Санкт-Петербургский Национальный Исследовательский Университет Информационных Технологий, Механики и Оптики

Кафедра Систем Управления и Информатики

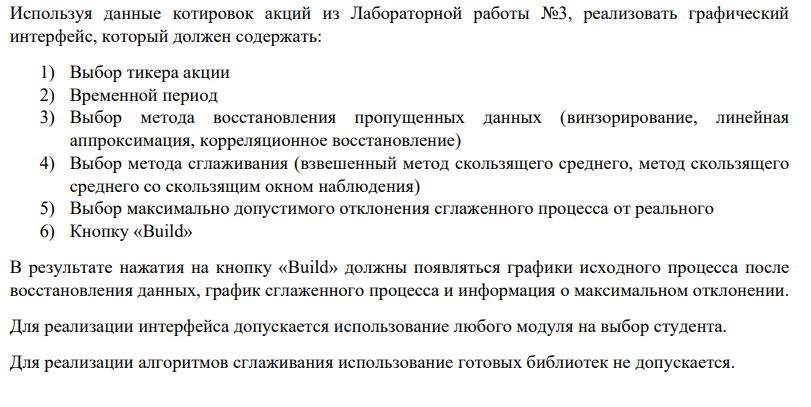
# Лабораторная работа №4

Выполнил(и:) Байков Иван

Проверил Мусаев А.А.

Санкт-Петербург, 2022

**Задание 1**



import yfinance as yf  
import matplotlib.pyplot as plt  
import random  
import math  
from random import randint  
import numpy as np  
  
  
def destroyer(array, count):  
 for i in range(count):  
 while True:  
 x = random.randint(0, len(array) - 1)  
 if array[x] != None: break  
 array[x] = None  
 print("destroyer works")  
 return array  
  
  
# approx  
def approximation(a):  
 for x in range(len(a)):  
 if a[x] == None:  
 i, j = 1, 1  
 i2, j2 = 1, 1  
 flag1 = False  
 flag2 = False  
 counter = 0  
 counter2 = 0  
  
 while True and not flag1:  
 if x + i < len(a):  
 if a[x + i] is not None:  
 y2 = a[x + i]  
 x2 = x + 1 + i  
 break  
 i += 1  
  
 else:  
 if counter < 2:  
 while True:  
 if x - j >= 0:  
 if a[x - j] is not None:  
 counter += 1  
 if counter == 2:  
 y2 = a[x - j]  
 x2 = x + 1 - j  
 flag1 = True  
 break  
 j += 1  
  
 while True and not flag2:  
 if x - i2 >= 0:  
 if a[x - i2] is not None:  
 y1 = a[x - i2]  
 x1 = x + 1 - i2  
 break  
 i2 += 1  
  
 else:  
 if counter2 < 2:  
 while True:  
 if x + j2 < len(a):  
 if a[x + j2] is not None:  
 counter2 += 1  
 if counter2 == 2:  
 y1 = a[x + j2]  
 x1 = x + 1 + j2  
 flag2 = True  
 break  
 j2 += 1  
  
 k = (y2 - y1) / (x2 - x1)  
 b = y1 - (k \* x1)  
 a[x] = k \* (x + 1) + b  
 print("approx works")  
 return a  
  
  
# винзолирование  
def vinz(array):  
 for i in range(len(array)): recover(i, array)  
 print("vinz works")  
 return array  
  
  
def correlation(array, array2):  
 list\_y = array  
 list\_x = []  
 sum\_x = 0  
 sum\_y = 0  
 list\_x = array2  
  
