МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет вычислительной техники

Кафедра «Вычислительная техника»

Направление подготовки: 09.03.01 «Информатика и вычислительная техника»

Профиль подготовки: «Вычислительные машины, комплексы, системы и сети»

БАКАЛАВРСКАЯ РАБОТА

на тему

СРЕДСТВА ДЛЯ ПРОВЕДЕНИЯ ПСИХОЛОГИЧЕСКОГО ТЕСТИРОВАНИЯ С ИСПОЛЬЗОВАНИЕМ НЕЙРОННОЙ СЕТИ

Студент _____ 29.05.23 Григорьева Д. Д.
(подпись, дата) (ФИО полностью)

Руководитель _____ 05.06.23 Зинкин С. А.
(подпись, дата) (фамилия, инициалы)

Нормоконтролёр _____ 05.06.23 А. С. Бычков
(подпись, дата) (фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 2.06.2023 г. № 14)

Заведующий кафедрой _____ М.А. Митрохин
(подпись) (фамилия, инициалы)

Работа защищена с отметкой отлично (протокол заседания ГЭК от 06.06.2023 № 1)

Секретарь ГЭК _____ Е.И. Гурин
(подпись) (фамилия, инициалы)

Пенза 2023

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
М.А. Митрохин
«26» 04 2023 г.

ЗАДАНИЕ
НА БАКАЛАВРСКУЮ РАБОТУ

1. Студентка Григорьева Дарья Дмитриевна гр. 19ВВВ1 факультета вычислительной техники

2.Тема работы «Средства для проведения психологического тестирования с использованием нейронной сети»

Тема утверждена приказом ПГУ № 458/0 от "26" апреля 2023г.

3. Руководитель бакалаврской работы д. т. н., профессор Зинкин С. А.

4. Задание на работу (назначение разработки, исходные данные и т.п.)

Разработать приложение «Средства для проведения психологического тестирования с использованием нейронной сети», проводящее тестирование методом Роршаха, классифицирующее введённые пользователем ответы с помощью нейронной сети. Приложение должно иметь понятный пользовательский интерфейс, отображать результаты и сохранять в базу данных формата *CSV*.

5. Перечень подлежащих разработке вопросов

Требуется произвести анализ предметных областей: изучить аналоги существующих приложений, осуществляющих психологическое тестирование, изучить актуальные способы реализации нейронных сетей, включая методы машинного и глубокого обучения. Необходимо изучить функционал фреймворков *PySide6, Keras, Skit-learn, Tensorflow*. Так же необходимо изучить библиотеку для работы с данными формата *CSV - Pandas*.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1.	Изучение предметной области	50	26.04.2023-30.04.2023	
2.	Анализ требований	10	01.05.2023-02.05.2023	
3.	Выбор технологий, применяемых при проектировании в разработке	50	03.05.2023-07.05.2023	
4.	Описание алгоритмов	40	08.05.2023-11.05.2023	
5.	Разработка приложения «Средства для проведения психологического тестирования с использованием нейронной сети»	150	12.05.2023-29.05.2023	
6.	Оформление пояснительной записки	24	20.05.2023-31.05.2023	

Дата выдачи задания "26"апреля 2023г.

Руководитель бакалаврской работы _____ Зинкин С. А.
(подпись, дата)

Задание к исполнению принял студент _____ Григорьева Д. Д.
(подпись, дата)

Работу к защите допустить
Декан факультета _____ 06.06.23 Фионова Л. Р.
(подпись, дата)

Реферат

Бакалаврская работа содержит 95 лст., 23 рис., 4 табл., 3 прил. и 13 источ.

АНАЛИЗ ДАННЫХ, БИНАРНАЯ КЛАССИФИКАЦИЯ ТЕКСТОВ, НЕЙРОННЫЕ СЕТИ, МАШИННОЕ ОБУЧЕНИЕ, ГЛУБОКОЕ ОБУЧЕНИЕ, ПСИХОЛОГИЧЕСКОЕ ТЕСТИРОВАНИЕ.

Цель работы - приложение, осуществляющее постановку предварительного диагноза по тесту Роршаха, интерпретирующее введённые пользователем ответы с помощью нейронной сети.

Объект работы - десктопное приложение, выполняющее следующие функции:

- сбор ответов пользователя по каждому изображению теста Роршаха;
- классификация ответов по категориям «Ответы по животным и популярные ответы» с помощью нейронных сетей;
- подсчёт баллов и постановка предварительного диагноза;
- сохранение результатов в базу данных и отображение на экране пользовательского приложения.

В ходе реализации данной работы был изучен метод проведения психологического тестирования Роршаха, методы анализа и классификации текстовых данных с помощью машинного и глубокого обучения нейронных сетей.

Во время реализации данной работы были разработаны: алгоритм проведения теста Роршаха с интерпретацией введённого пользователем ответа с помощью нейронных сетей, десктопное приложение на базе фреймворка *PySide6* с сохранением результата в базу данных формата *CSV*.

					ПГУ1.090301.04.001 ПЗ			
Изм.	Лист	№ докум	Подпись	Дата				
Разраб.		Григорьева Д.Д.			*Средства для проведения психологического тестирования с использованием нейронной сети*	Лит.	Лист	Листов
Провер.		Зинкин С. А.					2	
Н. Контр.		Бычков А. С.				ФВТ, каф. ВТ гр. 19ВВВ1		
Утверд.		Митрохин М.А.						

Оглавление

Перечень сокращений и обозначений 4
Введение 5
1 Средства для проведения психологического тестирования с использованием нейронной сети 8
1.1 Анализ предметной области 8
1.1.1 Обзор аналогов 12
1.1.2 Анализ требований к разработке 15
1.1.3 Выбор архитектуры приложения 17
1.1.4 Выбор языка программирования 18
1.1.5 Выбор библиотек 19
1.1.6 Выбор среды разработки 21
1.2 Проектирование и разработка 22
1.2.1 Разработка нейронной сети 22
1.2.2 Разработка Байесовского классификатора 28
1.2.3 Реализация пользовательского интерфейса 33
1.3 Анализ результатов 42
1.3.1 Тестирование приложения 43
2 Охрана труда 45
1.4 Потенциальные опасности и вредности при работе с ЭВМ 45
1.4.1 Общие принципы охраны труда при работе с ЭВМ 45
4.1.1 Физические факторы 46
4.1.2 Психологические факторы 46
4.1.3 Организационные факторы 47
4.2 Классификация и категорирование помещения, в котором производится разработка программного обеспечения с использованием ЭВМ 47

4. 2. 1 Значение классификации и категорирования помещения.....47
4. 2. 2 Категории помещений.....48
3. 1. 1 Факторы, влияющие на выбор классификации 49
3. 1. 1 Меры по обеспечению безопасности помещения 49
3. 2 Классификация помещений для работы с ЭВМ по взрывопожарной и пожарной опасности 50
3. 2. 1 Классификация помещений по взрывопожарной опасности ...50
3. 1. 1 Классификация помещений по пожарной опасности 51
3. 1. 1 Применение мер безопасности.....52
4. 1 Санитарно-гигиеническая характеристика производства 52
4. 1. 1 Физическое окружение 52
4. 1. 2 Эргономика рабочего места.....53
4. 1. 3 Клавиатуры и мыши 53
4. 1. 4 Гигиена рабочего места.....53
4. 2 Расчет контура заземления производственного помещения, где установлены персональные ЭВМ.....54
4. 2. 1 Определение требований к контуру заземления 54
4. 2. 2 Расчет сопротивления заземления 54
2. 1. 1 Выбор и размещение заземляющих устройств.....55
2. 1. 2 Проверка эффективности заземления.....55
Заключение 57
Список используемых источников 58
Приложение А - *UML*-диаграммы.....60
Приложение Б - Листинг программы.....64
Приложение В - Презентация 90

Перечень сокращений и обозначений

CSS (Cascading Style Sheets) - каскадные таблицы стилей.

CSV (Comma-Separated Values) - текстовый формат представления табличных данных.

GUI (Graphical User Interface) - графический интерфейс пользователя.

IDE (Integrated Development Environment) - интегрированная среда разработки.

UML (Unified Modeling Language) - унифицированный язык моделирования.

XML (eXtensible Markup Language) - расширяемый язык разметки.

Введение

Тест Роршаха или «пятна Роршаха» — психодиагностический тест для исследования личности, опубликован в 1921 году швейцарским психиатром и психологом Германом Роршахом. Тест состоит из 10 чёрно-белых, чёрно-красных и цветных симметричных изображений (Рис 1), похожих на чернильные или акварельные пятна. Тестируемый должен рассмотреть изображения и описать то, что он видит на них. Проводя исследование среди 405 испытуемых, Роршах заметил, что некоторые категории ответов свидетельствуют об определённых свойствах личности, показал, чем ответы здоровых людей отличаются от ответов психически больных. Роршах сумел формализовать ответы в таблицу с помощью кодов для последующей удобной трактовки. Методика получила известность уже после смерти Роршаха, её изучением занимались такие психологи и психиатры как С. Бек, М. Гертц, Б. Клопфер, З. Пиотровский, Д. Рапапорт, Э. Бом - они и сформировали основные 5 подходов к использованию теста. Бек и Гертц придерживались классического взгляда и предпочтение отдавали стандартизации тестирования и сбору данных, Клопфер основывался на психоаналитической интерпретации формальных характеристик ответов, Пиотровский проводил исследования теста на больных органическими патологиями головного мозга, Рапапорт и Шафер начали попытки отслеживания психодинамики испытуемых, Бом опубликовал наиболее близкое к классической идее Роршаха руководство по интерпретации ответов. Из-за разногласий между подходами, исследования теста начали значительно сокращаться, хотя сам тест являлся одним из наиболее часто используемых проективных тестов, занимал 1,3,4 места в США в 70-80х годах. В 1961 году Джон Экснер сравнил эти методики оценки и объединил их в одну систему оценки, которая используется до сих пор и является наиболее популярной - «Интегративная система Экснера»[1]. Концепция теста Роршаха в рамках данной системы позволяет использовать его для обнаружения достаточно тонких и глубоко скрытых нарушений в различных сферах

психической деятельности. Исследования теста в основном затрагивали интерпретацию ответов, а сами изображения и их формализация изменений не претерпели.

Рисунок 1 - Тест Роршаха

Уникальность теста заключается в том, что в его основе лежит проективная гипотеза, которая предполагает, что малоструктурированный стимульный материал наиболее точно отражает психологическую организацию человека [2]. Испытуемый называет ответ сам, а не выбирает один из предложенных вариантов.

Наиболее значимое отличие теста Роршаха от других проективных методик тестирования (Люшера, Сонди, Маркерта) -*Bewegung* - явление, в котором тестируемый на карточках видит движение. В результате тестирования Роршах выяснил, что здоровые люди, наблюдающие движение на статичных картинках имеют богатое воображение, а люди, имеющие психические расстройства, - склонность к нереалистичным фантазиям [3].

Тест Роршаха получил широкое распространение в клинической психологии, психиатрии, судебной экспертизе. В настоящее время тест Роршаха используется для диагностики заболеваний, таких как:

— задержка психического развития у детей;

— атеросклероз;

— шизофрения

— алкогольные делирий;

— опухоли мозга;

— депрессия;

— эпилепсия или эпилептоидная психопатия

— И др [4].

Актуальность данной дипломной работы заключается в том, что тест Роршаха является одним из наиболее известных мировых методов для индикации свойств личности и установки наличий отклонений психики. Тест занимает лидирующие позиции среди методов, используемых в клинической практике на протяжении более 60 лет. Однако, традиционные методы оценки этого теста требуют высокой квалификации психолога и могут быть подвержены субъективным ошибкам, а существующее на данный момент онлайн-тестирование содержит варианты ответов, вследствие чего является неликвидным. В этой связи, использование нейронной сети для сбора и обработки данных тестирования методом Роршаха представляет собой перспективный подход, который позволяет увеличить точность и объективность результатов тестирования, а так же самостоятельно формализовать ответы быстрее, чем человек. Кроме того, разработка такого алгоритма может значительно упростить процесс проведения тестирования и снизить затраты на обучение психологов.

Цель работы - приложение, осуществляющей постановку предварительного диагноза по тесту Роршаха, интерпретирующее введённые пользователем ответы с помощью нейронной сети.

Задачи:

1. Изучить существующие методы и алгоритмы анализа данных для проведения психологического тестирования.
2. Разработать методы обработки и классификации данных теста Роршаха.
3. Изучить способы проектирования и разработки десктопного приложения с использованием фреймворка *PySide6* для проведения теста Роршаха с использованием нейронной сети.
4. Разработать алгоритмы обработки ответов и построения моделей

нейронных сетей для проведения теста Роршаха.

5. Изучить преимущества и недостатки существующих средств для проведения теста Роршаха с использованием нейронной сети и определить наиболее эффективные методы для создания десктопного приложения.
6. Разработать прототип десктопного приложения для проведения теста Роршаха на основе нейронной сети с использованием фреймворка *PySide6*.
7. Провести тестирование и оценку эффективности разработанного приложения на базе реальных данных.
8. Описать результаты исследования в дипломной работе и сделать выводы о возможностях и перспективах использования нейронных сетей для проведения теста Роршаха.

1 Средства для проведения психологического тестирования с использованием нейронной сети

1.1 Анализ предметной области

В ходе проведения теста, экспериментатор не может давать оценку ответам испытуемого, подсказывать, только задавать уточняющие вопросы на определённом этапе тестирования. Тест проходит в 4 этапа, затем ответы формализуются и выдаётся результат [5].

Первый этап. Тестируемый рассматривает карточки с изображениями, которые даются ему в определённой последовательности, и задаётся вопрос «Что вы видите на данном изображении? Какие образы напоминают вам эти пятна?» Время не ограничивается. Если испытуемый не может дать ответ, ему предлагается повернуть изображение. Поворот карточки фиксируется экспериментатором:

∧ - основное положение карточки;

> — - верхний край карточки справа,

∨ - таблица перевёрнута;

<— - верхний край таблицы слева.

Время фиксируется по минутной стрелке часов.

Второй этап. Экспериментатор проводит уточняющий опрос, содержащий слова «Где?», «Почему?», «Как?», не нарушающие личностного восприятия испытуемого,и задаёт вопросы «Движется ли увиденный образ?» и «Важен ли цвет для распознавания этого образа?»

Третий этап. Экспериментатор уточняет, можно ли данные детерминанты (цвет, оттенки, движение) применить к другим ответам.

Четвёртвый этап. Данный этап необходим для определения пределов чувствительности испытуемого.Определяется:

— может ли испытуемый интегрировать детали изображения в целое;

— может ли он представить движение человеческих образов на изображениях;

— может ли он воспринимать цвет и популярные образы.

На этом этапе экспериментатор может задавать конкретные и уточняющие вопросы, может продемонстрировать популярные образы, предложить испытуемому указать на приятные и неприятные цвета.

Результаты тестирования формализуется по следующим параметрам:

1. Локализация. Ответ включает часть или всё изображение? Локализация кодируется символами *W* (целое или почти целое изображение), *DW* (испытуемый чётко видит одну деталь, а остальной образ додумывается), *WS*(белое пространство).
2. Детерминанты. Важен ли цвет или светотень для интерпретации ответа? Можно ли увидеть указанный образ на монохромном изображении? При формализации: *M* означает движение; *FC* (доминирование формы), *CF* (цветоформовые ответы), *C* (доминирование цвета) - цветовые формы, *FC* (высокая контрастность), *CF* (поверхностный эффект не деффиринцирован), *C* (испытуемый игнорирует элемент формы, встречается только в тяжёлой паталогии), *FK* (светотень способствует эффекту глубины), *K* (свет, заполняющий пространство) - светотень.

3. Содержание ответов. Содержание кодируется символами, например *«Ti»* означает огонь, *«(Hd)»* - части человеческих фигур. Редкие содержания кодируются целым словом.
4. Оригинальность. Под неоригинальными ответами Роршах подразумевал интерпретации, которые даются каждым третьим испытуемым, для каждой карточки имеется свой список популярных ответов. Данный параметр кодируется *Orig+* и *Orig-*, означающие оригинальность и неоригинальность ответа соответственно. Популярные ответы - ответы, встречающиеся примерно у 15% испытуемых [6]. Список популярных ответов на каждую карточку представлен в таблице 1.

Таблица 1 - Популярные ответы на тест Роршаха

Номер карточки	Популярные ответы
1	Летучая мышь, бабочка, птица, человек.
2	Человек, медведь, слон, собака.
3	Человек.
4	Летучая мышь, бабочка.
5	Летучая мышь, бабочка, птица.
6	Черепаха.
7	Облако.
8	Собака, ящерица, тигр.
9	Голова человека.
10	Краб, осьминог, паук, собака, кролик, заяц, лев.

5. Уровень формы. Оценка чёткости восприятия изображения даётся в цифрах, которые могут суммироваться:

— 0,5 - определённому образу приписывается неопределённая концепция;

— 1,0 - тестируемый пытается сопоставить концепцию с образом;

— 1,5 - ложные (конфубуляторные) ответы *DW*;

— 2,0 - концепция не совпадает с образом.

После проведения тестирования выполняются расчёты: посчитывается общее количество ответов, определяется среднее время реакции, отдельно считаются целостные интерпретации, где *F+%* - количество ответов с чёткой формой к общему числу ответов *(R)*, *A%* - процент ответов по животным *(A+Ad)* к общему числу ответов, *P%* - количество популярных ответов к общему числу ответов, *Orig%* - количество оригинальных ответов к общему числу. Далее посчитывается количество ответов по каждой локализации и самый важный параметр - соотношение ответов по цвету и движению. *M* приравнивается к 1, *FC* - 0,5, *CF* = 1, *C* = 1,5. Все значение суммируются и выявляется соотношение ответов, в которых наблюдается движение *(M)* к общему числу ответов *(C)*. В зависимости от результата различают 5 типов переживаний:

1. коартированный узкий - цифры с обеих сторон соотношения равны 0 или 1;
2. коартированный суженный - до 3 с каждой стороны;
3. амбиэквальный - высокие оценки с обеих сторон (больше 3);
4. интроверсивный - преобладание ответов с движением;
5. экстратензивный - преобладание *C*.

Индекс реалистичности *(Ri)* вычисляется по наличию ответов:

1. наличие движения на карточке 3;
2. ответ «летучая мышь» на карточке 5;
3. ответ с наличием животного, находящегося в розовой области на карточке 8;
4. ответ с наличием животного на карточке на карточке 10.

За каждый такой ответ даётся 2 балла, в сумме количество баллов по этому критерию не должно быть больше 8 [5].

Тестируемые с одинаковыми заболеваниями часто дают одинаковые ответы на некоторые категории вопросов, используя эту закономерность и исходя из полученных результатов, ставится предварительный диагноз [7]. В данной работе используются предварительные диагнозы расстройств,

представленные в таблице 2.

Таблица 2 - Показатели теста Роршаха при различных заболеваниях

Показатель теста	Показатели, %				
	Здоровые тестируемые	Больные с алкогольным делирием	Больные с остбредовыми формами шизофрении	Больные с атеросклерозом	Органические поражения головного мозга
Популярные ответы *(orig/ p)*	>90	20	—*	29	>25
Ответы по животным *(A)*	30-35	60	—*	50	—*
Ответы с чёткой формой *(F)*	70-80	<50	67	70	>70
Количество ответов по движению *(M)*	<10	7	10	10	>10
Целостные ответы *(W)*	65-70	27	42	28	—*
Ответы на белый фон *(S)*	—*	4,8	—*	—*	—*
Ответы на мелкие детали *(d)*	—*	58	7	3	—*
Цветоформовые ответы и первичные по цвету *(C)*	16	7	16	5	<90

Примечание: * - данный показатель не влияет на результат

1. 1. 1Обзор аналогов

Использование нейронных сетей для проведения теста Роршаха не проводилось. В качестве альтернативы тестирования специалистом,

существуют онлайн-тесты с вариантами ответов. Данные тесты не могут дать точную оценку личности, так как именно отсутствие правильного ответа или вариантов выбора даёт возможность составить более полную и приближенную к реальности картину психологического состояния человека. В процессе тестирования не должно быть никаких отвлекающих факторов, которые могут "натолкнуть" тестируемого на тот или иной ответ, поэтому данные тесты можно рассматривать как развлекательные, они не могут использовать во врачебной практике в отличие от программы для тестирования с использованием нейронной сети со свободным вариантом ответа.

Сравнение наиболее известных из вышеописанных онлайн-тестов по наличию основных параметров, необходимых для формализации ответа представлено в таблице 3.

Таблица 3 - Анализ аналогов разрабатываемой программы

Категории вопросов, необходимых для интерпретации	Тест Роршаха на testometrika (Рис. 2)	Тест Роршаха на psytests (Рис. 3)	Тест Роршаха на LML (Рис. 4)
Локализация	+	-	+
Детерминанты (форма)	±	-	-
Детерминанты (движение)	+	-	+
Детерминанты (цвет или светотень)	±	-	-
Содержание ответа	+	+	+
Оценка уровня формы	-	-	-

Как видно из таблицы, рассмотренные онлайн-тесты обладают большим количеством недостатков.

Тест на «*testometrika»*[8] опрашивает наиболее подробно, однако отсутствуют уточняющие вопросы по категориям «Оценка уровня формы» и

«Детерминанты (светотень), а вопросы по детерминантам цвета и формы объединены в один. Варианты ответа для категории «Содержание ответа» включают в себя всего 4 неоригинальных пункта.

Рисунок 2 - Тест Роршаха на «*testometrika*»

Опрос на «*psytests*»[9] имеет более полный список содержания ответов, но в нём полностью отсутствуют уточняющие вопросы по остальным категориям.

