

Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»
(Университет ИТМО)

Кафедра Систем Управления и Информатики

Лабораторная работа №4

Вариант №6

Выполнил:

Таякин Д.Р.

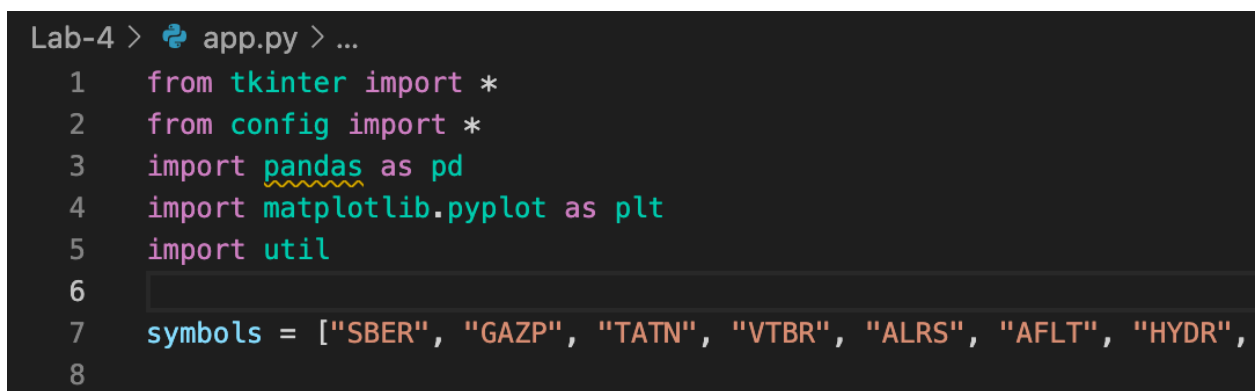
Проверил:

Мусаев А.А.

Санкт-Петербург, 2022

1 Задание

В первом и единственном задании нужно реализовать графический интерфейс, используя данные котировок акций из 3-й лабораторной работы. Помимо интерфейса программа должна уметь восстанавливать и сглаживать входные данные разными методами. Взвешенный метод скользящего среднего, метод скользящего среднего со скользящим окном наблюдения - для сглаживания. Винзорирование, линейная аппроксимация, корреляция для восстановления.



```
Lab-4 > app.py > ...
1  from tkinter import *
2  from config import *
3  import pandas as pd
4  import matplotlib.pyplot as plt
5  import util
6
7  symbols = ["SBER", "GAZP", "TATN", "VTBR", "ALRS", "AFLT", "HYDR",
8
```

Рисунок 1 – Импортируемые модули и перечисленные акции

На рисунке 1 изображены импортируемые модули. “config” и “util” являются модулями, которые были сделаны мной. В util модуле реализованы полезные функции такие как МНК, нахождение ближайших индексов справа и слева от пропущенного значения, случайное удаление данных из входного массива.

На рисунке 2 изображен метод винзорирования, который восстанавливает пропущенные значения заменяя предыдущим значением. Если предыдущего не нашлось, то ищется следующее рядом стоящее значение.

```

9      # recovery methods
10     # - winsoring method
11     def winsoring(data: list):
12         data_copy = data.copy()
13         prev_val = None; next_val = None
14         for (i, x) in enumerate(data_copy):
15             if x == None:
16                 j = i+1
17                 while j < len(data_copy):
18                     if data_copy[j] != None:
19                         next_val = data_copy[j]; break
20                     j += 1
21             else:
22                 prev_val = x
23
24         data_copy[i] = prev_val or next_val
25
26     return data_copy
27

```

Рисунок 2 – Метод винзорирования

```

28     # - linear approximation
29     def linear_approximation(data: list):
30         data_copy = data.copy()
31
32         indices = [i for i, x in enumerate(data_copy) if x == None]
33         for i in indices:
34             # find nearest indexes to get an approximation
35             prv, nxt = util.find_nearest_indexes(data_copy, i)
36
37             a, b = util.mnk([prv, nxt], [data_copy[prv], data_copy[nxt]])
38
39             r_y = a*i + b
40             data_copy[i] = r_y
41
42     return data_copy

```

Рисунок 3 – Метод линейной аппроксимации

На рисунке 3 представлен метод линейной аппроксимации, который находит два ближайших значения справа и слева от пропущенного значения и находит коэффициенты a и b при помощи метода наименьших квадратов. Далее, находятся пропущенные значения через линейную функцию $y(x) = ax + b$.

