Приложение

Листинг

```
import numpy as np
import math as m
import skfuzzy as fuzz
import matplotlib.pyplot as plt

coef = [-1, 1, 3, 5, 7]
“””
t – массив значений критерия Стьюдента для уровня значимости
0.05
“””
t =[12.71, 4.3, 3.18, 2.78, 2.57, 2.45, 2.37, 2.31, 2.26, 2.23,
2.20, 2.18]

“””
Функция для поиска наименьшего элемента в матрице
“””
def min_in_matrix(mat):
    min_elem = mat[0,0]
    for i in range(0,len(mat)-1):
        for j in range(0,len(mat)-1):
            if mat[i,j]<min_elem:
                min_elem=mat[i,j]
    return min_elem

“””
Функция для поиска наибольшего элемента в матрице
“””
def max_in_matrix(mat):
    max_elem = mat[0,0]
    for i in range(0,len(mat)-1):
        for j in range(0,len(mat)-1):
            if mat[i,j]>max_elem:
                max_elem=mat[i,j]
    return max_elem
```

```
“””
Функция, возвращающая конкретную матрицу year_matrix с доходами
для определенного параметра year
“””
def matrix_for_one_year(dataframe, year, c):
    rows, columns = np.shape(dataframe)
    year_matrix = np.zeros((4,3))
    for i in range((year-1)*3,year*3):
        year_matrix[:,(i-year-c)] = dataframe[:,i]
    return year_matrix

“””
Функция, возвращающая ближайшее четкое множество для переданной
нечеткой матрицы  year_matrix
“””
def simple_plenty(year_matrix):
    simple_matrix = np.zeros((4,2))
    simple_matrix[:,0] = year_matrix[:,0] +
0.5*(year_matrix[:,1]-year_matrix[:,0])
    simple_matrix[:,1] = year_matrix[:,2] +
0.5*(year_matrix[:,1]-year_matrix[:,2])
    return simple_matrix

“””
Функция находит среднее значение по столбцам переданной матрицы
“””
def M_x(mat,experts):
    M = [sum(mat[:,0])/experts, sum(mat[:,1])/experts,
sum(mat[:,2])/experts]
    return M

“””
Функция находит пересечение обычных множеств и возвращает длину
найденного пересечения
“””
def intersection(mat):
    inters = [max(mat[:,0]), min(mat[:,1])]
    difference = inters[1] - inters[0]
    return difference
```

```
“””
функция находит объединение обычных множеств и возвращает длину
найденного объединения
“””
def union(mat):
    unions = [min(mat[:,0]), max(mat[:,1])]
    difference = unions[1] - unions[0]
    return difference

“””
функция возвращает матрицу согласованности экспертных критериев
“””
def consistency_matrix(mat,experts):
    row,col = mat.shape
    little_mat = np.zeros((2,2))
    mat_coef = np.eye(experts)
    for i in range(0,row-1):
        for j in range(i+1,row):
            little_mat[0] = mat[i]
            little_mat[1] = mat[j]
            mat_coef[i,j] =
intersection(little_mat)/union(little_mat)
            mat_coef[j,i] =
intersection(little_mat)/union(little_mat)
    return mat_coef

“””
функция вычисляет сумму по строкам для матрицы согласованности
экспертных критериев
“””
def sum_opinion(mat):
    sum_op = []
    for i in range(0,len(mat)):
        sum_op.append(sum(mat[:,i]))
    return sum_op
```

```
“””
Функция проверки коэффициента общей согласованности, возвращает
экспертные критерии и обычные множества, которые будут
использоваться в дальнейшем анализе
“””
def consistency_check(s_m,m):
    k = intersection(s_m)/union(s_m)
    if (k>=0.7):
        return(s_m,m)
    else:
        mark = sum_opinion(consistency_matrix(s_m, len(s_m)))
        j = mark.index(min(mark))
        s_m = np.delete(s_m, j, axis=0)
        m = np.delete(m, j, axis=0)
        return(consistency_check(s_m,m))

“””
Функция, которая приводит экспертные критерии к обобщенному
экспертному критерию в виде интервального нечеткого числа
второго типа
“””def f():
    consistency_coeff = []
    f_up = []
    f_down = []
    for i in range(1,duration+1):
        temp_matrix = matrix_for_one_year(data, i, coef[i-1])
