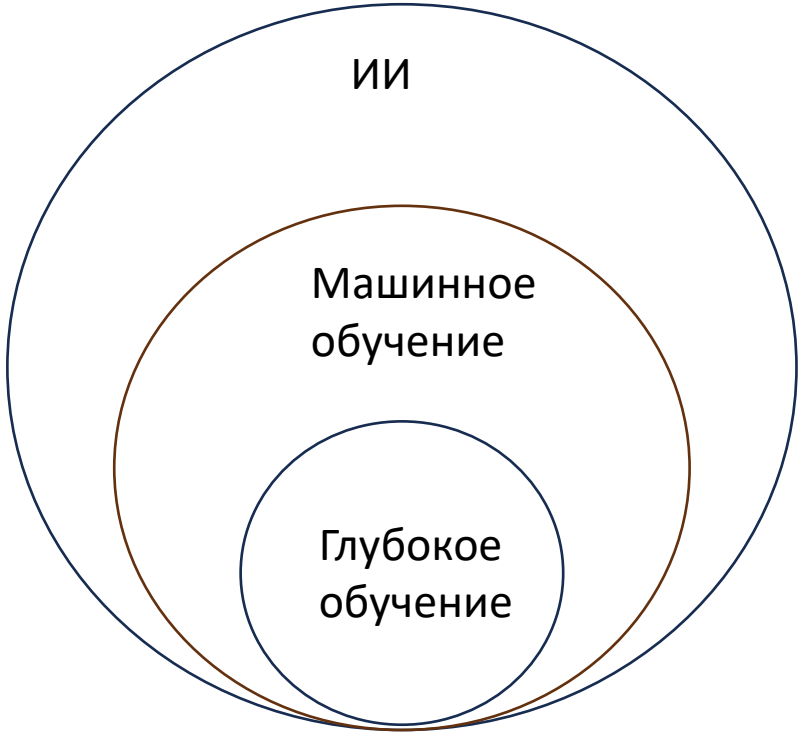
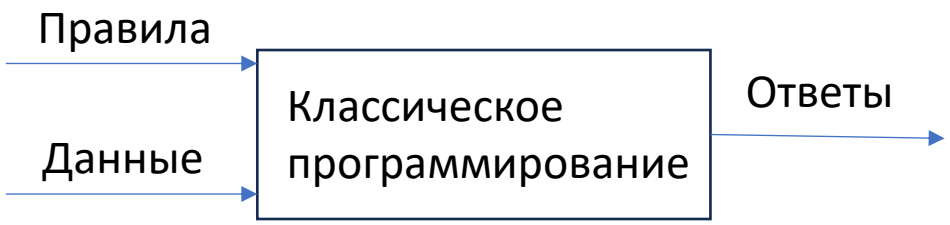
Введение в глубокое обучение

Искусственный интеллект



Идея ИИ зародилась в 1950х гг с вопроса – можно ли заставить компьютеры «думать»?
ИИ – это набор методов, позволяющие выполнять интеллектуальные задачи, которые выполняют люди.
До 1980х гг методов ИИ на основе обучения не существовало. Популярный подход – достаточный набор явных правил для обработки данных – символический ИИ (логические игры, например, шахматы).
В большинстве задач строгие правила задать невозможно (распознавание изображений, речи и т.д.) – машинное обучение.