На рисунке 4 изображен метод корреляционного восстановления данных. Данный метод берет два массива и за счет корреляции восстанавливает значения.

```

44 # - correlation
45 def correlation(values1, values2):
46     for i, _ in enumerate(values1):
47         if values1[i] == None:
48             if i == len(values1) - 1:
49                 p1 = values1[i - 1]
50                 v1 = values2[i - 1]
51                 v2 = values2[i]
52                 values1[i] = p1 * v1 / v2
53             else:
54                 p2 = values1[i + 1]
55                 v1 = values2[i]
56                 v2 = values2[i + 1]
57                 values1[i] = p2 * v2 / v1
58         elif values2[i] == None:
59             if i == len(values2) - 1:
60                 p1 = values1[i - 1]
61                 p2 = values1[i]
62                 v1 = values2[i - 1]
63                 values2[i] = p1 * v1 / p2
64             else:
65                 p1 = values1[i]
66                 p2 = values1[i + 1]
67                 v2 = values2[i + 1]
68                 values2[i] = p2 * v2 / p1
69     return values1, values2
70

```

Рисунок 4 – Метод корреляционного восстановления

```

75 # smoothing methods
76 # - moving average with windowing
77 # https://wiki.loginom.ru/articles/windowing-method.html
78 def MAW(Y, k=2):
79     n = len(Y)
80     temp_Y = Y[0:k]
81     for t in range(k, n):
82         slice_Y = Y[t-k:t]
83         temp_Y.append(sum(slice_Y) / len(slice_Y))
84     return temp_Y
85
86 def WMA(data):
87     data_copy = data.copy()
88
89     # количество значений исходной функции для расчёта скользящего среднего
90     n = 2
91
92     for t in range(n, len(data)):
93         sum_data = 0
94         for i in range(n):
95             sum_data += (n-i)*data[t-i]
96
97         for i in range(n):
98             data_copy[t] = (2 / (n * (n+1))) * sum_data
99
100     return data_copy
101

```

Рисунок 5 – Методы сглаживания данных

На рисунке 5 представлен код двух методов сглаживания данных: метод скользящего среднего со скользящим окном наблюдения, взвешенный метод скользящего среднего соответственно.

В первом методе использована формула: $\frac{1}{n} \sum_{t=k}^{n+k} X(t)$.

Во втором использована формула: $\frac{2}{n(n+1)} \sum_{i=0}^{n-1} (n-i) * p_{t-i}$.

```

114 def App():
115
116     root = Tk()
117     root.title("Начинающий Брокер")
118     root.geometry("500x200")
119     root.resizable(width=False, height=False)
120
121     frame = Frame(root, bg=bg_color)
122     frame.place(relwidth=1, relheight=1)
123
124     def option_menu(data: list, default_value) -> StringVar:
125         variable = StringVar(frame)
126         variable.set(default_value)
127         OptionMenu(frame, variable, *data).pack()
128         return variable
129
130     symbol_variable = option_menu(symbols, "Select Ticket")
131     recovery_variable = option_menu(["Winsoring", "Linear approximation", "Correlation"], "Select Recovery Method")
132     antialiasing_variable = option_menu(["Moving Average", "Moving Average w/ Windowing"], "Select Anti-aliasing Method")
133
134     title = Label(frame, text="Select second ticker if you've selected Correlation recovery method.", bg=bg_color)
135     title.pack()
136
137     symbol2_variable = option_menu(symbols, "Select Ticket")
138

```

Рисунок 6 – Первая часть кода интерфейса и обработки данных

На рисунке 6 изображена первая часть кода, в которой осуществляется интерфейс приложения. Функция `option_menu` - создает меню выбора.

На рисунке 7 изображена вторая часть кода приложения. `btn_click` является обработчиком при нажатии на кнопку Build. В первых четырех строках данного метода, мы принимаем значения с нескольких меню выбора. Далее, мы считываем нужный нам файл с данными акций компании. В зависимости от выбранных методов восстановления и сглаживания, мы получаем массивы данных. В последних строках кода реализуется вывод данных в графики.

Интерфейс приложения можно увидеть на рисунке 8. Вывод графиков на рисунке 9.

```

138
139 def build_click():
140     symbol = symbol_variable.get()
141     symbol2 = symbol2_variable.get()
142     recovery = recovery_variable.get()
143     antialising = antialising_variable.get()
144
145     df = pd.read_csv(f"./data/{symbol}.csv", sep=';', names=['date', 'price', 'change', 'cap'])
146     # prices for 18 months
147     i=0
148     prices = util.rand_remove(list(df['price'][i:i+18]))
149
150     recovered_prices = []
151     if recovery == "Winsoring":
152         recovered_prices = winsoring(prices)
153     elif recovery == "Linear approximation":
154         recovered_prices = linear_approximation(prices)
155     elif recovery == "Correlation":
156         df = pd.read_csv(f"./data/{symbol}.csv", sep=';', names=['date', 'price', 'change', 'cap'])
157         # prices for 18 months
158         i=0
159         prices2 = util.rand_remove(list(df['price'][i:i+18]))
160         recovered_prices = correlation(prices, prices2)[0]
161
162     antialised_prices = []
163     if antialising == "Moving Average":
164         antialised_prices = MAW(recovered_prices, k=2)
165     elif antialising == "Moving Average w/ Windowing":
166         antialised_prices = WMA(recovered_prices)
167
168     # show what we've done
169     fig, axs = plt.subplots(3)
170     axs[0].set_title("Initial data")
171     axs[0].plot(range(len(prices)), prices, color="#CE7A60")
172     axs[1].set_title("Recovered data")
173     axs[1].plot(range(len(recovered_prices)), recovered_prices, color="#CE7A60")
174     axs[2].set_title("Antialised data")
175     axs[2].plot(range(len(antialised_prices)), antialised_prices, color="#FE7A60")
176
177     plt.text(0.5, 0.5, f" $\sigma$  = {std_deviation(recovered_prices, antialised_prices)}")
178
179     fig.tight_layout()
180     plt.show()
181
182     build_btn = Button(frame, text="Build", bg=bg_color, borderwidth=0, command=build_click)
183     build_btn.pack()
184
185     root.mainloop()