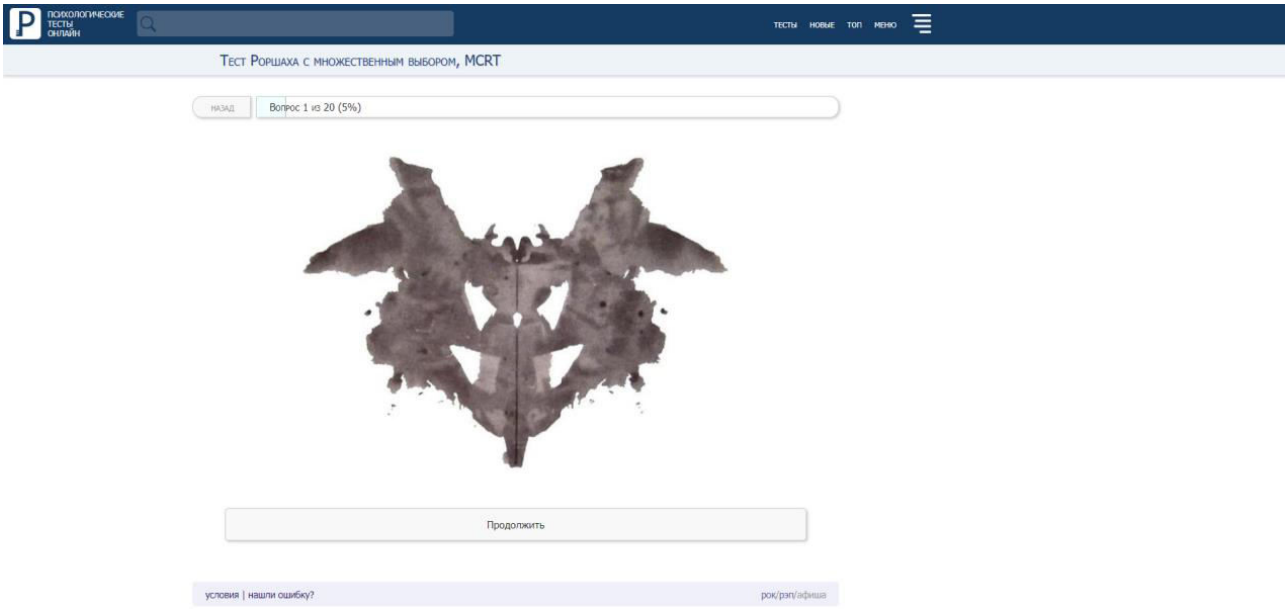


Рисунок 3 - Тест Роршаха на «*psytests*»

Опрос на «*LML*» [10] прелагает варианты ответов из категории «Содержание ответов», которые являются заведомо неоригинальными.

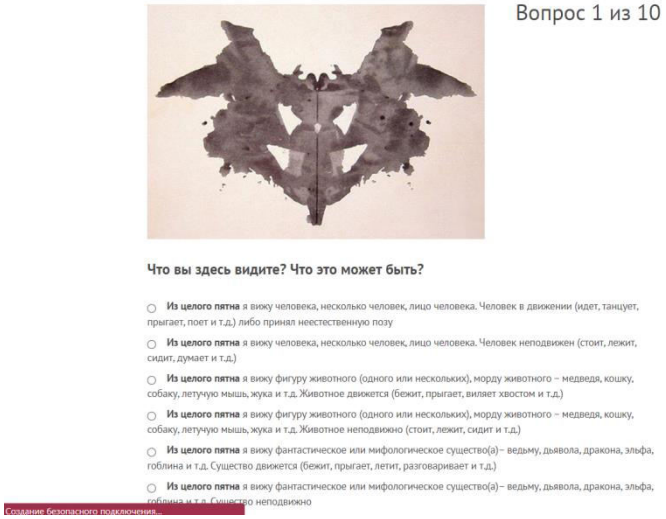


Рисунок 4 - Тест Роршаха на «*LML*»

1. 1. 2Анализ требований к разработке

Целью работы является разработка приложения для проведения теста Роршаха.

Требования к приложению:

Возможность проведения психологического тестирования методом Роршаха;

— обработка результатов тестирования с использованием нейронной сети;
— возможность сохранения результатов тестирования в базу данных;
— возможность просмотра результатов тестирования из базы данных;
— приложение должно быть кроссплатформенным и работать на операционных системах *Windows*, *Linux* и *macOS*;
— интерфейс приложения должен быть интуитивно понятным и удобным для использования.

Данное приложение может использоваться врачами для проведения персонального тестирования или постановки предварительного диагноза или людьми, не имеющими медицинского образования в рамках самообследования. Варианты использования приложения представлены на Рисунке 5.

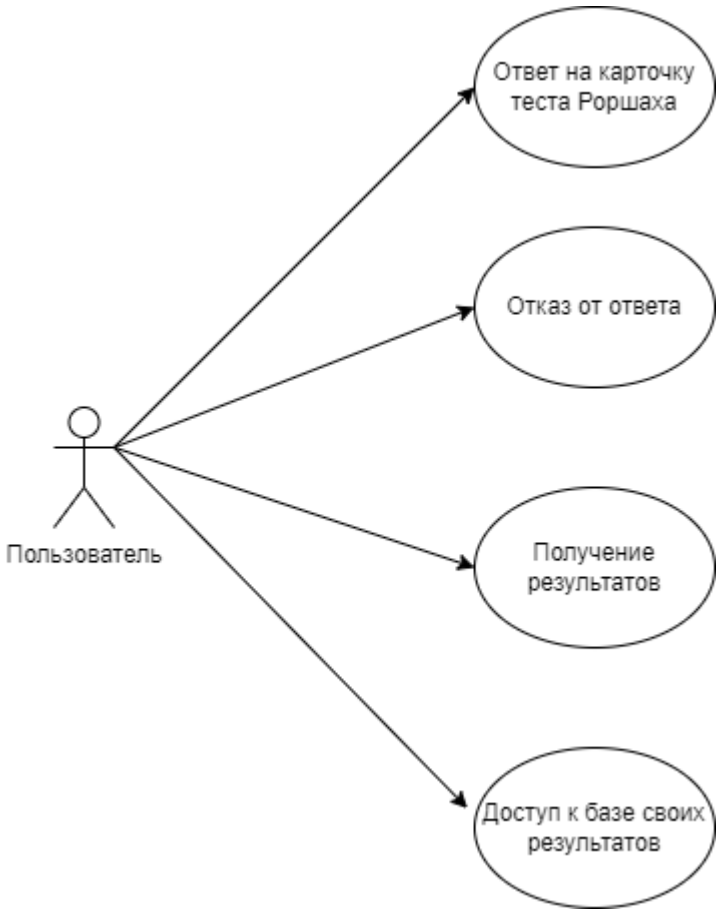


Рисунок 5 - *UML*-диаграмма вариантов использования

При сравнительном анализе аналогичных разработок для проведения психологического тестирования методом Роршаха было выявлено несколько основных типов приложений.

Первый тип - это веб-приложения, которые работают в браузере и требуют постоянного подключения к интернету. Такие приложения могут быть менее удобными в использовании, так как требуют загрузки страниц и обработки запросов на сервере. Этот вариант реализации является не самым удобным, так как тестирования могут проходить люди, страдающие психическими заболеваниями, одним из проявлений которых является паранойя, при которой пользователи будут отказываться проходить тестирование, зная, что их ответы и предварительный диагноз будут передаваться по сети. Многие организации отказываются работать с веб-приложениями, так как они более уязвимы к атакам и утечке данных, особенно когда речь идёт о медицинских диагнозах.

Второй тип - это десктопные приложения, которые работают на локальном

компьютере и не требуют постоянного подключения к интернету. Они обладают более высокой производительностью и скоростью работы, так как не требуют загрузки страниц и обработки запросов на сервере. Также такие приложения могут иметь более удобный и интуитивно понятный интерфейс.

Третий тип - это мобильные приложения, которые работают на смартфонах и планшетах. Они могут быть удобными в использовании в случае, если необходимо проводить тестирование вне офиса или лаборатории. Главным недостатком таких приложений является сравнительно маленькое расширение экрана, при котором неудобно рассматривать изображения, что является важным пунктом при проведении теста Роршаха.

Таким образом, десктопное приложение представляет собой более удобное и эффективное решение для проведения данного тестирования. Оно обладает высокой производительностью и скоростью работы, не зависит от интернет-соединения и может иметь более удобный интерфейс, чем веб-приложения.

Данное приложение будет запускаться пользователем на компьютере, результаты тестирования будут сохраняться в базу данных в формате *csv*, что не займёт много места на стандартном персональном компьютере. Сервер так же имеет пользовательский интерфейс для проведения тестирования, что позволяет пользователям, проводящим самообследование не в медицинском учреждении, запускать только одно приложение, а не его клиентскую и серверную часть.

Так как приложение будет использоваться пользователями без медицинского образования, формулировка уточняющих вопросов должна быть понятной, сам тест должен сопровождаться подробной инструкцией.

1. 1. 3Выбор архитектуры приложения

Пользователь запускает тест, рассматривает изображение, пишет ответ в поле ввода, отвечает на уточняющие вопросы, и отправляет информацию для подсчёта баллов по каждой картинке. Пользователь так же может отказаться

отвечать на вопрос, нажав на кнопку «Отказаться от ответа». После ответа на последнее изображение, на экране отображается предварительный диагноз, результаты которого сохраняются в базу данных. Переходит к следующему изображению, после обработки ответов сервером - пользователь получает интерпретацию. Приложение состоит из клиентской и серверной части.

Требования к разработке:

— создание класса для клиентской части приложения, который будет содержать методы для взаимодействия с пользовательским интерфейсом;
— разработка пользовательского интерфейса с помощью *PySide6*, включая элементы управления для отображения картинок теста Роршаха, кнопки для отправки запросов на сервер и отображения результатов тестирования;
— создание класса, который будет содержать методы для загрузки обученной нейронной сети для обработки ответов;
— реализация функционала для сохранения результатов тестирования

1. 1. 4Выбор языка программирования

Так как приложение будет использовать нейронные сети для реализации приложения был выбран язык *Python*, имеющий наибольшее количество библиотек для машинного и глубокого обучения [11]. *Python* имеет большое и активное сообщество разработчиков, что привело к созданию множества мощных библиотек и фреймворков для построения нейронных сетей, включая *TensorFlow*, *Keras* и *Skit-learn*. Простота использования *Python* делает его отличным выбором для отладки и тестирования нейронных сетей, а многие популярные фреймворки включают встроенные инструменты тестирования и отладки для дальнейшей оптимизации процесса разработки. *Python* предлагает множество библиотек, которые можно использовать для визуализации моделей нейронных сетей, что упрощает понимание результатов и выявление любых потенциальных проблем. Простота, универсальность и поддержка библиотек делают *Python* популярным выбором для машинного обучения и разработки

нейронных сетей.

Помимо удобства в создании и обучении нейронных сетей, *Python* является интерпретируемым языком, что означает, что код может выполняться немедленно без необходимости компиляции, что ускоряет процесс разработки.

1. 1. 5Выбор библиотек

Для создания нейронной сети был выбран *Keras* - фреймворк, созданный для глубокого обучения, написанный на *Python*. Он распространяется бесплатно, позволяет создавать различные типы нейронных сетей, а так же может работать с большим количеством библиотек глубокого обучения. *Keras* имеет модульный подход, то есть к нему можно подключать низкоуровневые библиотеки, наиболее удобная - *TensorFlow* - одна из ведущих платформ глубокого обучения, которая была выбрана для глубокого обучения [12].

Для реализации наивного байесовского классификатора, выбран фреймворк *skit-learn* - это библиотека для машинного обучения на языке *Python*. Она содержит множество инструментов и алгоритмов для классификации, регрессии, кластеризации, предварительной обработки данных и многих других задач машинного обучения. Библиотека содержит большой набор алгоритмов для различных задач машинного обучения. *Scikit-learn* использует оптимизированные алгоритмы и структуры данных, что обеспечивает высокую скорость работы.

Нейронные сети будут обучаться на датасетах формата csv, для работы с ними будет использоваться библиотека *Pandas* - это библиотека для работы с данными на языке *Python*. Она предоставляет удобные структуры данных для хранения и манипуляции с таблицами и временными рядами, он также может автоматически определить типы данных для каждого столбца. Библиотека *Pandas* предоставляет широкий спектр функций и методов для работы с данными, сохраненными в формате *csv*. Это включает в себя выборку данных, манипулирование, агрегацию, преобразование, объединение и группировку. *Pandas* предоставляет высокопроизводительную работу с данными и обработку

больших объемов данных из *csv* файлов.

Для наглядной демонстрации результатов работы нейронных сетей выбрана библиотека *Matplotlib* - это библиотека для построения графиков и визуализации данных на языке программирования *Python*. Она позволяет строить множество разнообразных типов графиков и диаграмм, включая линейные графики, графики рассеяния, гистограммы, и многое другое. *Matplotlib* имеет простой и интуитивно понятный интерфейс, который позволяет легко создавать и настраивать графики. Она также имеет отличную документацию и обширную базу знаний, что делает ее простой для изучения. *Matplotlib* поддерживает множество разнообразных типов графиков, что позволяет отобразить множество различных видов данных. Она также позволяет делать анимации, настраивать шрифты и оси, и многое другое. *Matplotlib* может легко интегрироваться с другими библиотеками Python для научных вычислений, такими как *NumPy* и *Pandas*.

Для работы с массивами данных выбрана библиотека *NumPy (Numerical Python)* – это библиотека для языка программирования *Python*, которая предоставляет многочисленные массивы данных, классы для работы с линейной алгеброй, преобразования Фурье и возможности для интеграции с кодом, написанным на языках программирования *C/C++*, *Fortran* и др. *NumPy* использует оптимизированный код на языке C для многомерных массивов, что позволяет выполнять операции над ними быстрее, чем встроенные типы данных *Python*. *NumPy* предоставляет множество функций для создания, модификации и обработки массивов и матриц, таких как транспонирование, решение уравнений, свертки и *NumPy* может легко интегрироваться с другими библиотеками *Python* для научных вычислений, такими как *SciPy*, *Pandas* и *Matplotlib*, расширяя возможности языка в области научных вычислений.

Для создания графического интерфейса приложения - библиотека *PySide6*. *PySide6* является официальной библиотекой для разработки графического интерфейса пользователя *(GUI)* на языке *Python*, поддерживаемой *Qt Company*.

Главными её преимуществами являются кроссплатформенность (поддерживается на *Windows*, *Linux*, *MacOS*) и широкий набор инструментов. *PySide* имеет редактор для создания интерфейса, в то время как большинство библиотек на *Python*, реализующие *GUI*, позволяют создавать дизайн только написанием кода.

1. 1. 6Выбор среды разработки

PyCharm — это интегрированная среда разработки *(IDE)* для языка программирования Python, которая специально разработана, чтобы помочь разработчикам писать код более эффективно и результативно. Интерфейс IDE представлен на Рисунке 6.

Преимущества PyCharm включают в себя:

1. PyCharm обладает мощным и удобным редактором кода с подсветкой синтаксиса, автодополнением и другими возможностями для ускорения написания кода;

2. PyCharm предоставляет различные инструменты для отладки, включая поддержку использования интерактивной консоли, наблюдение за значениями переменных и брейкпоинты;

3. PyCharm имеет встроенные инструменты для управления зависимостями проекта, включая интеграцию с pip и поддержку виртуальных сред;

4. PyCharm предоставляет поддержку всех основных систем контроля версий, таких как Git, Mercurial, SVN и другие;

5. PyCharm работает на различных операционных системах, включая Windows, macOS и Linux;

6. PyCharm Community Edition является бесплатной и открытой программой с открытым исходным кодом.

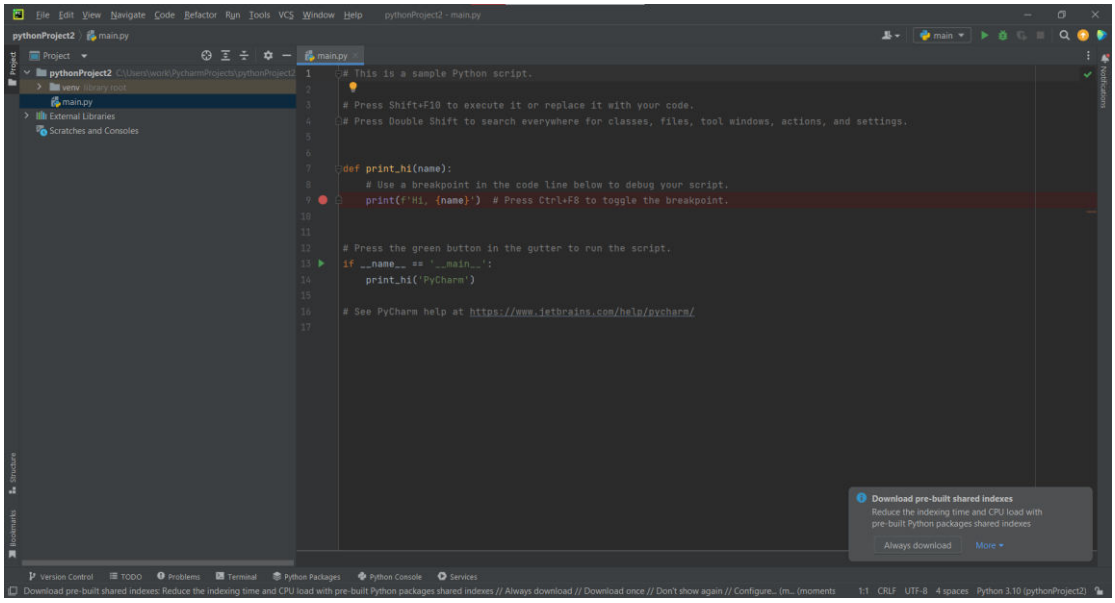


Рисунок 6 - PyCharm

1. 2 **Проектирование и разработка**

Перед запуском программы выполняется инициализация нейронной сети, осуществляющей бинарную классификацию, введённых пользователем ответов, по категориям «Животное», «Не животное».

Автоматически загружается первое изображение, на ответ на которое должен дать пользователь, после нажатия кнопок «Отказаться от ответа» или «Отправить ответ» изображение сменяются. После нажатия на кнопку «Отправить ответ» все введённые пользователем данные сохраняются для последующей интерпретации, а данные их текстового поля ввода классифицируется нейронными сетями. После того, как данные прошли обработку, интерактивные кнопки снова становятся доступными для нажатия, а изображение сменяется на следующее, после ответа на последнее изображение, в окне приложения отображается предварительный диагноз. Предварительный диагноз так же сохраняется в базу данных формата *csv* вместе с введённым именем пользователя.

1. 2. 1Разработка нейронной сети

Для классификации ответа пользователя по категориям «Животное», «Не

животное» рациональней использовать бинарную классификацию с применением глубокого обучения, так как упоминание животного может встретиться в любой части предложения.

Создадим датасет для обучения и тренировки модели, представленный на Рисунке 7. Датасет формата *csv* содержит названия животных и простые слова, второй столбец содержит результаты классификации: 1 - слово относится к категории «Животные», 0 - нет.

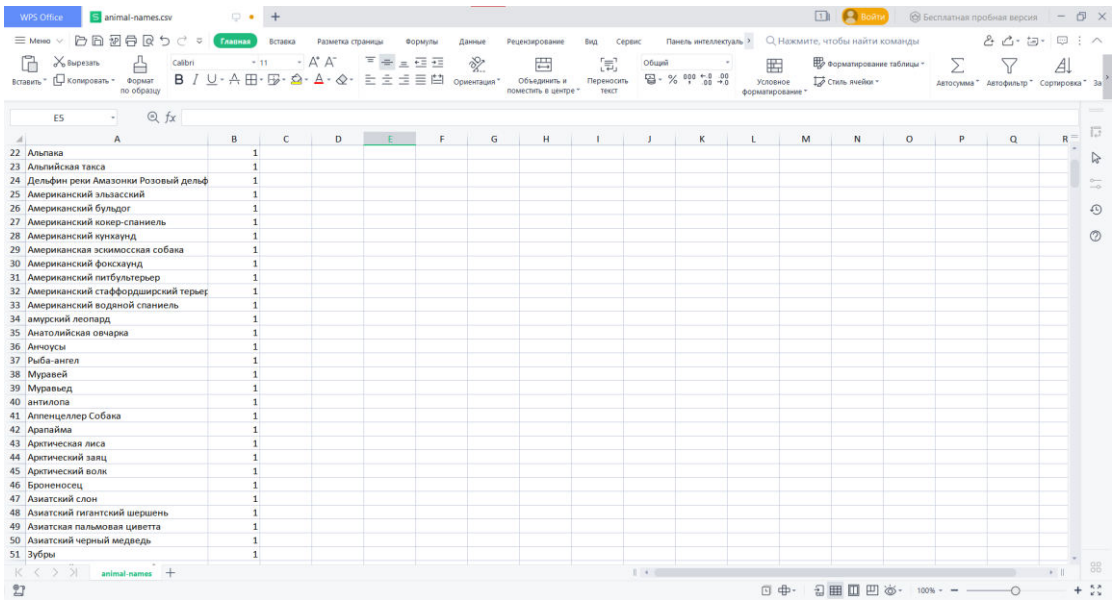


Рисунок 7 - Датасет для обучения нейронной сети

Изменим созданный датасет

```
import pandas as pd
data = pd.read_csv('animal-names.csv', sep=';', names=['text', 'label'], encoding='cp1251', nrows = 2029)
labels = data['label']
texts = data['text']
data.to_csv('out.csv', sep=',', index=False)
```

Так как *keras* не работает напрямую с *csv*, невозможно сразу передать в модель данные, необходимо выполнить преобразование в бинарную матрицу. Преобразование для каждой колонки выполняется отдельно, так как они разных типов данных. Для текстовых данных преобразование выполняется с помощью библиотечкой функции *Vectorizer()*, для колонки результата -

преобразование в массив *np.array*.

```
    # Инициализация векторизатора
    vectorizer = CountVectorizer(binary=True)
    # Преобразование данных в бинарную матрицу
    x_train =
vectorizer.fit_transform(dataframe['text']).toarray().astype('floa
t32')
    y_train = np.asarray(dataframe['label']).astype('float32')
```

Полученные данные необходимо разделить на обучающие и валидационные.

```
    x_val = x_train[-1000:]
    y_val = y_train[-1000:]
    x_train = x_train[:-1000]
    y_train = y_train[:-1000]
```

Далее необходимо сконструировать нейронную сеть и научить её отличать названия животных от остальных слов. Для этого добавим несколько полносвязных слоёв с операцией активации *relu* - блок линейной ретификации, преобразующий в 0 отрицательные значения, а остальные распределяет в интервале [0;1], благодаря чему результат работы функции можно рассматривать как вероятность. Создадим модель и добавим слои. Количество скрытых нейронов в слое не должно быть больше удвоенного размера входного слоя. Последний слой выводит скалярное значение.

```
    model = Sequential()
    model.add(Dense(16, activation='relu', input_shape=num_words))
    model.add(Dense(16, activation='relu'))
    model.add(Dense(1, activation='sigmoid'))
```

При компиляции модели добавим оптимизатор и функцию потерь для перекрёстной энтропии, так она даёт наиболее точные результаты при выполнении бинарной классификации. В качестве метрики будем отслеживать точность.

```
    model.compile(loss='binary_crossentropy', optimizer='rmsprop',
metrics=['accuracy'])
```

Обучим модель.

```
result = model.fit(x_train, y_train, epochs=4, batch_size=512, validation_data=(x_val, y_val))
```

После обучения модели, с помощью библиотеки pyplot выведем график потерь и точности работы модели. Точность повышается, а потери падают с количеством эпох, что видно на Рисунках 8 и 9.