Машинное обучение



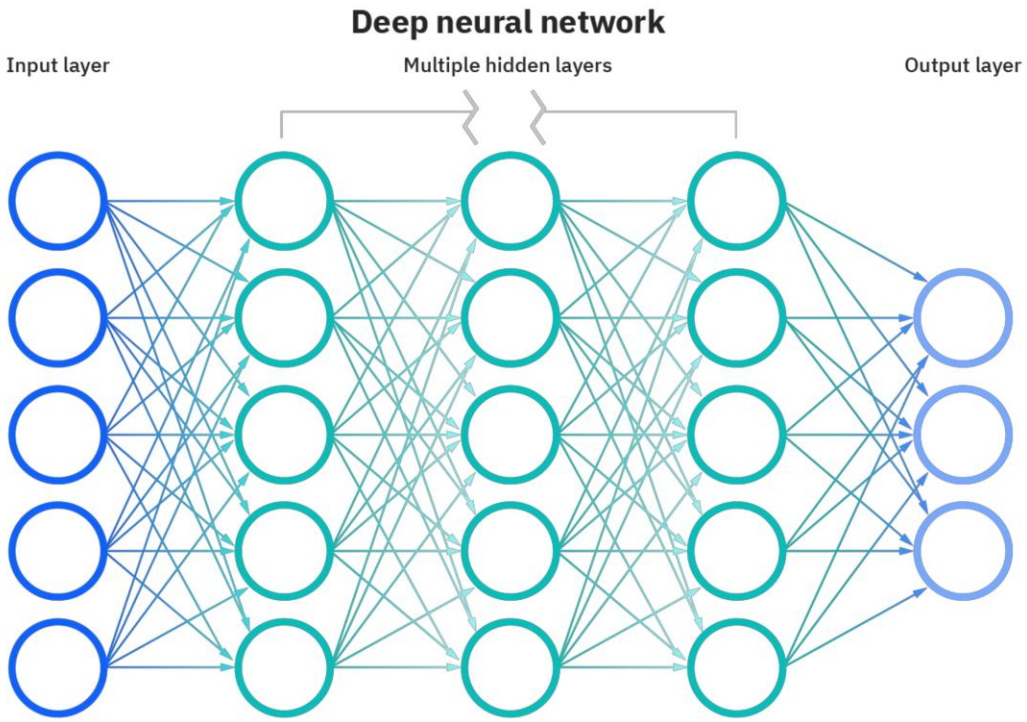
В классическом программировании нужно создать правила, чтобы входные данные преобразовать в ответы.
В машинном обучении система обучается, а не программируется явно.

Машинное обучение тесно связано со статистикой, поиском корреляций между входными данными и ответами. Однако, работает с гораздо большими наборами данных.

Машинное обучение – это практичная сфера, основанная на эмпирических данных и достижениях в области информатики и вычислительной техники.

Глубокое обучение

- Подмножество методов машинного обучения

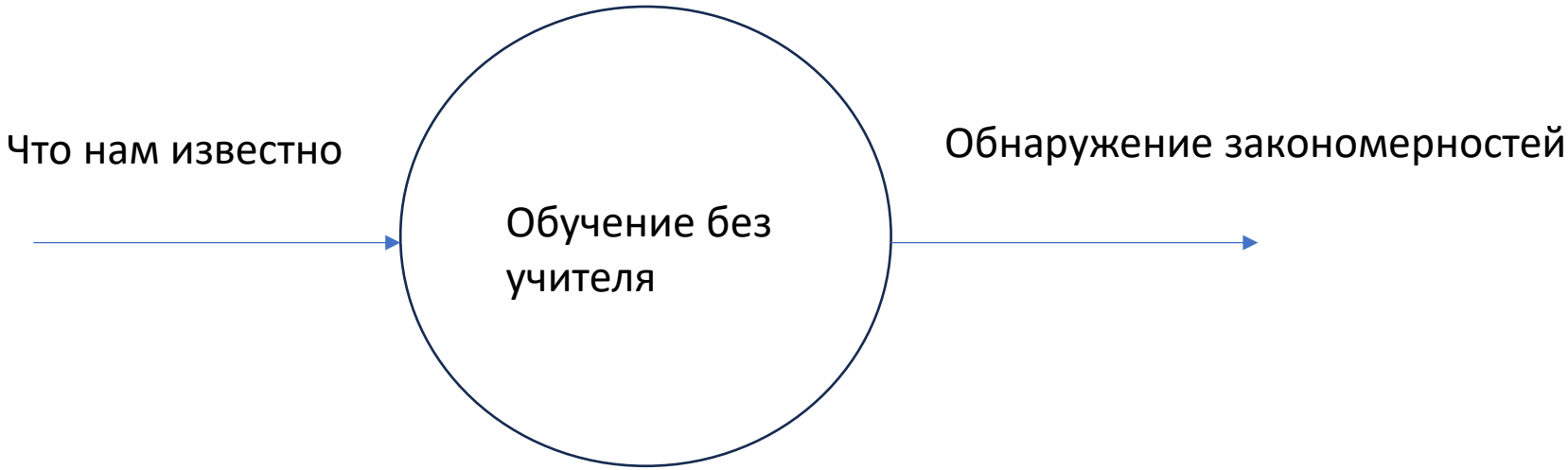


Машинное обучение с учителем

- Метод преобразования одного набора данных в другой

Машинное обучение без учителя

- Метод преобразования одного набора данных в другой



Результат работы – прежде не был известен
Здесь нет «правильного» ответа – нейросеть находит закономерности во входных данных и сообщает их
Кластеризация набора данных на группы – пример обучения без учителя

Параметрическое и непараметрическое обучение

- **Параметрическая модель характеризуется наличием фиксированного числа параметров и обычно используют подход «проб и ошибок»**
- Непараметрическая модель имеет число параметров, которые задаются входными данными и, как правило, основаны на вычислениях.