 # for i in range(1, len(array) + 1):  
 # list\_x.append(i)  
 copied\_array = array.copy()  
 zero\_counter = 0  
 for i in range(len(list\_x)):  
 if list\_y[i] != 0:  
 sum\_x += list\_x[i]  
 sum\_y += list\_y[i]  
 else:  
 zero\_counter += 1  
  
 average\_x = sum\_x / (len(list\_x) - zero\_counter)  
 average\_y = sum\_y / (len(list\_y) - zero\_counter)  
 disp\_sum\_up = 0  
 disp\_sum\_down\_x = 0  
 disp\_sum\_down\_y = 0  
  
 for i in range(len(list\_x)):  
 if list\_y[i] != 0:  
 disp\_sum\_up += (list\_x[i] - average\_x) \* (list\_y[i] - average\_y)  
 disp\_sum\_down\_x += (list\_x[i] - average\_x) \*\* 2  
 disp\_sum\_down\_y += (list\_y[i] - average\_y) \*\* 2  
 r = disp\_sum\_up / ((disp\_sum\_down\_x \* disp\_sum\_down\_y) \*\* 0.5)  
 print("r= ", r)  
  
 for i in range(len(list\_x)):  
 if copied\_array[i] == 0:  
 if r > 0:  
 copied\_array[i] = r \* (list\_x[i] \* average\_y) / average\_x  
 else:  
 copied\_array[i] = (-1 / r) \* (list\_x[len(list\_x) - i - 1] \* average\_y) / average\_x  
  
 return copied\_array  
  
  
# восстановление числа по индексу  
def recover(index, array):  
 dop\_index = 0  
 while array[index] == None:  
 if index - dop\_index > 0 and array[index - dop\_index] != None: array[index] = array[index - dop\_index]  
 if index + dop\_index < len(array) and array[index + dop\_index] != None: array[index] = array[index + dop\_index]  
 dop\_index += 1  
  
  
def smoothing(array, k):  
 res = []  
 for i in range(len(array)): res.append(smoothing\_for\_element(array[:i + 1], k))  
 print("Smoothing method = 2nd")  
 return res  
  
  
# сглаживание для 1 элемента  
def smoothing\_for\_element(window, k):  
 while math.fabs(window[-1] - (sum(window) / len(window))) / window[-1] > k: window.pop(0)  
 return sum(window) / len(window)  
  
  
# --------------------------------------------------------------------  
  
def smoothing\_lite(array, k):  
 res = []  
 a = 1  
 for i in range(len(array)):  
 if i < k:  
 a = k - i  
 res.append(smoothing\_for\_element\_lite(array[0:i + a], k))  
 else:  
 res.append(smoothing\_for\_element\_lite(array[i - k:i + a], k))  
 print("Smoothing method = 1nd")  
 return res  
  
  
# сглаживание для 1 элемента  
def smoothing\_for\_element\_lite(window, k):  
 return sum(window) / len(window)  
  
  
def smoothingChoose(index, plot, k, c):  
 if index == 5:  
 plot = smoothing(plot, k)  
 if index == 4:  
 plot = smoothing\_lite(plot, c)  
 return plot  
  
  
ticker\_list = ['OGZPY', 'TATN.ME', 'SBER.ME', 'VTBR.ME', 'ALRS.ME', 'AFLT.ME', 'HYDR.ME',  
 'MOEX.ME', 'NLMK.ME', 'CHMF.ME', 'DSKY.ME', 'POLY.ME', 'YNDX', 'AFKS.ME',  
 'LSRG.ME', 'LSNG.ME', 'LKOH.ME', 'MTSS.ME', 'NVTK.ME', 'PIKK.ME']  
  
