

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: 09.04.01 «Информатика и вычислительная техника»

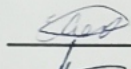
Магистерская программа: «Прикладной искусственный интеллект»

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА МАГИСТРА

на тему

**РАЗРАБОТКА ИНТЕЛЛЕКТУАЛЬНОЙ
СИСТЕМЫ ДИАГНОСТИКИ МЕДИЦИНСКОГО
СОСТОЯНИЯ ЧЕЛОВЕКА**

Студент


(подпись, дата) 09.06.2026

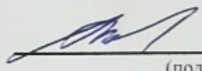
Савин Егор Дмитриевич
(ФИО полностью)

Руководитель


(подпись, дата) 09.06.2026

Тимонин А.Ю.
(фамилия, инициалы)

Нормоконтролёр


(подпись, дата) 16.06.2026

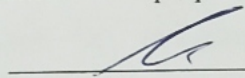
Бычков А.С.
(фамилия, инициалы)

Рецензент

главный специалист
АО, ННП, рудник
(должность, место работы) Василий
(подпись, дата) 22.05.2026
(фамилия, инициалы) Васильев А.В.

Работа допущена к защите (протокол зас. кафедры от) 22.05.2026 № 14

Заведующий кафедрой

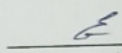

(подпись)

Митрохин М.А.
(фамилия, инициалы)

Работа защищена с отметкой

отлично
(протокол заседания ГЭК от 23.06.2026 № 32)

Секретарь ГЭК


(подпись)

Гурин Е.В.
(фамилия, инициалы)

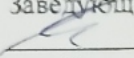
Пенза 2026

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
 М.А. Митрохин
«20» сентября 2024 г.

**ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
МАГИСТРА**

1. Студент Савин Егор Дмитриевич гр. 24ВВИм2 факультета
вычислительной техники, направления 09.04.01
2. Тема работы «Разработка интеллектуальной системы диагностики
медицинского состояния человека».

Тема утверждена приказом ПГУ № 920/о от "20" сентября 2024 г.

3. Руководитель магистерской диссертации к.т.н., Тимонин А.Ю.
4. Задание на работу (назначение разработки, исходные данные и т.п.)

Назначение разработки: осуществления медицинской диагностики
пациента с применением технологий искусственного интеллекта. Выдача
показаний и рекомендаций на основе предсказания модели.

Разработать интеллектуальную систему, способную производить
диагностику пациента на основе предоставленных данных. Разработать
графический интерфейс для быстрой навигации и взаимодействия с элементами
системы.

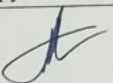
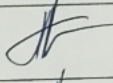
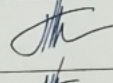
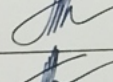
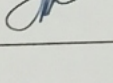
5. Перечень подлежащих разработке вопросов:

1. Анализ предметной области применения технологий искусственного интеллекта в сфере медицины.

2. Разработка системы для обучения и применения моделей искусственного интеллекта.

3. Отладка работы системы интеллектуальной диагностики состояния здоровья человека.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1.	Анализ предметной области применения интеллектуальных систем в сфере медицины	10%	21.09.2024-21.11.2024	
2.	Анализ существующих методов машинного обучения	10%	22.11.2024-22.01.2025	
3.	Проектирование структуры разрабатываемой системы	20%	23.01.2025-23.04.2025	
4.	Разработка всех модулей приложения	40%	24.04.2025-29.01.2026	
5.	Отладка работы системы	10%	30.01.2026-30.03.2026	
6.	Оформление выпускной работы	10%	31.03.2026-31.05.2026	

Дата выдачи задания "20" сентября 2024 г.

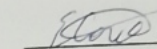
Руководитель магистерской диссертации



Тимонин А.Ю.

(подпись, дата)

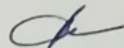
Задание к исполнению принял студент



Савин Е.Д.

(подпись, дата)

Работу к защите допустить
Декан факультета



Фионова Л.Р.

(подпись, дата)

22.06.2026

РЕФЕРАТ

Магистерская работа содержит 91 лист, 29 рисунков, 6 таблиц, 17 источников, 2 приложения.

МАШИННОЕ ОБУЧЕНИЕ, ИНТЕЛЛЕКТУАЛЬНАЯ МЕДИЦИНСКАЯ ДИАГНОСТИКА, ВЫЯВЛЕНИЕ ЗАБОЛЕВАНИЙ, НЕЙРОННЫЕ СЕТИ, PYTHON, ОБРАБОТКА ДАННЫХ, ОЦЕНКА КАЧЕСТВА МОДЕЛЕЙ

Цель работы – разработка системы реализующей обработку медицинских данных, обучение различных моделей с применением технологий машинного обучения и проведение оценки качества работы данных моделей с целью проведения медицинского анализа состояния человека и предсказания возможных заболеваний.

Объект исследования – проведение диагностических предсказаний по данным введенным пользователем с помощью модели обученной на реальных данных.

Предмет исследования – методы и алгоритмы машинного обучения и обработки данных для получения достоверных предсказаний.

Результатом работы является разработанная система, способная обрабатывать медицинские данные, обучать на них различные модели, сравнивать их между собой и оценивать результаты каждой по отдельности. Система способна производить предсказания с помощью обученных моделей по данным введенным пользователем. Каждая из реализованных в системе моделей обладает своими преимуществами и недостатками. В ходе проведения практических исследований и проведения тонкой настройки моделей удалось добиться высокого показателя точности предсказаний.

Лист	№ докум.	Подпись	Дата
1			
раб	Савин Е.Д.		18.06.21
испр.	Тимошин А.Ю.		18.06.21
Сонтр	Бычков А.С.		18.06.21
ерд	Митрохин М.А.		18.06.21

ПГУ 1.090401.13.001 ПЗ

Разработка интеллектуальной
системы диагностики
медицинского состояния
человека
Пояснительная записка

Лит.	Лист	Листов
	2	91
ФВТ, каф. ВТ		
гр. 246ВИм2		

Содержание

ВВЕДЕНИЕ	6
1 Анализ предметной области и существующих методов решения	8
1.1 Актуальность задачи	8
1.2 Обзор существующих систем	11
1.3 Подведение итогов анализа	14
1.4 Вывод по разделу	15
2 Исследование технологий решения поставленной задачи	16
2.1 Введение в проблему	16
2.1.3 Отбор ключевых признаков	18
2.1.4 Проблема балансировки данных	19
2.2 Выбор архитектуры обучаемой модели	19
2.3 Оценка качества предсказаний	21
2.4 Интеграция модели в приложение	22
2.5 Вывод по разделу	23
3 Разработка структуры системы	24
3.1 Выбор методов и инструментов. Обоснование архитектуры системы ..	25
3.1.1 Выбор алгоритмов и критериев оценки	25
3.1.2 Сравнительная характеристика каждого из алгоритмов	27
3.2 Выбор инструментов разработки	28
3.3 Проектирование структуры системы	30
3.4 Вывод по разделу	32
4 Программная реализация системы	33
4.1 Обработка данных	33
4.2 Реализация моделей, их обучение и оценка	55
4.3 Применение модели в конечном приложении	64
ЗАКЛЮЧЕНИЕ	70
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	71
Приложение А – листинг программы	74
Приложение Б – презентация	83

ВВЕДЕНИЕ

Компьютерные технологии всецело поглотили нашу жизнь. Спустя двадцать лет с начала 21-го века цифровая индустрия претерпела невообразимое количество преобразований, а скачок развития компьютерных технологий поражает воображение.

Данное развитие естественным образом повлекло за собой прогресс и других сфер нашей жизни. Вместе с появлением новых возможностей хранения и обработки данных в цифровом формате в различных сферах начались изменения в сторону компьютеризации различных аспектов.

На данный момент будет тяжело назвать какую-либо отрасль, которую бы никак не затронул процесс цифровизации. Начиная со сложных научно-технологических аспектов деятельности людей, заканчивая самыми обыденными походами в магазин, везде людей сопровождают электронные устройства, упрощающие работу и взаимодействие.

Для ведения учета товаров в магазине уже давно используются компьютеры. Чтобы получить необходимые документы или выписки далеко не всегда требуется ехать в специальные центры, ведь некоторые из них можно заказать на отдельных официальных интернет ресурсах.

Этот процесс не мог не затронуть такую важную часть индустрии как медицина. Ведение цифрового учета пациентов, электронные больничные записи – все это осуществляется с использованием компьютерных технологий. Однако это не предел развития индустрии. Параллельно с развитием самих компьютерных систем развивалось и направление машинного обучения, цель которого – применение искусственного интеллекта для достижений результатов, на которые не способен человек. Пациент может записаться к врачу на прием еще до того, как посетит медицинское заведение, а в свою очередь врачу не требуется выполнять всю работу исключительно на бумаге, а благодаря искусственному интеллекту удастся повысить потенциальную точность выполняемой работы.

Данная работа затрагивает тематику применения компьютерных технологий в медицине, а точнее отдельного направления применения данных технологий – интеллектуальных систем. По мере развития электронных методов обработки данных увеличилась и область их применения. На данный момент фраза «Искусственный интеллект» уже перестала восприниматься, как нечто далекое, непонятное, неосознанное, так как на примере множества различных систем была доказана эффективность применения алгоритмов машинного обучения.

Помимо известных многим речевых моделей, выступающих в качестве экспертных систем, способных отвечать на задаваемые вопросы, составлять связные предложения и имитировать человеческое общение, технологии искусственного интеллекта находят свое применение практически везде. Системы ведения сельского хозяйства, способные выявлять проблемы при развитии культур, вовремя применять различные химикаты и менять условия содержания растений в зависимости от внешних климатических воздействий. Применение в сфере маркетинга, где искусственный интеллект способен выявлять различные паттерны человеческого поведения в момент совершения покупок для дальнейшей максимизации прибыли.

Применение ИИ в сфере медицины – является важной ступенью в развитии самой отрасли. С возможностью анализировать большие массивы данных появляется и возможность автоматизировать отдельные процессы в работе медицинских учреждений. С помощью интеллектуальных систем можно упростить выдачу показаний по случаям болезни, ускорить процесс диагностики состояния пациентов, а также выявить различные заболевания в зарождающемся состоянии по вторичным признакам, когда никаких очевидных симптомов еще не было замечено.

Интеграция данных технологий на основе машинного обучения позволит снять часть нагрузки с сотрудников заведения. Системы мониторинга состояния пациентов смогут оценивать ситуацию и обновлять данные в реальном времени и, возможно, даже принять экстренные меры,

которые смогут спасти жизнь человеку в случае экстренных ситуаций, пока на место не прибыл медицинский персонал.

Несмотря на то, что применение технологий искусственного интеллекта позволяет повысить точность и эффективность диагностирования состояния пациентов, внедрение подобных технологий также может и принести вред. Технологии машинного обучения требуют сбора и стандартизации большого количества данных. Такие данные должны быть отобраны, обработаны и верифицированы, как корректные. Также внедрение невозможно произвести по щелчку пальца, ведь для этого потребуется соответствующее оборудование и повышение квалификации персонала медучреждений. При всем этом стоит задача решения этических вопросов применения ИИ, ведь в случае ошибки страдают люди, что в свою очередь не позволяет дать интеллектуальным системам полную свободу действия.

Сам по себе искусственный интеллект не является волшебным решением для всех проблем, и только в руках умелого специалиста он может демонстрировать по-настоящему хороший результат.

1 Анализ предметной области и существующих методов решения поставленной задачи

1.1 Актуальность задачи

На данный момент сфера искусственного интеллекта в медицине испытывает полномасштабное развитие. Системы на основе технологий машинного обучения находят широкое применение в различных областях медицины и здравоохранения.

