**Федеральное государственное образовательное бюджетное**

**учреждение**

**высшего образования**

**«ФИНАНСОВЫЙ УНИВЕРСИТЕТ**

**ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ»**

**(Финансовый университет)**

**Факультет**

информационных технологий и анализа больших данных

Кафедра «Бизнес-информатика»

**Домашнее задание № 7**

«Многокритериальная оптимизация»

Студенты группы БИ20-4:

Иванова Ксения

Киракосян Виген

Крылов Никита

Мытарева Ангелина

Петрова Арина

Чайковская Анна

Руководитель:

Аксенов Дмитрий Андреевич

**Москва 2022**

**ОГЛАВЛЕНИЕ**

[1.ФИЗИЧЕСКАЯ МОДЕЛЬ 3](#_Toc102087182)

[2.МАТЕМАТИЧЕСКАЯ МОДЕЛЬ 4](#_Toc102087183)

[3.АЛГОРИТМ 5](#_Toc102087184)

[4.АРХИТЕКТУРА РЕШЕНИЯ 11](#_Toc102087185)

[5.EXCEL 14](#_Toc102087186)

[6.ТЕСТИРОВАНИЕ 24](#_Toc102087187)

[7.ЗАКЛЮЧЕНИЕ 25](#_Toc102087188)

## ФИЗИЧЕСКАЯ МОДЕЛЬ

Рассмотрим конкретный кейс из жизни: студент ищет однокомнатную квартиру для съёма у здания Финансового университета на Рязанском проспекте. На сайте, специализирующемся на недвижимости, было найдено несколько вариантов квартир. В таблице представлены основные характеристики квартир:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| № | Стоимость(руб./мес.) | Площадь (м^2) | До метро (мин.) | Год постройки | Оценка постояльцев (от 1 до 10) | Этаж | Высота потолков (м) | Размер залога (руб.) |
| 1 | 35 000 | 30 | 15 | 2 010 | 6 | 6 | 2,7 | 10 000 |
| 2 | 47 000 | 39 | 10 | 2 015 | 7 | 9 | 2,5 | 15 000 |
| 3 | 42 500 | 35 | 12 | 2 009 | 5 | 4 | 3,0 | 13 000 |
| 4 | 65 000 | 43 | 10 | 2 017 | 8 | 8 | 2,7 | 30 000 |
| 5 | 38 000 | 35 | 10 | 2 013 | 5 | 5 | 2,8 | 12 000 |
| 6 | 50 500 | 44 | 15 | 2 008 | 6 | 9 | 2,5 | 20 000 |
| 7 | 48 000 | 35 | 10 | 2 006 | 7 | 4 | 3,0 | 14 000 |
| 8 | 58 000 | 42 | 10 | 2 010 | 8 | 8 | 2,8 | 28 000 |
| 9 | 54 500 | 39 | 13 | 2 018 | 7 | 6 | 2,7 | 22 000 |
| 10 | 45 000 | 40 | 17 | 2 013 | 5 | 9 | 3,0 | 17 000 |

Необходимо выбрать лучший вариант недвижимости.

## МАТЕМАТИЧЕСКАЯ МОДЕЛЬ

**Парето:**

Точка x ∗ ∈ X называется оптимальной по Парето (парето-оптимальной или эффективной), если не существует такой точки x ∈ X, для которой выполнено неравенство f(x) ≥ f(x ∗). При этом вектор f(x ∗) также именуют оптимальным по Парето или эффективным.

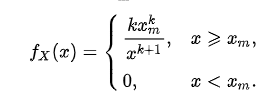
В случае одного критерия (m = 1) оптимальная по Парето точка превращается в точку максимума функции f1. Отсюда следует, что понятие Парето-оптимальной точки представляет собой прямое обобщение точки максимума скалярной числовой функции на случай векторного критерия.

Тогда формула для расчета распределения Парето выглядит следующим образом:





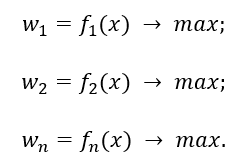
Тогда говорят, что Х имеет распределение Парето с параметрами  Плотность распределения Парето имеет вид:



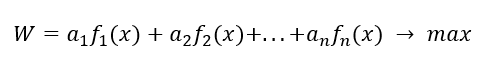
**ЛИНЕЙНАЯ СВЕРТКА ВСЕХ КРИТЕРИЕВ**

Для поиска единственного оптимального решения сведем задачу к однокритериальной и воспользуемся методом линейной свертки по критериям. Для этого необходимо присвоить каждому критерию показатель значимости в процентах из ста.

Переход от нескольких критериев к одному:



происходит путем их сворачивания и объединения в выражении:



где – весовые коэффициенты, отражающие значимость критериев:



Необходимо также учесть, что среди рассматриваемых критериев оптимальными значениями могут быть как максимальное среди имеющихся показателей критериев, так и минимальное. В случае, если оптимальным значением является минимальное, необходимо умножить коэффициент a на -1.

**Метод «идеальной точки»:**

В основу метода "идеальной точки" положен расчет расстояния в многомерном пространстве критериев между точкой, соответствующей идеальной альтернативе, и точкой, соответствующей рассматриваемой альтернативе. Идеальной называется такая альтернатива, которая имеет наилучшие значения всех критериев. Естественно, в реальности такой альтернативы не существует. Но наиболее приемлемой считается альтернатива, у которой расстояние от "идеальной точки" минимально:

Изображение выглядит как текст

Автоматически созданное описание

где **N** – количество критериев оценки альтернатив;  
**Xid j** – идеальное значение по j-му критерию для идеального варианта;

**Xij** – значение по j-му критерию для i-ой альтернативы.

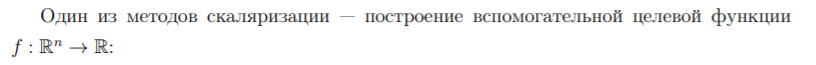
## АЛГОРИТМ

**По Парето**

Оптимизационные алгоритмы итеративны. Стартуя с некоторого начального приближения, алгоритм генерирует последовательность точек до тех пор, пока не будет выполнено условие остановки. Таким образом, в процессе оптимизации целевые функции и ограничения вычисляются на конечном множестве точек. При этом множество Парето-оптимальных решений в общем случае содержит бесконечное количество точек.

