Neural Networks for Bitcoin Predictions Buy/Sell

Setup

In [3]:
```
#tensorflow
import tensorflow as tf
from tensorflow import keras

#sklearn
import sklearn
from sklearn.preprocessing import StandardScaler,MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer

# Common imports
import numpy as np
import pandas as pd
import os
import os.path
import urllib

# To plot pretty figures
import matplotlib.pyplot as plt
```

In [4]:
```
!pip install yfinance -q
!pip install plotly -q
```

In [4]:
```
def plot_learning_curve( history ):
    pd.DataFrame(history.history).plot(figsize=(8, 5))
    plt.grid(True)
    ymin, ymax = [], []
    for x in history.history.keys():
        ymax.append( max(history.history[x]))
        ymin.append( min(history.history[x]))
    plt.gca().set_ylim(min(ymin), max(ymax))
    plt.xlabel("EPOCHS")
    plt.show()
```

In [5]:
```
def plot_actual_pred( actual, prediction ):
    plt.plot(actual, ".-", alpha=.6, label="Actual")
    plt.plot(prediction, ".-", alpha=.6, label="Prediction")
    plt.grid(True)
    plt.legend()
    plt.show()
```

In [6]:
```
def plot_forecast(Y, Y_pred, Y_actual=None):
    n_steps = Y.shape[0]
    ahead = Y_pred.shape[0]
    plt.plot(Y,"b-", alpha=.6)
    plt.plot(np.arange(n_steps, n_steps + ahead), Y_pred, "rx-", label="Forecast")
```

```
    plt.plot(np.arange(n_steps, n_steps + ahead), Y_actual, "b.-", label="Actual")
    plt.legend()
    plt.grid()
```

In [7]:
```
#define a metric that provides the MSE on the last step only (the real test for this sq
def last_time_step_mse(Y_true, Y_pred):
    return keras.metrics.mean_squared_error(Y_true[:, -1], Y_pred[:, -1])
```

Source Data

In [ ]:
```
import numpy as np
import pandas as pd

#Data Source
import yfinance as yf

#Data viz
import plotly.graph_objs as go
```

In [15]:
```
data = yf.download(tickers='BTC-USD', period = '730d', interval = '1m')
```

```
[**********************100%**********************]  1 of 1 completed

1 Failed download:
- BTC-USD: 1m data not available for startTime=1566489694 and endTime=1629561694. Only 7
days worth of 1m granularity data are allowed to be fetched per request.
```

In [14]:
```
data
```

Out[14]:

Datetime	Open	High	Low	Close	Adj Close	Volume
2019-08-23 00:00:00+01:00	10165.219727	10170.639648	10098.740234	10105.150391	10105.150391	0
2019-08-23 01:00:00+01:00	10104.750000	10148.740234	10056.280273	10129.469727	10129.469727	0
2019-08-23 02:00:00+01:00	10130.540039	10135.889648	10089.019531	10098.790039	10098.790039	6041282
2019-08-23 03:00:00+01:00	10099.860352	10135.589844	10073.019531	10112.269531	10112.269531	4493358
2019-08-23 04:00:00+01:00	10113.070312	10171.860352	10104.879883	10158.389648	10158.389648	6314782
...	...	...	...	...	...	...
2021-08-21 13:00:00+01:00	48532.269531	49282.613281	48516.335938	49238.253906	49238.253906	1343184896
2021-08-21 14:00:00+01:00	49233.621094	49381.378906	49091.433594	49145.945312	49145.945312	455217152
2021-08-21 15:00:00+01:00	49154.878906	49173.074219	48981.316406	49064.402344	49064.402344	227131392
2021-08-21 16:00:00+01:00	49034.507812	49278.613281	49015.531250	49222.429688	49222.429688	401707008

Datetime	Open	High	Low	Close	Adj Close	Volume
2021-08-21 16:57:02+01:00	49231.625000	49231.625000	49231.625000	49231.625000	49231.625000	0

17466 rows × 6 columns

In [16]:
```
!pip install bitfinex-tencars
```

```
Collecting bitfinex-tencars
  Downloading bitfinex_tencars-0.0.3-py3-none-any.whl (7.6 kB)
Installing collected packages: bitfinex-tencars
Successfully installed bitfinex-tencars-0.0.3
```

In [211…
```
import bitfinex
import datetime
import time
import pandas as pd

# Define query parameters
pair = 'BTCUSD' # Currency pair of interest
TIMEFRAME = '1h'#,'4h','1h','15m','1m'
TIMEFRAME_S = 3600 # seconds in TIMEFRAME

# Define the start date
t_start = datetime.datetime(2017, 1, 1, 0, 0)
t_start = time.mktime(t_start.timetuple()) * 1000

# Define the end date
t_stop = datetime.datetime(2021, 9, 5, 0, 0)
t_stop = time.mktime(t_stop.timetuple()) * 1000

def fetch_data(start, stop, symbol, interval, TIMEFRAME_S):
    limit = 1000    # We want the maximum of 1000 data points
    # Create api instance
    api_v2 = bitfinex.bitfinex_v2.api_v2()
    hour = TIMEFRAME_S * 1000
    step = hour * limit
    data = []

    total_steps = (stop-start)/hour
    while total_steps > 0:
        if total_steps < limit: # recalculating ending steps
            step = total_steps * hour

        end = start + step
        data += api_v2.candles(symbol=symbol, interval=interval, limit=limit, start=sta
        print(pd.to_datetime(start, unit='ms'), pd.to_datetime(end, unit='ms'), "steps
        start = start + step
        total_steps -= limit
        time.sleep(1.5)
    return data

result = fetch_data(t_start, t_stop, pair, TIMEFRAME, TIMEFRAME_S)
names = ['Date', 'Open', 'Close', 'High', 'Low', 'Volume']
df = pd.DataFrame(result, columns=names)
df.drop_duplicates(inplace=True)
df['Date'] = pd.to_datetime(df['Date'], unit='ms')
```

```
df.set_index('Date', inplace=True)
df.sort_index(inplace=True)
df.to_csv(f"{pair}_{TIMEFRAME}.csv")
```

```
No keys, only access to public API functions
2017-01-01 05:00:00 2017-02-11 21:00:00 steps left: 40991.0
2017-02-11 21:00:00 2017-03-25 13:00:00 steps left: 39991.0
2017-03-25 13:00:00 2017-05-06 05:00:00 steps left: 38991.0
2017-05-06 05:00:00 2017-06-16 21:00:00 steps left: 37991.0
2017-06-16 21:00:00 2017-07-28 13:00:00 steps left: 36991.0
2017-07-28 13:00:00 2017-09-08 05:00:00 steps left: 35991.0
2017-09-08 05:00:00 2017-10-19 21:00:00 steps left: 34991.0
2017-10-19 21:00:00 2017-11-30 13:00:00 steps left: 33991.0
2017-11-30 13:00:00 2018-01-11 05:00:00 steps left: 32991.0
2018-01-11 05:00:00 2018-02-21 21:00:00 steps left: 31991.0
2018-02-21 21:00:00 2018-04-04 13:00:00 steps left: 30991.0
2018-04-04 13:00:00 2018-05-16 05:00:00 steps left: 29991.0
2018-05-16 05:00:00 2018-06-26 21:00:00 steps left: 28991.0
2018-06-26 21:00:00 2018-08-07 13:00:00 steps left: 27991.0
2018-08-07 13:00:00 2018-09-18 05:00:00 steps left: 26991.0
2018-09-18 05:00:00 2018-10-29 21:00:00 steps left: 25991.0
2018-10-29 21:00:00 2018-12-10 13:00:00 steps left: 24991.0
2018-12-10 13:00:00 2019-01-21 05:00:00 steps left: 23991.0
2019-01-21 05:00:00 2019-03-03 21:00:00 steps left: 22991.0
2019-03-03 21:00:00 2019-04-14 13:00:00 steps left: 21991.0
2019-04-14 13:00:00 2019-05-26 05:00:00 steps left: 20991.0
2019-05-26 05:00:00 2019-07-06 21:00:00 steps left: 19991.0
2019-07-06 21:00:00 2019-08-17 13:00:00 steps left: 18991.0
2019-08-17 13:00:00 2019-09-28 05:00:00 steps left: 17991.0
2019-09-28 05:00:00 2019-11-08 21:00:00 steps left: 16991.0
2019-11-08 21:00:00 2019-12-20 13:00:00 steps left: 15991.0
2019-12-20 13:00:00 2020-01-31 05:00:00 steps left: 14991.0
2020-01-31 05:00:00 2020-03-12 21:00:00 steps left: 13991.0
2020-03-12 21:00:00 2020-04-23 13:00:00 steps left: 12991.0
2020-04-23 13:00:00 2020-06-04 05:00:00 steps left: 11991.0
2020-06-04 05:00:00 2020-07-15 21:00:00 steps left: 10991.0
2020-07-15 21:00:00 2020-08-26 13:00:00 steps left: 9991.0
2020-08-26 13:00:00 2020-10-07 05:00:00 steps left: 8991.0
2020-10-07 05:00:00 2020-11-17 21:00:00 steps left: 7991.0
2020-11-17 21:00:00 2020-12-29 13:00:00 steps left: 6991.0
2020-12-29 13:00:00 2021-02-09 05:00:00 steps left: 5991.0
2021-02-09 05:00:00 2021-03-22 21:00:00 steps left: 4991.0
2021-03-22 21:00:00 2021-05-03 13:00:00 steps left: 3991.0
2021-05-03 13:00:00 2021-06-14 05:00:00 steps left: 2991.0
2021-06-14 05:00:00 2021-07-25 21:00:00 steps left: 1991.0
2021-07-25 21:00:00 2021-09-05 04:00:00 steps left: 991.0
```

In [218…

```
import pandas_datareader as pdr
import datetime
import time

start = datetime.datetime(2017, 1, 1)
end = datetime.datetime(2021, 9, 3)
syms = ['DGS1MO', 'DGS3MO', 'DGS1', 'DGS3', 'DGS10']
df = pd.DataFrame()
for sym in syms:
      ts = pdr.fred.FredReader(sym, start=start, end=end)
      df1 = ts.read()
      df = pd.concat([df, df1], axis=1)
df
```