Разработка новых систем и экспериментирование в данной отрасли позволит продвинуть уровень качества медицинского обслуживания на новые высоты. Данные системы могут стать серьезным подспорьем в руках врачей различных направлений.

Системы интеллектуальной диагностики не ограничиваются какой-либо конкретной областью медицины и могут быть применены во множестве случаев. С помощью технологий анализа изображений возможно выявление

проблем с кожей. Дерматологи могут получать более подробную статистику на основе обработанных снимков и выбирать наиболее эффективные стратегии для лечения.

Наиболее хорошо такие системы могут проявить себя в выявлении заболеваний по самым неочевидным признакам. В статье «Искусственный интеллект в медицине» [1] упоминается, что даже самые опытные врачи далеко не всегда могут уверенно поставить диагноз, из-за невозможности оценить полную картину заболевания. Известны нередкие случаи неверных диагнозов, приведшие к ухудшению состояния пациентов.

Экспертные медицинские системы способны оценить все имеющиеся данные пациента, все его симптомы, после чего предоставить различные способы и методики лечения. Некоторые подобные системы даже способны консультировать больного и выдать ему рекомендации по лечению.

На этом применение искусственного интеллекта в среде медицины не ограничивается. Так как компьютеры способны обрабатывать немислимые объемы данных, с их помощью появляется возможность решать проблемы разработки и создания новых химических препаратов. Как указано в еще одной статье [2], на данный момент уже существуют и активно применяются системы по разработке лекарственных препаратов, которые способны при этом давать врачам рекомендации на основе имеющихся данных и даже обучить их новым навыкам. Компании применяют технологии глубокого обучения для анализа молекул медпрепаратов и дальнейшей оценки их пригодности. С 2019 года различные корпорации одна за другой начинали запуск новых технологий для поиска способов лечения COVID-19 и выработки вакцины в условиях пандемии. В ходе разработки применялись технологии анализа изображений. В истории человечества также уже есть опыт создания вакцин целиком разработанных средствами искусственного интеллекта без участия человека в процессе.

Благодаря развитию технологий применения интеллектуальных систем перед медициной открываются новые возможности удаленного контроля

состояния пациентов. Так, в работе Шахназарова Б. А. [3] описывается возможность применения ИИ для удаленного сбора данных больного с помощью оценки изображения и данных с датчиков для своевременной консультации через сеть интернет.

Внедрение подобных технологий имеет множество преимуществ, описанных в статье [4], которые положительно скажутся как на эффективности медицинских учреждений, так и на развитии всей медицины в целом. Тем не менее, требуется учитывать и потенциальные проблемы и риски связанные с внедрением искусственного интеллекта в жизнь человека, особенно в области медицины. Так как работа ведется со здоровьем реальных людей, а потенциальная ошибка способна привести к неисправимому исходу, очень важно гарантировать надежность не только разрабатываемых систем, но и данных [5], на которых они будут обучены. К сожалению, в сборе данных иногда невозможно избежать влияния человеческого фактора. Совершаемые ошибки могут создавать ложные выборки, что в свою очередь может сделать всю систему не просто неактуальной, а скорее даже вредоносной.

Таким образом, можно выделить основные характеристики определяющие основу хорошей системы: возможность адаптации к меняющимся условиям, способность находить оптимальные решения, актуальность и точность данных в основе обучения и обеспечение безопасности решений для конечного потребителя (пациента).

Проблема обработки данных занимает далеко не последнее место в сфере интеллектуальной диагностики состояния здоровья человека. Помимо важности обучения самих моделей искусственного интеллекта и выбора подходящих методов обучения существует необходимость серьезнее подойти к самому материалу обучения, что станет основой для результатов и решений, выданных системой.

Данные, которые поступают на вход для обучения модели, также обладают собственными характеристиками, будь то наличие или отсутствие

аномальных записей, известных как выбросы, подчинение данных нормальному закону распределения, сбалансированность классов или наличие пропущенных данных в признаках выборки. Все эти характеристики несут в себе не только научно теоретический смысл, но и обладают реальным влиянием на весь дальнейший процесс работы интеллектуальной системы, связанный с предсказанием результатов. Подобные свойства данных формируются естественным путем во время их сбора на медицинских предприятиях, а выявляются они уже в ходе статистической обработки посредством применения различных алгоритмов обработки для определения ключевых метрик выборки. Задача специалиста, разрабатывающего интеллектуальную систему, заключается не только в умении применять технологии самого искусственного интеллекта, но и в умении провести качественную предобработку данных, чему будет посвящено большое внимание в ходе выполнения данной работы. В целом, задача обработки данных и по сей день является одной из ключевых проблем в сфере машинного обучения.

Актуальность разрабатываемой системы напрямую зависит от гибкости подхода специалиста к её разработке, знаний известных методов решения существующих проблем и умения применять необходимые методы в контексте поставленной задачей. Прогресс никогда не стоит на месте и ежегодно, если не ежемесячно, происходит обновление доступных инструментов и выявление новых способов подхода к имеющимся проблемам, что говорит о высоком уровне развития всей индустрии компьютерных технологий и машинного интеллекта в целом. Однако это также и повышает планку требований к самому специалисту, который обязан уметь адаптироваться под постоянно изменяющуюся среду и обновлять свой багаж знаний, чтобы сохранять собственную актуальность.

1.2 Обзор существующих систем

Для того чтобы разработать по-настоящему качественную систему, разработчику требуется не только проявить свои навыки, как программиста

или дизайнера, но и при этом, быть уверенным, что разрабатываемая им система будет востребована. Разрабатываемой интеллектуальной системе необходимо быть достаточно актуальной в условиях конкурентной среды, а значит, что решения должны быть достаточно оригинальными, либо перекрывать недостатки уже имеющихся систем.

Так как в условиях научно-исследовательской работы студента создать полноценную систему, способную конкурировать с «гигантами индустрии» будет затруднительно, то и постановка задачи и поиск аналогов должны соответствовать текущим возможностям. В качестве обзриваемых аналогов будут рассматриваться диагностические и консультационные системы доступные рядовому пользователю сети интернет.

Среди таковых были найдены основные представители своего класса:

- WebMD – сайт предоставляющий информацию о заболеваниях, симптомах и лечении от специалистов,

- K Health – приложение, которое использует ИИ для анализа большой базы данных различных случаев пользователей, для предоставления советов и консультации,

- Ada Health – мобильное приложение, использующее искусственный интеллект для оценки симптомов пользователей и предоставления решений по заболеваниям.

Для того чтобы сравнить основные качества данных сервисов с разрабатываемой системой, был проведен анализ их качеств и составлена система субъективных оценок. Все данные будут представлены в виде таблицы с различными свойствами и субъективной оценкой для каждой из систем.

В ходе сравнения будут рассматриваться основные качества систем, связанные с обработкой данных для работы интеллектуальных моделей системы и получения результата. Учитывается подход к взаимодействию с пользователем, скорость получения результата и способность системы

адаптироваться под новые проблемы и задачи, которые обязательно возникнут в будущем. Результаты проведенного анализа представлены в табличном формате для простоты восприятия (Таблица 1).

Таблица 1 – Сравнение характеристик аналогов

Критерии оценки	WebMD	K Health	Ada Health	Разрабатываемая система
Обработка информации	9	7	8	8
Обучение системы	6	9	8	8
Автономность системы	5	7	6	8
Интерактивность системы	7	6	7	7
Решение проблем	9	8	7	7
Возможность к адаптации	5	9	7	9
Этика и безопасность системы	8	8	7	8
Функциональное разнообразие	7	6	8	7
Простота и удобство использования	5	7	7	9
Быстрота отклика системы	6	8	8	10

Как можно заметить, разрабатываемая система необязательно превосходит аналоги по всем возможным параметрам, однако обладает своими собственными преимуществами.

1.3 Подведение итогов анализа

Система «WebDM» – это классическая экспертная система медицинской диагностики, в основе которой является взаимодействие пациента с реальными врачами. Реальный врач способен более человечно подходить к вопросам постановки диагноза, однако количество врачей ограничено, из-за чего скорость получения ответа может быть сильно замедлена.

Приложения «K Health» и «Ada Health» уже базируются на первичном взаимодействии с пользователем через ассистента на базе искусственного интеллекта. Такие системы могут предоставить куда более быстрый ответ и иногда выявить такие проблемы, которые бы настоящий врач мог и не заметить.

Разрабатываемая система предполагает использование искусственного интеллекта для основного взаимодействия с пациентом. Главной задачей в работе такой системы является выдача рекомендаций на основе медицинских данных, решение простейших задач, связанных со здоровьем. В качестве основной цели ставится не самая высокая точность оценки, а удобство навигации в самом интерфейсе приложения и скорейшее получение результатов анализа для выявления у пользователя проблем со здоровьем по полученным данным.

Каждая из выше описанных систем обладает своими достоинствами и недостатками, и это нормально, ведь создание идеальной системы, соответствующей высочайшим стандартам качества по всевозможным критериям является задачей нереалистичной. Поэтому в ходе разработки данной системы целью было решено добиться наибольшей простоты для пользователя при взаимодействии с системой и наискорейшее получение простого решения.

1.4 Вывод по разделу

В результате был проведен анализ предметной области и дана характеристика текущему состоянию интеллектуальных систем в сфере медицины. В ходе анализа было выявлено, что текущее состояние индустрии не находится в финальной точке развития и многие текущие решения подлежат дальнейшей доработке.

Разработка системы интеллектуальной диагностики призвана дополнить рынок различных сайтов и приложений, за счет устранения отдельных недостатков уже имеющихся систем. Системы интеллектуальной диагностики являются вечно актуальной темой в сфере медицины. Возможно, данное приложение сможет привнести свой вклад в развитие экспертных систем в области медицины и сделает возможным более доступное получение диагноза и советов относительно состояния здоровья людей.

2 Исследование технологий решения поставленной задачи

2.1 Введение в проблему

Разработка системы интеллектуальной диагностики состояния человека

– это комплексный процесс, сочетающий в себе несколько важных аспектов:

- поиск и предварительная обработка данных,
- Выбор архитектуры обучаемой модели,
- Оценка качества предсказаний,
- Интеграция модели в приложение.

2.1.1 Поиск и предварительная обработка данных

Прежде чем выбирать какой-либо известный метод машинного обучения стоит обратить внимание на то, что будет лежать в основе самого процесса обучения. Выбор самих данных будет зависеть от конкретно поставленной задачи машинного обучения, будь то выявление потенциальных заболеваний или определение типа болезни. Однако вне зависимости от того, какую задачу мы решаем с помощью методов машинного обучения, проблема качества данных всегда будет иметь высокий приоритет, ведь правильная обработка данных определяет то, как хорошо будет проходить процесс обучения.

В проанализированной статье [6] упоминается важность правильного подхода к определению качества самих данных, возможных проблемах связанных с ними и тому, как эти проблемы решать.

Выясняется также и то, что игнорирование этих проблем может привести к сильному ухудшению качества работы модели, что недопустимо в такой важной индустрии, как медицина, ведь в случае ложного предсказания может быть нанесен вред здоровью людей или даже их жизнь может оказаться под угрозой. Рассмотрим ключевые моменты и обозначим важные детали предварительной обработки данных:

2.1.2 Очистка данных, удаление аномалий и обработка пропусков

Данный этап является одним из ключевых в предварительной обработке данных. Он ставит задачу избавиться выборку от тех данных, что сильно выбиваются из общей группы и только мешают процессу обучения.

