1. Министерство образования и науки Российской Федерации
2. Санкт-Петербургский Политехнический Университет Петра Великого
3. —
4. Институт прикладной математики и механики
5. **Кафедра «Информационная безопасность компьютерных систем»**

**ЛАБОРАТОРНАЯ РАБОТА № 3**

1. «**Использование Python-библиотек для работы**
2. **с ИИ**»
3. по дисциплине «Цифровая культура»
4. Выполнил
5. студент гр. 5131001/20003 Черникова В.М.

<*подпись*>

1. Преподаватель
2. Писков А. А.

<*подпись*>

1. Санкт-Петербург
2. 2024
3. **Цель работы**

Получение навыков создания Telegram-ботов на языке программирования Python.

1. **Выделить в текущей реализации Telegram-бота недостатки хранения пользовательских данных и усовершенствовать метод хранения.**

Данные хранились в открытом виде в .сsv файле, поэтому было решено добавить одностороннее шифрование Id пользователя из телеграмма и пароля.

Была подкобчена библиотека hashlib, и использованы алгоритмы SHA-256 и MD5. БД стала выглядеть так:



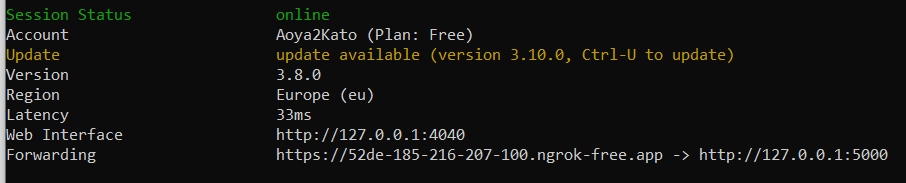
1. POST – запрос для связывания веб-приложения и телеграмма на flask.

@app.route( '/', methods=['POST'])  
def webhook():  
 if flask.request.headers.get('content-type') == 'application/json':  
 json\_string = flask.request.get\_data().decode('utf-8')  
 update = telebot.types.Update.de\_json(json\_string)  
 bot.process\_new\_updates([update])  
 else:  
 flask.abort(403)  
 return "!", 200

1. **Подключение ngrok**

Ngrok выступает сервисом, который предоставляет сервер в интернете и проводит туннель на локальный хост

**ngrok http** [**http://127.0.0.1:5000**](http://127.0.0.1:5000)



1. Зарегистрировать созданный webhook в Telegram.

В браузере был отправлен запрос

[https://api.telegram.org/bot{token\_bot}/setWebhook?url={URL}](https://api.telegram.org/bot%7btoken_bot%7d/setWebhook?url=%7bURL%7d),

где token – это токен бота

URL – веб-приложение с которым надо связать бота

1. **Код бота с пояснениями**

import os

os.environ['TF\_CPP\_MIN\_LOG\_LEVEL'] = '2'

import telebot

import pandas as pd

import csv

from PIL import Image

import numpy as np

import matplotlib.pyplot as plt

# ------------------------load\_nn----------------

from keras.api.preprocessing import image

from keras.\_tf\_keras.keras.models import model\_from\_json

print("Загружаю сеть из файлов")

json\_file = open("mnist\_model.json", "r")

loaded\_model\_json = json\_file.read()

json\_file.close()

model = model\_from\_json(loaded\_model\_json)

model.load\_weights("mnist\_model.weights.h5")

print("Загрузка сети завершена")

# ------------------------end\_load\_nn----------------

bot = telebot.TeleBot(my\_token)

APP\_HOST="127.0.0.1"

APP\_PORT=5000

bot = telebot.TeleBot(API\_TOKEN)

app = flask.Flask(\_\_name\_\_)

@app.route('/')

def hello\_world():

return 'Hello from Flask!'

@app.route( '/', methods=['POST'])

def webhook():

if flask.request.headers.get('content-type') == 'application/json':

json\_string = flask.request.get\_data().decode('utf-8')

update = telebot.types.Update.de\_json(json\_string)

bot.process\_new\_updates([update])

else:

flask.abort(403)

return "!", 200

#-----------------------------------start\_help---------------------

@bot.message\_handler(commands=['start'])

def start(message):

bot.send\_message(message.chat.id, "Привет, вот список команд:")

bot.send\_message(message.chat.id,

"/register - Зарегистрироваться. Только зарегистрированные пользователи могут использовать бота.\n"

"/login - Войти в аккаунт.\n"

"/predict - Отправьте картинку, а бот скажет, это человек или белка.\n"

"/logout - Выйти из аккаунта.\n"

"/help - Список команд.")

@bot.message\_handler(commands=['help'])

def help(message):

bot.send\_message(message.chat.id, "Вот список команд:")

bot.send\_message(message.chat.id,

"/register - Зарегистрироваться. Только зарегистрированные пользователи могут использовать бота.\n"

"/login - Войти в аккаунт.\n"

"/predict - Отправьте картинку, а бот скажет, это человек или белка.\n"

"/logout - Выйти из аккаунта.\n"

"/help - Список команд.")

#--------------------------------end\_start\_help---------------------

@bot.message\_handler(commands=['register'])

def register(message):

mesg = bot.send\_message(message.chat.id, 'Пожалуйста, введите пароль')

bot.register\_next\_step\_handler(mesg, test)# ловит следующее сообщение пользователя

def test(message):

if ',' in message.text or (';' in message.text or '/' in message.text):

bot.send\_message(message.chat.id, "Вы использовали , или ; или /\n"

"Это нельзя использовать в пароле")

return

try:

f\_csv = pd.read\_csv("ttt.csv", delimiter=",")

# print(f\_csv)

fields = []

if (message.from\_user.id not in list(f\_csv["Id"])):

fields = [message.from\_user.username, message.from\_user.id, message.text, 0, 0]

with open("ttt.csv", 'a+') as csvfile:

csvwriter = csv.writer(csvfile)

csvwriter.writerow(fields)

bot.send\_message(message.chat.id, "Вы успешно зарегистрированны!")

else:

bot.send\_message(message.chat.id, "Вы уже зарегистрированны")

except:

bot.send\_message(message.chat.id, "Что-то пошло не так, попробуйте позже.")

@bot.message\_handler(commands=['login'])

def login(message):

mesg = bot.send\_message(message.chat.id, 'Пожалуйста, введите пароль')

bot.register\_next\_step\_handler(mesg, test1)

def test1(message):

try:

f\_csv = pd.read\_csv("ttt.csv", delimiter=",")

# print(f\_csv)

if (message.from\_user.id in list(f\_csv["Id"])):

for row in range(len(f\_csv["Name"])):

if (f\_csv["Id"][row] == message.from\_user.id and f\_csv["password"][row] == message.text):

f\_csv.loc[row, "Flag"] = 1

f\_csv.to\_csv("ttt.csv", index=False)

bot.send\_message(message.chat.id, "Вы успешно вошли в систему!")

return

else:

bot.send\_message(message.chat.id, "Вы не зарегистрированны")

return

bot.send\_message(message.chat.id, "Неверный пароль!")

return

except:

bot.send\_message(message.chat.id, "Что-то пошло не так, попробуйте позже.")

@bot.message\_handler(commands=['logout'])

def logout(message):

try:

f\_csv = pd.read\_csv("ttt.csv", delimiter=",")

# print(f\_csv)

if (message.from\_user.id in list(f\_csv["Id"])):

for row in range(len(f\_csv["Name"])):

if (f\_csv["Id"][row] == message.from\_user.id and f\_csv.loc[row, "Flag"] == 1):

f\_csv.loc[row, "Flag"] = 0

f\_csv.to\_csv("ttt.csv", index=False)

bot.send\_message(message.chat.id, "Вы успешно вышли из системы!")

return

elif (f\_csv["Name"][row] == message.from\_user.username and f\_csv.loc[row, "Flag"] == 0):

bot.send\_message(message.chat.id, "Вы не входили в систему")

return

else:

bot.send\_message(message.chat.id, "Вы не зарегистрированны")

except:

bot.send\_message(message.chat.id, "Что-то пошло не так, попробуйте позже.")

@bot.message\_handler(commands=['predict'])

def predict(message):

try:

f\_csv = pd.read\_csv("ttt.csv", delimiter=",")

# print(f\_csv)

if (message.from\_user.id in list(f\_csv["Id"])):#проверки, что пользователь есть в системе и он в сети

for row in range(len(f\_csv["Name"])):

if (f\_csv["Id"][row] == message.from\_user.id and f\_csv.loc[row, "Flag"] == 1):

mesg = bot.send\_message(message.chat.id, 'Пожалуйста, отправьте картинку')#ловит картинку

return

elif (f\_csv["Name"][row] == message.from\_user.username and f\_csv.loc[row, "Flag"] == 0):

bot.send\_message(message.chat.id, "Вы не вошли в систему")

return

else:

bot.send\_message(message.chat.id, "Вы не зарегистрированны")

except:

bot.send\_message(message.chat.id, "Что-то пошло не так, попробуйте позже.")

def test2(message):

if message.photo:

try:

#сохранение изображения

file\_info = bot.get\_file(message.photo[len(message.photo) - 1].file\_id)

downloaded\_file = bot.download\_file(file\_info.file\_path)

src = 'C:/Users/Vasilisa/Music/NN3/tmp/' + message.photo[1].file\_id

with open(src, 'wb') as new\_file:

new\_file.write(downloaded\_file)

img = Image.open(src)

path = 'C:/Users/Vasilisa/Music/NN3/tmp/' + message.photo[1].file\_id + '.jpg'

img.save(path)

bot.reply\_to(message, "Фото добавлено")

img = image.load\_img(path, target\_size=(100, 100))

x = image.img\_to\_array(img)

x = np.expand\_dims(x, axis=0)

#предсказание

classes = model.predict(x, batch\_size=10)

if classes[0] < 0.5:

print(" is a squirrel")

bot.send\_message(message.chat.id, "Я думаю, это белка")

else:

print(" is a man")

bot.send\_message(message.chat.id, "Я думаю, это человек")

#удаление изображения из памяти

os.remove(path)

os.remove(src)

except:

bot.send\_message(message.chat.id, "Что-то пошло не так, попробуйте позже.")

#автоответчик

@bot.message\_handler(content\_types=['text', 'audio', 'document', 'photo', 'sticker', 'video', 'video\_note',

'voice', 'location', 'contact', 'new\_chat\_members', 'left\_chat\_member',

'new\_chat\_title', 'new\_chat\_photo', 'delete\_chat\_photo', 'group\_chat\_created',

'supergroup\_chat\_created', 'channel\_chat\_created', 'migrate\_to\_chat\_id',

'migrate\_from\_chat\_id', 'pinned\_message'])

def answer(message):

bot.send\_message(message.chat.id,

"Это прсто сообщение, я его не понимаю. Мне нужна команда. Введите /help, чтобы увидеть список команд")

# if message.text == "Кодовое слово":

print(message.text, message.from\_user.username, message.from\_user.id)

bot.polling(none\_stop=True, interval=0)

1. **Код создания нейронной сети с пояснениями**

import os  
os.environ['TF\_ENABLE\_ONEDNN\_OPTS'] = '0'  
  
import numpy as np  
import pickle  
import tensorflow as tf  
import matplotlib.pyplot as plt  
import matplotlib.image as mpimg  
from PIL import Image  
from scipy import interpolate  
from itertools import cycle  
from sklearn import svm, datasets  
from sklearn.metrics import roc\_auc\_score  
from sklearn.metrics import roc\_curve, auc  
from sklearn.preprocessing import label\_binarize  
from keras.api.preprocessing import image  
from sklearn.multiclass import OneVsRestClassifier  
from sklearn.model\_selection import train\_test\_split  
from keras.src.legacy.preprocessing.image import ImageDataGenerator

#создание датасетов  
train\_dandelion\_dir = os.path.join('data/data/train/squirrel')  
train\_grass\_dir = os.path.join('data/data/train/man')  
valid\_dandelion\_dir = os.path.join('data/data/val/squirrel')  
valid\_grass\_dir = os.path.join('data/data/val/man')  
train\_dandelion\_names = os.listdir(train\_dandelion\_dir)  
train\_grass\_names = os.listdir(train\_grass\_dir)  
validation\_grass\_names = os.listdir(valid\_grass\_dir)  
  
print('total training squirrel images:', len(os.listdir(train\_dandelion\_dir)))  
print('total training man images:', len(os.listdir(train\_grass\_dir)))  
print('total validation squirrel images:', len(os.listdir(valid\_dandelion\_dir)))  
print('total validation man images:', len(os.listdir(valid\_grass\_dir)))  
  
nrows = 4  
ncols = 4  
pic\_index = 0  
fig = plt.gcf()  
fig.set\_size\_inches(ncols \* 4, nrows \* 4)  
pic\_index += 8  
next\_dandelion\_pic = [os.path.join(train\_dandelion\_dir, fname)  
 for fname in train\_dandelion\_names[pic\_index - 8:pic\_index]]  
next\_grass\_pic = [os.path.join(train\_grass\_dir, fname)  
 for fname in train\_grass\_names[pic\_index - 8:pic\_index]]  
for i, img\_path in enumerate(next\_dandelion\_pic + next\_grass\_pic):  
 sp = plt.subplot(nrows, ncols, i + 1)  
 sp.axis('Off')  
  
 img = mpimg.imread(img\_path)  
   
  
train\_datagen = ImageDataGenerator(rescale=1 / 1)  
validation\_datagen = ImageDataGenerator(rescale=1 / 1)  
  
train\_generator = train\_datagen.flow\_from\_directory(  
 'data/data/train/',  
 classes=['squirrel', 'man'],  
 target\_size=(100, 100),  
 batch\_size=4,  
 class\_mode='binary')  
  
validation\_generator = validation\_datagen.flow\_from\_directory(  
 'data/data/val/',  
 classes=['squirrel', 'man'],  
 target\_size=(100, 100),  
 batch\_size=4,  
 class\_mode='binary',  
 shuffle=False)

#конец создания датасетов  
'''  
model = tf.keras.Sequential([  
 #tf.keras.layers.Flatten(input\_shape=(100, 100, 3)),  
 tf.keras.layers.Conv2D(32, (3, 3), padding='same', activation='relu', input\_shape=(100, 100, 3)),  
 tf.keras.layers.MaxPooling2D((2, 2), strides=2),  
 tf.keras.layers.Dense(128, activation='relu'),  
 tf.keras.layers.Conv2D(64, (3, 3), padding='same', activation='relu'),  
 tf.keras.layers.MaxPooling2D((2, 2), strides=2),  
 tf.keras.layers.Flatten(),  
 tf.keras.layers.Dense(1, activation='relu'),  
])  
  
'''

#архитекутра модели  
model = tf.keras.models.Sequential([tf.keras.layers.Flatten(input\_shape=(100, 100, 3)),  
 tf.keras.layers.Dense(128, activation=tf.nn.relu),  
 tf.keras.layers.Dense(1, activation=tf.nn.sigmoid)])  
  
model.summary()

#компиляция модели  
model.compile(optimizer="adam",  
 loss='binary\_crossentropy',  
 metrics=['accuracy'])  
#обучение модели  
history = model.fit(train\_generator,  
 steps\_per\_epoch=8,  
 batch\_size=24,  
 epochs=30,  
 verbose=1,  
 validation\_data=validation\_generator,  
 validation\_steps=8)

