**Ордена Трудового Красного Знамени**

**Федеральное государственное бюджетное образовательное учреждение**

**Высшего профессионального образования**

**Московский технический университет связи и информатики**

Факультет повышения квалификации

**Лабораторная работа**

**по дисциплине**

**«Тестирование программных продуктов»**

Выполнил:

магистрант гр. 3МПП1901

Иванюк П. Г.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись дата

Проверил:

к.т.н., доцент каф. МКиИТ

Махров С.С.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись дата

**Москва 2021**

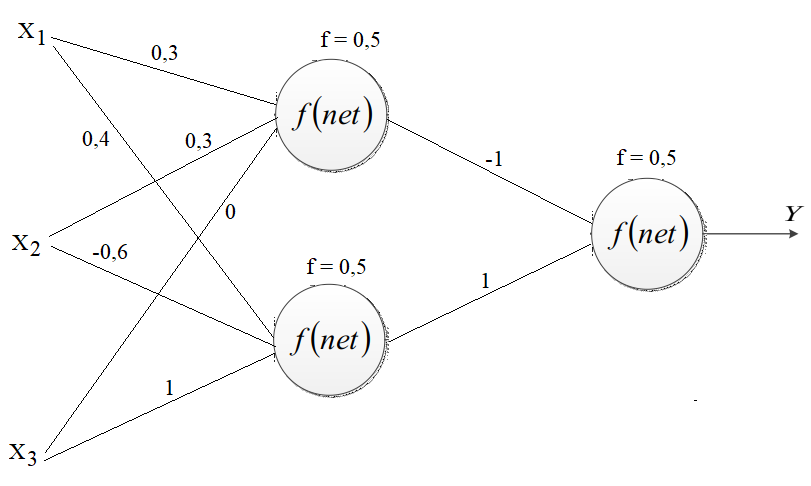
# **Цель работы:**

Изучение принципов проведения тестирования программных продуктов.

# **Описание выбранной предметной области:**

Для проведения тестирования возьмем программу полносвязной нейронной сети, написанную на Python без использования дополнительных библиотек.

Программа создает полносвязную нейронную сеть прямого распространения с двумя слоями и с тремя нейронами. Для активации используется пороговая функция с порогом равны 0,5.



Пусть нейронная сеть определяет, как девушка выбирает себе парня исходя из следующих трех предпочтений.

Вход X1 – если единица у парня есть квартира, если ноль то квартиры нет.

Вход X2 – если единица значит парень слушает тяжелый рок, а ноль значит парень слушает классическую музыку.

Вход X3 – если единица значит парень симпатичный, иначе парень не симпатичный.

1. Создаем данную нейронную сеть
2. Обучаем эту нейронную сеть заранее определёнными значениями весов.
3. Задаём список всех возможных входных данных.
4. Проведём тестирование данной нейронной сети на этих наборах данных.
5. Получим структуру данной нейронной сети.

**Программа для тестирования нейронной сети**

import pytest

from function.neuronnet import \*

n = net(3, 2, 1)

n.weights(1, 0, 0, 0.3)

n.weights(1, 0, 1, 0.3)

n.weights(1, 0, 2, 0)

n.weights(1, 1, 0, 0.4)

n.weights(1, 1, 1, -0.6)

n.weights(1, 1, 2, 1)

n.weights(2, 0, 0, -1)

n.weights(2, 0, 1, 1)

p = [(0, 0, 0), (0, 0, 1), (0, 1, 0), (0, 1, 1), (1, 0, 0), (1, 0, 1), (1, 1, 0), (1, 1, 1)]

@pytest.mark.parametrize('a', p)

def test\_neuron\_predict(a):

t = [[], [], []]

if a[0] == 1:

t[0] = 'с квартирой'

else:

t[0] = 'без квартиры'

if a[1] == 1:

t[1] = 'и слушает рок музыку,'

else:

t[1] = 'и слушает классическую музыку,'

if a[2] == 1:

t[2] = 'Симпатичный парень'

else:

t[2] = 'Уродливый парень'

v = n.predict(a)

if v[0] == 1:

print('\n', t[2], t[0], t[1], 'ты мне нравишься, давай дружить')

else:

print('\n', t[2], t[0], t[1], 'мы с тобой созвонимся может быть потом')

def test\_print\_summary\_neuronnet():

print(n.summary())

**Вывод работы программы**

============================= test session starts ===============

collecting ... collected 9 items

tests/test\_neuronnet.py::test\_neuron\_predict[a0] PASSED [ 11%]