Выбросы или аномалии представляют собой редкие наблюдения, что сильно отличаются от остальной массы данных и не соответствуют общей картине. Как правило, это отдельные редкие частные случаи, которые часто еще и являются исключением из общего набора правил. Они недостаточно часто происходят, из-за чего предсказывать их модели будет тяжело, и в большинстве своем они только будут мешать процессу обучения, “сбивая” модель с правильного направления.

Обработка пропусков – это непростой процесс, требующий индивидуального подхода. Где-то пропуски были созданы намеренно, где-то данные не удалось получить, или они были потеряны в результате некой ошибки. Решение подобных проблем ложится на специалиста, что будет с этими данными работать. Самым простым методом решения проблемы пропущенных данных является банальное удаление строк, где эти пропуски присутствуют. Подобный метод прост и понятен, однако не всегда оправдан или вовсе может оказаться неприемлем. В таком случае прибегают к попыткам эти пропуски заполнить.

Заполнение пропусков может осуществлять по нескольким принципам:

- заполнение нулем или средним значением,
- более устойчивое к выбросам заполнение медианой или модой,
- заполнение методом ближайших соседей – данные восстанавливаются по похожим наблюдениям,
- Предсказание пропусков с помощью модели.

Выбор конкретного метода может быть обоснован особенностью задачи или самих данных. Предпочтение в медицинских задачах часто отдается заполнению по значению медианы по группам пациентов со схожими симптомами. Это помогает сохранить общую характеристику

группы и минимизирует риск того, что в данные будет включена систематическая ошибка.

2.1.3 Отбор ключевых признаков

Данные, которые собираются в медицинских предприятиях могут хранить в себе множество отдельных параметров, которые могут иметь связь с результатом обследования. Однако ничто не дает гарантию, что абсолютно все признаки, которые были собраны, являются полезными для обучения модели.

Это важная задача, неправильное решение которой может серьезно сказаться на качестве финальной выборки. В ходе получения практических навыков во время обучению курсу машинного обучения было выявлено, что далеко не всегда увеличение числа признаков приравнивается к улучшению работы модели. Также стоит упомянуть, что слишком большое число признаков может негативно сказаться на времени проведения процесса обучения, что может оказаться неприятным фактором в случае решения задачи требующей тяжелых вычислений.

Самый простой и понятный визуально метод оценки полезности отдельных признаков – это метод проверки корреляции по целевому признаку. Он показывает, насколько сильна статистическая связь между выбранной характеристикой и результатом наблюдения. Если коэффициент связи высокий, то можно предположить, что с большой вероятностью, эти два признака зависят друг от друга.

Однако данный метод не является идеальным, ведь игнорирует потенциальное взаимодействие между несколькими признаками. Существуют связи свойств, которые подчиняются неочевидному закону. Подобные наборы свойств можно определить различными методами. Наиболее выделяющимся из них во время получения практических знаний оказался метод обертки.

Данный метод подразумевает подбор наиболее подходящего подмножества признаков из общего набора. Метод рекурсивного исключения

признаков проводит обучение модели на каждой итерации работы алгоритма и поочередно исключает наименее важные признаки до тех пор, пока порог точности обучения не будет достигнут или нужное число признаков не будет отобрано.

Возможен и обратный сценарий, где число признаков поочередно добавляется в набор. Выбираются те признаки, добавление которых лучше всего сказывается на качестве работы модели.

2.1.4 Проблема балансировки данных

Несбалансированность данных может привести к тому, что модель не сможет хорошо ориентироваться в случаях связанных с определением результата по меньшему классу. Пониженная чувствительность повышает риск ложного предсказания результата. При недостаточном числе объектов меньшего класса модель может просто плохо обучиться обобщать результат на новых данных.

Одним из относительно простых методов решения данной проблемы является применение методов сэмплинга данных. Если объемы выборки позволяют, то проводится уменьшение числа объектов наибольшего класса с сохранением его ключевых черт. Однако иногда такой подход может не подойти. В таком случае производится обратное действие – число объектов меньшего класса синтетически генерируется до тех пор, пока баланс классов не будет достигнут.

2.2 Выбор архитектуры обучаемой модели

Когда данные подготовлены, нужно решить, какую модель выбрать для нашей задачи. Существует множество различных моделей подходящих под данную ситуацию описанных в проанализированном мною источнике [7].

Случайный лес. Он представляет собой ансамбль из множества деревьев решений, где финальный результат выбирается путем голосования, в случае решения задачи классификации, или усреднения, если стоит задача решить регрессию. При таком подходе модель должна демонстрировать

высокую устойчивость к переобучению, что является одним из важных моментов при применении искусственного интеллекта.

Метод опорных векторов позволяет решить задачу классификации, образуя гиперплоскость разделяющую классы. Подход эффективен при работе с выборками больших размеров, однако довольно чувствителен к настройке и обладает ограниченной интерпретируемостью, из-за чего может не всегда подойти для решения отдельных задач.

Логистическая регрессия – линейный алгоритм, позволяющий определить вероятность принадлежности объекта к тому или иному классу. Несмотря на свою линейность и ограниченную способность улавливать черты отдельных классов, логистическая регрессия остается популярным методом в силу простоты обучения.

Нейронная сеть – это многослойная модель способная выявлять сложные нелинейные законы в данных за счет многочисленных преобразований, производящихся последовательно. Модель формируется из нескольких слоев, количество которых выбирается относительно поставленной задачи, у нее высокое количество настраиваемых параметров, что позволяет подвести эту модель под практически любую ситуацию. Однако это же и становится камнем преткновения при работе с ней, ведь для достижения баланса между высокой точностью и скоростью обучения может потребоваться немалое количество попыток, прежде чем хоть какой-то удовлетворительный результат будет достигнут.

Данную модель практически невозможно интерпретировать, так как она работает по принципу черного ящика. Не то чтобы ее нельзя было разобрать на составляющие, однако для анализа ее конечной работы, может потребоваться невообразимо больше времени, чем для реализации этой модели. Тяжелые вычисления в ходе процесса обучения компенсируются высокой точностью работы. Выбор данного метода может оказаться спорным в случае анализа медицинских данных, однако он же, пожалуй, является и самым интересным как в плане реализации, так и в плане потенциала.

2.3 Оценка качества предсказаний

Для того чтобы понять какой из методов нам подходит, их нужно как-то сравнивать друг с другом. Также важно понять насколько качественным вышел результат в целом.

Для того чтобы оценить качество модели, существует ряд метрик, предназначенных как раз для описания результатов ее работы. Кроме очевидного процента правильных предсказаний, существует еще матрица ошибок модели. Она представляет собой таблицу, столбцы которой описывают то, что предсказала модель, а строки – то, что было на самом деле. Такая простая структура позволяет наглядно сказать, сколько реальных классов модель предсказала верно. С первого взгляда, можно не понять, в чем здесь отличие от процента точности предсказания, но суть кроется в возможности увидеть распределение неправильных предсказаний, ведь может выйти так, что все классы из первой категории модель оценивает правильно, но при этом в половине случаев ошибается с классом под номером два. Такая оценка может позволить не только описать качество работы модели, но и потенциально указать на проблемы в самих данных.

Полнота – данная метрика оценивает, насколько хорошо модель способна выявить болезнь. Процент берется по числу верно предсказанных случаев от суммы этих случаев и случаев, когда заболевание не было выявлено. Данная метрика игнорирует случаи, когда здорового человека назвали больным. Изначально это может показаться нелогичным, однако, в случае задачи обнаружения потенциального заболевания будет лучше послать здорового человека на обследование к врачу в качестве профилактики, нежели пропустить того, кто болен по-настоящему.

В обратной ситуации, когда пациента не стоит лишний раз беспокоить, важность приобретает метрика точности. Она оценивает, какая доля положительных предсказаний сбылась. По сути, метрика описывает, насколько уверенно модель может выдавать диагноз «болен».

AUC-ROC характеристика [8] или «Area Under the ROC Curve» представляет собой метрику, описывающую способность модели отличать один случай от другого при разных порогах классификации. Чем ближе значение этой характеристики к единице, тем лучше модель способна отличить один класс от другого. Значение, равное 0.5 говорит о том, что модель угадывает классы случайно. Если значение ниже 0.5, то это уже говорит о том, что в процессе обучения модели произошла ошибка и модель систематически ошибается. Полезно для выявления ошибок при применении или разработке модели.

В итоге многие метрики оценки качества работы модели позволяют с разных сторон оценить результат проделанной работы, выявить более и менее эффективные стратегии. Однако стоит упомянуть, что в такой индустрии, как медицина, конечное решение о пригодности модели к использованию должно всегда совершаться медицинскими экспертами, а не процентами и красивыми числами.

2.4 Интеграция модели в приложение

Завершающим этапом разработки данной системы является реализация и введение ранее подготовленной модели в само приложение. Модель должна быть обучена, после чего подготовлена к транспортировке в финальную программу, где она уже будет эксплуатироваться.

На этом этапе выбор конкретного подхода уже ложится на разработчика. Модель может быть предварительно обучена на имеющихся данных с помощью более мощной машины, которая сможет эффективно и быстро запустить все процессы обработки данных и обучения. Либо если таково будет поставленное требование, само приложение в своем финальном виде может содержать модуль для обучения модели в любое время на новых данных. Это даст приложению большую гибкость, однако создаст дополнительную нагрузку на разработчика, а также повысит требования к оборудованию, на котором будет проводиться анализ. Одним из возможных вариантов решения данной проблемы может оказаться вынос самой модели

на внешний сервер, где будут проводиться тяжелые вычисления, однако тогда уже это сделает клиентскую часть приложения зависимой от данного сервера.

Как можно понять, не существует единственно верного решения, и выбранный вариант будет зависеть от конкретной ситуации и поставленной задачи. Одним из наиболее компромиссных вариантов будет обучить модель заранее на мощном сервере, после чего экспортированная она уже будет загружена на целевое устройство, где будет применена в прямой прогонке, то есть без возможности обучаться. При необходимости обучить модель заново процесс может быть повторен.

2.5 Вывод по разделу

В результате рассмотрения существующих методов решения поставленной задачи были сформированы общие представления о дальнейшем направлении разработки выбранной системы. Рассмотренные ранее примеры являются широко известными средствами, применяемыми в процессе машинного обучения. Средства обработки данных и оценки качества обученных моделей помогут выбрать наилучший подход для решения поставленной задачи.

3 Разработка структуры системы

Интеллектуальная система диагностики состояния человека подразумевает под собой применение технологий машинного обучения для выдачи предварительных медицинских показаний по результатам пройденных обследований.

Даже при такой формулировке тема все еще является слишком обширной, а финальный образ программы остается размытым и неясным. Поэтому для внесения ясности в данную работу, стоит привести более конкретную формулировку задачи, которую можно было бы выполнить в рамках магистерской работы.

Хорошим примером для демонстрации работы алгоритмов и всего процесса разработки в целом станет выбор решения задачи бинарной классификации по выявлению заболеваний. Для того чтобы показать работу модели и оценить её работоспособность, стоит выбрать конкретную проблему в сфере медицины.

Такой проблемой стало выявление сердечных заболеваний у пациентов. Для того чтобы результат, полученный в ходе работы обладал реальным весом и значимостью, стоит с умом подойти к выбору данных, которые будут анализироваться. Мой выбор в этой работе пал на датасет Сердечных Заболеваний [9] находящийся в свободном доступе. Данные были собраны в конце 20-го века и содержат в себе множество столбцов, каждый из которых отвечает за отдельную графу при прохождении обследования. Целевым столбцом обозначается наличие заболевания у пациента. Данная выборка является известным общедоступным ресурсом, который хорошо подходит для тренировки и демонстрации навыков специалиста в разработке интеллектуальных систем.