  
# 1 Выбор тикера"  
# 2 Временной период  
# 3 Метод восстановления  
# 4 Коэффициент  
# 5 Метод сглаживания  
  
  
def clicked():  
 sel1 = selected.get()  
 sel2 = selected2.get()  
 print(sel1)  
 print(sel2)  
  
 array = {}  
 plot = []  
 ticker = combo\_ticker.get()  
 start = txt.get()  
 end = txt2.get()  
 print("date = ", start, end)  
 data = yf.download(ticker, start=start, end=end, interval="1d")  
 array[ticker] = data['Adj Close']  
 for i in range(len(array[ticker])):  
 plot.append(array[ticker][i])  
 if sel1 == 1:  
 plot = vinz(destroyer(plot, 450))  
 if sel2 == 2:  
 plot = approximation(destroyer(plot, 20))  
 if selected == 3:  
 plot2\_ticker = ticker\_list[1]  
 if plot2\_ticker == ticker:  
 plot2\_ticker = ticker\_list[2]  
 plot2 = []  
 for i in range(len(array[plot2\_ticker])):  
 plot2.append(array[plot2\_ticker][i])  
 plot = correlation(destroyer(plot, 20), plot2)  
 k = float(txtK.get())  
 c = 5  
 sm\_plot = smoothingChoose(sel2, plot, k, c)  
 plt.plot(plot)  
 plt.show()  
 plt.plot(sm\_plot)  
 plt.show()  
  
  
from tkinter import \*  
from tkinter.ttk import Combobox  
  
Iwindow = Tk()  
Iwindow.title("Shares analysis")  
lbl = Label(Iwindow, text="Choose token")  
lbl.grid(column=0, row=0)  
Iwindow.geometry('820x460')  
btn = Button(Iwindow, text="Don't touch!", command=clicked)  
btn.grid(column=2, row=5)  
  
combo\_ticker = Combobox(Iwindow)  
combo\_ticker['values'] = tuple(ticker\_list)  
combo\_ticker.current(0)  
combo\_ticker.grid(column=0, row=1)  
  
selected = IntVar()  
selected2 = IntVar()  
  
lblRecover = Label(Iwindow, text="Select method")  
lblRecover.grid(column=1, row=0)  
rad1 = Radiobutton(Iwindow, text='Винзовирование', value=1, variable=selected)  
rad2 = Radiobutton(Iwindow, text='Линейная аппроксимация', value=2, variable=selected)  
rad3 = Radiobutton(Iwindow, text='Корреляция', value=3, variable=selected)  
rad1.grid(column=1, row=1)  
rad2.grid(column=1, row=2)  
rad3.grid(column=1, row=3)  
  
lblSmooth = Label(Iwindow, text="Select smoothing method")  
lblSmooth.grid(column=2, row=0)  
rad4 = Radiobutton(Iwindow, text='Скользящее среднее', value=4, variable=selected2)  
rad5 = Radiobutton(Iwindow, text='Скользящее среднее с динамическим окном', value=5, variable=selected2)  
rad4.grid(column=2, row=1)  
rad5.grid(column=2, row=2)  
  
lblDate = Label(Iwindow, text="Enter Date Start")  
lblDate.grid(column=1, row=4)  
txt = Entry(Iwindow, width=20)  
txt.grid(column=1, row=5)  
  
lblDate = Label(Iwindow, text="Enter Date End")  
lblDate.grid(column=1, row=6)  
txt2 = Entry(Iwindow, width=20)  
txt2.grid(column=1, row=7)  
  
lbltxtK = Label(Iwindow, text="Enter K param")  
lbltxtK.grid(column=0, row=4)  
txtK = Entry(Iwindow, width=20)  
txtK.grid(column=0, row=5)  
  
Iwindow.mainloop()

Для реализации графического интерфейса был использован стандартный Python модуль Tkinter. Функция Tk() создает окно интерфейса. Tk().title() добавляет заголовок, Label() – текст внутрь интерфейса. Для реализации выпадающего списка использовано Combobox(), для кнопок – Button(), для выбора способа восстановления данных – Radiobutton. Поле ввода даты и коэффициента сглаживания реализовано через Entry(). Для размещения элементов на окне используется сетка, где задаются column и row для выбора номера столбца и строки. Показ интерфейса вызывается методом .mainloop().

Весь функционал реализован без использования готовых библиотек, для получения котировок акций использовался сервис Yahoo!Finance.

**Результат выполнения**