Соответственно, на практике задача многокритериальной оптимизации переформулируется так, чтобы ее можно было решить с помощью итеративного алгоритма. Существуют по крайней мере три таких формулировки: глобальная, локальная, интерактивная.

Возможный вариант решения - сведение к однокритериальной задаче методом скаляризации.





путем введения функционала h : принимающего значения компонент векторной целевой функции, и сводящего, таким образом, задачу к однокритериальной. Функция h, как правило, зависит от некоторых параметров: варьируя их и решая каждый раз задачу, получаем различные точки Парето-фронта.

Популярным подходом является метод взвешенной суммы, в котором функционал

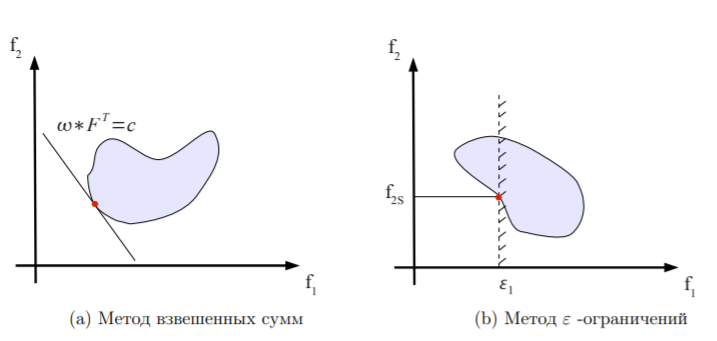
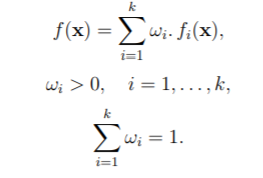
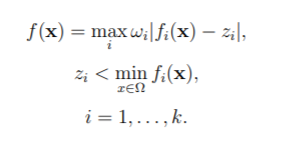


Рис. - методы скаляризации

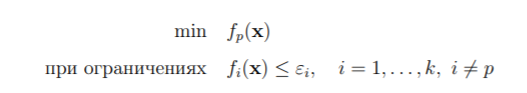
берется линейным:



Геометрическая интерпретация в случае двух целевых функций представлена выше. Минимизация интерпретируется как поиск такого значения c, при котором линия ω · f T = c касается границы множества допустимых значений целевых функций (f(Ω)). Этот метод имеет, однако, существенный недостаток, т.к. с помощью него не могут быть получены невыпуклые части Парето-фронта. Этого недостатка лишена функция скаляризации Чебышева:



Однако f(x) в данном случае является недифференцируемой, что затрудняет решение получившейся однокритериальной задачи с помощью градиентных методов. 7 Другим способом является метод ε-ограничений:

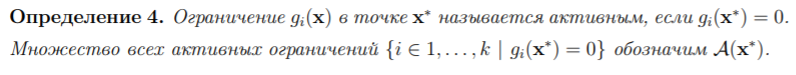


Изменяя p и εi можно получать различные точки Парето-фронта. Одним из недостатков рассмотренных методов является то, что для получения нескольких точек на Парето-фронте нужно несколько раз отдельно решить однокритериальную задачу, причем на каждой итерации этого решения вычисляются значения всех целевых функций.

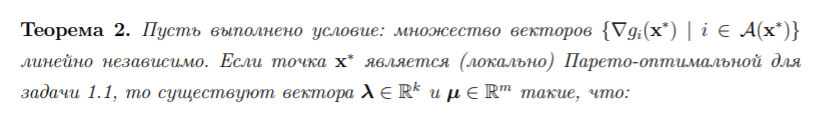
Значит, чтобы восстановить Парето-фронт с приемлемой точностью, потребуется большое количество итераций. Отдельной проблемой является подбор параметров (ωi , εi), т.к. его трудно выполнить автоматически без каких-либо предположений относительно вида Парето-фронта.

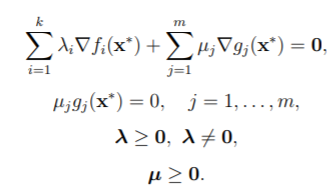
Возможный вариант решения задачи с ограничениями:

Определим понятие ограничения для задачи

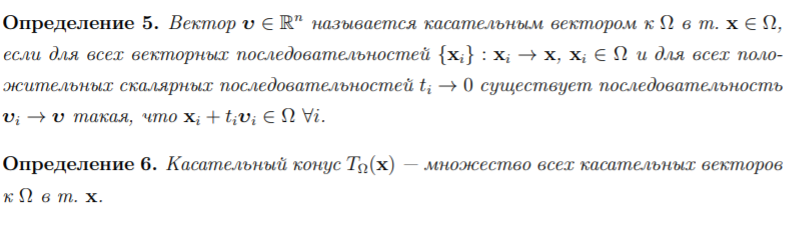


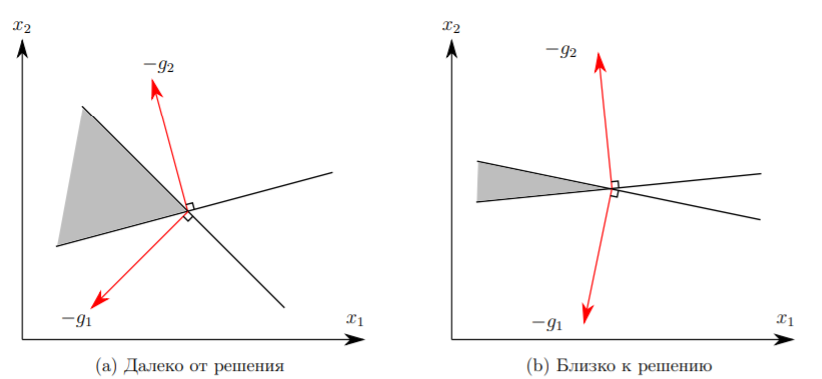
Сформулируем следующую теорему (необходимое условие Парето-оптимальности Каруша–Куна-Такера)



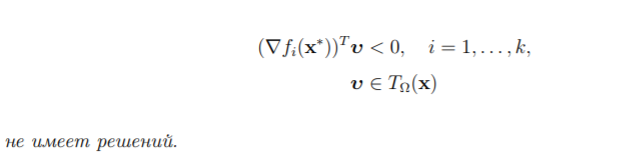


В ряде случаев удобно использовать формулировку необходимого условия для задачи 1.3, которая зависит только от геометрии множества допустимых значений Ω, а не от конкретного ее алгебраического описания с помощью функции g(x). Для этого нужно ввести понятия касательного вектора и касательного конуса.