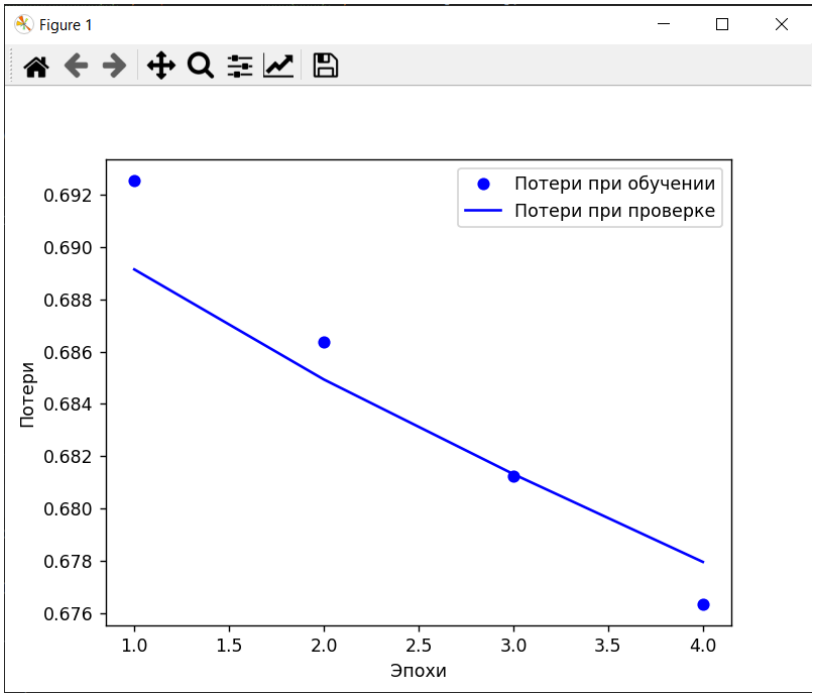


Рисунок 8 - График потерь модели при обучении

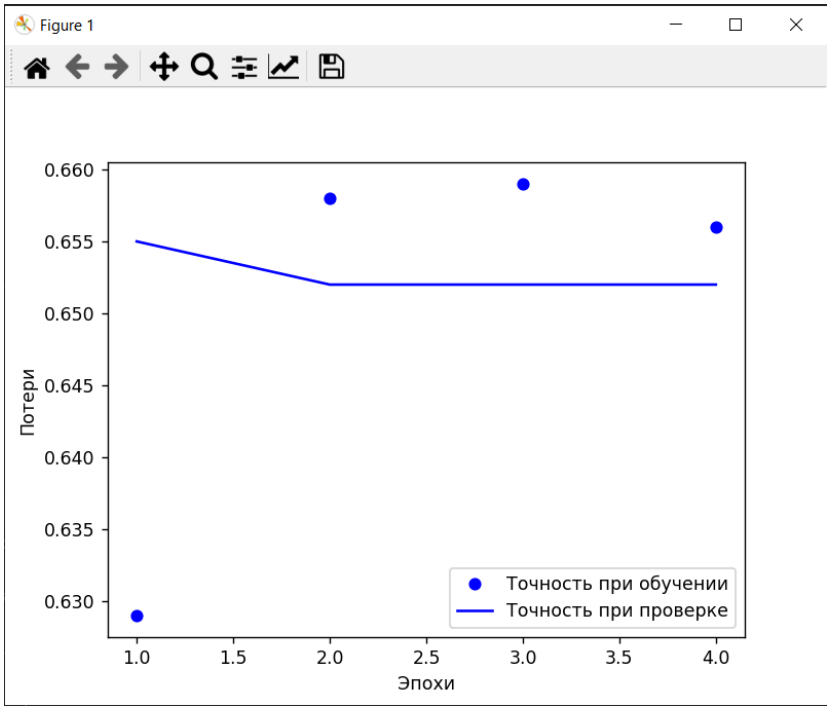


Рисунок 9 - График точности модели при обучении

Так как это бинарная классификация, показатели довольно низкие, изменим параметры обучения модели, как видно на Рисунках 10 и 11, потери падают, а точность повышается практически до 1.

```
    result = model.fit(x_train, y_train, epochs=24, batch_size=64,
validation_data=(x_val, y_val))
```

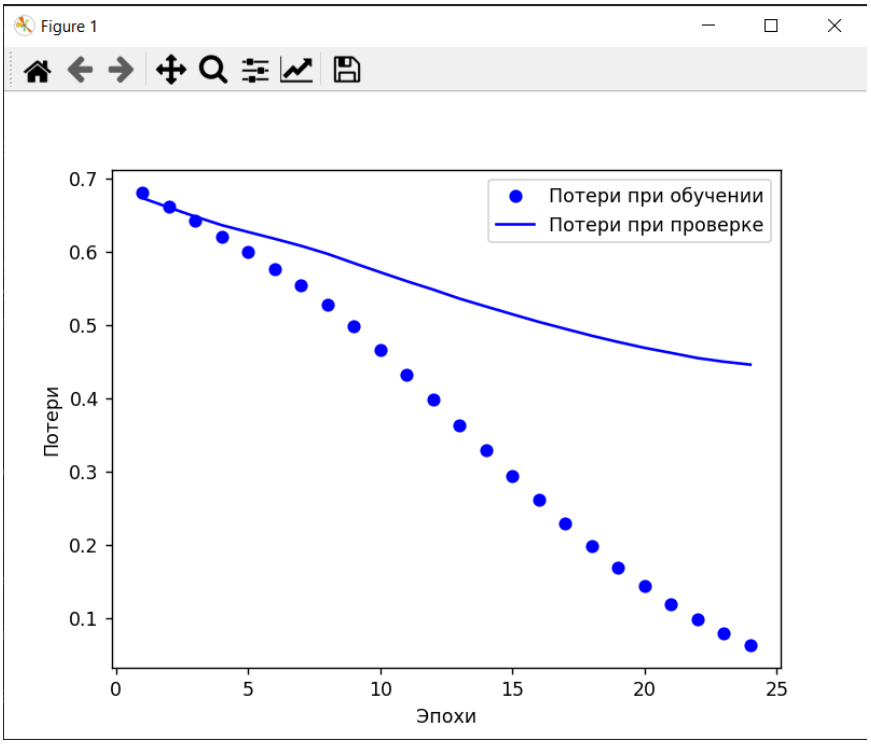


Рисунок 10 - График потерь модели при обучении после изменения параметров

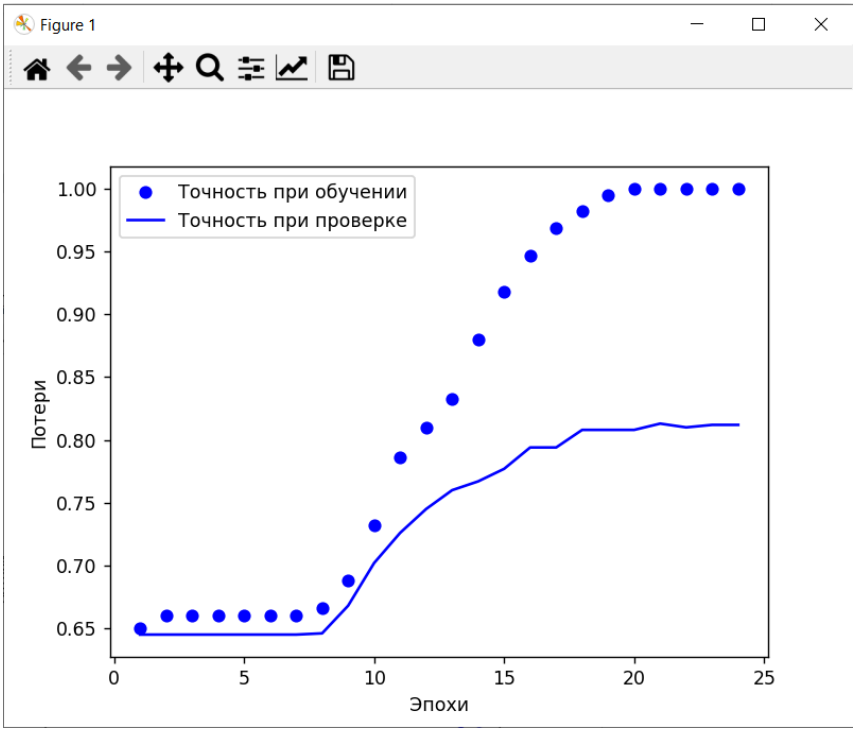


Рисунок 11 - График точности модели при обучении после изменения параметров

Если полученное предсказание равно 0, введённое слово не распознаётся как «Животное» и возвращается 0, если разпознаётся - возвращается 1.

```
count = np.where(prediction > 0.5, 1,0)
```

```
if count == [[0]]:
    return 0
elif count == [[1]]:
    return 1
```

Нейронная сеть обучена, результат работы представлен на Рисунке 12.

```
большая фиолетовая бабочка
1/1 [==============================] - 0s 87ms/step
Prediction:  [[0.98138803]]
1
```

Рисунок 12 - Результат работы нейронной сети

1. 2. 2Разработка Байесовского классификатора

Наиболее рациональным является использование наивного Байесовского классификатора или наивного Байеса - алгоритма машинного обучения для многоклассовой классификации, так как он прост в реализации, потребляет меньше памяти в сравнении с другими методами классификации и работает с высокой скоростью.

Наивный Байес объединяет модель и правило решения. Он представляет собой формулу, вычисляющую условные вероятности, основанную на теореме Байеса - это простая математическая формула, используемая для вычисления условных вероятностей.

$$p(C|F1,\dots,Fn) = \frac{p(C)p(F1,\dots Fn|C)}{p(F1,\dots,Fn)} \qquad (1)$$

где C - класс,

Fn - признаки.

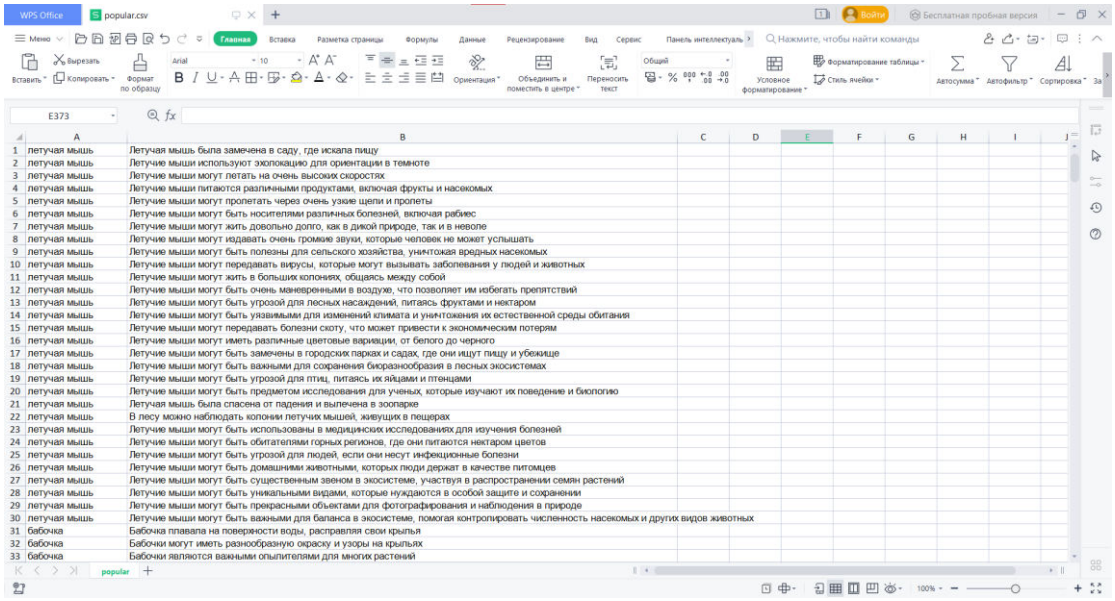
Если в введённом пользователем ответе, встречается слово или словосочетание из категории «Популярные ответы», ответ является популярным и переменная p увеличивается на 1. Вариант с использованием поиска популярного ответа в строке ответа не подходит, так как пользователь может использовать синонимы или изменить склонение и падеж. В таком случае ответ будет являться популярным, но функция поиска расценит его как

false.

Для классификации текста по категориям популярных ответов используется наивный Байесовский классификатор, он сортирует введённый пользователем текст по категориям:

— летучая мышь,
— бабочка,
— птица,
— человек,
— медведь,
— слон,
— собака,
— черепаха,
— облако,
— ящерица,
— тигр,
— голова человека,
— краб,
— осьминог,
— паук,
— кролик,
— заяц,
— лев
— остальное.

Для обучения модели, создаётся датасет в формате *csv*, представленный на рисунке 13. Для каждой вышеописанной категории написано от 18 до 35 предложений.



	A	B
1	летучая мышь	Летучая мышь была замечена в саду, где искала пищу
2	летучая мышь	Летучие мыши используют эхолокацию для ориентации в темноте
3	летучая мышь	Летучие мыши могут летать на очень высоких скоростях
4	летучая мышь	Летучие мыши питаются различными продуктами, включая фрукты и насекомых
5	летучая мышь	Летучие мыши могут пролетать через очень узкие щели и пропеты
6	летучая мышь	Летучие мыши могут быть носителями различных болезней, включая рабиес
7	летучая мышь	Летучие мыши могут жить довольно долго, как в дикой природе, так и в неволе
8	летучая мышь	Летучие мыши могут издавать очень громкие звуки, которые человек не может услышать
9	летучая мышь	Летучие мыши могут быть полезны для сельского хозяйства, уничтожая вредных насекомых
10	летучая мышь	Летучие мыши могут передавать вирусы, которые могут вызывать заболевания у людей и животных
11	летучая мышь	Летучие мыши могут жить в больших колониях, общаясь между собой
12	летучая мышь	Летучие мыши могут быть очень маневренными в воздухе, что позволяет им избегать препятствий
13	летучая мышь	Летучие мыши могут быть угрозой для лесных насаждений, питаясь фруктами и нектаром
14	летучая мышь	Летучие мыши могут быть уязвимыми для изменений климата и уничтожения их естественной среды обитания
15	летучая мышь	Летучие мыши могут передавать болезни скоту, что может привести к экономическим потерям
16	летучая мышь	Летучие мыши могут иметь различные цветовые вариации, от белого до черного
17	летучая мышь	Летучие мыши могут быть замечены в городских парках и садах, где они ищут пищу и убежище
18	летучая мышь	Летучие мыши могут быть важными для сохранения биоразнообразия в лесных экосистемах
19	летучая мышь	Летучие мыши могут быть угрозой для птиц, питаясь их яйцами и птенцами
20	летучая мышь	Летучие мыши могут быть предметом исследования для ученых, которые изучают их поведение и биологию
21	летучая мышь	Летучая мышь была спасена от падения и выпечена в зоопарке
22	летучая мышь	В лесу можно наблюдать колонии летучих мышей, живущих в пещерах
23	летучая мышь	Летучие мыши могут быть использованы в медицинских исследованиях для изучения болезней
24	летучая мышь	Летучие мыши могут быть обитателями горных регионов, где они питаются нектаром цветов
25	летучая мышь	Летучие мыши могут быть угрозой для людей, если они несут инфекционные болезни
26	летучая мышь	Летучие мыши могут быть домашними животными, которых люди держат в качестве питомцев
27	летучая мышь	Летучие мыши могут быть существенным звеном в экосистеме, участвуя в распространении семян растений
28	летучая мышь	Летучие мыши могут быть уникальными видами, которые нуждаются в особой защите и сохранении
29	летучая мышь	Летучие мыши могут быть прекрасными объектами для фотографирования и наблюдения в природе
30	летучая мышь	Летучие мыши могут быть важными для баланса в экосистеме, помогая контролировать численность насекомых и других видов животных
31	бабочка	Бабочка плавала на поверхности воды, расправляя свои крылья
32	бабочка	Бабочки могут иметь разнообразную окраску и узоры на крыльях
33	бабочка	Бабочки являются важными опылителями для многих растений

Рисунок 13 - Датасет для наивного Байесовского классификатора

При первом запуске классификатора до обучения процент правильных ответов по категориям составлял 0.7222222222222222, что видно на Рисунке 14.

```
Метрика accuracy:  0.7222222222222222
Правильные результаты:  0        летучая мышь
1              бабочка
2                птица
3              человек
4              медведь
5                 слон
6               собака
7             черепаха
8               облако
9              ящерица
10                тигр
11    голова человека
12                краб
13            осьминог
14                паук
15              кролик
16                заяц
17                 лев
Name: label, dtype: object
Результаты классификации:  ['летучая мышь' 'бабочка' 'птица' 'человек' 'медведь' 'слон' 'птица'
 'птица' 'птица' 'ящерица' 'тигр' 'голова человека' 'краб' 'птица' 'птица'
 'кролик' 'заяц' 'лев']
```

Рисунок 14 - Результаты работы наивного Байесовского классификатора до подбора параметров

Добавим перебор гиперпараметры, чтобы вычислить наибольшую точность.

Для удобства представим метрику на графике *pyplot*.

```
    def isPopular(inputTxt):
        scores = []
        alpha = [0.1, 0.5, 1.0, 2.5, 3.5, 4.5, 5.0, 6.5, 8.5,
10.0]
        for i in alpha:
            clf = MultinomialNB(alpha=i)
            # Обучение
            clf.fit(x_train, labels)
            # Определение категории введённого текста
            x_test = vectorizer.transform(inputTxt)
            result = clf.predict(x_test)
            scores.append(accuracy_score(result,
train_data['label']))
        plt.plot(alpha, scores)
        plt.xticks(alpha)
        plt.yticks(scores)
        plt.xlabel("alpha")
        plt.ylabel("accuracy_score")
        plt.show()
```

Изменим параметр *alpha*, который будем передавать при обучении модели. Рассмотрим alpha в пределах [0.1, 0.5, 1.0, 2.5, 3.5, 4.5, 5.0, 6.5, 8.5, 10.0]. Наилучший результат модель показала при alpha=0.1, что видно на графике Рисунка 15.

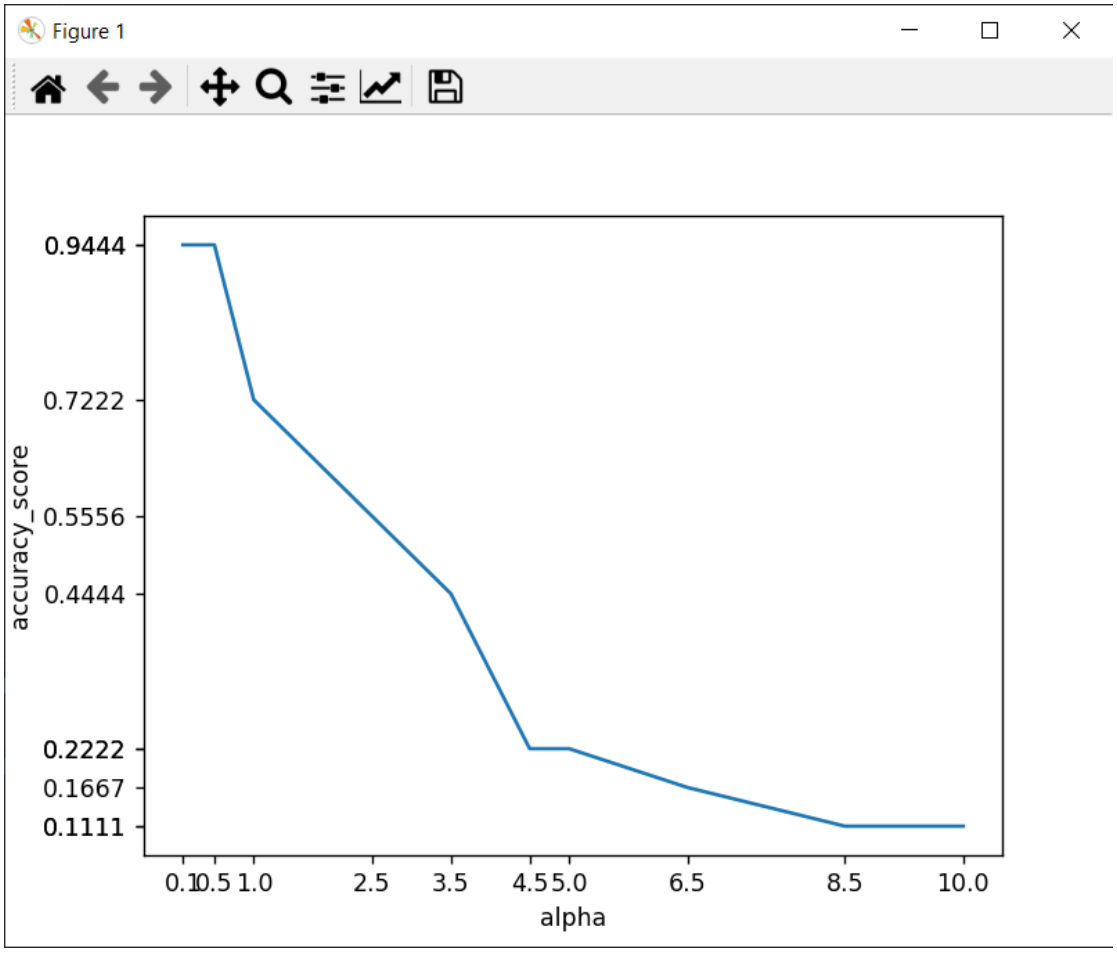


Рисунок 15 - График зависимости точности от alpha наивного Байесовского классификатора

Изменим параметр alpha в соответствии с графиком, на результатах работы модели, представленных на Рисунке 16, видно, что точность прогноза повысилась до 0.9444444.

```
Метрика accuracy:  0.9444444444444444
Правильные результаты:  0        летучая мышь
1              бабочка
2                птица
3              человек
4              медведь
5                 слон
6               собака
7             черепаха
8               облако
9              ящерица
10               тигр
11    голова человека
12               краб
13           осьминог
14               паук
15             кролик
16               заяц
17                лев
Name: label, dtype: object
Результаты классификации:  ['летучая мышь' 'бабочка' 'птица' 'человек' 'медведь' 'слон' 'птица'
 'черепаха' 'облако' 'ящерица' 'тигр' 'голова человека' 'краб' 'осьминог'
 'паук' 'кролик' 'заяц' 'лев']
```

Рисунок 16 - Результаты работы наивного Байесовского классификатора после подбора параметров

Добавим проверку точности результата. Если она меньше 0.5, функция будет возвращать категорию «другое». Это означает, что введённый текст не подходит ни под одну категорию популярных ответов.

```
if max(predicted_probabilities) > 0.5:
    return result
return "другое"
```

1. 2. 3Реализация пользовательского интерфейса

Для разработки интуитивно понятного пользовательского интерфейса, было выбрано приложение *Figma*.

