

Санкт-Петербургский политехнический университет Петра Великого  
Институт компьютерных наук и технологий  
**Высшая школа интеллектуальных систем и суперкомпьютерных  
технологий**

## **КУРСОВАЯ РАБОТА**

### **Бинаризация данных**

по дисциплине «Вычислительная математика»

Выполнил  
студент гр. 3530903/00003

А. С. Усыченко

Руководитель  
доцент, д.т.н.

Е. Г. Хитров

27.05.2022

Санкт-Петербург

2022

## **ЗАДАНИЕ**

### **Содержание**

<b><i>ВВЕДЕНИЕ.....</i></b>	<b><i>3</i></b>
<b><i>1. Разработка алгоритма .....</i></b>	<b><i>4</i></b>
<b><i>2. Разработка прикладной программы .....</i></b>	<b><i>10</i></b>
<b><i>3. Тестирование программы и алгоритма.....</i></b>	<b><i>11</i></b>
<b><i>ЗАКЛЮЧЕНИЕ.....</i></b>	<b><i>13</i></b>
<b><i>Библиографический список .....</i></b>	<b><i>14</i></b>
<b><i>Приложения.....</i></b>	<b><i>15</i></b>

## ВВЕДЕНИЕ

### Введение

#### Цель курсового проекта:

- Научиться приему бинаризации данных, разработать алгоритм, использующий этот прием, создать прикладную программу, которую можно будет использовать обычным пользователям.

#### Задачи курсового проекта:

- Разработать и реализовать алгоритм “обучения” по результатам большой выборки данных.
- Проверить работоспособность алгоритма на реальных данных
- Создать программу, с которой будет взаимодействовать пользователь, для доказательства работы алгоритма
- Подготовить отчет по выполненной работе

Данная тема является актуальной, так как довольно большая часть алгоритмов машинного обучения строится на бинаризации данных. А в наше время эта сфера очень популярна. Было решено создать программу, имеющую практическое применение с использованием знаний, полученных во время изучения Вычислительной математики и программирования.

В курсовом проекте будет рассматриваться задача получения ответа на вопрос сдаст ли студент зачет в зависимости от времени подготовки к нему. В программе PyCharm будет написан код, который анализирует все имеющиеся данные и выдает результат в виде функции и весов к ней. Далее будет разработана прикладная программа, которая будет применять алгоритм, полученный на предыдущем этапе, и выводить предположение о том, сдаст пользователь зачет или нет, в зависимости от времени подготовки к нему.

## 1. Разработка алгоритма

Для начала стоит получить хотя бы небольшую базу данных чтобы разрабатывать алгоритм по реальным данным. Все попытки найти нужные мне данные в интернете оказались неуспешными, поэтому я решил собрать данную базу данных самостоятельно. Создал Яндекс-форму (Рис 1) с двумя вопросами “Сколько ты готовился к зачету?”, “Получилось ли у тебя сдать зачет?”, потом отправил эту форму своим друзьям и знакомым с просьбой пройти её.

Рис 1

The image shows a Yandex Form interface. At the top, it says 'Привет!' (Hello!) followed by a subtitle: 'Ты очень поможешь мне, если честно ответишь на пару вопросов' (You will help me a lot if you honestly answer a couple of questions) and 'Всё абсолютно анонимно' (Everything is absolutely anonymous). Below this is a section titled 'Непосредственно вопрос' (Directly the question). The question text is: 'Вспомни зачет, который ты сдавал' (Remember the exam you took), 'Сколько времени (в часах) ты к нему готовился и какой был результат ?' (How much time (in hours) did you spend preparing for it and what was the result?). Below the question is a note: 'Не забывай и о неудачных попытках' (Don't forget about unsuccessful attempts). There is a text input field labeled 'Ответ:' (Answer:). Below the input field is a checkbox labeled 'Зачет сдал успешно' (Exam passed successfully). At the bottom of the form is a button labeled 'Отправить' (Send).

Всего я получил 16 ответов – это и будет наша изначальная база данных, по которой мы будем разрабатывать алгоритм. Для удобства работы с этими данными они были перенесены в таблицу Excel.