Out[218…

	DGS1MO	DGS3MO	DGS1	DGS3	DGS10

DATE	DGS1MO	DGS3MO	DGS1	DGS3	DGS10

DATE					
2017-01-02	NaN	NaN	NaN	NaN	NaN
2017-01-03	0.52	0.53	0.89	1.50	2.45
2017-01-04	0.49	0.53	0.87	1.50	2.46
2017-01-05	0.51	0.52	0.83	1.43	2.37
2017-01-06	0.50	0.53	0.85	1.50	2.42
...	...	...	...	...	...
2021-08-27	0.04	0.05	0.07	0.41	1.31
2021-08-30	0.04	0.05	0.08	0.40	1.29
2021-08-31	0.03	0.04	0.07	0.40	1.30
2021-09-01	0.04	0.05	0.07	0.42	1.31
2021-09-02	0.05	0.05	0.07	0.42	1.29

1219 rows × 5 columns

In [218…
```
InterestRates= df
```

In [218…
```
InterestRates=InterestRates.reset_index()
```

In [11]:
```
!pip install mplfinance
```

```
Requirement already satisfied: mplfinance in c:\programdata\anaconda3\lib\site-packages
(0.12.7a17)
Requirement already satisfied: pandas in c:\programdata\anaconda3\lib\site-packages (fro
m mplfinance) (1.1.3)
Requirement already satisfied: matplotlib in c:\programdata\anaconda3\lib\site-packages
(from mplfinance) (3.3.2)
Requirement already satisfied: pytz>=2017.2 in c:\programdata\anaconda3\lib\site-package
s (from pandas->mplfinance) (2020.1)
Requirement already satisfied: numpy>=1.15.4 in c:\programdata\anaconda3\lib\site-packag
es (from pandas->mplfinance) (1.19.2)
Requirement already satisfied: python-dateutil>=2.7.3 in c:\programdata\anaconda3\lib\si
te-packages (from pandas->mplfinance) (2.8.1)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.3 in c:\programdat
a\anaconda3\lib\site-packages (from matplotlib->mplfinance) (2.4.7)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\programdata\anaconda3\lib\site-pa
ckages (from matplotlib->mplfinance) (1.3.0)
Requirement already satisfied: certifi>=2020.06.20 in c:\programdata\anaconda3\lib\site-
packages (from matplotlib->mplfinance) (2020.6.20)
Requirement already satisfied: pillow>=6.2.0 in c:\programdata\anaconda3\lib\site-packag
es (from matplotlib->mplfinance) (8.0.1)
Requirement already satisfied: cycler>=0.10 in c:\programdata\anaconda3\lib\site-package
s (from matplotlib->mplfinance) (0.10.0)
Requirement already satisfied: six>=1.5 in c:\programdata\anaconda3\lib\site-packages (f
rom python-dateutil>=2.7.3->pandas->mplfinance) (1.15.0)
```

In [12]:
```
import pandas as pd
import pandas_datareader as pdr
```

```
import mplfinance as mpf
ts = pdr.av.time_series.AVTimeSeriesReader('SPY', api_key='A0FRPV75ZBATMOHH')
df = ts.read()
df.index = pd.to_datetime(df.index, format='%Y-%m-%d')
df
```

Out[12]:

	open	high	low	close	volume
2001-09-07	110.02	111.25	108.630	108.72	33133900
2001-09-10	107.70	110.35	107.700	110.05	23408700
2001-09-17	101.00	106.40	100.000	104.30	32388700
2001-09-18	104.33	105.30	103.360	104.05	22029200
2001-09-19	104.10	104.50	98.560	101.95	42771800
...	...	...	...	...	...
2021-08-26	448.61	448.86	446.160	446.26	57829572
2021-08-27	447.12	450.65	447.060	450.25	77235113
2021-08-30	450.97	453.07	450.710	452.23	48357355
2021-08-31	452.13	452.49	450.920	451.56	59300213
2021-09-01	452.56	453.11	451.545	451.80	48721378

5028 rows × 5 columns

In [13]:
```
SPY = df
```

In [14]:
```
SPY
```

Out[14]:

	open	high	low	close	volume
2001-09-07	110.02	111.25	108.630	108.72	33133900
2001-09-10	107.70	110.35	107.700	110.05	23408700
2001-09-17	101.00	106.40	100.000	104.30	32388700
2001-09-18	104.33	105.30	103.360	104.05	22029200
2001-09-19	104.10	104.50	98.560	101.95	42771800
...	...	...	...	...	...
2021-08-26	448.61	448.86	446.160	446.26	57829572
2021-08-27	447.12	450.65	447.060	450.25	77235113
2021-08-30	450.97	453.07	450.710	452.23	48357355
2021-08-31	452.13	452.49	450.920	451.56	59300213
2021-09-01	452.56	453.11	451.545	451.80	48721378

5028 rows × 5 columns

In [15]:
```
SPY.columns = ['SPYopen','SPYhigh','SPYlow','SPYclose','SPYvolume']
```

In [ ]:

In [16]:
```
SPY['SPYChange_Close']=SPY.SPYclose.pct_change()
SPY['SPYChange_Open']=SPY.SPYopen.pct_change()
SPY['SPYChange_high']=SPY.SPYhigh.pct_change()
SPY['SPYChange_low']=SPY.SPYlow.pct_change()
```

In [346…

In [ ]:

In [425…
```
#download data files from webscraping - Manually add in the BUY/SELL flag in excel base
#rallied/sold off by 5%
DOWNLOAD_DIR = "C:/Users/Mary Jane/Downloads/"
filename1 = "BTCUSD_1h.csv"

#create a dataframe with the data from the CSV file
data = pd.read_csv(DOWNLOAD_DIR+filename1)
```

In [425…
```
data['JoinDate']= pd.to_datetime(data['JoinDate'])
```

In [425…
```
data
```

Out[425…

	JoinDate	Date	Buy/Sell	Open	Close	High	Low	Volume
0	2016-12-30	1/1/2017 6:00	Hold	963.90000	967.45	967.45	962.72000	105.664480
1	2016-12-30	1/1/2017 7:00	Buy	966.48000	965.81	967.41	965.80000	27.489011
2	2016-12-30	1/1/2017 8:00	Buy	965.81000	965.46	965.86	962.60000	149.617697
3	2016-12-30	1/1/2017 9:00	Buy	965.41000	965.98	966.57	965.37000	32.288503
4	2016-12-30	1/1/2017 10:00	Buy	965.99000	977.01	978.02	965.99000	1061.834778
...	...	...	...	...	...	...	...	...
40974	2021-09-03	9/5/2021 0:00	Sell	49899.00000	50157.00	50222.00	49800.00000	211.402855
40975	2021-09-03	9/5/2021 1:00	Sell	50160.00000	50091.00	50171.00	49934.00000	56.785138
40976	2021-09-03	9/5/2021 2:00	Sell	50110.00000	50104.00	50309.00	50057.00000	154.743230
40977	2021-09-03	9/5/2021 3:00	Sell	50112.00000	49700.00	50122.00	49693.00000	127.802066
40978	2021-09-03	9/5/2021 4:00	Sell	49697.56459	49842.00	49900.00	49676.59708	55.807240

40979 rows × 8 columns

In [425…
```
SPY
```

Out[425…

	level_0	index	SPYopen	SPYhigh	SPYlow	SPYclose	SPYvolume	SPYChange_Close	SPYChange_(
0	0	2001-09-07	110.02	111.25	108.630	108.72	33133900	NaN	
1	1	2001-09-10	107.70	110.35	107.700	110.05	23408700	0.012233	-0.02
2	2	2001-09-17	101.00	106.40	100.000	104.30	32388700	-0.052249	-0.06
3	3	2001-09-18	104.33	105.30	103.360	104.05	22029200	-0.002397	0.03
4	4	2001-09-19	104.10	104.50	98.560	101.95	42771800	-0.020183	-0.00
...	...	...	...	...	...	...	...	...	
5023	5023	2021-08-26	448.61	448.86	446.160	446.26	57829572	-0.005903	0.00
5024	5024	2021-08-27	447.12	450.65	447.060	450.25	77235113	0.008941	-0.00
5025	5025	2021-08-30	450.97	453.07	450.710	452.23	48357355	0.004398	0.00
5026	5026	2021-08-31	452.13	452.49	450.920	451.56	59300213	-0.001482	0.00
5027	5027	2021-09-01	452.56	453.11	451.545	451.80	48721378	0.000531	0.00

5028 rows × 11 columns

In [425…
```
X=data.merge(SPY, left_on='JoinDate',right_on='index',how='inner')
```

In [425…
```
X
```

Out[425…

	JoinDate	Date	Buy/Sell	Open	Close	High	Low	Volume	lev
0	2016-12-30	1/1/2017 6:00	Hold	963.90000	967.45	967.45000	962.72000	105.664480	3
1	2016-12-30	1/1/2017 7:00	Buy	966.48000	965.81	967.41000	965.80000	27.489011	3
2	2016-12-30	1/1/2017 8:00	Buy	965.81000	965.46	965.86000	962.60000	149.617697	3
3	2016-12-30	1/1/2017 9:00	Buy	965.41000	965.98	966.57000	965.37000	32.288503	3
4	2016-12-30	1/1/2017 10:00	Buy	965.99000	977.01	978.02000	965.99000	1061.834778	3
...	...	...	...	...	...	...	...	...	

