```
# Set of helper functions for main algorithm

import numpy as np
import quandl
import pandas as pd
import pandas_talib as talib
from Bessie_feature import EWMA,BBANDS,STOK,STO,MACD,CCI,RSI,Chaikin
import datetime
from pandas_datareader import data as pdr
import fix_yahoo_finance as yf



# Adopted from: https://stackoverflow.com/questions/29721228/given-a-date-range-how-can-we-break-it-up-into-n-contiguous-sub-intervals
def date_range_new(start, end, prop, set_time = "None"):

    if set_time == "None":
        from datetime import datetime
        # start = datetime.strptime(start,"%b %d %Y")
        # end = datetime.strptime(end,"%b %d %Y")
        format = "%b %d %Y"
        start = start.strftime(format)
        end = end.strftime(format)
        diff = (end  - start ) * prop
        yield (start + diff).strftime("%b %d %Y")

    # If specific split time is included, return the specific time
    else:
        yield set_time

def date_range(start, end, prop, set_time = "None"):

    if set_time == "None":
        from datetime import datetime
        start = datetime.strptime(start,"%b %d %Y")
        end = datetime.strptime(end,"%b %d %Y")
        diff = (end  - start ) * prop
        yield (start + diff).strftime("%b %d %Y")

    # If specific split time is included, return the specific time
    else:
        yield set_time

def get_curr_date():

```

```
46      # Gets current date as month-day-year
47      now = datetime.datetime.now()
48      now = now.strftime('%m %d %Y')
49
50      # Reformats date with month as a string
51      time_arr = now.split(" ")
52      month = datetime.date(1900, int(time_arr[0]), 1).
    strftime('%b')
53      print month
54      time_arr[0] = month
55      curr_time = " ".join(time_arr)
56
57      return curr_time
58
59  def read_txt(fname):
60
61      # Opens the file
62      with open(fname) as f:
63          content = f.readlines()
64
65      # Reads the lines of the file and removes empty lines
66      content = [line.strip() for line in content] #Reads the
     lines of the file
67      # print content #['Algorithm Parameters', "Note: Not
    all stocks are supported in the Quandl 'Wiki' database
    .", '========================', 'Stocks to Forecast (Comma
    Separated Valid Tickers):', 'AAPL, ADBE, ABBV, LUK, MAC,
    MSFT, PNR, PPL, PSA, SLB', '', 'Forecast Period (Integer
    ):', '7']
68      # content = [s for s in content if s != ""] #removes
    empty lines
69      # print content #['Algorithm Parameters', "Note: Not
    all stocks are supported in the Quandl 'Wiki' database
    .", '========================', 'Stocks to Forecast (Comma
    Separated Valid Tickers):', 'AAPL, ADBE, ABBV, LUK, MAC,
    MSFT, PNR, PPL, PSA, SLB', 'Forecast Period (Integer):', '7
    ']
70
71      # Finds index of the "Ticker" input and gets the stock
    array
72      matching = [s for s in content if "Ticker" in s]
73      # print matching #print: ['Stocks to Forecast (Comma
    Separated Valid Tickers):']
74      tickers_index = content.index(matching[0])
75      # print matching[0] #Stocks to Forecast (Comma
    Separated Valid Tickers):
76      # print tickers_index #3
77      stocks = [ticker.strip() for ticker in content[
    tickers_index+1].split(',')]
78
```

```
 79     # Finds the index of the "Forecast Period" input and
    gets the forecast period
 80     matching = [s for s in content if "Forecast Period" in
     s]
 81     pred_period = content.index(matching[0])
 82     period = content[pred_period+1]
 83 
 84     return stocks, int(period)
 85 
 86 def exponential_smoothing(alpha, input_data):
 87 
 88     # Exponentially smooths input prices beginning from
    the most recent
 89     for i in reversed(range(0, len(input_data)-1)):
 90         input_data.iloc[i] = input_data.iloc[i+1]*(1-alpha
    ) + alpha * input_data.iloc[i]
 91 
 92 def create_shifted_orderbook(ticker, start_date, end_date,
     lag_period = 5, pred_period = 7):
 93 
 94     # Retrieve the Nifty data from Yahoo finance:
 95     format = '%Y-%m-%d'  # Formatting directives
 96     start = start_date.strftime(format)
 97     end = end_date.strftime(format)
 98 
 99     yf.pdr_override()  # <== that's all it takes :-)
100     stock_data = pdr.get_data_yahoo(ticker, start=start,
    end=end)
101 
102     # Creates stock lag
103     stock_data.dropna()
104     stock_lag = pd.DataFrame(data = stock_data, index=
    stock_data.index)
105 
106     stock_returns = pd.DataFrame()
107 
108     # Initializes dataframe values and smooths the closing
     price data
109     stock_data_smooth = stock_data['Adj Close']
110     exponential_smoothing(0.7, stock_data_smooth) #so the
    stock_data_smooth is smoothing
111 
112 
113     stock_lag["Close"] = stock_data_smooth #so, now the
    stock_lag["Close"] is derive from Adj Close + smoothing.
