МИНОБОРНАУКИ РОССИИ

федеральное государственное бюджетное образовательное учреждение

высшего образования

«Московский государственный технологический университет «СТАНКИН»

(ФГБОУ ВО «МГТУ «СТАНКИН»)

Институт
информационных
технологий

Кафедра
информационных систем

Основная образовательная программа 09.03.02

«Информационные системы и технологии»

Отчет по дисциплине «Интеллектуальные и экспертные системы»

по лабораторной работе № 4

по теме: «Работа с нейросетью в среде Google Colab: анализ работы системы через обнаружение аномалий данных»

Студент
группы ИДБ-21-07 — Музафаров К. Р.
Преподаватель — Перепелкина Ю.В.

Москва 2023 г.

ОГЛАВЛЕНИЕ

ЗАДАНИЕ

Исследовать набор данных:

Cardio http://odds.cs.stonybrook.edu/cardiotocogrpahy-dataset/

Исходный набор данных Cardiotocography Data Set (https://archive.ics.uci.edu/ml/datasets/Cardiotocography) из репозитория машинного обучения UCI состоит из измерений частоты сердечных сокращений плода и сокращений матки на кардиотокограммах, классифицированных экспертами- акушерами. Исходный набор данных предназначен для классификации. В нем представлено 3 класса: «норма», «подозрение» и «патология». Для обнаружения аномалий класс «норма» принимается за норму, класс «патология» принимается за аномалии, а класс «подозрение» был отброшен.

ВВЕДЕНИЕ

Цели работы: получить практические навыки создания, обучения и применения искусственных нейронных сетей типа автокодировщик на базе платформы Google Collab. Исследовать влияние архитектуры автокодировщика и количества эпох обучения на области в пространстве признаков, распознаваемые автокодировщиком после обучения. Научиться решать актуальную задачу обнаружения аномалий в данных с помощью автокодировщика как одноклассового классификатора.

Интерактивная среда программирования Google Colaboratory (https://colab.research.google.com/notebooks/intro.ipynb) позволяет писать и выполнять код на Python прямо в браузере. Преимуществом работы в Colab, по сравнению с локальной установкой Python и Jupyter Notebook на свой компьютер, является то, что в Colab инженеры Google уже позаботились об установке всевозможных пакетов, а также о совместимости версий различных пакетов между собой. При работе в Colab пользователю предоставляется бесплатный доступ к графическим процессорам, вычисления на которых могут существенно ускорять процесс создания моделей машинного обучения. Для работы в Google Colab потребуется аккаунт Google.

Google Colab нужен всем, кто работает с Big Data: аналитикам данных (сортировать данные в файлах за долгий период, делать визуализацию или выстраивать закономерности); исследователям данных (разрабатывать и тестировать новые модели машинного обучения, составлять прогнозы); инженерам данных (разрабатывать ПО, системы для хранения больших данных). Главная особенность Google Colab — бесплатные мощные графические процессоры GPU и TPU, благодаря которым можно заниматься не только базовой аналитикой данных, но и более сложными исследованиями в области машинного обучения. GPU или TPU справляются за минуты или секунды с объемами задач, которые обычный CPU вычисляет часами. В основе — блокнот Jupyter для работы с кодом на языке Python, только с базой на

Google Диске, а не на компьютере. Здесь те же ячейки (cells), которые поддерживают текст, формулы, изображения, разметку HTML и не только. Можно заниматься программированием на языке Python и не скачивать лишние файлы, библиотеки, не перегружать машину и не заполнять место на жестком диске.

CPU — центральный процессор — мозг компьютера, который выполняет операции с файлами. Настолько универсален, что может использоваться почти для всех задач: от записи фотографий на флешку до моделирования физических процессов.

GPU — графический процессор. Обрабатывает файлы быстрее, так как задачи выполняет параллельно, а не последовательно, как CPU. Он заточен исключительно под графику, поэтому на нем удобнее работать с изображением и видео, например заниматься 3D-моделированием или монтажом.

TPU — тензорный процессор, разработка Google. Он предназначен для тренировки нейросетей. У этого процессора в разы выше производительность при больших объемах вычислительных задач. Сами процессоры дорогие, и не каждый может их себе позволить. Платформа Google Colaboratory дает возможность бесплатно и непрерывно пользоваться ими на протяжении 12 часов. Будьте внимательны: как только это время истечет, Colab сотрет все данные и файлы и придется начинать сначала. Кроме того, Google отключает файлы блокнота после примерно 30 минут бездействия, чтобы не перегружать процессоры. Система Colab так устроена специально: например, многие факторы, в том числе время простоя, максимальная активность, общие ограничения на объем памяти иногда динамически меняются. Активным участникам ненадолго могут ограничить доступ к GPU, чтобы дать возможность использовать процессор другим.