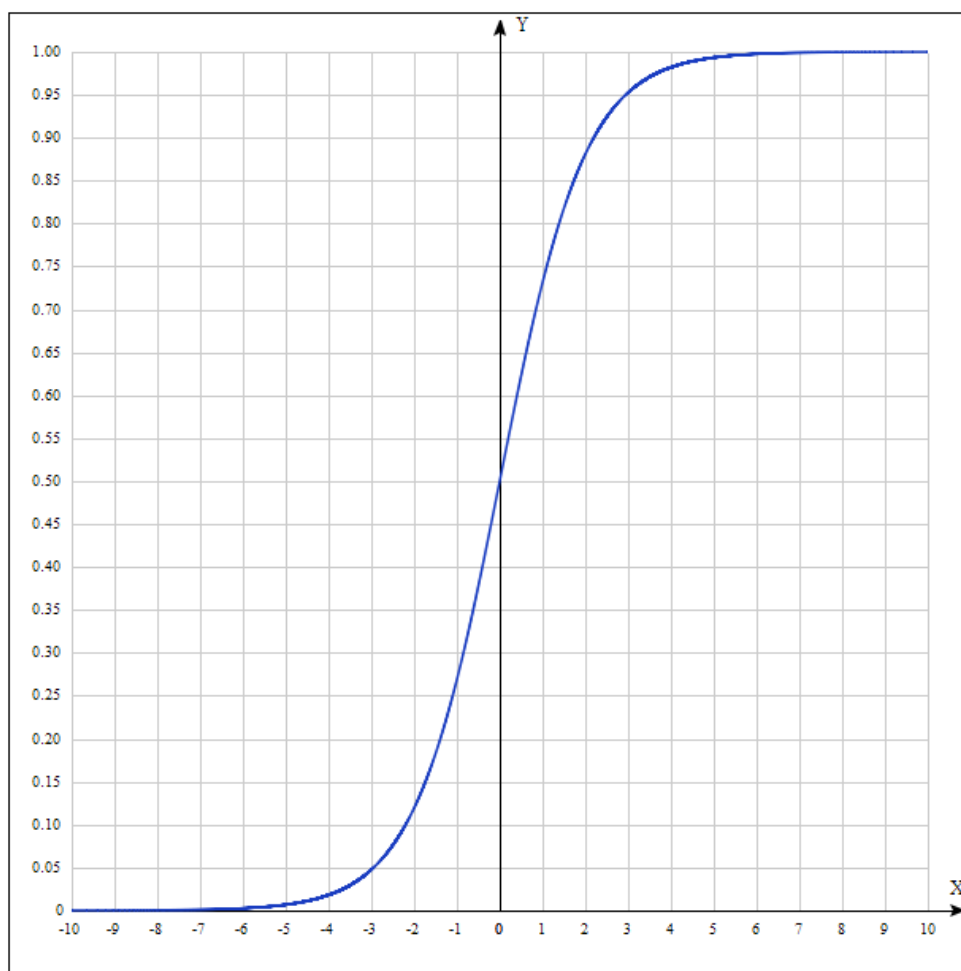
## АЛГОРИТМ

Начнем написание алгоритма с поиска подходящей функции, подставляя значения в которую мы будем получать значения от 0 до 1. Будем использовать логистическую функцию (см на следующей странице).

$$y(x) = \frac{1}{1 + e^{-a-bx}}$$

Эта функция всегда выдает значения от 0 до 1, однако она обладает аргументами  $a$  и  $b$ , которые следует подбирать самостоятельно. В своем стандартном виде со значениями  $a = 0$ ,  $b = 1$ , функция имеет вид: (Рис 2).

Рис 2



Нам не подходит вариант с подбором значений, так как аргументы  $a$  и  $b$  алгоритм должен определять самостоятельно, но есть другой путь, но нему мы и пойдем. Заменяем данную функцию на линейную и подберем аргументы  $a_0$  и  $a_1$  для неё методом наименьших квадратов.

Замена:

$$y^* = a_0 + a_1 \cdot x^*; x^* = e^{-x};$$

Формула для поиска аргументов методом наименьших квадратов:

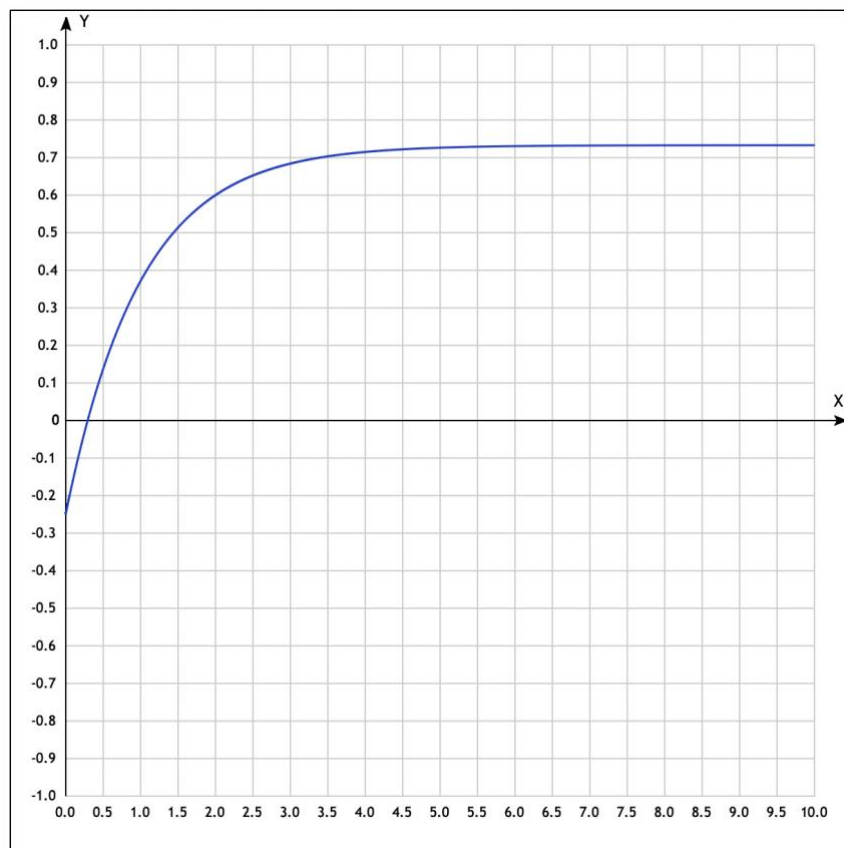
$$a = (P^T P)^{-1} P^T Y$$

Где:

$$\text{basis} = [1 \quad x^*]$$
$$P = \begin{bmatrix} 1 & e^{-x_0} \\ 1 & e^{-x_1} \\ \dots & \dots \\ 1 & e^{-x_n} \end{bmatrix}; \quad Y = \begin{bmatrix} 1/y_0 \\ 1/y_1 \\ \dots \\ 1/y_n \end{bmatrix};$$

Реализовав данный метод в программе (Приложение 1), мы получаем значения аргументов  $a_0 = 0.73289$ ;  $a_1 = -0.98153$ , а наша новая функция  $y^* = 0.73289 - 0.98153 \cdot e^{-x}$ . График этой функции (Рис 3) довольно сильно отличается от изначальной функции, однако ведет себя схожим образом. К тому же, для нас не столь важно как выглядит график, главное чтобы он был монотонным, ведь программа видит всё цифрами.

Рис 3



Теперь обратимся к нашей выборке, потенциально в ней могут быть исключительные случаи, которые могут сильно испортить картину, если мы решим обучить наш алгоритм с использованием этих данных. Есть 2 пути решения проблемы с выпадающими значениями.

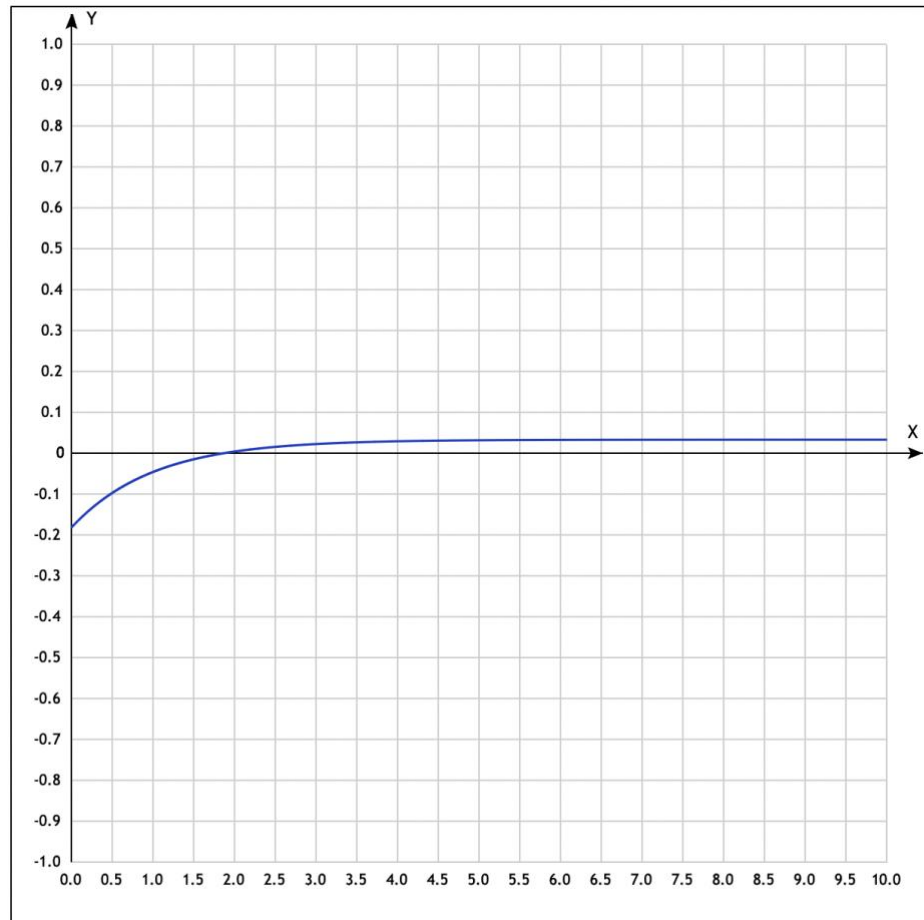
1. Стандартный метод – откинуть 10% минимальных и максимальных значений и производить обучение по полученным данным. Данный подход является простым, но в условиях нашей задачи, выкидывать запись “0 часов подготовки – незачет” – неправильно, так как это логичный исход и на подобных примерах обязательно должна учиться программа.
2. Задание весов для каждого значения. Этот подход сложнее предыдущего, однако он универсальный. К тому же его можно реализовать через все тот же метод наименьших квадратов с небольшими изменениями.