Figma — это инструмент для совместной разработки интерфейсов, который позволяет командам создавать, создавать прототипы и собирать отзывы в одном месте. Это программа для векторного рисования, которая позволяет дизайнерам создавать масштабируемые фигуры, размеры которых можно изменять, не теряя качества. Он предлагает множество преимуществ, в том числе *Figma* позволяет дизайнерам создавать интерактивные прототипы,

которые исследуют, как пользователь может взаимодействовать с их проектами. *Figma* позволяет дизайнерам создавать интерактивные прототипы, которые исследуют, как пользователь может взаимодействовать с их проектами.

Так как разработка пользовательского интерфейса будет проводиться в *Qt Designer* на *XML* с использованием *CSS* основным преимуществом Figma является возможность получить описание созданного элемента на языке *CSS*.

Так как при проведении теста не должно быть факторов, отвлекающих пользователя, было принято решение использовать приглушённые оттенки в качестве основной палитры цветов.

Приложение будет предназначено для свободного использования, поэтому интерфейс должен быть интуитивно понятным и простым, например, обладать подписями к полям ввода, читаемым шрифтом, изменением стиля интерактивного элемента при взаимодействии пользователя с ним. В результате разработки дизайна приложения с учётом всех вышеописанных требований, был получен результат, представленный на рисунке 17.

Рисунок 17 - Дизайн пользовательского интерфейса, разработанный в Figma

Для реализации пользовательского интерфейса был выбран фреймворк *PySide6*. *PySide* имеет удобный редактор интерфейса *QtDesigner*, котором и проводилась разработка. Дизайн приложения описывается на языке *XML* с использованием *CSS*.

Стиль элементов задаётся на языке *CSS* в окне *styleSheets*. Код,

представленный ниже, описывает дизайн главного окна приложения, в результате рисунке 18.

```
*{background-color: #F2E6DF;
font-family: Montserrat Alternates;}

QPushButton {border-radius: 25px;
background-color: #D9C5C1; }

QPushButton:hover:enabled {
background-color: white;
border: 2px solid #D9C5C1; }

QPushButton:enabled { background-color: #D9C5C1;}

QLabel { background: transparent; }
QTextEdit, QLineEdit {background-color: #D9C5C1;
color: #362121;
padding-left: 10px;}

QRadioButton { background: transparent; }
QRadioButton::indicator {
width: 11px;
height: 11px;
border-radius: 5px;
background-color: #D9C5C1;
}
QRadioButton::indicator::checked{
border: 3px solid #F2E6DF;
background-color: #362121;
border-radius: 4px;
width: 7px;
height: 7px;
}
```

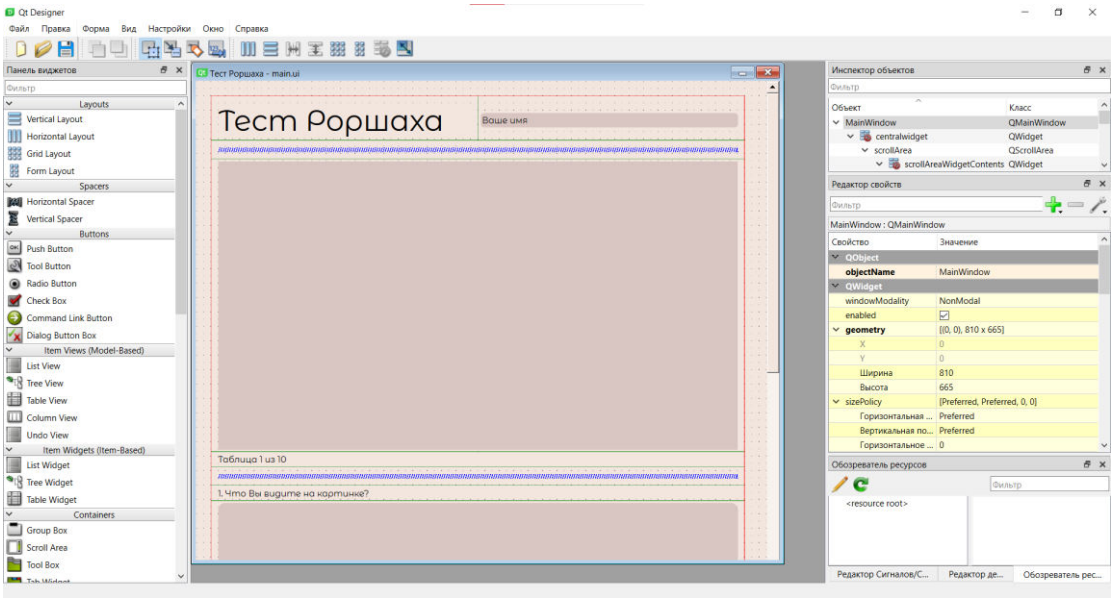


Рисунок 18 - Дизайн пользовательского интерфейса, разработанный в *Qt Designer*

Созданный интерфейс сохраняется с расширением *ui*, для дальнейшей работы необходимо конвертировать его в формат *py*. Конвертация осуществляется командой в консольной строке программы разработки:

```
Pyside6-uic main.ui -o MainWindow.py
```

После конвертации, код, написанный выше на *CSS*, на языке *Python* будет выглядеть следующим образом:

```
    self.scrollAreaWidgetContents.setSizePolicy(sizePolicy)
        self.scrollAreaWidgetContents.setMinimumSize(QSize(830,
1300))
self.scrollAreaWidgetContents.setCursor(QCursor(Qt.ArrowCursor))
self.scrollAreaWidgetContents.setLayoutDirection(Qt.LeftToRight)
self.scrollAreaWidgetContents.setStyleSheet(u"*{background-color:
#F2E6DF;\n"
    "font-family: Montserrat Alternates;}\n"
    "\n"
    "QPushButton {border-radius: 25px;\n"
    "background-color: #D9C5C1; }\n"
    "\n"
    "QPushButton:hover:enabled { \n"
    "background-color: white;\n"
```

Для запуска приложения и последующего добавления функционала

создаём отдельный класс и добавляем в него конструктор.

```
class Main(QMainWindow):
    def __init__(self):
        super(Main, self).__init__()
        self.ui = Ui_MainWindow()
        self.ui.setupUi(self)
```

Особенностью данного приложения является отсутствие вариантов выбора увиденного на изображении. Поле ввода, представленное на Рисунке 19, передаёт полученное значение в нейронные сети, которые классифицируют изображение по параметрам «животное», «популярное». Так как только эти 2 параметра являются необходимыми критериями для постановки предварительного диагноза.

Рисунок 19 - Поле ввода

Далее пользователь выбирает один из предложенных вариантов на каждый уточняющий вопрос и нажимает на любую из кнопок отправки (Рисунок 20).

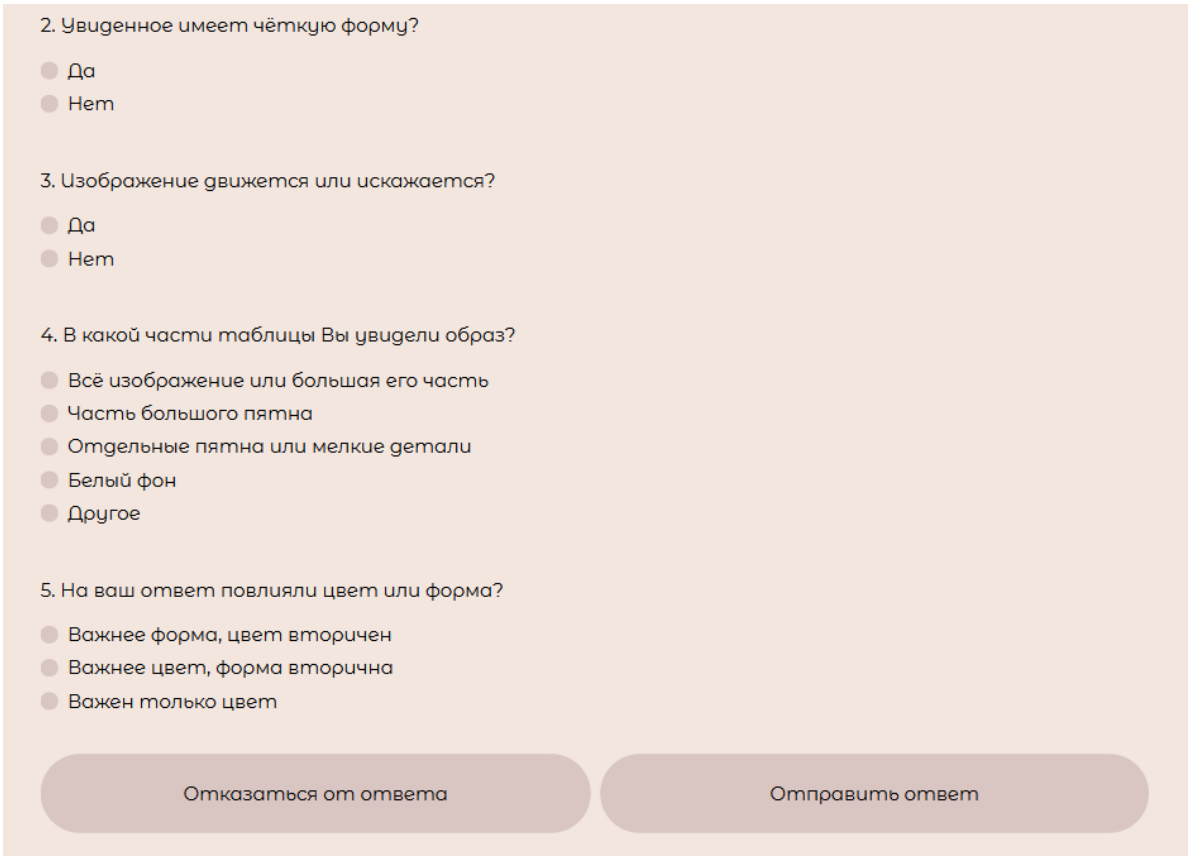


Рисунок 20 - Интерактивные поля отправки ответа и ответов на уточняющие вопросы

Вне зависимости от выбора кнопки, посчитывается количество нажатий, и в интерфейсе отображается изображение под номером, равному числу нажатий, на Рисунке 21 представлен вид приложения после одного нажатия кнопки, что соответствует изображению № 2.

```
    # Загрузка изображения по порядковому номеру
        def setImg(self, numCard):
            pixmap =
QPixmap(('cards/{numCard}.jpg').format(numCard=numCard))
            self.ui.cardImg.setPixmap(pixmap)
        @Slot()
        def countSlideNumber(self):
            self.slideNumber += 1
            match self.slideNumber:
                case 0:
                    self.setImg(1)
                case 1:
```

```
            self.setImg(2)
        case 2:
            self.setImg(3)
        case 3:
            self.setImg(4)
        case 4:
            self.setImg(5)
        case 5:
            self.setImg(6)
        case 6:
            self.setImg(7)
        case 7:
            self.setImg(8)
        case 8:
            self.setImg(9)
        case 9:
            self.setImg(10)
        case _:
            self.ui.cardImg.setText(self.chooseDiagnosis())
            for i in self.elements_to_hide:
                I.hide()
```

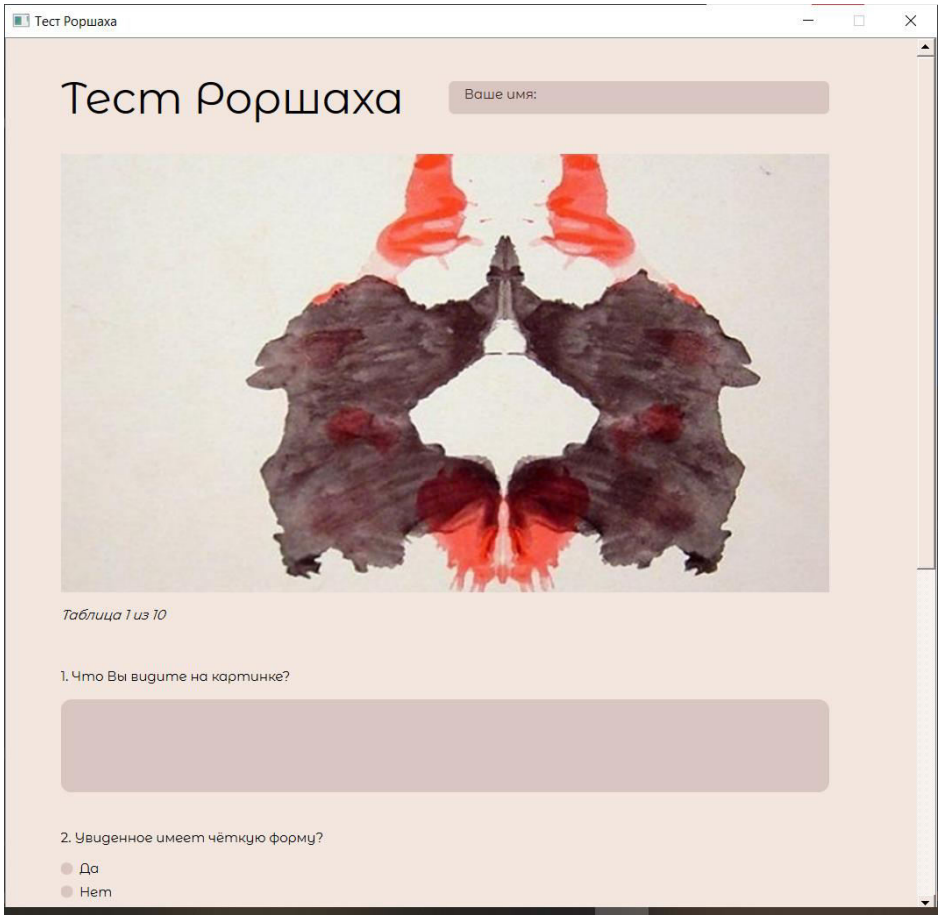


Рисунок 21 - Смена изображения после одного нажатия на кнопку

Каждый выбранный или введённый пользователем параметр, сохраняется в переменную, затем посчитывается общее число ответов (количество нажатий на кнопку «Отправить ответ») и высчитывается процентное соотношение. Процентные соотношения по каждому параметру сравниваются с заданными показателями, при наличии совпадений пользователь получает сообщение о предварительном диагнозе.

```
# Подсчёт ответа по категории в процентах
    def countScoreInPercent(self, param):
        return param / self.numberOfReplies * 100
```

После отправки последнего ответа, вычисляется процент ответов, затем сравнивается и отображается на экране пользователя в соответствии с диагнозом так, как показано на Рисунке 21.

```
        if (scoreP>90 and 30>scoreA>35 and 70>scoreF>80 and
scoreM<10 and 65<scoreW<70 and 14<scoreC<18):
```

```
            self.diagnosis = "Здоровый тестируемый"
        elif (18>scoreP>22 and 58>scoreA>62 and scoreF<50 and
5>scoreF>9 and 25>scoreW>29 and 4>scoreS>7 and 56>scoreD>60 and
5>scoreC>9):
            self.diagnosis = "Больные с алкогольным делирием"
        elif (65>scoreF>69 and 8>scoreM>12 and 40>scoreW>44 and
5>scoreD>9 and 14>scoreC>18):
            self.diagnosis = "Больные с остбредовыми формами
шизофрении"
        elif (27>scoreP>31 and 48>scoreA>52 and 68>scoreF>72 and
8>scoreM>12 and 26>scoreW>30 and 1>scoreD>5 and 3>scoreC>7):
            self.diagnosis = "Атеросклероз"
        elif (scoreP>25 and scoreF>70 and scoreM>10 and
scoreC<90):
            self.diagnosis = "Органические поражения головного
мозга"
        else:
            self.diagnosis = "Ваши показатели не подходят к
категориям, которые тестирует данное приложение"
        # Сохранение результата в базу данных
        with open('result.csv', mode='a', newline='') as file:
            writer = csv.writer(file)
            writer.writerow([self.ui.nameInput.toPlainText(),
self.diagnosis])
        return "Предварительный диагноз: "+self.diagnosis
```

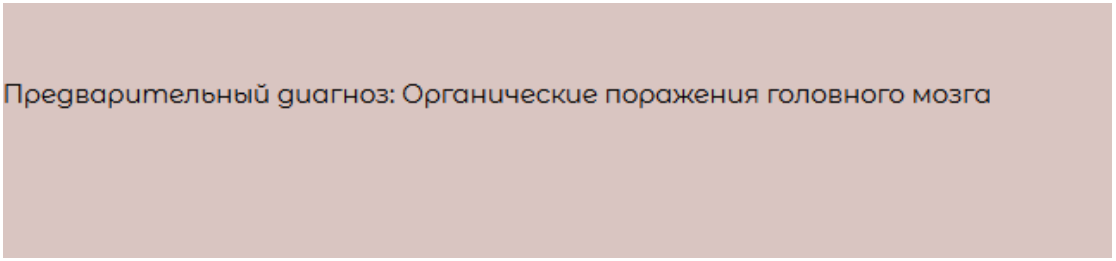


Рисунок 22 - Отображение диагноза на экране пользователя

1. 3 **Анализ результатов**

Было реализовано десктопное приложение с применением фреймворка

Pyside6, осуществляющее тестирование и постановку предварительного диагноза по тесту Роршаха с интерпретацией введённых пользователем ответов с помощью наивного Байсовского классификатора, реализованного на основе библиотеки *skit-learn* и нейронной сети, осуществляющей бинарную классификацию на основе библиотеки *keras*. Для обучения нейронных сетей были созданы датасеты в формате *csv*, загрузка которых осуществлялась с помощью библиотеки *pandas*.

1. 3. 1Тестирование приложения

Тестирование приложения проводилось путём введения данных из открытых протоколов больных, имеющих один из диагнозов, предварительную постановку которого осуществляет данное приложение [13]. Введённые данные одного из тестируемых с диагнозом «Острые бредовые формы шизофрении» представлены в Таблице 4.

Таблица 4 - Ответы тестируемого с диагностированными острыми бредовыми формами шизофрении

Вопрос	Ответ тестируемого									
	1	2	3	4	5	6	7	8	9	10
Что вы видите на картинке?	Летучая мышь	Отверстие в стене	Кувшин	Животные	Летучая мышь	Ручка в чернилах	Два зайчика из	Два хомяка	Бокал	Цветы жёлтой
Увиденное имеет чёткую форму?	Да	Да	Да	Да	Да	Нет	Да	Да	Нет	Да

Продолжение Таблицы 4.

Вопрос	Ответ тестируемого									
	1	2	3	4	5	6	7	8	9	10
Изображение движется или искажается?	Нет	Нет	Нет	Нет	Нет	Нет	Нет	Нет	Да	Нет
В какой части таблицы вы увидели образ?	Всё пятно	Всё пятно	Всё пятно	Всё пятно	Всё пятно	Всё пятно	Всё пятно	Всё пятно	Всё пятно	Всё пятно
На ваш ответ повлиял цвет или форма?	Форма	Форма	Форма	Форма	Форма	Форма	Цвет	Форма	Форма	Цвет

Предварительный диагноз совпал с диагнозом, поставленным врачом, что подтверждает точность и правильность работы приложения.

Окно клиентского приложения, с выведенным предварительным

диагнозом представлено на Рисунке 23.

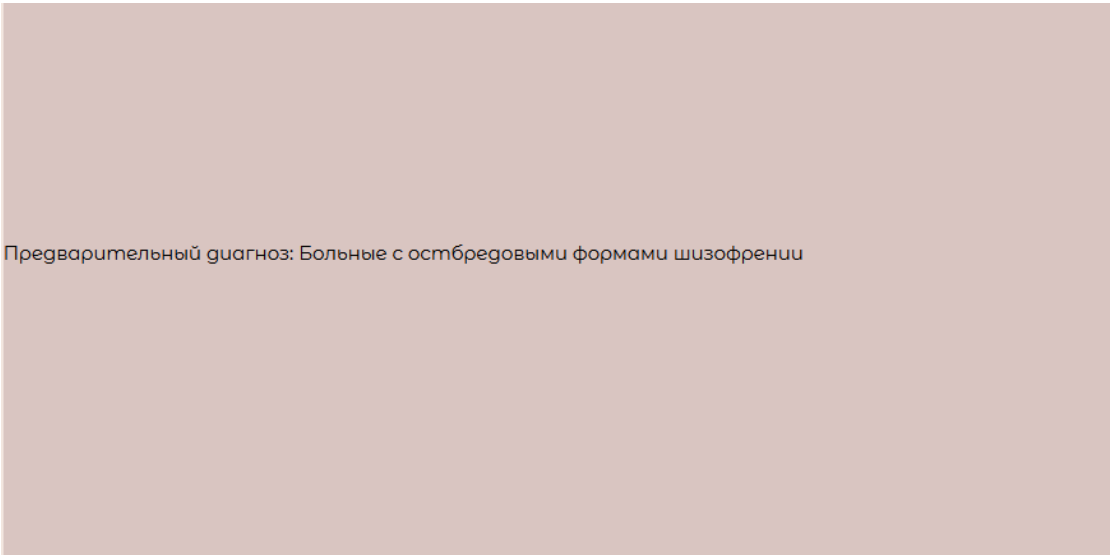


Рисунок 23 - Предварительный диагноз, поставленный приложением

2 Охрана труда

1. 4 Потенциальные опасности и вредности при работе с ЭВМ

1. 4. 1Общие принципы охраны труда при работе с ЭВМ

При работе с электронно-вычислительными машинами (ЭВМ) возникают определенные потенциальные опасности и вредности для здоровья работников. Для обеспечения безопасности и здоровья персонала необходимо придерживаться следующих принципов охраны труда:

1. Работодатель и работники должны быть осведомлены о требованиях безопасности, правилах использования ЭВМ и соблюдать их в процессе работы.
2. Работники, особенно те, кто проводит длительное время за ЭВМ, должны проходить регулярные медицинские осмотры для выявления возможных проблем здоровья, связанных с работой.
3. Рабочее место должно быть организовано с учетом принципов эргономики, чтобы минимизировать физическую нагрузку на работников и предотвратить возникновение травм и заболеваний опорно-двигательной

системы.