114 
115     # Sets lagging price data (previous days' price data
    as feature inputs)
116     for i in range(0, lag_period):
117         column_label = 'Lag{:d}'.format(i)
```

```
118         stock_lag[column_label] = stock_lag['Close'].shift
    (1+i)
119
120     # EMA- Momentum
121     ndays = 30
122     name_EWMA = 'EWMA_' + str(ndays)
123     stock_lag['EWMA_'] = EWMA(stock_lag,ndays )[name_EWMA]
124
125     # Bollinger Bands
126     aa = BBANDS(stock_lag, ndays=30)
127     stock_lag['upperband'] = aa['Upper BollingerBand']
128     stock_lag['lowerband'] = aa['Lower BollingerBand']
129
130     # StochK
131     n = 30
132     name_slowk = 'SO%k'
133     name_slowd = 'SO%d_' + str(n)
134     stock_lag['slowk'] = STOK(stock_lag)[name_slowk]
135     stock_lag['slowd'] = STO(stock_lag, n)[name_slowd]
136
137     # MACD- Momentum
138     n_fast = 12
139     n_slow = 26
140     name_macd = 'MACD_' + str(n_fast) + '_' + str(n_slow)
141     name_macdsignal = 'MACDsign_' + str(n_fast) + '_' +
    str(n_slow)
142     name_macdhist = 'MACDdiff_' + str(n_fast) + '_' + str(
    n_slow)
143     macd = MACD(stock_lag, n_fast, n_slow)[name_macd]
144     macdsignal = MACD(stock_lag, n_fast, n_slow)[
    name_macdsignal]
145     stock_lag['macdhist'] = MACD(stock_lag, n_fast, n_slow
    )[name_macdhist]
146
147     # CCI- Momentum
148     stock_lag['CCI'] = CCI(stock_lag, ndays = 30)["CCI"]
149
150     # Chaikin- Volume
151     stock_lag['Chaikin'] = Chaikin(stock_lag)['Chaikin']
152
153     stock_returns['Day Returns'] = stock_data['Adj Close']
    .pct_change() * 100
154     # Sets lagging percent change data
155     for i in range(0, lag_period):
156         column_label = 'Lag{:d}'.format(i)
157         stock_returns[column_label] = stock_lag[
    column_label].pct_change() * 100
158
159
160     # Remove NaN's from stock lag
```

```
161     print "shape of stock_lag before dropna: ",stock_lag.
    shape[0]
162     stock_lag = stock_lag.dropna(axis=0, how='any')
163     print "shape of stock_lag before dropna: ",stock_lag.
    shape[0]
164
165     print "shape of stock_returns before dropna: ",
    stock_returns.shape[0]
166     # Adjusts stock_return data to same length as
    stock_lag
167     stock_returns = stock_returns.tail(stock_lag.shape[0])
168     print "shape of stock_returns after dropna: ",
    stock_returns.shape[0]
169
170
171     # Determine stock movement direction and lagging
    movement
172     stock_movement = pd.DataFrame(index=stock_returns.
    index)
173     stock_movement['Movement_0'] = np.sign(stock_returns['
    Day Returns'])
174     stock_movement['Movement_0'][0] = 1
175     for i in range(0, pred_period):
176         column_label = 'Movement_{:d}'.format(i + 1)
177         stock_movement[column_label] = stock_movement['
    Movement_0'].shift(i + 1)
178
179     # Removes NaNs from 'stock_movement' and resizes '
    stocks_returns' and 'stock_lag' accordingly
180     print "shape of stock_movement before dropna: ",
    stock_movement.shape[0]
181     stock_movement = stock_movement.dropna(axis=0, how='
    any')
182     print "shape of stock_movement after dropna: ",
    stock_movement.shape[0]
183
184     stock_returns = stock_returns[stock_returns.index <=
    stock_movement.index[stock_movement.index.__len__() - 1]]
185     stock_returns = stock_returns.tail(stock_movement.
    shape[0])
186     stock_lag = stock_lag[stock_lag.index <=
    stock_movement.index[stock_movement.index.__len__() - 1]]
187     stock_lag = stock_lag.tail(stock_movement.shape[0])
188
189     return stock_data, stock_returns, stock_lag,
    stock_movement
190
```