        simple_matrix = simple_plenty(temp_matrix)
        simple_matrix, temp_matrix =
consistency_check(simple_matrix, temp_matrix)
        experts = len(temp_matrix)
        M_boards = M_x(temp_matrix,experts)
     consistency_coeff.append(intersection(simple_matrix)/union(
simple_matrix))
        sl = m.sqrt((1/(experts - 1))*(sum((temp_matrix[:,0] -
M_boards[0])**2)))
        sr = m.sqrt((1/(experts - 1))*(sum((temp_matrix[:,2] -
M_boards[2])**2)))
        f_up.append(M_boards[0] - sl*(t[experts-
2])/(m.sqrt(experts)))
        f_up.append(M_boards[1])
```

```
        f_up.append(M_boards[2] + sr*(t[experts-
2])/(m.sqrt(experts)))
        f_down.append(M_boards[0] + sl*(t[experts-
2])/(m.sqrt(experts)))
        f_down.append(M_boards[1])
        f_down.append(M_boards[2] - sr*(t[experts-
2])/(m.sqrt(experts)))
    return consistency_coeff, np.array(f_up), np.array(f_down)

“””
Функция находит сумму доходов для каждого параметра
интервального нечеткого числа
“””def cf(mas):
    cf_l = 0
    cf_m = 0
    cf_r = 0
    for i in range(0,3*duration,3):
        cf_l+=mas[i]
        cf_m+=mas[i+1]
        cf_r+=mas[i+2]
    return np.array((cf_l,cf_m,cf_r))

“””
Функция для подсчета интервального нечеткого NPV
“””def fuzzy_npv():
    for i in range(1, duration+1):
        fup[(i-1)*3:i*3] = fup[(i-
1)*3:i*3]/fuzzy_discount_rate[i-1]
        fdown[(i-1)*3:i*3] = fdown[(i-
1)*3:i*3]/fuzzy_discount_rate[i-1]
    f_up = [cf(fup)[0] - i_up[2], cf(fup)[1] - i_up[1],
cf(fup)[2] - i_up[0]]
    f_down = [cf(fdown)[0] - i_down[2], cf(fdown)[1] -
i_down[1], cf(fdown)[2] - i_down[0]]
    return f_up,f_down
```

```
“””функция проверки среднего значения NPV для определения
эффективности проекта
“””
def check_project():
    if (E<0):
        print('Данный проект не следует принимать')
    elif (E==0):
        print('Принятие проекта не принесет как прибыли, так и
убытков')
    elif (E>0):
        print('Данный проект можно принять')

“””
Построение графиков
“””
#составление массивов, по которым будут строиться графики
up = np.arange(npv_up[0]-10, npv_up[2]+10,10)
up_lo = fuzz.trimf(up, [npv_up[0], npv_up[1], npv_up[2]])
down = np.arange(npv_down[0]-10, npv_down[2]+10,5)
down_lo = fuzz.trimf(down, [npv_down[0], npv_down[1],
npv_down[2]])
x=[npv_down[1],npv_down[1]]
y = [0,1]

#создание области для построения графиков
fig, (ax0) = plt.subplots(nrows=1, figsize=(10, 8))
ax0.plot(up, up_lo, color='purple', linewidth=1.5,
label='UP_NPV')
ax0.plot(down, down_lo, color='purple', linewidth=1.5,
label='DOWN_NPV')
ax0.plot(x,y,color='orange', linestyle='--')
ax0.set_title('Fuzzy NPV')
ax0.legend()
fig.savefig('picture01.png')


“””
Построение вспомогательных графиков (экспертные критерии)
“””
for i in range(1,duration+1):
```

```
    temp_matrix = matrix_for_one_year(data, i, coef[i-1])
    expert = np.arange(min_in_matrix(temp_matrix)-20,
max_in_matrix(temp_matrix)+20,10)
    exp_1_lo = fuzz.trimf(expert, [temp_matrix[0,0],
temp_matrix[0,1], temp_matrix[0,2]])
    exp_2_lo = fuzz.trimf(expert, [temp_matrix[1,0],
temp_matrix[1,1], temp_matrix[1,2]])
    exp_3_lo = fuzz.trimf(expert, [temp_matrix[2,0],
temp_matrix[2,1], temp_matrix[2,2]])
    exp_4_lo = fuzz.trimf(expert, [temp_matrix[3,0],
temp_matrix[3,1], temp_matrix[3,2]])
    fig, (ax1) = plt.subplots(nrows=1, figsize=(12, 5))

    ax1.plot(expert, exp_1_lo, 'b', linewidth=1.5, label='Первый
эксперт')
    ax1.plot(expert, exp_2_lo, 'r', linewidth=1.5, label='Второй
эксперт')
    ax1.plot(expert, exp_3_lo, 'g', linewidth=1.5, label='Третий
эксперт')
    ax1.plot(expert, exp_4_lo, 'y', linewidth=1.5,
label='Четвертый эксперт')
    ax1.set_title('Мнения экспертов за '+str(i)+' год:')
    ax1.legend()

