Аналитика данных с помощью pandas и matplotlib

В этом задании вам предлагается выполнить анализ данных криптовалют с помощью библиотек pandas и matplotlib. Задание выглядит как лабораторная работа, в которой вам предстоит заполнить недостающие клетки и ответить на ряд вопросов.

Минимальные баллы для зачёта по этой работе - 1 балл. Если вы не набираете тут 1 балл, то по всему курсу вы получаете неуд (см. слайды с семинара №1)

- Официальная документация pandas
- Официальная документация matplotlib

1. Данные (2 балла)

Начнем с необходимых приготовлений.

```
import numpy as np
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt
import ipywidgets  # Библиотека для интерактивных контролов в jupyter notebook'e
%matplotlib inline
```

Загрузите заранее подготовленный датасет из файла "coins.csv". Создайте объект типа pandas.DataFrame с именем coins и в качестве индекса выберите колонку с датой.

```
coins = pd.read_csv("coins.csv")
```

Посмотрим что получилось

```
coins.head(4)
```

```
         date   price  txCount      txVolume  activeAddresses
symbol  \
0  2013-04-28  135.30  41702.0  6.879868e+07         117984.0    BTC

1  2013-04-28    4.30   9174.0  4.431952e+07          17216.0    LTC

2  2013-04-29  134.44  51602.0  1.138128e+08          86925.0    BTC

3  2013-04-29    4.37   9275.0  3.647810e+07          18395.0    LTC


       name    open    high     low   close  volume        market
0   Bitcoin  135.30  135.98  132.10  134.21     0.0  1.500520e+09
1  Litecoin    4.30    4.40    4.18    4.35     0.0  7.377340e+07
```

```
2   Bitcoin  134.44  147.49  134.00  144.54     0.0  1.491160e+09
3  Litecoin    4.37    4.57    4.23    4.38     0.0  7.495270e+07
```

Поясним значения хранящиеся в колонках

- date - дата измерений
- name - полное название монеты
- symbol - сокращенное название монеты
- price - средняя цена монеты за торговый день в USD
- txCount - количество транзакций в сети данной монеты
- txVolume - объем монет переведенных между адресами в сети данной монеты
- activeAddresses - количество адресов совершавших а данный день транзации в сети данной монеты
- open - цена монеты в начале торгов данного дня
- close - цена монеты в конце торгов данного дня
- high - самая высокая цена данной монеты в течение данного торгового дня
- low - самая низкая цена данной монеты в течение данного торгового дня
- volume - объем торгов данной монетой на биржах в данный день
- market - капитализация данной монеты в данный день

Изучим полученные данные. Ответьте на следующие вопросы (вставляйте клетки с кодом и тектом ниже):

1. Сколько всего различных монет представлено в датасете? (0.4 балла)

```
di = dict()
for label, content in coins.items():
    if label == 'name':
        for coin_type in content:
            di[coin_type]=1
        break
print(len(di))
```

```
70
```

2. За какой период данные мы имеем? (0.4 балла)

```
for label, content in coins.items():
    if label == 'date':
        print(f'From :{content[0]}')
        print(f'To : {content[len(content)-1]}')
        break
```

```
From :2013-04-28
To : 2018-06-06
```

3. Есть ли пропуски в данных? Какой природы эти пропуски? (0.5 балла)

```
for label, content in coins.items():
    if coins.isnull().sum()[label] > 0:
        print(f'{label}:{coins.isnull().sum()[label]}')
```

```
price:327
txCount:1520
txVolume:1830
activeAddresses:1520
```

4. У какой монеты и когда была самая высокая цена? (0.2 балла)

```
maxhighprice = max(coins['high'])
for i in range(len(coins['high'])):
    if coins['high'][i] == maxhighprice:
        index = i
        break
print(f'Date:{coins["date"][index]}')
print(f'Symbol:{coins["symbol"][index]}')
print(f'Name:{coins["name"][index]}')
print(f'HIGHEST_Price:{coins["high"][index]}')
```

```
Date:2017-12-17
Symbol:BTC
Name:Bitcoin
HIGHEST_Price:20089.0
```

5. У какой монеты самая высокая и самая низкая суммарная капитализация? Постройте круговую диаграмму с долями. (0.5 балла)

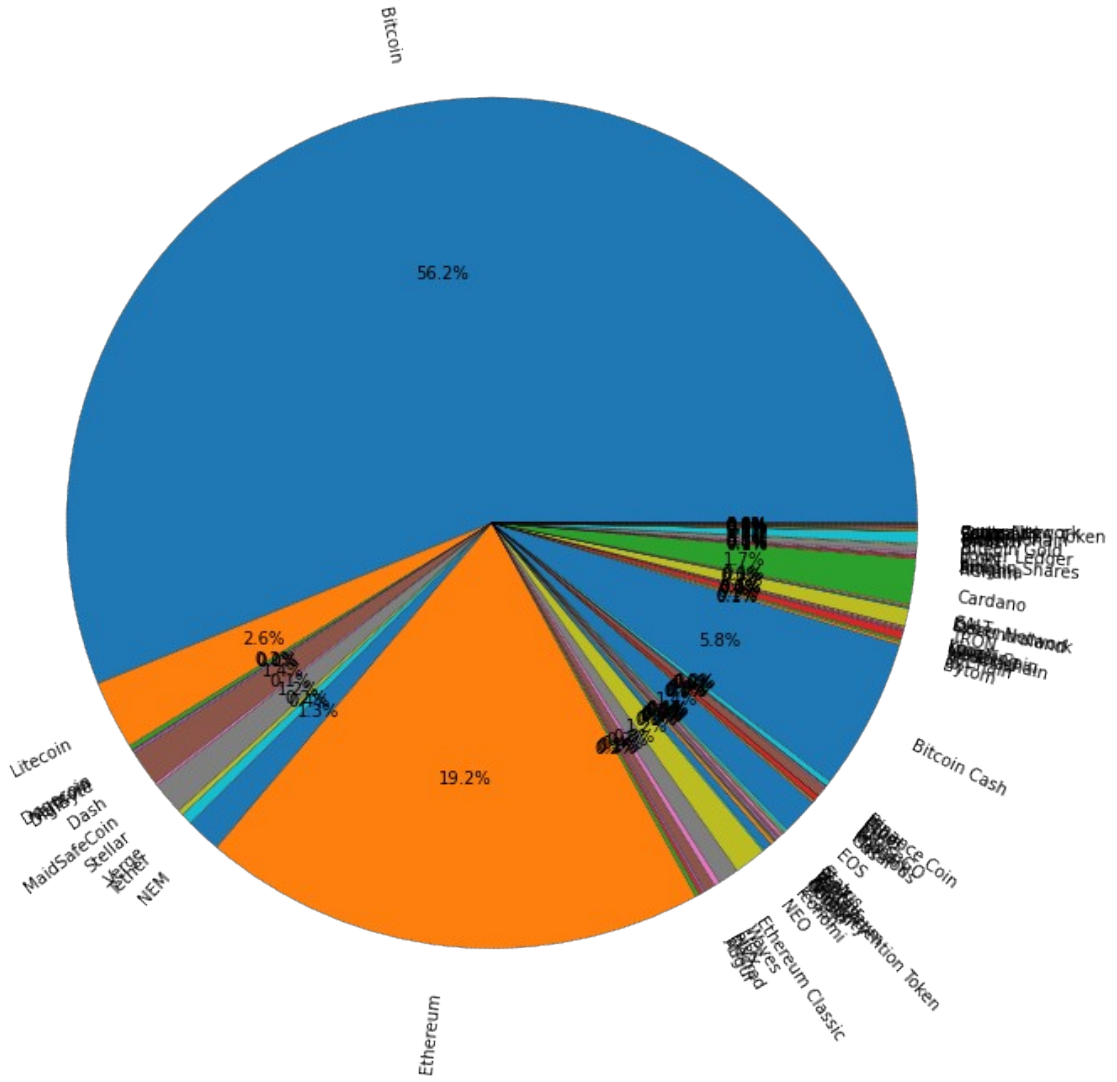
```
data = dict()
for label, content in coins.items():
    if label == 'name':
        for coin_type in content:
            data[coin_type]=0
        break
```

```
for i in range(len(coins)):
    data[coins['name'][i]]+=coins['market'][i]

key_max = max(data.keys(), key=(lambda k: data[k]))
key_min = min(data.keys(), key=(lambda k: data[k]))

print(f'Самая Высокая Суммарная Каптализация: {key_max}:
{data[key_max]}')
print(f'Самая Низкая Суммарная Каптализация: {key_min}:
{data[key_min]}')
```

```
val = list()
lab = list()
for i in data.keys():
    val.append(data[i])
    lab.append(i)


fig, ax = plt.subplots(figsize = (10,10))
ax.pie(val, labels=lab, autopct='%1.1f%%', shadow=False,
wedgeprops={'lw':0.3, 'ls':'--','edgecolor':"k"}, rotatelabels=True)
ax.axis("equal")
```

```
Самая Высокая Суммарная Каптализация: Bitcoin: 57439466431000.0
Самая Низкая Суммарная Каптализация: KingN Coin: 10608251.0
```

```
(-1.106829497948901,
 1.1003252141880429,
 -1.1166507199299711,
 1.1109767178232492)
```



2. Визуализация (1 балл)

Самая интересная часть работы аналитика состоит во внимательном вглядывании в правильно выбранные и построенные графики.

Реализуйте функцию для визуализации цен выбранной валюты за выбранный диапазон дат.

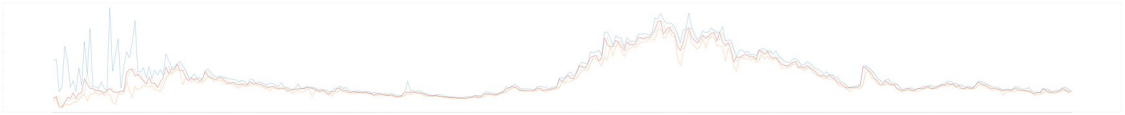
На графике должны быть видны цены начала и конца продаж. А так же минимальная и максимальная цена за этот день. Подпишите график и оси координат. Добавьте сетку. Увеличьте размер изображения. Можете попробовать использовать matplotlib.finance.candlestick_ohlc (mpl_finance.candlestick_ohlc), но можно и без него.

```
def plot_fancy_price_action(coins, symbol, start_date, end_date):
    lst=[]
    tf = 0
```

```
    ls = start_date.strip('-')
    le = end_date.strip('-')
    l1 = coins['date'][0].strip('-')
    l2 = coins['date'][len(coins)-1].strip('-')
    if int(ls[0]) < int(l1[0]) or (ls[0] == l1[0] and int(ls[1]) <
int(l1[1])) or (ls[0] == l1[0] and ls[1] == l1[1] and int(ls[2]) <
int(l1[2])):
        start_date = coins['date'][0]
    if int(le[0]) < int(l1[2]) or (le[0]==l2[0] and
int(le[1])<int(l2[1])) or (le[0]==l2[0] and le[1]==l2[1] and
int(ls[2])<int(l1[2])):
        end_date = coins['date'][len(coins)-1]
    for i in range(len(coins)):
        if coins['date'][i] == start_date:
            tf = 1
        if coins['date'][i] == end_date:
            break
        if tf == 1 and coins['symbol'][i]==symbol:
            lst.append([coins['date'][i], coins['high'][i],
coins['low'][i], coins['open'][i], coins['close'][i], coins['price']
[i]])
    df = pd.DataFrame(lst, columns =['date',
'high','low','open','close', 'middle'])
    plt.figure(figsize=(500, 50))
    plt.plot(df['date'], df['high'],linestyle = '--', linewidth= 5,
title=f'{symbol} High')
    plt.plot(df['date'], df['low'], linestyle = '-.', linewidth= 5,
title=f'{symbol} Low')
    plt.plot(df['date'], df['open'], linestyle = ':', linewidth= 5,
title=f'{symbol} Open')
    plt.plot(df['date'], df['close'], linestyle = '-', linewidth= 5,
title=f'{symbol} Close')
    plt.show()
```

Посмотрим, что получилось:

```
plot_fancy_price_action(coins=coins, symbol='VERI', start_date='2001-
06-01', end_date='2019-06-30')
```



Никакого датасаенса в этом задании нет. Просто аналитик должен уметь строить графики, либо знать готовые инструменты.

3. Накачка и сброс (1 балл)

Криптовалютные биржи до сих пор остаются маргинальным местом, эдаким диким западом финансового мира. Как следствие, здесь процветают схемы относительно честного отъема денег. Одна из них -

pump'n'dump (накачка и сброс). Она выглядит следуюшим образом. Несколько крупных игроков или много мелких договариваются вместе купить малоизвестную монету с низкой ценой и объемом торгов. Это приводит к мгновенному взлету цены (pump), далее приходят неопытные игроки в надежде успеть заработать на таком росте. В этот момент организаторы схемы начнают все продавать (dump). Весь процесс занимает от нескольких минут до нескольких часов.

Ваша задача найти самый сильный pump'n'dump монеты на заданном промежутке времени. Для этого для каждого дня определим число pnd равное отношению максимальной цены монеты в данный день к максимуму из цен открытия и закрытия в тот же день. Нужно найти день когда pnd был максимален и величину pnd.

```
def find_most_severe_pump_and_dump(coins, symbol, start_date,
end_date):
    lst=[]
    tf = 0
    pnd = 0
    dtpnd = ''
    ls = start_date.strip('-')
    le = end_date.strip('-')
    l1 = coins['date'][0].strip('-')
    l2 = coins['date'][len(coins)-1].strip('-')
    if int(ls[0]) < int(l1[0]) or (ls[0] == l1[0] and int(ls[1]) <
int(l1[1])) or (ls[0] == l1[0] and ls[1] == l1[1] and int(ls[2]) <
int(l1[2])):
        start_date = coins['date'][0]
    if int(le[0]) < int(l1[2]) or (le[0]==l2[0] and
int(le[1])<int(l2[1])) or (le[0]==l2[0] and le[1]==l2[1] and
int(ls[2])<int(l1[2])):
        end_date = coins['date'][len(coins)-1]
    for i in range(len(coins)):
        if coins['date'][i] == start_date:
            tf = 1
        if coins['date'][i] == end_date:
            break
        if tf == 1 and coins['symbol'][i]==symbol:
            lst.append([coins['date'][i], coins['high'][i],
coins['low'][i], coins['open'][i], coins['close'][i], coins['price']
[i]])
    df = pd.DataFrame(lst, columns =['date',
'high','low','open','close', 'middle'])
    for i in range(len(df)):
        if df['open'][i] >= df['close'][i]:
            if pnd <= df['high'][i]/df['open'][i]:
                pnd = df['high'][i]/df['open'][i]
                dtpnd = df['date'][i]
        else:
            if pnd < df['high'][i]/df['close'][i]:
                pnd = df['high'][i]/df['close'][i]
```

```
                dtpnd = df['date'][i]
    return [dtpnd, pnd]

find_most_severe_pump_and_dump(coins, symbol='BTC', start_date='2017-
06-01', end_date='2018-06-01')
```

```
['2017-11-29', 1.1428940004366206]
```

Сравните эти значения для разных монет.

```
pndBTC = find_most_severe_pump_and_dump(coins, symbol='BTC',
start_date='2001-09-06', end_date='2022-09-06')
pndLTC = find_most_severe_pump_and_dump(coins, symbol='LTC',
start_date='2001-09-06', end_date='2022-09-06')

print(pndBTC)
print(pndLTC)

print(pndBTC[1]<pndLTC[1])
print(pndBTC[1]>pndLTC[1])
print(pndBTC[1]==pndLTC[1])
```

```
['2015-11-04', 1.2041014675867432]
['2014-08-04', 1.4549071618037137]
True
False
False
```

4. Окупаемость инвестиций (1 балл)

Вам нужно посчитать окупаемость инвестиций в криптовалюты на заданном промежутке времени. Окупаемость определяется как отношение изменения цены портфеля к исходной цене портфеля. Цена портфеля - это суммарная стоимость (в USD) всех монет в портфеле.

investments - dict в котором ключи - это названия монет, значения - это сумма вложений в эту монету (в USD)

```
def compute_roi(coins, investments, start_date, end_date):

    Vstart = 0
    Vend = 0

    ls = start_date.strip('-')
    le = end_date.strip('-')
    l1 = coins['date'][0].strip('-')
    l2 = coins['date'][len(coins)-1].strip('-')
    if int(ls[0]) < int(l1[0]) or (ls[0] == l1[0] and int(ls[1]) <
int(l1[1])) or (ls[0] == l1[0] and ls[1] == l1[1] and int(ls[2]) <
int(l1[2])):
        start_date = coins['date'][0]
```

```
    if int(le[0]) < int(l1[2]) or (le[0]==l2[0] and
int(le[1])<int(l2[1])) or (le[0]==l2[0] and le[1]==l2[1] and
int(ls[2])<int(l1[2])):
        end_date = coins['date'][len(coins)-1]

    for i in range(len(coins)):
        for j in investments.keys():
            if coins['symbol'][i] == j and coins['date'][i] ==
start_date:
                Vstart += investments[j]*coins['price'][i]
                break
            if coins['symbol'][i] == j and coins['date'][i] ==
end_date:
                Vend += investments[j]*coins['price'][i]
                break
    return (Vend - Vstart) / Vstart



compute_roi(coins, investments={'BTC': 1000, 'LTC': 500},
start_date='2018-04-04', end_date='2018-06-01')
```

```
0.004831614576721986
```

```
compute_roi(coins, investments={'BTC': 1000, 'LTC': 500},
start_date='2013-05-28', end_date='2018-06-06')
```

```
57.53575236064575
```

5. Технический анализ (1 балл)

Технический анализ это способ предсказания поведения графика по некоторым вспомогательным величинам построенным по исходному графику. Один из простейших методов технического анализа - границы Болинджера. Кто-то верит, что график касаясь границы от него должен отражаться.

Нарисуйте график цены, скользящее среднее и границы Боллинджера с параметрами N (window) = 21, K (width) = 2.

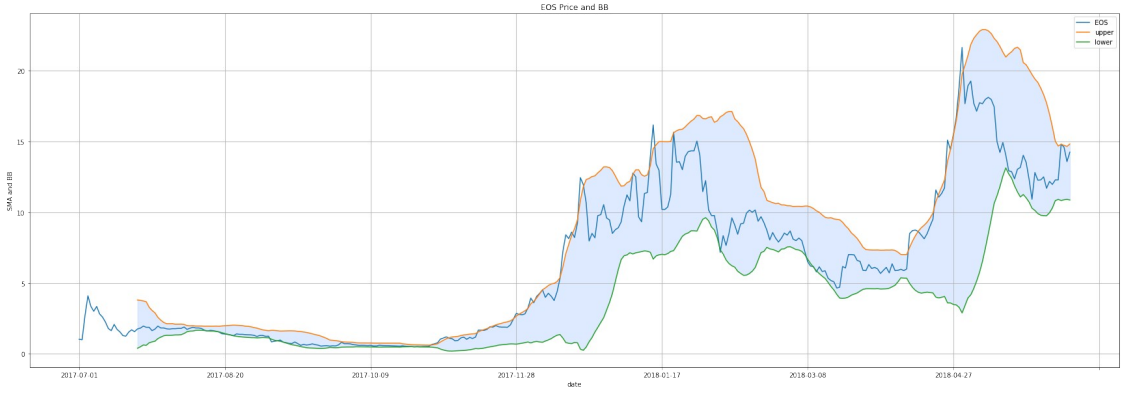
Границы считаются очень просто: (MA + Kσ) и (MA - Kσ), где MA - скользящее среднее за N дней, а σ - скользящее стандартное отклонение за N дней.

Тут вам поможет функция rolling для подсчёта среднего и стандартного отклонения по скользящему окну.

Не забудьте подписать график и оси, отрисовать легенду и выбрать для нее лучшее расположение.

```
def plot_bollinger_bands(coins, symbol, window, width):
    lst=[]
```

```
    for i in range(len(coins)):
        if coins['symbol'][i]==symbol:
            lst.append([coins['date'][i], coins['price'][i]])
    df = pd.DataFrame(lst, columns =['date', 'price'])

    df = df.set_index('date')
    df = df.rename(columns={'price': symbol})
    df.dropna(inplace=True)

    # calculate Simple Moving Average with 20 days window
    sma = df.rolling(window).mean()

    # calculate the standar deviation
    rstd = df.rolling(window).std()

    upper_band = sma + width * rstd
    upper_band = upper_band.rename(columns={symbol: 'upper'})
    lower_band = sma - width * rstd
    lower_band = lower_band.rename(columns={symbol: 'lower'})


    df = df.join(upper_band).join(lower_band)
    ax = df.plot(title='{} Price and
BB'.format(symbol),figsize=(30,10))
    ax.fill_between(df.index, lower_band['lower'],
upper_band['upper'], color='#ADCCFF', alpha=0.4)
    ax.set_xlabel('date')
    ax.set_ylabel('SMA and BB')
    ax.grid()
    plt.show()

plot_bollinger_bands(coins=coins, symbol='EOS', window=21, width=2)  #
тут должен появиться график
```



Сделайте вывод о том, выполнялось ли правило Боллинджера.

Вывод: По графику видно что он в в основном отражется от границ и почти всегда остается между ними, так что для конкретно этого случая теория верна.

6. Капитализация как индикатор (1 балл)

Многие люди, которые торгуют криптовалютой, любят смотреть на капитализацию. Давайте поймём почему.

Нарисуйте еще два графика. На первом должна быть общая капитализация биткойна (BTC), эфира (ETH), еос (EOS), биткойн кэша (BCH), стеллара (XLM) и лайткойна (LTC). На втором - доли капитализаций этих монет от общей капитализации рынка. При этом используйте данные начиная с 2017-07-01.

```
def plot_coins_capitalizations(coins, symbols, start_date):
    lst = []
    l = 0
    j = 0
    plt.figure(figsize=(500, 50))
    for i in range(len(coins)):
        if coins['symbol'][i] in symbols:
            if i!=0 and coins['date'][i] == coins['date'][i-1]:
                l += coins['market'][i]
            else:
                if i!=0:
                    lst.append([coins['date'][i-1],l])
                    l = 0

    df = pd.DataFrame(lst, columns =['date', 'Market'])
    fig, axs = plt.subplots(2,figsize=(500, 100))
    axs[0].plot(df['date'], df['Market'],linestyle = '-', linewidth=
20)
    axs[0].tick_params(labelrotation=90)

    axs[0].set_xlabel('Date', fontsize = 50)
    axs[0].set_ylabel('Market',fontsize = 50)
    axs[0].title('Общая капитализация биткойна (BTC), эфира (ETH), еос
(EOS), биткойн кэша (BCH), стеллара (XLM) и лайткойна (LTC).')
    plt.show()

plot_coins_capitalizations(
    coins=coins,
    symbols=('BTC', 'ETH', 'EOS', 'BCH', 'XLM', 'LTC'),
    start_date='2017-07-01'
)
```

```
---------------------------------------------------------------------
-----
AttributeError                            Traceback (most recent call
```

```
last)
Input In [178], in <cell line: 1>()
----> 1 plot_coins_capitalizations(
      2     coins=coins,
      3     symbols=('BTC', 'ETH', 'EOS', 'BCH', 'XLM', 'LTC'),
      4     start_date='2017-07-01'
      5 )

Input In [177], in plot_coins_capitalizations(coins, symbols,
start_date)
     17 axs[0].plot(df['date'], df['Market'],linestyle = '-',
linewidth= 20)
     18 axs[0].tick_params(labelrotation=90)
---> 20 axs[0].xlabel('Date', fontsize = 50)
     21 axs[0].ylabel('Market',fontsize = 50)
     22 axs[0].title('Общая капитализация биткойна (BTC), эфира (ETH),
еос (EOS), биткойн кэша (BCH), стеллара (XLM) и лайткойна (LTC).')

AttributeError: 'AxesSubplot' object has no attribute 'xlabel'

<Figure size 36000x3600 with 0 Axes>
```

Проанализируйте зависимость доли капитализации альткойнов от доли капитализации биткойна. Как выдумаете, в чём причина такой зависимости?

7. Корреляции монет (1 балл)

Теперь нужно подробнее посмотреть на корреляции средних долей капитализаций монет. При этом будем смотреть на среднее сглаженное за последние window дней до дня date с коэффициентом сглаживания alpha для набора монет symbols.

Реализуйте функцию, которая будет возвращать квадратный DataFrame с числом строк и столбцов равным числу рассматриваемых монет и со значениями корреляций.

```
def calc_coins_correlations(coins, date, symbols, window, alpha):
    # Paste your code here

correlations = calc_coins_correlations(coins, date="2018-06-06",
                                       symbols=['BTC', 'ETH', 'EOS',
'BCH', 'XLM', 'LTC', 'ADA'],
                                       window=21, alpha=0.1)
# Теперь посмотрим на эти корреляции следующим образом:
correlations.style.background_gradient(cmap='coolwarm').set_precision(
2)
```

Довольно интересно ещё взглянуть на 2017-12-27:

```
correlations = calc_coins_correlations(coins, date="2017-12-27",
                                       symbols=['BTC', 'ETH', 'EOS',
'BCH', 'XLM', 'LTC', 'ADA'],
                                       window=21, alpha=0.1)
```

```
# Теперь посмотрим на эти корреляции следующим образом:
correlations.style.background_gradient(cmap='coolwarm').set_precision(
2)
```

8. Анализ одной стратегии (2 балла)

Разберем один мечтательный пример. Посмотрим какую прибыль могла бы нам принести хрестоматийная торговая стратегия основанная на скользящих средних. Стратегия выглядит следующим образом: мы строим две скользящие среднии для графика цены. С маленьким окном (ведущее скользящее среднее) и с бОльшим окном (запаздывающее скользящее среднее). Мы покупаем, когда ведущее среднее становится больше запаздывающего, и продаем в противном случае. Посмотрим на пример

```
def plot_moving_averages(coins, symbol, leading_window,
lagging_window, start_date, end_date):
    coin = coins[coins['symbol'] == symbol][start_date:end_date]
    price = coin['price']
    leading_mean = price.rolling(window=leading_window).mean()
    lagging_mean = price.rolling(window=lagging_window).mean()

    fig = plt.figure(figsize=(16, 9))
    ax = fig.add_subplot(111)

    ax.set_title('Price action for {}'.format(symbol))
    ax.plot(leading_mean, color='green',
label='MA{}'.format(leading_window))
    ax.plot(lagging_mean, color='red',
label='MA{}'.format(lagging_window))
    ax.plot(price, color='blue', label='price')
    ax.set_xlabel('Date')
    ax.set_ylabel('Price')
    ax.legend(loc='best')
    ax.grid(True)
    plt.show()

plot_moving_averages(
    coins=coins,
    symbol='BTC',
    leading_window=21,
    lagging_window=50,
    start_date='2017-05-01',
    end_date='2018-08-01')
```