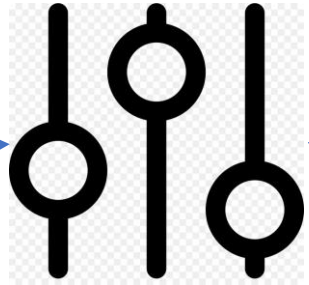
Параметрическое обучение с учителем

- Обучение методом «проб и ошибок» с использованием регуляторов.

Этап 1: Прогноз

Данные

- Стресс: 10 баллов
- Вредные привычки: 2
- Физическая активность: 3



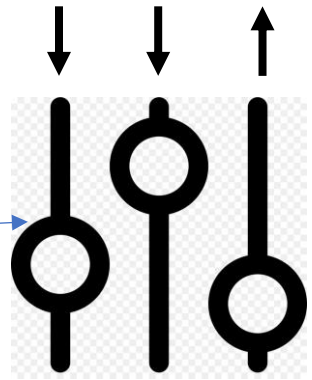
Предсказание

Вероятность инфаркта
96%

Этап 2: Сравнение с истиной
Прогноз – 96% - Истина (Инфаркт не случился)
Нужно дообучение модели.

Этап 3: обучение

- Стресс: 10
- Вредные привычки: 2
- Физическая активность: 3

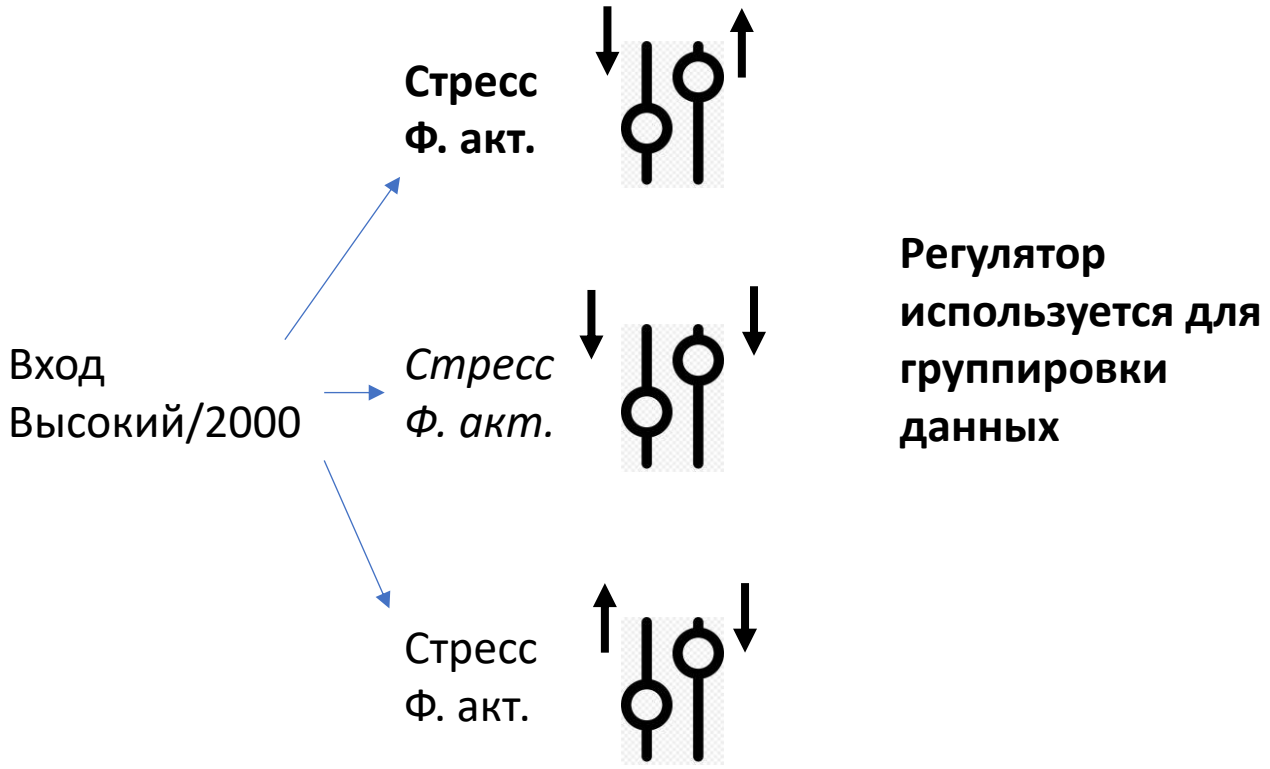


Вероятность инфаркта
34%

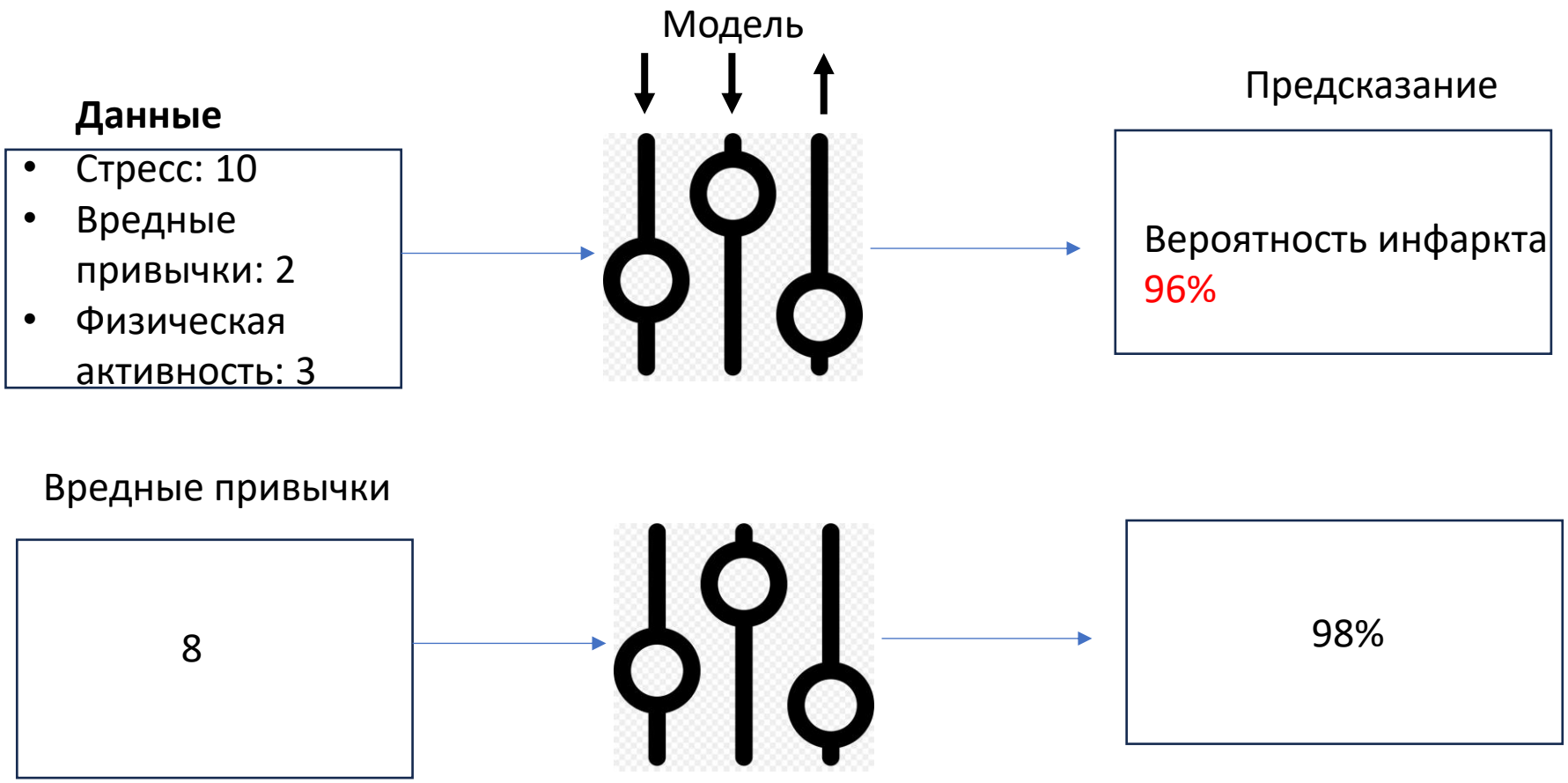
Регулятор представляет собой чувствительность прогноза к разным типам входных данных