	JoinDate	Date	Buy/Sell	Open	Close	High	Low	Volume	lev
39481	2021-09-01	9/1/2021 19:00	Hold	48680.00000	48179.00	48694.00000	48074.00000	301.031871	5
39482	2021-09-01	9/1/2021 20:00	Hold	48172.00000	48292.00	48339.90762	48131.00000	104.802860	5
39483	2021-09-01	9/1/2021 21:00	Hold	48291.30187	48446.00	48536.00000	48283.00000	77.683107	5
39484	2021-09-01	9/1/2021 22:00	Hold	48448.00000	48436.00	48775.98262	48318.00000	163.642656	5
39485	2021-09-01	9/1/2021 23:00	Hold	48429.00000	48823.00	48929.00000	48411.77483	165.588515	5

39486 rows × 19 columns

In [426…
```
InterestRates['RatesChange_1MO']=InterestRates.DGS1MO.pct_change(periods=5)
InterestRates['RatesChange_3MO']=InterestRates.DGS3MO.pct_change(periods=5)
InterestRates['RatesChange_1Y']=InterestRates.DGS1.pct_change(periods=5)
InterestRates['RatesChange_3Y']=InterestRates.DGS3.pct_change(periods=5)
InterestRates['RatesChange_10Y']=InterestRates.DGS10.pct_change(periods=5)
```

In [426…
```
Z=X.merge(InterestRates, left_on='JoinDate',right_on='DATE',how='inner')
```

In [426…
```
Z['t_minus1_High'] = Z.High.shift(+1)
Z['t_minus2_High'] = Z.High.shift(+2)
Z['t_minus3_High'] = Z.High.shift(+3)
Z['t_minus4_High'] = Z.High.shift(+4)
Z['t_minus5_High'] = Z.High.shift(+5)
Z['t_minus6_High'] = Z.High.shift(+6)
Z['t_minus7_High'] = Z.High.shift(+7)
Z['t_minus8_High'] = Z.High.shift(+8)
Z['t_minus9_High'] = Z.High.shift(+9)


Z['t_minus1_Low'] = Z.Low.shift(+1)
Z['t_minus2_Low'] = Z.Low.shift(+2)
Z['t_minus3_Low'] = Z.Low.shift(+3)
Z['t_minus4_Low'] = Z.Low.shift(+4)
Z['t_minus5_Low'] = Z.Low.shift(+5)
Z['t_minus6_Low'] = Z.Low.shift(+6)
Z['t_minus7_Low'] = Z.Low.shift(+7)
Z['t_minus8_Low'] = Z.Low.shift(+8)
Z['t_minus9_Low'] = Z.Low.shift(+9)
```

In [426…
```
list(Z.columns)
```

Out[426…
```
['JoinDate',
 'Date',
 'Buy/Sell',
 'Open',
 'Close',
 'High',
 'Low',
 'Volume',
```

```
'level_0',
'index',
'SPYopen',
'SPYhigh',
'SPYlow',
'SPYclose',
'SPYvolume',
'SPYChange_Close',
'SPYChange_Open',
'SPYChange_high',
'SPYChange_low',
'DATE',
'DGS1MO',
'DGS3MO',
'DGS1',
'DGS3',
'DGS10',
'RatesChange_1MO',
'RatesChange_3MO',
'RatesChange_1Y',
'RatesChange_3Y',
'RatesChange_10Y',
't_minus1_High',
't_minus2_High',
't_minus3_High',
't_minus4_High',
't_minus5_High',
't_minus6_High',
't_minus7_High',
't_minus8_High',
't_minus9_High',
't_minus1_Low',
't_minus2_Low',
't_minus3_Low',
't_minus4_Low',
't_minus5_Low',
't_minus6_Low',
't_minus7_Low',
't_minus8_Low',
't_minus9_Low']
```

In [ ]:

In [ ]:

In [426…

```
Z
```

Out[426…

	JoinDate	Date	Buy/Sell	Open	Close	High	Low	Volume	level_
0	2017-01-03	1/3/2017 0:00	Hold	1019.60000	1024.9	1029.70000	1019.20000	580.898814	385
1	2017-01-03	1/3/2017 1:00	Hold	1024.80000	1026.6	1028.80000	1023.10000	280.327532	385
2	2017-01-03	1/3/2017 2:00	Hold	1027.50000	1024.9	1028.30000	1016.20000	717.615930	385
3	2017-01-03	1/3/2017 3:00	Hold	1025.80000	1027.9	1031.00000	1025.80000	459.078353	385
4	2017-01-03	1/3/2017 4:00	Hold	1027.90000	1025.4	1029.70000	1024.60000	181.295589	385

	JoinDate	Date	Buy/Sell	Open	Close	High	Low	Volume	level_
...	...	...	...	...	...	...	...	...	
39463	2021-09-01	9/1/2021 19:00	Hold	48680.00000	48179.0	48694.00000	48074.00000	301.031871	502
39464	2021-09-01	9/1/2021 20:00	Hold	48172.00000	48292.0	48339.90762	48131.00000	104.802860	502
39465	2021-09-01	9/1/2021 21:00	Hold	48291.30187	48446.0	48536.00000	48283.00000	77.683107	502
39466	2021-09-01	9/1/2021 22:00	Hold	48448.00000	48436.0	48775.98262	48318.00000	163.642656	502
39467	2021-09-01	9/1/2021 23:00	Hold	48429.00000	48823.0	48929.00000	48411.77483	165.588515	502

39468 rows × 48 columns

In [426...
```
Z['24_TrailAvg'] = Z.Close.rolling(window=24).mean().shift(periods=1)
Z['48_TrailAvg'] = Z.Close.rolling(window=48).mean().shift(periods=1)
```

In [426...
```
Z['Trail_Diff']=Z['24_TrailAvg']-Z['48_TrailAvg']
```

In [426...
```
Z
```

Out[426...

	JoinDate	Date	Buy/Sell	Open	Close	High	Low	Volume	level_
0	2017-01-03	1/3/2017 0:00	Hold	1019.60000	1024.9	1029.70000	1019.20000	580.898814	385
1	2017-01-03	1/3/2017 1:00	Hold	1024.80000	1026.6	1028.80000	1023.10000	280.327532	385
2	2017-01-03	1/3/2017 2:00	Hold	1027.50000	1024.9	1028.30000	1016.20000	717.615930	385
3	2017-01-03	1/3/2017 3:00	Hold	1025.80000	1027.9	1031.00000	1025.80000	459.078353	385
4	2017-01-03	1/3/2017 4:00	Hold	1027.90000	1025.4	1029.70000	1024.60000	181.295589	385
...	...	...	...	...	...	...	...	...	
39463	2021-09-01	9/1/2021 19:00	Hold	48680.00000	48179.0	48694.00000	48074.00000	301.031871	502
39464	2021-09-01	9/1/2021 20:00	Hold	48172.00000	48292.0	48339.90762	48131.00000	104.802860	502
39465	2021-09-01	9/1/2021 21:00	Hold	48291.30187	48446.0	48536.00000	48283.00000	77.683107	502
39466	2021-09-01	9/1/2021 22:00	Hold	48448.00000	48436.0	48775.98262	48318.00000	163.642656	502
39467	2021-09-01	9/1/2021 23:00	Hold	48429.00000	48823.0	48929.00000	48411.77483	165.588515	502

39468 rows × 51 columns

In [426…
```
Z['prior_volume_-1']=(Z.Volume.shift(+1)/Z.Volume.shift(+1).rolling(24).std())/(Z.Volum
Z['prior_volume_-2']=(Z.Volume.shift(+2)/Z.Volume.shift(+2).rolling(24).std())/(Z.Volum
Z['prior_volume_-3']=(Z.Volume.shift(+3)/Z.Volume.shift(+3).rolling(24).std())/(Z.Volum
Z['prior_volume_-4']=(Z.Volume.shift(+4)/Z.Volume.shift(+4).rolling(24).std())/(Z.Volum
Z['prior_volume_-5']=(Z.Volume.shift(+5)/Z.Volume.shift(+5).rolling(24).std())/(Z.Volum
Z['prior_volume_-6']=(Z.Volume.shift(+6)/Z.Volume.shift(+6).rolling(24).std())/(Z.Volum
Z['prior_volume_-7']=(Z.Volume.shift(+7)/Z.Volume.shift(+7).rolling(24).std())/(Z.Volum
Z['prior_volume_-8']=(Z.Volume.shift(+8)/Z.Volume.shift(+8).rolling(24).std())/(Z.Volum
Z['prior_volume_-9']=(Z.Volume.shift(+9)/Z.Volume.shift(+9).rolling(24).std())/(Z.Volum
Z['prior_volume_-10']=(Z.Volume.shift(+10)/Z.Volume.shift(+10).rolling(24).std())/(Z.Vo
Z['prior_volume_-11']=(Z.Volume.shift(+11)/Z.Volume.shift(+11).rolling(24).std())/(Z.Vo
```

In [426…
```
from statistics import median

Z['ChangeInHigh']=Z.High.shift(+25)/Z.Close.shift(+1)
Z['ChangeInLow']=Z.Close.shift(+1)/Z.Low.shift(+25)
Z['TrailingStd']=Z.Close.shift(+1).rolling(25).std()/Z.Close.shift(+1).rolling(25).mean
Z['HighDifferential']=Z['ChangeInHigh']/Z['TrailingStd']
Z['LowDifferential']=Z['ChangeInLow']/Z['TrailingStd']

Z['Momentum']=(Z['LowDifferential']-Z['HighDifferential'])
Z['Momentum%']=Z['Momentum']/100
```

In [427…
```
Z
```

Out[427…

	JoinDate	Date	Buy/Sell	Open	Close	High	Low	Volume	level_
0	2017-01-03	1/3/2017 0:00	Hold	1019.60000	1024.9	1029.70000	1019.20000	580.898814	385
1	2017-01-03	1/3/2017 1:00	Hold	1024.80000	1026.6	1028.80000	1023.10000	280.327532	385
2	2017-01-03	1/3/2017 2:00	Hold	1027.50000	1024.9	1028.30000	1016.20000	717.615930	385
3	2017-01-03	1/3/2017 3:00	Hold	1025.80000	1027.9	1031.00000	1025.80000	459.078353	385
4	2017-01-03	1/3/2017 4:00	Hold	1027.90000	1025.4	1029.70000	1024.60000	181.295589	385
...	...	...	...	...	...	...	...	...	
39463	2021-09-01	9/1/2021 19:00	Hold	48680.00000	48179.0	48694.00000	48074.00000	301.031871	502
39464	2021-09-01	9/1/2021 20:00	Hold	48172.00000	48292.0	48339.90762	48131.00000	104.802860	502
39465	2021-09-01	9/1/2021 21:00	Hold	48291.30187	48446.0	48536.00000	48283.00000	77.683107	502
39466	2021-09-01	9/1/2021 22:00	Hold	48448.00000	48436.0	48775.98262	48318.00000	163.642656	502
39467	2021-09-01	9/1/2021 23:00	Hold	48429.00000	48823.0	48929.00000	48411.77483	165.588515	502

39468 rows × 69 columns

In [ ]:

In [ ]:

In [427…
```
Z=Z.drop(['High',
 'Low','Open'],axis=1)
```

In [ ]:

In [460…
```
data=Z
```

In [460…
```
#create new features to deepen the learning and memory from previous Bitcoin prices

data = pd.DataFrame(data)
data['t_minus1'] = data.Close.shift(+1)
data['t_minus2'] = data.Close.shift(+2)
data['t_minus3'] = data.Close.shift(+3)
data['t_minus4'] = data.Close.shift(+4)
data['t_minus5'] = data.Close.shift(+5)
data['t_minus6'] = data.Close.shift(+6)
data['t_minus7'] = data.Close.shift(+7)
data['t_minus8'] = data.Close.shift(+8)
data['t_minus9'] = data.Close.shift(+9)
data.head(10)
```

Out[460…

	JoinDate	Date	Buy/Sell	Close	Volume	level_0	index	SPYopen	SPYhigh	SPYlow	...	t_n
0	2017-01-03	1/3/2017 0:00	Hold	1024.9	580.898814	3853	2017-01-03	225.04	225.83	223.88	...	
1	2017-01-03	1/3/2017 1:00	Hold	1026.6	280.327532	3853	2017-01-03	225.04	225.83	223.88	...	
2	2017-01-03	1/3/2017 2:00	Hold	1024.9	717.615930	3853	2017-01-03	225.04	225.83	223.88	...	
3	2017-01-03	1/3/2017 3:00	Hold	1027.9	459.078353	3853	2017-01-03	225.04	225.83	223.88	...	
4	2017-01-03	1/3/2017 4:00	Hold	1025.4	181.295589	3853	2017-01-03	225.04	225.83	223.88	...	
5	2017-01-03	1/3/2017 5:00	Hold	1025.1	284.891733	3853	2017-01-03	225.04	225.83	223.88	...	
6	2017-01-03	1/3/2017 6:00	Hold	1021.1	376.513529	3853	2017-01-03	225.04	225.83	223.88	...	
7	2017-01-03	1/3/2017 7:00	Hold	1021.2	610.489164	3853	2017-01-03	225.04	225.83	223.88	...	
8	2017-01-03	1/3/2017 8:00	Hold	1018.1	435.108331	3853	2017-01-03	225.04	225.83	223.88	...	
9	2017-01-03	1/3/2017 9:00	Hold	1016.0	697.653988	3853	2017-01-03	225.04	225.83	223.88	...	

10 rows × 84 columns

In [460…
```
data['t_minus1_Change'] = data.t_minus1.pct_change(periods=5)
data['t_minus2_Change'] = data.t_minus2.pct_change(periods=5)
data['t_minus3_Change'] = data.t_minus3.pct_change(periods=5)
data['t_minus4_Change'] = data.t_minus4.pct_change(periods=5)
data['t_minus5_Change'] = data.t_minus5.pct_change(periods=5)
data['t_minus6_Change'] = data.t_minus6.pct_change(periods=5)
data['t_minus7_Change'] = data.t_minus7.pct_change(periods=5)
data['t_minus8_Change'] = data.t_minus8.pct_change(periods=5)
data['t_minus9_Change'] = data.t_minus9.pct_change(periods=5)
```

In [460…
```
data=data.drop('JoinDate',axis=1)
```

In [460…
```
data=data.dropna()
```

In [460…
```
list(data.columns)
```

Out[460…
```
['Date',
 'Buy/Sell',
 'Close',
 'Volume',
 'level_0',
 'index',
 'SPYopen',
 'SPYhigh',
 'SPYlow',
 'SPYclose',
 'SPYvolume',
 'SPYChange_Close',
 'SPYChange_Open',
 'SPYChange_high',
 'SPYChange_low',
 'DATE',
 'DGS1MO',
 'DGS3MO',
 'DGS1',
 'DGS3',
 'DGS10',
 'RatesChange_1MO',
 'RatesChange_3MO',
 'RatesChange_1Y',
 'RatesChange_3Y',
 'RatesChange_10Y',
 't_minus1_High',
 't_minus2_High',
 't_minus3_High',
 't_minus4_High',
 't_minus5_High',
 't_minus6_High',
 't_minus7_High',
 't_minus8_High',
 't_minus9_High',
 't_minus1_Low',
 't_minus2_Low',
 't_minus3_Low',
 't_minus4_Low',
 't_minus5_Low',
 't_minus6_Low',
```

```
't_minus7_Low',
't_minus8_Low',
't_minus9_Low',
'24_TrailAvg',
'48_TrailAvg',
'Trail_Diff',
'prior_volume_-1',
'prior_volume_-2',
'prior_volume_-3',
'prior_volume_-4',
'prior_volume_-5',
'prior_volume_-6',
'prior_volume_-7',
'prior_volume_-8',
'prior_volume_-9',
'prior_volume_-10',
'prior_volume_-11',
'ChangeInHigh',
'ChangeInLow',
'TrailingStd',
'HighDifferential',
'LowDifferential',
'Momentum',
'Momentum%',
't_minus1',
't_minus2',
't_minus3',
't_minus4',
't_minus5',
't_minus6',
't_minus7',
't_minus8',
't_minus9',
't_minus1_Change',
't_minus2_Change',
't_minus3_Change',
't_minus4_Change',
't_minus5_Change',
't_minus6_Change',
't_minus7_Change',
't_minus8_Change',
't_minus9_Change']
```

In [460…

```
data
```

Out[460…

	Date	Buy/Sell	Close	Volume	level_0	index	SPYopen	SPYhigh	SPYlow	SPYclose
168	1/10/2017 0:00	Hold	909.98	492.243500	3858	2017-01-10	226.48	227.45	226.010	226.46
169	1/10/2017 1:00	Hold	911.51	1405.295562	3858	2017-01-10	226.48	227.45	226.010	226.46
170	1/10/2017 2:00	Hold	911.84	246.744073	3858	2017-01-10	226.48	227.45	226.010	226.46
171	1/10/2017 3:00	Hold	908.11	744.208144	3858	2017-01-10	226.48	227.45	226.010	226.46
172	1/10/2017 4:00	Hold	903.72	445.670476	3858	2017-01-10	226.48	227.45	226.010	226.46
...	...	...	...	...	...	...	...	...	...	...

	Date	Buy/Sell	Close	Volume	level_0	index	SPYopen	SPYhigh	SPYlow	SPYclose
39463	9/1/2021 19:00	Hold	48179.00	301.031871	5027	2021-09-01	452.56	453.11	451.545	451.8(
39464	9/1/2021 20:00	Hold	48292.00	104.802860	5027	2021-09-01	452.56	453.11	451.545	451.8(
39465	9/1/2021 21:00	Hold	48446.00	77.683107	5027	2021-09-01	452.56	453.11	451.545	451.8(
39466	9/1/2021 22:00	Hold	48436.00	163.642656	5027	2021-09-01	452.56	453.11	451.545	451.8(
39467	9/1/2021 23:00	Hold	48823.00	165.588515	5027	2021-09-01	452.56	453.11	451.545	451.8(

39084 rows × 83 columns

In [461…
```
data['Date']=pd.to_datetime(data['Date'])
```

In [461…
```
data=data[(data['Date']>pd.Timestamp(2020,6,1))]
```

In [461…
```
#create dataX to exclude September 2021 data from training
dataX=data
```

In [461…
```
dataX
```

Out[461…

	Date	Buy/Sell	Close	Volume	level_0	index	SPYopen	SPYhigh	SPYlow	SPYcl
28912	2020-06-01 01:00:00	Buy	9545.400000	166.126771	4710	2020-06-01	303.62	306.21	303.060	305
28913	2020-06-01 02:00:00	Buy	9531.000000	189.274729	4710	2020-06-01	303.62	306.21	303.060	305
28914	2020-06-01 03:00:00	Buy	9550.000000	75.824388	4710	2020-06-01	303.62	306.21	303.060	305
28915	2020-06-01 04:00:00	Buy	9550.200000	298.844616	4710	2020-06-01	303.62	306.21	303.060	305
28916	2020-06-01 05:00:00	Buy	9527.135459	1153.954144	4710	2020-06-01	303.62	306.21	303.060	305
...	...	...	...	...	...	...	...	...	...	
39463	2021-09-01 19:00:00	Hold	48179.000000	301.031871	5027	2021-09-01	452.56	453.11	451.545	451
39464	2021-09-01 20:00:00	Hold	48292.000000	104.802860	5027	2021-09-01	452.56	453.11	451.545	451

	Date	Buy/Sell	Close	Volume	level_0	index	SPYopen	SPYhigh	SPYlow	SPYcl
39465	2021-09-01 21:00:00	Hold	48446.000000	77.683107	5027	2021-09-01	452.56	453.11	451.545	451
39466	2021-09-01 22:00:00	Hold	48436.000000	163.642656	5027	2021-09-01	452.56	453.11	451.545	451
39467	2021-09-01 23:00:00	Hold	48823.000000	165.588515	5027	2021-09-01	452.56	453.11	451.545	451

10460 rows × 83 columns

In [461…
```
dataX=dataX[(dataX['Date']>pd.Timestamp(2021,8,31))]
```

In [461…
```
data=data[(data['Date']<pd.Timestamp(2021,8,30))]
```

In [461…
```
data
```

Out[461…

	Date	Buy/Sell	Close	Volume	level_0	index	SPYopen	SPYhigh	SPYlow	SPYcl
28912	2020-06-01 01:00:00	Buy	9545.400000	166.126771	4710	2020-06-01	303.62	306.21	303.06	305
28913	2020-06-01 02:00:00	Buy	9531.000000	189.274729	4710	2020-06-01	303.62	306.21	303.06	305
28914	2020-06-01 03:00:00	Buy	9550.000000	75.824388	4710	2020-06-01	303.62	306.21	303.06	305
28915	2020-06-01 04:00:00	Buy	9550.200000	298.844616	4710	2020-06-01	303.62	306.21	303.06	305
28916	2020-06-01 05:00:00	Buy	9527.135459	1153.954144	4710	2020-06-01	303.62	306.21	303.06	305
...	...	...	...	...	...	...	...	...	...	...
39391	2021-08-29 19:00:00	Hold	48820.000000	57.145003	5024	2021-08-27	447.12	450.65	447.06	450
39392	2021-08-29 20:00:00	Hold	48925.000000	100.357132	5024	2021-08-27	447.12	450.65	447.06	450
39393	2021-08-29 21:00:00	Hold	48909.000000	240.609178	5024	2021-08-27	447.12	450.65	447.06	450