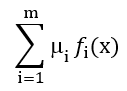




Если точка является (локально) Парето-оптимальной для задачи, то система:



**АЛГОРИТМ МЕТОДА ЛИНЕЙНОЙ СВЕРТКИ**

Решение многокритериальных задач на основе линейной свертки критериев состоит в назначении тем или иным способом неотрицательных (а чаще положительных) коэффициентов , в сумме дающих единицу (хотя это не обязательно), и последующей максимизации линейной комбинации критериев на множестве X

Как уже известно из математической модели, необходимо присвоить каждому критерию критерий значимости: в нашем случае

* Критерий стоимости – 9/10,
* Критерий площади – 6/10,
* Критерий расстояния до метро – 8/10,
* Критерий года постройки – 3/10,
* Критерий оценки постояльцев – 4/10,
* Критерий этажности – 5/10,
* Критерий высоты потолков – 3/10,
* Величина залога – 6/10.

Для того, чтобы перевести данные показатели в процентное выражение, необходимо сложить приведенные выше критерии и поделить каждый показатель на полученную сумму.

Для вычисления коэффициента каждого из вариантов в каждом критерии необходимо найти максимальное значение по критерию и поделить каждый из критериев на найденный максимум по критерию.

Для вычисления оптимального решения необходимо произвести суммирование произведений коэффициента из вариантов и процентного выражения критерия значимости. В данном случае для нормализации критериев, необходимо учесть, что в каждом из критериев нам необходимо найти либо минимальное, либо максимальное значение для поиска оптимального из них:

* Критерий стоимости – min,
* Критерий площади – max,
* Критерий расстояния до метро – min,
* Критерий года постройки – max,
* Критерий оценки постояльцев – max,
* Критерий этажности – max,
* Критерий высоты потолков – max,
* Величина залога – min.

Значения минимальные следует умножить на -1. В итоге формула будет иметь следующий вид:

где - коэффициент, отражающий значимость критерия "стоимость",

f1 - показатель критерия «стоимость» каждого из вариантов и т.д. по аналогии.

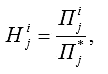
Максимальное значение из вычисленных сумм является оптимальным решением.

**Метод «идеальной точки»**

Алгоритм метода «идеальной точки» состоит в следующем:

Определяется «идеальная точка», т.е. лучшие параметры, с которыми сравниваются параметры рассматриваемых проектов. Такими идеальными параметрами могут быть лучшие значения параметров у всех рассматриваемых проектов или значения, определяемые экспертами.

Поскольку параметры проектов могут быть разными по масштабу, определяются их нормированные значения:

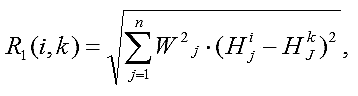


где **Нij** – нормированные значение j-го параметра i-го проекта, **Пij** – значение j-го параметра i-го проекта; **Пj\*** − значение j-го параметра «идеальной точки»; **i** - номер проекта, **j** - номер параметра.

Поскольку важность параметров проектов может быть разной, то экспертами определяются коэффициенты важности параметров проектов (Wj). Для определения коэффициентов важности можно использовать, например, 5-ти или 10-ти бальную шкалу. Коэффициенты Wj определяются делением на максимальный балл, как в табл. 1.

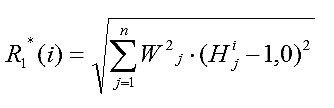
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Важность параметра | Очень важный | Важный | Менее важный | Мало важный | Не важный |
| Балл | 10-9 | 8-7 | 6-5 | 4-2 | 1–0 |
| Коэффициент Wj | 1,0-0,9 | 0,8-0,7 | 0.6-0,5 | 0,4-0,2 | 0,1-0 |

Определяются «расстояния» между проектами. Расстояния между проектами можно определить по двум формулам:



где **R1 (i, k)** – расстояние между i-м и k-м проектами, **Нij - Нкj** – нормированные значение j-го параметра для i- го и k-го проекта, **R1\*(i)** – расстояние от i-го проекта до идеальной точки, **Wj** – коэффициент важности j-го параметра, n – количество параметров проектов.

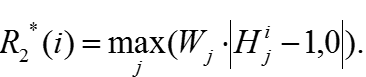
Поскольку нормированные значения параметров «идеальной точки» равны 1,0, то расстояние от проектов до «идеальной точки» определяется по формуле:



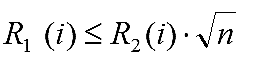
Второй формулой определения «расстояния» между проектами является:

page3image3362563376

В этом случае расстояние от проекта до «идеальной точки» определяется по формуле:



Имеет место соотношение:



Затем проекты ранжируются по возрастанию расстояния до «идеальной точки» R\*(i). Лучшим считается проект, у которого расстояние до «идеальной точки» является минимальным.

**ВАРИАНТЫ ИСПОЛЬЗОВАНИЯ**

Перед запуском кода необходимо поместить csv файл flat.csv и файл с кодом в рабочую директорию. Для того, чтобы узнать её, введите следующий код и запустите его:

Graphical user interface, text, application, email

Description automatically generated

Рис. 1 – «Поиск рабочей директории»

Запустим код в строке In [2]. После запуска на экране появляется таблица из csv файла:

Graphical user interface, application, table

Description automatically generated

Рис. 2 – «Таблица из csv файла»

**Линейная свёртка**

Запустим код в строке In [3] «Линейная свертка». После запуска на экране появляется таблица с максимальным значением по каждой из характеристик объекта недвижимости:

Graphical user interface, text, application

Description automatically generated

Рис. 3 – «Максимум по характеристикам квартиры»

Запустим код в строке In [4]. После запуска на экране появляется таблица из csv файла с коэффициентами значимости по по каждой из характеристик объекта недвижимости из csv файла:

Graphical user interface, text, application, email

Description automatically generated

Рис. 4 – «Коэффициент значимости»

Запустим код в строке In [5]. После запуска на экране появляется таблица c обработанными значениями по линейной свертке:

Graphical user interface, application, table

Description automatically generated

Рис. 5 – «Обработанные значения»

Запустим код в строке In [6]. После запуска на экране появляется ответ на поставленную задачу:

Graphical user interface, text, application, email

Description automatically generated

Рис. 6 – «Ответ»