Для нашей задачи был выбран 2 вариант. Осталось разобраться с тем, как задавать веса. Было выбрано правило – чем дальше (по X) находится значение от его эталонного, тем меньше оно весит. Эталонное значение для НЕ сдавших – 0, для сдавших – максимальное значение часов подготовки среди сдавших. Итого мы имеем формулы:

$$a = (P^T W^2 P)^{-1} P^T W Y; \quad w = 1 - |len|; \quad W = \begin{bmatrix} w & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & w \end{bmatrix}$$

Прогоним данные, полученные после первого использования метода наименьших квадратов, и получим матрицу с весами для каждого элемента (Приложение 2). Далее применим формулу и посчитаем новые значения аргументов  $a_0$  и  $a_1$ . Они равны  $a_0 = 0.03254$ ;  $a_1 = -0.21475$ , а наша новая функция  $y^* = 0.03254 - 0.21475 \cdot e^{-x}$ . График (Рис 4) похож на график, полученный на предыдущем этапе, а значит все сделано верно.

Рис 4



Осталось только рассчитать контрольное значение. Это значение, которое определяет в какую сторону мы будем округлять вычисление – в сторону 0 или 1. Можно взять значение 0.031, оно достигается при  $x = 7$ , что является минимальным временем среди сдавших зачет, однако этот подход нам не подходит по нескольким причинам. Задача курсовой – реализовать алгоритм “обучения” по результатам большой выборки данных, а значит, алгоритм должен обучаться самостоятельно от любого количества данных и ограничивать его значением, которое мы имеем с нашей выборкой – неправильный подход. К тому же – это не математический подход, так как в нашем случае выборка небольшая мы можем найти минимальное значение, однако это исключительный случай, нужно сделать алгоритм, который работает с любым количеством данных.



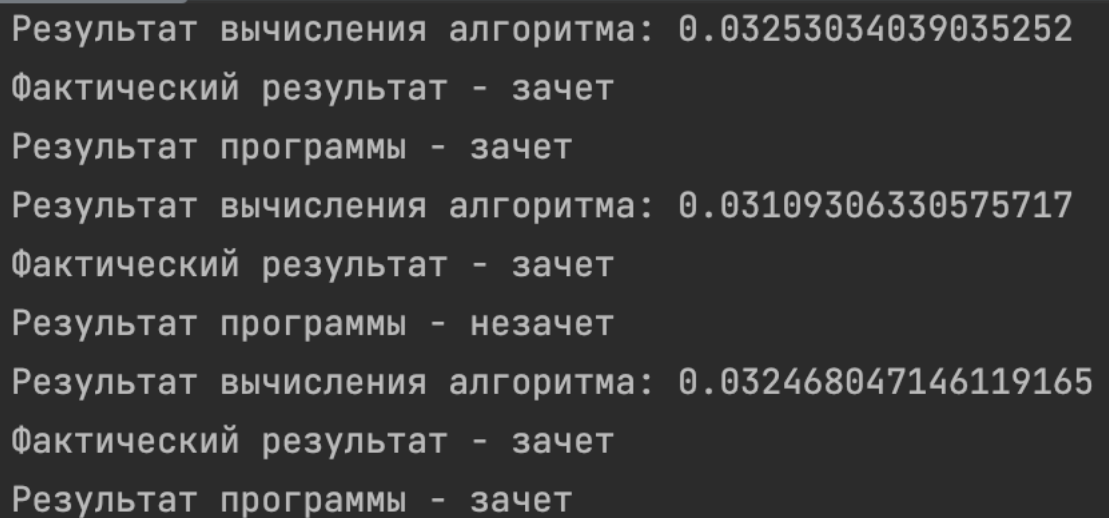
Было принято решение определять контрольное значение (далее КЗ) так:

$$K.3 = \frac{averP + averN}{2}$$

Где *averP* – среднее время среди сдавших зачет, *averN* - среднее время среди не сдавших зачет.

Модифицируем алгоритм добавив в него вычисление КЗ (Приложение 3) и начнем выводить фактический результат и результат, который нам выдал алгоритм (Рис ). Код мы пишем на языке Python с использованием библиотек *math* [3], *openpyxl* [4] и *numpy* [6].

Рис 5



```
Результат вычисления алгоритма: 0.03253034039035252
Фактический результат - зачет
Результат программы - зачет
Результат вычисления алгоритма: 0.03109306330575717
Фактический результат - зачет
Результат программы - незачет
Результат вычисления алгоритма: 0.032468047146119165
Фактический результат - зачет
Результат программы - зачет
```

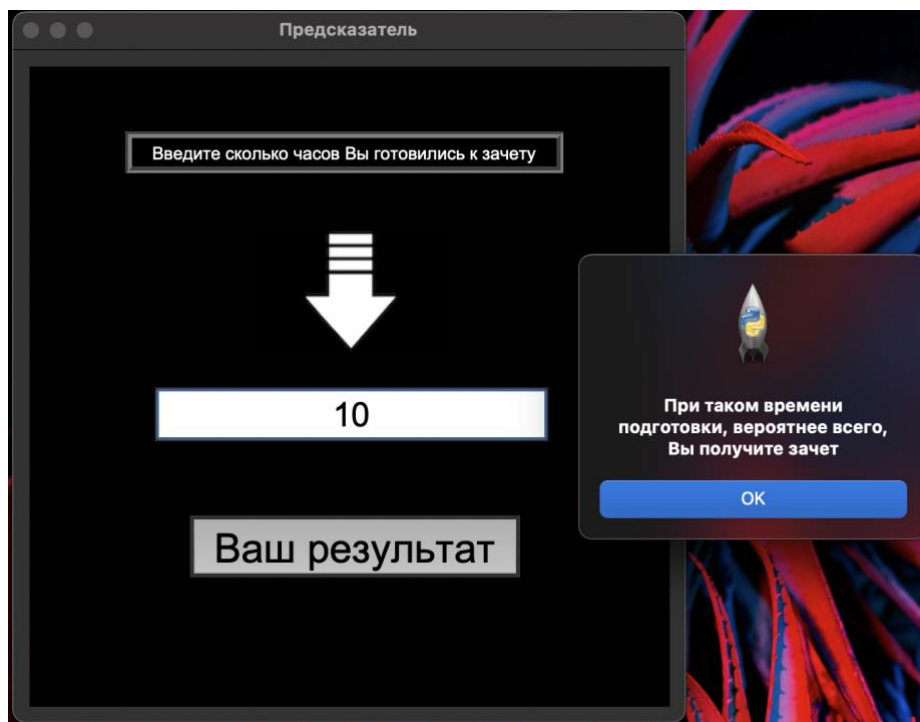
Как мы видим – одно значение посчитано алгоритмом не верно – это произошло из-за того, что результат вычисления в случае этого значения = 0.03109, а итоговое КЗ = 0.03247. Данная несостыковка происходит из-за малого количества данных для анализа, при увеличении количества записей результатов студентов КЗ должно уменьшиться и, вероятно, станет меньше, однако этого может и не произойти, так как результат зачета зависит не только от времени подготовки, но и от преподавателя и предмета, а эти параметры не рассматриваются в программе.

Итоговая точность алгоритма получилась 93,75%, 15 из 16 результатов были получены верно.