from PyQt5 import QtCore, QtGui, QtWidgets

class first_page(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("first")
        MainWindow.resize(530, 335)
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.gridLayout =
QtWidgets.QGridLayout(self.centralwidget)
        self.gridLayout.setObjectName("gridLayout")
        self.project_name = QtWidgets.QLabel(self.centralwidget)
        self.project_name.setObjectName("label")
        self.gridLayout.addWidget(self.project_name, 0, 0, 1, 1)
        self.project_name =
QtWidgets.QLineEdit(self.centralwidget)
```

```
        self.project_name.setObjectName("project_name")
        self.gridLayout.addWidget(self.project_name, 1, 0, 1, 1)
        self. duration = QtWidgets.QLabel(self.centralwidget)
        self. duration.setObjectName("duration ")
        self.gridLayout.addWidget(self. duration, 2, 0, 1, 1)
        self.duration = QtWidgets.QLineEdit(self.centralwidget)
        self.duration.setObjectName("duration")
        self.gridLayout.addWidget(self.duration, 3, 0, 1, 1)
        self.experts = QtWidgets.QLabel(self.centralwidget)
        self.experts.setObjectName("label_3")
        self.gridLayout.addWidget(self.experts, 4, 0, 1, 1)
        self.count_experts =
QtWidgets.QLineEdit(self.centralwidget)
        self.count_experts.setEnabled(True)
        self.count_experts.setMaximumSize(QtCore.QSize(520, 20))
        self.count_experts.setTabletTracking(False)
        self.count_experts.setObjectName("count_experts")
        self.gridLayout.addWidget(self.count_experts, 5, 0, 1,
1)
        self.discount_rate =
QtWidgets.QLabel(self.centralwidget)
        self.discount_rate.setObjectName("label_4")
        self.gridLayout.addWidget(self.discount_rate, 6, 0, 1,
1)
        self.discount_rate =
QtWidgets.QLineEdit(self.centralwidget)
        self.discount_rate.setObjectName("discount_rate")
        self.gridLayout.addWidget(self.discount_rate, 7, 0, 1,
1)
        self.next_but =
QtWidgets.QPushButton(self.centralwidget)
        self.next_but.setObjectName("NextPage")
        self.gridLayout.addWidget(self.next_but, 8, 1, 1, 1)
        MainWindow.setCentralWidget(self.centralwidget)

        self.retranslateUi(MainWindow)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)

    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
```

```
        MainWindow.setWindowTitle(_translate("first", "Оценка
эффективности инвестиционного проекта"))
        self.project_name.setText(_translate("first", "Введите
название инвестиционного проекта"))
        self.duration.setText(_translate("first", "Введите срок
реализации"))
        self.experts.setText(_translate("first", "Введите
количество экпертов, "

"участвующих в оценке инвестицонных параметров"))
        self.discount_rate.setText(_translate("first", "Введите
ставку дисконтирования"))
        self.next_but.setText(_translate("first", "Далее"))

class second_page(object):
    def setupUi(self, second):
        second.setObjectName("second")
        second.resize(544, 336)
        self.centralwidget = QtWidgets.QWidget(second)
        self.centralwidget.setObjectName("centralwidget")
        self.gridLayout =
QtWidgets.QGridLayout(self.centralwidget)
        self.gridLayout.setObjectName("gridLayout")
        self.contributions =
QtWidgets.QLabel(self.centralwidget)
        self.contributions.setObjectName("contributions ")
        self.gridLayout.addWidget(self.contributions, 0, 0, 1,
1)
        self.first_contributions =
QtWidgets.QTableWidget(self.centralwidget)

self.first_contributions.setObjectName("first_contributions ")
        self.first_contributions.setColumnCount(0)
        self.first_contributions.setRowCount(0)
