|  |
| --- |
|  |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА − Российский технологический университет»**  **РТУ МИРЭА** |
| Институт кибербезопасности и цифровых технологий |
| Цифровая кафедра |

|  |
| --- |
| **РАБОТА ДОПУЩЕНА К ЗАЩИТЕ** |
| Заведующий программой  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Ш.Г. Магомедов |
| «31» мая 2024 г. |

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| по программе | |  |  |  |
| **«Программные средства решения прикладных задач искусственного интеллекта»** | | | | |
| на тему: | Модель распознавания вариант 43 | | | |
|  |  | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Обучающийся | \_\_\_\_\_\_\_\_\_\_\_ *подпись* | Бахарев Евгений Владимирович  *Фамилия, имя, отчество* | |
| Руководитель  работы | \_\_\_\_\_\_\_\_\_\_\_ *подпись* | \_\_\_\_\_\_\_\_\_\_\_\_\_\_  *ученая степень, должность* | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ *Фамилия, имя, отчество* |

**Москва 2024 г.**

|  |
| --- |
| ÐÐ°ÑÑÐ¸Ð½ÐºÐ¸ Ð¿Ð¾ Ð·Ð°Ð¿ÑÐ¾ÑÑ Ð±ÐµÐ»ÑÐ¹ ÑÐ¾Ð½ |
| МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  РОССИЙСКОЙ ФЕДЕРАЦИИ |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** |
| **Институт кибербезопасности и цифровых технологий** |
| Цифровая кафедра |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | СОГЛАСОВАНО | | | |
|  | | Заведующий программой  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  *(подпись)* | | | |
|  | **Ш.Г. Магомедов** | | | |
|  | «27» | | мая | 2024 г. |

**ЗАДАНИЕ**

**на выполнение выпускной квалификационной работы**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Обучающийся | **Бахарев Евгений Владимирович** | | | |
|  |  | | | |
| Направление программы | **«Программные средства решения прикладных задач искусственного интеллекта»** | | |
|  |  |  |  | | |

**1. Тема выпускной квалификационной работы**

|  |
| --- |
| Модель распознавания вариант 43 |

**2. Цель и задачи выпускной квалификационной работы**

|  |  |  |
| --- | --- | --- |
| Цель работы: | | Получение практических навыков в решении прикладных задач ИИ |
|  |  | |
| Задачи работы: | | Разработать исследовательскую модель нейронной сети по классификации типов операций на основе данных пациентов. |

**3. Этапы выпускной квалификационной работы**

|  |  |  |  |
| --- | --- | --- | --- |
| №  этапа | Содержание этапа выпускной квалификационной работы | Результат выполнения этапа ВКР | Срок выполнения |
| 1. | **Постановка прикладной задачи** |  | — |
| 2. | **Алгоритм решения поставленной задачи** |  | — |
| 3. | **Реализация поставленной задачи** |  | — |

**4. Перечень разрабатываемых документов и графических материалов**

Цель и задачи ВКР, характеристика предметной области, метод и алгоритм решения задачи, апробация (моделирование или программное обеспечение), выводы.

|  |  |
| --- | --- |
|  | Задание принял к исполнению |
|  | Обучающийся: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  | *подпись* |
|  | «27» мая 2024 г. |

**ОГЛАВЛЕНИЕ**

[**ВВЕДЕНИЕ** 4](#_Toc168953463)

[**ГЛАВА 1. ИСТОРИОГРАФИЯ ТЕМЫ РАБОТЫ.** 6](#_Toc168953464)

[**ГЛАВА 2. ОБЩИЕ СВЕДЕНИЯ** 7](#_Toc168953465)

[**2.1. Грыжи и виды операций.** 7](#_Toc168953466)

[**2.2. Описание данных.** 8](#_Toc168953467)

[**2.3. Графики данных.** 11](#_Toc168953468)

[**2.3.4. Возраст.** 11](#_Toc168953469)

[**2.3.5. Половой признак.** 12](#_Toc168953470)

[**2.3.6. Приоритеты операции.** 12](#_Toc168953471)

[**2.3.7. Тип грыжи.** 12](#_Toc168953472)

[**2.3.8. Тип паховой грыжи.** 12](#_Toc168953473)

[**2.3.9. Метод проведения операции.** 12](#_Toc168953474)

[**ГЛАВА 3. НАПИСАНИЕ НЕЙРОННОЙ СЕТИ.** 13](#_Toc168953475)

[**3.1. Предобработка данных.** 13](#_Toc168953476)

[**3.2. Векторизация слов.** 13](#_Toc168953477)

[**3.3. Написание и обучение нейронной сети.** 14](#_Toc168953478)

[**ГЛАВА 4. АНАЛИЗ РЕЗУЛЬТАТОВ.** 18](#_Toc168953479)

[**4.1. Анализ точности.** 18](#_Toc168953480)

[**4.2. Построение графиков.** 18](#_Toc168953481)

[**ЗАКЛЮЧЕНИЕ** 19](#_Toc168953482)

[**ВЫВОД** 20](#_Toc168953483)

[**БИБЛИОГРАФИЧЕСКИЙ СПИСОК** 21](#_Toc168953484)

[**ПРИЛОЖЕНИЯ** 22](#_Toc168953485)

# **ВВЕДЕНИЕ**

**Актуальность.** С незапамятных времён люди создавали всё более качественные инструменты, помогающие понять окружающий мир и управлять им. Одними из ключевых таких инструментов, созданных за последние 100 лет, считаются радио, компьютер, интернет. Особый вклад внесло изобретение компьютера. Компьютеры позволяют производить сложные математические вычисления, хранить данные, обрабатывать их. Однако человеку требуется много времени на обработку данных. Именно поэтому были придуманы алгоритмы, выполняющие рутинную работу за людей, а вместе с тем появилась новая ветвь развития технологий – искусственный интеллект.

В настоящее время, искусственный интеллект широко используется практически во всех видах деятельности, начиная от интернета и сайтов и заканчивая медициной и производством.

В связи с вышеизложенным проведена исследовательская практическая работа, в ходе которой была написана нейронная сеть, прогнозирующая тип операций по устранению брюшной грыжи на основе данных о пациентах.

**Цель работы: -** написать нейронную сеть, прогнозирующую тип операции.

**Задачи работы:**

1. Изучить и обработать данные об пациентах.
2. Подготовить архитектуру нейронной сети.
3. Создать нейронную сеть.
4. Проанализировать результат

**Гипотеза исследования.** Нейронная сеть может быть использована в госпиталях и больницах в качестве помощника при выборе типа операции.

**Практическая значимость работы.** Полученные данные могут подтвердить или опровергнуть выдвинутую гипотезу об использовании нейронной сети в медицинских учреждениях.

# **ГЛАВА 1. ИСТОРИОГРАФИЯ ТЕМЫ РАБОТЫ.**

**История нейронных сетей.** С момента начала эпохи искусственного интеллекта, в 1960 году Френком Розенблаттом была создана первая модель нейронной сети, которая называлась персептрон. Исследователи говорили, что в будущем их мыслящая машина научится читать и писать, распознавать людей и называть их по именам, мгновенно переводить речь с одного языка на другой.

Персептрон Фрэнка Розенблатта стал первой попыткой создать нейронную сеть, способную обучаться на данных. Эта однослойная модель имела ограничения в решении сложных задач, но она подразумевала возможность создания более глубоких архитектур в будущем.

В 1980 году японским учёным в области искусственных нейронных сетей был изобретён неокогнитрон – модель нейронной сети, использующая свёрточные слои. Основная идея свёрточного слоя состоит в том, чтобы вместо одного большого, плотного, линейного слоя, связывающего каждый вход с каждым выходом, использовать в каждой позиции на входе множество очень маленьких линейных слоев. Нейронные сети с свёрточными слоями широко используются в обработке фото и видео.

С появлением интернета, веб-сервисов, текстовых редакторов появилась необходимость обработки текстовой информации. Так, в 1982 году была предложена первая рекуррентная нейронная сеть. Рекуррентная нейронная сеть – вид нейронных сетей, где связи между элементами образуют направленную последовательность. Благодаря этому открытию, появилась возможность обрабатывать тексты, не теряя при этом «смысл». Нейронные сети научились понимать человеческую речь.

Современные нейронные сети умеют выполнять широкий спектр задач. Однако прогресс не стоит на месте, и есть много задач и проблем в области нейронных сетей, которые нужно решить.

# **ГЛАВА 2. ОБЩИЕ СВЕДЕНИЯ**

## **2.1. Грыжи и виды операций.**

Грыжа – это выпячивание органов брюшной полости через мышечный слой передней брюшной стенки (см. приложение 1). Грыжи могут возникнуть двумя способами:

1. Это некие врожденные изменения в соединительной ткани людей, способные привести к тому, что в месте соединения апоневроза и мышц, частей апоневроза и т.д. – происходит его истончение и возникает грыжа.

2. В результате операций на органах брюшной полости – апоневроз в мете ушивания истончается, деформируется рубцовой тканью и в этом месте появляется грыжевой дефект.

Соответственно задача хирурга в хирургическом лечении – это закрыть дефект в апоневрозе и вправить «выпавшее» содержимое в брюшную полость.

Если это выполнять с помощью местных тканей, то происходит уменьшение поверхности брюшной стенки и тем самым увеличивается давление брюшной стенки на внутренние органы, что ведет к повышению внутрибрюшного давления. Таким образом, на швы появляется дополнительная нагрузка и далее или появляются осложнения со стороны внутренних «сдавленных» органов (кишки, желудка и т.д.) или швы рвутся, и грыжа появляется снова (рецидивирует).

