Санкт-Петербургский Национальный Исследовательский Университет

Информационных Технологий, Механики и Оптики

Кафедра Систем Управления и Информатики

**Лабораторная работа №4**

**Вариант №5**

Выполнил:

Олейникова П.Л.

Проверил:

Мусаев А.А.

Санкт-Петербург,

2022

**Задача №1**

Гитхаб: https://github.com/OleinikovaPolina/1kurs/tree/main/Python-Programming/Lab-4.

Задание.

Используя данные котировок акций из Лабораторной работы №3, реализовать графический интерфейс, который должен содержать:

1) Выбор тикера акции

2) Временной период

3) Выбор метода восстановления пропущенных данных (винзорирование, линейная аппроксимация, корреляционное восстановление)

4) Выбор метода сглаживания (взвешенный метод скользящего среднего, метод скользящего среднего со скользящим окном наблюдения)

5) Выбор максимально допустимого отклонения сглаженного процесса от реального

6) Кнопку «Build»

В результате нажатия на кнопку «Build» должны появляться графики исходного процесса после восстановления данных, график сглаженного процесса и информация о максимальном отклонении.

Для реализации интерфейса допускается использование любого модуля на выбор студента.

Для реализации алгоритмов сглаживания использование готовых библиотек не допускается.

Данные были взяты с ресурсов [1, 2] и все файлы представлены на гитхабе.

Решение. (Рисунок 1, 2)

from datetime import datetime  
  
  
def get\_data(res, start, end, f):  
 format1 = '%d.%m.%Y'  
 format2 = '%Y-%m-%d'  
 first = 0  
 last = len(res) - 1  
 arr1 = []  
 for i in f.readlines():  
 a = i.split('\t')  
 num = a[1].replace('\n', '').replace(' ', '').replace(',', '.')  
 if num and (datetime.strptime(a[0], format1) >= datetime.strptime(start, format2)) and (  
 datetime.strptime(a[0], format1) <= datetime.strptime(end, format2)):  
 arr1.append([datetime.strptime(a[0], format1), float(num)])  
 elif num and len(arr1) == 0:  
 last = [datetime.strptime(a[0], format1), float(num)]  
 elif num and len(arr1) > 0 and first == 0:  
 first = [datetime.strptime(a[0], format1), float(num)]  
 arr1.reverse()  
  
 a\_days = (arr1[0][0] - datetime.strptime(start, format2)).days  
 if a\_days > 1:  
 res.append(first[1])  
 for j in range(1, a\_days):  
 res.append(None)  
  
 res.append(arr1[0][1])  
 for i in range(1, len(arr1)):  
 a\_days = (arr1[i][0] - arr1[i - 1][0]).days  
 if a\_days > 1:  
 for j in range(1, a\_days):  
 res.append(None)  
 res.append(arr1[i][1])  
 else:  
 res.append(arr1[i][1])  
  
 a\_days = (datetime.strptime(end, format2) - arr1[-1][0]).days  
 if a\_days > 1:  
 for j in range(1, a\_days):  
 res.append(None)  
 res.append(last[1])  
  
 return res, []  
  
  
def recovery(res, type\_recovery):  
 def winzoning(a):  
 for i in range(1, len(a)):  
 if a[i] is None:  
 a[i] = a[i - 1]  
 return a  
  
 def appr(a):  
 count = 0  
 for i in range(1, len(a)):  
 if a[i] is None:  
 count += 1  
 elif count > 0:  
 for j in range(count):  
 a[i - count + j] = a[i - count + j - 1] + (a[i] - a[i - count - 1]) / (count + 1)  
 count = 0  
 return a  
  
 res2 = []  
 if type\_recovery == 'винзорирование':  
 res2 = winzoning(res)  
 if type\_recovery == 'линейная аппроксимация':  
 res2 = appr(res)  
 return res2  
  
  
def smoothing(res, type\_smoothing, max\_deviation):  
 def window\_smoothing():  
 res2 = [[0.0], [res[0]]]  
 start = res[0]  
 summ = res[0]  
 countt = 1  
 maxs = []  
 for i in range(1, len(res)):  
 if abs(res[i] - start) < max\_deviation:  
 summ += res[i]  
 countt += 1  
 else:  
 res2[0].append(i - countt / 2)  
 res2[1].append(summ / countt)  
 maxs.append(abs(res[i] - start))  
 summ = res[i]  
 countt = 1  
 start = res[i]  
 res2[0].append(len(res) - 1)  
 res2[1].append(res[-1])  
  
 return res2, round(max(maxs), 3)  
  