	Date	Buy/Sell	Close	Volume	level_0	index	SPYopen	SPYhigh	SPYlow	SPYcl
39394	2021-08-29 22:00:00	Hold	49175.000000	324.390294	5024	2021-08-27	447.12	450.65	447.06	45(
39395	2021-08-29 23:00:00	Sell	48800.000000	120.135427	5024	2021-08-27	447.12	450.65	447.06	45(

10388 rows × 83 columns

In [461…
```
data=data.dropna()
dataX=dataX.dropna()
```

In [461…
```
numpy=data.to_numpy()
```

In [461…
```
numpy
```

Out[461…
```
array([[Timestamp('2020-06-01 01:00:00'), 'Buy', 9545.4, ...,
        -0.006605752873005466, -0.007393252091099867,
        -0.006021589326864807],
       [Timestamp('2020-06-01 02:00:00'), 'Buy', 9531.0, ...,
        0.005491082271196213, -0.006605752873005466,
        -0.007393252091099867],
       [Timestamp('2020-06-01 03:00:00'), 'Buy', 9550.0, ...,
        0.003595890028057269, 0.005491082271196213,
        -0.006605752873005466],
       ...,
       [Timestamp('2021-08-29 21:00:00'), 'Hold', 48909.0, ...,
        -0.007262926156821559, -0.0036985780241977073,
        -0.0026384136537906944],
       [Timestamp('2021-08-29 22:00:00'), 'Hold', 49175.0, ...,
        -0.002428783138481827, -0.007262926156821559,
        -0.0036985780241977073],
       [Timestamp('2021-08-29 23:00:00'), 'Sell', 48800.0, ...,
        0.009580689311932344, -0.002428783138481827,
        -0.007262926156821559]], dtype=object)
```

In [462…
```
numpy[:,num_features-1]
```

Out[462…
```
array([0.24309156249728403, -7.956908437503444, -15.697270395836313, ...,
       -142.62892166669917, -143.3580883333634, -128.52475500002765],
      dtype=object)
```

In [462…
```
from sklearn.compose import ColumnTransformer


cat_attribs = ['Buy/Sell']

full_pipeline = ColumnTransformer( [

    ('cat', OneHotEncoder(sparse=False), cat_attribs)
])


Y= full_pipeline.fit_transform(data)
YX= full_pipeline.transform(dataX)
```

In [462…
```
data=data.drop('level_0',axis=1)
dataX=dataX.drop('level_0',axis=1)
```

In [462…
```
data=data.reset_index()
dataX=dataX.reset_index()
```

In [462…
```
Y=pd.DataFrame(Y)
YX=pd.DataFrame(YX)
data=data.merge(Y,left_index=True,right_index=True,how='left')
dataX=data.merge(YX,left_index=True,right_index=True,how='left')
```

In [462…
```
data
```

Out[462…

	level_0	Date	Buy/Sell	Close	Volume	index	SPYopen	SPYhigh	SPYlow	SPYcl
0	28912	2020-06-01 01:00:00	Buy	9545.400000	166.126771	2020-06-01	303.62	306.21	303.06	305
1	28913	2020-06-01 02:00:00	Buy	9531.000000	189.274729	2020-06-01	303.62	306.21	303.06	305
2	28914	2020-06-01 03:00:00	Buy	9550.000000	75.824388	2020-06-01	303.62	306.21	303.06	305
3	28915	2020-06-01 04:00:00	Buy	9550.200000	298.844616	2020-06-01	303.62	306.21	303.06	305
4	28916	2020-06-01 05:00:00	Buy	9527.135459	1153.954144	2020-06-01	303.62	306.21	303.06	305
...	...	...	...	...	...	...	...	...	...	
10383	39391	2021-08-29 19:00:00	Hold	48820.000000	57.145003	2021-08-27	447.12	450.65	447.06	450
10384	39392	2021-08-29 20:00:00	Hold	48925.000000	100.357132	2021-08-27	447.12	450.65	447.06	450
10385	39393	2021-08-29 21:00:00	Hold	48909.000000	240.609178	2021-08-27	447.12	450.65	447.06	450
10386	39394	2021-08-29 22:00:00	Hold	49175.000000	324.390294	2021-08-27	447.12	450.65	447.06	450
10387	39395	2021-08-29 23:00:00	Sell	48800.000000	120.135427	2021-08-27	447.12	450.65	447.06	450

10388 rows × 86 columns

In [462…
```
data=data.set_index('Date')
dataX=dataX.set_index('Date')
```

In [ ]:
```
```

In [462…
```
data=data.drop('index',axis=1)
dataX=dataX.drop('index',axis=1)
```

In [462…
```
list(data.columns)
```

Out[462…
```
['level_0',
 'Buy/Sell',
 'Close',
 'Volume',
 'SPYopen',
 'SPYhigh',
 'SPYlow',
 'SPYclose',
 'SPYvolume',
 'SPYChange_Close',
 'SPYChange_Open',
 'SPYChange_high',
 'SPYChange_low',
 'DATE',
 'DGS1MO',
 'DGS3MO',
 'DGS1',
 'DGS3',
 'DGS10',
 'RatesChange_1MO',
 'RatesChange_3MO',
 'RatesChange_1Y',
 'RatesChange_3Y',
 'RatesChange_10Y',
 't_minus1_High',
 't_minus2_High',
 't_minus3_High',
 't_minus4_High',
 't_minus5_High',
 't_minus6_High',
 't_minus7_High',
 't_minus8_High',
 't_minus9_High',
 't_minus1_Low',
 't_minus2_Low',
 't_minus3_Low',
 't_minus4_Low',
 't_minus5_Low',
 't_minus6_Low',
 't_minus7_Low',
 't_minus8_Low',
 't_minus9_Low',
 '24_TrailAvg',
 '48_TrailAvg',
 'Trail_Diff',
 'prior_volume_-1',
 'prior_volume_-2',
 'prior_volume_-3',
 'prior_volume_-4',
 'prior_volume_-5',
 'prior_volume_-6',
 'prior_volume_-7',
 'prior_volume_-8',
```

```
'prior_volume_-9',
'prior_volume_-10',
'prior_volume_-11',
'ChangeInHigh',
'ChangeInLow',
'TrailingStd',
'HighDifferential',
'LowDifferential',
'Momentum',
'Momentum%',
't_minus1',
't_minus2',
't_minus3',
't_minus4',
't_minus5',
't_minus6',
't_minus7',
't_minus8',
't_minus9',
't_minus1_Change',
't_minus2_Change',
't_minus3_Change',
't_minus4_Change',
't_minus5_Change',
't_minus6_Change',
't_minus7_Change',
't_minus8_Change',
't_minus9_Change',
0,
1,
2]
```

In [462…
```
data=data.drop('DATE',axis=1)
dataX=dataX.drop('DATE',axis=1)
```

In [463…
```
import pandas as pd

def clean_dataset(df):
    assert isinstance(df, pd.DataFrame), "df needs to be a pd.DataFrame"
    df.dropna(inplace=True)
    indices_to_keep = ~df.isin([np.nan, np.inf, -np.inf]).any(1)
    return df[indices_to_keep].astype(np.float64)
```

In [463…
```
data=data.drop('Buy/Sell',axis=1)
dataX=dataX.drop('Buy/Sell',axis=1)
```

In [463…
```
df=clean_dataset(data)
df2=clean_dataset(dataX)
```

In [463…
```
df.to_csv("df.csv",index=True)
```

In [463…
```
data=df
dataX=df2
```

In [463…
```
data=data.drop('level_0',axis=1)
dataX=dataX.drop('level_0',axis=1)
```

In [463…
```
data
```

Out[463…

Date	Close	Volume	SPYopen	SPYhigh	SPYlow	SPYclose	SPYvolume	SPYChange_Clos
2020-06-01 01:00:00	9545.400000	166.126771	303.62	306.21	303.06	305.55	55696013.0	0.00404
2020-06-01 02:00:00	9531.000000	189.274729	303.62	306.21	303.06	305.55	55696013.0	0.00404
2020-06-01 03:00:00	9550.000000	75.824388	303.62	306.21	303.06	305.55	55696013.0	0.00404
2020-06-01 04:00:00	9550.200000	298.844616	303.62	306.21	303.06	305.55	55696013.0	0.00404
2020-06-01 05:00:00	9527.135459	1153.954144	303.62	306.21	303.06	305.55	55696013.0	0.00404
...	...	...	...	...	...	...	...	
2021-08-29 19:00:00	48820.000000	57.145003	447.12	450.65	447.06	450.25	77235113.0	0.00894
2021-08-29 20:00:00	48925.000000	100.357132	447.12	450.65	447.06	450.25	77235113.0	0.00894
2021-08-29 21:00:00	48909.000000	240.609178	447.12	450.65	447.06	450.25	77235113.0	0.00894
2021-08-29 22:00:00	49175.000000	324.390294	447.12	450.65	447.06	450.25	77235113.0	0.00894
2021-08-29 23:00:00	48800.000000	120.135427	447.12	450.65	447.06	450.25	77235113.0	0.00894

10172 rows × 81 columns

In [463…

```
#scale the data using MinMaxScaler
#challenge here is that we need to preserve the scaler used on the labeled data for fut
#as well as understanding how well the model is performing

#scaler used for features
scalerW = MinMaxScaler()
scalerX=MinMaxScaler()
scalerY= MinMaxScaler()

scalerZ= MinMaxScaler()


SCALER= MinMaxScaler()
```

```
scaled_data = np.concatenate([scalerW.fit_transform(data[[


't_minus1',
 't_minus2',
 't_minus3',
 't_minus4',
 't_minus5',
 't_minus6',
 't_minus7',
 't_minus8',
 't_minus9',
 't_minus1_High',
 't_minus2_High',
 't_minus3_High',
 't_minus4_High',
 't_minus5_High',
 't_minus6_High',
 't_minus7_High',
 't_minus8_High',
 't_minus9_High',
 't_minus1_Low',
 't_minus2_Low',
 't_minus3_Low',
 't_minus4_Low',
 't_minus5_Low',
 't_minus6_Low',
 't_minus7_Low',
 't_minus8_Low',
 't_minus9_Low',
'24_TrailAvg',
'48_TrailAvg',
 'Trail_Diff'
]]),


                              scalerX.fit_transform(data[[

'prior_volume_-1',
'prior_volume_-2',
'prior_volume_-3',
'prior_volume_-4',
'prior_volume_-5',
'prior_volume_-6',
'prior_volume_-7',
'prior_volume_-8',
'prior_volume_-9',
'prior_volume_-10',
'prior_volume_-11'
 ]]),

                              scalerY.fit_transform(data[[
'HighDifferential',
'LowDifferential',
'Momentum%']]),
```

```
                                scalerZ.fit_transform(data[[
 'RatesChange_1MO',
 'RatesChange_3MO',
 'RatesChange_1Y',
 'RatesChange_3Y',
 'RatesChange_10Y',
 't_minus1_Change',
 't_minus2_Change',
 't_minus3_Change',
 't_minus4_Change',
 't_minus5_Change',
 't_minus6_Change',
 't_minus7_Change',
 't_minus8_Change',
 't_minus9_Change'

 ]]),
                                   data[[0,1,2]] ], axis = 1)
```