Для решения этой проблемы придуманы сетчатые имплантаты – дополнительные структуры из полипропилена, полиэстера и др., которые как «заплатки» закрывают дефекты в апоневрозе.

Однако существует масса способов установки этих сетчатых имплантатов. Самый первый из них был, как самая настоящая «заплатка» - сверху в подкожную клетчатку. Достаточно простой способ, который, однако, дал множество осложнений, связанных с необходимостью отделения слоя подкожной клетчатки и кожи от апоневроза с дальнейшим нарушением кровоснабжения. В таких местах часто развивались жидкостные скопления (серомы) или нагноения. Также пациент чаще всего чувствовал в себе сетчатый имплантат.

Поэтому следующим шагом стали установки сетчатых имплантатов под мышцы, чтоб они работали как клапаны – при повышении внутрибрюшного давления останавливали выход содержимого из брюшной полости.

Однако, при больших грыжевых дефектах затруднительно замещать большой процент передней брюшной стенки сеткой – это опять же чревато осложнениями и снижением качества жизни пациента.

В рамках описанного можно выделить ряд проблем:

1. Как выбрать сетчатый имплантат для конкретной грыжи?

2. Какую грыжу какой методикой оперировать

3. Как прогнозировать возможные осложнения у пациента? Можно ли всех пациентов классифицировать по каким-либо признакам?

Для решения проблем было принято решение написать нейронную сеть, прогнозирующую тип операции. Выбранный тип архитектуры – персептрон (см. приложение 2). Это универсальный тип, позволяющий обрабатывать большие объёмы данных и прогнозировать результат с высокой точностью.

## **2.2. Описание данных.**

В качестве обучающих и тестовых данных – небольшая статистика, файл excel, пациентов из хирургического отделения. Данные о пациентах собирались в течение двух лет.

Файл имеет следующие столбцы:

* gender – пол человека: str

m – мужской

f - женский

* age – возраст: int
* first\_diagnoz –диагноз 1: str
* secons\_diagnoz – диагноз 2: str
* complication – осложнения: str
* anam – дополнение(описание) от врача: str
* emerg – тип операции: str

P – плановая

E – экстренная

* type – тип паховой грыжи: str

M – прямая

L – косая

F – бедренная

Mc – комбинированная

* side – сторона грыжи: str

RIGHT – правая

LEFT – левая

BOTH – обе стороны

* size – размер: int
* M1 - Наличие М1 локализации послеоперационной грыжи передней брюшной полости: bool

1 – есть

0 – нет (в случае отсутствия также 0)

* М2 - Наличие М2 локализации послеоперационной грыжи передней брюшной полости: bool
* М3 - Наличие М3 локализации послеоперационной грыжи передней брюшной полости: bool
* М4 - Наличие М4 локализации послеоперационной грыжи передней брюшной полости: bool
* М5 - Наличие М5 локализации послеоперационной грыжи передней брюшной полости: bool
* L1 - Наличие L1 локализации послеоперационной грыжи передней брюшной полости: bool
* L2 - Наличие L2 локализации послеоперационной грыжи передней брюшной полости: bool
* L3 - Наличие L4 локализации послеоперационной грыжи передней брюшной полости: bool
* L4 - Наличие L4 локализации послеоперационной грыжи передней брюшной полости: bool
* nums\_of\_recidiv – количество рецидивов: int
* Diastasis – наличие диастаза: bool
* Ptosis – степень ПТОЗа: int
* YABG\_and\_DPK – заболевание желудка – наличие язвенной болезни желудка и двенадцатиперстной кишки: bool
* HHC – заболевание желудка – наличие хронического гастрита: bool
* Hpankr – наличие панкреатита: bool
* DGR – наличие дуоденогастрального рефлюкса: bool
* Reflux-ezofagit – рефлюкс-эзофагит: bool
* Gastrit – гастрит: bool
* HGD – хронический гатродуоденит: bool
* GKB – желчнокаменная болезнь: bool
* AG/GB – гипертония: bool
* HSN – хроническая сердечная недостаточность: bool
* IBS – ишемическая болезнь сердца: bool
* Fibrilyaciya\_predserdi – фибрилляция предсердий: bool
* Anemiya – анемия: bool
* HZV – хроническое заболевание вен: bool
* Vtoraya\_grizha\_bez\_operaci – вторая грыжа без операции: bool
* grizha – наличие грыжи: bool
* HOBL – хроническая обструктивная болезнь лёгких: bool
* BA – бронхиальная астма: bool
* Hronichecki\_bronhit – хронический бронхит: bool
* SD – сахарный диабет: bool
* Zabolevaniya\_SZH – болезнь щитовидной железы: bool
* CBV – цереброваскулярные болезни: bool
* Epilepsiya – эпилепсия: bool
* DE – дисциркуляторная энцефалопатия: bool
* Zabolevaniya\_ODA – заболевания опорно-двигательного аппарата: bool
* Cirroz – цирроз печени: bool
* Virusni\_gepatit – вирусный гепатит: bool
* Zabolevaniya\_MVS – заболевания моче-выделительной системы: bool
* Hprost – простатит: bool
* DGPZH – доброкачественная гиперплазия предстательной железы: bool
* Ozhirenie – ожирение: int

0 = нет

1 = 1 степень

1. = 2 степень

3 = 3 степень

* Kollagenozi – коллагенозы: bool
* Onkologiya – онкология: bool
* SPON – синдром полиорганной недостаточности: bool
* result – тип применённой операции: str

## **2.3. Графики данных.**

**2.3.4. Возраст.** Одним из основных факторов принятия решения о выборе типа операции играет возраст пациента.Для определения закона распределения данных требуется построить графики. Так, были построены столбчатая диаграмма (см. приложение 3), гистограммы при разных интервалах (10, 5, 13, 15) возрастов (см. приложение 4-7) и графики данных о возрастах пациентов. Как можно заметить, линия оранжевого цвета, соединяющая вершины каждого столбца, схожа с линией графика нормального распределения (см. приложение 8). Можно сделать вывод о том, что пациенты распределены по нормальному закону. Таким образом, большая часть обратившихся в больницу людей находится в интервале от 55 до 70 лет.

**2.3.5. Половой признак.** Была построена столбчатая диаграмма половых признаков пациентов с паховыми грыжами, по результатам которой видно, что мужчины обращаются в поликлинику чаще, чем женщины (см. приложение 9).

**2.3.6. Приоритеты операции.** Была построена столбчатая диаграмма приоритетов операций (см. приложение 10). Как видно, экстренные операции проводятся реже, чем плановые, однако, доля экстренных от общего числа операций составляет почти 30%.

**2.3.7. Тип грыжи.** Исходя из данных столбчатой диаграммы типов грыж, видно, что большее число пациентов обратилось с грыжей правой стороны, а с грыжами обеих сторон почти не было обращений (см. приложение 11).

**2.3.8. Тип паховой грыжи.** Была построена столбчатая диаграмма на основе данных о типах паховых грыж (см. приложение 12). Как видно, преобладающее число пациентов обратилось в поликлинику с паховой грыжей косого типа.

**2.3.9. Метод проведения операции.** По итогу, всем пациентам была проведена операция по лечению паховой грыжи. На основе этих данных была построена столбчатая диаграмма выбранных методов операций (см. приложение 13).

# **ГЛАВА 3. НАПИСАНИЕ НЕЙРОННОЙ СЕТИ.**

**3.1. Предобработка данных.** Обработку данных, написание нейронной сети и анализ её работы будем выполнять в инструменте Jupyter Notebook.Для нейронной сети требуется уже не информация о пациентах, а готовый для обучения датасет. Преобразование информации в датасет заключается в удалении пустых столбцов и строк, заменой пустых ячеек на нулевые, преобразовании названий методов операций в векторный вид, приведении строковых значений некоторых параметров в цифровой вид, нормировании данных. Эта процедура представлена в приложениях 14-15.

**3.2. Векторизация слов.** Неотъемлемой частью данных о пациентах считается диагноз лечащего врача. Для того, чтобы информацию о диагнозах можно было использовать в обучении нейронной сети, её нужно обработать. Используем метод «мешка» слов. Этим методом довольно просто пользоваться, однако он не обеспечивает высокую точность прогнозирования. Для его использования необходимо векторизовать слова. Векторизация слов – представление строкового типа данных, к примеру, предложений, в числовом виде. Для понимания всего процесса разберём простой пример. У нас есть предложение: «Кошка всё ещё сидит на балконе». Каждому уникальному слову из предложения присваивается свой индекс, так, у слова «Кошка» будет индекс 0, у слова «всё» - 1, у слова «ещё» - 2 и т.д. После этого создаётся нулевой вектор, длина которого равна количеству уникальных слов предложения, значение 1 присваивается тем индексам вектора, слова которых присутствуют в предложении. У нашего предложения будет вектор [1 1 1 1 1 1]. У предложения «Кошка сидит на балконе» вектор будет [1 0 0 1 1 1], а у «Кошка сидит» - [1 0 0 1 0 0]. Этим методом преобразуем каждый диагноз в вектор. Для каждого столбца построим свою собственную матрицу. Отберём все уникальные слова из всех столбцов диагнозов, осложнений и дополнений, присвоим каждому свой индекс и для каждого диагноза создадим собственный вектор (см. приложения 16-19).

**3.3. Написание и обучение нейронной сети.** Цель нашей работы состоит в том, чтобы каждому пациенту выбрать наиболее подходящий метод операции. По сути, нам предстоит решить задачу классификации.