        self.gridLayout.addWidget(self.first_contributions, 1,
0, 1, 1)
        self.next_sec =
QtWidgets.QPushButton(self.centralwidget)
        self.next_sec.setObjectName("next_sec")
```

```
        self.gridLayout.addWidget(self.next_sec, 2, 1, 1, 1)
        second.setCentralWidget(self.centralwidget)

        self.retranslateUi(second)
        QtCore.QMetaObject.connectSlotsByName(second)

    def retranslateUi(self, second):
        _translate = QtCore.QCoreApplication.translate
        second.setWindowTitle(_translate("second", "Оценка
эффективности инвестиционного проекта"))
        self.contributions.setText(_translate("second",
"Первоначальные взносы"))
        self.next_sec.setText(_translate("second", "Далее"))

class third_page(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("third")
        MainWindow.resize(546, 339)
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.gridLayout =
QtWidgets.QGridLayout(self.centralwidget)
        self.gridLayout.setObjectName("gridLayout")
        self.performance_evaluation =
QtWidgets.QLabel(self.centralwidget)

self.performance_evaluation.setObjectName("performance_evaluatio
n ")
        self.gridLayout.addWidget(self.performance_evaluation,
0, 0, 1, 1)
        self.expert_criteria =
QtWidgets.QTableWidget(self.centralwidget)
        self.expert_criteria.setObjectName("expert_criteria")
        self.expert_criteria.setColumnCount(0)
        self.expert_criteria.setRowCount(0)
        self.gridLayout.addWidget(self.expert_criteria, 1, 0, 1,
1)
        self.next_but =
QtWidgets.QPushButton(self.centralwidget)
        self.next_but.setObjectName("pushButton")
```

```
        self.gridLayout.addWidget(self.next_but, 2, 1, 1, 1)
        MainWindow.setCentralWidget(self.centralwidget)

        self.retranslateUi(MainWindow)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)

    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("third", "Оценка
эффективности инвестиционного проекта"))
        self.performance_evaluation.setText(_translate("third",
"Экспертные критерии"))
        self.next_but.setText(_translate("third", "Рассчитать"))

class forth(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("forth")
        MainWindow.resize(518, 330)
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.gridLayout =
QtWidgets.QGridLayout(self.centralwidget)
        self.gridLayout.setObjectName("gridLayout")
        self.result = QtWidgets.QLabel(self.centralwidget)
        self.result.setObjectName("result")
        self.gridLayout.addWidget(self.result, 0, 0, 1, 1)
        self.npv = QtWidgets.QTableWidget(self.centralwidget)
        self.npv.setObjectName("npv")
        self.npv.setColumnCount(0)
        self.npv.setRowCount(0)
        self.gridLayout.addWidget(self.npv, 1, 0, 1, 1)
        self.mean_npv = QtWidgets.QLineEdit(self.centralwidget)
        self.mean_npv.setObjectName("mean_npv")
        self.gridLayout.addWidget(self.mean_npv, 2, 0, 1, 1)
        self.conc = QtWidgets.QLineEdit(self.centralwidget)
        self.conc.setObjectName("conc")
        self.gridLayout.addWidget(self.conc, 3, 0, 1, 1)
        self.end_work =
QtWidgets.QPushButton(self.centralwidget)
        self.end_work.setObjectName("end_work")
```

```
        self.gridLayout.addWidget(self.end_work, 4, 1, 1, 1)
        MainWindow.setCentralWidget(self.centralwidget))

        self.retranslateUi(MainWindow)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)

    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("forth", "Оценка
эффективности инвестиционного проекта"))
        self.label.setText(_translate("forth", "Результаты
вычислений"))
        self.pushButton.setText(_translate("forth", "Закончить
работу"))
```