In [ ]:

In [463…
```
#Look at the shape of our scaled_data
scaled_data.shape
```

Out[463… (10172, 61)

In [463…
```
#take a look at the first 5 entries that will be the "features" for this model
scaled_data[0:5]
```

Out[463…
```
array([[0.00968716, 0.00901647, 0.00908838, 0.00924656, 0.01015955,
        0.01028913, 0.00981638, 0.00999434, 0.01041676, 0.00919647,
        0.00883658, 0.00952056, 0.00957249, 0.00970259, 0.00992344,
        0.00940776, 0.00992166, 0.00983212, 0.01021846, 0.00987302,
        0.01003889, 0.0102228 , 0.01156961, 0.01139472, 0.0110792 ,
        0.01151913, 0.01091874, 0.00920289, 0.00850676, 0.48476711,
        0.00769166, 0.0186542 , 0.01761679, 0.02910963, 0.00333754,
        0.00828439, 0.0079986 , 0.01221306, 0.01275603, 0.03117617,
        0.03062205, 0.17586188, 0.1675589 , 0.19254513, 0.26666667,
        0.52380952, 0.33333333, 0.27862209, 0.35460993, 0.45070499,
        0.44707068, 0.4451568 , 0.440403  , 0.47281116, 0.47870123,
        0.44110546, 0.43865799, 0.44292099, 1.        , 0.        ,
        0.        ],
       [0.01071156, 0.00968716, 0.00901647, 0.00908838, 0.00924656,
        0.01015955, 0.01028913, 0.00981638, 0.00999434, 0.01047492,
        0.00919647, 0.00883658, 0.00952056, 0.00957249, 0.00970259,
        0.00992344, 0.00940776, 0.00992166, 0.01096136, 0.01021846,
        0.00987302, 0.01003889, 0.0102228 , 0.01156961, 0.01139472,
        0.0110792 , 0.01151913, 0.00911695, 0.0085717 , 0.48350565,
        0.01076908, 0.00769166, 0.0186542 , 0.01761679, 0.02910963,
        0.00333754, 0.00828439, 0.0079986 , 0.01221306, 0.01275603,
        0.03117617, 0.19284348, 0.18704006, 0.28361382, 0.26666667,
        0.52380952, 0.33333333, 0.27862209, 0.35460993, 0.47166629,
        0.45070499, 0.44707068, 0.4451568 , 0.440403  , 0.47281116,
        0.47870123, 0.44110546, 0.43865799, 1.        , 0.        ,
```

```
        0.        ],
       [0.01045271, 0.01071156, 0.00968716, 0.00901647, 0.00908838,
        0.00924656, 0.01015955, 0.01028913, 0.00981638, 0.0103843 ,
        0.01047492, 0.00919647, 0.00883658, 0.00952056, 0.00957249,
        0.00970259, 0.00992344, 0.00940776, 0.01206362, 0.01096136,
        0.01021846, 0.00987302, 0.01003889, 0.0102228 , 0.01156961,
        0.01139472, 0.0110792 , 0.00903167, 0.00862883, 0.48231489,
        0.01224563, 0.01076908, 0.00769166, 0.0186542 , 0.01761679,
        0.02910963, 0.00333754, 0.00828439, 0.0079986 , 0.01221306,
        0.01275603, 0.21690128, 0.2109973 , 0.27497934, 0.26666667,
        0.52380952, 0.33333333, 0.27862209, 0.35460993, 0.48367082,
        0.47166629, 0.45070499, 0.44707068, 0.4451568 , 0.440403  ,
        0.47281116, 0.47870123, 0.44110546, 1.        , 0.        ,
        0.        ],
       [0.01079424, 0.01045271, 0.01071156, 0.00968716, 0.00901647,
        0.00908838, 0.00924656, 0.01015955, 0.01028913, 0.01020455,
        0.0103843 , 0.01047492, 0.00919647, 0.00883658, 0.00952056,
        0.00957249, 0.00970259, 0.00992344, 0.01198408, 0.01206362,
        0.01096136, 0.01021846, 0.00987302, 0.01003889, 0.0102228 ,
        0.01156961, 0.01139472, 0.00900356, 0.00869077, 0.48156365,
        0.00500527, 0.01224563, 0.01076908, 0.00769166, 0.0186542 ,
        0.01761679, 0.02910963, 0.00333754, 0.00828439, 0.0079986 ,
        0.01221306, 0.24409777, 0.24042109, 0.35370516, 0.26666667,
        0.52380952, 0.33333333, 0.27862209, 0.35460993, 0.49282922,
        0.48367082, 0.47166629, 0.45070499, 0.44707068, 0.4451568 ,
        0.440403  , 0.47281116, 0.47870123, 1.        , 0.        ,
        0.        ],
       [0.01079784, 0.01079424, 0.01045271, 0.01071156, 0.00968716,
        0.00901647, 0.00908838, 0.00924656, 0.01015955, 0.01136841,
        0.01020455, 0.0103843 , 0.01047492, 0.00919647, 0.00883658,
        0.00952056, 0.00957249, 0.00970259, 0.01233587, 0.01198408,
        0.01206362, 0.01096136, 0.01021846, 0.00987302, 0.01003889,
        0.0102228 , 0.01156961, 0.00899569, 0.00870433, 0.48138482,
        0.02072149, 0.00500527, 0.01224563, 0.01076908, 0.00769166,
        0.0186542 , 0.01761679, 0.02910963, 0.00333754, 0.00828439,
        0.0079986 , 0.25045681, 0.2496616 , 0.46134299, 0.26666667,
        0.52380952, 0.33333333, 0.27862209, 0.35460993, 0.49422355,
        0.49282922, 0.48367082, 0.47166629, 0.45070499, 0.44707068,
        0.4451568 , 0.440403  , 0.47281116, 1.        , 0.        ,
        0.        ]])
```

In [464…
```
#We want to predict what Bitcoins price will be in 24 hours from now

Y = pd.DataFrame(scaled_data[:,-3:])
```

In [464…
```
Y.tail()
```

Out[464…

	0	1	2
10167	0.0	1.0	0.0
10168	0.0	1.0	0.0
10169	0.0	1.0	0.0
10170	0.0	1.0	0.0
10171	0.0	0.0	1.0

In [464…
```
Y.dropna(inplace=True)
```

In [464…
```
Y
```

Out[464…

	0	1	2
0	1.0	0.0	0.0
1	1.0	0.0	0.0
2	1.0	0.0	0.0
3	1.0	0.0	0.0
4	1.0	0.0	0.0
...	...	...	...
10167	0.0	1.0	0.0
10168	0.0	1.0	0.0
10169	0.0	1.0	0.0
10170	0.0	1.0	0.0
10171	0.0	0.0	1.0

10172 rows × 3 columns

In [464…
```
Y=Y.to_numpy()
```

In [464…
```
scaled_data.shape
```

Out[464… (10172, 61)

In [488…
```
# split into train and test sets
X_train, X_test, y_train, y_test = train_test_split(scaled_data,Y, test_size=0.2, rando
X_train, X_valid, y_train, y_valid = train_test_split(X_train,y_train, test_size=0.2, r
```

In [488…
```
STEPS = 3
```

In [488…
```
trainSeries = keras.preprocessing.sequence.TimeseriesGenerator(X_train, y_train, length
testSeries = keras.preprocessing.sequence.TimeseriesGenerator(X_test, y_test, length=ST
validSeries = keras.preprocessing.sequence.TimeseriesGenerator(X_valid,y_valid,length=S
```

In [488…
```
trainSeries[0][1].shape
```

Out[488… (50, 3)

In [ ]:

In [ ]:

In [488…
```
#Let's look at a few items in a timeseries to understand what LAG means
#and how data is structured for processing
for i in range(3):
    a,b= trainSeries[i]
    print('%s => %s' % (a, b))
```

```
[[[0.01 0.01 0.01 ... 1.   0.   0.  ]
  [0.01 0.01 0.01 ... 1.   0.   0.  ]
```

```
 [0.01 0.01 0.01 ... 1.   0.   0.  ]]

[[0.01 0.01 0.01 ... 1.   0.   0.  ]
 [0.01 0.01 0.01 ... 1.   0.   0.  ]
 [0.01 0.01 0.01 ... 1.   0.   0.  ]]

[[0.01 0.01 0.01 ... 1.   0.   0.  ]
 [0.01 0.01 0.01 ... 1.   0.   0.  ]
 [0.01 0.01 0.01 ... 1.   0.   0.  ]]

...

[[0.01 0.01 0.01 ... 0.   1.   0.  ]
 [0.01 0.01 0.01 ... 0.   1.   0.  ]
 [0.01 0.01 0.01 ... 0.   1.   0.  ]]

[[0.01 0.01 0.01 ... 0.   1.   0.  ]
 [0.01 0.01 0.01 ... 0.   1.   0.  ]
 [0.01 0.01 0.01 ... 0.   1.   0.  ]]

[[0.01 0.01 0.01 ... 0.   1.   0.  ]
 [0.01 0.01 0.01 ... 0.   1.   0.  ]
 [0.01 0.01 0.01 ... 0.   1.   0.  ]]] => [[1. 0. 0.]
[1. 0. 0.]
[1. 0. 0.]
[1. 0. 0.]
[1. 0. 0.]
[1. 0. 0.]
[1. 0. 0.]
[1. 0. 0.]
[1. 0. 0.]
[1. 0. 0.]
[1. 0. 0.]
[0. 1. 0.]
[0. 1. 0.]
[0. 1. 0.]
[0. 1. 0.]
[0. 1. 0.]
[0. 1. 0.]
[0. 1. 0.]
[0. 1. 0.]
[0. 1. 0.]
[0. 1. 0.]
[0. 0. 1.]
[0. 0. 1.]
[0. 0. 1.]
[0. 0. 1.]
[0. 0. 1.]
[0. 0. 1.]
[0. 0. 1.]
[0. 0. 1.]
[0. 0. 1.]
[0. 0. 1.]
[0. 0. 1.]
[0. 0. 1.]
[0. 0. 1.]
[0. 0. 1.]
[0. 0. 1.]
[0. 1. 0.]
[0. 1. 0.]
[0. 1. 0.]
[0. 1. 0.]
[0. 1. 0.]
[0. 1. 0.]
[0. 1. 0.]
[0. 1. 0.]
[0. 1. 0.]
```

```
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]]
[[[0.01 0.01 0.01 ... 0.   1.   0.  ]
  [0.01 0.01 0.01 ... 0.   1.   0.  ]
  [0.01 0.01 0.01 ... 0.   1.   0.  ]]

 [[0.01 0.01 0.01 ... 0.   1.   0.  ]
  [0.01 0.01 0.01 ... 0.   1.   0.  ]
  [0.01 0.01 0.01 ... 0.   1.   0.  ]]

 [[0.01 0.01 0.01 ... 0.   1.   0.  ]
  [0.01 0.01 0.01 ... 0.   1.   0.  ]
  [0.01 0.01 0.01 ... 0.   1.   0.  ]]

 ...

 [[0.02 0.02 0.02 ... 0.   1.   0.  ]
  [0.02 0.02 0.02 ... 0.   1.   0.  ]
  [0.02 0.02 0.02 ... 0.   1.   0.  ]]

 [[0.02 0.02 0.02 ... 0.   1.   0.  ]
  [0.02 0.02 0.02 ... 0.   1.   0.  ]
  [0.01 0.02 0.02 ... 0.   1.   0.  ]]

 [[0.02 0.02 0.02 ... 0.   1.   0.  ]
  [0.01 0.02 0.02 ... 0.   1.   0.  ]
  [0.02 0.01 0.02 ... 0.   1.   0.  ]]] => [[0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
```

```
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]]
[[[0.01 0.02 0.02 ... 0.   1.   0.  ]
  [0.02 0.01 0.02 ... 0.   1.   0.  ]
  [0.02 0.02 0.01 ... 0.   1.   0.  ]]

 [[0.02 0.01 0.02 ... 0.   1.   0.  ]
  [0.02 0.02 0.01 ... 0.   1.   0.  ]
  [0.01 0.02 0.02 ... 0.   1.   0.  ]]

 [[0.02 0.02 0.01 ... 0.   1.   0.  ]
  [0.01 0.02 0.02 ... 0.   1.   0.  ]
  [0.02 0.01 0.02 ... 0.   1.   0.  ]]

 ...

 [[0.01 0.01 0.01 ... 0.   1.   0.  ]
  [0.01 0.01 0.01 ... 0.   1.   0.  ]
  [0.01 0.01 0.01 ... 0.   1.   0.  ]]

 [[0.01 0.01 0.01 ... 0.   1.   0.  ]
  [0.01 0.01 0.01 ... 0.   1.   0.  ]
  [0.01 0.01 0.01 ... 0.   1.   0.  ]]

 [[0.01 0.01 0.01 ... 0.   1.   0.  ]
  [0.01 0.01 0.01 ... 0.   1.   0.  ]
  [0.01 0.01 0.01 ... 0.   1.   0.  ]]] => [[0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
```

```
[0. 1. 0.]
[0. 1. 0.]
[0. 1. 0.]
[0. 1. 0.]
[0. 1. 0.]
[0. 1. 0.]
[0. 1. 0.]
[0. 1. 0.]
[0. 1. 0.]
[0. 1. 0.]
[0. 1. 0.]
[0. 1. 0.]
[0. 1. 0.]
[0. 1. 0.]
[0. 1. 0.]
[0. 1. 0.]
[0. 1. 0.]
[0. 1. 0.]
[0. 1. 0.]
[0. 1. 0.]
[0. 1. 0.]
[0. 1. 0.]
[0. 1. 0.]
[0. 1. 0.]
[0. 1. 0.]]
```

LSTM

In [489…

```
#starting by setting random seeds and restarting keras blackend session
np.random.seed(42)
tf.random.set_seed(42)

#resets the Keras global state - helps avoid clutter from old models and layers, especi
keras.backend.clear_session()
```

In [489…

```
model = keras.models.Sequential([
    keras.layers.GRU(300, return_sequences=True,input_shape=(STEPS, 61)),
    keras.layers.GRU(360, return_sequences=True),
    keras.layers.GRU(480, return_sequences=True),
    keras.layers.Flatten(),
    keras.layers.Dense(3,activation = 'softmax')])
```

In [489…

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
gru (GRU)                    (None, 3, 300)            326700
_________________________________________________________________
gru_1 (GRU)                  (None, 3, 360)            714960
_________________________________________________________________
gru_2 (GRU)                  (None, 3, 480)            1212480
_________________________________________________________________
flatten (Flatten)            (None, 1440)              0
_________________________________________________________________
dense (Dense)                (None, 3)                 4323
```

```
=================================================================
Total params: 2,258,463
Trainable params: 2,258,463
Non-trainable params: 0
_________________________________________________________________
```

In [489…
```
early_stopping2 = keras.callbacks.EarlyStopping(monitor='val_loss',mode='min',min_delta
```

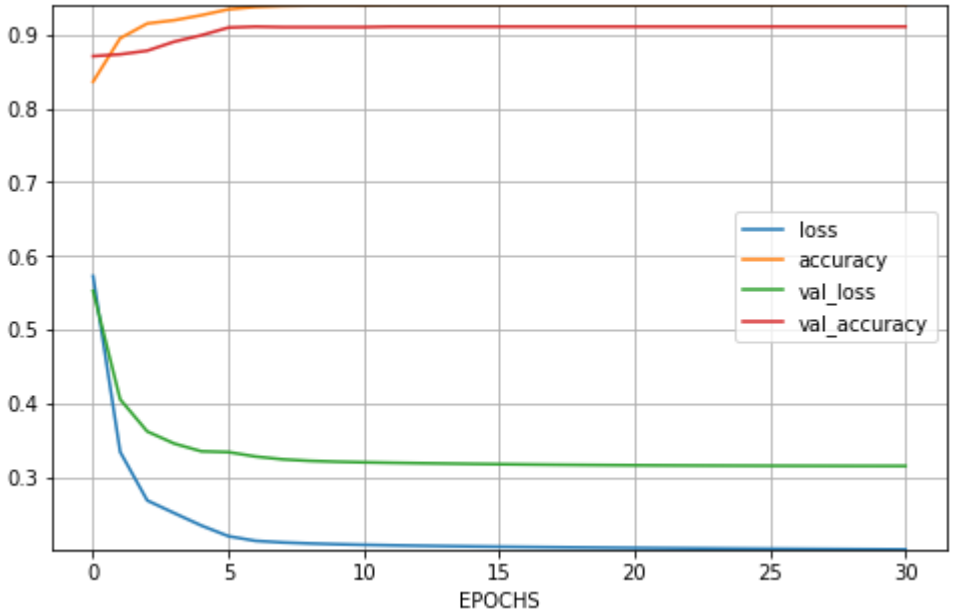
In [489…
```
#compile the model and use the new metric defined in the previous step
model.compile(loss='categorical_crossentropy', optimizer=keras.optimizers.Adam(lr=0.000

#fit the model
history = model.fit(trainSeries, epochs=100, shuffle=False,
                    validation_data=validSeries, callbacks=[early_stopping2])
```

```
Epoch 1/100
131/131 [==============================] - 16s 84ms/step - loss: 0.6341 - accuracy: 0.87
93 - val_loss: 0.5528 - val_accuracy: 0.8708
Epoch 2/100
131/131 [==============================] - 9s 68ms/step - loss: 0.2401 - accuracy: 0.910
0 - val_loss: 0.4059 - val_accuracy: 0.8732
Epoch 3/100
131/131 [==============================] - 9s 70ms/step - loss: 0.1804 - accuracy: 0.951
9 - val_loss: 0.3624 - val_accuracy: 0.8782
Epoch 4/100
131/131 [==============================] - 9s 70ms/step - loss: 0.1705 - accuracy: 0.952
0 - val_loss: 0.3459 - val_accuracy: 0.8905
Epoch 5/100
131/131 [==============================] - 9s 71ms/step - loss: 0.1596 - accuracy: 0.955
9 - val_loss: 0.3355 - val_accuracy: 0.8991
Epoch 6/100
131/131 [==============================] - 9s 70ms/step - loss: 0.1459 - accuracy: 0.961
3 - val_loss: 0.3344 - val_accuracy: 0.9095
Epoch 7/100
131/131 [==============================] - 9s 71ms/step - loss: 0.1371 - accuracy: 0.963
3 - val_loss: 0.3284 - val_accuracy: 0.9108
Epoch 8/100
131/131 [==============================] - 9s 72ms/step - loss: 0.1344 - accuracy: 0.964
1 - val_loss: 0.3246 - val_accuracy: 0.9102
Epoch 9/100
131/131 [==============================] - 9s 71ms/step - loss: 0.1324 - accuracy: 0.964
5 - val_loss: 0.3226 - val_accuracy: 0.9102
Epoch 10/100
131/131 [==============================] - 9s 72ms/step - loss: 0.1310 - accuracy: 0.965
3 - val_loss: 0.3214 - val_accuracy: 0.9102
Epoch 11/100
131/131 [==============================] - 9s 71ms/step - loss: 0.1299 - accuracy: 0.965
1 - val_loss: 0.3206 - val_accuracy: 0.9102
Epoch 12/100
131/131 [==============================] - 10s 73ms/step - loss: 0.1290 - accuracy: 0.96
56 - val_loss: 0.3200 - val_accuracy: 0.9108
Epoch 13/100
131/131 [==============================] - 9s 72ms/step - loss: 0.1283 - accuracy: 0.965
6 - val_loss: 0.3194 - val_accuracy: 0.9108
Epoch 14/100
131/131 [==============================] - 10s 75ms/step - loss: 0.1277 - accuracy: 0.96
58 - val_loss: 0.3189 - val_accuracy: 0.9108
Epoch 15/100
131/131 [==============================] - 10s 79ms/step - loss: 0.1271 - accuracy: 0.96
60 - val_loss: 0.3185 - val_accuracy: 0.9108
Epoch 16/100
131/131 [==============================] - 10s 74ms/step - loss: 0.1267 - accuracy: 0.96
62 - val_loss: 0.3181 - val_accuracy: 0.9108
Epoch 17/100
```

