

Санкт-Петербургский Национальный Исследовательский Университет

Информационных Технологий, Механики и Оптики

Мегафакультет трансляционных и информационных технологий

Лабораторная работа №4

Вариант №5

Выполнил(и:)

Жаров Александр Павлович

Проверил

Мусаев А.А.

Санкт-Петербург,

2022

Задание 1

Текст задания

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.

Решение

Решение я начал с создания интерфейса. Для него выбрал стандартную библиотеку Tkinter и после нехитрых манипуляций получился следующий интерфейс (Рис. 1)

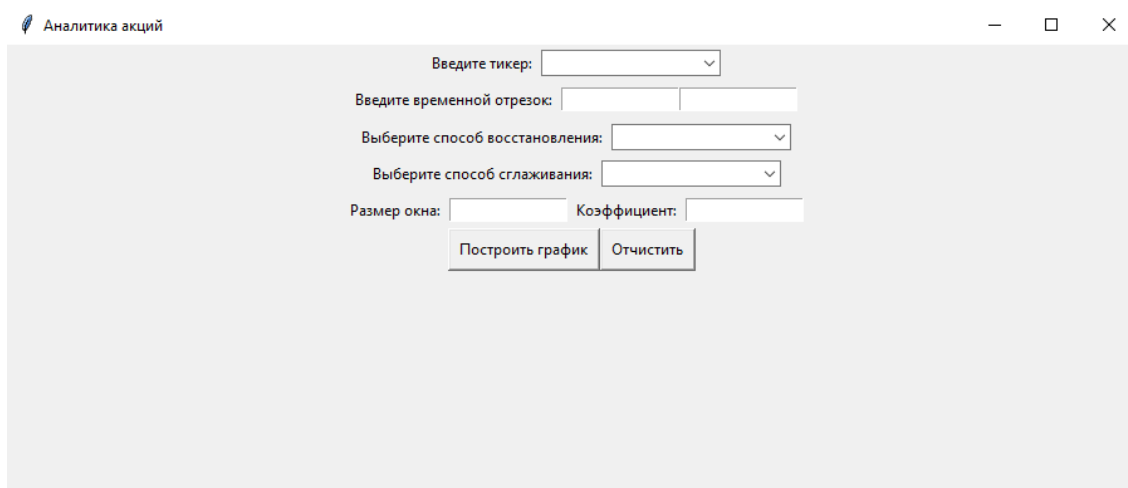


Рисунок 1

В первом выпадающем окне выбираем тикер компании, далее вводим временной отрезок, затем выбираем способ восстановления данных, способ сглаживания, вводим дополнительную информацию для сглаживания (размер окна сглаживания и коэффициент) и наконец две кнопки: по нажатию на первую – строиться график, вторая же отчищает холст (Рис. 2).

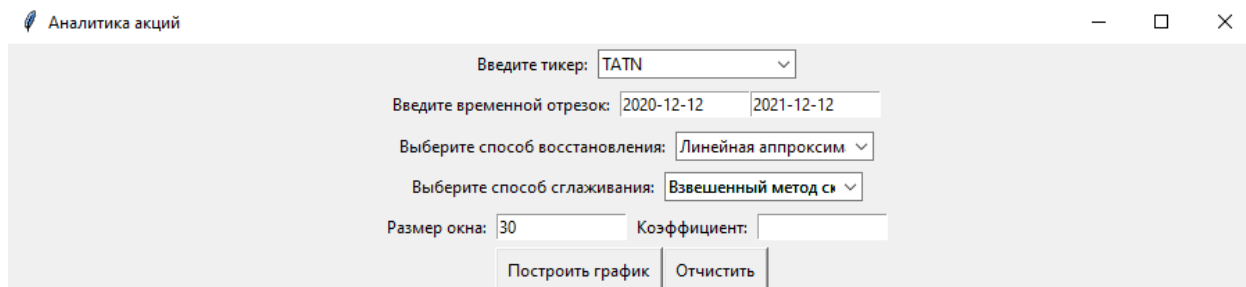


Рисунок 2

При нажатии на “Построить график” строиться 3 графика (Рис.3):

Оранжевый – полученные котировки за указанный промежуток (встречаются пропуски т.к. Мосбиржа работает не всегда)