**Метод идеальной точки**

Запустим код в строке In [7]. После запуска на экране появляется таблица cо значениями идеальной очки из csv файла:

Graphical user interface, text, application, email

Description automatically generated

Рис. 7 – «Идеальная точка»

Запустим код в строке In [8]. После запуска на экране появляется таблица c обработанными значениями по идеальной точке:

Graphical user interface, application, table, Excel

Description automatically generated

Рис. 8 – «Обработанные значения»

Запустим код в строке In [9]. После запуска на экране появляется ответ на поставленную задачу и таблица с расстояниями до идеальной точки:

Graphical user interface, text, application

Description automatically generated

Рис. 9 – «Ответ»

Запустим код в следующей строке. После запуска на экране появляется лепесковая диаграмма по медоду идеальной точки:

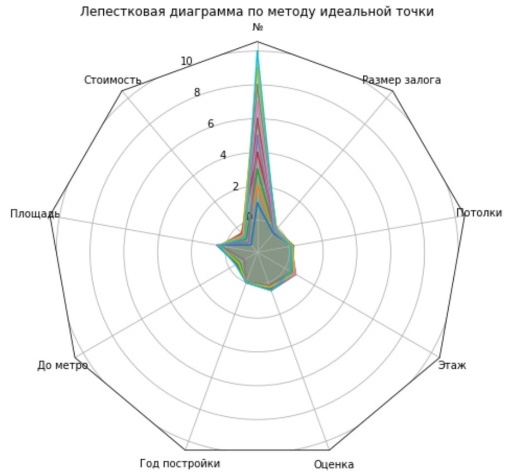


Рис. 10 – «Лепесковая диаграмма по медоду идеальной точки»

**Метод контрольных показателей**

Запустим код в строке In [10]. После запуска на экране появляется таблица cо значениями контрольных показателей из csv файла:

Graphical user interface, text, application, email

Description automatically generated

Рис. 11 – «Контрольные показатели»

Запустим код в строке In [11]. После запуска на экране появляется таблица c обработанными значениями по контрольным показателям:

Graphical user interface, application, table

Description automatically generated

Рис. 12 – «Обработанные значения»

Запустим код в следующей строке. После запуска на экране появляется лепесковая диаграмма по медоду контрольных показателей:

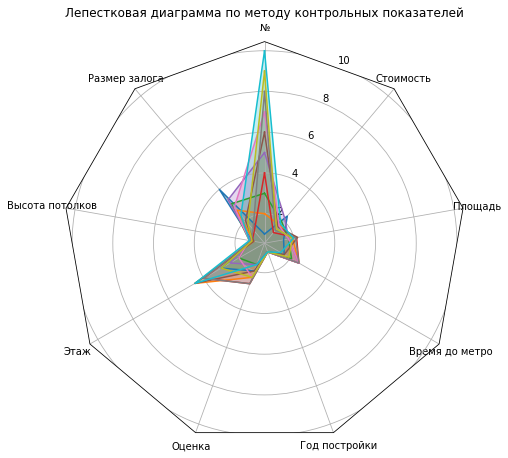


Рис. 14 – «Лепесковая диаграмма по медоду контрольных показателей»

Запустим код в строке In [12]. После запуска на экране появляется ответ на поставленную задачу и таблица с расстояниями до контрольных показателей:

Graphical user interface, text, application, email

Description automatically generated

Рис. 13 – «Ответ»

**АРХИТЕКТУРА РЕШЕНИЯ**

В ходе выполнения задания на языке программирования Python было реализовано три кода:

* Линейная свертка;
* Метод идеальной точки;
* Метод контрольных показателей.

**1. Импорт библиотек для выполнения обработки данных и операций с массивами:**

import pandas as pd

import numpy as np

**2. Считывание данных из csv-файла:**

i = 0

import csv

with open("flat.csv", 'r') as file:

reader = csv.reader(file)

for row in reader:

i = i+1

if i == 1:

number = list(map(int,row[0].split(';')))

if i == 2:

cost = list(map(int,row[0].split(';')))

if i == 3:

square = list(map(int,row[0].split(';')))

if i == 4:

time = list(map(int,row[0].split(';')))

if i == 5:

year = list(map(int,row[0].split(';')))

if i == 6:

grade = list(map(int,row[0].split(';')))

if i == 7:

floor = list(map(int,row[0].split(';')))

if i == 8:

height = list(map(float,row[0].split(';')))

if i == 9:

pledge = list(map(int,row[0].split(';')))

if i == 10:

kkk = list(map(int,row[0].split(';')))

if i == 11:

ideal = list(map(float,row[0].split(';')))

if i == 12:

contr = list(map(float,row[0].split(';')))

**3. Формирование массивов данных и поиск максимальных значений по критериям:**

a = np.array(number)

b = np.array(cost)

max\_number1 = max(b)

c = np.array(square)

max\_number2 = max(c)

d = np.array(time)

max\_number3 = max(d)

e = np.array(year)

max\_number4 = max(e)

f = np.array(grade)

max\_number5 = max(f)

g = np.array(floor)

max\_number6 = max(g)

h = np.array(height)

max\_number7 = max(h)

i = np.array(pledge)

max\_number8 = max(i)

ii = np.array(ideal)

cc = np.array(contr)

q = np.array([b, c, d, e, f, g, h, i])

q = q.transpose()

**4. Формирование таблицы с исходными данными:**

df = pd.DataFrame(data=q, index = hhh, columns = hhh1)

df

**5. Формирование таблицы с максимальными значениями по критериям:**

def pretty\_table(data, cell\_sep=' | ', header\_separator=True) -> str:

rows = len(data)

cols = len(data[0])

col\_width = []

for col in range(cols):

columns = [str(data[row][col]) for row in range(rows)]

col\_width.append(len(max(columns, key=len)))

separator = "-+-".join('-' \* n for n in col\_width)

lines = []

for i, row in enumerate(range(rows)):

result = []

for col in range(cols):

item = str(data[row][col]).rjust(col\_width[col])

result.append(item)

lines.append(cell\_sep.join(result))

if i == 0 and header\_separator:

lines.append(separator)

return '\n'.join(lines)

data = [

["Максимум", "Значение"],

{"по стоимости": max\_number1, "по площади": max\_number2, "по времени до метро": max\_number3, "по году постройки": max\_number4, "по оценке постояльцев": max\_number5, "по этажу": max\_number6, "по высоте потолков": max\_number7, "по залогу":max\_number8}