Параметрическое обучение без учителя

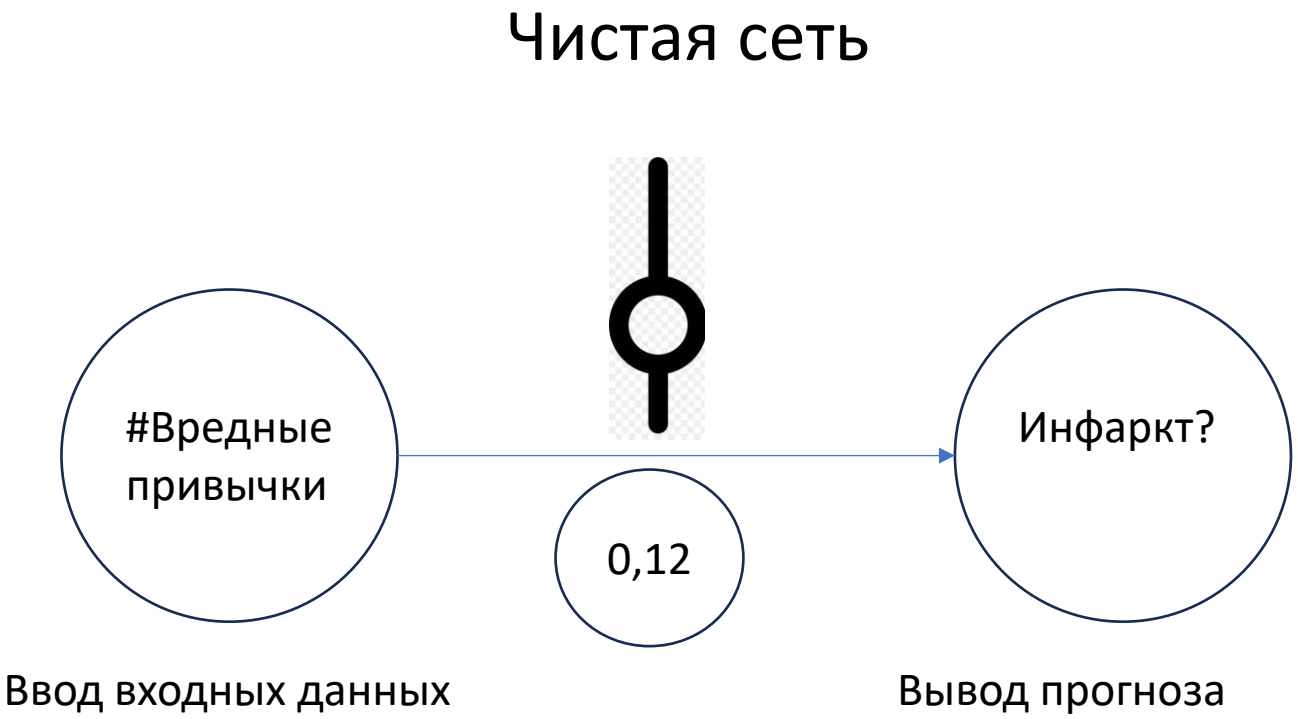
Стресс	Физическая активность, шагов в день
Высокий	*10 000*
Низкий	*5 000*
Средний	**7 000**
Низкий	**10 000**
Высокий	2 000
Низкий	*2 000*
Средний	1 000



Прямое распространение. Шаг 1: прогнозирование



Шаг 1: прогнозирование



Регуляторы называют «**весами**» или «**весовыми коэффициентами**»

Что такое нейронная сеть?

```
weight = 0.12
def neural_network(input, weight):
        prediction = input*weight
        return prediction

bad_habits = [8, 4, 2, 6]
input = bad_habits[0]
pred = neural_network(input, weight)
print(pred)
```

Нейронная сеть – по сути, это один или несколько *весовых коэффициентов (мера чувствительности)*, на которые нужно умножить *входные данные*, чтобы получить *прогноз*.