ОПИСАНИЕ РАБОТЫ

В ходе лабораторной работы была создана модель искусственного интеллекта на языке Python в среде Google Colab.

Подключаемся к Google диску.

```
[1]  from google.colab import drive
     drive.mount('/content/drive')
     import os
     os.chdir('/content/drive/MyDrive/Colab Notebooks/Lab4')
```

Импортируем модули

```
# импорт модулей
import numpy as np
from sklearn.datasets import make_blobs
!pip install tensorflow
```

```
import math
from pandas import DataFrame


import matplotlib.patches as patches
import matplotlib.pyplot as plt
from matplotlib import colors
import sklearn
from sklearn import preprocessing
from sklearn.metrics import  precision_score, recall_score, f1_score, confusion_matrix
from sklearn.metrics import zero_one_loss
from sklearn import svm
from sklearn.datasets import make_blobs
from sklearn.model_selection import train_test_split
from matplotlib import pyplot
from pandas import DataFrame
import numpy as np
import matplotlib.pylab as plt
#import aes_lib as aes
import tensorflow.keras
import numpy as np
import math
import matplotlib.pylab as plt
from sklearn import svm
```

Функция datagen

```
visual = True
verbose_show = False

# generate 2d classification dataset
#Генерирует двумерный классификационный набор данных,
#используя функцию make_blobs из sklearn.datasets. Набор данных визуализируется, если у него две характеристики.
def datagen(x_c, y_c, n_samples, n_features):

    center = [[x_c, y_c]] if n_features == 2 else None
    X, Y = make_blobs(n_samples = n_samples, centers = center, n_features = n_features, cluster_std = 0.1)
    if n_features == 2:
        plt.figure(figsize=(12, 8))
        plt.scatter(X[:,0], X[:,1], marker='o', s=7, color = 'b', label = 'Training set')
        plt.legend(loc = 'upper left', fontsize = 12)
        plt.title('Training set')
        plt.xlabel('x')
        plt.ylabel('y')
        plt.savefig('out/train_set.png')
        plt.show()
    np.savetxt('data.txt', X)

    return X
```

Функция IRE

```
#Функции расчета IRE (ire и ire_array): Определяют функции для расчета Идеальной Ошибки Восстановления (IRE) между двумя векторами или массивами.
def ire(vector1, vector2):
    x = 0
    for i in range(len(vector1)):
        x += (vector2[i] - vector1[i])**2
    # !! Round to .xx
    ire = round(math.sqrt(x), 2)
    return ire

def ire_array(array1, array2):
    ire_list = []
    for index in range(array1.shape[0]):
        ire_list.append(ire(array1[index], array2[index]))
    ire_array = np.array(ire_list)
    return ire_array
```