]

rows = [

data[0]

]

rows += [(k, v) for k, v in data[1].items()]

print(pretty\_table(rows))

**6. Визуализация коэффициентов значимости:**

kk = np.array(kkk)

kk1 = np.sum([kk])

koef = [-kk[0]/kk1, kk[1]/kk1, -kk[2]/kk1, kk[3]/kk1, kk[4]/kk1, kk[5]/kk1, kk[6]/kk1, -kk[7]/kk1]

kkkk = [kk, koef]

ind = ['Значимость', 'a (%) = ']

cpl = ['', '', '', '', '', '', '', '']

df1 = pd.DataFrame(data=kkkk, index = ind, columns = cpl)

df1

**7. Формирование таблицы с обработанными значениями по линейной свертке:**

b1 = b/max\_number1

c1 = c/max\_number2

d1 = d/max\_number3

e1 = e/max\_number4

f1 = f/max\_number5

g1 = g/max\_number6

h1 = h/max\_number7

i1 = i/max\_number8

p = np.array([b1, c1, d1, e1, f1, g1, h1, i1])

p = p.transpose()

hhh = ['1', '2', '3', '4', '5', '6', '7', '8', '9', '10']

hhh1 = ['Стоимость→min', 'Площадь→max', 'Время до метро→min', 'Год постройки→max', 'Оценка→max', 'Этаж→max', 'Потолок→max', 'Залог→min']

df2 = pd.DataFrame(data=p, index = hhh, columns = hhh1)

df2

**8. Расчет и вывод линейной свертки и оптимального значения:**

koef = [-kk[0]/kk1, kk[1]/kk1, -kk[2]/kk1, kk[3]/kk1, kk[4]/kk1, kk[5]/kk1, kk[6]/kk1, -kk[7]/kk1]

ww = p\*koef

www = np.array([np.sum([ww[0]]), np.sum([ww[1]]), np.sum([ww[2]]), np.sum([ww[3]]), np.sum([ww[4]]), np.sum([ww[5]]), np.sum([ww[6]]), np.sum([ww[7]]), np.sum([ww[8]]), np.sum([ww[9]])])

www = www.transpose()

print ('Оптимальное значение', max(www), '- квартира № 2')

hhh = ['1', '2', '3', '4', '5', '6', '7', '8', '9', '10']

hhh2 = ['Показатель по линейной свертке']

df3 = pd.DataFrame(data=www, index = hhh, columns = hhh2)

df3

**9. Визуализация идеальных значений**

ii = np.array(ideal)

col = ['Идеальное значение']

ind1 = ['', '', '', '', '', '', '', '']

df4 = pd.DataFrame(data=ii, index = ind1, columns = col)

df4

**10. Формирование таблицы с обработанными значениями по идеальной точке:**

b2 = 1/((ii[0]-b)/ii[0])

c2 = (ii[1]-c)/ii[1]

d2 = 1/((ii[2]-d)/ii[2])

e2 = (ii[3]-e)/ii[3]

f2 = (ii[4]-f)/ii[4]

g2 = (ii[5]-g)/ii[5]

h2 = (ii[6]-h)/ii[6]

i2 = 1/((ii[7]-i)/ii[7])

u = np.array([b2, c2, d2, e2, f2, g2, h2, i2])

u = u.transpose()

df5 = pd.DataFrame(data=u, index = hhh, columns = hhh1)

df5

**11. Расчет и вывод оптимального значения по идеальной точке:**

wwww = [np.sum([(u[0])\*\*2]), np.sum([(u[1])\*\*2]), np.sum([(u[2])\*\*2]), np.sum([(u[3])\*\*2]), np.sum([(u[4])\*\*2]), np.sum([(u[5])\*\*2]), np.sum([(u[6])\*\*2]),np.sum([(u[7])\*\*2]), np.sum([(u[8])\*\*2]), np.sum([(u[9])\*\*2])]

print ('Оптимальное значение', min(wwww), '- квартира № 6')

hhh = ['1', '2', '3', '4', '5', '6', '7', '8', '9', '10']

hhh3 = ['Расстояние']

df6 = pd.DataFrame(data=wwww, index = hhh, columns = hhh3)

df6

**12. Импорт библиотек для построения лепестковой диаграммы:**

import matplotlib.pyplot as plt

from matplotlib.patches import Circle, RegularPolygon

from matplotlib.path import Path

from matplotlib.projections.polar import PolarAxes

from matplotlib.projections import register\_projection

from matplotlib.spines import Spine

from matplotlib.transforms import Affine2D

**13. Задание функций с параметрами графика лепестковой диаграммы:**

def radar\_factory(num\_vars, frame='circle'):

theta = np.linspace(0, 2\*np.pi, num\_vars, endpoint=False)

class RadarAxes(PolarAxes):

name = 'radar'

def \_\_init\_\_(self, \*args, \*\*kwargs):

super().\_\_init\_\_(\*args, \*\*kwargs)

self.set\_theta\_zero\_location('N')

def fill(self, \*args, closed=True, \*\*kwargs):

"""Override fill so that line is closed by default"""

return super().fill(closed=closed, \*args, \*\*kwargs)

def plot(self, \*args, \*\*kwargs):

"""Override plot so that line is closed by default"""

lines = super().plot(\*args, \*\*kwargs)

for line in lines:

self.\_close\_line(line)

def \_close\_line(self, line):

x, y = line.get\_data()

if x[0] != x[-1]:

x = np.concatenate((x, [x[0]]))

y = np.concatenate((y, [y[0]]))

line.set\_data(x, y)

def set\_varlabels(self, labels):

self.set\_thetagrids(np.degrees(theta), labels)

def \_gen\_axes\_patch(self):

if frame == 'circle':

return Circle((0.5, 0.5), 0.5)

elif frame == 'polygon':

return RegularPolygon((0.5, 0.5), num\_vars, radius=0.5, edgecolor="k")

else:

raise ValueError("unknown value for 'frame': %s" % frame)

def draw(self, renderer):

if frame == 'polygon':

gridlines = self.yaxis.get\_gridlines()

for gl in gridlines:

gl.get\_path().\_interpolation\_steps = num\_vars

super().draw(renderer)

def \_gen\_axes\_spines(self):

if frame == 'circle':

return super().\_gen\_axes\_spines()

elif frame == 'polygon':

spine = Spine(axes=self,

spine\_type='circle',

path=Path.unit\_regular\_polygon(num\_vars))

spine.set\_transform(Affine2D().scale(.5).translate(.5, .5)