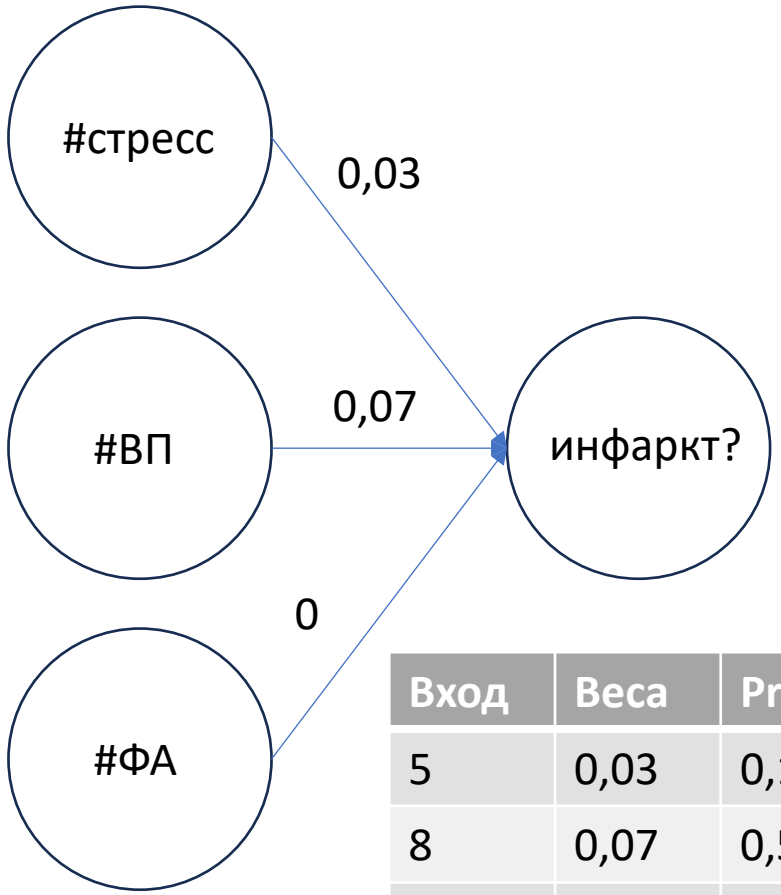
Входные данные – это числа из реального мира.

Прогноз (предсказание) – это то, что возвращает нейронная сеть после получения входных данных. Прогноз м.б. неверен, но нейросеть на нем обучается.

Нейросеть обучается методом проб и ошибок.

НС «масштабирует» входное значение на определенную величину.
Данная нейронная сеть «забудет» предыдущий прогноз, если подать другое значение на вход.

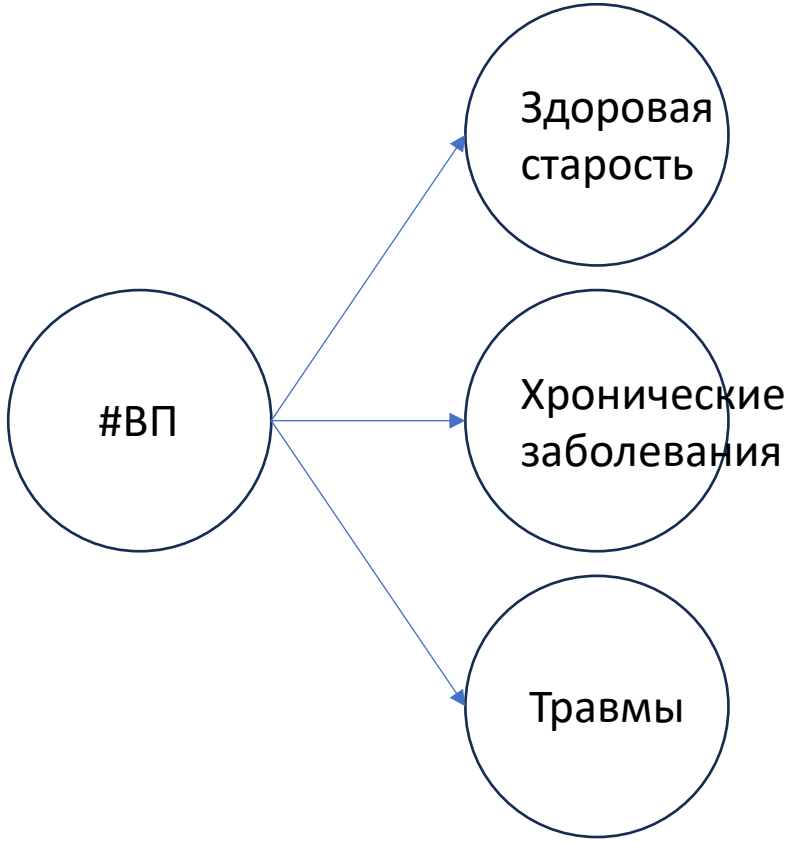
Несколько входов



Вход	Веса	Pred	
5	0,03	0,15	Стресс
8	0,07	0,56	ВП
1	0	0	ФА
		0,71	

```
#Каждый вход умножается на соответствующий ему вес,
#после чего результаты суммируются: взвешенная сумма
#(скалярное произведение)
def w_sum(a, b):
# ….

def neural_network(input, weight):
        prediction = w_sum(input,weight)
        return prediction
weights = [0.03, 0.07, 0]

stress = [5, 8, 15, 6, 30]
bad_habits = [8, 4, 2, 6, 1]
phys_active =  [1, 1, 2, 1, 3]
input = [stress[0], bad_habits[0], phys_active[0]]

pred = neural_network(input, weights)
```