Теперь, когда цель намечена, стало возможно сформулировать полноценный текст задачи, которую предстоит выполнить. Данная работа, направленная на разработку интеллектуальной системы наблюдения за состоянием здоровья человека ставит целью выявить потенциальное наличие

сердечного заболевания у пациента по данным показаниям. Для обучения модели используется набор данных из разных медицинских учреждений, проводивших медицинское обследование.

Нужно разработать программу для предварительной обработки данных, оценки качества работы моделей и сравнения алгоритмов обучения. Требуется обучить модель на имеющейся выборке. Обученную модель нужно интегрировать в финальное приложение. Приложение должно предоставлять результат предсказания по данным, введённым пользователем в предоставленной форме.

Данная формулировка задачи, является достаточно ясной, чтобы её можно было реализовать. Теперь, когда задача поставлена, можно переходить к следующему этапу планирования.

3.1 Выбор методов и инструментов. Обоснование архитектуры системы

В данном этапе предстоит выбрать основные методы и инструменты для решения поставленной задачи. Ранее в ходе выполнения исследовательской работы были рассмотрены самые известные модели для решения задачи машинного обучения. Теперь предстоит сравнить каждый из этих алгоритмов. Требуется понять, какой именно алгоритм обладает наибольшим потенциалом для решения поставленной задачи.

3.1.1 Выбор алгоритмов и критериев оценки

Так как без практической оценки работы алгоритмов нельзя сказать, какой из них окажется эффективнее, нужно рассмотреть сразу несколько различных видов классификаторов для решения данной задачи и провести предварительный анализ их эффективности на основе знаний полученных в ходе ознакомления с прочтёнными источниками.

Основными кандидатами на роль модели для предсказания наличия сердечных заболеваний стали:

- случайный лес,

- метод опорных векторов,
- логистическая регрессия,
- многослойный персептрон.

Каждый из данных представителей обладает уникальными чертами, отличающими его от других.

Разберем основные критерии сравнения алгоритмов: Для более наглядного представления информации каждый из критериев и его описание сведен в общую таблицу.

Таблица 2 – Критерии сравнительного анализа

Критерий	Описание
Точность классификации	Способность модели определять наличие или отсутствие сердечного заболевания
Полнота	Способность модели не упустить больных пациентов
Устойчивость к переобучению	Модель должна не запоминать результат, а обучаться, чтобы правильно предсказывать даже те случаи, к которым она не готовилась
Скорость обучения	Показатель, отвечающий за то, сколько понадобится времени модели прежде, чем она сможет выдать результат
Скорость разового ответа	Время ожидания предсказания для одного пользователя
Устойчивость к шуму и выбросам	Несмотря на предварительную обработку аномалий в данных, модель все равно может подвергнуться их влиянию. Модели необходимо корректно работать, не смотря на наличие шума в данных

После того как общее представление о критериях получено, можно приступить к конкретной оценке каждого из имеющихся алгоритмов.

3.1.2 Сравнительная характеристика каждого из алгоритмов

Случайный лес – ансамбль деревьев решений, где множество деревьев предсказывают результат, который будет выбран голосованием большинства.

Достоинства классификатора:

- обладает достаточно высокой точностью классификации на уровне нейронных сетей,
- алгоритм устойчив к переобучению в силу большого числа деревьев,
- возможность оценки важности признаков благодаря особенностям принципов построения дерева.

Недостатки:

- с числом и глубиной деревьев размер модели может стать по-настоящему большим
- интерпретировать сложнее, чем одно единственное дерево,
- возможна необходимость подбора оптимального числа параметров модели (глубина и число деревьев),
- метод опорных векторов – метод построения гиперплоскости, разделяющей классы.

Достоинства:

- довольно высокая точность на некрупных выборках
- метод устойчив к переобучению модели
- настройки позволяют добиться более качественного результата

– недостатки:

- чувствительность к настройке параметров
- плохо масштабируется на выборках больших размеров
- сложная интерпретация
- чувствительность к шуму и аномалиям данных

Логистическая регрессия – линейный алгоритм классификации, подсчитывающий вероятность принадлежности объекта к конкретному классу.

Достоинства:

- простота интерпретации – показатели весов отвечают за важность самих признаков,
- быстрое обучение модели и получение предсказания,
- устойчивость к переобучению благодаря параметрам регуляризации.

Недостатки:

- модель неспособна уловить сложные нелинейные зависимости,
- выбросы и аномалии могут повлиять на качество работы.

Многослойный персептрон – базовая нейронная сеть, состоящая из входного, скрытых и выходного слоев.

Достоинства:

- способность предсказывать любые сложные зависимости,
- высокая точность на хорошей выборке,
- гибкость настройки.

Недостатки:

- низкая, почти нулевая интерпретируемость,
- склонность к переобучению,
- высокая чувствительность к настройке,
- высокая вычислительная нагрузка, замедляющая обучение.

3.2 Выбор инструментов разработки

Методы машинного обучения – это достаточно обширная и известная тема. Многие принципы здесь строятся на чистой математике и статистике. Есть множество способов реализовать каждый из возможных этапов, вплоть до воссоздания их вручную.

Однако это крайне трудозатратно и больше подходит для глубокого изучения каждого аспекта данной задачи. Каждый из этапов строится на

множестве трудов и наук, построенных не за один год. На освоение их всех ушло бы куда больше времени, чем есть у магистра на его сроке обучения.

К счастью многих студентов, существует множество инструментов, дающих возможность воспользоваться уже готовыми реализациями всех этих алгоритмов. В моем случае выбор набора инструментов определен опытом и программой обучения. Поэтому в качестве инструментов для реализации алгоритмов работы с обучением моделей для предсказания был выбран обширный набор библиотек языка Python.

Данный язык является сценарным языком, код которого выполняется сразу без компиляции, а количество доступных модулей и библиотек поражает воображение.

Так для загрузки и представления данных использоваться будет библиотека Pandas. Данная библиотека позволяет загружать табличные выборки данных в удобном представлении в виде ДатаФреймов. Библиотека располагает прекрасным набором функций для работы с этими данными и их модификацией.

Для визуального отображения данных отлично подходит не менее известная библиотека Matplotlib, в наборе инструментов которой есть десятки различных графиков для визуального представления информации на экране пользователя. Все результаты обработки и процесса обучения легко отобразить на экране, будь то гистограммы, линейные графики или инструменты отображения статистических данных.

И, наконец, библиотека SKlearn, что предоставляет невообразимое множество инструментов машинного обучения в одном наборе. Какой бы метод классификации не появился бы на уме, он уже есть в этом наборе. Данная библиотека станет ключевым инструментом в решении задачи определения наличия болезни у пациентов.

Для реализации конечного приложения, где будет уже эксплуатироваться рабочая обученная модель, был выбран метод

взаимодействия с моделью посредством вызываемых окон, которые будут представлять собой графический интерфейс приложения.

3.3 Проектирование структуры системы

Данная система будет состоять из трех основных модулей. Первый представляет собой модуль для загрузки и обработки данных выборки для обучения.

В задачи данного модуля входит загрузка данных датасета. Статистический анализ особенностей и выявление возможных проблем данных. Обработка пропусков и различных аномалий.

Так как не все алгоритмы обучения устойчивы к шуму, стоит серьёзно подойти к вопросу очистки данных. Далее следует вопрос кодировки данных. Многие параметры могут быть записаны в текстовом формате, для того, чтобы с ними могли работать алгоритмы машинного обучения, требуется преобразовать эти данные в числовые значения, которые куда ближе к восприятию алгоритмами.

В конце обработки данным предстоит быть нормализованными. Данный процесс очень важен и сильно влияет на качество обучения, ведь различные параметры могут находиться в слишком разных числовых интервалах. Нормализация решает данный вопрос. В конце выборка разбивается на наборы для обучения и проверки.

Следующим на очереди находится модуль обучения самой модели. Его задача заключается в том, чтобы подготовить будущую модель для эксплуатации. Показать результаты проверок и сохранить ее для транспортировки в финальное приложение.

Роль данного модуля очень важна — каждый из используемых алгоритмов требует собственного подхода. Поэтому требуется качественно подойти к настройке используемых методов обучения, чтобы достичь высоких результатов точности работы модели.

В конце всего технологического процесса лежит финальная, эксплуатационная стадия. Приложение загружает уже обученную модель и на основе ввода пользователя совершает предсказания.

На данном этапе разработки предстоит решить проблему графического представления приложения. Разобраться с тем, как загрузить и использовать каждую модель. Необходимо предоставить пользователю возможность ввести все ключевые данные для анализа, отобразить результат предсказания модели, предоставить необходимую интерпретацию результатов и выдать соответствующую рекомендацию на основе выхода работы модели.

Обобщенная схема работы системы представлена далее (Рисунок 1).

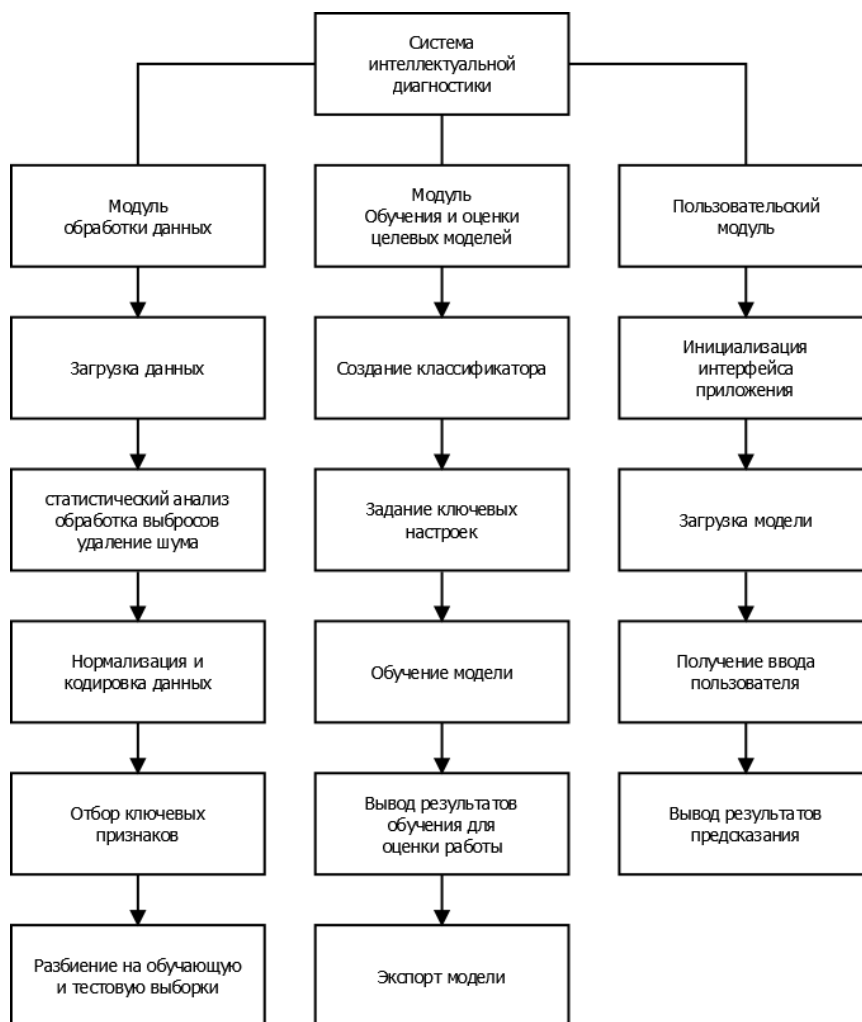


Рисунок 1 – Составная схема системы

3.4 Вывод по разделу

Работа данных трех модулей описывает полный жизненный цикл работы системы интеллектуального наблюдения за состоянием человека. Каждый модуль преследует уникальную и выполняет поставленную задачу. Результатом работы трех модулей является полноценное функционирование системы.