```

Рисунок 7 – Вторая часть кода интерфейса и обработки данных

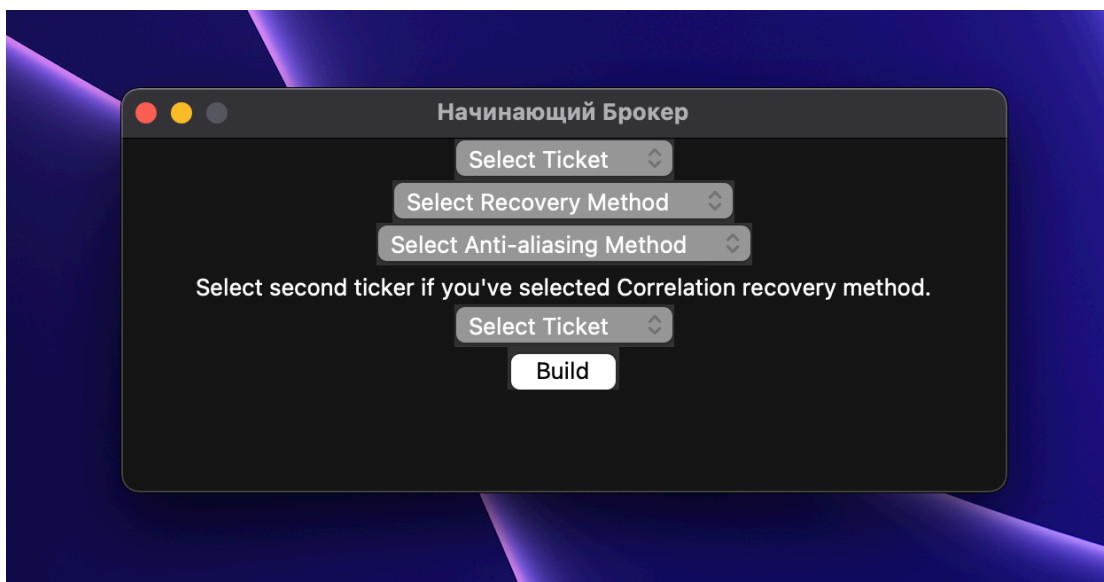


Рисунок 8 – Интерфейс приложения

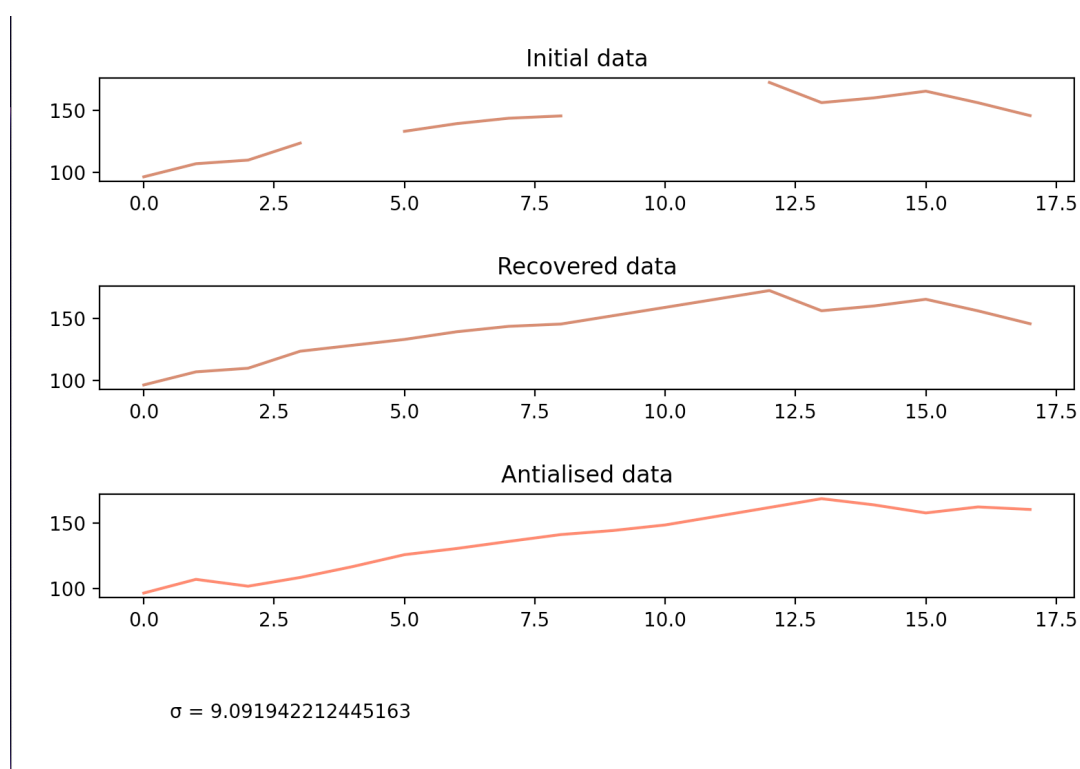


Рисунок 9 – Вывод итоговых графиков