Министерство образования Республики Беларусь

Учреждение образования

«Брестский Государственный технический университет»

Кафедра ИИТ

**Лабораторная работа №3**

По дисциплине «Модели решения задач в интеллектуальных системах»

Тема: «Классификация»

**Выполнил:**

Студент 3 курса

Группы ИИ-21

Кирилович А. А.

**Проверил:**

Туз И. С.

Брест 2023

**Цель:** классифицировать данные с помощью random forest и персептрона с одним скрытым слоем. Сравнить скорость, точность обучения моделей.

**Ход работы**

**Вариант 4**

Датасет: [ionosphere](https://www.kaggle.com/datasets/prashant111/ionosphere).

import numpy as np

class Perceptron:

def \_\_init\_\_(self, input\_size, hidden\_size, output\_size, learning\_rate=0.1):

self.input\_size = input\_size

self.hidden\_size = hidden\_size

self.output\_size = output\_size

self.learning\_rate = learning\_rate

self.weights\_input\_hidden = np.random.randn(self.input\_size, self.hidden\_size)

self.bias\_hidden = np.zeros((1, self.hidden\_size))

self.weights\_hidden\_output = np.random.randn(self.hidden\_size, self.output\_size)

self.bias\_output = np.zeros((1, self.output\_size))

def sigmoid(self, x):

return 1 / (1 + np.exp(-x))

def sigmoid\_derivative(self, x):

return x \* (1 - x)

def forward(self, inputs):

self.hidden\_input = np.dot(inputs, self.weights\_input\_hidden) + self.bias\_hidden

self.hidden\_output = self.sigmoid(self.hidden\_input)

self.output = np.dot(self.hidden\_output, self.weights\_hidden\_output) + self.bias\_output

return self.output

def backward(self, inputs, target, output):

error = target - output

delta\_output = error

delta\_hidden = delta\_output.dot(self.weights\_hidden\_output.T) \* self.sigmoid\_derivative(self.hidden\_output)

self.weights\_hidden\_output += self.hidden\_output.T.dot(delta\_output) \* self.learning\_rate

self.bias\_output += np.sum(delta\_output, axis=0, keepdims=True) \* self.learning\_rate

self.weights\_input\_hidden += inputs.T.dot(delta\_hidden) \* self.learning\_rate

self.bias\_hidden += np.sum(delta\_hidden, axis=0, keepdims=True) \* self.learning\_rate

def train(self, inputs, targets, epochs):

for epoch in range(epochs):

for i in range(len(inputs)):

input\_data = np.array([inputs[i]])

target\_data = np.array([targets[i]])

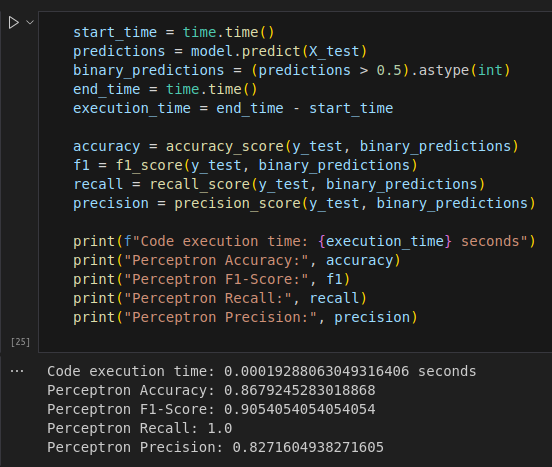
output = self.forward(input\_data)

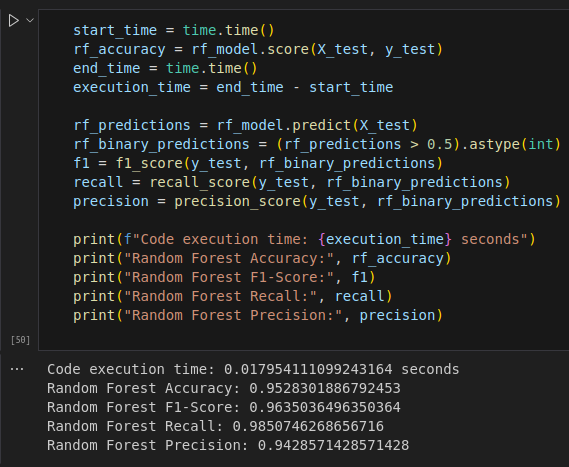
self.backward(input\_data, target\_data, output)

def predict(self, inputs):

output = self.forward(inputs)

return output





**Вывод:** в ходе лабораторной работы я классифицировал данные с помощью random forest и персептрона с одним скрытым слоем. Сравнил скорость, точность обучения моделей.