## 2. Разработка прикладной программы

Одной из задач курсового проекта является разработка прикладной программы, которой сможет пользоваться любой пользователь, а значит нам нужно создать пользовательский интерфейс. Так как это не имеет прямого отношения к Вычислительной математике, расписывать все подробно – не имеет смысла. В программе будет одна единственная функция - Возможность получить вероятный результат зачета в зависимости от количества часов, потраченных на подготовку. Делать функцию добавления данных в таблицу для обучения не имеет смысла, так как в таком случае любой пользователь сможет испортить алгоритм добавив неправдоподобные данные. При создании интерфейса использовалась библиотека tkinter [5]. Получившийся дизайн выглядит следующим образом (Рис )

Рис 6



Интерфейс выглядит достаточно понятно и просто, была выполнена проверка работы алгоритма на уже имеющихся данных – он работает точно так же, как и консольный вариант. Код программы см в Приложение

### 3. Тестирование программы и алгоритма

Для того, чтобы протестировать программу и алгоритм нужно получить дополнительные данные. Несмотря на то, что в рамках данного проекта их можно выдумать, было принято решение отправить Яндекс-форму (Рис 1) тем, кто не проходил её в первый раз. В этот раз удалось собрать 25 ответов (Рис 5), что больше, чем в первый раз. Исходя из этого, скорее всего, мы получим много неточностей в работе алгоритма, но после его проверки эти данные будут загружены в таблицу с первыми ответами и алгоритм уже будет обучаться по большему количеству данных.

Рис 5

	А	В	С
1	0	нет	нет
2	10	да	да
3	5	да	нет
4	6	нет	нет
5	8	да	да
6	1	нет	нет
7	15	да	да
8	20	да	да
9	18	нет	да
10	5	нет	нет
11	13	да	да
12	17	да	да
13	10	да	да
14	10	да	да
15	9	да	да
16	2	нет	нет
17	4	нет	нет
18	12	да	да
19	19	да	да
20	14	да	да
21	17	да	да
22	12	да	да
23	10	нет	да
24	8	да	да
25	16	да	да

В столбце “А” записано сколько часов человек готовился к зачету, в столбце “В” – его результат, а в столбце “С” – результат, выданный алгоритмом. Несмотря на опасения – он сработал довольно точно, а именно верно оценил 22/25 случаев, итоговая точность алгоритма – 88%.

Полная таблица для обучения (Приложение ) имеет 41 запись, что является весьма большим объемом данных в наших условиях. После объединения таблиц с данными и повторного тестирования алгоритма, его точность составила 90% (37/41 определены верно).

Если собрать еще большее количество данных, то, вероятно, получится немного улучшить точность, однако КЗ поменяется не сильно, так как после добавления 25 элементов к базе данных и анализа всех элементов, КЗ поменялось на несколько тысячных, а значит это значение действительно отражает среднее значение функции в зависимости от времени подготовки к зачету, при превышении которого вероятность сдать зачет сильно возрастает. Это время приблизительно равно 8-ми часам.

## ЗАКЛЮЧЕНИЕ

В результате работы были выполнены все изначальные задачи и цель. Был разработан и реализован алгоритм, который посредством приема бинаризации данных и алгоритмов искусственного обучения получает достоверные данные о результатах зачета. Более того, был разработан и реализован понятный интерфейс, благодаря которому программой теперь может пользоваться абсолютно любой человек. Стоит отметить, что обучение и тестирование проводилось на реальных данных полученных после опроса друзей и знакомых. Так же, после тестирования был сделан вывод, что алгоритм работает исправно с точностью до 90%, к тому же, данная цифра практически окончательная, так как при добавлении новой информации среднее время подготовки к зачету для его успешной сдачи практически не изменилось.

Данный алгоритм не совершенен, так как на условие сдачи зачета так же влияет несколько других очень весомых факторов, например преподаватель или что именно за предмет идет на зачет. Но работа однозначно проделана не зря, так как было получено обширное количество знаний по теме «бинаризация данных».

В качестве ответа на вопрос “Сколько нужно готовиться к зачету чтобы сильно повысить свои шансы на сдачу?” теперь можно дать обоснованный ответ – 8 часов.

### Библиографический список

1. URL: <http://900igr.net/prezentacija/biologija/makroekologija-allometricheskie-zavisimosti-i-preimuschestvennoe-vymiranie-krupnykh-vidov-v-pozdnem-plejstotsene-86536/logisticheskaja-funktsija-16.html> – (дата обращения: 12.05.2022).
2. URL: <https://neurohive.io/ru/osnovy-data-science/activation-functions/> – (дата обращения: 12.05.2022).
3. URL: <https://docs.python.org/3/library/math.html> – (дата обращения: 14.05.2022).
4. URL: <https://openpyxl.readthedocs.io/en/stable/> – (дата обращения: 14.05.2022).
5. URL: <https://docs.python.org/3/library/tkinter.html> – (дата обращения: 15.05.2022).
6. URL: <https://numpy.org> – (дата обращения: 17.05.2022).