 def avg\_smoothing():  
 res2 = [[0.0], [res[0]]]  
 maxs = []  
 for i in range(1, len(res)):  
 s = 0  
 s2 = 0  
 for j in range(min(i, max\_deviation)):  
 s += res[i - j] \* (max\_deviation - j + 1)  
 s2 += (max\_deviation - j + 1)  
 res2[0].append(i)  
 res2[1].append(s / s2)  
 maxs.append(abs(res[i] - res2[1][i]))  
 return res2, round(max(maxs), 3)  
  
 if type\_smoothing == 'метод скользящего среднего со скользящим окном наблюдения':  
 return window\_smoothing()  
  
 if type\_smoothing == 'взвешенный метод скользящего среднего':  
 return avg\_smoothing()  
  
  
def main(companies, start, end, type\_recovery, type\_smoothing, max\_deviation):  
 res = []  
 with open('companies/' + companies + '.txt', encoding='utf-8') as f:  
 res, dates = get\_data(res, start, end, f)  
 res = recovery(res, type\_recovery)  
 res2, max\_deviation\_res = smoothing(res, type\_smoothing, max\_deviation)  
 return res, res2, max\_deviation\_res

Рисунок 1 – Код модуля для 1 задачи

В данной программе реализован основной функционал для восстановления данных, их сглаживания.

from dash import Dash, dcc, html  
from dash.dependencies import Input, Output, State  
from datetime import date, timedelta, datetime  
from module import main  
import plotly.graph\_objs as go  
import math  
  
companies = ['алроса', 'афк', 'аэрофлот', 'втб', 'газпром', 'ленэнерго', 'лср', 'лукойл', 'мосбиржа', 'мтс', 'нлмк',  
 'новатэк', 'пик', 'русгидро', 'сбер', 'северсталь', 'татнефть', 'яндекс']  
  
external\_stylesheets = ["https://codepen.io/chriddyp/pen/bWLwgP.css"]  
  
app = Dash(\_\_name\_\_, external\_stylesheets=external\_stylesheets)  
  
app.layout = html.Div([  
 dcc.Dropdown(companies, 'алроса', id='company', placeholder='Тикер', style={'width': '265px'}),  
 dcc.DatePickerSingle(id='start', date=date(2016, 12, 1), min\_date\_allowed=date(2016, 12, 1),  
 max\_date\_allowed=date(2019, 12, 31), placeholder='Начало', ),  
 dcc.DatePickerSingle(id='end', date=date(2019, 12, 31), min\_date\_allowed=date(2016, 12, 1),  
 max\_date\_allowed=date(2019, 12, 31), placeholder='Конец', ),  
 dcc.Dropdown(['винзорирование', 'линейная аппроксимация'], 'линейная аппроксимация', id='type\_recovery',  
 placeholder='Метод восстановления', style={'width': '265px'}),  
 dcc.Dropdown(  
 ['взвешенный метод скользящего среднего', 'метод скользящего среднего со скользящим окном наблюдения'],  
 'метод скользящего среднего со скользящим окном наблюдения', id='type\_smoothing',  
 placeholder='Метод сглажтивания', style={'width': '265px'}),  
 dcc.Input(1, id="max\_deviation", type="number", placeholder='Окно',  
 style={'width': '265px'}),  
 html.Br(),  
 html.Button('Build', id='submit-val', n\_clicks=0),  
 dcc.Graph(id='graph-with-slider'),  
])  
  
  
@app.callback(  
 Output('graph-with-slider', 'figure'),  
 Input('submit-val', 'n\_clicks'),  
 State('company', 'value'),  
 State('start', 'date'),  
 State('end', 'date'),  
 State('type\_recovery', 'value'),  
 State('type\_smoothing', 'value'),  
 State('max\_deviation', 'value'),  
)  
def update\_output(n\_clicks, company, start, end, type\_recovery, type\_smoothing, max\_deviation):  
 text = 'Результат'  
 if company is None:  
 text = 'Выберите компанию'  
 if start is None:  
 text = 'Выберите дату начала'  
 if end is None:  
 text = 'Выберите дату окончания'  
 if type\_recovery is None:  
 text = 'Выберите метод восстановления'  
 if type\_smoothing is None:  
 text = 'Выберите метод сглаживания'  
 if max\_deviation is None:  
 text = 'Выберите максимальное отклонение сглаживающего процесса'  
 if text != 'Результат':  
 fig = go.Figure(layout={'title': text})  
 fig.update\_layout(transition\_duration=500)  
 return fig  
 try:  
 old, new, maxs = main(company, start, end, type\_recovery, type\_smoothing, max\_deviation)  
 dates1 = [(datetime.strptime(start, '%Y-%m-%d') + timedelta(days=i)).strftime('%d.%m.%Y') for i in  
 range(0, (datetime.strptime(end, '%Y-%m-%d') - datetime.strptime(start, '%Y-%m-%d')).days + 1)]  
 print(maxs)  
 dates2 = []  
 for i in new[0]:  
 dates2.append(dates1[math.floor(i)])  
 fig = go.Figure()  
 fig.add\_trace(go.Scatter(x=dates1, y=old, name='before'))  
 fig.add\_trace(go.Scatter(x=dates2, y=new[1], name='after'))  
 fig.update\_layout(title\_text=f'Результат (максимально отклонение = {maxs} )')  
 except ValueError:  
 fig = go.Figure()  
 fig.update\_layout(title\_text='Уменьшите максимальное отклонение или увеличьте период')  
  
 return fig  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 app.run\_server(debug=True)