4. Работодатель должен обеспечить обучение работников правилам безопасной работы с ЭВМ и предоставлять им необходимую информацию о потенциальных опасностях и мерах предосторожности.

4. 1. 1Физические факторы

При работе с ЭВМ работники могут подвергаться воздействию электромагнитных полей и излучению. Возможные источники включают мониторы, клавиатуры, системные блоки и другое компьютерное оборудование. Эти факторы могут вызывать негативное влияние на здоровье, такое как утомляемость глаз, головные боли и электромагнитная чувствительность. Поэтому необходимо проводить соответствующие исследования, обеспечивать соответствие стандартам и рекомендациям по радиационной безопасности и применять защитные меры.

Недостаточное освещение или неправильное расположение источников света на рабочем месте может вызывать утомление глаз и затруднять восприятие информации на мониторе. Высокий уровень шума также может оказывать негативное воздействие на работников, приводить к стрессу и ухудшению концентрации. Рекомендуется обеспечивать достаточное освещение рабочей зоны и принимать меры по снижению шума, например, использовать звукоизолирующие материалы или наушники с пассивной звукоизоляцией.

4. 1. 2Психологические факторы

Длительная работа на мониторе может приводить к ухудшению зрения и развитию синдрома компьютерного зрения. Симптомы включают раздражение глаз, сухость, жжение и размытость зрения. Для снижения нагрузки на глаза рекомендуется соблюдать режим работы и перерывов, выполнять упражнения для глаз и использовать антибликовые экраны или очки.

Длительное нахождение в статическом положении и неправильная поза

могут вызывать боли в спине, шее, плечах и руках. Рекомендуется использовать эргономичное оборудование, такое как регулируемые стулья и столы, чтобы обеспечить правильную позу и поддерживать оптимальное положение тела.

4. 1. 3Организационные факторы

Длительная непрерывная работа за ЭВМ может приводить к усталости и снижению производительности. Работодатель должен установить регламентированные перерывы для отдыха и расслабления, а также способствовать эффективному использованию рабочего времени.

Работа с ЭВМ может быть связана со стрессом, особенно в случаях, когда требуется справляться с высоким объемом информации или работать под давлением сроков. Работодатель должен предоставлять поддержку работникам, обеспечивать психологическую безопасность и регулярно проводить оценку психоэмоционального состояния персонала.

Работодатель может разрабатывать программы тренировок, включающие упражнения для глаз, шеи, спины и рук, чтобы снизить напряжение и предотвратить развитие травм и заболеваний. Также могут проводиться профилактические мероприятия, такие как лекции, семинары или консультации по охране труда при работе с ЭВМ.

4. 2 Классификация и категорирование помещения, в котором производится разработка программного обеспечения с использованием ЭВМ

4. 2. 1Значение классификации и категорирования помещения

При разработке программного обеспечения с использованием ЭВМ особое внимание должно уделяться безопасности и охране труда. Классификация и категорирование помещения, где происходит работа, играют важную роль в определении требований к безопасности, организации рабочего пространства и

применяемых мер предосторожности. Классификация помещения основывается на его характеристиках и рисках, связанных с работой с ЭВМ, и позволяет разработать соответствующие стратегии по обеспечению безопасности персонала.

4. 2. 2Категории помещений

Помещения в которых проводится разработка, можно разделить на категории:

1. Открытые помещения (open space).

Открытые помещения являются одной из наиболее распространенных форм организации рабочего пространства при разработке программного обеспечения. Они характеризуются отсутствием физических перегородок между рабочими местами и способствуют коммуникации и взаимодействию сотрудников. Однако, открытые помещения могут иметь негативное влияние на концентрацию и приватность работников, поэтому необходимо принять соответствующие меры для минимизации возможных рисков, такие как обеспечение акустической и визуальной изоляции, использование пеегородок или зон для индивидуальной работы.

2. Изолированные помещения

Изолированные помещения предоставляют работникам более приватное и независимое рабочее пространство. Это могут быть кабинеты или отдельные офисы, где каждый сотрудник имеет свое собственное рабочее место. Изолированные помещения обеспечивают более высокий уровень концентрации и предотвращают отвлечения. Однако, они также могут создавать ощущение изоляции и затруднять коммуникацию между сотрудниками. Поэтому важно найти баланс между приватностью и коллаборацией, обеспечивая возможности для взаимодействия и общения.

3. Смешанные помещения

Смешанные помещения представляют собой комбинацию открытых и изолированных пространств. Они могут быть организованы в виде зон с

открытыми рабочими пространствами для коллективной работы и переговоров, а также отдельными кабинетами или офисами для индивидуальной работы или встреч. Такой подход позволяет совместить преимущества открытого и изолированного рабочего пространства, обеспечивая комфортные условия для разработчиков программного обеспечения.

3. 1. 1Факторы, влияющие на выбор классификации

При выборе классификации и категоризации помещения для разработки программного обеспечения следует учитывать ряд факторов:

1. Размер команды и ее структура: Большие команды могут требовать открытых пространств для облегчения коммуникации, в то время как небольшие команды могут предпочитать более индивидуальные рабочие места.
2. Чувствительность информации: Если работа включает обработку конфиденциальных данных или интеллектуальной собственности, может потребоваться более изолированное рабочее пространство для обеспечения конфиденциальности.
3. Требования к безопасности: В зависимости от характера проекта или организации могут быть дополнительные требования к безопасности, такие как ограниченный доступ к помещению или применение физических мер защиты.

3. 1. 1Меры по обеспечению безопасности помещения

Не зависимо от выбранной классификации помещения, необходимо принимать меры для обеспечения безопасности и охраны труда. Важными мерами являются:

— Обеспечение достаточного освещения и хорошей вентиляции.
— Предоставление эргономичной мебели и оборудования, чтобы предотвратить развитие травм и заболеваний опорно-двигательной системы.
— Обеспечение надлежащего уровня шума и контроль за радиационной

безопасностью, если применимо.

— Предоставление возможностей для перерывов и отдыха, чтобы снизить утомление и повысить эффективность работы.

— Обеспечение соблюдения правил пожарной безопасности и эвакуационных планов.

3. 2 Классификация помещений для работы с ЭВМ по взрывопожарной и пожарной опасности

При работе с электронно-вычислительными машинами (ЭВМ) существует определенный уровень взрывопожарной и пожарной опасности. Электроника, электрические компоненты и проводка могут стать источниками возгорания или взрыва, если не соблюдаются соответствующие меры предосторожности. Поэтому важно классифицировать помещения, в которых производится работа с ЭВМ, с учетом этих опасностей и применять соответствующие меры по предотвращению пожаров и взрывов.

3. 2. 1Классификация помещений по взрывопожарной опасности

По пожароопасности помещения можно классифицировать по следующим категориям:

1. Помещения, относящиеся к категории А.

Помещения категории А представляют наибольшую взрывопожарную опасность. В них находятся источники горения, такие как большие серверные помещения, где размещаются серверы и сетевое оборудование. Такие помещения требуют строгого соблюдения норм и правил пожарной безопасности, а также применения специальных систем пожаротушения, например, газовых или аэрозольных систем.

2. Помещения, относящиеся к категории Б.

Помещения категории Б имеют умеренную взрывопожарную опасность. Это могут быть помещения разработчиков программного обеспечения, где размещаются компьютеры и периферийные устройства. Для таких помещений

требуется обеспечение надлежащей проводки и электрического оборудования, регулярная проверка на наличие коротких замыканий и перегрузок, а также применение систем автоматического пожаротушения и аварийного отключения электропитания.

3. Помещения, относящиеся к категории В.

Помещения категории В имеют наименьшую взрывопожарную опасность. Это могут быть офисы или рабочие пространства, где происходит непосредственная работа с ЭВМ, но без больших серверных комнат. Такие помещения требуют соблюдения правил электробезопасности, регулярной проверки электрического оборудования и использования средств пожаротушения, таких как огнетушители.

3. 1. 1Классификация помещений по пожарной опасности

По пожароопасности помещения можно классифицировать по следующим категориям:

1. Помещения, относящиеся к классу А.

Помещения класса А отличаются высокой пожарной опасностью. Это могут быть помещения с большим количеством горючих материалов или легковоспламеняющихся жидкостей, таких как склады электронных компонентов или хранилища запасных частей. Для таких помещений требуется установка автоматических систем пожаротушения, регулярная инспекция пожарной безопасности и наличие планов эвакуации.

2. Помещения, относящиеся к классу Б.

Помещения класса Б имеют умеренную пожарную опасность. Это могут быть офисные помещения или помещения с ограниченным количеством горючих материалов. Для таких помещений требуется обеспечение наличия огнетушителей, систем предупреждения пожара и планов эвакуации.

3. Помещения, относящиеся к классу В.

Помещения класса В имеют низкую пожарную опасность. Это могут быть обычные офисные помещения или рабочие пространства, где используется

минимальное количество горючих материалов. Для таких помещений требуется регулярная проверка электропроводки, наличие огнетушителей и организация тренировок по эвакуации в случае пожара.

3. 1. 1Применение мер безопасности

Вне зависимости от классификации и категории помещений, в которых производится работа с ЭВМ, важно применять меры безопасности для предотвращения пожаров и взрывов. Это включает следующие меры:

1. Регулярная проверка и техническое обслуживание электрического оборудования.
2. Использование огнетушителей и систем пожаротушения соответствующего типа и класса.
3. Обучение персонала правилам пожарной безопасности и проведение тренировок по эвакуации.
4. Создание планов эвакуации и размещение соответствующих указателей и выходных путей.

4. 1 Санитарно-гигиеническая характеристика производства

Санитарно-гигиенические условия на рабочем месте играют важную роль в поддержании здоровья и благополучия работников. Компании, занимающиеся разработкой программного обеспечения, включают в свою структуру офисы, где размещаются электронно-вычислительные машины (ЭВМ), и поэтому необходимо провести санитарно-гигиеническую характеристику производства для обеспечения безопасных и здоровых условий работы.

4. 1. 1Физическое окружение

Освещение является одним из ключевых факторов, влияющих на комфорт и здоровье работников. Офисы, в которых размещаются ЭВМ, должны обеспечивать достаточный уровень освещения, чтобы избежать напряжения

глаз и усталости. Рекомендуется использовать натуральное освещение, где это возможно, и дополнительное искусственное освещение для поддержания комфортной рабочей среды.

Хорошая вентиляция является неотъемлемой частью санитарно-гигиенических условий в офисных помещениях. Она способствует поддержанию свежего воздуха и удалению загрязнений и вредных веществ из воздуха. Помещения должны быть оборудованы эффективной системой вентиляции или иметь доступ к достаточному количеству свежего воздуха.

4. 1. 2Эргономика рабочего места

Рабочая мебель должна быть эргономичной и поддерживать правильное положение тела работника. Столы и стулья должны быть регулируемыми по высоте, чтобы обеспечить оптимальное положение спины, шеи и рук. Кроме того, необходимо предоставить достаточное пространство для размещения компьютерного оборудования и периферийных устройств.

4. 1. 3Клавиатуры и мыши

Клавиатуры и мыши, используемые при работе с ЭВМ, также должны быть эргономичными. Они должны обеспечивать удобное и естественное положение рук и предотвращать нагрузку на запястья и пальцы. Компания должна предоставить сотрудникам возможность выбора оптимальных устройств и обеспечить их регулярную проверку и обслуживание.

4. 1. 4Гигиена рабочего места

Офисы компаний, занимающихся разработкой программного обеспечения, должны иметь соответствующие санитарные помещения, такие как туалеты и умывальники, которые должны быть чистыми и обеспечивать удобные условия для работников.

Рабочее пространство должно быть организовано таким образом, чтобы обеспечить удобство и безопасность работников. Это включает поддержку

порядка и чистоты на рабочих столах, размещение кабелей и проводов таким образом, чтобы они не создавали помехи и рисковали споткнуться, а также наличие удобного доступа к необходимым материалам и оборудованию.

4. 2 Расчет контура заземления производственного помещения, где установлены персональные ЭВМ

Контур заземления является важной составляющей системы электробезопасности в производственных помещениях, где установлены персональные электронно-вычислительные машины (ЭВМ). Он обеспечивает надежное заземление для защиты от электростатического разряда и предотвращения повреждений оборудования и травмирования работников. В данной главе будет представлен расчет контура заземления производственного помещения с установленными персональными ЭВМ.

4. 2. 1Определение требований к контуру заземления

Перед проведением расчета контура заземления необходимо определить требования, предъявляемые к системе заземления в производственном помещении с ЭВМ. Эти требования могут включать:

- Необходимость обеспечения низкого уровня сопротивления заземления для эффективного отвода токов короткого замыкания и предотвращения повышенного напряжения на оборудовании и корпусах ЭВМ.
- Защита от электростатического разряда для предотвращения повреждений электронных компонентов и сбоев в работе ЭВМ.
- Соответствие нормативным требованиям и стандартам безопасности, установленным в отрасли или регулирующими органами.

4. 2. 2Расчет сопротивления заземления

Расчет сопротивления заземления является важной частью процесса проектирования контура заземления. Для этого необходимо учитывать следующие факторы:

1. Форма и размеры помещения могут влиять на выбор и размещение заземляющих устройств. Необходимо учесть размеры комнаты, расположение стен и распределение персональных ЭВМ для определения оптимального места размещения заземляющих электродов.
2. Тип грунта, на котором расположено производственное помещение, может влиять на сопротивление заземления. Различные типы грунта имеют различные электрические свойства, и это следует учитывать при расчете сопротивления заземления.

2.3 Количество и мощность персональных ЭВМ

Количество и мощность установленных персональных ЭВМ влияют на сопротивление заземления. Чем больше ЭВМ, тем больше токов короткого замыкания может потребоваться отвести, и тем ниже должно быть сопротивление заземления.

2. 1. 1Выбор и размещение заземляющих устройств

На основе расчета сопротивления заземления необходимо выбрать и разместить заземляющие устройства в производственном помещении. Это могут быть заземляющие электроды, заземляющие полосы или другие подобные устройства. Размещение должно быть таким, чтобы обеспечить эффективное заземление для всего оборудования и минимизировать риски электростатического разряда.

2. 1. 2Проверка эффективности заземления

После установки заземляющих устройств необходимо проверить их эффективность. Это может быть выполнено с помощью измерительных приборов для проверки сопротивления заземления и электростатического потенциала в производственном помещении.

Расчет контура заземления производственного помещения с установленными персональными ЭВМ является важным шагом в обеспечении безопасной и надежной работы системы электробезопасности. Правильное

определение требований, расчет сопротивления заземления, выбор и размещение заземляющих устройств и проверка их эффективности помогут минимизировать риски электростатического разряда и обеспечить безопасные условия работы с персональными ЭВМ.

Заключение

В ходе реализации данной работы был изучен метод проведения психологического тестирования Роршаха, методы анализа и классификации текстовых данных с помощью машинного и глубокого обучения нейронных сетей.

Так же были получены навыки создания десктопного приложения с помощью фреймворка *PySide6* на языке *Python 3.10*, усовершенствованы навыки объектно-ориентированного программирования и графической реализации пользовательского интерфейса.

Во время реализации данной работы были разработаны: алгоритм проведения теста Роршаха с интерпретацией введённого пользователем ответа с помощью нейронных сетей, десктопное приложение на базе фреймворка *PySide6* с сохранением результата в базу данных формата *CSV*.

В дальнейшем программу можно усовершенствовать, расширив количество предварительных диагнозов, которые может классифицировать данное приложение. Так же точность постановки предваятельного диагноза данным приложением может улучшить возможность пользователя давать множественные ответы на каждую из карточек.

Список используемых источников

1. Jr. John E. Exner The Rorschach: A Comprehensive System, Vol 1: Basic Foundations. - Volume One edition изд. - Wiley, 1993. - 642 с.

2. Ванеян С. С. Тест Роршаха: случайности восприятия и закономерности воображения // Современная терапия в психиатрии и неврологии. - 2014. - №2. - С. 45-52.

3. Эткинд А. М. Тест Роршаха и структура психического образа // Вопр. психологии. 1981 № 5. С. 106-115.

4. Бурлачук Л. Ф. Проблема исследования бессознательного психического проективными методами // Бессознательное: Природа, функции, методы исследования, Тбилиси, 1978. Т.3. С. 638-643.

5. Белый Б. И. Тест Роршаха. Практика и теория / Под ред. Л. Н. Собчик. — СПб.: ООО «Каскад», 2005. - 240 с.

6. Pramod Kumar POPULAR RESPONSES IN THE RORSCHACH TEST // Psychoiogia. - 1962. - №5. - С. 161-169.

7. Интегративная система Экснера // Child-psy URL: https://www.child-psy.ru/inkblots/422.html (дата обращения: 24.04.2023).

8. Чернильные пятна Роршаха: все ли в порядке с вашей психикой? // testometrika URL: https://testometrika.com/personality-and-temper/rorschach-test/ (дата обращения: 24.04.2023).

9. Тест Роршаха с множественным выбором, MCRT // Психологические тесты онлайн URL: https://psytests.org/classic/hemcr.html (дата обращения: 24.04.2023).

10. Тест Роршаха: пройди онлайн без смс и регистрации! // LML URL: http://lmlnews.ru/test-rorshaha/ (дата обращения: 24.04.2023).

11. Кольцов Д. М. Python. Полное руководство. - СПб: Издательство Наука и Техника, 2022. - 480 с.

12. Франсуа Шолле Глубокое обучение на Python. - СПб: Питер, 2022. - 400 с.

13. James R. Choca, PhD RORSCHACH PROTOCOL REPORT: 2017. - 16 с.

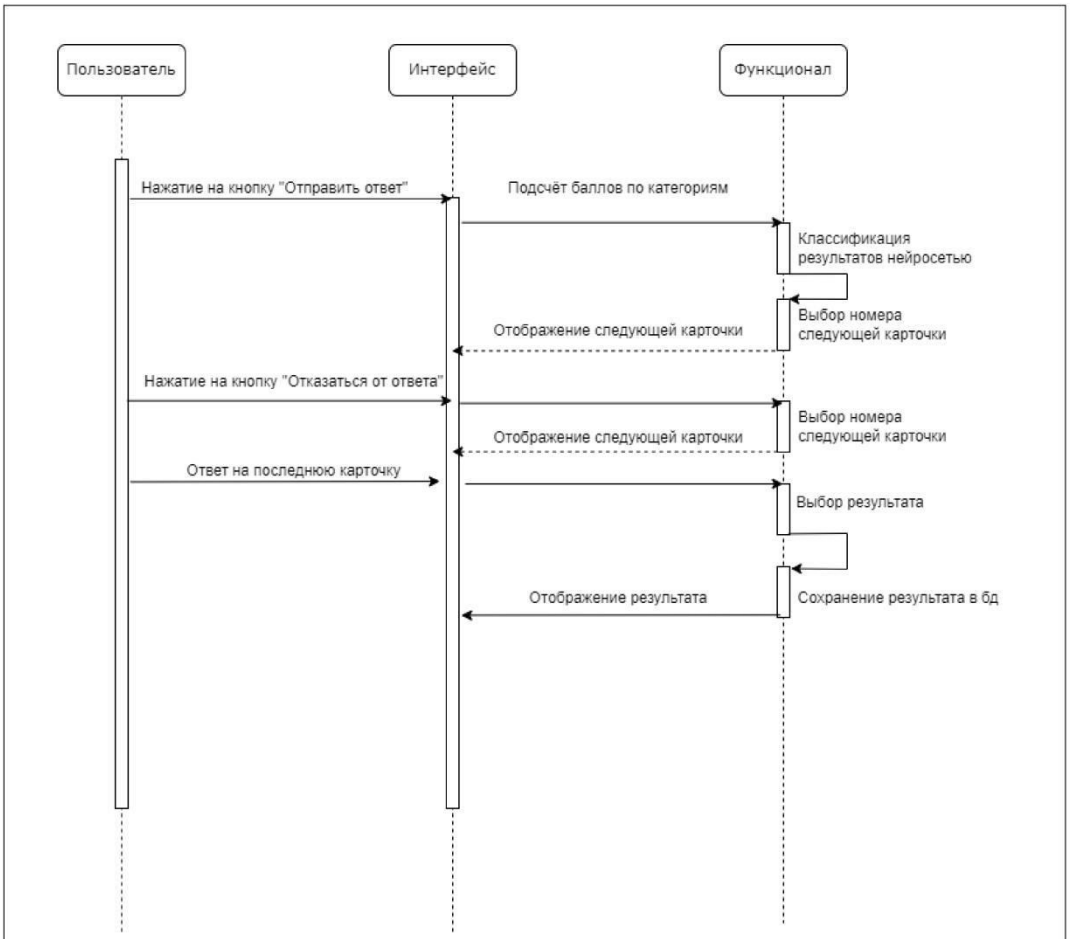
Приложение А

(Обязательное)

Средства для проведения психологического тестирования с использованием нейронной сети