Класс обратного вызова ранней остановки.

```
#Класс обратного вызова ранней остановки (EarlyStoppingOnValue):
#Определяет пользовательский класс обратного вызова для ранней остановки во время обучения модели на основе заданной метрики.
class EarlyStoppingOnValue(tensorflow.keras.callbacks.Callback):

    def __init__(self, monitor='loss', baseline=None):
        super(tensorflow.keras.callbacks.Callback, self).__init__()
        self.baseline = baseline
        self.monitor = monitor

    def on_epoch_end(self, epoch, logs=None):
        current_value = self.get_monitor_value(logs)
        if current_value < self.baseline:
            self.model.stop_training = True

    def get_monitor_value(self, logs):
        monitor_value = logs.get(self.monitor)
        if monitor_value is None:
            print(
                'Early stopping conditioned on metric `%s` '
                'which is not available. Available metrics are: %s' %
                (self.monitor, ','.join(list(logs.keys()))), RuntimeWarning
            )
        return monitor_value
```

```python
#создание и обучение модели автокодировщика
def create_fit_save_ae(cl_train, ae_file, irefile, epohs, verbose_show, patience):

    size = cl_train.shape[1]
    #ans = '2'
    ans = input('Задать архитектуру автокодировщиков или использовать архитектуру по умолчанию? (1/2): ')
    if ans == '1':
        n = int(input("Задайте количество скрытых слоёв (нечетное число) : "))
        # Ниже строки читать входные данные пользователя с помощью функции map ()
        ae_arch = list(map(int, input("Задайте архитектуру скрытых слоёв автокодировщика, например, в виде 3 1 3 : ").strip().split()))[:n]
        ae = tensorflow.keras.models.Sequential()
        ae.add(tensorflow.keras.layers.Dense(size))
        ae.add(tensorflow.keras.layers.Activation('tanh'))
        for i in range(len(ae_arch)):
            ae.add(tensorflow.keras.layers.Dense(ae_arch[i]))
            ae.add(tensorflow.keras.layers.Activation('tanh'))
        ae.add(tensorflow.keras.layers.Dense(size))
        ae.add(tensorflow.keras.layers.Activation('linear'))
    else:
        ae = tensorflow.keras.models.Sequential()
        ae.add(tensorflow.keras.layers.Dense(size))
        ae.add(tensorflow.keras.layers.Activation('tanh'))
        ae.add(tensorflow.keras.layers.Dense(3))
        ae.add(tensorflow.keras.layers.Activation('tanh'))
```

```python
else:
    ae = tensorflow.keras.models.Sequential()
    ae.add(tensorflow.keras.layers.Dense(size))
    ae.add(tensorflow.keras.layers.Activation('tanh'))
    ae.add(tensorflow.keras.layers.Dense(3))
    ae.add(tensorflow.keras.layers.Activation('tanh'))
    #ae.add(tensorflow.keras.layers.Dense(4))
    #ae.add(tensorflow.keras.layers.Activation('tanh'))
    #ae.add(tensorflow.keras.layers.Dense(5))
    #ae.add(tensorflow.keras.layers.Activation('tanh'))
    ae.add(tensorflow.keras.layers.Dense(2))
    ae.add(tensorflow.keras.layers.Activation('tanh'))
    ae.add(tensorflow.keras.layers.Dense(1))
    ae.add(tensorflow.keras.layers.Activation('tanh'))
    ae.add(tensorflow.keras.layers.Dense(2))
    ae.add(tensorflow.keras.layers.Activation('tanh'))
    #ae.add(tensorflow.keras.layers.Dense(5))
    #ae.add(tensorflow.keras.layers.Activation('tanh'))
    #ae.add(tensorflow.keras.layers.Dense(4))
    #ae.add(tensorflow.keras.layers.Activation('tanh'))
    ae.add(tensorflow.keras.layers.Dense(3))
    ae.add(tensorflow.keras.layers.Activation('tanh'))
    ae.add(tensorflow.keras.layers.Dense(size))
    ae.add(tensorflow.keras.layers.Activation('linear'))
```

```
optimizer = tensorflow.keras.optimizers.Adam(learning_rate=0.001, beta_1=0.9, beta_2=0.999, amsgrad=False)
ae.compile(loss='mean_squared_error', optimizer=optimizer)
error_stop = 0.0001
epo = epohs

early_stopping_callback_on_error = EarlyStoppingOnValue(monitor='loss', baseline=error_stop)
early_stopping_callback_on_improving = tensorflow.keras.callbacks.EarlyStopping(monitor='loss',