+ self.transAxes)

return {'polar': spine}

else:

raise ValueError("unknown value for 'frame': %s" % frame)

register\_projection(RadarAxes)

return theta

**14. Вывод соответствующего графика по методу идеальной точки:**

data = [['№', 'Стоимость', 'Площадь', 'До метро', 'Год постройки', 'Оценка', 'Этаж', 'Высота потолков', 'Размер залога'],

('Лепестковая диаграмма по методу идеальной точки',

[vect[0], vect[1], vect[2], vect[3], vect[4], vect[5], vect[6], vect[7], vect[8], vect[9]],

)]

N = len(data[0])

theta = radar\_factory(N, frame='polygon')

spoke\_labels = data.pop(0)

title, case\_data = data[0]

fig, ax = plt.subplots(figsize=(15, 7), subplot\_kw=dict(projection='radar'))

fig.subplots\_adjust(top=0.85, bottom=0.05)

ax.set\_rgrids([-2, 0, 2, 4, 6, 8, 10])

ax.set\_title(title, position=(0.5, 1.1), ha='center')

for ddd in case\_data:

line = ax.plot(theta, ddd)

ax.fill(theta, ddd, alpha=0.25)

ax.set\_varlabels(spoke\_labels)

ax.set\_theta\_direction(-1)

plt.show()

**15. Формирование таблицы с обработанными значениями по контрольным показателям:**

cc = np.array(contr)

col = ['Контрольные показатели']

ind1 = ['', '', '', '', '', '', '', '']

df7 = pd.DataFrame(data=ii, index = ind1, columns = col)

df7

**16. Формирование таблицы с обработанными значениями по контрольным показателям:**

b3 = 1/(b/cc[0])

c3 = c/cc[1]

d3 = 1/(d/cc[2])

e3 = e/cc[3]

f3 = f/cc[4]

g3 = g/cc[5]

h3 = h/cc[6]

i3 = 1/(i/cc[7])

u1 = np.array([b3, c3, d3, e3, f3, g3, h3, i3])

u1 = u1.transpose()

df8 = pd.DataFrame(data=u1, index = hhh, columns = hhh1)

df8

**17. Расчет и вывод оптимального значения по контрольным показателям:**

wwwww = [min(u1[0]), min(u1[1]), min(u1[2]), min(u1[3]), min(u1[4]), min(u1[5]), min(u1[6]), min(u1[7]), min(u1[8]), min(u1[9])]

for i in wwwww:

if i <1:

wwwww.remove (i)

print ('Оптимальное значение', max(wwwww), '- квартира № 9')

df9 = pd.DataFrame(data=wwwww, index = hhh, columns = hhh3)

df9

**18. Вывод соответствующего графика по методу контрольных показателей:**

data1 = [['№', 'Стоимость', 'Площадь', 'Время до метро', 'Год постройки', 'Оценка', 'Этаж', 'Высота потолков', 'Размер залога'],

('Лепестковая диаграмма по методу контрольных показателей',

[vect1[0], vect1[1], vect1[2], vect1[3], vect1[4], vect1[5], vect1[6], vect1[7], vect1[8], vect1[9]],

)]

N = len(data1[0])

theta = radar\_factory(N, frame='polygon')

spoke\_labels = data1.pop(0)

title, case\_data = data1[0]

fig, ax = plt.subplots(figsize=(15, 7), subplot\_kw=dict(projection='radar'))

fig.subplots\_adjust(top=0.85, bottom=0.05)

ax.set\_rgrids([-2, 0, 2, 4, 6, 8, 10])

ax.set\_title(title, position=(0.5, 1.1), ha='center')

for ddd in case\_data:

line = ax.plot(theta, ddd)

ax.fill(theta, ddd, alpha=0.25)

ax.set\_varlabels(spoke\_labels)

ax.set\_theta\_direction(-1)

plt.show()

**5. EXCEL**

С помощью EXCEL было решено 5 задач методами Парето, Линейной свёртки, Идеальной точки и Контрольных показателей. Все задачи решаются аналогично, поэтому рассмотрим ход решения одной из них (условие соответствует изложенному в физической модели).

**Парето**

1. Заполним исходные данные, используя информацию, изложенную на портале Циан. Будут рассмотрены 10 объектов недвижимости и 8 критериев сравнения:

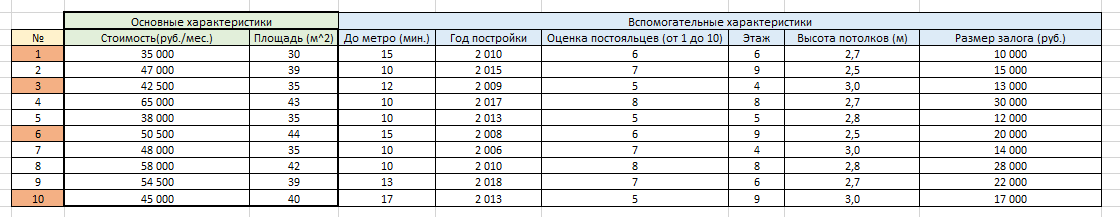


Рис. 1 – «Исходные данные»

1. Выделим 2 основные характеристики: арендная плата (цена(руб./мес.)) и площадь(м^2). Остальные - вспомогательные характеристики;
2. Построим точечную диаграмму по основным характеристикам:

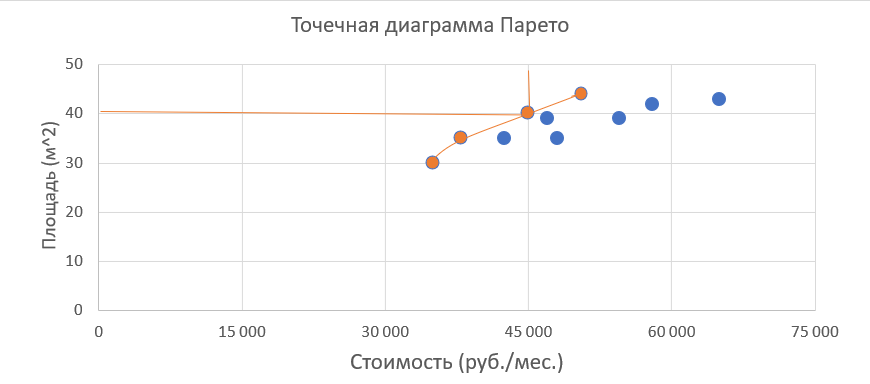


Рис. 2 – «Точечная диаграмма Парето»