Рисунок 2 – Код программы для 1 задачи

В данном коде представлен функционал для создания интерфейса к заданию. Графики были построены с помощью библиотеки Plotly [3].

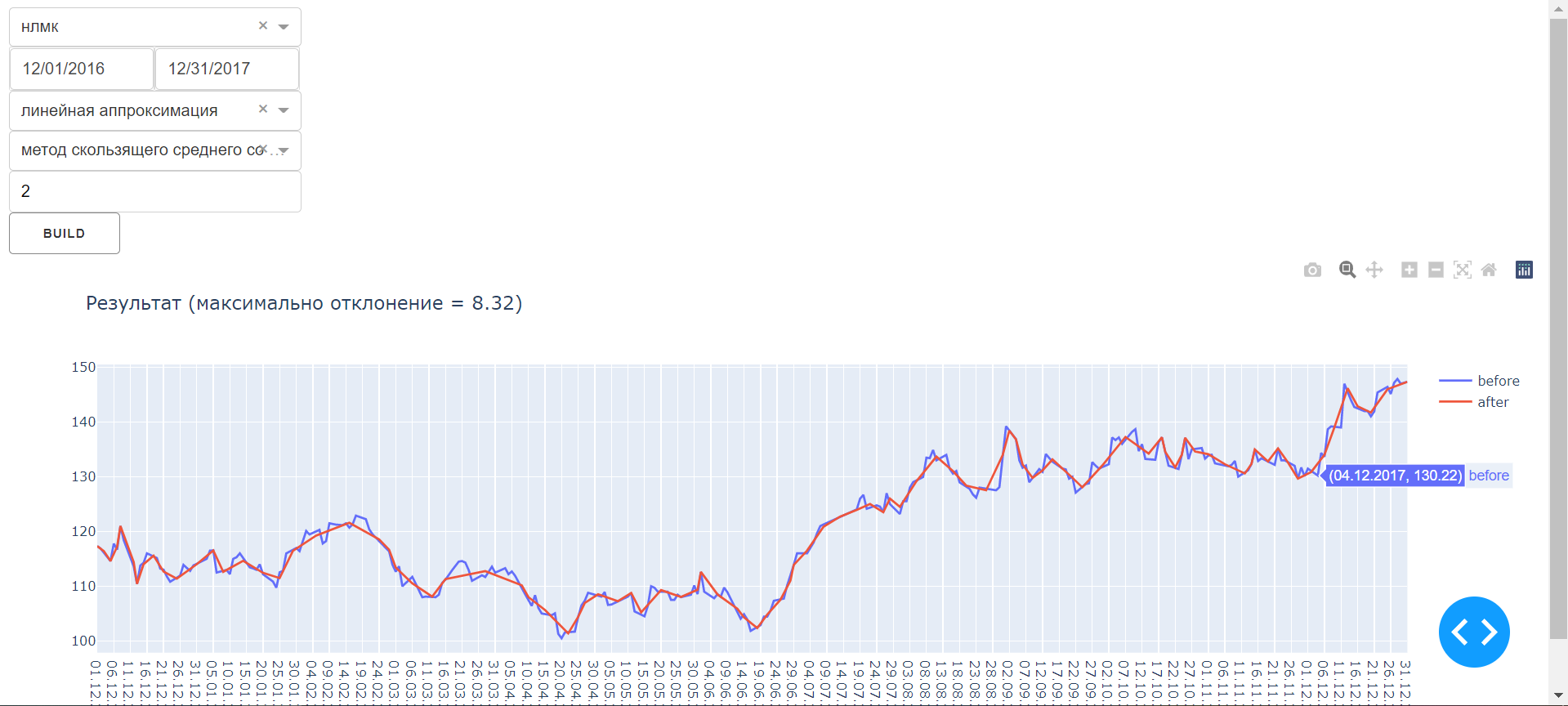


Рисунок 3 – Результат работы программы для 1 задачи

На данном графике представлены данные, восстановленные методом линейной аппроксимации и сглаженные методом скользящего среднего со скользящим окном наблюдения в 2 рубля (около 1.5%).