```
131/131 [==============================] - 10s 77ms/step - loss: 0.1262 - accuracy: 0.96
63 - val_loss: 0.3177 - val_accuracy: 0.9108
Epoch 18/100
131/131 [==============================] - 10s 73ms/step - loss: 0.1259 - accuracy: 0.96
63 - val_loss: 0.3174 - val_accuracy: 0.9108
Epoch 19/100
131/131 [==============================] - 9s 72ms/step - loss: 0.1256 - accuracy: 0.966
3 - val_loss: 0.3171 - val_accuracy: 0.9108
Epoch 20/100
131/131 [==============================] - 10s 74ms/step - loss: 0.1253 - accuracy: 0.96
63 - val_loss: 0.3168 - val_accuracy: 0.9108
Epoch 21/100
131/131 [==============================] - 10s 73ms/step - loss: 0.1250 - accuracy: 0.96
62 - val_loss: 0.3166 - val_accuracy: 0.9108
Epoch 22/100
131/131 [==============================] - 9s 72ms/step - loss: 0.1248 - accuracy: 0.966
2 - val_loss: 0.3164 - val_accuracy: 0.9108
Epoch 23/100
131/131 [==============================] - 9s 73ms/step - loss: 0.1246 - accuracy: 0.966
3 - val_loss: 0.3162 - val_accuracy: 0.9108
Epoch 24/100
131/131 [==============================] - 9s 72ms/step - loss: 0.1244 - accuracy: 0.966
3 - val_loss: 0.3161 - val_accuracy: 0.9108
Epoch 25/100
131/131 [==============================] - 10s 73ms/step - loss: 0.1243 - accuracy: 0.96
63 - val_loss: 0.3160 - val_accuracy: 0.9108
Epoch 26/100
131/131 [==============================] - 9s 72ms/step - loss: 0.1241 - accuracy: 0.966
3 - val_loss: 0.3159 - val_accuracy: 0.9108
Epoch 27/100
131/131 [==============================] - 10s 73ms/step - loss: 0.1239 - accuracy: 0.96
62 - val_loss: 0.3158 - val_accuracy: 0.9108
Epoch 28/100
131/131 [==============================] - 10s 75ms/step - loss: 0.1238 - accuracy: 0.96
63 - val_loss: 0.3158 - val_accuracy: 0.9108
Epoch 29/100
131/131 [==============================] - 9s 72ms/step - loss: 0.1237 - accuracy: 0.966
4 - val_loss: 0.3157 - val_accuracy: 0.9108
Epoch 30/100
131/131 [==============================] - 9s 72ms/step - loss: 0.1235 - accuracy: 0.966
4 - val_loss: 0.3157 - val_accuracy: 0.9108
Epoch 31/100
131/131 [==============================] - 9s 71ms/step - loss: 0.1234 - accuracy: 0.966
3 - val_loss: 0.3157 - val_accuracy: 0.9108
Epoch 00031: early stopping
```

In [489…

```
#plot the learning curve of the model
plot_learning_curve(history)
plt.show()
```



In [489…
```
y_pred = model.predict(testSeries)
```

In [489…
```
y_pred
```

Out[489…
```
array([[0.04, 0.94, 0.02],
       [0.04, 0.95, 0.02],
       [0.03, 0.95, 0.02],
       ...,
       [0.04, 0.94, 0.02],
       [0.04, 0.94, 0.02],
       [0.04, 0.93, 0.02]], dtype=float32)
```

In [489…
```
y_pred=pd.DataFrame(y_pred)
```

In [489…
```
 y_pred[0].max()
```

Out[489…
```
0.8890899419784546
```

In [490…
```
 y_pred[1].max()
```

Out[490…
```
0.9726846814155579
```

In [490…
```
 y_pred[2].max()
```

Out[490…
```
0.9645912051200867
```

In [490…
```
##Model Predicts at 90% on test data

model.evaluate(testSeries)
```

```
41/41 [==============================] - 1s 30ms/step - loss: 0.3366 - accuracy: 0.9001
```

Out[490…
```
[0.3365519642829895, 0.900098443031311]
```

In [490…
```
from sklearn.metrics import confusion_matrix
```

```
y_pred=y_pred.to_numpy()
```

In [490…

In [490…
```
y_test.shape
```

Out[490… (2035, 3)

In [490…
```
y_pred = np.append(y_pred, np.array([[1,0,0]]), axis=0)
y_pred = np.append(y_pred, np.array([[1,0,0]]), axis=0)
y_pred = np.append(y_pred, np.array([[1,0,0]]), axis=0)

y_pred.shape
```

Out[490… (2035, 3)

In [490…
```
classes=['Buy','Hold','Sell']
```

In [490…
```
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')

    print(cm)

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    fmt = '.2f' if normalize else 'd'
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], fmt),
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.ylabel('True label')
    plt.xlabel('Predicted label')
    plt.tight_layout()
```

In [490…
```
from sklearn.metrics import confusion_matrix


Actuals=y_test[:,:].argmax(axis=1)
y_pred2=y_pred[:,:].argmax(axis=1)
```

```
matrix = confusion_matrix(Actuals, y_pred2)
np.set_printoptions(precision=2)
```

In [491…

```
matrix
```

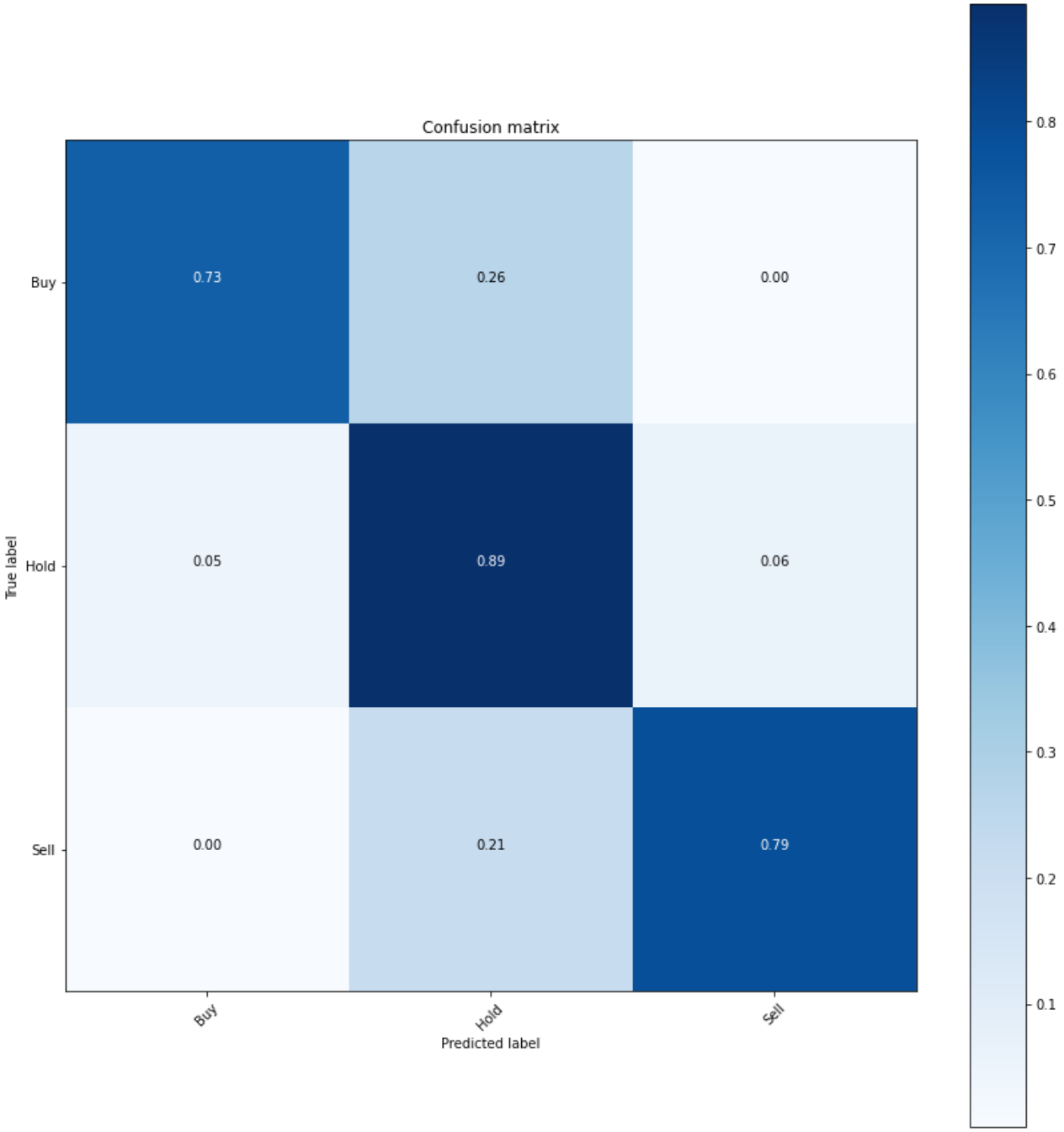
Out[491…

```
array([[ 170,   61,    1],
       [  64, 1252,   86],
       [   1,   85,  315]], dtype=int64)
```

In [491…

```
import itertools

plt.figure(figsize=(12,12))
foo = plot_confusion_matrix(matrix, classes=classes,normalize=True,
                      title='Confusion matrix')
```

```
Normalized confusion matrix
[[0.73 0.26 0.  ]
 [0.05 0.89 0.06]
 [0.   0.21 0.79]]
```


Confusion matrix
Buy
Hold
Sell
True label
0.73
0.26
0.00
0.05
0.89
0.06
0.00
0.21
0.79
Buy
Hold
Sell
Predicted label
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1

In [ ]: