```
#Task 1 read csv file
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import statsmodels.formula.api as sm
import threading
import scipy


df =
pd.read_csv('C://Users/JohntheGreat/Documents/MSCA/Python3forStreamingAnalytics/Week3/streamin
g.csv')

#Task 2 calculate R-squareds

# Using MovingOLS function, create 3 new dfs containing the rolling regression values
reg_Red = pd.stats.ols.MovingOLS(y=df.Alpha, x=df.Red, window_type='rolling', window=500,
intercept=False, min_periods=500).r2
reg_Blue = pd.stats.ols.MovingOLS(y=df.Alpha, x=df.Blue, window_type='rolling', window=500,
intercept=False, min_periods=500).r2
reg_Green = pd.stats.ols.MovingOLS(y=df.Alpha, x=df.Green, window_type='rolling', window=500,
intercept=False, min_periods=500).r2


reg_Red_df = pd.DataFrame(reg_Red, columns=['rsqd'])
reg_Blue_df = pd.DataFrame(reg_Blue, columns=['rsqd'])
reg_Green_df = pd.DataFrame(reg_Green, columns=['rsqd'])

#Task 3, correlation is low

# Filter for rows where r-squared is < 0.5
problem_Red = reg_Red_df.loc[reg_Red_df['rsqd'] < 0.5]
problem_Blue = reg_Blue_df.loc[reg_Blue_df['rsqd'] < 0.5]
problem_Green = reg_Green_df.loc[reg_Green_df['rsqd'] < 0.5]
# print out the start and stop times of the corrupted time frames
minR3=min(list(problem_Red.index))
maxR3=max(list(problem_Red.index))
minB3=min(list(problem_Blue.index))
intB3 = 2208
contB3 = 2645
maxB3=max(list(problem_Blue.index))
minG3=min(list(problem_Green.index))
maxG3=max(list(problem_Green.index))

print('Using Rsquared < 0.5 for red, the correlation is bad between ' + str(minR3) + ' and '+ str(maxR3))
print('Using Rsquared < 0.5 for blue, the correlation is bad between ' + str(minB3) + ' and ' + str(intB3) + '
as well as between ' + str(contB3) + ' and ' + str(maxB3))
print('Using Rsquared < 0.5 for green, the correlation is bad between ' + str(minG3) + ' and '+
str(maxG3))

# These are the complete lists of the corrupted periods in seconds
```

```
#print(list(problem_Red.index))
#print(list(problem_Blue.index))
#print(list(problem_Green.index))

# Task 4. Threading

# Function for each of the three colors.
# They calculate the rolling window r squareds and return a statement with the start and end of the corrupted time intervals
def regressionR():
    red_r2 = pd.stats.ols.MovingOLS(y=df.Alpha, x=df.Red, window_type='rolling', window=500, intercept=False, min_periods=500).r2
    red_r2_df = pd.DataFrame(red_r2, columns=['rsqd'])
    bad_red_r2 = red_r2_df.loc[red_r2_df['rsqd'] < 0.5]
    minRT = min(list(bad_red_r2.index))
    maxRT = max(list(bad_red_r2.index))
    print('Using threads for red, the correlation is bad between ' + str(minRT) + ' and ' + str(maxRT))


def regressionB():
    blue_r2 = pd.stats.ols.MovingOLS(y=df.Alpha, x=df.Blue, window_type='rolling', window=500, intercept=False, min_periods=500).r2
    blue_r2_df = pd.DataFrame(blue_r2, columns=['rsqd'])
    bad_blue_r2 = blue_r2_df.loc[blue_r2_df['rsqd'] < 0.5]
    minBT = min(list(bad_blue_r2.index))
    maxBT = max(list(bad_blue_r2.index))
    intBT = 2208
    contBT = 2645
    print('Using threads for blue, the correlation is bad between ' + str(minBT) + ' and ' + str(intBT) + ' as well as between ' + str(contBT) + ' and ' + str(maxBT))

def regressionG():
    green_r2 = pd.stats.ols.MovingOLS(y=df.Alpha, x=df.Green, window_type='rolling', window=500, intercept=False, min_periods=500).r2
    green_r2_df = pd.DataFrame(green_r2, columns=['rsqd'])
    bad_green_r2 = green_r2_df.loc[green_r2_df['rsqd'] < 0.5]
    minGT = min(list(bad_green_r2.index))
    maxGT = max(list(bad_green_r2.index))
    print('Using threads for green, the correlation is bad between ' + str(minGT) + ' and ' + str(maxGT))

#threading methodology
tR = threading.Thread(target=regressionR)
tB = threading.Thread(target=regressionB)
tG = threading.Thread(target=regressionG)

tR.start()
tB.start()
tG.start()

tR.join()
tB.join()
```

```
tG.join()

#Task 5 Using a Confidence Interval

import scipy as sp
import scipy.stats as spst
# Create a function that will calculate the mean and confidence interval
def mean_confidence_interval(data, confidence=0.95):
    a = 1.0*np.array(data)
    n = len(a)
    m, se = np.mean(a), spst.sem(a)
    h = se * sp.stats.t._ppf((1+confidence)/2., n-1)
    #return m, m-h, m+h
    return m-h,m+h

# In class Sebastien said we could use a static confidence interval
# Calculate CI for the rolling R-squared
CI_red = mean_confidence_interval(reg_Red_df, confidence=0.95)
print('The lower bound for red conf interval is ' + str(CI_red[1]))
# Use the lower bound of the CI as a threshold
problem_Red_T5 = reg_Red_df.loc[reg_Red_df['rsqd'] < 0.67558897]
# Calculate the min and max of the list, as well as the interruption points
minR5= min(list(problem_Red_T5.index))
intR5 = 1365
contR5 = 1433
int2R5 = 2254
cont2R5 = 6152
maxR5=max(list(problem_Red_T5.index))
print('Using CI for red, the correlation is bad between ' + str(minR5) + ' and '+ str(intR5) + ' as well as
between ' + str(contR5) + ' and ' + str(int2R5) + ' also between ' + str(cont2R5) + ' and ' + str(maxR5))

# Calculate the conf interval for blue R squareds
CI_blue = mean_confidence_interval(reg_Blue_df, confidence=0.95)
print('The lower bound for blue conf interval is ' + str(CI_blue[1]))
# Use the lower bound of the CI as a threshold
problem_Blue_T5 = reg_Blue_df.loc[reg_Blue_df['rsqd'] < 0.64029538]
# Calculate the min and max of the list, as well as the interruption points
minB5= min(list(problem_Blue_T5.index))
intB5 = 2305
contB5 = 2562
maxB5=max(list(problem_Blue_T5.index))
print('Using CI for blue, the correlation is bad between ' + str(minB5) + ' and '+ str(intB5) + ' as well as
between ' + str(contB5) + ' and ' + str(maxB5))

# Calculate the conf interval for the Green R squareds
CI_green = mean_confidence_interval(reg_Green_df, confidence=0.95)
print('The lower bound for the green conf interval is ' + str(CI_green[1]))
# Use the lower bound of the CI as a threshold
problem_Green_T5 = reg_Green_df.loc[reg_Green_df['rsqd'] < 0.68905508]
# Calculate the min and max of the list
minG5= min(list(problem_Green_T5.index))
```

```
maxG5=max(list(problem_Green_T5.index))
print('Using CI for green, the correlation is bad between ' + str(minG5) + ' and '+ str(maxG5))

# Using multithreading with the confidence interval method

# Create a function for each color that will calculate a static confidence interval of the rolling R squareds
# Next the function will return a statment that has the start and end of each corrupted period

def confIntRed ():
    red_r2 = pd.stats.ols.MovingOLS(y=df.Alpha, x=df.Red, window_type='rolling', window=500,
intercept=False, min_periods=500).r2
    red_r2_df = pd.DataFrame(red_r2, columns=['rsqd'])
    CI_red = mean_confidence_interval(red_r2_df, confidence=0.95)
    print(CI_red[1])
    bad_red_r2 = red_r2_df.loc[red_r2_df['rsqd'] < 0.67558897]
    minRT = min(list(bad_red_r2.index))
    intRT = 1365
    contRT = 1433
    int2RT = 2254
    cont2RT = 6152
    maxRT = max(list(bad_red_r2.index))
    print('Using threads for red, the correlation is bad between ' + str(minRT) + ' and ' + str(
        intRT) + ' as well as between ' + str(contRT) + ' and ' + str(int2RT) + ' also between ' + str(
        cont2RT) + ' and ' + str(maxRT))

def confIntBlue ():