Классификация – это разделение множества объектов на классы по некоторым наиболее существенным признакам.

В области нейронных сетей используются следующие определения:

Входной слой – это готовые для обучения данные, представляющие из себя вектор чисел, описывающий тот или иной объект. Входной слой для нашей нейронной сети – вектор чисел, представляющий из себя данные об одном пациенте.

Скрытый слой – это матрица чисел, полученная путём математических операций. На вход скрытому слою подаётся входной слой, этот слой умножается на матрицу весов, после к произведению применяется функция активации, результат функции активации выходит из скрытого слоя.

Выходной слой – это вектор с информацией, которую мы ожидаем получить. На вход выходному слою подаётся выход скрытого слоя, этот выход умножается на матрицу весов, к произведению применяется функция активации, результат этой функции будет считаться результатом выходного слоя.

Функция активации – это математическая функция, которая применяется к нейронам в слое в процессе прогнозирования. Не каждая математическая функция может быть функцией активации, для этого функция должна быть:

* Непрерывной и бесконечной на всей области определения
* Монотонной и не меняющей направления
* Нелинейной
* Простой в вычислении

К стандартным функциям активации относят relu (кусочно-линейная функция), tanh (гиперболический тангенс), sigmoid (сигмоида), softmax.

Матрица весов – это матрица заданного размера случайно (псевдослучайно) спрогнозированных чисел, которая умножается на векторы. Значения весов корректируются в процессе обучения для достижения минимальной ошибки.

Переобучение – это состояние нейронной сети, при котором точность на обучающем наборе данных растёт, а на тестовом – падает. Нейронная сеть перестаёт обобщать данные и начинает запоминать их, уделяя большое внимание шуму в данных.

Регуляризация – это подмножество методов, помогающих обобщить модель для распознавания новых данных, вместо простого запоминания обучающих примеров. Регуляризация – это методы, предотвращающие переобучение нейронной сети.

Основные некоторые методы регуляризации:

Дроп-аут слой – это метод регуляризации, матрица, размер которой совпадает с размером выхода скрытого слоя. Дроп-аут слой состоит наполовину из 0 и наполовину из 1. Выход скрытого слоя умножается поэлементно на дроп-аут слой и на коэффициент 2 для компенсации «отключённых нейронов». Такая техника «отключения» нейронов довольно эффективна и в некоторых случаях помогает предотвратить переобучение.

Ранняя остановка обучения – это один из методов регуляризации, помогающий предотвратить переобучение. Основная задумка в том, чтобы остановить обучение нейронной сети, когда она начинает запоминать данные, т.е. переобучаться.

Пакетное обучение – это один из методов регуляризации, в ходе которого данные разделяются на пакеты, и эти пакеты подаются на входной слой. Вычисляется ошибка для каждого примера в пакете, ошибки усредняются, и уже готовая ошибка используется в методе обратного распространения ошибки.

Ошибка – это разница между полученным ответом и исходным, который мы подали на обучение. Обычно в обучении используется квадратичная ошибка, её преимущества перед абсолютной:

* Усиливает крупные ошибки в данных и уменьшает шумы за счёт возведения в квадрат
* Производная существует в любой точке

Метод обратного распространения ошибки – это процесс корректировки всех весов на основе полученной ошибки.

Обучение – это корректировка весов нейронной сети.

Шум – это неустранимые помехи в исходных данных.

Альфа-коэффициент – это число, влияющее на скорость обучения нейронной сети.

Распишем этапы обучения нейронной сети:

1. Инициализация гиперпараметров нейронной сети.
2. Подача в вход данных
3. Умножение на матрицу весов
4. Применение функции активации
5. Применение дроп-аут слоя
6. Умножение на матрицу весов
7. Применение функции активации
8. Вычисление ошибки
9. Вычисление производной от ошибки
10. Обучение методом обратного распространения ошибки

Определим функции и гиперпараметры для написания нашей нейронной сети:

Кол-во эпох epochs: 10

Альфа-коэффициент alpha: 1

Кол-во итераций iterations: 100

Кол-во нейронов в скрытом слое hidden\_size: 100

Размер пакета batch\_size: 100

Функция активации для скрытого слоя: tanh (см. приложение 20).

Функция активации для выходного слоя: softmax (см. приложение 21).

Объём тестовых данных от общего числа составляет 30%, обучающих, соответственно, 70%.

Почему был выбран tanh в качестве функции активации для скрытого слоя? Tanh удовлетворяет всем свойствам функции активации, описанным выше. Также tanh – это одна из лучших функций активации при малом количестве (до 8) скрытых слоёв.

Почему была выбрана softmax в качестве функции активации для выходного слоя? Softmax – это функция, которая явно показывает степень принадлежности объекта к определённому классу.

Учитывая все особенности, была написана нейронная сеть, решающая задачу классификации (см. листинг).

# **ГЛАВА 4. АНАЛИЗ РЕЗУЛЬТАТОВ.**

**4.1. Анализ точности.** Точность – это метрика качества обучения. Она отражает, какая часть поданных данных была определена верно.

Рассмотрим показатели точности одной эпохи обучения (см. приложение 22). При анализе точности всей модели в целом особую важность представляет точность на тестовых данных, нежели чем точность на обучающих, так как нейронная сеть обучается и запоминает обучающие данные, а тестовые данные нет, нейронная сеть о них ничего не знает. Самый высокий показатель точности на тестовых данных составляет 0,68364. Это довольно низкий показатель для модели нашего типа.

Что касается точности на обучающей выборке, то наибольший показатель равен 0,7125.

Начиная с 46 итерации виден резкий спад обоих показателей точностей. Это можно объяснить переобучением модели.

**4.2. Построение графиков.** Для наглядного отображения процесса обучения были построены графики показателей точности обучающей и тестовой выборок в 20 эпох. Данные были усреднены (см. приложение 23). Попытаемся сделать графики более гладкими (см. приложения 24-27). Мы можем подтвердить нашу гипотезу о том, что точность начинает падать в интервале с 40 по 50 итерации. Также видно то, что точность тестовой выборки начинает убывать практически с самого начала обучения, в то время как точность обучающей выборки продолжает расти. Данный факт можно обосновать тем, что модель начала переобучение практически в самом начале обучения. Это может быть связано с отсутствием большого объёма информации о пациентах, неполнотой предоставленной информации и наличием в ней ошибочных записей.

# **ЗАКЛЮЧЕНИЕ**

Написанная нейронная сеть по определению метода операции на основе данных о пациентах смогла верно определить тип операции лишь на 68%. Это довольно низкий показатель.

# **ВЫВОД**

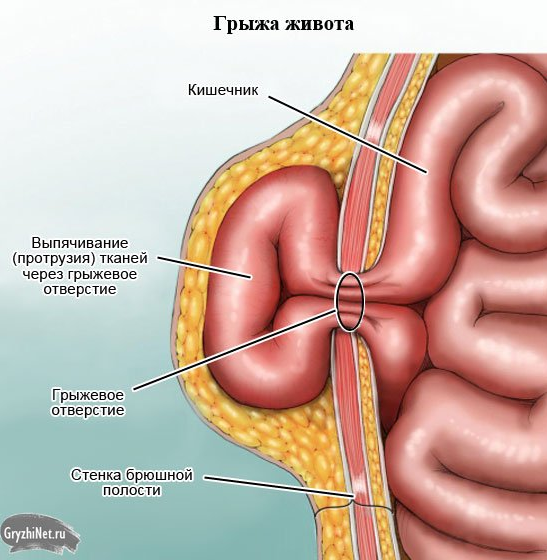
Написанная нейронная сеть по определению метода операции на основе данных о пациентах, в теории, может быть использована только в качестве помощника хирурга в принятии решения о методе операции, конечный результат должен оставаться за опытным специалистом.

# **БИБЛИОГРАФИЧЕСКИЙ СПИСОК**

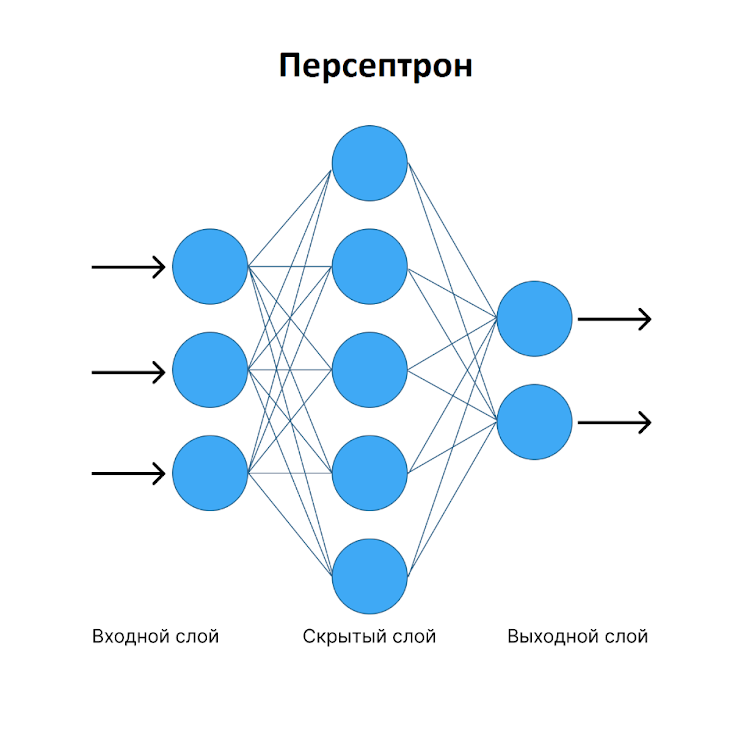
1. Эндрю Траск, Грокаем глубокое обучение / Эндрю Траск, перевод А. Киселёв – СПб: Питер, 2019. – 352 с.: ил. – (Серия «Библиотека программиста»).
2. Гиперпараметр (машинное обучение) / <https://ru.wikipedia.org/wiki/Гиперпараметр_(машинное_обучение)>
3. Очень краткая история нейросетей: от разработок 20-го века до ChatGPT / <https://vc.ru/future/606777-ochen-kratkaya-istoriya-neirosetei-ot-razrabotok-20-go-veka-do-chatgpt>
4. Функции активации нейросети: сигмоида, линейная, ступенчатая, ReLu, tahn / <https://neurohive.io/ru/osnovy-data-science/activation-functions/>
5. Нейронные сети — будущие помощники врачей / <https://biomolecula.ru/articles/neironnye-seti-budushchie-pomoshchniki-vrachei>

