Udacity AI for trading Project2

def get\_high\_lows\_lookback(high, low, lookback\_days):

"""

Get the highs and lows in a lookback window.

Parameters

----------

high : DataFrame

High price for each ticker and date

low : DataFrame

Low price for each ticker and date

lookback\_days : int

The number of days to look back

Returns

-------

lookback\_high : DataFrame

Lookback high price for each ticker and date

lookback\_low : DataFrame

Lookback low price for each ticker and date

"""

#TODO: Implement function

import copy

lookback\_high = copy.deepcopy(high)

lookback\_low = copy.deepcopy(low)

for i in range(len(lookback\_high)-1,-1,-1):

if i < lookback\_days:

for j in high.columns:

lookback\_high.iloc[i][j] = "NaN"

lookback\_low.iloc[i][j] = "NaN"

else:

for j in high.columns:

lookback\_high.iloc[i][j] = max(lookback\_high.iloc[i-lookback\_days:i][j] )

lookback\_low.iloc[i][j] = min(lookback\_low.iloc[i-lookback\_days:i][j] )

return lookback\_high, lookback\_low

project\_tests.test\_get\_high\_lows\_lookback(get\_high\_lows\_lookback)

OOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOO

00000000000000000000000000000000000000000000000000000000000000000000000

import copy

def get\_long\_short(close, lookback\_high, lookback\_low):

"""

Generate the signals long, short, and do nothing.

Parameters

----------

close : DataFrame

Close price for each ticker and date

lookback\_high : DataFrame

Lookback high price for each ticker and date

lookback\_low : DataFrame

Lookback low price for each ticker and date

Returns

-------

long\_short : DataFrame

The long, short, and do nothing signals for each ticker and date

"""

#TODO: Implement function

#ValueError: Can only compare identically-labeled DataFrame objects

# lookback\_high, lookback\_low = get\_high\_lows\_lookback(high, low, lookback\_days)

# print(close)

# LSCD BLX FFS

# 2005-07-09 25.67880000 35.13920000 34.05270000

# 2005-07-10 25.18840000 14.34530000 39.93730000

# 2005-07-11 78.28030000 34.38540000 23.29320000

# 2005-07-12 88.87250000 52.22300000 34.41070000

# print(lookback\_low)

# DQL VHS THDI

# 2007-06-13 nan nan nan

# 2007-06-14 34.17050000 92.45300000 58.51070000

# 2007-06-15 15.67180000 12.34530000 34.05270000

# 2007-06-16 27.18340000 12.34530000 23.29320000

# print(lookback\_high)

# DQL VHS THDI

# 2007-06-13 nan nan nan

# 2007-06-14 92.11310000 91.05430000 90.95720000

# 2007-06-15 35.44110000 34.17990000 34.02230000

# 2007-06-16 92.11310000 91.05430000 90.95720000

# if lookback\_low.any() > close.any():

# return -1

# elif lookback\_high.any() < close.any():

# return 1

# else:

# return 0

print( lookback\_high.index ) # Index([2011-05-09, 2011-05-10, 2011-05-11, 2011-05-12], dtype='object')

print( lookback\_high.columns) # Index(['UMAD', 'GJB', 'ONX'], dtype='object')

ans = copy.deepcopy(lookback\_high)

for i in range(len(close)):

for j in range(len(close.columns)) :

if lookback\_high.iloc[i][j] < close.iloc[i][j] :

ans.iloc[i][j] = 1

elif lookback\_low.iloc[i][j] > close.iloc[i][j] :

ans.iloc[i][j] = -1

else:

ans.iloc[i][j] = 0

return ans.astype(int)

project\_tests.test\_get\_long\_short(get\_long\_short)

OOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOO

"""

def clear\_signals(signals, window\_size):

Clear out signals in a Series of just long or short signals.

Remove the number of signals down to 1 within the window size time period.

Parameters

----------

signals : Pandas Series

The long, short, or do nothing signals

window\_size : int

The number of days to have a single signal

Returns

-------

signals : Pandas Series

Signals with the signals removed from the window size

# Start with buffer of window size

# This handles the edge case of calculating past\_signal in the beginning

clean\_signals = [0] \* window\_size

for signal\_i, current\_signal in enumerate(signals):

# Check if there was a signal in the past window\_size of days

has\_past\_signal = bool(sum(clean\_signals[signal\_i:signal\_i + window\_size ]))

# Use the current signal if there's no past signal, else 0/False

clean\_signals.append(not has\_past\_signal and current\_signal)

# Remove buffer

clean\_signals = clean\_signals[window\_size:]

# Return the signals as a Series of Ints

return pd.Series(np.array(clean\_signals).astype(np.int), signals.index)

"""

def clear\_signals(signals, window\_size):

"""

Clear out signals in a Series of just long or short signals.