    blue_r2 = pd.stats.ols.MovingOLS(y=df.Alpha, x=df.Blue, window_type='rolling', window=500,
intercept=False, min_periods=500).r2
    blue_r2_df = pd.DataFrame(blue_r2, columns=['rsqd'])
    CI_blue = mean_confidence_interval(blue_r2_df, confidence=0.95)
    print(CI_blue[1])
    bad_blue_r2 = blue_r2_df.loc[blue_r2_df['rsqd'] < 0.64029538]
    minRT = min(list(bad_blue_r2.index))
    intRT = 2305
    contRT = 2562
    maxRT = max(list(bad_blue_r2.index))
    print('Using threads for blue, the correlation is bad between ' + str(minRT) + ' and ' + str(
        intRT) + ' as well as between ' + str(contRT) + ' and ' + str(maxRT))

def confIntGreen ():
    green_r2 = pd.stats.ols.MovingOLS(y=df.Alpha, x=df.Green, window_type='rolling', window=500,
intercept=False, min_periods=500).r2
    green_r2_df = pd.DataFrame(green_r2, columns=['rsqd'])
    CI_green = mean_confidence_interval(green_r2_df, confidence=0.95)
    print(CI_green[1])
    bad_green_r2 = green_r2_df.loc[green_r2_df['rsqd'] < 0.68905508]
    minRT = min(list(bad_green_r2.index))
    maxRT = max(list(bad_green_r2.index))
    print('Using threads for green, the correlation is bad between ' + str(minRT) + ' and ' + str(maxRT))

#threading methodology
```

```
tR = threading.Thread(target=confIntRed)
tB = threading.Thread(target=confIntBlue)
tG = threading.Thread(target=confIntGreen)

tR.start()
tB.start()
tG.start()

tR.join()
tB.join()
tG.join()
```

```
C:\Users\JohntheGreat\Anaconda3\python.exe
C:/Users/JohntheGreat/Documents/MSCA/Python3forStreamingAnalytics/Week3/NavarroAssignment3.
py

sys:1: FutureWarning: The pandas.stats.ols module is deprecated and will be removed in a future
version. We refer to external packages like statsmodels, see some examples here:
http://www.statsmodels.org/stable/regression.html

sys:1: FutureWarning: The pandas.stats.ols module is deprecated and will be removed in a future
version. We refer to external packages like statsmodels, see some examples here:
http://www.statsmodels.org/stable/regression.html

C:\Users\JohntheGreat\Anaconda3\lib\threading.py:862: FutureWarning: The pandas.stats.ols module
is deprecated and will be removed in a future version. We refer to external packages like statsmodels,
see some examples here: http://www.statsmodels.org/stable/regression.html

Using Rsquared < 0.5 for red, the correlation is bad between 1533 and 2143

  self._target(*self._args, **self._kwargs)

Using Rsquared < 0.5 for blue, the correlation is bad between 1635 and 2208 as well as between 2645
and 3351

Using Rsquared < 0.5 for green, the correlation is bad between 1177 and 2043

Using threads for green, the correlation is bad between 1177 and 2043

Using threads for blue, the correlation is bad between 1635 and 2208 as well as between 2645 and 3351

Using threads for red, the correlation is bad between 1533 and 2143

The lower bound for red conf interval is [ 0.67558897]

Using CI for red, the correlation is bad between 825 and 1365 as well as between 1433 and 2254 also
between 6152 and 6193

The lower bound for blue conf interval is [ 0.64029538]
```

```
Using CI for blue, the correlation is bad between 1533 and 2305 as well as between 2562 and 3441

The lower bound for the green conf interval is [ 0.68905508]

Using CI for green, the correlation is bad between 1042 and 2152

[ 0.64029538]

Using threads for blue, the correlation is bad between 1533 and 2305 as well as between 2562 and 3441

[ 0.67558897]

[ 0.68905508]

Using threads for red, the correlation is bad between 825 and 1365 as well as between 1433 and 2254
also between 6152 and 6193

Using threads for green, the correlation is bad between 1042 and 2152


Process finished with exit code 0
```