                                                                                min_delta=0.0001, patience = patience,
                                                                                verbose=1, mode='auto',
                                                                                baseline=None,
                                                                                restore_best_weights=False)

history_callback = tensorflow.keras.callbacks.History()
verbose = 1 if verbose_show else 0
history_object = ae.fit(cl_train, cl_train,
                        batch_size=cl_train.shape[0],
                        epochs=epo,
                        callbacks=[early_stopping_callback_on_error, history_callback,
                        early_stopping_callback_on_improving],
                        verbose=verbose)
ae_trainned = ae
ae_pred = ae_trainned.predict(cl_train)
ae_trainned.save(ae_file)
```

```
#Оценивает производительность модели, сравнивая предсказанные и истинные метки.
def test(y_pred, Y_test):
    y_pred[y_pred != Y_test] = -100 # find and mark classification error
    n_errors = (y_pred == -100).astype(int).sum()
    return n_errors


#Использует обученный автокодировщик для предсказания меток и рассчитывает значения IRE для заданного набора данных.
def predict_ae(nn, x_test, threshold):
    x_test_predicted = nn.predict(x_test)
    ire = ire_array(x_test, x_test_predicted)
    # Расчет ошибки при нормализации: иначе закоментировать и раскоментировать 81 и 82
    #x_test_norm = norm_array(x_test, 0)
    #x_test_predicted_norm = nn.predict(x_test_norm)
    #x_test_predicted = norm_array(x_test_predicted_norm, 1)
    #ire = ire_array(x_test, x_test_predicted)
    predicted_labels = (ire > threshold).astype(float)
    predicted_labels = predicted_labels.reshape((predicted_labels.shape[0], 1))
    ire = np.transpose(np.array([ire]))
    return predicted_labels, ire
```

```
def load_ae(path_to_ae_file):
    return tensorflow.keras.models.load_model(path_to_ae_file)


def square_calc(numb_square, X_train, ae, IRE_th, num, visual):
    # scan
    x_min, x_max = X_train[:, 0].min() - 2, X_train[:, 0].max() + 1
    # print(x_min, x_max)
    y_min, y_max = X_train[:, 1].min() - 1, X_train[:, 1].max() + 1
    # print(y_min, y_max)
    h_x = (x_max - x_min) / 100
    h_y = (y_max - y_min) / 100
    h_y = h_x
    #print('ШАГ x:', h_x)
    #print('ШАГ y:', h_y)
    xx, yy = np.meshgrid(np.arange(x_min, x_max, h_x), np.arange(y_min, y_max, h_y))
    X_plot = np.c_[xx.ravel(), yy.ravel()]

    #  получение ответов автоэнкодера
    Z, ire = predict_ae(ae, X_plot, IRE_th)
    # print('z')
    # print(Z)
```

```
X_def = np.array([0, 0], ndmin=2)
for ind, ans in enumerate(Z):
    if ans == 0:
        # print(ans, ' kl= 1')
        # print(ind, len (svm_predicted_scan))
        X_def = np.append(X_def, [X_plot[ind]], axis=0)

# построение областей покрытия и границ классов
X_def = np.delete(X_def, 0, axis=0)
Z = Z.reshape(xx.shape)

if visual:
    plt.figure(figsize=(12, 6))
    # fig, ax = plt.subplots()
    plt.contourf(xx, yy, Z, cmap=plt.cm.tab10, alpha=0.5)
    plt.scatter(X_train[:, 0], X_train[:, 1], marker='o', s=7, color='b')
    plt.legend(['C1'])
    plt.xlabel('X')
    plt.ylabel('Y')
    plt.xlim(xx.min(), xx.max())
    plt.ylim(yy.min(), yy.max())
    plt.title('Autoencoder AE' + str(num) + '. Training set. Class boundary')
    plt.savefig('out/AE' + str(num) + '_train_def.png')
    plt.show()
```

```
    h_x = (x_max - x_min) / numb_square
    h_y = (y_max - y_min) / numb_square
    h_x = abs(h_x)
    h_y = abs(h_y)

    col_id = np.zeros(numb_square)
    col_id_ae = np.zeros(numb_square)

    for i in range(numb_square):
        for x in X_train[:, 0]:
            if x_min + i * h_x <= x < x_min + (i + 1) * h_x:
                col_id[i] = 1
        for x in X_def[:, 0]:
            if x_min + i * h_x <= x < x_min + (i + 1) * h_x:
                col_id_ae[i] = 1
```

```
amount = 0