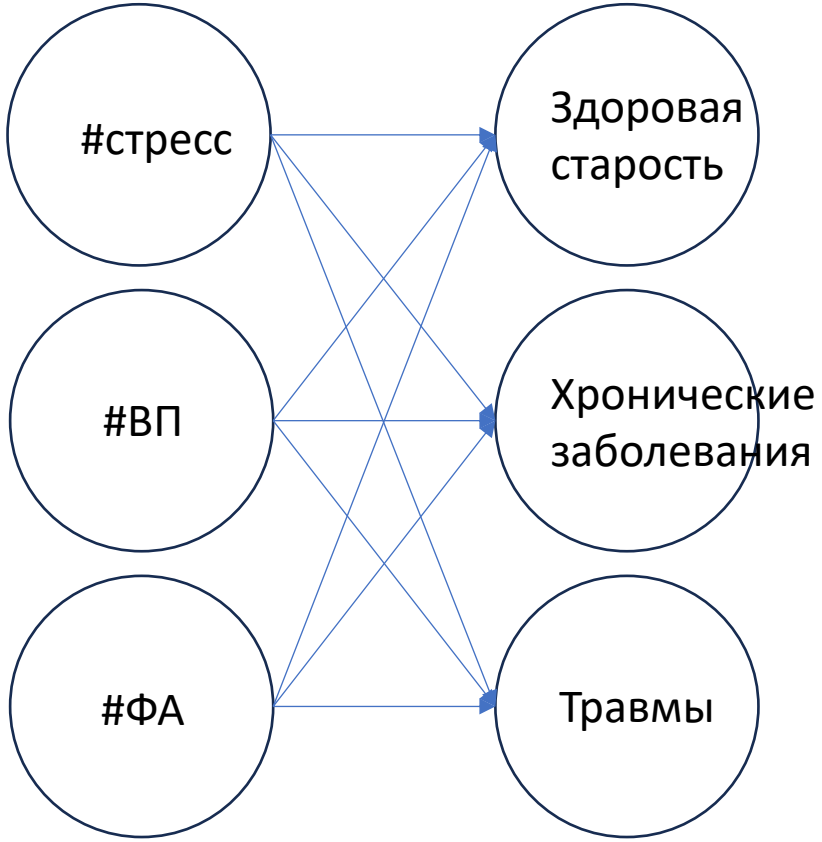
Несколько выходов




```
#Каждый вход умножается на соответствующий ему вес,
#после чего результаты суммируются: взвешенная сумма
#(скалярное произведение)
weights = [0.01, 0.09, 0.01]

def ele_mul(number, vector):
# ….
def neural_network(input, weight):
        prediction = ele_mul(input,weight)
        return prediction

bad_habits = [8, 4, 2, 6, 1]
input = bad_habits[0]

pred = neural_network(input, weights)
```

Вход	Веса	Pred	
8	0,01	0,08	Здоровая старость
8	0,23	0,72	Хронические заболевания
8	0,09	0,08	Травмы