В ходе анализа используемых инструментов было сформировано представление о том, как именно должна быть устроена система. Разделение системы на несколько модулей позволяет упростить разработку, привносит логику в общую структуру, позволяет четко выделить ключевые свойства каждого модуля.

Структура системы позволяет привносить изменения в её работу, путем добавления новых методов реализации алгоритмов машинного обучения, позволяет обработать различные наборы данных без необходимости внесения капитальных изменений в код программы. Благодаря простоте и удобству средств разработки Python внесение новых функций не требует больших усилий со стороны программиста.

4 Программная реализация системы

Данный раздел посвящен практической реализации системы для интеллектуальной оценки состояния здоровья человека. Ключевой задачей системы является выявление возможного наличия заболеваний у пациента.

Для разработки системы используется среда разработки Python. Для более удобного и наглядного выполнения программы код поделен на блоки в файлах отчетов irunb. Данный формат позволяет выполнить отдельные блоки кода по очереди, чтобы в случае необходимости внесения правок не пришлось заново запускать весь цикл работы системы.

4.1 Обработка данных

Данный процесс подразумевает загрузку датасета и его очистку. В изначальном виде датасет представляет собой набор множества признаков и объектов (Рисунок 2).

```
df=pd.read_csv('../heart.csv', delimiter=',')
```

df

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3	0
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3	0
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3	0
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3	0
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2	0

303 rows × 14 columns

Рисунок 2 – Загрузка данных

В исходном виде нам предоставляется выборка с данными, состоящими из множества признаков. Всего в выборке находится 303 записи о пациенте и 14 столбцов, каждый из которых имеет свое уникальное значение. Чтобы получить более качественное представление о показанной выборке, данные будут сведены в таблицу, описывающую ключевую информацию (Таблица 3).

Таблица 3 – Описание столбцов выборки

Имя столбца	Формат данных	Описание
age	Целочисленный	Возраст пациента
sex	Категориальный	Пол пациента: 1 – Мужчина 0 – Женщина
ср (аббревиатура Chest Pain)	Категориальный	Тип боли в груди у пациента: 1 – Типичная стенокардия 2 – Атипичная стенокардия 3 – Некардинальная боль в груди (не связана с нарушением кровотока) 4 – Бессимптомный (скрытый)
restbpps (от Resting BP)	Целочисленный	Давление в покое
chol	Целочисленный	Уровень холестерина в крови пациента

Продолжение таблицы 3

Имя столбца	Формат данных	Описание
fbс	Категориальный	Уровень сахара в крови пациента (натощак): 1 – Более или равно 120 мг/дл 0 – Менее 120 мг/дл (мг/дл – миллиграмм на децилитр)
restecg	Категориальный	Результат ЭКГ (покой): 0 – Норма 1 – Наличие аномалий комплекса ST-T (восстановление сердечной мышцы после сокращения) 2 – Потенциальная гипертрофия левого желудочка (увеличение массы мышц главной насосной камеры сердца)
thalach	Целочисленный	Максимальная частота сердечных сокращений
exang	Категориальный	Стенокардия при физической нагрузке: 1 – есть 2 – нет

Продолжение таблицы 3

Имя столбца	Формат данных	Описание
oldpeak	Вещественный	Депрессия сегмента ST при нагрузке (мышца получает меньше крови, чем положено)
slope	Категориальный	Тип наклона сегмента ST: 1 – Подъемный 2 – Прямой 3 – Спускающийся
ca	Целочисленный	Количество сосудов, что окрасились во время флюороскопии (от 0 до 3 сосудов)
thal	Категориальный	Результат стресс теста с таллием: 1 – фиксированный дефект 2 – Нормальный кровоток 3 – Обратимый дефект 4 – Не заполнено
target	Категориальный	Целевой столбец, диагноз: 0 – Заболевание отсутствует 1 – Выявлено заболевание

Как можно понять, эта выборка, содержит довольно большой набор данных медицинского обследования для диагностирования сердечных заболеваний. Столбцы описывают информацию полученную врачами в результате медицинского осмотра пациентов проводившегося в восьмидесятые годы прошлого столетия.

Данная выборка хранит полезные данные, однако пытаться использовать их в изначальном виде будет неправильно. Для того чтобы эти данные принесли пользу датасет должен быть качественно обработан.

Баланс классов выборки уже соблюден, поэтому можно не волноваться по поводу необходимости проводить синтезирование данных (Рисунок 3)

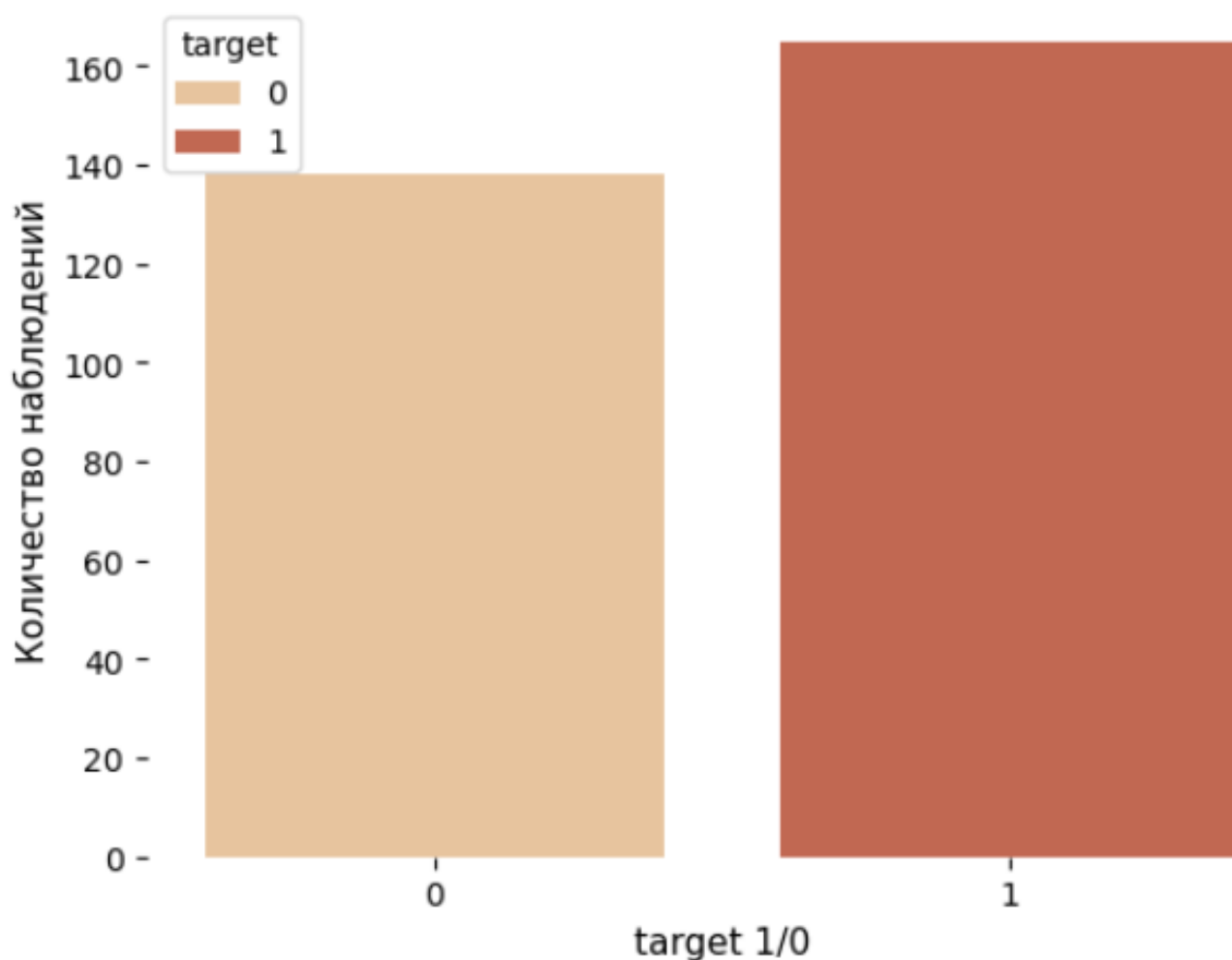


Рисунок 3 – Баланс классов (столбчатая диаграмма)

Как можно заметить, выборка в своем изначальном виде соблюдает достаточно неплохой баланс классов объектов, что в свою очередь немного упрощает работу.

Однако может случиться и ситуация, где равновесие классов будет слишком сильно нарушено. В таком случае можно использовать методы семплирования такие как: undersampling (уменьшение числа объектов у классов превосходящего размера) и oversampling (увеличение числа объектов меньших классов до размеров самого большого из них).

Оба данных метода подразумевают собой не просто случайное удаление или создание случайных объектов в выборке, а именно произведение операции семплирования, которая подразумевает изменение размеров выборки с сохранением её характерных свойств. Как указывается в проанализированном мною источнике [10], правильное использование алгоритмов семплирования способно значительно увеличить точность предсказания модели.

На рисунке 4 представлен результат работы алгоритма, который подгоняет класс меньшего размера до размера большего класса, при таком варианте соотношение классов будет один к одному.

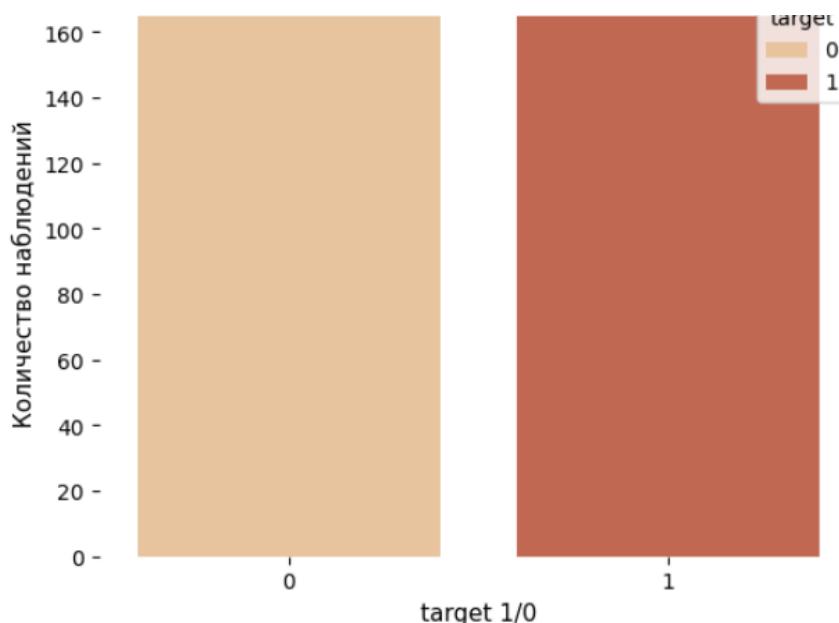


Рисунок 4 – семплирование SMOTE

Следующим важным этапом обработки будет выявление и устранение аномалий. Воспользуемся ящиком с усами для отображения выбросов.

```
max_cols = 3
total_columns = dfc.shape[1]
rows = (total_columns + max_cols - 1) // max_cols
plt.figure(figsize=(15, rows * 6))

for idx, col in enumerate(dfc.columns):
    plt.subplot(rows, max_cols, idx + 1)
    plt.boxplot(dfc[col].dropna())
    plt.title(f'Boxplot: {col}')
    plt.ylabel('Значения')
    plt.xlabel(col)
    plt.xticks([])

plt.tight_layout(pad=3.0)
plt.show()
```

Приведенный код позволяет сформировать ящик с усами для каждого из имеющихся столбцов выборки. Что позволяет быстро оценить каждый из НИХ.

