```
# -*- coding: utf-8 -*-
"""
Created on Sun May 13 23:06:22 2018

@author: Tawanda Vera
"""
# Assignment 4
"""
1.Study the above for Dow Jones Transportation Average and Dow Jones Industrial
Average. Make a note of the technical indicator panels provided –
The stochastic and the Moving Average Convergence Divergence (MACD)
a) Using all the technical information available, provide a pairs trading
recommendation. Provide concrete reasoning to support your choice.
b)Did the Stochastics and MACD help you in deciding on the direction of pairs
trade? If not, which indicator would have been ideal?
2. Write a Python program to download data for Dow Jones Transportation Average
and Dow Jones Industrial Average for the last 5 Years
Create and calculate any one indicator that would allow you to decide on
making a pairs trade between these 2 indices.
Based on the historical values of that indicator, calculate and graphically
represent the return profile of a pairs trading strategy
"""

# 1a)Make a note of the technical indicator panels provided -
# The stochastic and the Moving Average Convergence Divergence (MACD)
"""
The MACD is a momentum and trend-following indicator that is based on the
information of moving averages and, thus, ideal to act as momentum tool and
 momentum filter for trading. When you see the two MACD indicator lines move
 away from each other, it means that momentum is increasing and the trend is
 getting stronger. When the two lines are coming closer to each other,
 it shows that price is losing strength.

The STOCHASTIC indicator shows us information about momentum and trend
strength. The indicator analyses price movements and tells us how fast and
how strong the price moves. The stochastic indicator assumes that prices close
near the low at the end of a rally and near their highs at the end of a
downtrend. The stochastic indicator is confined between 0 and 100 and consists
 of two lines, the %K and the %D. The Stochastic indicators lend themselves to
 crossovers, divergences, hinges,extremes, and reverse divergences and
 there are plotted in their slowed version.

 The stochastic oscillator and MACD function on different technical
 premises and work alone. Compared to the stochastic, which ignores market
 jolts, the MACD is a more reliable option as a sole trading indicator.

"""
# 1b) Using all the technical information available, provide a pairs trading
# recommendation. Provide concrete reasoning to support your choice.
"""
The DJTA/DJIA Spread series measured the spread between two series, which is
often calculated by subtracting the numerator from the denominator rather than
dividing. However, if the spread is calculated over a relatively short period,
it is not important whether subtraction or division is used. The pairs
recommendation will therefore be based on the spread of these two indices.

Near the end of March 2009, the MACD lines had a crossover signalling the end
```

```
to the bearish period. Thereafter, the MACD lines act as support and resistance
during a trend, (April 2009 to September 2009). From Oct 2009 to Mar 2010, is
 a consolidation period, the MACD contracts sharply as well and traders should
wait for the breakout of the wedge to signal a new trend. From Mar 2010 to May
2010, the bollinger bands show a narrow spread range and when the spread breaks
out, the two MACD indicator lines pull away from the 0 line and also separate
each other. Therefore, signals a continued bullish trend.

On the other hand, the stochastic indicator analyzed the spread range over the
 specific time period. The slowed stochastic was set at 14 periods, which meant
 that the indicator took the absolute high and the absolute low of the given
 period and compares it to the closing spread.

In Mar 2009, the low Stochastic value of less than 20 indicated that downside
momentum was very strong. Thereafter, the Stochastic keeps crossed in one
direction indicating that the the bullish trend was still valid. The period,
May 2009 to November 2009, the Stochastic lines show strong trends and is in
the oversold/overbought area, the recommendation is not to fight the trend
but try to hold on to your trades and stick with the bullish trend. Moreover,
when the Stochastic is changing the direction and leaves the overbought/
oversold areas, it can foreshadow a reversal. This is confirmed, when we
combine the Stochastic with a moving average. An important signal for trend
reversals, is when we need to see the Stochastic lines crosss and suddenly
accelarates and leave the overbought-oversold area (September 2009 to May 2010)
,with the two Stochastic bands widening, signalling the start of new trend.
Since, the stochastic indicator was mostly above the 50% line, in this period
which suggested that the bullish trend was still valid. The stochastic value of
59.67% which means that price only closed 40.33% (100% – 59.67%) from the
absolute top, confirming the continuation of this bullish trend.

Similarly, the MACD line supported and confirmed a bullish trend with values
0.008.A signalling that a bull market was starting came in Mid March 2009,
when the MACD reached an overbought and from then on the trend line suggested
an upward and bullish trend. Since, the MACD is positive, it signals that the
shorter term moving average is above the longer term moving average and
suggests upward momentum. The MACD crossover in July 2009 signnalled an
oversold condition that merely triggering a sideways trading that continued
till May 2010.

"""

# 1c) Did the Stochastics and MACD help you in deciding on the direction of
# pairs trade? If not, which indicator would have been ideal?
"""
Yes the MACD and stochastic helped with direction of the pairs trade
recommendation.
"""

##############################################################################
# Write a Python program to download data for Dow Jones Transportation Average
# and Dow Jones Industrial Average for the Last 5 Years
##############################################################################
import pandas as pd
import numpy as np
import pandas_datareader.data as web
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
```

```python
from statsmodels.tsa.stattools import coint
import statsmodels.api as sm
import seaborn as sns
import datetime
import pynance as pn  #pip install
import pyfolio as pf  # pip install pyfolio

sns.set(style='darkgrid', context='talk', palette='Dark2')
my_year_month_fmt = mdates.DateFormatter('%m/%y')

# Download the data for the las 5 years for DJIA and DJTA
sdt = datetime.datetime.now() - datetime.timedelta(5*365)
edt = datetime.datetime.now()
djia = web.DataReader("DJIA", "fred", sdt, edt)
djta = web.DataReader("DJTA", "fred", sdt, edt)

# Create a pandas DataFrame with the close prices of each symbol
# correctly aligned and dropping missing entries
x = pd.DataFrame(index=djia.index)
x['DJIA'] = djia
x['DJTA'] = djta
x = x.dropna()
x.head(10)

def plot_price_series(df, ts1, ts2):
    months = mdates.MonthLocator()  # every month
    fig, ax = plt.subplots(figsize=(18,8))
    ax.plot(df.index, df[ts1], label=ts1)
    ax.plot(df.index, df[ts2], label=ts2)
    ax.xaxis.set_major_locator(months)
    ax.xaxis.set_major_formatter(mdates.DateFormatter('%b %Y'))
    ax.set_xlim(sdt, edt)
    ax.grid(True)
    fig.autofmt_xdate()
    plt.xlabel('Month/Year')
    plt.ylabel('Price ($)')
    plt.title('%s and %s Daily Prices' % (ts1, ts2))
    plt.legend()
    plt.show()

plot_price_series(x, 'DJIA', 'DJTA')

################################################################################
# Create and calculate any one indicator that would allow you to decide on
# making a pairs trade between these 2 indices.
################################################################################
# ## Calculating the Price Spread
# Now we will plot the spread of the two series.
# In order to actually calculate the spread, we use a linear regression to
# get the coefficient for the linear combination to construct between our
# two indices

S1 = sm.add_constant(x.DJTA, prepend=True)
S2 = x.DJIA
results = sm.OLS(S2, S1).fit()
S1 = S1['DJTA']
b = results.params['DJTA']
```

```python
p_spread = S2 - b * S1
p_spread.plot(title="The Spread of DJIA and DJTA")  # Plot the spread
plt.axhline(p_spread.mean(), color='black')  # Add the mean
plt.legend(['Price Spread']);

# ## Calculating the Returns Spread

# Index returns
ret = (x - x.shift(1))/x.shift(1)
ret.tail()

# Cumulative Returns
cumret = ret.cumsum().dropna()
cumret.tail()

# compute the p-value of the cointegration test
# will inform us as to whether the spread btwn the 2 timeseries is stationary
# around its mean

print('Correlation: ' + str(cumret.DJIA.corr(cumret.DJTA)))
score, pvalue, _ = coint(cumret.DJIA, cumret.DJTA)
print ('Cointegration test p-value: ' + str(pvalue))

# Corr = 0.928
# Coint = 0.986


# Now we will plot the spread of the two series.
# In order to actually calculate the spread, we use a linear regression to
# get the coefficient for the linear combination to construct between our
# two indices

R1 = sm.add_constant(cumret.DJTA, prepend=True)
R2 = cumret.DJIA
results = sm.OLS(R2, R1).fit()
b = results.params[1]

# Calculate and plot the spread
spread = R2 - b * R1
spread.plot(title="The Spread of DJIA and DJTA")  # Plot the spread
plt.axhline(spread.mean(), color='black')  # Add the mean
plt.legend(['Spread']);

# Calculate z-score of the Spread
def zscore(series):
    return (series - series.mean()) / np.std(series)

zscore(spread).plot(title="The z-score|Spread of DJIA and DJTA")
plt.axhline(zscore(spread).mean(), color='black')
plt.axhline(1.5, color='red', linestyle='--')
plt.axhline(-1.5, color='green', linestyle='--')
plt.legend(['Spread z-score', 'Mean', '+1.5', '-1.5']);

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

# rolling_spread = DJIA - change in DJIA / change in DJTA * DJTA
```

```
"""
Examining the price ratio of a trading pair is a traditional way to handle
pairs trading. Part of why this works as a signal is based in our assumptions
of how stock prices move, specifically because stock prices are typically
assumed to be log-normally distributed. What this implies is that by taking a
ratio of the prices, we are taking a linear combination of the returns
associated with them (since prices are just the exponentiated returns).
#
# In order to calculate
 We choose instead to move forward with simply calculating the spread between
 the cointegrated stocks using linear regression. This is a very simple way to
 handle the relationship, however, and is likely not feasible for some pairs.
 There are other potential methods for estimating the spread listed at the
 bottom of this lecture.
#
# The absolute spread isn't very useful in statistical terms. It is more
helpful to normalize our signal by treating it as a z-score.
In practice this is usually done to try to give some scale to the data, but
this assumes some underlying distribution, usually a normal distribution.
Under a normal distribution, we would know that approximately 84% of all
spread values will be smaller. However, much financial data is not normally
distributed, and one must be very careful not to assume normality,
nor any specific distribution when generating statistics. It could be the
case that the true distribution of spreads was very fat-tailed and prone to
extreme values. This could mess up our model and result in large losses.
Therefore, a pairs strategy using the ratio of DJIA and DJTA is also included
for comparison purposes.
"""
# Trading Strategy

# Get the roll_spread between the 2 stocks
# Calculate rolling beta coefficient
rolling_beta = pf.timeseries.rolling_beta(R2, R1, 30)
roll_spread = cumret.DJIA - rolling_beta * cumret.DJTA
roll_spread.name = 'roll_spread'
df = pd.Series()
df['roll_spread'] = roll_spread

# Get the 1 day moving average of the price roll_spread
roll_spread_mavg1 = pd.rolling_mean(roll_spread, window=1)
roll_spread_mavg1.name = 'roll_spread 1d mavg'

# Get the 30 day moving average
roll_spread_mavg30 = pd.rolling_mean(roll_spread, window=30)
roll_spread_mavg30.name = 'roll_spread 30d mavg'

plt.plot(roll_spread_mavg1.index, roll_spread_mavg1.values)
plt.plot(roll_spread_mavg30.index, roll_spread_mavg30.values)

plt.legend(['1 Day roll_spread MAVG', '30 Day roll_spread MAVG'])

plt.ylabel('roll_spread');

# Take a rolling 30 day standard deviation

std_30 = pd.rolling_std(roll_spread, window=30)
```

```
std_30.name = 'std 30d'

# Compute the z score for each day
zscore_30_1 = (roll_spread_mavg1 - roll_spread_mavg30)/std_30
zscore_30_1.name = 'roll_spread z-score'
zscore_30_1.plot()
plt.axhline(0, color='black')
plt.axhline(1.0, color='red', linestyle='--');

#Compare Spread and NRatio

ratio = 1 + cumret.DJIA /cumret.DJTA    # ratio
Nratio = np.log(ratio)                  # Normalised ratio
Nratio = Nratio.dropna()                # drop NaN values
df['Nratio']= Nratio                    # add to df

# Plot the normalised spread and ratio for the last 1000 days
zscore(df.tail(1000)).plot(title="Normalised Spread & Ratio")
plt.axhline(zscore(Nratio).mean(), color='yellow')
plt.axhline(1.5, color='red', linestyle='--')
plt.axhline(-1.5, color='green', linestyle='--')
plt.legend(['Spread z-score','Nratio z-score', 'Mean', '+1.5', '-1.5']);