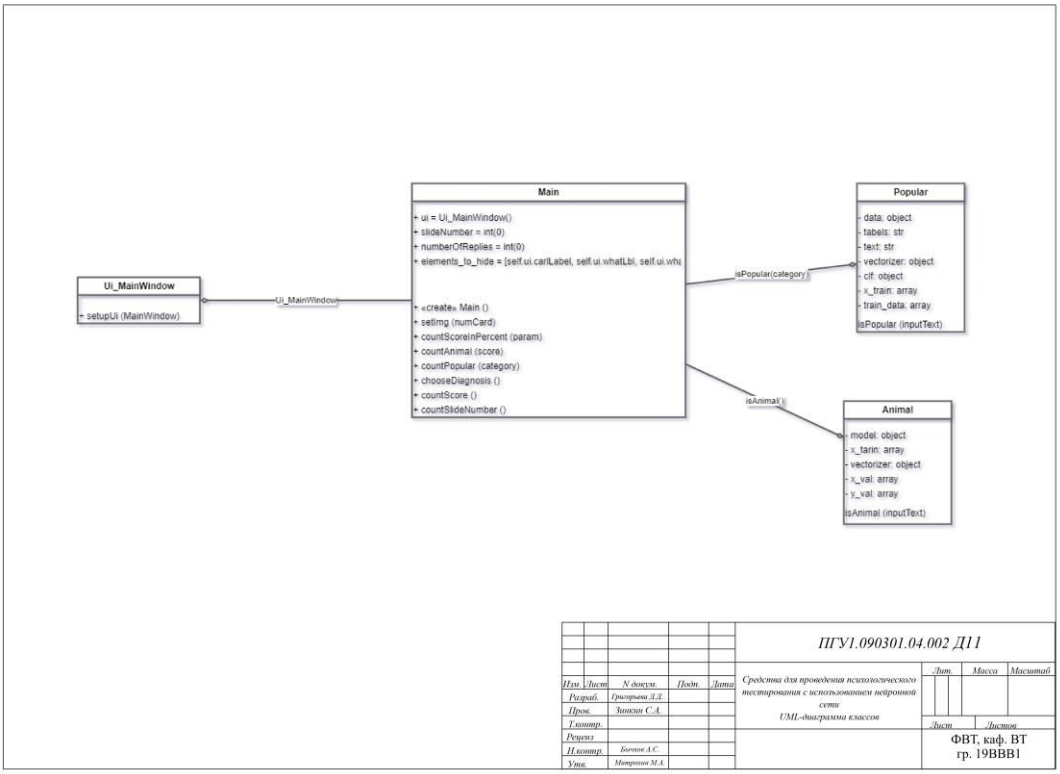
UML-диаграммы

Приложение А.1 - Диаграмма последовательностей



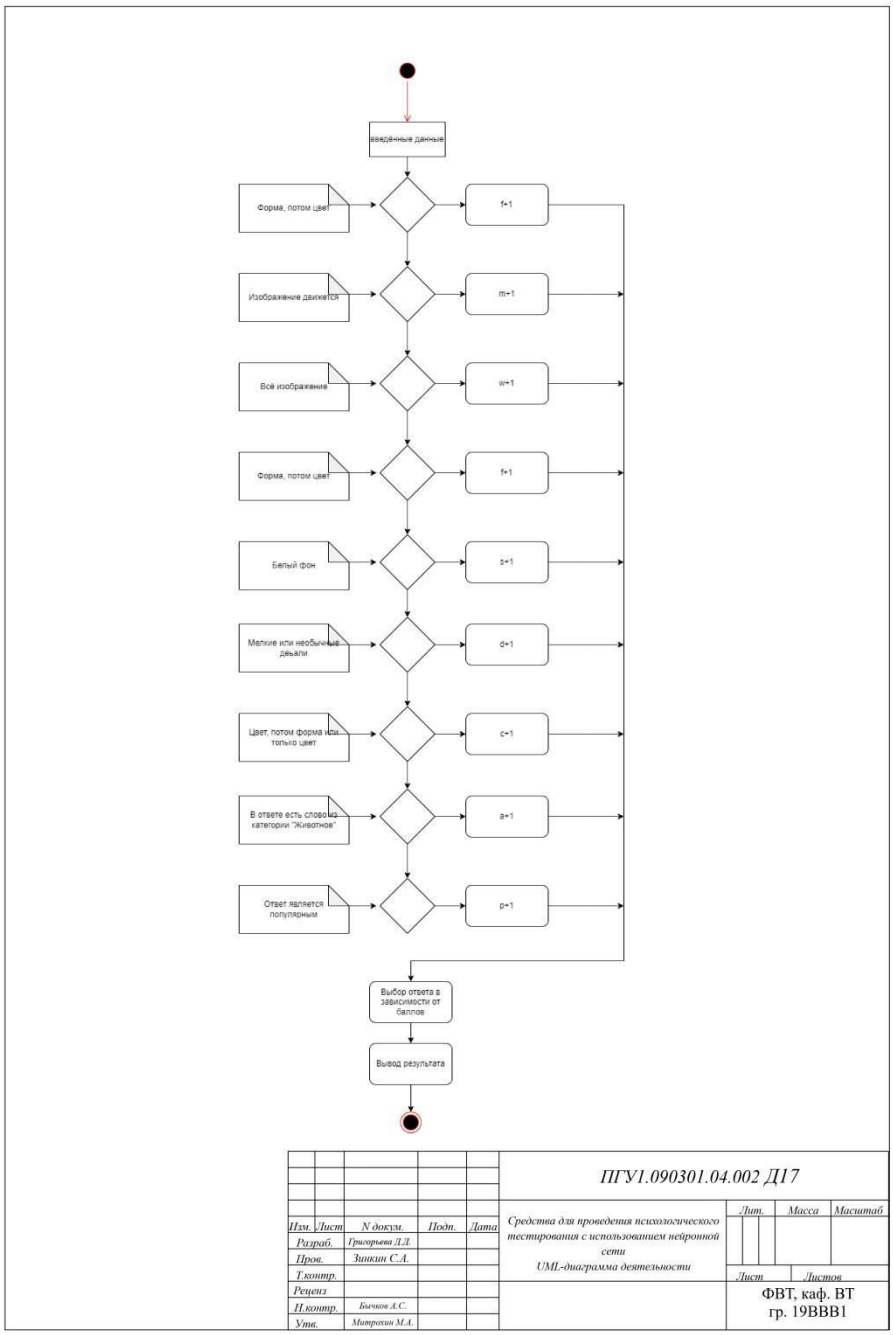
					ПГУ1.090301.04.002 Д14			
					Средства для проведения психологического тестирования с использованием нейронной сети UML-диаграмма последовательностей	Лит.	Масса	Масштаб
Изм.	Лист	N докум.	Подп.	Дата				
Разраб.		Григорьева Л.Д.				Лист	Листов	
Пров.		Зинкин С.А.						
Т.контр.						ФВТ, каф. ВТ гр. 19ВВВ1		
Реценз								
Н.контр.		Бычков А.С.						
Утв.		Митрохин М.А.						

Приложение А.2 - Диаграмма классов



						ПГУ1.090301.04.002 Д11			
Изм.	Лист	N докум.	Подп.	Дата	Средства для проведения психологического тестирования с использованием нейронной сети UML-диаграмма классов		Лит.	Масса	Масштаб
Разраб.		Григорьева Л.Д.							
Пров.		Зинкин С.А.					Лист	Листов	
Т.контр.									
Реценз.							ФВТ, каф. ВТ гр. 19ВВВ1		
Н.контр.		Бочков А.С.							
Утв.		Митрохин М.А.							

Приложение А.3 - Диаграмма деятельности



					ПГУ1.090301.04.002 Д17			
						Лит.	Масса	Масштаб
Изм.	Лист	N докум.	Подп.	Дата	Средства для проведения психологического тестирования с использованием нейронной сети UML-диаграмма деятельности			
Разраб.		Григорьева Д.Д.				Лист	Листов	
Пров.		Зинкин С.А.						
Т.контр.						ФВТ, каф. ВТ гр. 19ВВВ1		
Реценз								
Н.контр.		Бычков А.С.						
Утв.		Митрохин М.А.						

Приложение Б

(Обязательное)

Средства для проведения психологического тестирования с использованием нейронной сети

Листинг программы

Файл mainwindow.py

```
# -*- coding: utf-8 -*-
from PySide6.QtCore import (QCoreApplication, QDate, QDateTime,
QLocale,
    QMetaObject, QObject, QPoint, QRect,
    QSize, QTime, QUrl, Qt)
from PySide6.QtGui import (QBrush, QColor, QConicalGradient,
QCursor,
    QFont, QFontDatabase, QGradient, QIcon,
    QImage, QKeySequence, QLinearGradient, QPainter,
    QPalette, QPixmap, QRadialGradient, QTransform)
from PySide6.QtWidgets import (QApplication, QButtonGroup,
QGridLayout, QLabel,
    QLineEdit, QMainWindow, QPushButton, QRadioButton,
    QScrollArea, QSizePolicy, QSpacerItem, QTextEdit,
    QWidget)

class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        if not MainWindow.objectName():
            MainWindow.setObjectName(u"MainWindow")
        MainWindow.resize(850, 800)
        sizePolicy = QSizePolicy(QSizePolicy.Minimum,
QSizePolicy.Minimum)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(MainWindow.sizePolicy().hasHeightForW
idth())
        MainWindow.setSizePolicy(sizePolicy)
        MainWindow.setMinimumSize(QSize(850, 800))
        MainWindow.setMaximumSize(QSize(850, 800))
        MainWindow.setStyleSheet(u"")
        self.centralwidget = QWidget(MainWindow)
        self.centralwidget.setObjectName(u"centralwidget")
```

```
sizePolicy.setHeightForWidth(self.centralwidget.sizePolicy().hasHe
ightForWidth())
          self.centralwidget.setSizePolicy(sizePolicy)
          self.centralwidget.setStyleSheet(u"background-color:
#F2E6DF;\n"
    "font-family: Montserrat Alternates;\n"
    "QPushButton { border-radius: 25px;\n"
    "background-color: #D9C5C1; }\n"
    "QLabel { background: transparent }")
          self.scrollArea = QScrollArea(self.centralwidget)
          self.scrollArea.setObjectName(u"scrollArea")
          self.scrollArea.setEnabled(True)
          self.scrollArea.setGeometry(QRect(0, 0, 850, 800))
          sizePolicy1 = QSizePolicy(QSizePolicy.Preferred,
QSizePolicy.Preferred)
          sizePolicy1.setHorizontalStretch(0)
          sizePolicy1.setVerticalStretch(0)

sizePolicy1.setHeightForWidth(self.scrollArea.sizePolicy().hasHeig
htForWidth())
          self.scrollArea.setSizePolicy(sizePolicy1)
          self.scrollArea.setMinimumSize(QSize(810, 800))
          self.scrollArea.setLayoutDirection(Qt.LeftToRight)
          self.scrollArea.setWidgetResizable(False)
          self.scrollAreaWidgetContents = QWidget()

self.scrollAreaWidgetContents.setObjectName(u"scrollAreaWidgetCont
ents")
          self.scrollAreaWidgetContents.setEnabled(True)
          self.scrollAreaWidgetContents.setGeometry(QRect(0, 0,
830, 1300))

sizePolicy.setHeightForWidth(self.scrollAreaWidgetContents.sizePol
icy().hasHeightForWidth())
```

```
            self.scrollAreaWidgetContents.setSizePolicy(sizePolicy)
            self.scrollAreaWidgetContents.setMinimumSize(QSize(830,
1300))

self.scrollAreaWidgetContents.setCursor(QCursor(Qt.ArrowCursor))

self.scrollAreaWidgetContents.setLayoutDirection(Qt.LeftToRight)

self.scrollAreaWidgetContents.setStyleSheet(u"*{background-color:
#F2E6DF;\n"
    "font-family: Montserrat Alternates;}\n"
    "\n"
    "QPushButton {border-radius: 25px;\n"
    "background-color: #D9C5C1; }\n"
    "\n"
    "QPushButton:hover:enabled { \n"
    "background-color: white;\n"
    "border: 2px solid #D9C5C1; }\n"
    "\n"
    "QPushButton:enabled  { background-color: #D9C5C1;}\n"
    "\n"
    "QLabel { background: transparent; }\n"
    "QTextEdit, QLineEdit {background-color: #D9C5C1;\n"
    "color: #362121;\n"
    "padding-left: 10px;}\n"
    "\n"
    "QRadioButton { background: transparent; }\n"
    "QRadioButton::indicator {\n"
    "width: 11px; \n"
    "height: 11px; \n"
    "border-radius: 5px;\n"
    "background-color: #D9C5C1;\n"
    "}\n"
    "QRadioButton::indicator::checked{ \n"
    "border: 3px solid #F2E6DF;\n"
```

```
    "background-color: #362121;\n"
    "border-radius: 4px;\n"
    "width: 7px; \n"
    "height: 7px; \n"
    "}")
            self.gridLayoutWidget =
QWidget(self.scrollAreaWidgetContents)
            self.gridLayoutWidget.setObjectName(u"gridLayoutWidget")
            self.gridLayoutWidget.setGeometry(QRect(20, 20, 741,
1230))
            self.gridLayout = QGridLayout(self.gridLayoutWidget)
            self.gridLayout.setSpacing(6)
            self.gridLayout.setObjectName(u"gridLayout")
            self.gridLayout.setContentsMargins(30, 10, 10, 10)

            sizePolicy2 = QSizePolicy(QSizePolicy.Minimum,
QSizePolicy.Fixed)
            sizePolicy2.setHorizontalStretch(0)
            sizePolicy2.setVerticalStretch(0)

            # row 0
            self.testLabel = QLabel(self.gridLayoutWidget)
            self.testLabel.setObjectName(u"testLabel")
            self.testLabel.setMinimumSize(QSize(0, 0))
            self.testLabel.setStyleSheet(u"font-size: 40px;")
            self.gridLayout.addWidget(self.testLabel, 0, 0, 1, 1)

            self.nameInput = QTextEdit(self.gridLayoutWidget)
            self.nameInput.setObjectName(u"nameInput")

sizePolicy2.setHeightForWidth(self.nameInput.sizePolicy().hasHeigh
tForWidth())
            self.nameInput.setSizePolicy(sizePolicy2)
            self.nameInput.setMinimumSize(QSize(250, 30))
            self.nameInput.setText("Ваше имя: ")
```

```
            self.nameInput.setMaximumHeight(30)
            self.nameInput.setStyleSheet(u"border-radius: 5px;")
            self.gridLayout.addWidget(self.nameInput, 0, 1, 1, 1)

            # row 1 spacer
            self.horizontalSpacer = QSpacerItem(40, 20,
QSizePolicy.Expanding, QSizePolicy.Minimum)
            self.gridLayout.addItem(self.horizontalSpacer, 1, 0, 1,
2)

            # row 2
            self.cardImg = QLabel(self.gridLayoutWidget)
            self.cardImg.setObjectName(u"cardImg")
            sizePolicy3 = QSizePolicy(QSizePolicy.Maximum,
QSizePolicy.Maximum)
            sizePolicy3.setHorizontalStretch(0)
            sizePolicy3.setVerticalStretch(40)

sizePolicy3.setHeightForWidth(self.cardImg.sizePolicy().hasHeightF
orWidth())
            self.cardImg.setSizePolicy(sizePolicy3)
            self.cardImg.setMinimumSize(QSize(0, 0))
            self.cardImg.setMaximumSize(QSize(720, 400))
            self.cardImg.setLayoutDirection(Qt.LeftToRight)
            self.cardImg.setStyleSheet(u"background-color:  #D9C5C1;")
            self.gridLayout.addWidget(self.cardImg, 2, 0, 1, 2)

            # row 3
            self.carlLabel = QLabel(self.gridLayoutWidget)
            self.carlLabel.setObjectName(u"carlLabel")
            self.carlLabel.setMinimumSize(QSize(0, 0))
            font = QFont()
            font.setFamilies([u"Montserrat Alternates"])
            font.setItalic(True)
            self.carlLabel.setFont(font)
```

```
            self.carlLabel.setStyleSheet(u"")
            self.gridLayout.addWidget(self.carlLabel, 3, 0, 1, 2)

            # row 4 spacer
            self.horizontalSpacer_2 = QSpacerItem(40, 20,
QSizePolicy.Expanding, QSizePolicy.Minimum)
            self.gridLayout.addItem(self.horizontalSpacer_2, 4, 0, 1,
2)

            # row 5
            self.whatLbl = QLabel(self.gridLayoutWidget)
            self.whatLbl.setObjectName(u"whatLbl")
            self.whatLbl.setMinimumSize(QSize(0, 0))
            self.gridLayout.addWidget(self.whatLbl, 5, 0, 1, 2)

            # row 7
            self.whatInput = QTextEdit(self.gridLayoutWidget)
            self.whatInput.setObjectName(u"whatInput")
            sizePolicy4 = QSizePolicy(QSizePolicy.Maximum,
QSizePolicy.Minimum)
            sizePolicy4.setHorizontalStretch(0)
            sizePolicy4.setVerticalStretch(0)

sizePolicy4.setHeightForWidth(self.whatInput.sizePolicy().hasHeigh
tForWidth())
            self.whatInput.setSizePolicy(sizePolicy4)
            self.whatInput.setMinimumSize(QSize(0, 85))
            self.whatInput.setMaximumSize(QSize(750, 85))
            self.whatInput.setStyleSheet(u"border-radius: 10px;")
            self.gridLayout.addWidget(self.whatInput, 6, 0, 1, 2)

            # row 7 spacer
            self.horizontalSpacer_3 = QSpacerItem(40, 20,
QSizePolicy.Expanding, QSizePolicy.Minimum)
            self.gridLayout.addItem(self.horizontalSpacer_3, 7, 0, 1,
```

```
2)

            # row 8
            self.formLbl = QLabel(self.gridLayoutWidget)
            self.formLbl.setObjectName(u"formLbl")
            self.formLbl.setMinimumSize(QSize(0, 0))
            self.gridLayout.addWidget(self.formLbl, 8, 0, 1, 2)

            # row 9-10
            self.formYesRadioBtn =
QRadioButton(self.gridLayoutWidget)
            self.formYesRadioBtn.setObjectName(u"formYesRadioBtn")
            self.formYesRadioBtn.setMinimumSize(QSize(0, 0))
            self.gridLayout.addWidget(self.formYesRadioBtn, 9, 0, 1,
2)

            self.formNoRadioBtn = QRadioButton(self.gridLayoutWidget)
            self.formNoRadioBtn.setObjectName(u"formNoRadioBtn")
            self.formNoRadioBtn.setMinimumSize(QSize(0, 0))
            self.gridLayout.addWidget(self.formNoRadioBtn, 10, 0, 1,
2)

            self.formGroup = QButtonGroup(MainWindow)
            self.formGroup.setObjectName(u"formGroup")
            self.formGroup.addButton(self.formYesRadioBtn)
            self.formGroup.addButton(self.formNoRadioBtn)

            # row 11 spacer
            self.horizontalSpacer_4 = QSpacerItem(40, 20,
QSizePolicy.Expanding, QSizePolicy.Minimum)
            self.gridLayout.addItem(self.horizontalSpacer_4, 11, 0,
1, 2)

            # row 12
            self.actionLbl = QLabel(self.gridLayoutWidget)
```

```
            self.actionLbl.setObjectName(u"actionLbl")
            self.actionLbl.setMinimumSize(QSize(0, 0))
            self.gridLayout.addWidget(self.actionLbl, 12, 0, 1, 2)

            # row 13-14
            self.actionYesRadioBtn =
QRadioButton(self.gridLayoutWidget)

self.actionYesRadioBtn.setObjectName(u"actionYesRadioBtn")
            self.actionYesRadioBtn.setMinimumSize(QSize(0, 0))
            self.actionYesRadioBtn.setStyleSheet(u"")
            self.gridLayout.addWidget(self.actionYesRadioBtn, 13, 0,
1, 2)

            self.actionNoRadioBtn =
QRadioButton(self.gridLayoutWidget)
            self.actionNoRadioBtn.setObjectName(u"actionNoRadioBtn")
            self.actionNoRadioBtn.setMinimumSize(QSize(0, 0))
            self.gridLayout.addWidget(self.actionNoRadioBtn, 14, 0,
1, 2)

            self.actionGroup = QButtonGroup(MainWindow)
            self.actionGroup.setObjectName(u"actionGroup")
            self.actionGroup.addButton(self.actionYesRadioBtn)
            self.actionGroup.addButton(self.actionNoRadioBtn)

            # row 15 spacer
            self.horizontalSpacer_5 = QSpacerItem(40, 20,
QSizePolicy.Expanding, QSizePolicy.Minimum)
            self.gridLayout.addItem(self.horizontalSpacer_5, 15, 0,
1, 2)

            # row 16
            self.partLbl = QLabel(self.gridLayoutWidget)
            self.partLbl.setObjectName(u"partLbl")
```

```
            self.partLbl.setMinimumSize(QSize(0, 0))
            self.gridLayout.addWidget(self.partLbl, 16, 0, 1, 2)

            # row 17-21
            self.allPartRadioBtn =
QRadioButton(self.gridLayoutWidget)
            self.allPartRadioBtn.setObjectName(u"allPartRadioBtn")
            self.allPartRadioBtn.setMinimumSize(QSize(0, 0))
            self.gridLayout.addWidget(self.allPartRadioBtn, 17, 0, 1,
2)

            self.partPartRadioBtn =
QRadioButton(self.gridLayoutWidget)
            self.partPartRadioBtn.setObjectName(u"partPartRadioBtn")
            self.partPartRadioBtn.setMinimumSize(QSize(0, 0))
            self.gridLayout.addWidget(self.partPartRadioBtn, 18, 0,
1, 2)

            self.partSeparateRadioBtn =
QRadioButton(self.gridLayoutWidget)

self.partSeparateRadioBtn.setObjectName(u"partSeparateRadioBtn")
            self.partSeparateRadioBtn.setMinimumSize(QSize(0, 0))
            self.gridLayout.addWidget(self.partSeparateRadioBtn, 19,
0, 1, 2)

            self.partBackRadioBtn =
QRadioButton(self.gridLayoutWidget)
            self.partBackRadioBtn.setObjectName(u"partBackRadioBtn")
            self.partBackRadioBtn.setMinimumSize(QSize(0, 0))
            self.gridLayout.addWidget(self.partBackRadioBtn, 20, 0,
1, 2)

            self.partOtherRadioBtn =
QRadioButton(self.gridLayoutWidget)
```

```
self.partOtherRadioBtn.setObjectName(u"partOtherRadioBtn")
            self.partOtherRadioBtn.setMinimumSize(QSize(0, 0))
            self.gridLayout.addWidget(self.partOtherRadioBtn, 21, 0,
1, 2)

            self.partGroup = QButtonGroup(MainWindow)
            self.partGroup.setObjectName(u"partGroup")
            self.partGroup.addButton(self.allPartRadioBtn)
            self.partGroup.addButton(self.partPartRadioBtn)
            self.partGroup.addButton(self.partSeparateRadioBtn)
            self.partGroup.addButton(self.partBackRadioBtn)
            self.partGroup.addButton(self.partOtherRadioBtn)

            # row 22 spacer
            self.horizontalSpacer_6 = QSpacerItem(40, 20,
QSizePolicy.Expanding, QSizePolicy.Minimum)
            self.gridLayout.addItem(self.horizontalSpacer_6, 22, 0,
1, 2)

            # row 23
            self.signLbl = QLabel(self.gridLayoutWidget)
            self.signLbl.setObjectName(u"signLbl")
            self.signLbl.setMinimumSize(QSize(0, 0))
            self.gridLayout.addWidget(self.signLbl, 23, 0, 1, 2)

            # row 24 - 26
            self.signFormColorRadioBtn =
QRadioButton(self.gridLayoutWidget)

self.signFormColorRadioBtn.setObjectName(u"signFormColorRadioBtn")
            self.signFormColorRadioBtn.setMinimumSize(QSize(0, 0))
            self.gridLayout.addWidget(self.signFormColorRadioBtn,  24,
0, 1, 2)
```

```
            self.signColorFormRadioBtn =
QRadioButton(self.gridLayoutWidget)

self.signColorFormRadioBtn.setObjectName(u"signColorFormRadioBtn")
            self.signColorFormRadioBtn.setMinimumSize(QSize(0, 0))
            self.gridLayout.addWidget(self.signColorFormRadioBtn,  25,
0, 1, 2)

            self.signColoRadioBtn =
QRadioButton(self.gridLayoutWidget)
            self.signColoRadioBtn.setObjectName(u"signColoRadioBtn")
            self.signColoRadioBtn.setMinimumSize(QSize(0, 0))
            self.gridLayout.addWidget(self.signColoRadioBtn, 26, 0,
1, 2)

            self.signGroup = QButtonGroup(MainWindow)
            self.signGroup.setObjectName(u"signGroup")
            self.signGroup.addButton(self.signFormColorRadioBtn)
            self.signGroup.addButton(self.signColorFormRadioBtn)
            self.signGroup.addButton(self.signColoRadioBtn)

            # row 27 spacer
            self.horizontalSpacer_7 = QSpacerItem(40, 20,
QSizePolicy.Expanding, QSizePolicy.Minimum)
            self.gridLayout.addItem(self.horizontalSpacer_7, 27, 0,
1, 2)

            # row 28
            self.rejectionBtn = QPushButton(self.gridLayoutWidget)
            self.rejectionBtn.setObjectName(u"rejectionBtn")
            self.rejectionBtn.setMinimumSize(QSize(150, 50))

self.rejectionBtn.setCursor(QCursor(Qt.PointingHandCursor))
            self.rejectionBtn.setStyleSheet(u"")
            self.rejectionBtn.setCheckable(True)
```

```
sizePolicy2.setHeightForWidth(self.rejectionBtn.sizePolicy().hasHe
ightForWidth())
            self.rejectionBtn.setSizePolicy(sizePolicy2)
            self.gridLayout.addWidget(self.rejectionBtn, 28, 0, 1, 1)

            self.sendBtn = QPushButton(self.gridLayoutWidget)
            self.sendBtn.setObjectName(u"sendBtn")

sizePolicy2.setHeightForWidth(self.sendBtn.sizePolicy().hasHeightF
orWidth())
            self.sendBtn.setSizePolicy(sizePolicy2)
            self.sendBtn.setMinimumSize(QSize(150, 50))
            self.sendBtn.setCursor(QCursor(Qt.PointingHandCursor))
            self.sendBtn.setStyleSheet(u"")
            self.sendBtn.setCheckable(True)
            self.gridLayout.addWidget(self.sendBtn, 28, 1, 1, 1)

            self.scrollArea.setWidget(self.scrollAreaWidgetContents)
            MainWindow.setCentralWidget(self.centralwidget)
            self.retranslateUi(MainWindow)

            QMetaObject.connectSlotsByName(MainWindow)
        # setupUi

        def retranslateUi(self, MainWindow):

MainWindow.setWindowTitle(QCoreApplication.translate("MainWindow",
u"\u0422\u0435\u0441\u0442
\u0420\u043e\u0440\u0448\u0430\u0445\u0430", None))

self.signColorFormRadioBtn.setText(QCoreApplication.translate("Mai
nWindow", u"\u0412\u0430\u0436\u043d\u0435\u0435
\u0446\u0432\u0435\u0442, \u0444\u043e\u0440\u043c\u0430
\u0432\u0442\u043e\u0440\u0438\u0447\u043d\u0430", None))
```

```
self.formYesRadioBtn.setText(QCoreApplication.translate("MainWindo
w", u"\u0414\u0430", None))

self.partSeparateRadioBtn.setText(QCoreApplication.translate("Main
Window", u"\u041e\u0442\u0434\u0435\u043b\u044c\u043d\u044b\u0435
\u043f\u044f\u0442\u043d\u0430 \u0438\u043b\u0438
\u043c\u0435\u043b\u043a\u0438\u0435
\u0434\u0435\u0442\u0430\u043b\u0438", None))

self.rejectionBtn.setText(QCoreApplication.translate("MainWindow",
u"\u041e\u0442\u043a\u0430\u0437\u0430\u0442\u044c\u0441\u044f
\u043e\u0442 \u043e\u0442\u0432\u0435\u0442\u0430", None))

self.allPartRadioBtn.setText(QCoreApplication.translate("MainWindo
w", u"\u0412\u0441\u0451
\u0438\u0437\u043e\u0431\u0440\u0430\u0436\u0435\u043d\u0438\u0435
\u0438\u043b\u0438 \u0431\u043e\u043b\u044c\u0448\u0430\u044f
\u0435\u0433\u043e \u0447\u0430\u0441\u0442\u044c", None))

self.testLabel.setText(QCoreApplication.translate("MainWindow",
u"\u0422\u0435\u0441\u0442
\u0420\u043e\u0440\u0448\u0430\u0445\u0430", None))

self.formNoRadioBtn.setText(QCoreApplication.translate("MainWindow
", u"\u041d\u0435\u0442", None))
          self.cardImg.setText("")

self.actionNoRadioBtn.setText(QCoreApplication.translate("MainWind
ow", u"\u041d\u0435\u0442", None))

self.partOtherRadioBtn.setText(QCoreApplication.translate("MainWin
dow", u"\u0414\u0440\u0443\u0433\u043e\u0435", None))

self.signColoRadioBtn.setText(QCoreApplication.translate("MainWind
```

```
ow", u"\u0412\u0430\u0436\u0435\u043d
\u0442\u043e\u043b\u044c\u043a\u043e \u0446\u0432\u0435\u0442",
None))
    #if QT_CONFIG(whatsthis)

self.carlLabel.setWhatsThis(QCoreApplication.translate("MainWindow
", u"<html><head/><body><p
align=\"center\"><br/></p></body></html>", None))
    #endif // QT_CONFIG(whatsthis)

self.carlLabel.setText(QCoreApplication.translate("MainWindow",
u"\u0422\u0430\u0431\u043b\u0438\u0446\u0430 1 \u0438\u0437 10",
None))

self.sendBtn.setText(QCoreApplication.translate("MainWindow",
u"\u041e\u0442\u043f\u0440\u0430\u0432\u0438\u0442\u044c
\u043e\u0442\u0432\u0435\u0442", None))

self.partBackRadioBtn.setText(QCoreApplication.translate("MainWind
ow", u"\u0411\u0435\u043b\u044b\u0439 \u0444\u043e\u043d", None))

self.signLbl.setText(QCoreApplication.translate("MainWindow", u"5.
\u041d\u0430 \u0432\u0430\u0448 \u043e\u0442\u0432\u0435\u0442
\u043f\u043e\u0432\u043b\u0438\u044f\u043b\u0438
\u0446\u0432\u0435\u0442 \u0438\u043b\u0438
\u0444\u043e\u0440\u043c\u0430?", None))

self.formLbl.setText(QCoreApplication.translate("MainWindow", u"2.
\u0423\u0432\u0438\u0434\u0435\u043d\u043d\u043e\u0435
\u0438\u043c\u0435\u0435\u0442
\u0447\u0451\u0442\u043a\u0443\u044e
\u0444\u043e\u0440\u043c\u0443?", None))

self.partLbl.setText(QCoreApplication.translate("MainWindow", u"4.
\u0412 \u043a\u0430\u043a\u043e\u0439
```

```
\u0447\u0430\u0441\u0442\u0438
\u0442\u0430\u0431\u043b\u0438\u0446\u044b \u0412\u044b
\u0443\u0432\u0438\u0434\u0435\u043b\u0438
\u043e\u0431\u0440\u0430\u0437?", None))

self.actionLbl.setText(QCoreApplication.translate("MainWindow",
u"3.
\u0418\u0437\u043e\u0431\u0440\u0430\u0436\u0435\u043d\u0438\u0435
\u0434\u0432\u0438\u0436\u0435\u0442\u0441\u044f
\u0438\u043b\u0438
\u0438\u0441\u043a\u0430\u0436\u0430\u0435\u0442\u0441\u044f?",
None))

self.actionYesRadioBtn.setText(QCoreApplication.translate("MainWin
dow", u"\u0414\u0430", None))

self.whatLbl.setText(QCoreApplication.translate("MainWindow", u"1.
\u0427\u0442\u043e \u0412\u044b
\u0432\u0438\u0434\u0438\u0442\u0435 \u043d\u0430
\u043a\u0430\u0440\u0442\u0438\u043d\u043a\u0435?", None))

self.partPartRadioBtn.setText(QCoreApplication.translate("MainWind
ow", u"\u0427\u0430\u0441\u0442\u044c
\u0431\u043e\u043b\u044c\u0448\u043e\u0433\u043e
\u043f\u044f\u0442\u043d\u0430", None))

self.signFormColorRadioBtn.setText(QCoreApplication.translate("Mai
nWindow", u"\u0412\u0430\u0436\u043d\u0435\u0435
\u0444\u043e\u0440\u043c\u0430, \u0446\u0432\u0435\u0442
\u0432\u0442\u043e\u0440\u0438\u0447\u0435\u043d", None))
    # retranslateUi
```