Синий – восстановленные указанным способом потерянные данные

Зеленый – сглаженный график

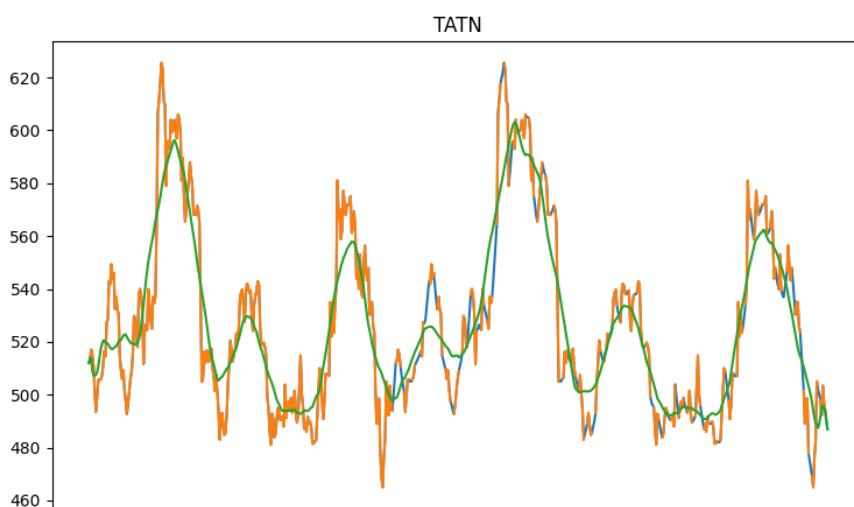
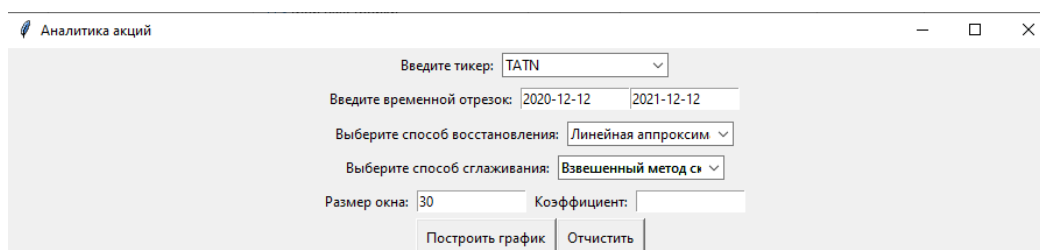


Рисунок 3

Теперь разберемся с технической стороной решения. Код главной странички

(Рис. 4, 5, 6).

```
main.py > Select
1  import copy
2  from tkinter import *
3  import tkinter as tk
4  from tkinter import ttk
5  import matplotlib.pyplot as plt
6  from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
7  import module as m
8  import recovery as r
9  import smoothing as s
10
11 tickers = ["YNDX", "GAZP", "TATN", "SBER", "VTBR", "ALRS", "AFLT", "HYDR"]
12 recovery = ["Винзорирование", "Линейная аппроксимация"]
13 smoothing = ["Взвешенный метод скользящего среднего", "Метод скользящего среднего со скользящим окном наблюдения"]
14
15 # Функции для обработки введенных данных
16 def Select():
17     ticker = cmb1.get()
18     start = date_start.get()
19     end = date_end.get()
20     print(start, end)
21     quot = m.getData(ticker, start, end)
22     rst = copy.copy(quot)
23     #Восстановление
24     if cmb2.get() == "Винзорирование":
25         print("Винзорирование")
26         rst = r.venz(rst)
27
28     elif cmb2.get() == "Линейная аппроксимация":
29         print("Линейная аппроксимация")
30         rst = r.aprokRest(rst)
31
32     #Сглаживание
33     smt = copy.copy(rst)
34     if cmb3.get() == "Взвешенный метод скользящего среднего":
35         print("Взвешенный метод скользящего среднего")
36         smt = s.smt1(smt, float(ntr1.get()))
37     elif cmb3.get() == "Метод скользящего среднего со скользящим окном наблюдения":
38         print("Метод скользящего среднего со скользящим окном наблюдения")
39         smt = s.smt2(smt, float(ntr2.get()))
40
41     Generate(quot, rst, smt)
42
43
44 def Generate(quot, rst, smt):
45     global conv
46     if conv:
47         conv.get_tk_widget().destroy()
48
49     figure2 = plt.Figure(figsize=(16, 9), dpi=100)
```