cart = np.zeros((numb_square, numb_square))
for_rect = np.array([0, 0], ndmin=2)
for index, element in enumerate(col_id):
    if element == 1:
        for i in range(numb_square):
            for xy in X_train:
                if y_min + i * h_y <= xy[1] < y_min + (i + 1) * h_y and x_min + index * h_x <= xy[0] < x_min + (
                        index + 1) * h_x:
                    amount = amount + 1
                    cart[numb_square - i - 1, index] = 1
                    for_rect = np.append(for_rect, np.array([x_min + index * h_x, y_min + i * h_y], ndmin=2),
                                         axis=0)
                    break

for_rect = np.delete(for_rect, 0, axis=0)
# print('cart', cart)
#print('amount: ', amount)
```

```
amount_ae = 0
cart_ae = np.zeros((numb_square, numb_square))
for_rect_ae = np.array([0, 0], ndmin=2)
for index, element in enumerate(col_id_ae):
    if element == 1:
        for i in range(numb_square):
            for xy in X_def:
                if y_min + i * h_y <= xy[1] < y_min + (i + 1) * h_y and x_min + index * h_x <= xy[0] < x_min + (
                        index + 1) * h_x:
                    amount_ae = amount_ae + 1
                    cart_ae[numb_square - i - 1, index] = 1
                    for_rect_ae = np.append(for_rect_ae, np.array([x_min + index * h_x, y_min + i * h_y], ndmin=2),
                                            axis=0)
                    break

for_rect_ae = np.delete(for_rect_ae, 0, axis=0)
# print('cart_ae', cart_ae)
print('amount: ', amount)
print('amount_ae: ', amount_ae)
```

```
    if visual:
        label0_ae = 'Распознанное АЕ' + str(num) + ' множество'
        s0_ae = 0.3
        label0 = 'Обучающее множество'
        s0 = 12

        fig = plt.figure(figsize=(16, 7))
        ax_1 = fig.add_subplot(1, 2, 1)
        ax_2 = fig.add_subplot(1, 2, 2)

        ax_1.grid(which='major', axis='both', linestyle='-', color='k', linewidth=0.5)
        ax_1.set_xticks(np.arange(x_min, x_max, h_x))
        ax_1.set_yticks(np.arange(y_min, y_max, h_y))
        x_lbl = np.round(np.arange(x_min, x_max, h_x), 1).tolist()
        y_lbl = np.round(np.arange(y_min, y_max, h_y), 1).tolist()

        ax_1.set_xticklabels(x_lbl)
        ax_1.set_yticklabels(y_lbl)
```

```
for xy in for_rect:
    rect = patches.Rectangle((xy[0], xy[1]), h_x, h_y, linewidth=1, edgecolor='none', facecolor='royalblue',
                             alpha=0.3)
    ax_1.add_patch(rect)

ax_1.scatter(X_train[:, 0], X_train[:, 1], marker='o', s=s0, color='indigo', label=label0)
ax_1.legend(loc='upper left', fontsize=12)
ax_1.set_title('Площадь обучающего множества |Xt|', fontsize=14)
ax_1.set_xlabel('X')
ax_1.set_ylabel('Y')
ax_1.set_xlim(x_min, x_max)
ax_1.set_ylim(y_min, y_max)

ax_2.grid(which='major', axis='both', linestyle='-', color='k', linewidth=0.5)
ax_2.set_xticks(np.arange(x_min, x_max, h_x))
ax_2.set_yticks(np.arange(y_min, y_max, h_y))
ax_2.set_xticklabels(x_lbl)
ax_2.set_yticklabels(y_lbl)

for xy in for_rect_ae:
    rect = patches.Rectangle((xy[0], xy[1]), h_x, h_y, linewidth=1, edgecolor='none', facecolor='coral', alpha=0.4)
    ax_2.add_patch(rect)
```

```
ax_2.scatter(X_def[:, 0], X_def[:, 1], marker='o', s=s0, color='b', label=label0_ae)
ax_2.legend(loc='upper left', fontsize=12)
ax_2.set_title('Площадь деформированного множества |Xd|', fontsize=14)
ax_2.set_xlabel('X')
ax_2.set_ylabel('Y')

ax_2.set_xlim(x_min, x_max)
ax_2.set_ylim(y_min, y_max)
# plt.xlim(x_min - 4*h_x ,x_max + 4*h_x)
# plt.ylim(y_min - 4*h_y, y_max + 4*h_y)
plt.savefig('out/XtXd_' + str(num) + '.png')
plt.show()
```

```python
if visual:
    fig, axes = plt.subplots(nrows=1, ncols=3, figsize=(22, 8))
    # n = 1

    for ax in axes.flat:
        # ax.set(title='axes_' + str(n), xticks=[], yticks=[])
        # n += 1
        # ax.scatter(X_ov[:, 0], X_ov[:, 1], marker='o', s=s0, color='b', label=label0)
        ax.grid(which='major', axis='both', linestyle='-', color='k', linewidth=0.5)