Видно, что скользящее среднее с бОльшим окном медленнее реагирует на изменение цены. Именно на этой идее и основана торговая стратегия.

Реализуйте функцию, которая строит два графика. На правом будут изображены цена и скользящие средние. На левом - во сколько раз изменится размер вложений при использовании нашей стратегии и при обычном инвестировании

Notes:

Давайте использовать только цены закрытия. При этом, чтобы узнать цены за вчерашний день, стоит использовать метод shift(1) у Series. Отношение цен закрытия за сегодня и за вчера - это мой multiplier за сегодняшний день. При этом давайте строить графики накопления для multipliers. Т.е. если мы смотрим на 3 дня и в первый день multiplier = 1.5, во второй- 0.5 и в третий 2. То график будет выглядеть так: (1.5, 1.5 * 0.5, 1.5 * 0.5 * 2).

При использовании нашей новой стратегии мы будем либо покупать, если ведущее среднее становится больше запаздующего на некоторый threshold (при этом лучше разницу сперва поделить на цену), либо оставлять всё как есть. При этом, конечно, нужно, принимая решения за сегодняшний день, смотреть только на статистику из прошлого.

```
def plot_moving_averages_strategy(
    coins, symbol, lead_window, lag_window, threshold, start_date,
end_date
):
    # Paste your code here
```

```
# Теперь на основе реализованной функции сделаем интерактивные графики
и поизучаем, что получилось:
symbol_selector = ipywidgets.Dropdown(
    options=('BTC', 'ETH', 'EOS', 'BCH', 'XLM', 'LTC', 'ADA'),
    index=0,
    value='BTC',
    layout={'width': '700px'},
    continuous_update=False
)

lead_window_slider = ipywidgets.IntSlider(
    value=21,
    min=1,
    max=200,
    step=1,
    layout={'width': '700px'},
    continuous_update=False)

lag_window_slider = ipywidgets.IntSlider(
    value=50,
    min=1,
    max=200,
    layout={'width': '700px'},
    step=1, continuous_update=False)

threshold_slider = ipywidgets.FloatSlider(
    min=0,
    max=0.20,
    step=0.001,
    value=0.025,
    layout={'width': '700px'},
    continuous_update=False)

start_date_slider = ipywidgets.SelectionSlider(
    options=pd.date_range('2013-04-28', '2018-06-06', freq='D'),
    index=0,
    value=pd.Timestamp('2017-05-01'),
    layout={'width': '700px'},
    continuous_update=False
)

end_date_slider = ipywidgets.SelectionSlider(
    options=pd.date_range('2013-04-28', '2018-06-06', freq='D'),
    index=0,
    value=pd.Timestamp('2018-01-01'),
    layout={'width': '700px'},
    continuous_update=False
)

ipywidgets.interact(
```

```
    plot_moving_averages_strategy,
    coins=ipywidgets.fixed(coins),
    symbol=symbol_selector,
    lead_window=lead_window_slider,
    lag_window=lag_window_slider,
    threshold=threshold_slider,
    start_date=start_date_slider,
    end_date=end_date_slider
)
```

Попробуйте разные значения параметров для разных монет и сделайте выводы о применимости такой модели.

9. Отказ от ответственности

Все примеры разобранных здесь стратегий являются игрушечными и не подходят для реальной торговли на бирже. Без серьезной подготовки вас там съедят с потрохами.