Рисунок 4

```

50     ax2 = figure2.add_subplot(111)
51     conv = FigureCanvasTkAgg(figure2, str8)
52     conv.get_tk_widget().pack(side=tk.LEFT, fill=tk.BOTH)
53     ax2.plot(rst)
54     ax2.plot(quot)
55     ax2.plot(smt)
56     ax2.set_title(cmb1.get())
57     return conv
58
59
60 def Close():
61     global conv
62     if conv:
63         conv.get_tk_widget().destroy()
64     return conv
65
66 # Структура окна
67 window = Tk()
68 window.geometry("900x700")
69 window.title("Аналитика акций")
70
71 conv = None
72
73 str1 = Frame(window)
74 str2 = Frame(window)
75 str3 = Frame(window)
76 str4 = Frame(window)
77 str5 = Frame(window)
78 str6 = Frame(window)
79 str7 = Frame(window)
80 str8 = Frame(window)
81
82 str1.pack()
83 str2.pack()
84 str3.pack()
85 str4.pack()
86 str5.pack()
87 str6.pack()
88 str7.pack()
89 str8.pack()
90
91 lb1 = Label(str1, text="Введите тикер:", padx=5, pady=5)
92 lb1.pack(side=LEFT)
93
94 lb2 = Label(str2, text="Введите временной отрезок:", padx=5, pady=5)
95 lb2.pack(side=LEFT)
96

```

Рисунок 5

```

91 lb1 = Label(str1, text="Введите тикер:", padx=5, pady=5)
92 lb1.pack(side=LEFT)
93
94 lb2 = Label(str2, text="Введите временной отрезок:", padx=5, pady=5)
95 lb2.pack(side=LEFT)
96
97 cmb1 = ttk.Combobox(str1, values=tickers , state="readonly")
98 cmb1.pack(side=LEFT)
99
100 date_start = Entry(str2, width=15)
101 date_end = Entry(str2, width=15)
102 date_start.pack(side=LEFT)
103 date_end.pack(side=LEFT)
104
105 lb3 = Label(str4, text="Выберите способ восстановления:", padx=5, pady=5)
106 lb3.pack(side=LEFT)
107
108 cmb2 = ttk.Combobox(str4, values=recovery, state="readonly")
109 cmb2.pack(side=LEFT)
110
111 lb3 = Label(str5, text="Выберите способ сглаживания:", padx=5, pady=5)
112 lb3.pack(side=LEFT)
113
114 cmb3 = ttk.Combobox(str5, values=smoothing, state="readonly")
115 cmb3.pack(side=LEFT)
116
117 lb4 = Label(str6, text="Размер окна:", padx=5, pady=5)
118 lb4.pack(side=LEFT)
119
120 ntr1 = Entry(str6, width=15)
121 ntr1.pack(side=LEFT)
122
123 lb5 = Label(str6, text="Коэффициент:", padx=5, pady=5)
124 lb5.pack(side=LEFT)
125
126 ntr2 = Entry(str6, width=15)
127 ntr2.pack(side=LEFT)
128
129 btn1 = Button(str7, text="Построить график", command=Select, padx=5, pady=5)
130 btn1.pack(side=LEFT)
131
132 button = Button(str7, text="Отчистить", command=Close, padx=5, pady=5)
133 button.pack(side=LEFT)
134
135 window.mainloop()
136

```

Рисунок 6

Тут есть 3 основные функции. Главной является Select. Она вызывается при нажатии на кнопку “Построить” и обрабатывает все данные, введенные пользователем, вызывая необходимые функции для обработки данных, и вызывает функцию Generate. Она в свою очередь отвечает за построение графика. Принимает на вход три массива сформированных

в Select: quot – котировки акций, rst – восстановленные котировки, smt – сглаженные данные. И последняя функция Close отчищает Canvas если требуется удалить график.

Теперь перейдем к попке Modules, где находится функция для получения котировок с Московской биржи (Рис. 7)

```
1 import requests
2 from datetime import datetime
3 import pandas as pd
4
5 def getData(ticker, start, end):
6     data = []
7     date = []
8     quot = []
9     last_date = start
10
11     d1 = datetime.strptime(start, '%Y-%m-%d')
12     d2 = datetime.strptime(end, '%Y-%m-%d')
13     res = pd.date_range(d1, d2).strftime('%Y-%m-%d').tolist()
14     for i in res:
15         data.append([i, None])
16
17     while last_date < end:
18         last_date = start
19         kat = requests.get('https://iss.moex.com/iss/history/engines/stock/markets/shares/boards/TQBR/securities/' +
20                             ticker + '.json?from=' + start + '&till=' + end + '&history.columns=TRADEDATE,OPEN&iss.meta=off')
21         for i in range(len(kat.json()["history"]["data"])):
22             date.append(kat.json()["history"]["data"][i][0])
23             quot.append(kat.json()["history"]["data"][i][1])
24             start = kat.json()["history"]["data"][-1][0]
25             if start == last_date:
26                 del date[-1]
27                 del quot[-1]
28                 break
29         for i in range(len(data)):
30             for j in range(len(date)):
31                 if data[i][0] == date[j]:
32                     data[i][1] = quot[j]
33         for i in range(len(data)):
34             quot.append(data[i][1])
35     return quot
```