        ax.set_xticks(np.arange(x_min, x_max, h_x))
        ax.set_yticks(np.arange(y_min, y_max, h_y))  # +0.7
        x_lbl = np.round(np.arange(x_min, x_max, h_x), 1).tolist()
        y_lbl = np.round(np.arange(y_min, y_max, h_y), 1).tolist()
        ax.set_xticklabels(x_lbl)
        ax.set_yticklabels(y_lbl)
        ax.set_xlim(x_min, x_max)
        ax.set_ylim(y_min, y_max)

        ax.set_xlabel('X')
        ax.set_ylabel('Y')
```

```python
    for xy in for_rect_ae:
        rect = patches.Rectangle((xy[0], xy[1]), h_x, h_y, linewidth=1, edgecolor='k', facecolor='coral')
        ax.add_patch(rect)
    rect = patches.Rectangle((xy[0], xy[1]), h_x, h_y, linewidth=1, edgecolor='none', facecolor='coral',
                             label='Площадь множества |Xd|')
    ax.add_patch(rect)
    for xy in for_rect:
        rect = patches.Rectangle((xy[0], xy[1]), h_x, h_y, linewidth=1, edgecolor='k', facecolor='royalblue')
        ax.add_patch(rect)
    rect = patches.Rectangle((xy[0], xy[1]), h_x, h_y, linewidth=1, edgecolor='none', facecolor='royalblue',
                             label='Площадь множества  |Xt|')
    ax.add_patch(rect)

    # ax.scatter(for_rect[0, 0] + 0.1, for_rect[0, 1] + 0.1, marker='o', s=s0, color='cornflowerblue', label='Объем об
    # ax.scatter(for_rect_ae[0,0]+ 0.1, for_rect_ae[0,1]+ 0.1, marker='o', s=s0, color='darkorange', label='Объем дефо
    ax.legend(loc='upper left', fontsize=16)
    ####ax.set_ylim(-2.5, 2.9)
    # ax.set_ylim(-1.7, 2.4)#ae2
```

```python
nn = 0
for xy_ae in for_rect_ae:
    for xy in for_rect:
        if xy_ae[0] == xy[0] and xy_ae[1] == xy[1]:
            nn = nn + 1
            if nn == 1:
                rect1 = patches.Rectangle((xy_ae[0], xy_ae[1]), h_x, h_y, linewidth=1, edgecolor='k',
                                          facecolor='none', hatch='/', label='Площадь на пересечении |Xt| и |Xd|')
                axes[2].add_patch(rect1)
            else:
                rect1 = patches.Rectangle((xy_ae[0], xy_ae[1]), h_x, h_y, linewidth=1, edgecolor='k',
                                          facecolor='none', hatch='/')
                axes[2].add_patch(rect1)
```

```
axes[2].legend(loc='upper left', fontsize=16)
# print('true')
flag = 1
n = 0
for xy_ae in for_rect_ae:
    flag = 1
    for xy in for_rect:
        if xy_ae[0] == xy[0] and xy_ae[1] == xy[1]:
            # print(xy_ae[0], '!=', xy[0],' and ', xy_ae[1], '!=', xy[1])
            flag = 0

    if flag == 1:
        n = n + 1
        if n == 1:
            rect2 = patches.Rectangle((xy_ae[0], xy_ae[1]), h_x, h_y, linewidth=1, edgecolor='k', facecolor='none',
                                      hatch='/', label='Площадь |Xd| за исключением |Xt| (|Xd\Xt|)')
            axes[0].add_patch(rect2)
        else:
            rect2 = patches.Rectangle((xy_ae[0], xy_ae[1]), h_x, h_y, linewidth=1, edgecolor='k', facecolor='none',
                                      hatch='/')
            axes[0].add_patch(rect2)
```

```
    rect1 = patches.Rectangle((for_rect_ae[0, 0], for_rect_ae[0, 1]), h_x, h_y, linewidth=1, edgecolor='k',
                              facecolor='none', label='Площадь |Xt| за исключением |Xd| (|Xt\Xd|)')
    axes[1].add_patch(rect1)
    # now#rect2 = patches.Rectangle((xy_ae[0], xy_ae[1]), h_x, h_y, linewidth=1, edgecolor='k', facecolor='non
    # now#axes[0].add_patch(rect2)
    axes[0].legend(loc='upper left', fontsize=16)
    axes[1].legend(loc='upper left', fontsize=16)
    axes[0].set_title('Excess. AE'  + str(num), fontsize=20)
    axes[1].set_title('Deficit. AE'  + str(num), fontsize=20)
    axes[2].set_title('Coating. AE'  + str(num), fontsize=20)
    plt.savefig('out/XtXd_' + str(num) + '_metrics.png')
    plt.show()

square_ov = amount * h_x * h_y
square_ae = amount_ae * h_x * h_y

print()
print('Оценка качества AE' + str(num))
extra_pre_ae = square_ov / square_ae
# print('square_ov:',  square_ov)
# print('square_ae:', square_ae)
```

```
Ex = cart_ae - cart
Excess = np.sum(Ex == 1) / amount