Remove the number of signals down to 1 within the window size time period.

Parameters

----------

signals : Pandas Series

The long, short, or do nothing signals

window\_size : int

The number of days to have a single signal

Returns

-------

signals : Pandas Series

Signals with the signals removed from the window size

"""

# Start with buffer of window size

# This handles the edge case of calculating past\_signal in the beginning

clean\_signals = [0]\*window\_size

for signal\_i, current\_signal in enumerate(signals):

# Check if there was a signal in the past window\_size of days

has\_past\_signal = bool(sum(clean\_signals[signal\_i:signal\_i+window\_size]))

# Use the current signal if there's no past signal, else 0/False

clean\_signals.append(not has\_past\_signal and current\_signal)

# Remove buffer

clean\_signals = clean\_signals[window\_size:]

# Return the signals as a Series of Ints

return pd.Series(np.array(clean\_signals).astype(np.int), signals.index)

##

import copy

def filter\_signals(signal, lookahead\_days):

"""

Filter out signals in a DataFrame.

Parameters

----------

signal : DataFrame

The long, short, and do nothing signals for each ticker and date

lookahead\_days : int

The number of days to look ahead

Returns

-------

filtered\_signal : DataFrame

The filtered long, short, and do nothing signals for each ticker and date

"""

#TODO: Implement function

# before = clear\_signals(signal,lookahead\_days)

# long\_signal = signal[labmda: x>0]

# long = signal.apply(lambda x: x if x >= 0)

# short = signal.apply(lambda x: x if x <= 0)

long\_signals = (signal > 0)

short\_signals = (signal< 0)

print(long\_signals)

# NEZ BKLI LUDL

# 2002-03-15 False False False

print(short\_signals)

for index, row in long\_signals.iterrows():

filtered\_long = clear\_signals(long\_signals, lookahead\_days)

for index, row in short\_signals.iterrows():

filtered\_short = clear\_signals(short\_signals, lookahead\_days)

print(filtered\_long)

print(filtered\_short)

filtered\_signal = filtered\_long + filtered\_short

return filtered\_signal

project\_tests.test\_filter\_signals(filter\_signals)

—————————————

def filter\_signals(signal, lookahead\_days):

#############################################################################################

#for ticker, ticker\_signals in signal.T.iterrows():

# #make a copy of the ticker\_Signals and store it as long\_signals

# #query the long\_signals and filter out rows where long\_signals==-1 and assign them 0

#

# #make a copy of the ticker\_Signals and store it as short\_signals

# #query the short\_signals and filter out rows where short\_signals==1 and assign them 0

#

# # make use of clear\_signals and pass long\_signals & lookahead days as args to it +

# make use of clear\_signals and pass short\_signal and lookahead days as args to it

# and store the sum obtained in filtered\_signal[ticker]

#

#return filtered\_signal

##############################################################################################

templong= copy.deepcopy( signal)

#Change -1 to 0

for col in templong.columns:

templong.loc[templong[col] < 0 , col] = 0

templong.loc[templong[col] < 0 , col] = 0

templong.loc[templong[col] < 0 , col] = 0

print(templong)

tempshort= copy.deepcopy( signal)

#Change 1 to 0

for col in tempshort.columns:

templong.loc[tempshort[col] > 0 , col] = 0

templong.loc[tempshort[col] > 0 , col] = 0

templong.loc[tempshort[col] > 0 , col] = 0

print(tempshort)

## for index, row in templong.iterrows():

# filtered\_long = clear\_signals(templong, lookahead\_days)

filtered\_long = clear\_signals(templong, lookahead\_days)

# for index, row in tempshort.iterrows():

# filtered\_short = clear\_signals(tempshort, lookahead\_days)

filtered\_short = clear\_signals(tempshort, lookahead\_days)

filtered\_signal = filtered\_long + filtered\_short

return filtered\_signal

project\_tests.test\_filter\_signals(filter\_signals)