Файл main.py

```
import csv
```

```
    from PySide6.QtCore import Slot
    from PySide6.QtGui import QPixmap
    from PySide6.QtWidgets import QApplication, QMainWindow
    import sys

    from neural.animal import Animal
    from neural.popular import Popular
    from ui.MainWindow import Ui_MainWindow


    class Main(QMainWindow):
        def __init__(self):
            super(Main, self).__init__()
            self.ui = Ui_MainWindow()
            self.ui.setupUi(self)
            self.ui.sendBtn.clicked.connect(self.countSlideNumber)
            self.ui.sendBtn.clicked.connect(self.countScore)

self.ui.rejectionBtn.clicked.connect(self.countSlideNumber)
            self.ui.cardImg.setPixmap(QPixmap('cards/1.jpg'))

            self.slideNumber = int(0)
            self.p, self.a, self.f, self.m, self.w, self.s, self.d,
self.c = 0, 0, 0, 0, 0, 0, 0, 0
            self.numberOfReplies = int(0)
            self.elements_to_hide = [self.ui.carlLabel,
self.ui.whatLbl, self.ui.whatInput, self.ui.formLbl,
self.ui.formYesRadioBtn, self.ui.formNoRadioBtn,
                                     self.ui.actionLbl,
self.ui.actionYesRadioBtn, self.ui.actionNoRadioBtn,
self.ui.partLbl, self.ui.allPartRadioBtn, self.ui.partPartRadioBtn,
                                     self.ui.partSeparateRadioBtn,
self.ui.partBackRadioBtn, self.ui.partOtherRadioBtn,
self.ui.signLbl, self.ui.signFormColorRadioBtn,
```

```
                            self.ui.signColorFormRadioBtn,
self.ui.signColoRadioBtn, self.ui.rejectionBtn, self.ui.sendBtn]

        # Загрузка изображения по порядковому номеру
        def setImg(self, numCard):
            pixmap =
QPixmap(('cards/{numCard}.jpg').format(numCard=numCard))
            self.ui.cardImg.setPixmap(pixmap)

        # Подсчёт ответа по категории в процентах
        def countScoreInPercent(self, param):
            return param / self.numberOfReplies * 100

        # Подсчёт количества ответов по животным
        def countAnimal(self, score):
            if score == 1:
                self.a += 1
            return self.a

        # Подсчёт количества популярных ответов
        def countPopular(self, category):
            if (category != "другое"):
                match self.slideNumber:
                    case 0:
                        if (category == "летучая мышь" or "бабочка" or
"птица" or "человек"):
                            self.p += 1
                    case 1:
                        if (category == "медведь" or "слон" or
"собака" or "человек"):
                            self.p += 1
                    case 2:
                        if (category == "человек"):
                            self.p += 1
                    case 3:
```

```
                    if (category == "летучая мышь" or "бабочка"):
                        self.p += 1
                case 4:
                    if (category == "летучая мышь" or "бабочка" or
"птица"):
                        self.p += 1
                case 5:
                    if (category == "черепаха"):
                        self.p += 1
                case 6:
                    if (category == "облако"):
                        self.p += 1
                case 7:
                    if (category == "собака" or "ящерица" or
"тигр"):
                        self.p += 1
                case 8:
                    if (category == "голова человека"):
                        self.p += 1
                case 9:
                    if (category == "краб" or "осьминог" or "паук"
or "собака" or "кролик" or "заяц" or "лев"):
                        self.p += 1
            return self.p

        # Соответствие ответа критериям, постановка преддиагноза
        def chooseDiagnosis(self):
            scoreP = self.countScoreInPercent(self.p)
            scoreA = self.countScoreInPercent(self.a)
            scoreF = self.countScoreInPercent(self.f)
            scoreM = self.countScoreInPercent(self.m)
            scoreW = self.countScoreInPercent(self.w)
            scoreS = self.countScoreInPercent(self.s)
            scoreD = self.countScoreInPercent(self.d)
            scoreC = self.countScoreInPercent(self.c)
```

```
            if (scoreP>90 and 30>scoreA>35 and 70>scoreF>80 and
scoreM<10 and 65<scoreW<70 and 14<scoreC<18):
                self.diagnosis = "Здоровый тестируемый"
            elif (18>scoreP>22 and 58>scoreA>62 and scoreF<50 and
5>scoreF>9 and 25>scoreW>29 and 4>scoreS>7 and 56>scoreD>60 and
5>scoreC>9):
                self.diagnosis = "Больные с алкогольным делирием"
            elif (65>scoreF>69 and 8>scoreM>12 and 40>scoreW>44 and
5>scoreD>9 and 14>scoreC>18):
                self.diagnosis = "Больные с остбредовыми формами
шизофрении"
            elif (27>scoreP>31 and 48>scoreA>52 and 68>scoreF>72 and
8>scoreM>12 and 26>scoreW>30 and 1>scoreD>5 and 3>scoreC>7):
                self.diagnosis = "Атеросклероз"
            elif (scoreP>25 and scoreF>70 and scoreM>10 and
scoreC<90):
                self.diagnosis = "Органические поражения головного
мозга"
            else:
                self.diagnosis = "Ваши показатели не подходят к
категориям, которые тестирует данное приложение"
            # Сохранение результата в базу данных
            with open('result.csv', mode='a', newline='') as file:
                writer = csv.writer(file)
                writer.writerow([self.ui.nameInput.toPlainText(),
self.diagnosis])
            return "Предварительный диагноз: "+self.diagnosis

        @Slot()
        def countScore(self):
            self.numberOfReplies += 1
            what_result = self.ui.whatInput.toPlainText()
            self.p =
self.countPopular(Popular.isPopular(what_result))
```

```
            self.a = self.countAnimal(Animal.isAnimal(what_result))

            if self.ui.formYesRadioBtn.isChecked(): #f (ответы с
чёткой формой)
                self.f += 1
            if self.ui.actionYesRadioBtn.isChecked():   #m (ответы по
движению)
                self.m += 1
            if self.ui.allPartRadioBtn.isChecked(): #w (целостные
ответы)
                self.w += 1
            if self.ui.partBackRadioBtn.isChecked(): #s (ответы на
белый фон)
                self.s += 1
            if self.ui.partSeparateRadioBtn.isChecked(): #d (ответы
на мелкие детали)
                self.d += 1
            if self.ui.signColorFormRadioBtn.isChecked() or
self.ui.signColoRadioBtn.isChecked(): #c (цветоформовые ответы и
первичные по цвету)
                self.c += 1


        @Slot()
        def countSlideNumber(self):
            self.slideNumber += 1
            match self.slideNumber:
                case 0:
                    self.setImg(1)
                case 1:
                    self.setImg(2)
                case 2:
                    self.setImg(3)
                case 3:
                    self.setImg(4)
```

```
            case 4:
                self.setImg(5)
            case 5:
                self.setImg(6)
            case 6:
                self.setImg(7)
            case 7:
                self.setImg(8)
            case 8:
                self.setImg(9)
            case 9:
                self.setImg(10)
            case _:
                self.ui.cardImg.setText(self.chooseDiagnosis())
                for i in self.elements_to_hide:
                    i.hide()

if __name__ == "__main__":
    app = QApplication(sys.argv)
    window = Main()
    window.show()

    sys.exit(app.exec())
```

Файл popular.py

```
import pandas as pd
from sklearn.naive_bayes import MultinomialNB
from sklearn.feature_extraction.text import CountVectorizer

class Popular():
    """наивный байес, категоризация популярного ответа"""
    global data, labels, texts, vectorizer, clf, x_train
    global train_data
    data= pd.read_csv('neural/popular.csv', sep=';',
```

```
names=['label', 'text'], encoding='cp1251')    # Dataset по
категориям
        train_data = pd.read_csv('neural/test-popular.csv', sep=';',
names=['label', 'text'],
                                encoding='cp1251')  # Dataset по
категориям

        labels = data['label']
        texts = data['text']
        # Инициализация векторизатора
        vectorizer = CountVectorizer()
        # Преобразование текста в числовые векторы
        x_train = vectorizer.fit_transform(texts)
        # Инициализация наивного байеса
        clf = MultinomialNB(alpha=0.1)
        # Обучение
        clf.fit(x_train, labels)

        @staticmethod
        def isPopular(inputTxt):
            # Определение категории введённого текста
            x_test = vectorizer.transform([inputTxt])
            result = clf.predict(x_test)[0]
            predicted_probabilities = clf.predict_proba(x_test)[0]

            if max(predicted_probabilities) > 0.2:
                return result
            return "другое"
```

Файл animal.py

```
import pandas as pd
from keras.models import Sequential
from keras.layers import Dense
from sklearn.feature_extraction.text import CountVectorizer
```

```
    import numpy as np
    import matplotlib.pyplot as plt
    from sklearn.utils import shuffle

    class Animal():
        """нейронная сеть, бинарная классификация животное/ не
животное"""
        global model, x_train, y_train, x_val, y_val, vectorizer
        # Загрузка датасета
        dataframe = pd.read_csv("neural/animal-names.csv",
names=['text', 'label'], sep=',', header=None, nrows=2000)
        dataframe = shuffle(dataframe)

        # Инициализация векторизатора
        vectorizer = CountVectorizer(binary=True)

        # Преобразование данных в бинарную матрицу
        x_train =
vectorizer.fit_transform(dataframe['text']).toarray().astype('floa
t32')
        y_train = np.asarray(dataframe['label']).astype('float32')
        # Разделение данных на тренинровочные и проверочные
        x_val = x_train[-1000:]
        y_val = y_train[-1000:]
        x_train = x_train[:-1000]
        y_train = y_train[:-1000]

        # Создание модели
        model = Sequential()
        model.add(Dense(16, activation='relu'))
        model.add(Dense(16, activation='relu'))
        model.add(Dense(1, activation='sigmoid'))
        # Компиляция модели
        model.compile(loss='binary_crossentropy',
optimizer='rmsprop', metrics=['acc'])
```

```
        # Обучение модели
        result = model.fit(x_train, y_train, epochs=24,
batch_size=64, validation_data=(x_val, y_val))

        @staticmethod
        def isAnimal(inputTxt):
            new_data = pd.Series([str(inputTxt)])
            new_binary_matrix =
vectorizer.transform(new_data).toarray()
            prediction = model.predict(new_binary_matrix)
            result_label = np.where(prediction > 0.5, 1, 0)
            if result_label == [[0]]:
                return 0
            elif result_label == [[1]]:
                return 1

        # result_dict = result.history
        # loss = result_dict['loss']
        # val_loss = result_dict['val_loss']
        # epochs = range(1, len(result_dict['acc']) + 1)
        #
        # plt.plot(epochs, loss, 'bo', label='Потери при обучении')
        # plt.plot(epochs, val_loss, 'b', label='Потери при
проверке')
        # plt.xlabel('Эпохи')
        # plt.ylabel('Потери')
        # plt.legend()
        # plt.show()
        #
        # plt.clf()
        # acc = result_dict['acc']
        # val_acc = result_dict['val_acc']
        #
        # plt.plot(epochs, acc, 'bo', label='Точность при обучении')
```