Рисунок 7

В функции getData мы принимаем тикер и временной отрезок, введенный пользователем. Далее создается 3 массива, в первый записываются все даты за отведенный период с дополнительным полем None для дальнейшего заполнения ценами. Во второй мы записываем даты полученные через запрос на Мосбиржу, а в третий – котировки в эту дату. Сам запрос мы обрабатываем циклом т.к за один запрос к API без подписки можно получить только 100 значений. Далее мы сравниваем даты полученные через запрос и даты из первого массива и заполняем первый массив соответствующими котировками. Далее перезаписываем массив котировок с уже пропущенными датами в дни когда Мосбиржа не работала. Возвращаем этот массив.

Далее рассмотрим функции для восстановления утерянных данных (Рис 8)

```

1  #Восстановление аппроксимацией
2  def aprok(s, f, arr):
3      k = 0
4      if f == len(arr):
5          f -= 1
6      if arr[f] == None:
7          k = arr[s - 1] - arr[s - 2]
8          b = arr[s - 1] - k * (s - 1)
9      else:
10         if s == 0:
11             b = arr[f]
12         else:
13             k = (arr[f] - arr[s - 1]) / (f - s + 1)
14             b = arr[s - 1] - k * (s - 1)
15     for i in range(s, f):
16         if int(k * i + b) >= 0:
17             arr[i] = int(k * i + b)
18         else:
19             arr[i] = 0
20     if arr[f] == None:
21         arr[f] = int(k * f + b)
22
23     def aprokRest(array):
24         i = 0
25         s = 0
26         f = 0
27         while i < len(array):
28             while i < len(array) and array[i] == None:
29                 i += 1
30                 f = i
31             if s != f:
32                 aprok(s, f, array)
33             i += 1
34             s = i
35             f = i
36         return array
37
38     #Винзорирование
39     def venz(array):
40         for i in range(len(array)):
41             while array[i] == None:
42                 n = array[i-1]
43                 array[i] = n
44         return array

```

Рисунок 8

Подробное описание восстановления аппроксимацией было приведено в предыдущих лабораторных работах. Поэтому в двух словах. Мы находим пропущенные промежутки и для них крайне точки. По этим точкам строим уравнение прямой. И по коэффициентам восстанавливаем значения. С винзорированием еще проще, мы находим пропущенные значения и приравниваем их к последним встретившимся.


```

1  #сглаживание методом скользящего среднего
2  def smt1(array, k):
3      smt = []
4      for i in range(len(array)):
5          j = 0
6          while (i + j < len(array)) and (2 * j < k) and (i - j >= 0):
7              j += 1
8          smt.append(sum(array[i - j + 1: i + j])/(2 * j - 1))
9      return smt
10
11 #метод скользящего среднего со скользящим окном наблюдения
12 def smt2(array, k):
13     print(array, k)
14     smt = []
15     for i in range(len(array)):
16         smt.append(smtFunc(array[:i + 1], k))
17     return smt
18
19 def smtFunc(array, k):
20     while abs(array[-1] - (sum(array) / len(array))) / array[-1] > k:
21         array.pop(0)
22     return sum(array) / len(array)

```

Рисунок 9

Рассмотрим функции сглаживания (рис. 9). Метод скользящего среднего представлен в первой функции. Она принимает массив и размер окна, затем высчитывает максимально допустимое окно, не превышающее указанного и по нему вычисляет среднее для каждого значения заполняя ими массив smt. Метод скользящего среднего со скользящим окном наблюдения представлен во второй функции. Она так же принимает массив и коэффициент. И для каждого элемента вызывает вспомогательную функцию smtFunc, которая в свою очередь принимает срез массива до самого числа и уменьшает полученное окно наблюдения до тех пор пока усредненное значение не будет отличаться от изначального в число, большее введенного коэффициента. Затем возвращает усредненное число в изначальную функцию, которая заполняет массив smt и возвращает его.

Вывод

Я научился работать с графическими интерфейсами в python. И смог применить все полученные знания алгоритмов сглаживания и восстановления, которые получил при выполнении предыдущих лабораторных работ.

Ссылка на гит

1. <https://github.com/SashaZharov/programming-lab4>