# **ПРИЛОЖЕНИЯ**

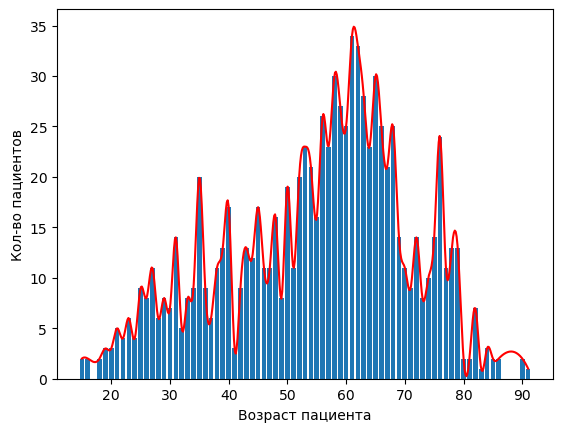
**ПРИЛОЖЕНИЕ 1**

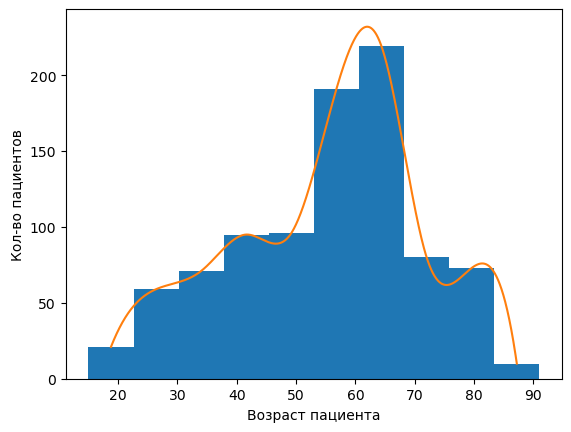


**ПРИЛОЖЕНИЕ 2**

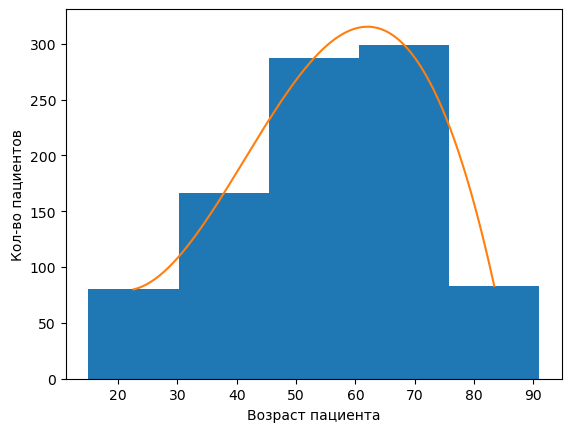


**ПРИЛОЖЕНИЕ 3**

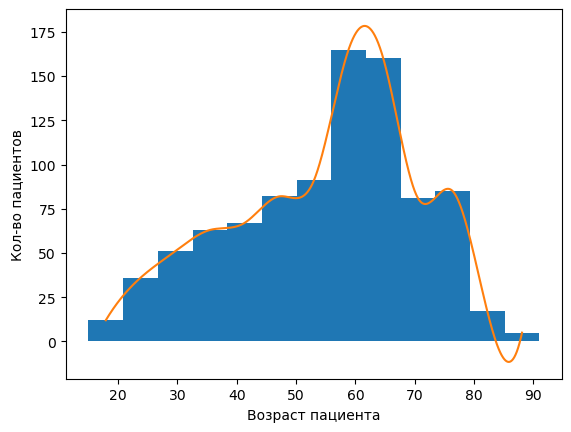
**ПРИЛОЖЕНИЕ 4**



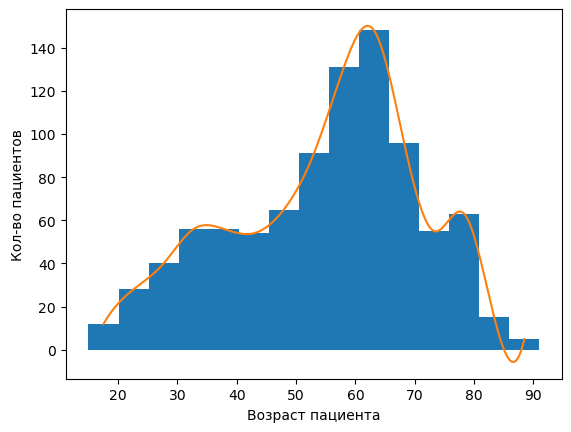
**ПРИЛОЖЕНИЕ 5**



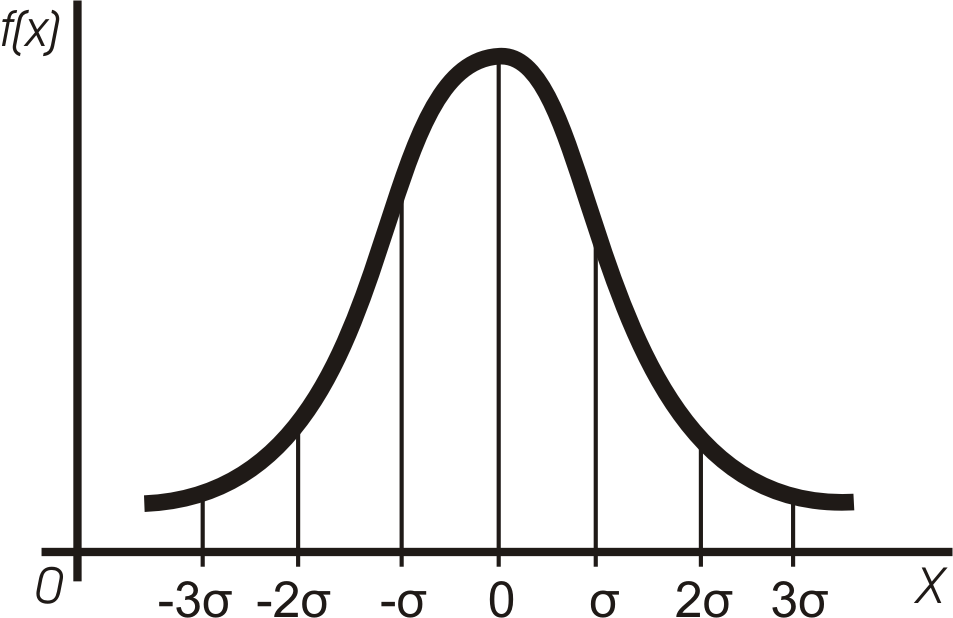
**ПРИЛОЖЕНИЕ 6**



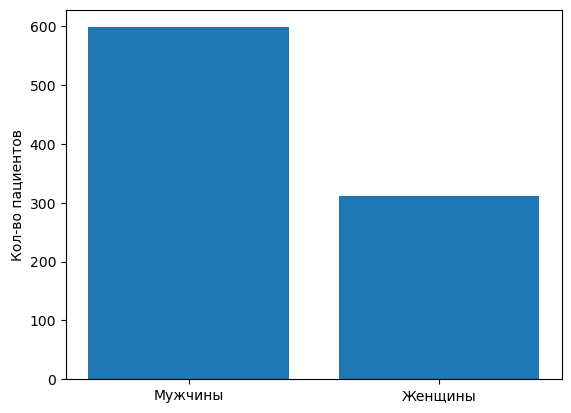
**ПРИЛОЖЕНИЕ 7**



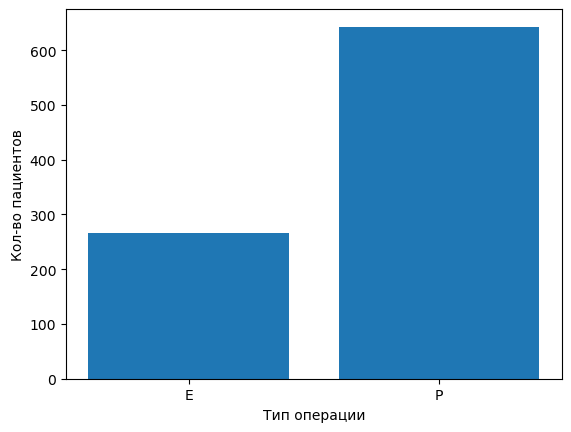
**ПРИЛОЖЕНИЕ 8**



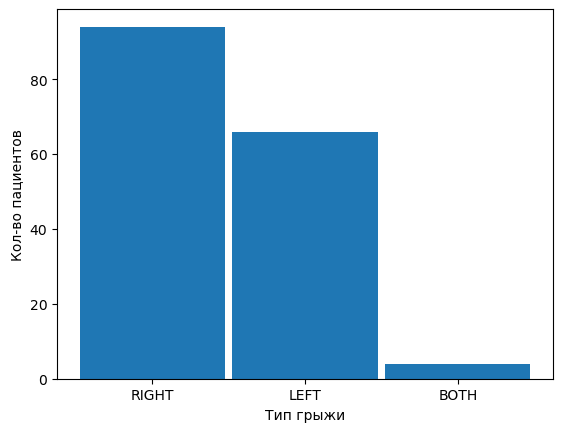
**ПРИЛОЖЕНИЕ 9**



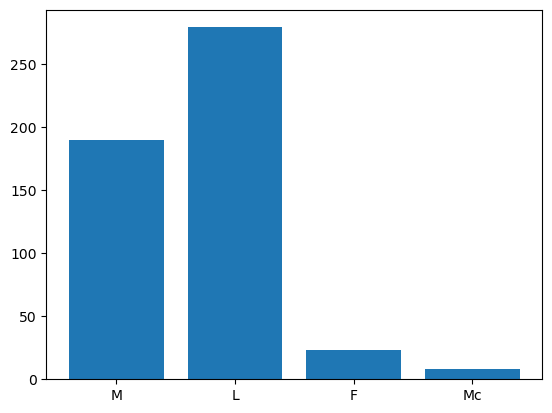
**ПРИЛОЖЕНИЕ 10**



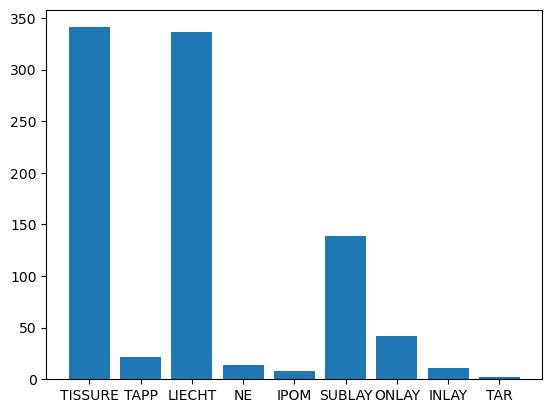
**ПРИЛОЖЕНИЕ 11**



**ПРИЛОЖЕНИЕ 12**



**ПРИЛОЖЕНИЕ 13**



**ПРИЛОЖЕНИЕ 14**

male = 0

female = 0

df\_gender = np.array(df['gender']) # столбец данных половых признаков пациентов

length = len(df\_gender)

gender = np.zeros((length, 2)) # создадим матрицу, в каждой строке которой первый индекс обозначает принадлежность

# к мужскому полу, а второй - к женскому

**for** i **in** range(len(df\_gender)):

**if** df\_gender[i] == 'm':

gender[i, 0] = 1

male += 1

**elif** df\_gender[i] == 'f':

gender[i, 1] = 1

female += 1

**ПРИЛОЖЕНИЕ 15**

df\_age = np.array(df['age'])

x\_min = np.min(df\_age)

x\_max = np.max(df\_age)

age = np.zeros(length)

**for** i **in** range(length):

age[i] = (df\_age[i] - x\_min) / (x\_max - x\_min) # нормирование данных

age.shape = -1, 1

**ПРИЛОЖЕНИЕ 16**

df\_first\_diagnoz = np.array(df['first\_diagnoz']) # первый диагноз

df\_second\_diagnoz = np.array(df['second\_diagnoz']) # второй диагноз

df\_complication = np.array(df['complication']) # осложнения

df\_anam = np.array(df['anam']) # дополнение

df\_diagnoz = np.hstack([df\_first\_diagnoz, df\_second\_diagnoz, df\_complication, df\_anam]) # все данные, объединённые в один массив

**ПРИЛОЖЕНИЕ 17**

words = np.unique([]) # множество уникальных слов

**for** diag **in** df\_diagnoz:

**try**:

uniq = np.unique(diag.split())

words = np.union1d(words, uniq)

**except** AttributeError:

**continue**

**ПРИЛОЖЕНИЕ 18**

words = list(words) # присваиваем каждому слову свой индекс

dictionary = dict()

**for** i, word **in** enumerate(words):

dictionary[word] = i

dictionary.pop('Отс')

**ПРИЛОЖЕНИЕ 19**

first\_diagnoz = np.zeros((length, len(words)))

attribute\_error\_1 = 0

key\_error\_1 = 0

i = 0

**for** c **in** df\_first\_diagnoz:

**try**:

line = c.split()

**for** s **in** line:

**try**:

first\_diagnoz[i, dictionary[s]] = 1

**except** KeyError:

key\_error\_1 += 1

**except** AttributeError:

attribute\_error\_1 += 1

i += 1

second\_diagnoz = np.zeros((length, len(words)))

i = 0

**for** c **in** df\_second\_diagnoz:

**try**:

line = c.split()

**for** s **in** line:

**try**:

second\_diagnoz[i, dictionary[s]] = 1

**except** KeyError:

key\_error\_1 += 1

**except** AttributeError:

attribute\_error\_1 += 1

i += 1

complication = np.zeros((length, len(words)))

i = 0

**for** c **in** df\_complication:

**try**:

line = c.split()

**for** s **in** line:

**try**:

complication[i, dictionary[s]] = 1

**except** KeyError:

key\_error\_1 += 1

**except** AttributeError:

attribute\_error\_1 += 1

i += 1

anam = np.zeros((length, len(words)))

i = 0

**for** c **in** df\_anam:

**try**:

line = c.split()

**for** s **in** line:

**try**:

anam[i, dictionary[s]] = 1

**except** KeyError:

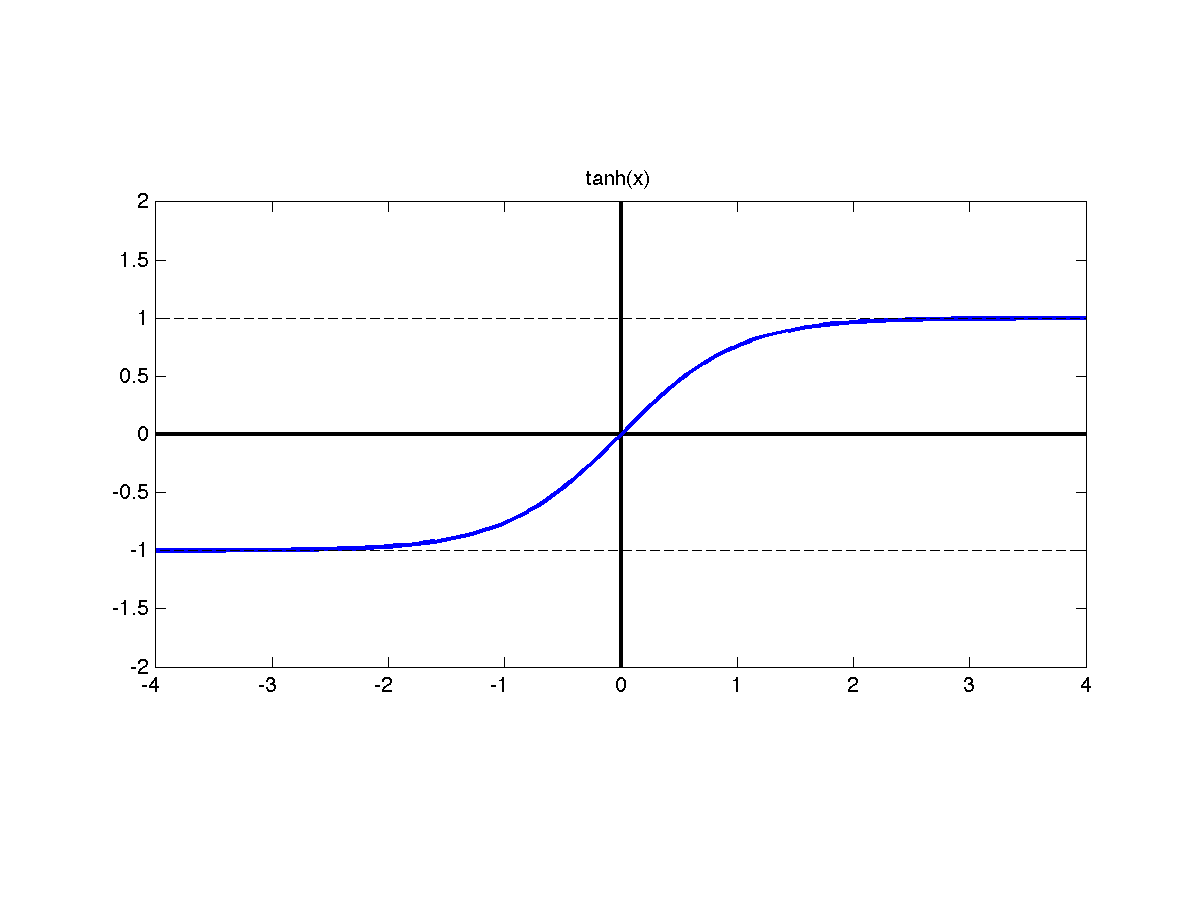
key\_error\_1 += 1

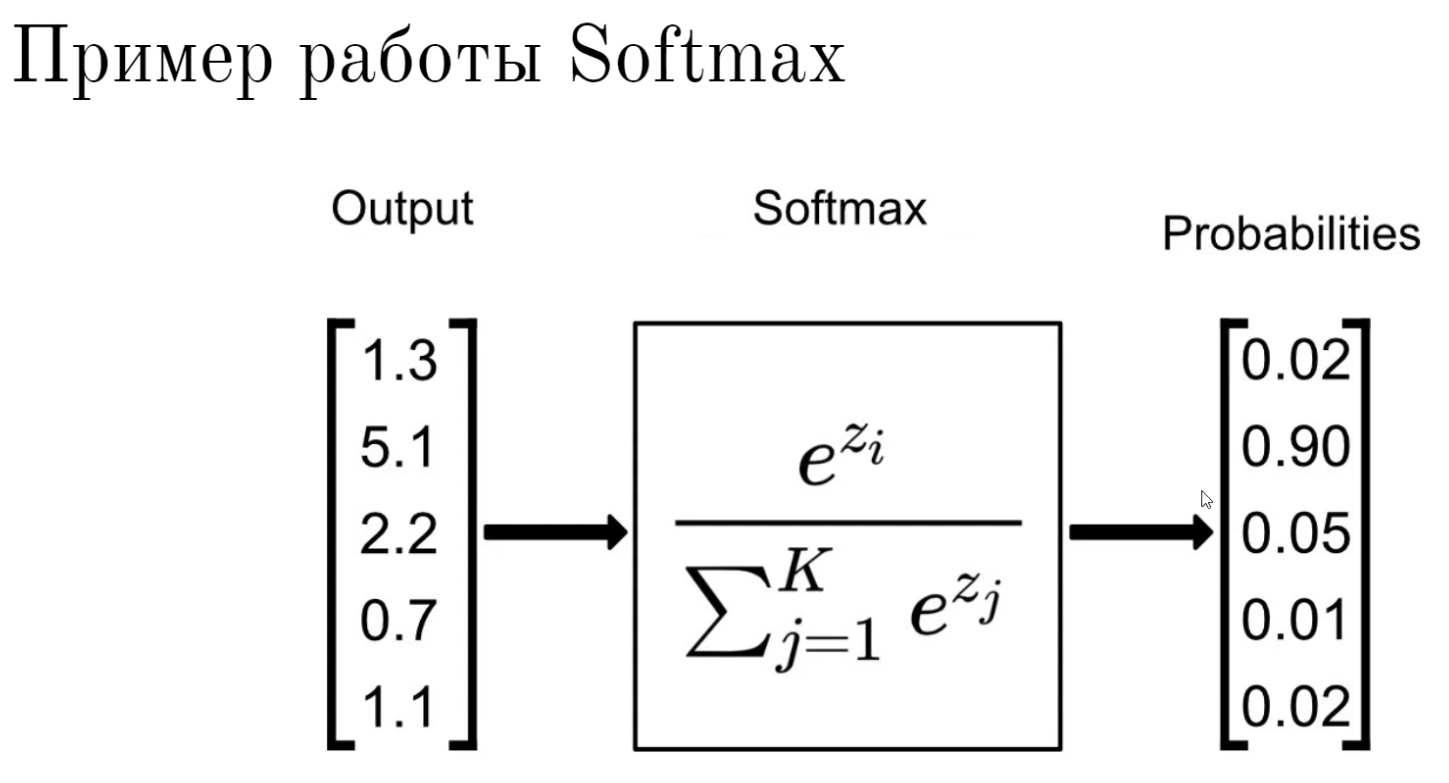
**except** AttributeError:

attribute\_error\_1 += 1

i += 1

**ПРИЛОЖЕНИЕ 20**

 **ПРИЛОЖЕНИЕ 21**



**ПРИЛОЖЕНИЕ 22**

Эпоха: 1

Итераций: 1, Точность обучения: 0.45312, Точность теста: 0.58182

Итераций: 2, Точность обучения: 0.56875, Точность теста: 0.59636

Итераций: 3, Точность обучения: 0.61719, Точность теста: 0.61091

Итераций: 4, Точность обучения: 0.65156, Точность теста: 0.61818

Итераций: 5, Точность обучения: 0.66562, Точность теста: 0.64364

Итераций: 6, Точность обучения: 0.67812, Точность теста: 0.64727

Итераций: 7, Точность обучения: 0.6625, Точность теста: 0.64

Итераций: 8, Точность обучения: 0.67969, Точность теста: 0.64727

Итераций: 9, Точность обучения: 0.68125, Точность теста: 0.66909

Итераций: 10, Точность обучения: 0.67969, Точность теста: 0.64727

Итераций: 11, Точность обучения: 0.69531, Точность теста: 0.64727

Итераций: 12, Точность обучения: 0.7, Точность теста: 0.65091

Итераций: 13, Точность обучения: 0.6875, Точность теста: 0.64

Итераций: 14, Точность обучения: 0.68125, Точность теста: 0.63273

Итераций: 15, Точность обучения: 0.69844, Точность теста: 0.65455

Итераций: 16, Точность обучения: 0.69531, Точность теста: 0.64364

Итераций: 17, Точность обучения: 0.67812, Точность теста: 0.66545

Итераций: 18, Точность обучения: 0.69219, Точность теста: 0.64

Итераций: 19, Точность обучения: 0.70312, Точность теста: 0.65091

Итераций: 20, Точность обучения: 0.69844, Точность теста: 0.62909

Итераций: 21, Точность обучения: 0.70469, Точность теста: 0.65455

Итераций: 22, Точность обучения: 0.69844, Точность теста: 0.65818

Итераций: 23, Точность обучения: 0.7125, Точность теста: 0.65091

Итераций: 24, Точность обучения: 0.69688, Точность теста: 0.63273

Итераций: 25, Точность обучения: 0.70938, Точность теста: 0.64

Итераций: 26, Точность обучения: 0.69062, Точность теста: 0.66182

Итераций: 27, Точность обучения: 0.70625, Точность теста: 0.65455

Итераций: 28, Точность обучения: 0.7125, Точность теста: 0.65455

Итераций: 29, Точность обучения: 0.71094, Точность теста: 0.62182

Итераций: 30, Точность обучения: 0.7, Точность теста: 0.61091

Итераций: 31, Точность обучения: 0.70469, Точность теста: 0.63273

Итераций: 32, Точность обучения: 0.67969, Точность теста: 0.53818

Итераций: 33, Точность обучения: 0.68125, Точность теста: 0.61091

Итераций: 34, Точность обучения: 0.69844, Точность теста: 0.64727

Итераций: 35, Точность обучения: 0.6875, Точность теста: 0.66182

Итераций: 36, Точность обучения: 0.67969, Точность теста: 0.66909

Итераций: 37, Точность обучения: 0.68125, Точность теста: 0.68364

Итераций: 38, Точность обучения: 0.68281, Точность теста: 0.65818

Итераций: 39, Точность обучения: 0.7, Точность теста: 0.57091

Итераций: 40, Точность обучения: 0.65938, Точность теста: 0.57455

Итераций: 41, Точность обучения: 0.6625, Точность теста: 0.52364

Итераций: 42, Точность обучения: 0.68438, Точность теста: 0.59273

Итераций: 43, Точность обучения: 0.68281, Точность теста: 0.60727

Итераций: 44, Точность обучения: 0.66562, Точность теста: 0.53455

Итераций: 45, Точность обучения: 0.67812, Точность теста: 0.58545

Итераций: 46, Точность обучения: 0.6375, Точность теста: 0.61455

Итераций: 47, Точность обучения: 0.58281, Точность теста: 0.62545

Итераций: 48, Точность обучения: 0.61562, Точность теста: 0.62909

Итераций: 49, Точность обучения: 0.59062, Точность теста: 0.65091

Итераций: 50, Точность обучения: 0.57188, Точность теста: 0.57455

Итераций: 51, Точность обучения: 0.53281, Точность теста: 0.59273

Итераций: 52, Точность обучения: 0.55156, Точность теста: 0.62545

Итераций: 53, Точность обучения: 0.47969, Точность теста: 0.45818

Итераций: 54, Точность обучения: 0.46406, Точность теста: 0.51636

Итераций: 55, Точность обучения: 0.43594, Точность теста: 0.44727

Итераций: 56, Точность обучения: 0.39531, Точность теста: 0.46909

Итераций: 57, Точность обучения: 0.34062, Точность теста: 0.49455

Итераций: 58, Точность обучения: 0.43281, Точность теста: 0.37455

Итераций: 59, Точность обучения: 0.44844, Точность теста: 0.52727

Итераций: 60, Точность обучения: 0.41406, Точность теста: 0.45818

Итераций: 61, Точность обучения: 0.43125, Точность теста: 0.49455

Итераций: 62, Точность обучения: 0.45938, Точность теста: 0.49091

Итераций: 63, Точность обучения: 0.41875, Точность теста: 0.41455

Итераций: 64, Точность обучения: 0.43594, Точность теста: 0.42909

Итераций: 65, Точность обучения: 0.35312, Точность теста: 0.49818

Итераций: 66, Точность обучения: 0.44219, Точность теста: 0.37818

Итераций: 67, Точность обучения: 0.31562, Точность теста: 0.34909

Итераций: 68, Точность обучения: 0.32656, Точность теста: 0.36364

Итераций: 69, Точность обучения: 0.3625, Точность теста: 0.33455

Итераций: 70, Точность обучения: 0.29688, Точность теста: 0.28364