##############################################################################
# Get the ratio between the 2 indices
Nratio = np.log(ratio)
Nratio = Nratio.tail(1000)       # Ratio for the last 1000 days
Nratio.name = 'Nratio'

# Get the 1 day moving average of the price Nratio
Nratio_mavg1 = pd.rolling_mean(Nratio, window=1)
Nratio_mavg1.name = 'Nratio 1d mavg'

# Get the 30 day moving average
Nratio_mavg30 = pd.rolling_mean(Nratio, window=30)
Nratio_mavg30.name = 'Nratio 30d mavg'

plt.plot(Nratio_mavg1.index, Nratio_mavg1.values)
plt.plot(Nratio_mavg30.index, Nratio_mavg30.values)

plt.legend(['1 Day Nratio MAVG', '30 Day Nratio MAVG'])

plt.ylabel('Nratio');

# Take a rolling 30 day standard deviation

std_30 = pd.rolling_std(Nratio, window=30)
std_30.name = 'std 30d'

# Compute the z score for each day
zscore_30_1 = (Nratio_mavg1 - Nratio_mavg30)/std_30
zscore_30_1.name = 'Nratio z-score'
zscore_30_1.plot()
plt.axhline(0, color='black')
plt.axhline(1.0, color='red', linestyle='--');

###############################################################################
```

```
# Plot the prices scaled down along with the negative z-score
# just divide the stock prices by 10 to make viewing it on the plot easier
plt.plot(x.index, x.DJIA.values/10)
plt.plot(x.index, x.DJTA.values/10)
plt.plot(zscore_30_1.index, zscore_30_1.values)
plt.legend(['DJIA Price / 10', 'DJTA Price / 10', 'Price spread Rolling z-Score']);

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

# Trading Strategy for the last 1000 days

# NRatio Stategy
"""
Now, lets define the logic to trade the roll_spread and generate the trading signals.
We will calulate the rolling mean and rolling standard deviation of the roll_spread.
Whenever the spread goes above its 10 period rolling mean by one standard
deviation, we'll short the spread expecting the mean reversion behaviour to
hold true. And whenever the spread goes below its 10 period rolling mean by
one standard deviation, we will go long on the spread
"""
# Get the 10 day moving average of the price Nratio
Nratio_mavg10 = pd.rolling_mean(Nratio, window=10)
Nratio_mavg1.name = 'Nratio 10d mavg'

std_10 = pd.rolling_std(Nratio, window=30)
std_10.name = 'std 10d'

#Fill our newly created position strategy - set to sell (-1) when the price hits
# the upper band, and set to buy (1) when it hits the lower band
y = df.tail(1000)      # Ratio for the last 1000 days
y['Upper'] = Nratio_mavg10 +  std_10
y['Lower'] = Nratio_mavg10 -  std_10
y['Position'] = None
y['Short'] = None
y['Long'] = None


for row in range(len(y)):
    if (df['Nratio'].iloc[row] > df['Upper'].iloc[row]) and (df['Nratio'].iloc[row-1] < df['Upper']
            df['Position'].iloc[row] = -1
    if (df['Nratio'].iloc[row] < df['Lower'].iloc[row]) and (df['Nratio'].iloc[row-1] > df['Lower']
            df['Position'].iloc[row] = 1
n = 1



signal <- ifelse(diff > ub, -1,
                 ifelse(diff < lb, 1, 0))
plot(signal)



# Spread Strategy
"""
* Go "Long" the spread whenever the z-score is below -1.5
* Go "Short" the spread when the z-score is above 1.5
* Exit positions when the z-score approaches zero
```

```
def initialize(context):
    context.stock1 = R1
    context.stock2 = R2

    context.threshold = 1
    context.in_high = False
    context.in_low = False

def handle_data(context, data):

    s1 = context.stock1
    s2 = context.stock2
60 = history(bar_count=60, frequency='1d', field='price')

    p5 = p60.iloc[-5:]

    # Get the 60 day mavg
    m60 = np.mean(p60[s1] - p60[s2])
    # Get the std of the last 60 days
    std60 = np.std(p60[s1] - p60[s2])


    # Current diff = 5 day mavg
    m5 = np.mean(p5[s1] - p5[s2])

    # Compute z-score
    if std60 > 0:
        zscore = (m5 - m60)/std60
    else:
        zscore = 0

    if zscore > context.threshold and not context.in_high:
        order_target_percent(s1, -0.5) # short top
        order_target_percent(s2, 0.5) # long bottom
        context.in_high = True
        context.in_low = False
    elif zscore < -context.threshold and not context.in_low:
        order_target_percent(s1, 0.5) # long top
        order_target_percent(s2, -0.5) # short bottom
        context.in_high = False
        context.in_low = True
    elif abs(zscore) < 1:
        order_target_percent(s1, 0)
        order_target_percent(s2, 0)
        context.in_high = False
        context.in_low = False

"""
```