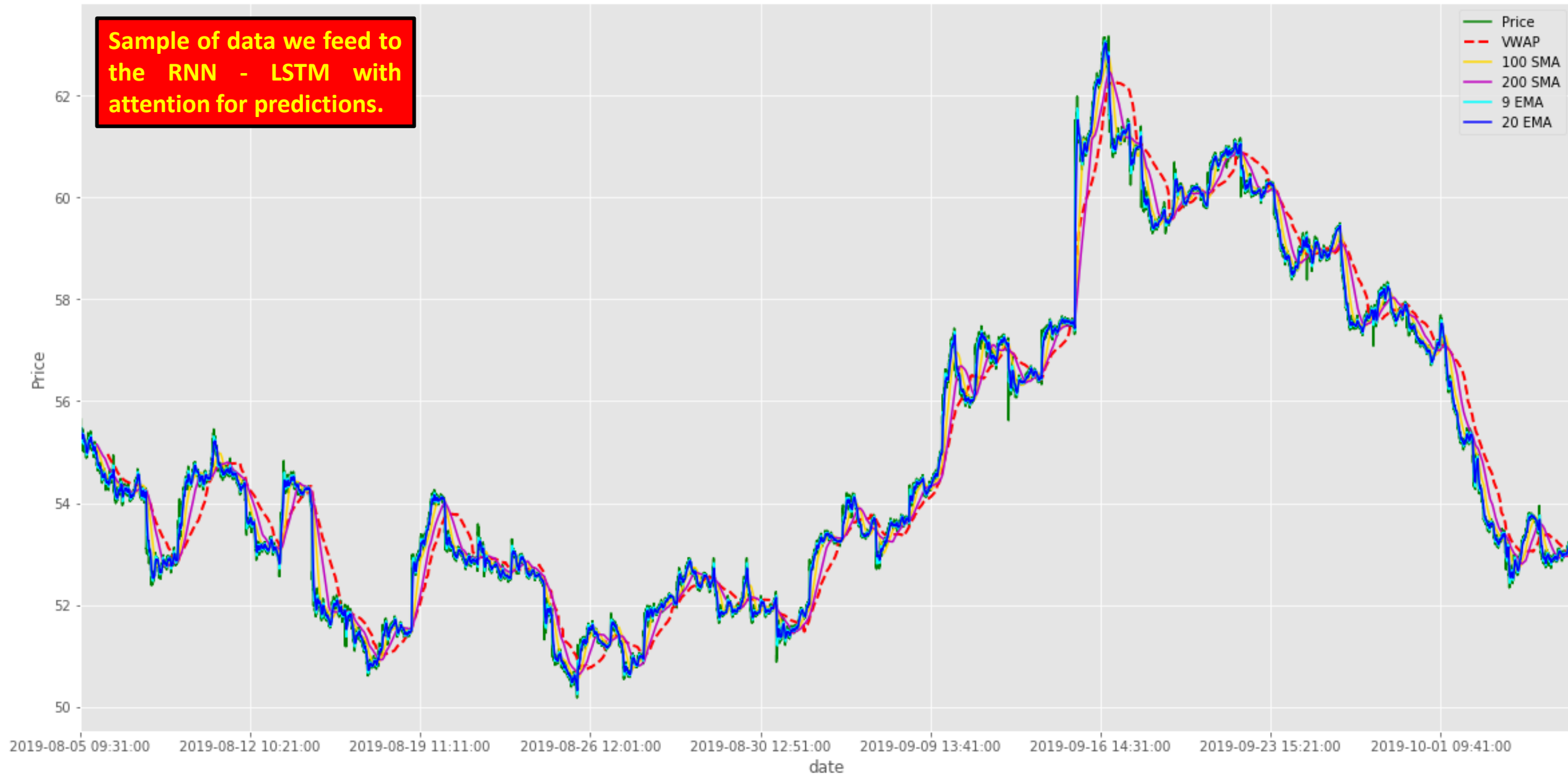
Stock Price - CVX



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.