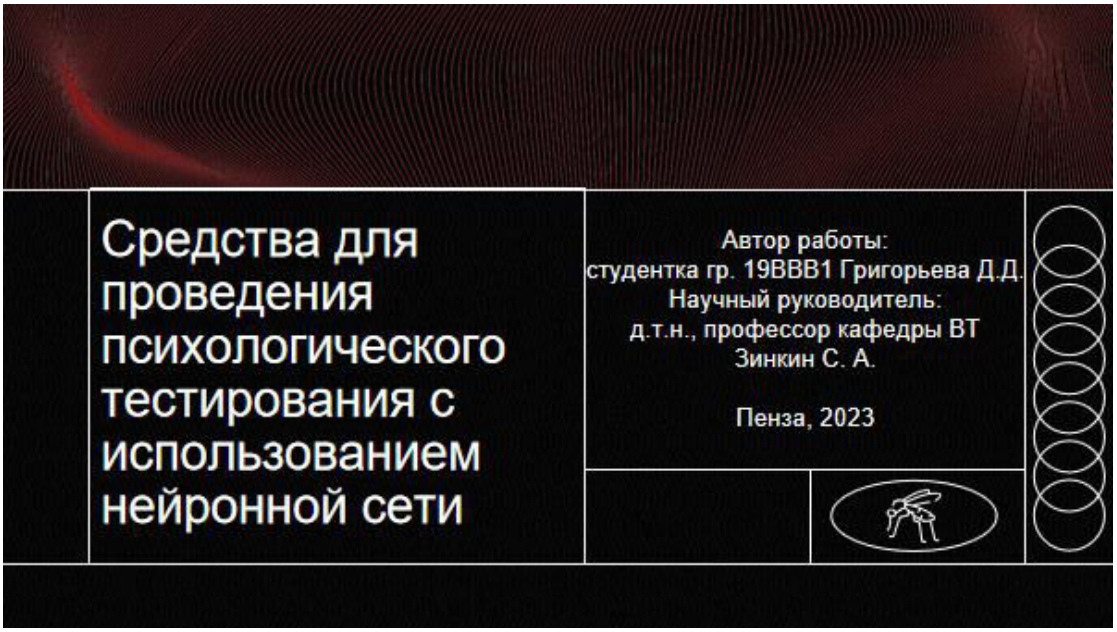
```
        # plt.plot(epochs, val_acc, 'b', label='Точность при
проверке')
        # plt.xlabel('Эпохи')
        # plt.ylabel('Потери')
        # plt.legend()
        # plt.show()
```

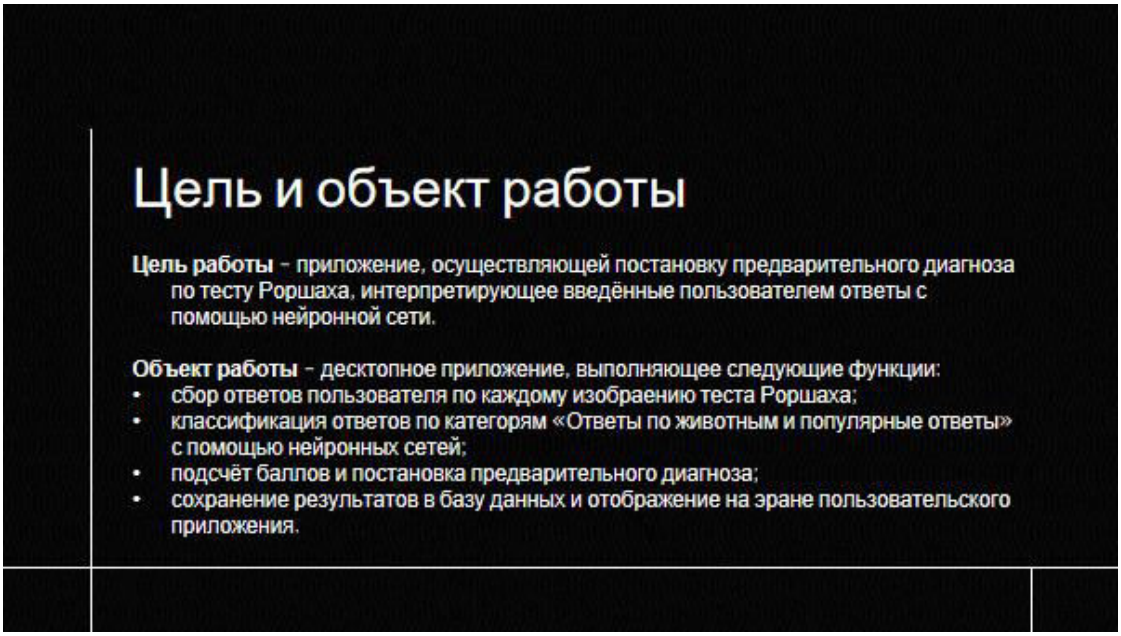
Приложение В

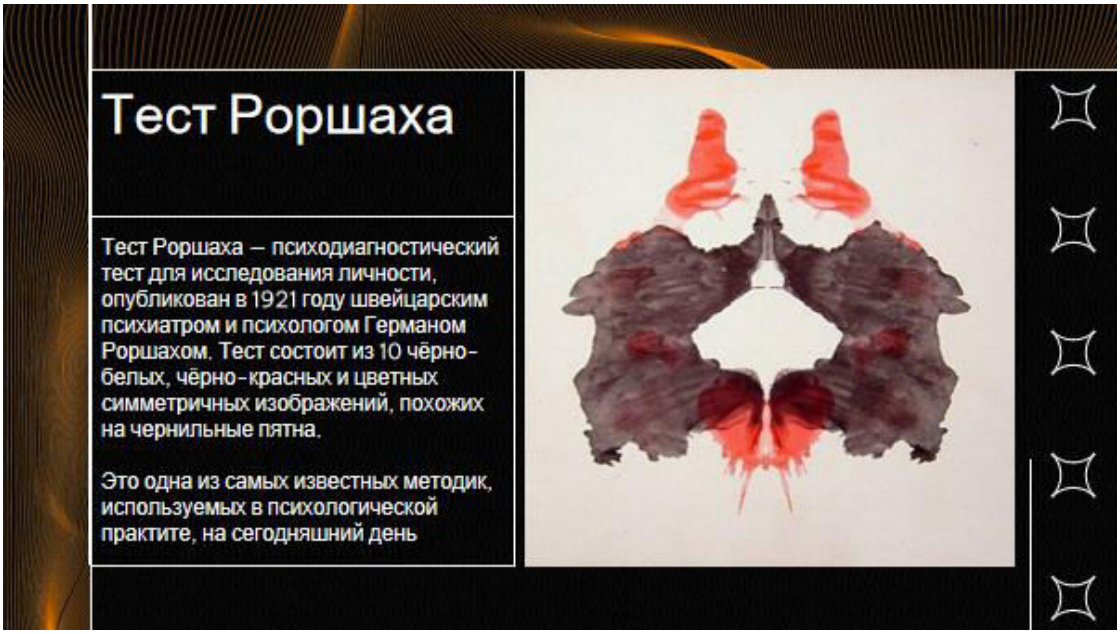
(Обязательное)

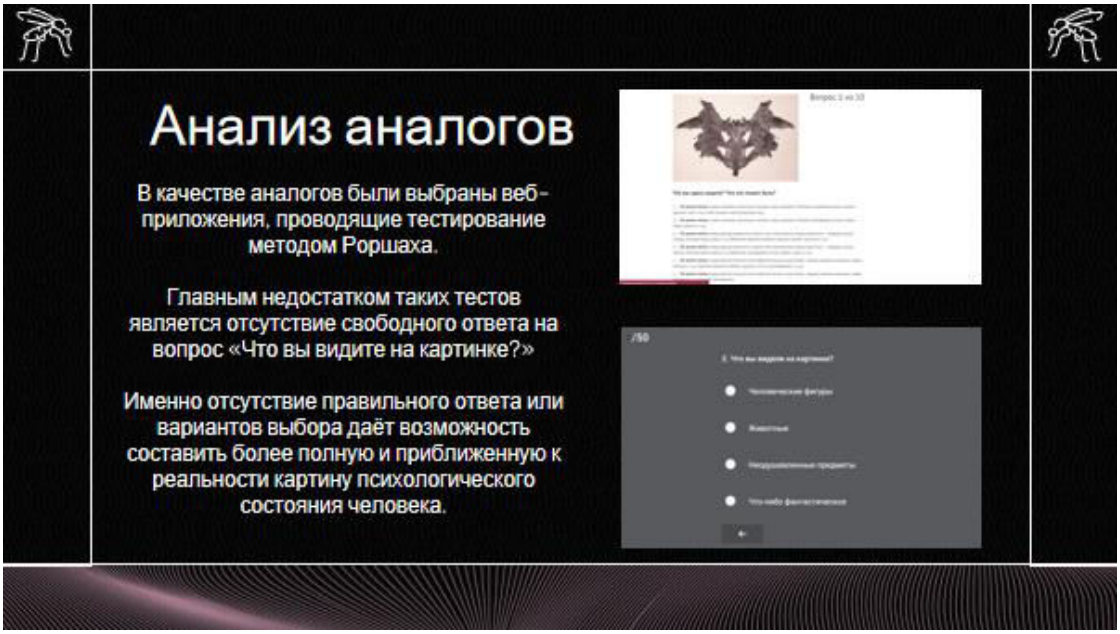
Средства для проведения психологического тестирования с использованием нейронной сети

Презентация


Средства для проведения психологического тестирования с использованием нейронной сети
Автор работы:
студентка гр. 19ВВВ1 Григорьева Д.Д.
Научный руководитель:
д.т.н., профессор кафедры ВТ
Зинкин С. А.
Пенза, 2023

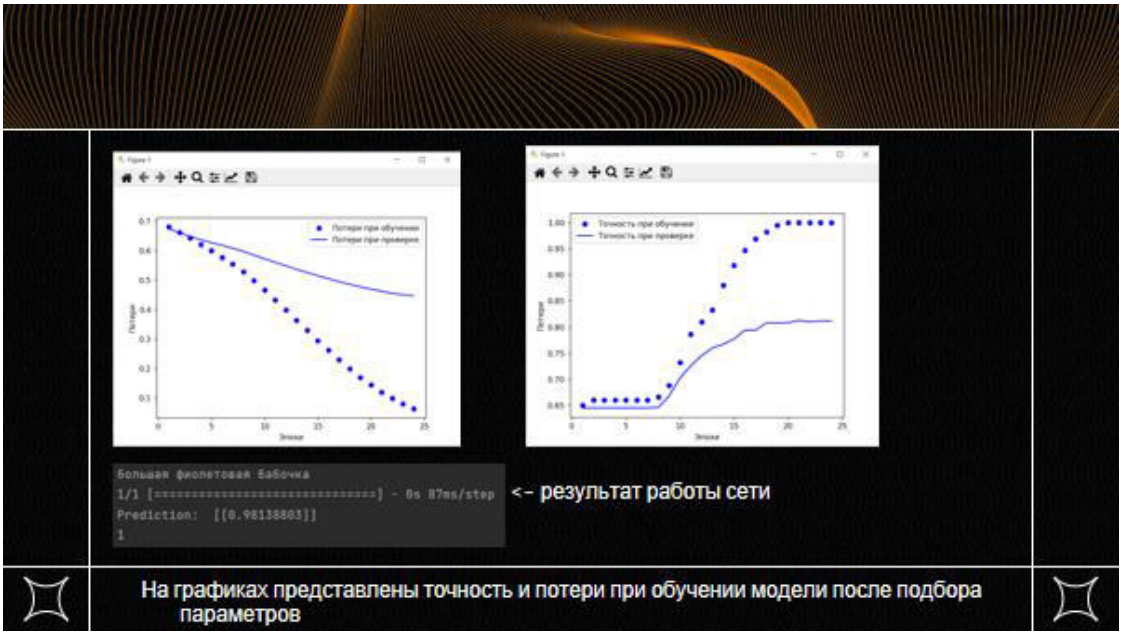

Цель и объект работы
Цель работы – приложение, осуществляющей постановку предварительного диагноза по тесту Роршаха, интерпретирующее введённые пользователем ответы с помощью нейронной сети.
Объект работы – десктопное приложение, выполняющее следующие функции:
• сбор ответов пользователя по каждому изобраению теста Роршаха;
• классификация ответов по категорям «Ответы по животным и популярные ответы» с помощью нейронных сетей;
• подсчёт баллов и постановка предварительного диагноза;
• сохранение результатов в базу данных и отображение на эране пользовательского приложения.


Тест Роршаха
Тест Роршаха — психодиагностический тест для исследования личности, опубликован в 1921 году швейцарским психиатром и психологом Германом Роршахом. Тест состоит из 10 чёрно-белых, чёрно-красных и цветных симметричных изображений, похожих на чернильные пятна.
Это одна из самых известных методик, используемых в психологической практите, на сегодняшний день


Анализ аналогов
В качестве аналогов были выбраны веб-приложения, проводящие тестирование методом Роршаха.
Главным недостатком таких тестов является отсутствие свободного ответа на вопрос «Что вы видите на картинке?»
Именно отсутствие правильного ответа или вариантов выбора даёт возможность составить более полную и приближенную к реальности картину психологического состояния человека.

Используемые технологии
В качетве языка разработки выбран Python 3.10
PySide6
Библиотека для создания графического интерфейса
Библиотека машинного обучения
Keras
Фреймворк глубокого обучения
TensorFlow
Библиотека машинного обучения
pandas
Библиотека для работы с данными
NumPy
Библиотека для математических вычислений

Создание нейронной сети, выполняющей бинарную классификацию
Для обучения модели был создан датасет формата csv
На графиках представлены точность и потери при обучении модели до подбора параметров


Потери при обучении
Потери при проверке
Точность при обучении
Точность при проверке
Большая фиолетовая бабочка
1/1 [==============================] - 0s 87ms/step
Prediction: [[0.98138803]]
1
<- результат работы сети
На графиках представлены точность и потери при обучении модели после подбора параметров

Создание наивного Байесовского классификатора
$p(C|F1,\ldots,Fn) = \frac{p(C)p(F1,\ldots Fn|C)}{p(F1,\ldots,Fn)}$
где C - класс,
Fn - его признаки

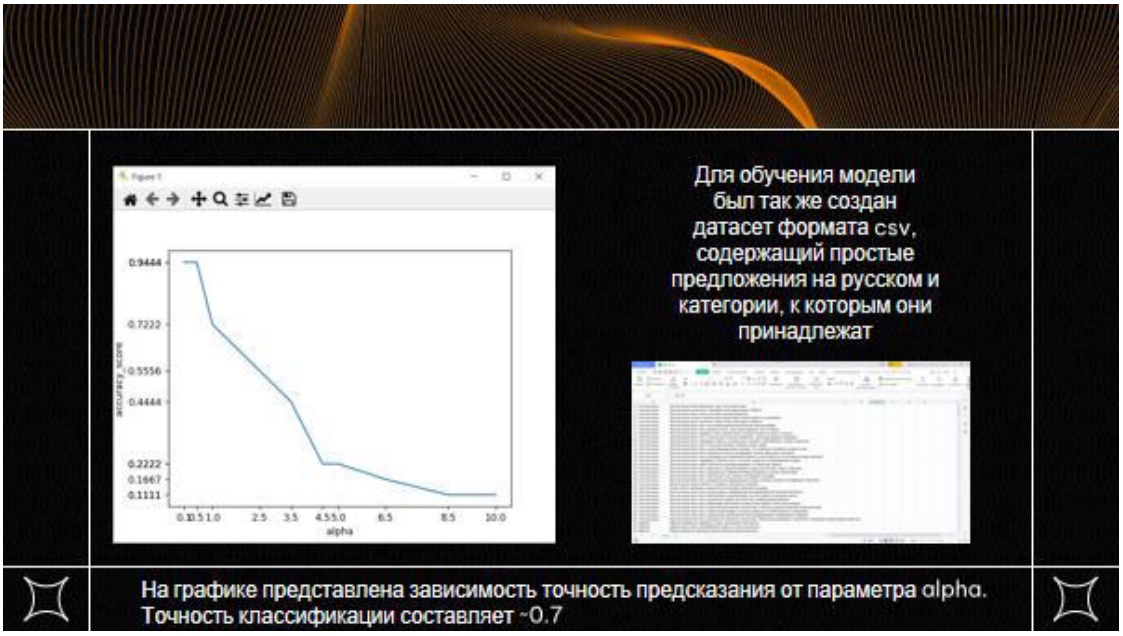
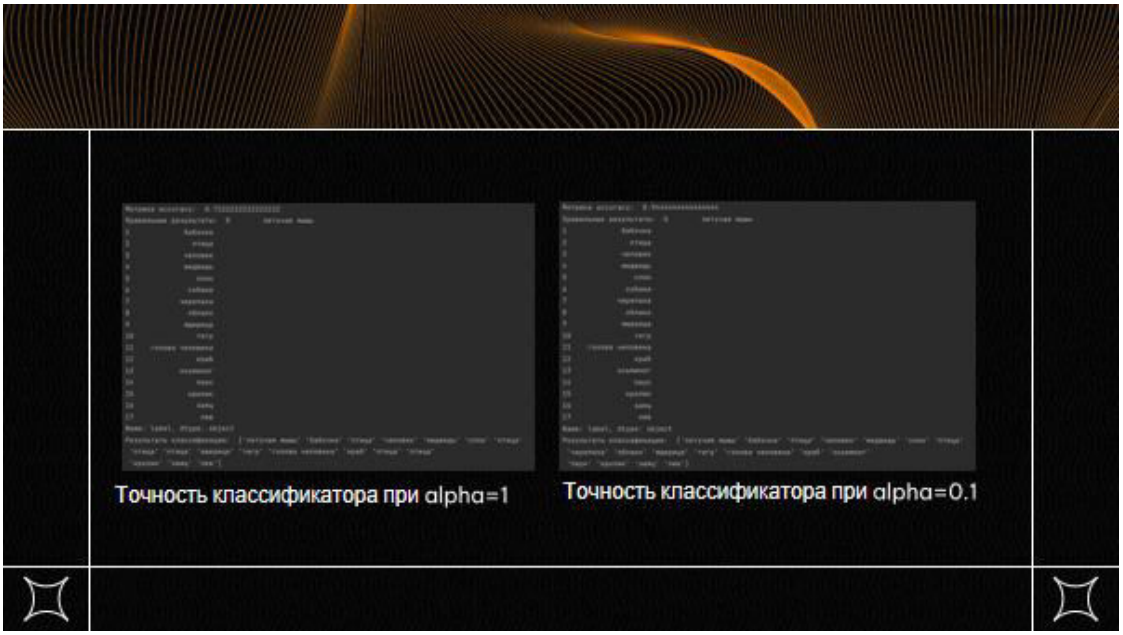


Figure 1
0.9444
0.7222
0.5556
0.4444
0.2222
0.1667
0.1111
accuracy_score
0.1 0.5 1.0 2.5 3.5 4.5 5.0 6.5 8.5 10.0
alpha
Для обучения модели был так же создан датасет формата csv, содержащий простые предложения на русском и категории, к которым они принадлежат
На графике представлена зависимость точность предсказания от параметра alpha.
Точность классификации составляет ~0.7


Точность классификатора при alpha=1
Точность классификатора при alpha=0.1

Интерпретация результатов

Данное приложение ставит предварительный диагноз по следующим заболеваниям:

- алкогольный делирий;
- острая бредовая форма шизофрении;
- атеросклероз;
- органические поражения головного мозга;
- здоровый тестируемый.

Тестируемые с одинаковыми заболеваниями часто дают одинаковые ответы на некоторые категории вопросов, используя эту закономерность и исходя из полученных результатов, ставится предварительный диагноз.

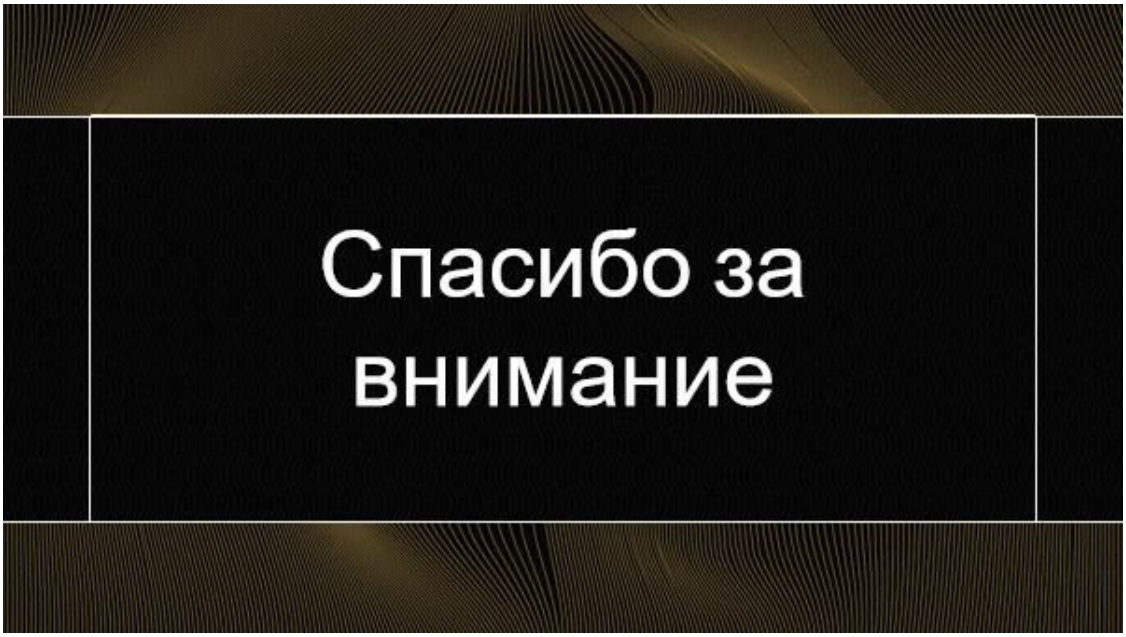
Создание графического интерфейса приложения

Для создания прототипа приложения было выбрано приложение Figma, для реализации – фреймворк PySide6

Релизованный графический интерфейс

Было реализовано десктопное приложение с применением фреймворка Pyside6, осуществляющее тестирование и постановку предварительного диагноза по тесту Роршаха с интерпретацией введённых пользователем ответов с помощью наивного Байсовского классификатора, реализованного на основе библиотеки skit-learn и нейронной сети, осуществляющей бинарную классификацию на основе библиотеки keras. Для обучения нейронных сетей были созданы датасеты в формате csv, загрузка которых осуществлялась с помощью библиотеки pandas.

Результаты работы


Спасибо за
внимание