print('IDEAL = 0. Excess: ', Excess)
Def = cart - cart_ae
Deficit = np.sum(Def == 1) / amount
print('IDEAL = 0. Deficit: ', Deficit)
cart[cart > 0] = 5
Coa = cart - cart_ae
Coating = np.sum(Coa == 4) / amount
print('IDEAL = 1. Coating: ', Coating)
summa = Deficit + Coating
print('summa: ', summa)
print('IDEAL = 1. Extrapolation precision (Approx): ', extra_pre_ae)
print()
print()

with open('out/result.txt', 'w') as file:
    file.write(
        '------------Оценка качества AE' + str(num) + ' С ПОМОЩЬЮ НОВЫХ МЕТРИК------------' + '\n' + \
        'Approx = ' + str(extra_pre_ae) + '\n' + \
        'Excess = ' + str(Excess) + '\n' + \
        'Deficit = ' + str(Deficit) + '\n' + \
        'Coating = ' + str(Coating) + '\n')
```

```
    return xx, yy, Z

#####2D
def plot_xdef(X_train, xx, yy, Z):

    plt.contourf(xx, yy, Z, cmap=plt.cm.tab10, alpha=0.5)
    plt.scatter(X_train[:, 0], X_train[:, 1], marker='o', s=7, color='b')
    plt.legend(['C1'])
    plt.xlabel('X')
    plt.ylabel('Y')
    plt.xlim(xx.min(), xx.max())
    plt.ylim(yy.min(), yy.max())

def plot2in1(X_train, xx, yy, Z1, Z2):

    plt.subplot(1, 2, 1)
    plot_xdef(X_train, xx, yy, Z1)
    plt.title('Autoencoder AE1')#. Training set. Class boundary')

    plt.subplot(1, 2, 2)
    plot_xdef(X_train, xx, yy, Z2)
    plt.title('Autoencoder AE2')#. Training set. Class boundary')
    plt.savefig('out/AE1_AE2_train_def.png')
    plt.show()
```

```
def anomaly_detection_ae(predicted_labels, ire, ire_th):
    ire = np.round(ire,2)
    ire_th = np.round(ire_th, 2)
    if predicted_labels.sum() == 0:
        print("Аномалий не обнаружено")
    else:
        print()
        print('%-10s%-10s%-10s%-10s' % ('i', 'Labels', 'IRE', 'IREth'))
        for i, pred in enumerate(predicted_labels):
            print('%-10s%-10s%-10s%-10s' % (i, pred, ire[i], ire_th))
        print('Обнаружено ', predicted_labels.sum(), ' аномалий')
```

```
def plot2in1_anomaly(X_train, xx, yy, Z1, Z2, anomalies):

    plt.subplot(1, 2, 1)
    plot_xdef(X_train, xx, yy, Z1)
    plt.scatter(anomalies[:, 0], anomalies[:, 1], marker='o', s=12, color='r')
    plt.title('Autoencoder AE1')#. Training set. Class boundary')

    plt.subplot(1, 2, 2)
    plot_xdef(X_train, xx, yy, Z2)
    plt.scatter(anomalies[:, 0], anomalies[:, 1], marker='o', s=12, color='r')
    plt.title('Autoencoder AE2')#. Training set. Class boundary')
    plt.savefig('out/AE1_AE2_train_def_anomalies.png')
    plt.show()
```

```
def ire_plot(title, IRE_test, IREth, ae_name):

    x = range(1, len(IRE_test) + 1)
    IREth_array = [IREth for x in x]
    plt.figure(figsize = (16, 8))
    plt.title('IRE for ' + title + ' set. ' + ae_name, fontsize = 24)
    plt.plot(x, IRE_test, linestyle = '-', color = 'r', lw = 2, label = 'IRE')
    plt.plot(x, IREth_array, linestyle = '-', color = 'k', lw = 2, label = 'IREth')
    #plt.xlim(0, len(x))
    ymax = 1.5 * max(np.amax(IRE_test), IREth)
    plt.ylim(0, ymax)
    plt.xlabel('Vector number', fontsize = 20)
    plt.ylabel('IRE', fontsize = 20)
    plt.grid()
    plt.legend(loc = 'upper left', fontsize = 16)
    plt.gcf().savefig('out/IRE_' + title + ae_name + '.png')
    plt.show()

    return
```

```
train = np.loadtxt('cardio_train.txt', dtype=np.float64)
test = np.loadtxt('cardio_test.txt', dtype=np.float64)
# вывод данных и размерности
print('Исходные данные:')
print(train)
print('Размерность данных:')
print(train.shape)
```

```
Исходные данные:
[[ 0.00491231  0.69319077 -0.20364049 ...  0.23149795 -0.28978574
  -0.49329397]
 [ 0.11072935 -0.07990259 -0.20364049 ...  0.09356344 -0.25638541
  -0.49329397]
 [ 0.21654639 -0.27244466 -0.20364049 ...  0.02459619 -0.25638541
   1.1400175 ]
 ...
 [ 0.85144861 -0.91998844 -0.20364049 ...  0.57633422 -0.65718941
   1.1400175 ]
 [ 0.85144861 -0.91998844 -0.20364049 ...  0.57633422 -0.62378908
  -0.49329397]
 [ 1.0630827  -0.51148142 -0.16958144 ...  0.57633422 -0.65718941
  -0.49329397]]
Размерность данных:
(1654, 21)
```