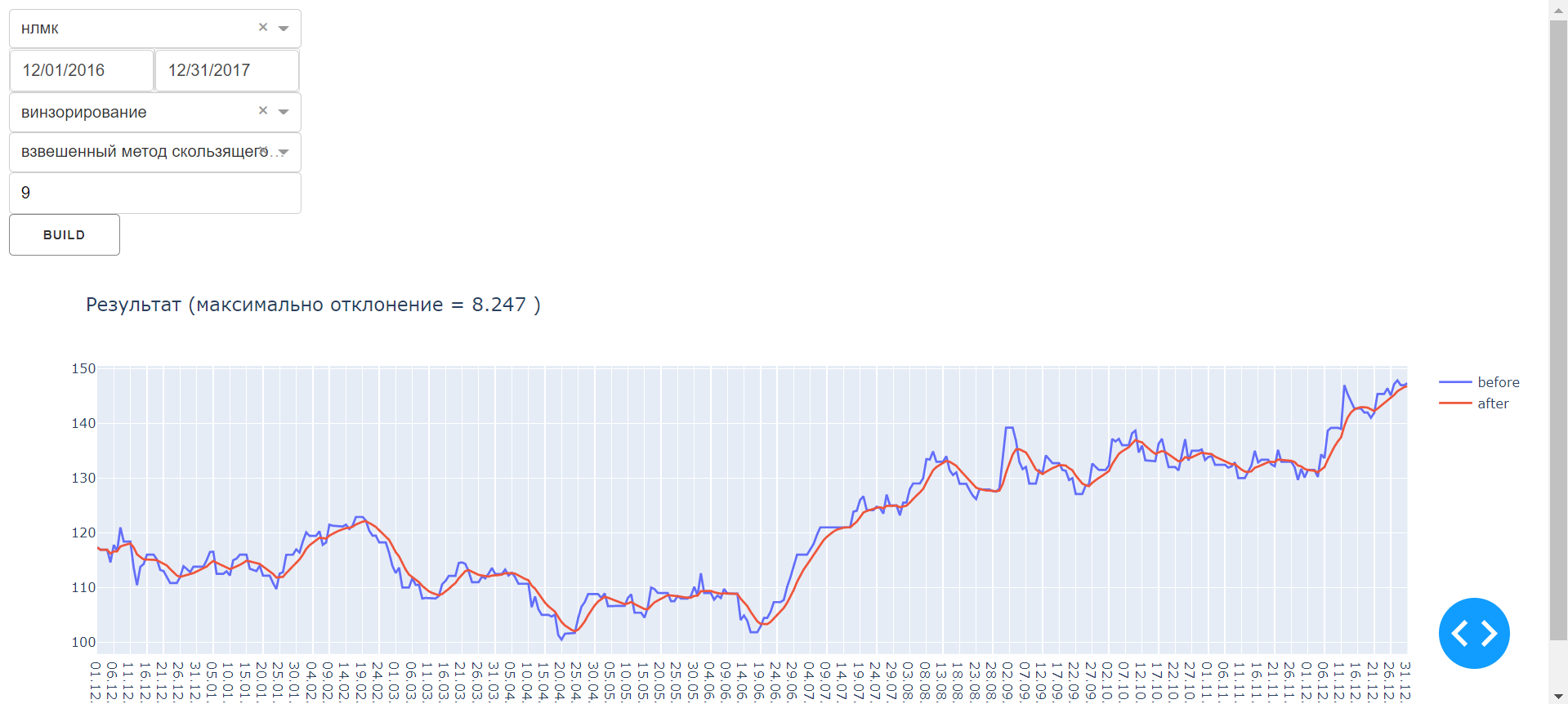


Рисунок 4 – Результат работы программы для 1 задачи

На данном графике представлены данные, восстановленные методом винзонирования и сглаженные методом взвешенного скользящего среднего с окном в 9 дней. Окно было выбрано так, чтобы максимальное отклонение было таким же, как в 1 случае.

Вывод. Функция сглаженных данных в первом случае получается более ломанной и подчеркивает многие экстремумы. Во втором же случае сглаживание получается более ровным и легко регулируемым. Восстановление данных во втором случае получается более нечетким, функция отклонена от сглаженной больше, чем при первом восстановлении.

В результате данной работы был создан интерфейс для отслеживания данных с возможностью их восстановление и последующего сглаживания с учетом пожеланий пользователя. Были повторены методы восстановления, изучены новые методы сглаживания, была проведена работа с библиотеками dash, plotly.

**СПИСОК ЛИТЕРАТУРЫ**

1. Investfunds.ru. [Электронный ресурс] – https://investfunds.ru/ (16.03.2022);
2. Finanz.ru. [Электронный ресурс] – https://www.finanz.ru/ (16.03.2022);
3. Plotly. Dash Python User Guide. [Электронный ресурс] – <https://dash.plotly.com/> (21.02.2022).