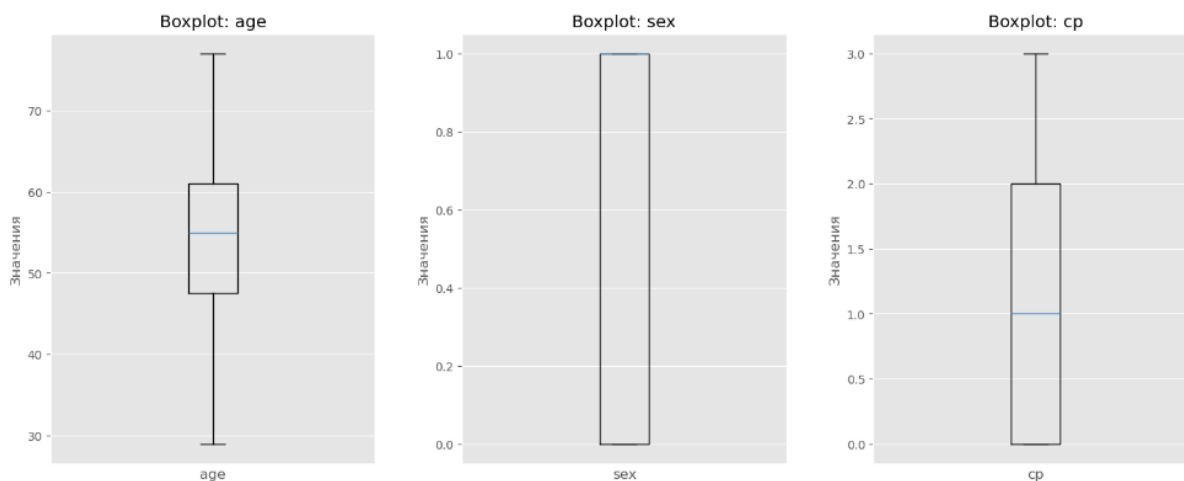


Рисунок 5 – Характеристики столбцов (1 - 3 столбцы)

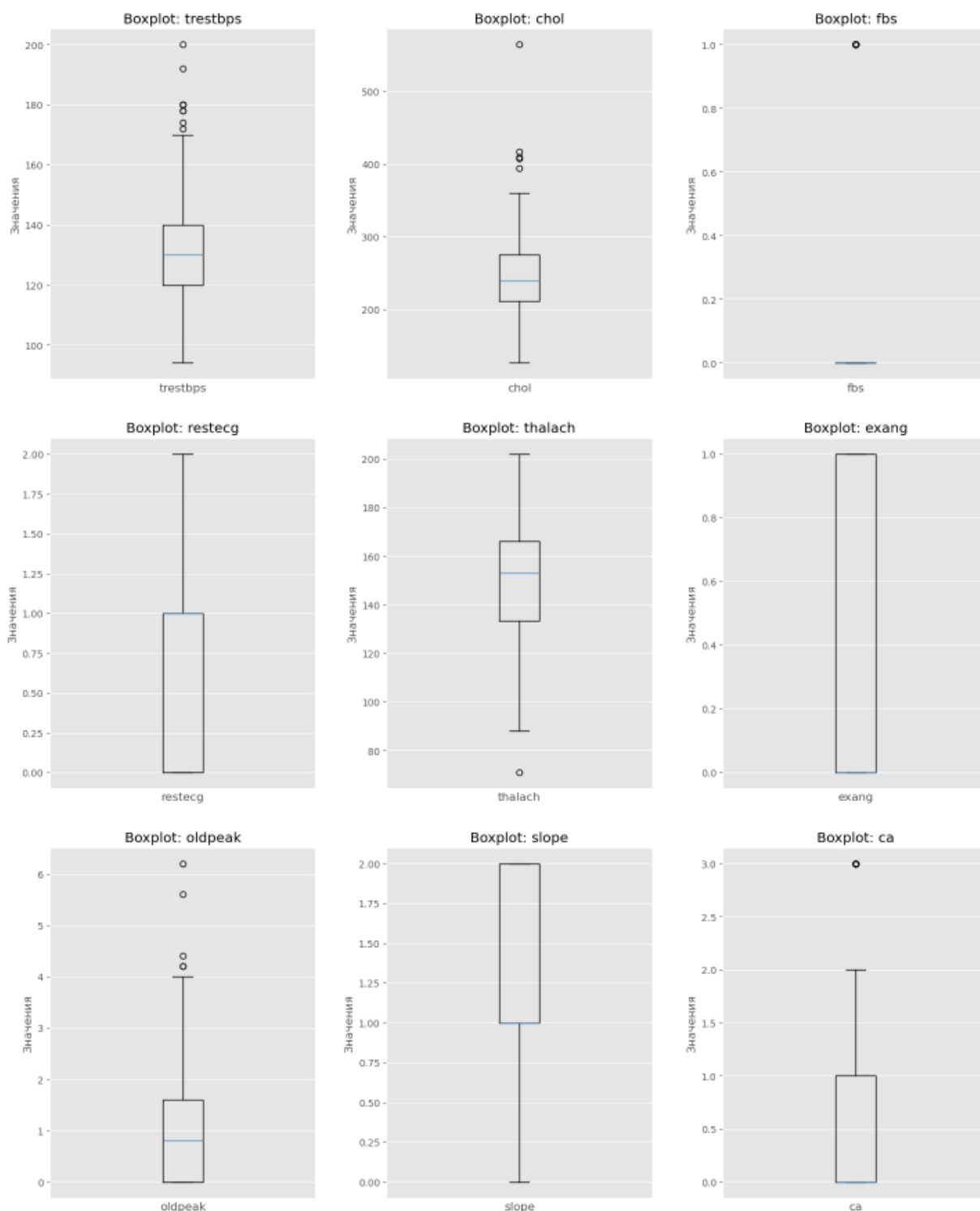


Рисунок 6 – Характеристики столбцов (4 - 12 столбцы)

На выше представленном изображении (Рисунок 6, Рисунок 6) представлены так называемые ящики с усами (на английском Boxplot). Чтобы получить пользу от указанной визуальной информации, нужно понять, как её интерпретировать. Для того, чтобы верно читать информацию, указанную на диаграмме, пришлось более подробно ознакомиться с несколькими

источниками [11, 12], в ходе анализа которых было сформировано общее представление о данном инструменте. Устройство ящиков с усами довольно просто – они предоставляют статистическую информацию о том, как распределены данные каждого конкретного признака в нашей выборке.

Для более простого восприятия данного инструмента сформируем таблицу, в которой опишем основные составляющие (Таблица 4).

Таблица 4 – Описание элементов ящика с усами

Свойство	Описание
Ящик (прямоугольная коробка)	Данный элемент описывает «сердце» выборки. Он показывает, в каком промежутке находится основная часть данных. Границы ящика обозначены 25-м и 75-м перцентилями (Иначе первый и третий квартили). Высота ящика указывает на плотность распределения данных.
Линия, находящаяся внутри ящика	Медиана выборки. 50-й перцентиль. Представляет собой «золотую середину» и делит выборку ровно на две части. Нахождение посередине ящика подразумевает равномерное распределение данных внутри него.
Усы ящика	Линии, отходящие от самого ящика. Не являются границами выборки. Показывают разброс пограничных значений. Чем короче усы, тем меньше число экстремальных значений.
Точки, находящиеся за усами	Так называемые аномалии выборки, или иначе – выбросы. Четко указывают на небольшое число записей, что сильно выбиваются из общей картины.

Как можно увидеть, ящик с усами – это замечательный инструмент, позволяющий быстро оценить состояние выборки, просто посмотрев на него. Он компактно отображает ключевые свойства данных и позволяет быстро выявить проблемные моменты.

Обратим внимание на аномалии в наших данных. Бороться с выбросами можно по-разному, однако большое количество объектов и малое число выбросов позволяют нам просто избавиться от строк, выбивающихся из общей выборки. Данный метод не всегда подходит, однако он позволяет избавиться выборку от случаев, которые чаще являются очень редкими случайностями или ошибками, нежели закономерностями, что позволит модели лучше определять общие случаи, не заикливаясь на частных.

Приведем пример очистки одного из признаков от выбросов. Для этого хорошо подойдет столбец с большим числом точек на ящике с усами (Рисунок 7).

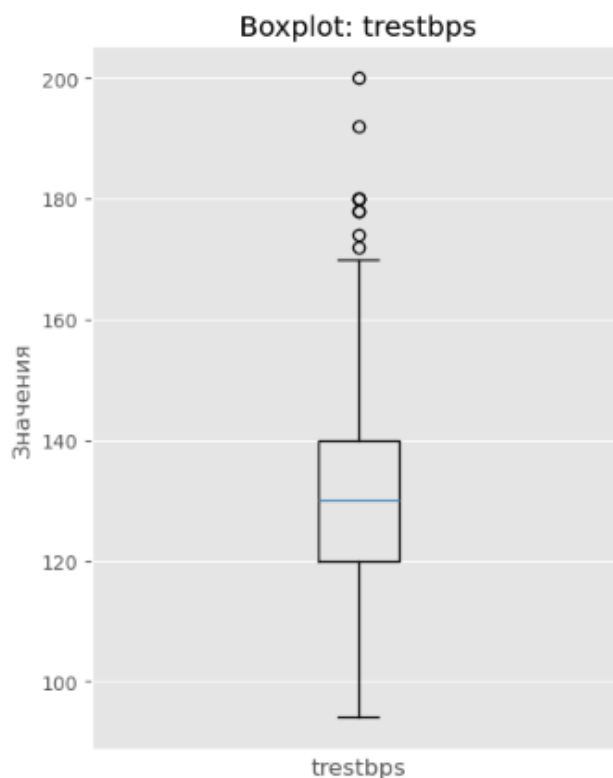


Рисунок 7 – Пример столбца с явными выбросами

На данном рисунке можно явно увидеть наличие заметного числа выбросов. Используя показания графика можно четко увидеть границу, по которой можно их отделить от общей выборки. Также для удобства оценки можно проверить диаграмму, что позволит дополнительно оценить данные аномалии (Рисунок 8).