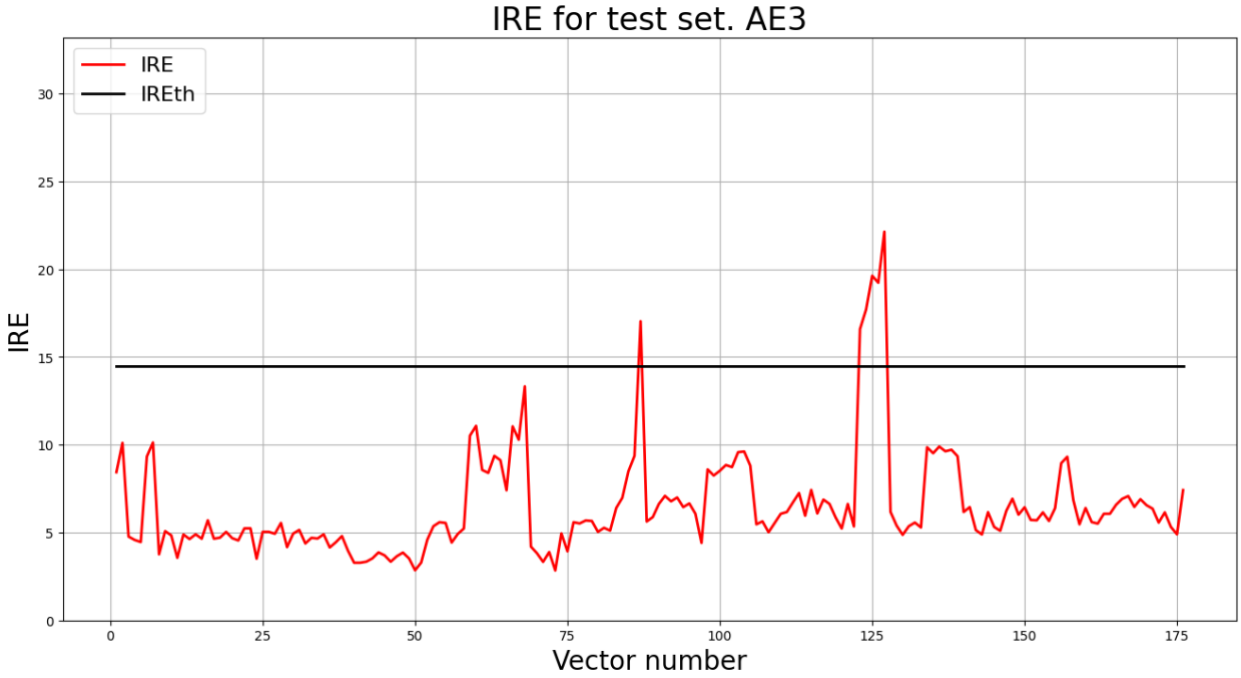
```
#Создает, компилирует и обучает модель автокодировщика (ae3_trained).
#train представляет собой обучающий набор данных.
#Модель сохраняется в файл 'out/AE3.h5'.
#Значения Идеальной Ошибки Восстановления (IRE) между входными и выходными данными сохраняются в IRE3.
#Порог IRE сохраняется в IREth3.
#В процессе обучения используется ранняя остановка с терпением равным 100 и максимальным числом эпох равным 1000.

ae3_trained, IRE3, IREth3 = create_fit_save_ae(train,'out/AE3.h5','out/AE3_ire_th.txt', 1000, True, 100)
ire_plot('test', IRE3, IREth3, 'AE3')
```

```
# Прогнозирует метки аномалий (predicted_labels3) и значения Идеальной Ошибки
# Восстановления (IRE) (ire3) для тестового набора данных с использованием обученной модели ae3_trained и порога IRE (IREth3).
predicted_labels3, ire3 = predict_ae(ae3_trained, test, IREth3)
# тестирование AE3
anomaly_detection_ae(predicted_labels3, ire3, IREth3)
```

```
6/6 [==============================] - 0s 5ms/step

i        Labels    IRE       IREth
0        [0.]      [8.44]    14.5
1        [0.]      [10.11]   14.5
2        [0.]      [4.77]    14.5
3        [0.]      [4.58]    14.5
4        [0.]      [4.46]    14.5
5        [0.]      [9.33]    14.5
6        [0.]      [10.13]   14.5
7        [0.]      [3.76]    14.5
8        [0.]      [5.08]    14.5
9        [0.]      [4.84]    14.5
10       [0.]      [3.56]    14.5
11       [0.]      [4.89]    14.5
12       [0.]      [4.63]    14.5
13       [0.]      [4.9]     14.5
14       [0.]      [4.65]    14.5
15       [0.]      [5.7]     14.5
16       [0.]      [4.65]    14.5
```


IRE for test set. AE3
IRE
IREth
IRE
Vector number
0
5
10
15
20
25
30
0
25
50
75
100
125
150
175

ВЫВОДЫ

В ходе лабораторной работы была создана и обучена на тестовом наборе данных модель искусственного интеллекта для распознавания цифр. Также была проверена работа на собственном наборе данных.

СПИСОК ЛИТЕРАТУРЫ

1. Лысачев М. Н. Искусственный интеллект. Анализ, тренды, мировой опыт / М.Н.Лысачев, А. Н. Прохоров; научный редактор Д. А. Ларионов. – Корпоративное издание. – Москва; Белгород: КОНСТАНТА-принт, 2023. – 460 с. : ил., табл. ISBN 978-5-6048180-7-7, Электронное издание (ссылка на Яндекс-диск https://disk.yandex.ru/i/d-ky8jRcWqHR6g)

2. Рындина С. В. Базовые возможности языка Python для анализа данных: учеб.-метод. пособие / С. В. Рындина. – Пенза : Изд-во ПГУ, 2022. – 72 с. (ссылка на Яндекс-диск https://disk.yandex.ru/i/kCmRFIxp3oXwCQ)