## Приложения

### Приложение 1

```
37 # заполнение матриц Y, P
38 def fill_matrix():
39     len_of_arrays = len(all_values)
40     global P
41     P = np.array([[0.00] * 2 for i in range(len_of_arrays)])
42     global Y
43     Y = np.array([[0] * 1 for i in range(len_of_arrays)])
44     global W
45     W = np.array([[1.00] * len_of_arrays for i in range(len_of_arrays)])
46
47     for i in range(len_of_arrays):
48         for j in range(2):
49             if j % 2 == 0:
50                 P[i][j] = 1
51             else:
52                 P[i][j] = xs(all_values[i])
53
54     for i in range(len_of_arrays):
55         if all_results[i]:
56             Y[i][0] = 1
57         else:
58             Y[i][0] = 0
77 # метод наименьших квадратов, используется для вычисления аргументов a0 и a1
78 def square_method(Y, P):
79     temp = np.array(linalg.inv(((P.transpose()).dot(np.linalg.matrix_power(W, 2))).dot(P)))
80     arguments = temp.dot(P.transpose().dot(np.linalg.matrix_power(W, 1)).dot(Y))
81
82     a[0] = arguments[0]
83     a[1] = arguments[1]
```

### Приложение 2

```
61 # расчет веса - чем дальше значение от эталонного, тем меньше его значимость
62 def weight():
63     global W
64     for i in range(len(positive) + len(negative)):
65         W[i][i] = 1 - (math.fabs(func(all_results[i] * 30) - func(all_values[i])))
66         # если результат незачет (0), то имеем выражение 0 * 30 = 0, результат зачет - 1 * 30 = 30
```

### Приложение 3

```

83     # замена для x
84     def xs(x):
85         return math.exp(-x)
86
87
88     # функция по которой работает алгоритм
89     def func(x):
90         return a[0] + a[1] * xs(x)
91
92
93     # в этой функции происходит сравнение переданного в функцию значения и К.3.
94     def artificial_result(cv, x):
95         if func(x) >= cv:
96             return 'зачет'
97         else:
98             return 'незачет'
99
100
101     def main():
102         imp() # получаем все данные, по которым будем обучать алгоритм, из таблицы
103         fill_matrix() # заполняем матрицы значениями
104         square_method(False, Y, P) # первый раз ищем аргументы функции (без весов)
105         weight() # находим веса для каждого значения ищ нашей базы данных
106         square_method(True, Y, P) # второй прогон - с весами для каждого значения
107         control_value = func(int((aver[0] / len(positive)) + (aver[1] / len(negative))) / 2)) # находим К3
108         return control_value

```

```

1 from tkinter import *
2 from tkinter import messagebox
3 from calculations import main, artificial_result
4
5
6 def click():
7     imp = int(Input.get())
8     text = artificial_result(control_value, imp)
9     messagebox.showinfo(title='Результат', message=f'При таком времени подготовки, вероятнее всего, Вы получите {text}')
10
11
12 control_value = main()
13 root = Tk()
14 ar = PhotoImage(file='arrow.gif')
15
16 root['bg'] = '#000000'
17 root.title('Предсказатель')
18 root.geometry('500x500')
19 root.resizable(width=False, height=False)
20
21 canvas = Canvas(root, width=500, height=500)
22 canvas.pack()
23
24 frame = Frame(root, bg='#010101')
25 frame.place(relx=0.025, rely=0.025, relwidth=0.95, relheight=0.95)
26 title = Label(frame, text='Введите сколько часов Вы готовились к зачету', bg='black', bd='5',
27               relief='groove', font='Arial 15')
28 title.place(relx=0.15, rely=0.1)
29 arrow = Label(frame, image=ar)
30 arrow.place(relx=0.35, rely=0.25, relwidth=0.3, relheight=0.2)
31 Input = Entry(frame, fg='#010101', bg='white', font='Arial 25', justify='center')
32 Input.place(relx=0.195, rely=0.5)
33 butt = Button(frame, text='Вас результат', font='Arial 30', justify='center', command=click)
34 butt.place(relx=0.25, rely=0.7)
35
36 root.mainloop()