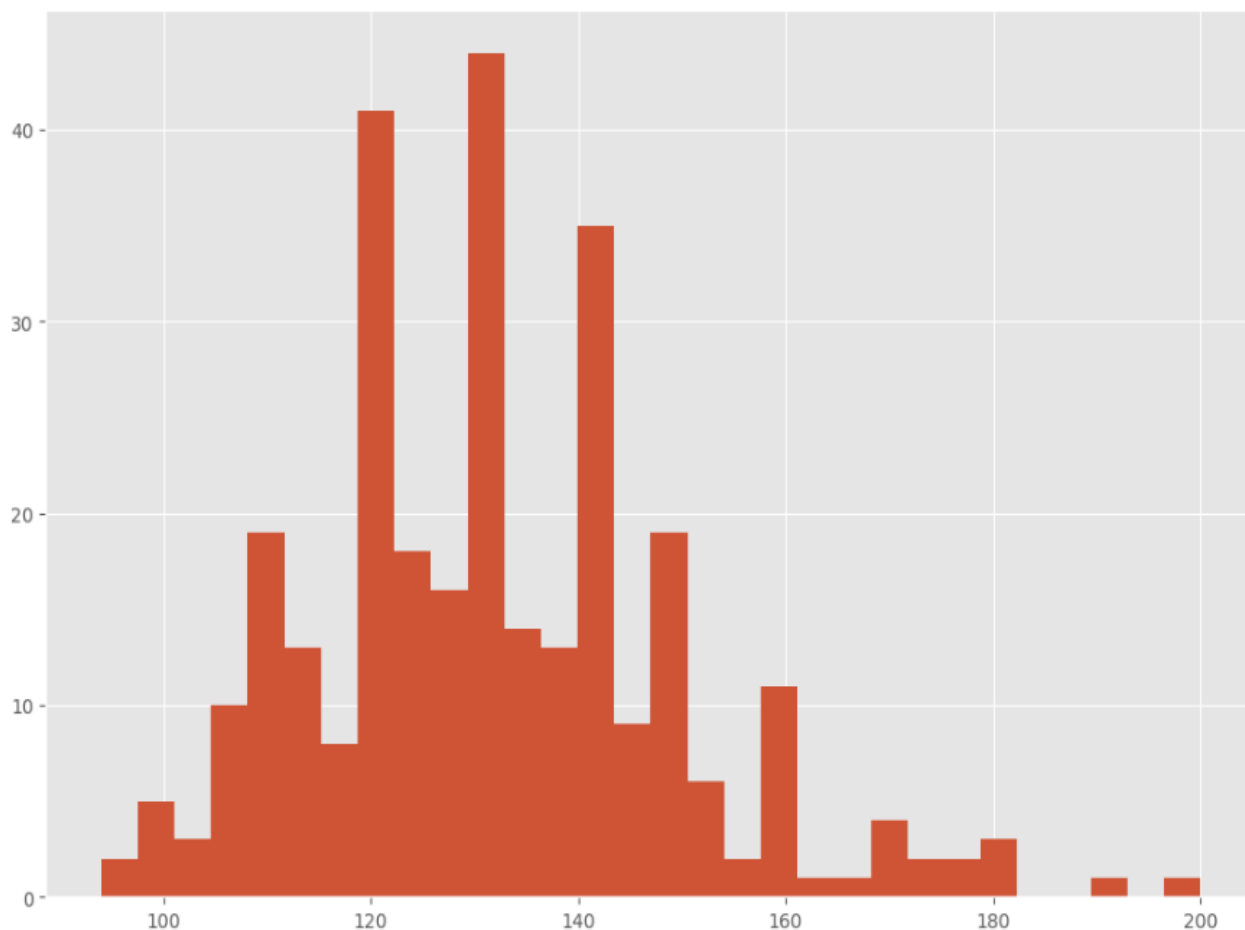


Рисунок 8 – Диаграмма признака с наличием выбросов

Как можно заметить, в этом варианте также хорошо можно увидеть наличие аномальных значений, выбивающихся из общей картины. Однако, видно, что визуально часть выбросов соединяется с общей диаграммой. Это зависит уже от конкретного случая и выбранных параметров построения диаграммы. Поэтому рекомендуется проводить оценку с использованием разных методов, чтобы можно было более уверенно говорить о свойствах выбранного признака.

В данном случае избавления признака от выбивающихся значений можно добиться путем отсеечения значений, превышающих некий порог. Ранее данный порог был определен с помощью графиков, теперь требуется только совершить очистку.

```
dfc_new = dfc.iloc[np.where(dfc['trestbps'].values <=
170)[0]]
dfc_new.shape
dfc_new['trestbps'].hist(bins=30)
plt.figure(figsize=(4,3))
plt.boxplot(dfc_new['trestbps'])
plt.show()
```

Данный код производит удаление значений свыше определенного порога, после чего строит диаграмму и отображает ящик с усами для демонстрации изменений указанных далее (Рисунок 9, Рисунок 10).

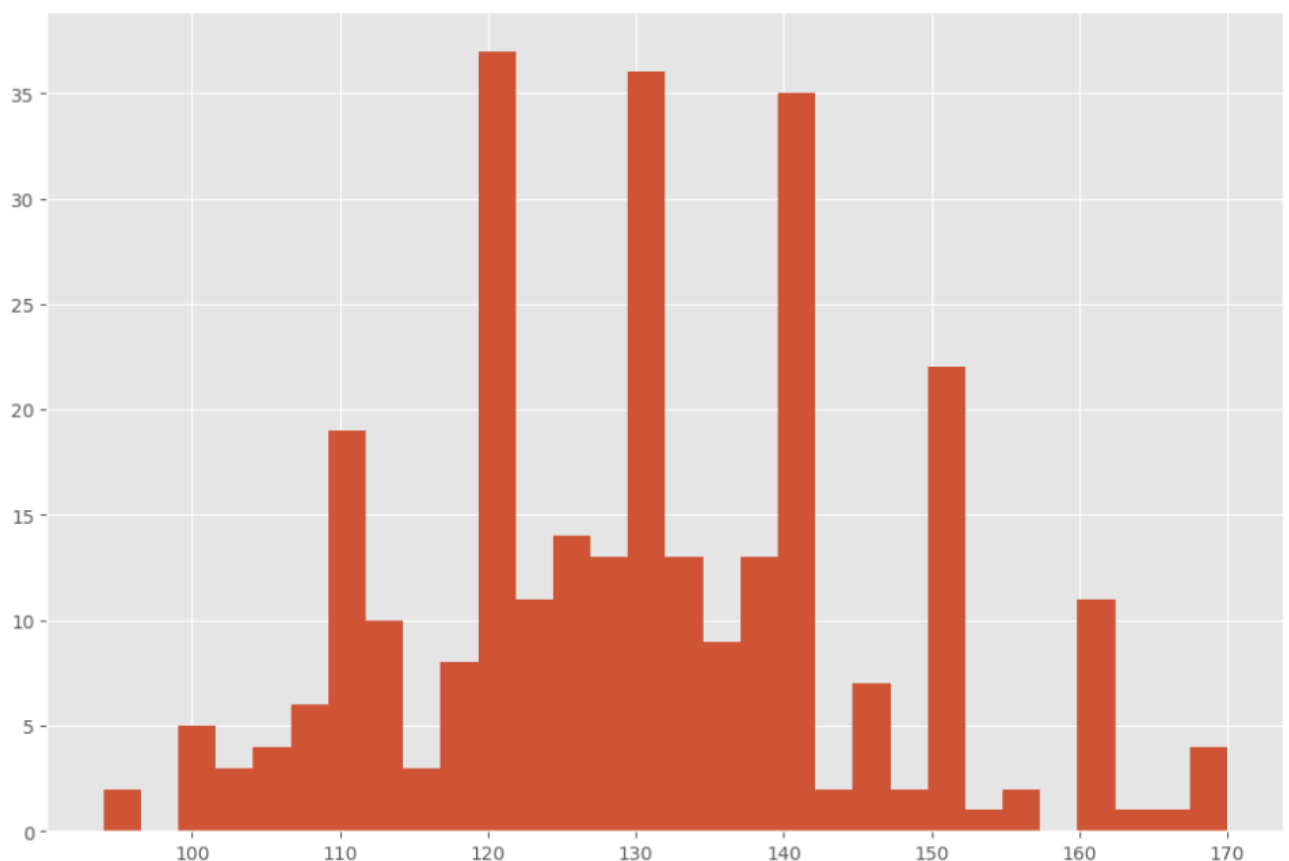


Рисунок 9 – Результат удаления выбросов из выборки

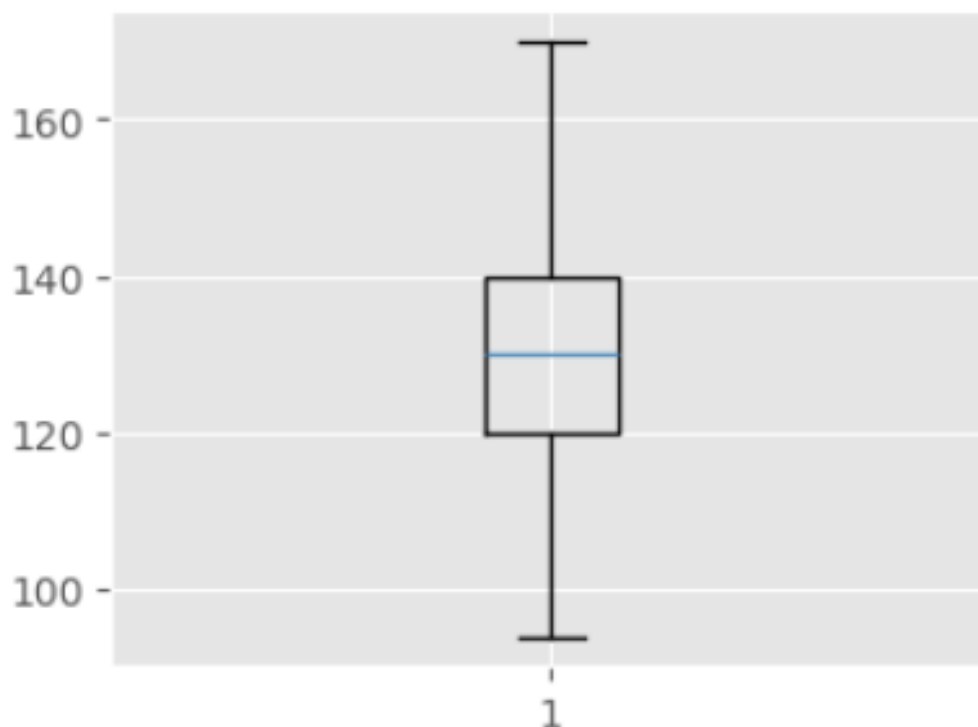


Рисунок 10 – Очищенный от выбросов признак (trestbps)

В ходе очистки из выборки были убраны значения, что слишком выбивались из общей картины, и теперь можно заметить, что график ящика с усами стал более равномерно распределенным, а точки, обозначающие аномальные значения, пропали. Диаграмма тоже претерпела визуальные изменения. Оставшиеся поля были обработаны по принципу, что был продемонстрирован ранее.

После того, как в данных не осталось шума, можно приступить к отбору ключевых признаков, для обучения модели. Первым наглядным методом оценки важности признаков может стать матрица корреляции, так как не все данные подчиняются нормальному закону, используем корреляцию Спирмана:

```
sns.heatmap(df.corr(method='spearman'), annot=True, cmap='RdYlGn', linewidths=0.2)
fig=plt.gcf()
fig.set_size_inches(12,12)
plt.show()
```

Код выше вызывает метод, формирующий матрицу корреляции признаков, и отображает ее в виде тепловой карты (Рисунок 11).