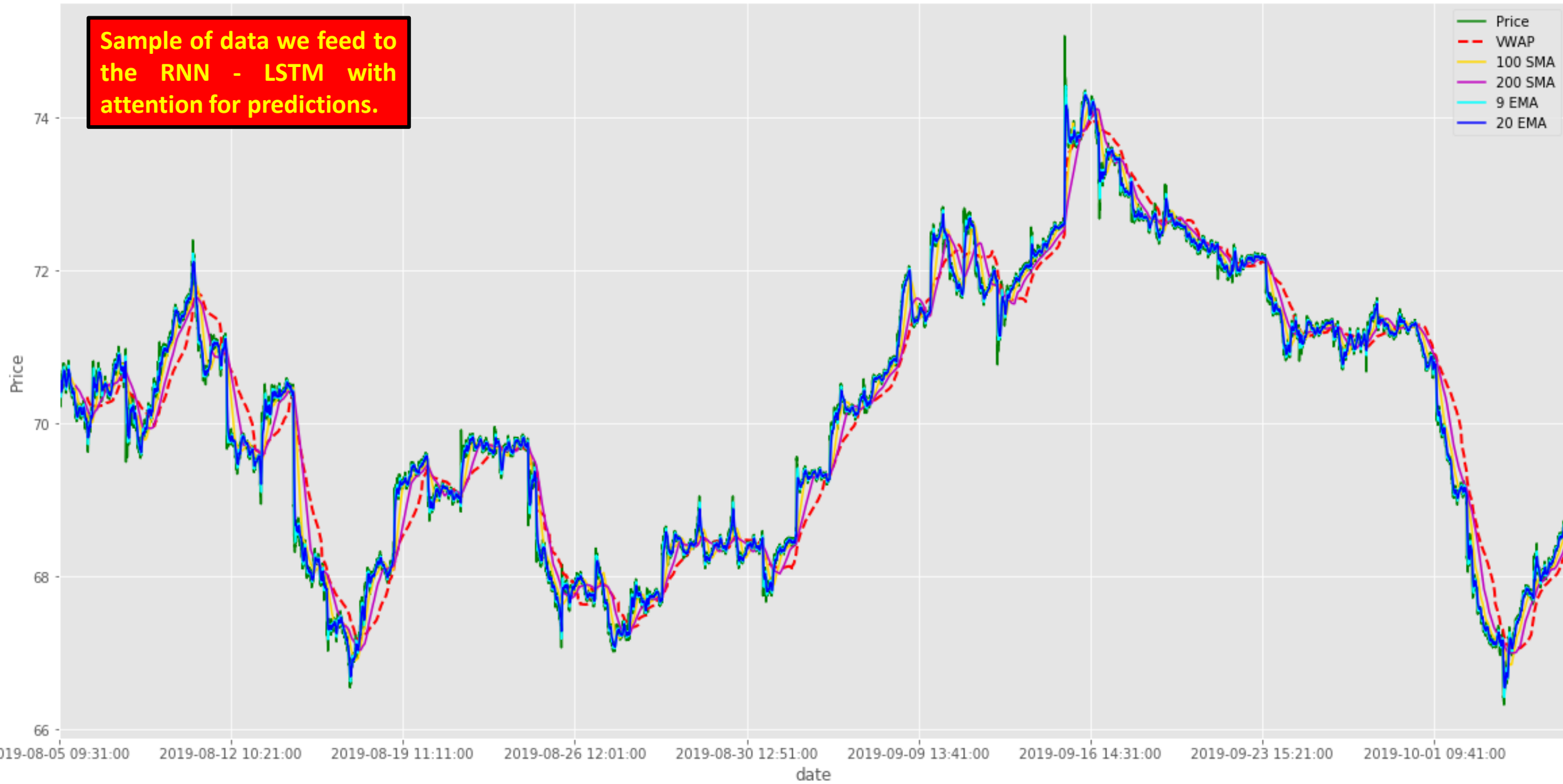
10/11/2019

14

N.B: Time frame : 1 min

Stock Price - XOM

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



10/11/2019

15

N.B: Time frame : 1 min

Algorithm Part 2: Data preprocessing

Algorithm for:

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

```
241 df.dropna(inplace=True)
242 df.dropna(inplace=True)
243
244 sequential_data = []
245 prev_period = deque(maxlen=SEQ_LENGTH)
246
247 for i in df.values:
248     prev_period.append([n for n in i[:-1]])
249     if len(prev_period) == SEQ_LENGTH:
250         sequential_data.append(prev_period)
251         prev_period = deque(maxlen=SEQ_LENGTH)
252
253 random.shuffle(sequential_data)
254
255 # bal
256 buys=[]
257 sells=[]
258 for seq,
259     if t
260     elif
261
262 random.shuffle(buys)
263 random.shuffle(sells)
264
265 # Look e
266 lower =
267 buys = b
268 sells =
269 sequenti
270 random.s
271
272 # separa
273 X = []
274 y = []
275 for seq,
276     X.ap
277     y.ap
278     return n
279
280 # =====
281 Vol_Wei_Avr_
282 Spl_Mov_Avr_
283 Spl_Mov_Avr_
284 Exp_Mov_Avr_
285 Exp_Mov_Avr_
286
287 # =====
288 Closes_corr
289 target_corr
290 target_corr
291 tickers_corr
292
293 all_tickers
294
295 tickers_to_drop
296 # =====
297 for i in range(len(tickers_to_drop)):
298     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 ...

Variable explorer History log

IPython console

Console 1/A

```
460
470
480
490
500
date      MMM_low ABT_low ABBV_low ... GLD_low SLV_low USD_low
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
```

IPython console File explorer Help

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

Algorithm Part 2: Model building

```
324 Spl_Mov_Avr_200_diff,
325 Exp_Mov_Avr_9_diff,
326 Exp_Mov_Avr_20_diff]
327
328 main_data = pd.concat(df, axis=1)
329 main_data = main_data[sorted(main_data.columns)]
330
331 # ===== define future (shift data) =====
332 main_data['future'] = main_data[f'{TARGET_TO_PREDICT}_Closes'].shift(-PERIOD_TO_PREDICT)
333 main_data['target'] = list(map(classify, main_data[f'{TARGET_TO_PREDICT}_Closes'], main_data['future']))
334
335 # this jsut
336 main_data
337
338
339
340 main_data
341
342 # =====
343 times =
344 last_20p
345
346 main_data
347 main_data
348
349 X_train,
350 X_validation,
351
352 print(f'
353 print(f'
354 print(f'
355
356 # =====
357 import keras
358 import tensorflow as tf
359 from keras.layers import LSTM, Input, Conv2D, MaxPool2D
360 from keras.models import Sequential
361 from tensorflow.keras import layers
362
363 model =
364 model.add(LSTM(128, input_shape=[1:], return_sequences = True))
365 model.add(LSTM(128, input_shape=[1:], return_sequences = True))
366 model.add(LSTM(128, input_shape=[1:], return_sequences = True))
367 #
368 model.add(LSTM(128, input_shape=[1:], return_sequences = True))
369 model.add(LSTM(128, input_shape=[1:], return_sequences = True))
370 model.add(LSTM(128, input_shape=[1:], return_sequences = True))
371 #
372 model.add(LSTM(128, input_shape=[1:], return_sequences = True))
373 model.add(LSTM(128, input_shape=[1:], return_sequences = True))
374 model.add(LSTM(128, input_shape=[1:], return_sequences = True))
375 model.add(LSTM(128, input_shape=[1:], return_sequences = True))
376
377 model.add(Dense(128))
378 model.add(Dense(128))
379 model.add(Dense(2, activation = 'softmax'))
380
381 model.summary()
```

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

Variable explorerHistory log

IPython console

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

IPython consoleFile explorerHelp

Model building:
- LSTM
- Attention Model

10/11/2019

17

Permissions: RWEnd-of-lines: CRLFEncoding: UTF-8Line: 1Column: 1Memory: 46 %

Algorithm Part 3: Running Model – preliminary results

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