Итераций: 71, Точность обучения: 0.24688, Точность теста: 0.37818

Итераций: 72, Точность обучения: 0.41562, Точность теста: 0.50909

Итераций: 73, Точность обучения: 0.45312, Точность теста: 0.50909

Итераций: 74, Точность обучения: 0.46406, Точность теста: 0.54909

Итераций: 75, Точность обучения: 0.48281, Точность теста: 0.50182

Итераций: 76, Точность обучения: 0.42188, Точность теста: 0.4

Итераций: 77, Точность обучения: 0.32031, Точность теста: 0.34545

Итераций: 78, Точность обучения: 0.27656, Точность теста: 0.20364

Итераций: 79, Точность обучения: 0.27812, Точность теста: 0.34909

Итераций: 80, Точность обучения: 0.29063, Точность теста: 0.25091

Итераций: 81, Точность обучения: 0.3, Точность теста: 0.38182

Итераций: 82, Точность обучения: 0.32656, Точность теста: 0.36364

Итераций: 83, Точность обучения: 0.27812, Точность теста: 0.35273

Итераций: 84, Точность обучения: 0.3125, Точность теста: 0.34545

Итераций: 85, Точность обучения: 0.30469, Точность теста: 0.34545

Итераций: 86, Точность обучения: 0.31406, Точность теста: 0.26182

Итераций: 87, Точность обучения: 0.28125, Точность теста: 0.34545

Итераций: 88, Точность обучения: 0.29375, Точность теста: 0.30182

Итераций: 89, Точность обучения: 0.33906, Точность теста: 0.39636

Итераций: 90, Точность обучения: 0.30625, Точность теста: 0.45091

Итераций: 91, Точность обучения: 0.35156, Точность теста: 0.46909

Итераций: 92, Точность обучения: 0.37656, Точность теста: 0.43273

Итераций: 93, Точность обучения: 0.33125, Точность теста: 0.28

Итераций: 94, Точность обучения: 0.24219, Точность теста: 0.2

Итераций: 95, Точность обучения: 0.28906, Точность теста: 0.13818

Итераций: 96, Точность обучения: 0.3, Точность теста: 0.24364

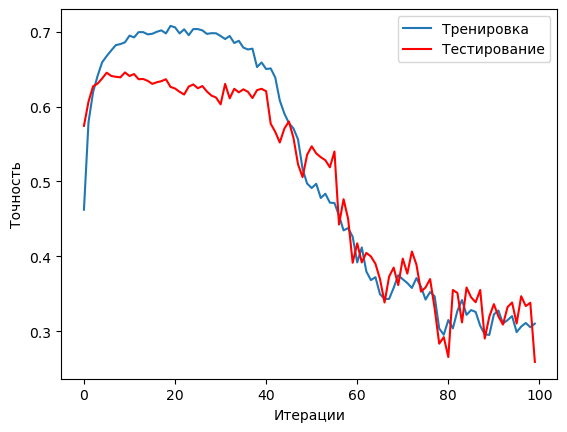
Итераций: 97, Точность обучения: 0.29375, Точность теста: 0.41091

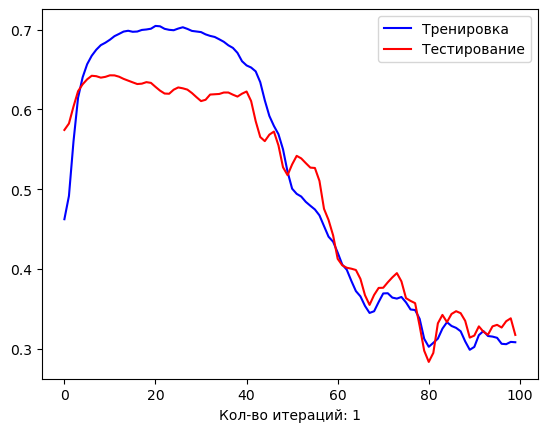
Итераций: 98, Точность обучения: 0.29844, Точность теста: 0.39273

Итераций: 99, Точность обучения: 0.32812, Точность теста: 0.18909

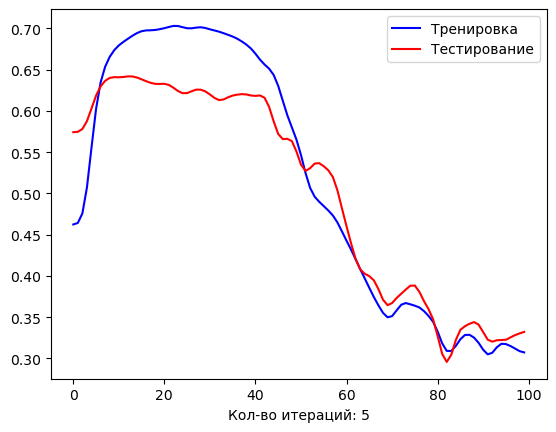
Итераций: 100, Точность обучения: 0.2875, Точность теста: 0.33818

**ПРИЛОЖЕНИЕ 23**

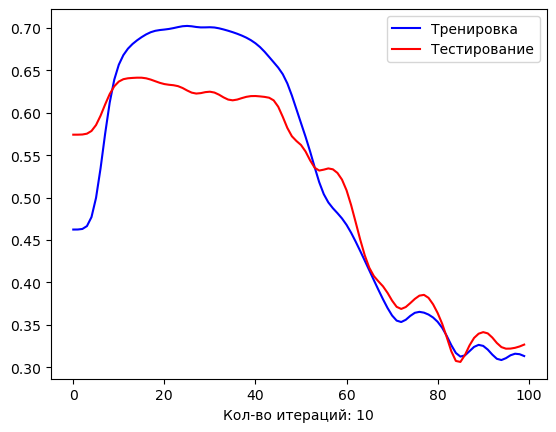
**ПРИЛОЖЕНИЕ 24**



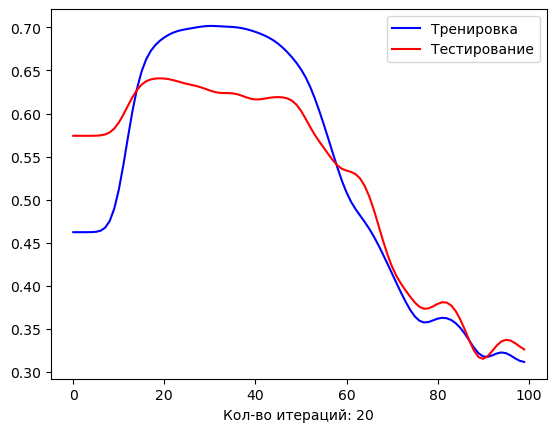
**ПРИЛОЖЕНИЕ 25**



**ПРИЛОЖЕНИЕ 26**



**ПРИЛОЖЕНИЕ 27**



**ЛИСТИНГ**

**import** pandas **as** pd

**import** numpy **as** np

**import** math

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

**import** matplotlib.pyplot **as** plt

**from** collections **import** Counter

**from** scipy.interpolate **import** make\_interp\_spline

df = pd.read\_excel('новые\_грыжи.xlsx')

male = 0

female = 0

df\_gender = np.array(df['gender']) # столбец данных половых признаков пациентов

length = len(df\_gender)

gender = np.zeros((length, 2)) # создадим матрицу, в каждой строке которой первый индекс обозначает принадлежность

# к мужскому полу, а второй - к женскому

**for** i **in** range(len(df\_gender)):

**if** df\_gender[i] == 'm':

gender[i, 0] = 1

male += 1

**elif** df\_gender[i] == 'f':

gender[i, 1] = 1

female += 1

df\_age = np.array(df['age'])

x\_min = np.min(df\_age)

x\_max = np.max(df\_age)

age = np.zeros(length)

**for** i **in** range(length):

age[i] = (df\_age[i] - x\_min) / (x\_max - x\_min)

age.shape = -1, 1

df\_first\_diagnoz = np.array(df['first\_diagnoz']) # первый диагноз

df\_second\_diagnoz = np.array(df['second\_diagnoz']) # второй диагноз

df\_complication = np.array(df['complication']) # осложнения

df\_anam = np.array(df['anam']) # дополнение

df\_diagnoz = np.hstack([df\_first\_diagnoz, df\_second\_diagnoz, df\_complication, df\_anam]) # все данные, объединённые в один массив

words = np.unique([]) # множество уникальных слов

**for** diag **in** df\_diagnoz:

**try**:

uniq = np.unique(diag.split())

words = np.union1d(words, uniq)

**except** AttributeError:

**continue**

words = list(words) # присваиваем каждому слову свой индекс

dictionary = dict()

**for** i, word **in** enumerate(words):

dictionary[word] = i

dictionary.pop('Отс')

first\_diagnoz = np.zeros((length, len(words)))

attribute\_error\_1 = 0

key\_error\_1 = 0

i = 0

**for** c **in** df\_first\_diagnoz:

**try**:

line = c.split()

**for** s **in** line:

**try**:

first\_diagnoz[i, dictionary[s]] = 1

**except** KeyError:

key\_error\_1 += 1

**except** AttributeError:

attribute\_error\_1 += 1

i += 1

second\_diagnoz = np.zeros((length, len(words)))

i = 0

**for** c **in** df\_second\_diagnoz:

**try**:

line = c.split()

**for** s **in** line:

**try**:

second\_diagnoz[i, dictionary[s]] = 1

**except** KeyError:

key\_error\_1 += 1

**except** AttributeError:

attribute\_error\_1 += 1

i += 1

complication = np.zeros((length, len(words)))

i = 0

**for** c **in** df\_complication:

**try**:

line = c.split()

