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

Литвинюк Т. В.

**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 7**

Датасет: diabet

Персептрон с одним скрытым слоем:

*import* numpy *as* np

*from* sklearn.model\_selection *import* train\_test\_split

*from* sklearn.metrics *import* accuracy\_score, classification\_report

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

*# Инициализация весов для слоев*

*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):

*# Прямое распространение сигнала*

*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):

*# Обратное распространение ошибки*

        error = target - output

*# Вычисление градиентов для обновления весов*

        delta\_output = error

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

*# Обновление весов и смещений*

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

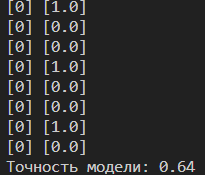
*self*.bias\_output += np.sum(delta\_output, axis=0, keepdims=True) \* *self*.learning\_rate

*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]])

*# Прямое и обратное распространение*

                output = *self*.forward(input\_data)

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

    def predict(self, inputs):

*# Предсказание классов на основе входных данных*

        output = *self*.forward(inputs)

*return* output

*# Пример использования:*

*if* \_\_name\_\_ == "\_\_main\_\_":

*import* csv

*# Создаем нейросеть с 8 входами, 4 нейронами в скрытом слое и 1 выходом*

    perceptron = Perceptron(8, 5, 1)

    X, Y = [], []

*# Генерируем случайные данные для обучения*

*with* open(r"MRZIS\lab3\diabetes.csv") *as* file:

        reader = csv.reader(file)

*for* i, row *in* enumerate(reader):

            X.append([float(value) *for* value *in* row[:-1]])

            Y.append([float(row[-1])])

    X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, Y, test\_size=0.2, random\_state=42)

*# Обучаем нейросеть*

    perceptron.train(X\_train, y\_train, 100)

*# Прогнозируем значения*

    y\_pred = (perceptron.predict(X\_test) > 0.5).astype(int)

    accuracy = accuracy\_score(y\_test, y\_pred)

*for* o, t *in* zip(y\_pred, y\_test):

        print(o, t)

    print(f"Точность модели: {accuracy:.2f}")

Случайный лес:

*import* csv

*import* numpy *as* np

*from* sklearn.ensemble *import* RandomForestClassifier

*from* sklearn.model\_selection *import* train\_test\_split

*from* sklearn.metrics *import* accuracy\_score, classification\_report

*# Загрузка данных*

X, Y = [], []

*with* open(r"MRZIS\lab3\diabetes.csv") *as* file:

    reader = csv.reader(file)

*for* i, row *in* enumerate(reader):

        X.append([float(value) *for* value *in* row[:-1]])

        Y.append(float(row[-1]))

X = np.array(X)

y = np.array(Y)

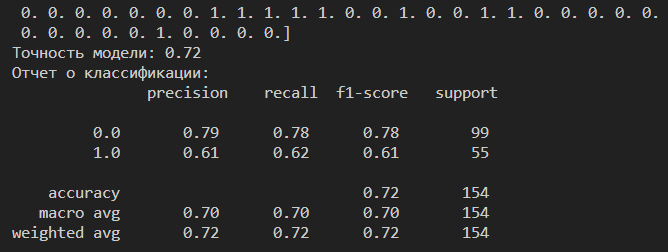
*# Разбивка данных на обучающий и тестовый наборы*

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

*# Создание и обучение модели Random Forest*

rf\_classifier = RandomForestClassifier(n\_estimators=100, random\_state=42)

rf\_classifier.fit(X\_train, y\_train)



*# Прогнозирование классов на тестовом наборе данных*

y\_pred = rf\_classifier.predict(X\_test)

print(y\_pred)

*# Оценка точности модели*

accuracy = accuracy\_score(y\_test, y\_pred)

print(f"Точность модели: {accuracy:.2f}")

*# Вывод отчета о классификации*

classification\_rep = classification\_report(y\_test, y\_pred)

print("Отчет о классификации:\n", classification\_rep)

Таким образом, получаем, что персептрон проигрывает методу случайного леса по точности.

**Вывод:** в ходе лабораторной работы я научился классифицировать данные с помощью алгоритма random forest, а также с помощью персептрона.