1. Наилучшим вариантом будет считаться квартира меньшая по цене, но большая по площади. Построим линии из каждой точки (в данном случае изобразим линии только из 1 точки) в направлении минимальной цены и максимальной площади. Рассмотрим четверть пространства, которые они отделяют. Если в четверти пространства, отделяемой линиями есть другие точки, то точка не является оптимальной, иначе выделяем данные точки оранжевым цветом. В данном случае линии на диаграмме проведены из неулучшаемого значения (оптимальной точки) с координатами [45 000; 40]. Другими неулучшаемыми значениями в данной задаче являются также точки с координатами: [35 000; 30], [50 500; 44], [42 500; 35]. Выделим все эти точки цветом;
2. Сделаем вывод: можно выбрать 1 из 4 вариантов квартир (№1, 3, 6, 10).

**Линейная свёртка**

1. Присвоим каждому из критериев вес и выберем, какой из критериев наиболее, а какой наименее важен при выборе квартиры. Укажем значимость каждого из критериев от 1 до 10. Затем приведём значимость к процентному формату. При этом необходимо домножить значение всех критериев, которые стремятся к min на -1, так как критерии разноимённые (max и min) и складывать их мы не можем, но если мы домножим min на -1, то получим -1\*(min) = max:



Рис. 3 – «Процент значимости критерия»

1. Произведём нормировку: в отдельной таблице разделить каждый столбец на максимальное по данному столбцу значение (нормировка по максимуму):



Рис. 4 – «Нахождение максимума»



Рис. 5 – «Нормировка по максимуму»

1. Используем формулу СУММПРОИЗВ (каждая строка таблицы; строка % значимости). Среди полученных значений выберем максимальное:

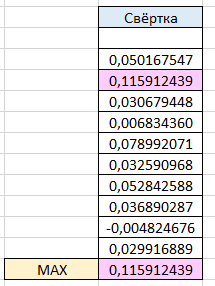


Рис. 6 – «Свёртка»

1. Сделаем вывод: при указанной значимости каждого из критериев наилучшей является квартира №2.

**Метод идеальной точки (ввод идеальных критериев производит пользователь)**

1. Зададим идеальные критерии квартиры:

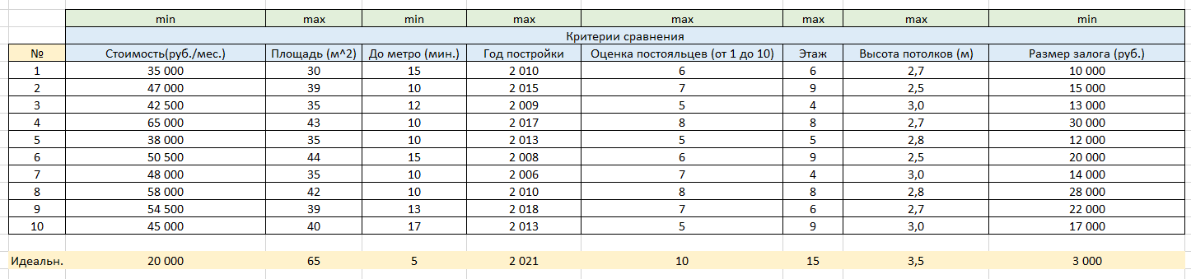


Рис. 7 – «Идеальные критерии»

1. Найдём нормированные значения: из идеального значения вычитаем текущее и делим полученное значение на идеальное. Для значений, которые стремятся к min перевернём дробь:



Рис. 8 – «Нормированные значения»

1. Найдём расстояние от идеальной точки, до каждой из других точек, путём нахождения суммы квадратов значений каждого из критериев для соответствующей точки-квартиры. Найдём минимальное из этих расстояний:

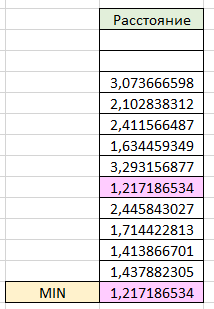


Рис. 9 – «Расстояния от идеальной точки»

1. Сделаем вывод: для данного условия точка-квартира №6 имеет наименьшее расстояние до идеальной точки-квартиры.
2. Построим лепестковую диаграмму:

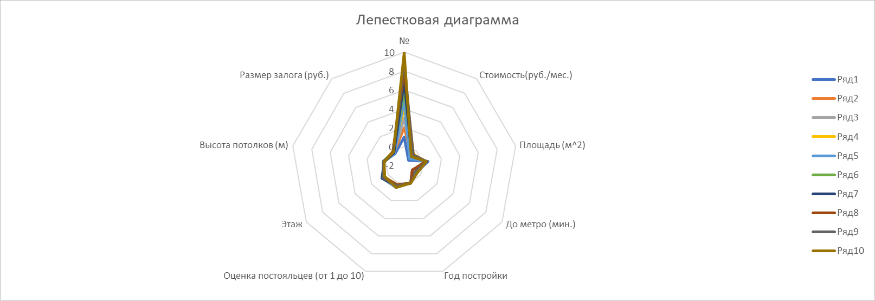


Рис. 10 – «Лепестковая диаграмма»

**Метод идеальной точки (идеальная точка задаются по умолчанию)**

Ход решения соответствует тому, что был рассмотрен выше - при вводе идеальных критериев пользователем, однако пункт 1 и 4 отличается:

1. По каждому столбцу найдём максимальное значение для тех критериев, которые стремятся к max и минимальное значение для тех критериев, которые стремятся к min. Эти значения и будут соответствовать идеальным;
2. Сделаем вывод: при задании идеальной точки по умолчанию, подходит вариант квартиры №6.