**for** s **in** line:

**try**:

complication[i, dictionary[s]] = 1

**except** KeyError:

key\_error\_1 += 1

**except** AttributeError:

attribute\_error\_1 += 1

i += 1

anam = np.zeros((length, len(words)))

i = 0

**for** c **in** df\_anam:

**try**:

line = c.split()

**for** s **in** line:

**try**:

anam[i, dictionary[s]] = 1

**except** KeyError:

key\_error\_1 += 1

**except** AttributeError:

attribute\_error\_1 += 1

i += 1

# diagnoz = first\_diagnoz + second\_diagnoz + complication + anam

print(attribute\_error\_1)

print(key\_error\_1)

df\_emerg = df['emerg']

emerg = np.zeros((length, 2))

**for** i **in** range(length):

**if** df\_emerg[i] == 'E':

emerg[i, 0] = 1

**elif** df\_emerg[i] == 'P':

emerg[i, 1] = 1

**else**:

**continue**

df\_type = df['type']

type = np.zeros((length, 4))

**for** i **in** range(length):

**if** df\_type[i] == 'M':

type[i, 0] = 1

**elif** df\_type[i] == 'L':

type[i, 1] = 1

**elif** df\_type[i] == 'F':

type[i, 2] = 1

**elif** df\_type[i] == 'Mc':

type[i, 3] = 1

**else**:

**continue**

df\_side = df['side']

side = np.zeros((length, 3))

**for** i **in** range(length):

**if** df\_side[i] == 'LEFT':

side[i, 0] = 1

**elif** df\_side[i] == 'RIGHT':

side[i, 1] = 1

**elif** df\_side[i] == 'BOTH':

side[i, 2] = 1

**else**:

**continue**

another\_information = np.array(df.iloc[:, 10:-1])

**for** i **in** range(length):

**for** j **in** range(len(another\_information[i])):

**if** math.isnan(another\_information[i, j]) **is** True:

another\_information[i, j] = 0

**else**:

**continue**

data = np.hstack([gender, age, first\_diagnoz, second\_diagnoz, complication, anam, emerg, type, side, another\_information])

y = df.iloc[:, -1] # столбец с методами проведённых операций

uniq\_labels = np.unique(y)

dict\_labels = dict()

**for** i, method **in** enumerate(uniq\_labels):

dict\_labels[method] = i

labels = np.zeros((length, len(uniq\_labels)))

**for** i **in** range(length):

labels[i, dict\_labels[y[i]]] = 1

x\_train, x\_test, y\_train, y\_test = train\_test\_split(data, labels, train\_size=0.7)

**def** **softmax**(layer: np.array):

exp = np.exp(layer)

total = np.sum(exp, axis=1, keepdims=True)

# print(np.sum(exp / total)) - всегда равна 1

**return** exp / total

**def** **tanh2deriv**(layer: np.array):

**return** 1 - (layer \*\* 2)

**def** **tanh**(layer: np.array):

**return** np.tanh(layer)

**def** **relu**(layer: np.array):

**return** np.where(layer > 0, layer, 0)

**def** **relu2deriv**(layer: np.array):

**return** np.where(layer > 0, 1, 0)

epochs = 10 # кол-во эпох

train\_epoch = []

test\_epoch = []

**for** epoch **in** range(epochs):

print(f'Эпоха: {epoch+1}')

alpha = 1

iterations = 100

hidden\_size = 100

batch\_size = 100

x\_size = len(x\_train[0])

y\_size = len(y\_train[0])

train\_accuracy = []

test\_accuracy = []

weights\_0\_1 = 0.2 \* np.random.random((x\_size, hidden\_size)) - 0.1

weights\_1\_2 = 0.2 \* np.random.random((hidden\_size, y\_size)) - 0.1

**for** j **in** range(iterations):

correct\_cnt = 0

**for** i **in** range(len(x\_train) // batch\_size):

batch\_start, batch\_end = ((i \* batch\_size), (i + 1) \* batch\_size)

layer\_0 = x\_train[batch\_start:batch\_end]

layer\_1 = tanh(np.dot(layer\_0, weights\_0\_1))

dropout\_mask = np.random.randint(2, size=layer\_1.shape)

layer\_1 \*= dropout\_mask \* 2

layer\_2 = softmax(np.dot(layer\_1, weights\_1\_2))

**for** k **in** range(batch\_size):

correct\_cnt += int(np.argmax(layer\_2[k:k+1]) == np.argmax(y\_train[batch\_start+k:batch\_start+k+1]))

layer\_2\_delta = (layer\_2 - y\_train[batch\_start:batch\_end]) / (batch\_size \* layer\_2.shape[0])

layer\_1\_delta = np.dot(layer\_2\_delta, weights\_1\_2.T) \* tanh2deriv(layer\_1)

layer\_1\_delta \*= dropout\_mask

weights\_0\_1 -= alpha \* np.dot(layer\_0.T, layer\_1\_delta)

weights\_1\_2 -= alpha \* np.dot(layer\_1.T, layer\_2\_delta)

test\_correct\_cnt = 0

**for** i **in** range(len(x\_test)):

layer\_0 = x\_test[i:i+1]

layer\_1 = tanh(np.dot(layer\_0, weights\_0\_1))

layer\_2 = softmax(np.dot(layer\_1, weights\_1\_2))

test\_correct\_cnt += int(np.argmax(layer\_2) == np.argmax(y\_test[i:i+1]))

test\_accuracy.append(np.round(test\_correct\_cnt / len(x\_test), 5))

train\_accuracy.append(np.round(correct\_cnt / len(x\_train), 5))

print(f'Итераций: {j+1}, Точность обучения: {np.round(correct\_cnt / len(x\_train), 5)}, Точность теста: {np.round(test\_correct\_cnt / len(x\_test), 5)}')

train\_epoch.append(train\_accuracy)

test\_epoch.append(test\_accuracy)

data\_train\_epoch = sum(np.array(train\_epoch)) / epochs

data\_test\_epoch = sum(np.array(test\_epoch)) / epochs

x = np.arange(len(test\_accuracy))

plt.plot(x, data\_train\_epoch, label='Тренировка')

plt.plot(x, data\_test\_epoch, label='Тестирование', color='red')

plt.legend()

plt.xlabel('Итерации')

plt.ylabel('Точность')

iter = 20

y = [data\_train\_epoch[0]]

**for** i **in** range(len(data\_train\_epoch) - 1):

y.append((data\_train\_epoch[i] + data\_train\_epoch[i + 1])/2)

**for** j **in** range(iter):

y\_1 = [y[0]]

**for** i **in** range(len(y) - 1):

y\_1.append((y[i] + y[i+1])/2)

y = y\_1.copy()

plt.plot(x, y\_1, color='blue', label='Тренировка')

y = [data\_test\_epoch[0]]

**for** i **in** range(len(data\_test\_epoch) - 1):

y.append((data\_test\_epoch[i] + data\_test\_epoch[i + 1])/2)

**for** j **in** range(iter):

y\_1 = [y[0]]

**for** i **in** range(len(y) - 1):

y\_1.append((y[i] + y[i+1])/2)

y = y\_1.copy()

plt.plot(x, y\_1, color='red', label='Тестирование')

plt.xlabel(f'Кол-во итераций: {iter}')

plt.legend()

plt.plot(x, train\_accuracy, label='Точность тренировки')

plt.legend()

plt.xlabel('Итерации')

plt.ylabel('Точность')

plt.plot(x, test\_accuracy, label='Точность тестирования', color='red')

plt.legend()

plt.xlabel('Итерации')

plt.ylabel('Точность')

plt.bar(['Мужчины', 'Женщины'], [male, female])

plt.ylabel('Кол-во пациентов')

print(female, male)

graf = plt.hist(df\_age, bins=10)

#plt.plot(graf[0], graf[1], color='red')

print(graf[0], graf[1])

graf\_x = list()

**for** i **in** range(1, len(graf[1])):

graf\_x.append(graf[1][i] - ((graf[1][i] - graf[1][i-1])/2))

#plt.plot(graf\_x, graf[0])

x\_y\_spline = make\_interp\_spline(graf\_x, graf[0])

x\_ = np.linspace(min(graf\_x), max(graf\_x), 2000)

y\_ = x\_y\_spline(x\_)

plt.plot(x\_, y\_)

plt.xlabel('Возраст пациента')

plt.ylabel('Кол-во пациентов')

counter\_emerg = Counter(df\_emerg)

plt.bar(list(counter\_emerg.keys())[:2], list(counter\_emerg.values())[:2])

plt.xlabel('Тип операции')

plt.ylabel('Кол-во пациентов')

print(list(counter\_emerg.keys())[:2], list(counter\_emerg.values())[:2])

print(267+643)

print(267 \* 100 / 910)

counter\_type = list(Counter(df\_type).items())

counter\_type.pop(2)

counter\_type = sorted(counter\_type, key=**lambda** x: x[1], reverse=True)

x = list(map(**lambda** x: x[0], counter\_type))

y = list(map(**lambda** x: x[1], counter\_type))

plt.bar(x, y, width=0.95)

counter\_side = list(Counter(df\_side).items())

counter\_side.pop()

counter\_side = sorted(counter\_side, key=**lambda** x: x[1], reverse=True)

x = list(map(**lambda** x: x[0], counter\_side))

y = list(map(**lambda** x: x[1], counter\_side))

plt.bar(x, y, width=0.95)

plt.xlabel('Тип грыжи')

plt.ylabel('Кол-во пациентов')

df\_result = np.array(df.iloc[:, -1])

counter\_result = Counter(df\_result)

plt.bar(counter\_result.keys(), counter\_result.values())