Уродливый парень без квартиры и слушает классическую музыку, мы с тобой созвонимся может быть потом

tests/test\_neuronnet.py::test\_neuron\_predict[a1] PASSED [ 22%]

Симпатичный парень без квартиры и слушает классическую музыку, ты мне нравишься, давай дружить

tests/test\_neuronnet.py::test\_neuron\_predict[a2] PASSED [ 33%]

Уродливый парень без квартиры и слушает рок музыку, мы с тобой созвонимся может быть потом

tests/test\_neuronnet.py::test\_neuron\_predict[a3] PASSED [ 44%]

Симпатичный парень без квартиры и слушает рок музыку, мы с тобой созвонимся может быть потом

tests/test\_neuronnet.py::test\_neuron\_predict[a4] PASSED [ 55%]

Уродливый парень с квартирой и слушает классическую музыку, мы с тобой созвонимся может быть потом

tests/test\_neuronnet.py::test\_neuron\_predict[a5] PASSED [ 66%]

Симпатичный парень с квартирой и слушает классическую музыку, ты мне нравишься, давай дружить

tests/test\_neuronnet.py::test\_neuron\_predict[a6] PASSED [ 77%]

Уродливый парень с квартирой и слушает рок музыку, мы с тобой созвонимся может быть потом

tests/test\_neuronnet.py::test\_neuron\_predict[a7] PASSED [ 88%]

Симпатичный парень с квартирой и слушает рок музыку, мы с тобой созвонимся может быть потом

tests/test\_neuronnet.py::test\_print\_summary\_neuronnet PASSED [100%]

============ Слой нейронной сети: 1 =================

============ Слой нейронной сети: 2 =================

Нейронов= 2

Вычисляемых параметров= 6

============ Слой нейронной сети: 3 =================

Нейронов= 1

Вычисляемых параметров= 2

==================================================

Всего вычисляемых параметров= 8

None

============================== 9 passed in 0.05s ===============

Process finished with exit code 0

**Программа полносвязной нейронной сети**

# # coding: utf-8

# ## Полносвязная нейронная сеть

# # подключение библиотек

# import math

# from random import random

# # Класс нейрон

# class neuron:

# def \_\_init\_\_(self, x):

# self.\_\_x = x

# self.\_\_net = 0.0

# self.\_\_y = 0.0

# def get\_x(self):

# return self.\_\_x

# def set\_x(self, x):

# self.\_\_x = x

# def get\_y(self):

# return self.\_\_y

# def calc(self):

# if type(self.\_\_x) == list:

# # Цикл суммирования

# for i in self.\_\_x:

# self.\_\_net += i[0].get\_y() \* i[1]

# # Цикл активации

# if self.\_\_net >= 0.5:

# self.\_\_y = 1

# else:

# self.\_\_y = 0

# self.\_\_net = 0.0

# else:

# self.\_\_y = self.\_\_x

# # Построение нейронной сети

# class net:

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

# self.\_\_wcount = args # Список переданных аргументов

# self.\_\_NN = [] # Список всей сети

# for i in self.\_\_wcount:

# self.\_\_NN.append([neuron([]) for n in range(i)])

# for i in range(len(self.\_\_NN)):

# for n in self.\_\_NN[i]:

# n.set\_x(0.0) if i == 0 else n.set\_x([[x, random()] for x in self.\_\_NN[i - 1]])

# def summary(self):

# iii = []

# for i in range(len(self.\_\_NN)):

# print('\n============ Слой нейронной сети:', i + 1, '=================')

# if i > 0:

# print('Нейронов=', len(self.\_\_NN[i]))

# ii = len(self.\_\_NN[i]) \* len(self.\_\_NN[i - 1])

# iii.append(ii)

# print('Вычисляемых параметров=', ii)

# print('==================================================')

# print('Всего вычисляемых параметров=', sum(iii))

# def weights(self, a, b, c, d):

# q = self.\_\_NN[a][b].get\_x()

# q[c][1] = d

# self.\_\_NN[a][b].set\_x(q)

# def predict(self, a):

# for i, j in zip(self.\_\_NN[0], a): i.set\_x(float(j))

# for i in range(len(self.\_\_NN)):

# for j in self.\_\_NN[i]: j.calc()

# n = [n.get\_y() for n in self.\_\_NN[-1]]

# return n

# **Вывод:**

В результате выполнения лабораторной работы, были изучены принципы проведения тестирования программых продуктов, написанных на Python.