**Метод контрольных показателей (ввод контрольных показателей производит пользователь)**

1. Для каждого показателя зададим контрольное значение, ниже/выше которого мы не будем рассматривать квартиру:

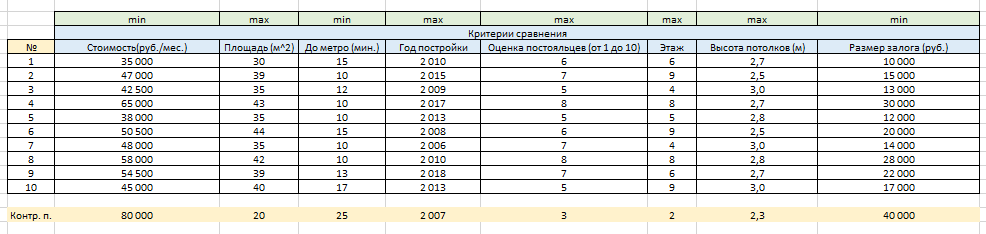


Рис. 11 – «Контрольные показатели»

1. Найдём нормированные значения: во сколько раз текущее значение лучше минимально опасного (контрольного показателя) = текущее/контр. п. Для значений, которые стремятся к min перевернём дробь:



Рис. 12 – «Нормированные значения»

1. Проверим, не заходит ли какое-то значение за контрольный показатель (число < 1). Для квартиры 7 по показателю год постройки значение критерия сравнения оказалось меньше 1. Следовательно, далее мы её не рассматриваем;
2. Для каждой квартиры найдём, по какому критерию она ближе всего к контрольному показателю. Для всех квартир этим показателем является год постройки;
3. Найдём минимум по каждому ряду, затем максимум по полученным значениям. Квартира, которой он соответствует находится дальше всех от контрольного показателя. Следовательно, она является наилучшим вариантом:

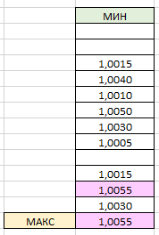


Рис. 13 – «Лучший вариант»

1. Сделаем вывод: наиболее сбалансированно удалённый вариант по всем показателям от того контрольного значения, которое нельзя переступать – квартира № 9;
2. Построим лепестковую диаграмму:

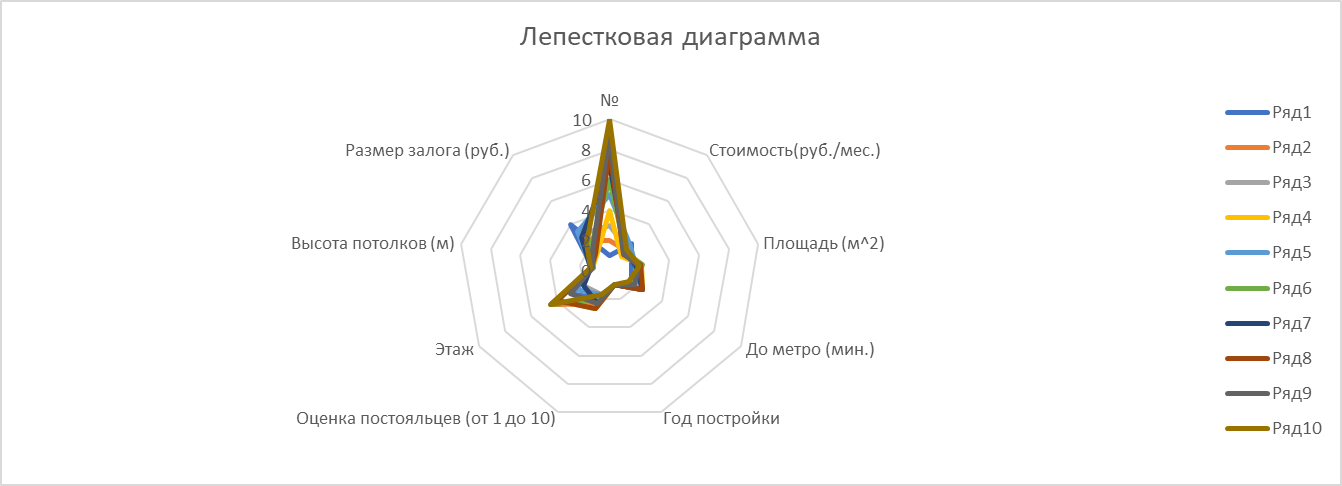


Рис. 14 – «Лепестковая диаграмма»

**Метод контрольных показателей (контрольные показатели задаются по умолчанию)**

Ход решения соответствует тому, что был рассмотрен выше - при вводе контрольных показателей пользователем, однако пункт 1 и 6 отличается:

1. По каждому столбцу найдём максимальное значение для тех критериев, которые стремятся к min и минимальное значение для тех критериев, которые стремятся к max. Эти значения и будут соответствовать контрольным показателям;
2. Сделаем вывод: при задании контрольных показателей по умолчанию, подходит вариант квартиры №9.

**6. ТЕСТИРОВАНИЕ**

Сравним эффективность использования каждого из представленных методов поиска решения:

Таблица №4 – «Сравнение методов»

|  |  |  |
| --- | --- | --- |
| Критерий | Excel | Метод наименьших квадратов Python |
| Количество параметров | Не ограничено | Не ограничено |
| Простота восприятия | Низкая | Высокая |
| Время | Затратно по времени | Не затратно по времени |
| Тестирование | Число приближенное к тестированию | Число приближенное к тестированию |
| Точность | До 10000 | До 10000 |

Результаты решения задачи, представленной в разделе Физическая модель (по нахождению наилучшего варианта квартиры), а также иных задач по данной теме, совпали при их решении с помощью Excel и Python.

Однако наиболее оптимальным методом решения задачи можно назвать метод линейной свёртки, реализованный с помощью Python, так как он более прост в восприятии пользователем и является менее время затратным.

При сравнении конкретных методов нахождения оптимального решения, наиболее оптимальным также можно назвать метод Линейной свёртки, как наиболее простой в реализации и дающий довольно точный и конкретный ответ.

**7. ЗАКЛЮЧЕНИЕ**

С помощью представленного алгоритма удалось решить поставленную перед нами задачу, используя несколько методов.

Можно предложить следующий вариант развития данного кода:

1. Введение данных с клавиатуры в пользовательском интерфейсе;
2. Задание количества критериев сравнения и объектов с клавиатуры.

В процессе решения мы поняли, что этот же алгоритм можно применить и для решения таких задач как:

1. Выбор отеля для отдыха;
2. Выбор блюда на праздник исходя из калорийности;
3. Выбор телефона;
4. Поиск лучшего спортсмена.