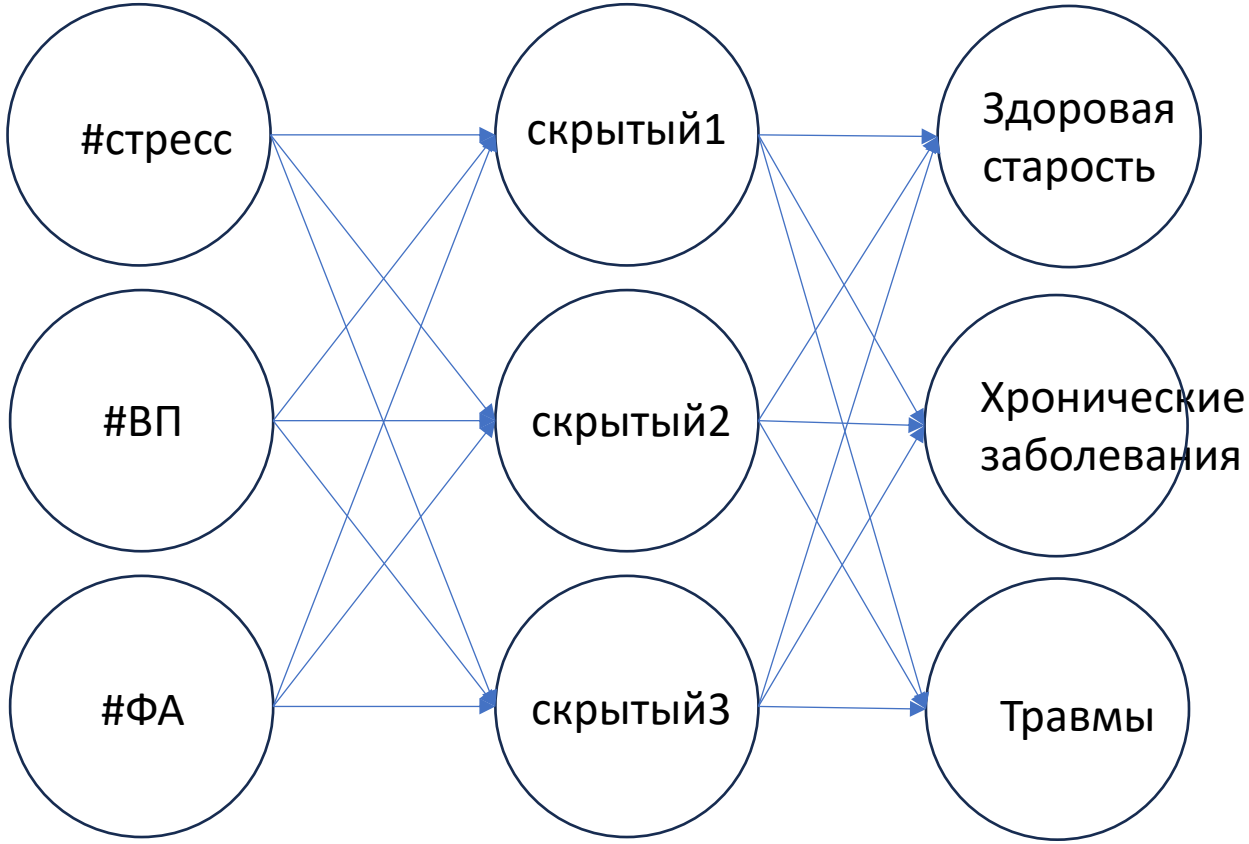
Несколько входов и выходов



Данную НС можно представить как три независимых **скалярных произведения – ф-я vect_mat_mul**

```
                #стресс   #ВП      #ФА
weights = [[0.01,      0.09,     0.01],  # здоровая старость
           [0.01,      -0.04,     0.00],   # хронические забол.
           [0.00,      0.17,     0.03]]   # травмы

stress = [5, 8, 15, 6, 30]
bad_habits = [8, 4, 2, 6, 1]
phys_active =  [1, 1, 2, 1, 3]
input = [stress[0], bad_habits[0], phys_active[0]]

# для КАЖДОГО выхода вычисляется взвешенная сумма
# входов
def w_sum(a, b):
#...
def vect_mat_mul(vect, matrix):
#...
def neural_network(input, weight):
        prediction = vect_mat_mul(input,weight)
        return prediction
pred = neural_network(input, weights)
```

Несколько входов и выходов




```
                #стресс   #ВП      #ФА
weights1 = [[0.01,      0.09,     0.01],  # скрытый1
            [0.01,     -0.04,     0.00],   # скрытый2
            [0.00,      0.17,     0.03]]   # скрытый3

                #скрыт1  #скрыт2  #скрыт3
weights2 = [[0.06,      0.02,     -0.01],  # здоровая старость
            [0.00,      0.10,      0.00],   # хронические забол.
            [0.02,     -0.17,      0.04]]   # травмы

weights = [weights1, weights2]

# входные значения
stress = [5, 8, 15, 6, 30]
bad_habits = [8, 4, 2, 6, 1]
phys_active =  [1, 1, 2, 1, 3]
input = [stress[0], bad_habits[0], phys_active[0]]
```

Данную НС можно представить как три независимых **скалярных произведения – ф-я vect_mat_mul**

Нейронная сеть со скрытым слоем

```
def neural_network(input, weights):
        pred_hid = vect_mat_mul(input,weights[0])
        pred_out = vect_mat_mul(pred_hid,weights[1])
        return prediction

pred = neural_network(input, weights)
```

Заключение

Для того, чтобы получить прогноз нейронные сети многократно вычисляют взвешенную сумму для входных данных.

На данной лекции рассмотрено прямое распространение информации.

Далее: Как повысить точность прогноза нейронной сети? Настройка весов (обучение). Градиентный спуск.

Глубокие нейронные сети. Обратное распространение ошибки.