# создание датасета для тестирования  
test\_datagen = ImageDataGenerator(rescale=1 / 1)  
test\_generator = test\_datagen.flow\_from\_directory(  
 'data/data/test/',  
 classes=['squirrel', 'man'],  
 target\_size=(100, 100),  
 batch\_size=5,  
 class\_mode='binary',  
 shuffle=False)

#тестирование  
scores = model.evaluate(test\_generator, verbose=0)  
print("Точность работы загруженной сети на тестовых данных: %.2f%%" % (scores[1] \* 100))  
  
print("Сохраняем сеть")  
# Сохраняем сеть для последующего использования  
# Генерируем описание модели в формате json  
model\_json = model.to\_json()  
json\_file = open("mnist\_model.json", "w")  
# Записываем архитектуру сети в файл  
json\_file.write(model\_json)  
json\_file.close()  
# Записываем данные о весах в файл  
model.save\_weights("mnist\_model.weights.h5")  
print("Сохранение сети завершено")

#проверка  
uploaded = ['test.jpg']  
  
for fn in uploaded:  
 path = fn  
 img = image.load\_img(path, target\_size=(100, 100))  
 x = image.img\_to\_array(img)  
 plt.imshow(x / 255.)  
 x = np.expand\_dims(x, axis=0)  
 images = np.vstack([x])  
 classes = model.predict(images, batch\_size=10)  
 print(classes[0])  
 if classes[0] < 0.5:  
 print(fn + " is a squirrel")  
 else:  
 print(fn + " is a man")

1. **Ответы на контрольные вопросы**
2. ***Какие классы задач могут быть решены с помощью методов искусственного интеллекта?***

Прогнозирование, написания текстов и других мультимедиа, автоответчик и поисковая строка, распознавание объектов и многое другое.

1. ***Чем отличается обучающий набор данных от тестового?***

Обучающая выборка – выборка, на которой модель обучается. Набор данных, который мы передаем нашей модели, чтобы изучить потенциальные закономерности и отношения.

Тестовая выборка – выборка для тестирования полученной модели, не участвуют в процессе обучения. Полученный, на тестовой выборке, результат мы сравниваем с фактическими значениями и исходя из этого, оцениваем точность нашей модели.

1. ***Что такое признак в контексте методов искусственного интеллекта? Что такое метка в контексте методов искусственного интеллекта? В чем их разница?***

Разница между признаками и метками заключается в том, что признаки - это характеристики объектов, которые используются для обучения модели, в то время как метки - это целевые переменные или классы, которые модель пытается предсказать.

1. ***Чем методы глубокого обучения отличаются от других методов искусственного интеллекта?***

Методы глубокого обучения отличаются от других методов искусственного интеллекта своей способностью автоматически извлекать признаки из данных с использованием глубоких нейронных сетей и обработки больших объемов данных для решения сложных задач.

***5) Из чего состоит слой в нейронной сети? Какие слои бывают? Что такое нейрон?***

Слой состоит из нейронов, слои бывают скрытые, входные, выходные. Нейрон – математическая функция, у которой меняется вес значимости в процессе валидации.

1. ***Что такое аутентификация?***

Аутентификация — это сам процесс проверки подлинности пользователя, чтобы убедиться, что он является тем, за кого себя выдает. Для аутентификации пользователи могут вводить логин и пароль, биометрические данные или другие методы идентификации.

1. ***Что такое авторизация?***

Авторизация — это процесс проверки прав доступа пользователя к определенным ресурсам или функциям. Он проводится после успешной аутентификации.

1. ***Чем аутентификация отличается от авторизации?***

Аутентификация нужна, чтобы проверить право доступа к данным, авторизация — это когда вы получаете доступ.

1. ***Для чего нужен токен Telegram-бота?***

Токен нужен, чтобы подключить бота к платформе управления рассылками.

С помощью токена можно контролировать чат-бот, поэтому его важно хранить в безопасном месте. Если вы потеряете его или случайно удалите диалог, просто отправьте в BotFather команду /token — и он вам его переотправит.