```

### Приложение 4

```

1 from openpyxl import load_workbook
2 import math
3 from array import *
4 import numpy as np
5 from numpy import linalg
6
7 positive = array('B', []) # создаем массив со значениями x при которых зачет сдан
8 negative = array('B', []) # при этих значениях x зачет не сдан
9 all_values = array('B', []) # массив всех значений
10 all_results = array('B', []) # массив всех результатов (1 - зачет, 0 - незачет)
11 aver = array('f', [0, 0]) # первое значение массива - среднее арифм среди положительных
12 a = array('f', [0, 0]) # аргументы функции
13 P = np.array([]) # матрица P (нужна в методе наименьших квадратов)
14 Y = np.array([]) # матрица Y (нужна в методе наименьших квадратов)
15 W = np.array([]) # матрица весов (нужна в методе наименьших квадратов)
16
17
18 # функция, которая импортирует все значения из таблицы в массивы
19 def imp():
20     global aver
21     sheet = load_workbook("data.xlsx")['Sheet1']
22     rows = sheet.max_row
23
24     for index in range(2, rows + 1):
25         value = int(sheet[f'A{index}'].value)
26         all_values.append(value)
27         if sheet[f'B{index}'].value == 'да':
28             aver[0] += value
29             positive.append(value)
30             all_results.append(1)
31         elif sheet[f'B{index}'].value == 'нет':
32             aver[1] += value
33             negative.append(value)
34             all_results.append(0)

```



```
37 # заполнение матриц Y, P
38 def fill_matrix():
39     len_of_arrays = len(all_values)
40     global P
41     P = np.array([[0.00] * 2 for i in range(len_of_arrays)])
42     global Y
43     Y = np.array([[0] * 1 for i in range(len_of_arrays)])
44     global W
45     W = np.array([[1.00] * len_of_arrays for i in range(len_of_arrays)])
46
47     for i in range(len_of_arrays):
48         for j in range(2):
49             if j % 2 == 0:
50                 P[i][j] = 1
51             else:
52                 P[i][j] = xs(all_values[i])
53
54     for i in range(len_of_arrays):
55         if all_results[i]:
56             Y[i][0] = 1
57         else:
58             Y[i][0] = 0
59
60 # расчет веса - чем дальше значение от эталонного, тем меньше его значимость
61 def weight():
62     global W
63     for i in range(len(positive) + len(negative)):
64         W[i][i] = 1 - (math.fabs(func(all_results[i] * 15) - func(all_values[i])))
65         # если результат незначителен (0), то имеем выражение 0 * 30 = 0, результат значителен - 1 * 30 = 30
66
67 # метод наименьших квадратов, используется для вычисления аргументов a0 и a1
68 def square_method(check, Y, P):
69     if not check:
70         arguments = np.array((linalg.inv((P.transpose()).dot(P))).dot(P.transpose()).dot(Y))
```

```
70 def square_method(check, Y, P):
71     if not check:
72         arguments = np.array((linalg.inv((P.transpose()).dot(P))).dot(P.transpose()).dot(Y))
73     else:
74         temp = np.array(linalg.inv((P.transpose()).dot(np.linalg.matrix_power(W, 2))).dot(P))
75         arguments = temp.dot(P.transpose()).dot(np.linalg.matrix_power(W, 1)).dot(Y)
76
77     a[0] = arguments[0]
78     a[1] = arguments[1]
79
80 # замена для x
81 def xs(x):
82     return math.exp(-x)
83
84 # функция по которой работает алгоритм
85 def func(x):
86     return a[0] + a[1] * xs(x)
87
88 # в этой функции происходит сравнение переданного в функции значения и K3.
89 def artificial_result(cv, x):
90     if func(x) >= cv:
91         return 'зачет'
92     else:
93         return 'незачет'
94
95 def main():
96     imp() # получаем все данные, по которым будем обучать алгоритм, из таблицы
97     fill_matrix() # заполняем матрицы значениями
98     square_method(False, Y, P) # первый раз ищем аргументы функции (без весов)
99     weight() # находим веса для каждого значения из нашей базы данных
100     square_method(True, Y, P) # второй прогон - с весами для каждого значения
101     control_value = func(int(((aver[0] / len(positive)) + (aver[1] / len(negative))) / 2)) # находим K3
102     return control_value
```

	A	B
1	time	result
2	5	нет
3	2	нет
4	1	нет
5	15	да
6	20	да
7	16	да
8	10	да
9	5	нет
10	20	да
11	12	да
12	15	да
13	3	нет
14	0	нет
15	7	да
16	12	да
17	11	да
18	0	нет
19	10	да
20	5	да

Приложение 5

21	6	нет
22	8	да
23	1	нет
24	15	да
25	20	да
26	18	нет
27	5	нет
28	13	да
29	17	да
30	10	да
31	10	да
32	9	да
33	2	нет
34	4	нет
35	12	да
36	19	да
37	14	да
38	17	да
39	12	да
40	10	нет
41	8	да
42	16	да