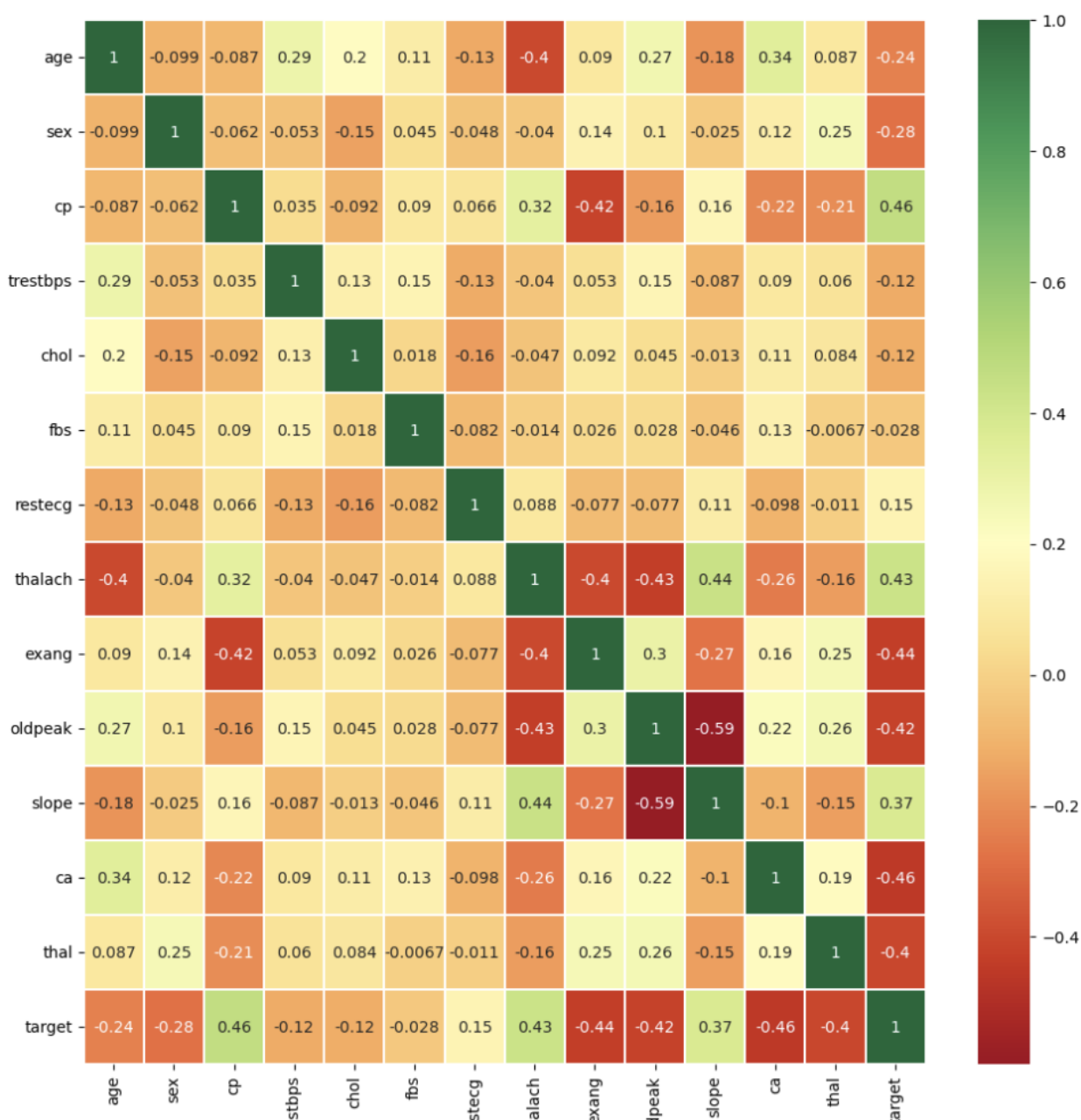


Рисунок 11 – Матрица корреляции признаков

Для того чтобы прочесть данную матрицу нужно сопоставить строчку и столбец, пересечение которых обозначено числовым значением. Таким образом, отображается зависимость каждого признака друг от друга. Сам коэффициент корреляции можно интерпретировать по-разному, в зависимости от того, какие значения принимать в качестве слабой, средней и сильной корреляции. Однако чем больше число отличается от нуля, тем сильнее корреляция.

Данный метод наглядный, но не очень эффективный, так как позволяет сравнить только взаимодействие всего двух признаков, что не позволит выявить более глубокие закономерности и скрытые правила.

Воспользуемся методом отбора признаков по принципу рекурсивного исключения, путем постепенного удаления признаков по принципу меньше важности:

```
from sklearn.linear_model import LogisticRegression
from sklearn.feature_selection import RFE

lr = LogisticRegression(
    class_weight = 'balanced',
    solver = 'lbfgs',
    random_state=42,
    n_jobs=-1,
    max_iter=50)

rfe = RFECV(lr, n_features_to_select=7)
rfe=rfe.fit(X, y)

selector_ind = rfe.get_support()
df1 = X.iloc[:, selector_ind]
df1.head(5)
```

Данный код вызывает метод рекурсивного отбора признаков по принципу исключения. Он будет удалять свойства до тех пор, пока не достигнет указанного числа признаков.

Сформированный набор данных с уменьшенным числом признаков поможет упростить задачу обучения модели. Во время ознакомления с темой отбора признаков была проанализирована статья [13], в которой подробно описываются важность, принципы и различия при выделении ключевых признаков в выборке. Правильное применение алгоритмов отбора положительно сказывается как на скорости обучения, так и на точности предсказания. После того как необходимое количество было достигнуто, их нужно отобразить. Описываемый код получает индексы сохраненных признаков и формирует новый дата фрейм (Рисунок 12).

	sex	cp	restecg	exang	oldpeak	slope	ca	thal
0	1	3	0	0	2.3	0	0	1
1	1	2	1	0	3.5	0	0	2
2	0	1	0	0	1.4	2	0	2
3	1	1	1	0	0.8	2	0	2
4	0	0	1	1	0.6	2	0	2

Рисунок 12 – Отобранные признаки

Однако у данного метода есть два заметных недостатка:

- данный подход требует ручного выбора числа признаков которые будут отобраны, что требует от программиста угадывать число признаков, вместо того чтобы выбрать оптимальное,

- классификатор. Для того чтобы оценить важность признаков алгоритм использует оценку путем обучения небольшой модели, однако это же и говорит о том, что отобранные признаки будут подходить больше всего именно для текущей выбранной модели.

Чтобы упразднить первую проблему, требуется переключить алгоритм в автоматический режим работы. В таком случае он начнет самостоятельно подбирать наиболее успешное число признаков.

Слегка доработаем код и добавим отображение числа признаков в ходе их отбора:

```
from sklearn.linear_model import LogisticRegression
from sklearn.feature_selection import RFE

lr = LogisticRegression(
    class_weight = 'balanced',
    solver = 'lbfgs',
    random_state=42,
    n_jobs=-1,
    max_iter=50)

rfe = RFE(lr, verbose=1)
rfe=rfe.fit(X, y)
```

```

selector_ind = rfe.get_support()
df1 = X.iloc[:, selector_ind]
df1.head(5)

```

Теперь отбор признаков полностью ложится в руки алгоритма. В итоге происходит отбор необходимого числа признаков с наибольшей значимостью, который и станет использоваться для обучения модели в будущем (Рисунок 13).

```

Fitting estimator with 13 features.
Fitting estimator with 12 features.
Fitting estimator with 11 features.
Fitting estimator with 10 features.
Fitting estimator with 9 features.
Fitting estimator with 8 features.
Fitting estimator with 7 features.

```

[29]:

	sex	cp	exang	slope	ca	thal
0	1	3	0	0	0	1
1	1	2	0	0	0	2
2	0	1	0	2	0	2
3	1	1	0	2	0	2
4	0	0	1	2	0	2

Рисунок 13 – Отобранные признаки

Можно заметить, что данный алгоритм не останавливается до тех пор, пока не подберет набор из необходимого числа признаков. В выводе работы алгоритма отображается журнал в ходе процесса обучения. На каждой итерации алгоритм тестирует разные наборы признаков для выбранного их количества. Если наборы с меньшим числом признаков показывают результат лучше, действие алгоритма продолжается.

Однако все еще остается вопрос, связанный с тем, что модель, используемая в ходе отбора признаков, является логистической регрессией.

Поэтому для того, чтобы оценить верность выбранного набора признаков, попробуем использовать и другие модели.

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.feature_selection import RFE

clf = RandomForestClassifier(n_estimators=50, max_depth=4,
random_state=42)

rfe = RFE(clf, verbose=1)
rfe=rfe.fit(X, y)

selector_ind = rfe.get_support()
df1 = X.iloc[:, selector_ind]
df1.head(5)
```

Данный код уже использует классификатор случайный лес. Результаты отбора можно увидеть на изображении далее (Рисунок 14).

```
Fitting estimator with 13 features.
Fitting estimator with 12 features.
Fitting estimator with 11 features.
Fitting estimator with 10 features.
Fitting estimator with 9 features.
Fitting estimator with 8 features.
Fitting estimator with 7 features.
```

	cp	thalach	exang	oldpeak	ca	thal
0	3	150	0	2.3	0	1
1	2	187	0	3.5	0	2
2	1	172	0	1.4	0	2
3	1	178	0	0.8	0	2
4	0	163	1	0.6	0	2

Рисунок 14 – Отбор признаков Случайный лес

Можно заметить, что набор признаков немного изменился, однако общая картина осталась схожей.

```
Для последней проверки используем метод опорных векторов:
from sklearn.feature_selection import RFE
from sklearn.svm import SVC
clf = SVC(kernel='linear', random_state=42,
class_weight='balanced')
rfe = RFE(clf, verbose=1)
rfe=rfe.fit(X, y)
selector_ind = rfe.get_support()
df1 = X.iloc[:, selector_ind]
df1.head(5)
```

Данный вариант уже вызывает метод опорных векторов для определения важных признаков, однако из-за ограничений алгоритма пришлось выбрать линейное ядро метода, у которого есть параметры коэффициентов, необходимые алгоритму отбора для определения важности свойств.

```
Fitting estimator with 13 features.
Fitting estimator with 12 features.
Fitting estimator with 11 features.
Fitting estimator with 10 features.
Fitting estimator with 9 features.
Fitting estimator with 8 features.
Fitting estimator with 7 features.
```

	sex	cp	exang	slope	ca	thal
0	1	3	0	0	0	1
1	1	2	0	0	0	2
2	0	1	0	2	0	2
3	1	1	0	2	0	2
4	0	0	1	2	0	2

Рисунок 15 – Отбор признаков по классификатору Метода опорных векторов

Можно заметить сходство с признаками, отобранными при использовании логистической регрессии в качестве оценочной модели. В таком случае стоит использовать именно этот набор признаков, ведь он подходит большему числу моделей.

После того, как самые важные признаки были отобраны, их стоит сохранить в виде отдельного датасета, который в дальнейшем будет использоваться для обучения используемых моделей:

```
df2 = pd.concat([df1, y], axis=1)
df2.to_csv('../HeartNorm.csv', index=False)
df2
```

Результат работы данного кода представлен на Рисунок 16:

	sex	cp	exang	slope	ca	thal	target
0	1	3	0	0	0	1	1
1	1	2	0	0	0	2	1
2	0	1	0	2	0	2	1
3	1	1	0	2	0	2	1
4	0	0	1	2	0	2	1
...	...	...	...	...	...	...	...
298	0	0	1	1	0	3	0
299	1	3	0	1	0	3	0
300	1	0	0	1	2	3	0
301	1	0	1	1	1	3	0
302	0	1	0	1	1	2	0
303 rows × 7 columns							

Рисунок 16 – Финальная выборка

После того как данные были обработаны все, что нам остается сделать – это нормализовать их и разделить на выборки для обучения и проверки работы модели.

Процесс нормализации приводит данные к общему формату, а разделение на выборку поставит точку в этапе обработки:

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler,
label_binarize

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
X_train, X_test, y_train, y_test =
train_test_split(X_scaled, y, test_size=.2, random_state=42)
X_train[:5], y_train[:5]
```

Используя стандартные методы библиотеки sklearn, производим нормализацию данных. Данный код отобразит то, как теперь выглядят нормализованные данные (Рисунок 17)

```
(array([[ 0.68100522,  0.03203122, -0.69663055, -0.89686172,  0.97635214,
        -0.71442887, -0.51292188],
       [ 0.68100522, -0.93851463,  1.43548113, -0.20670527,  0.97635214,
        -0.71442887,  1.12302895],
       [ 0.68100522,  1.00257707, -0.69663055,  2.20884228, -0.64911323,
        -0.71442887, -0.51292188],
       [-1.46841752,  0.03203122, -0.69663055,  0.31091206, -0.64911323,
        -0.71442887, -0.51292188],
       [ 0.68100522, -0.93851463,  1.43548113,  0.31091206,  0.97635214,
        1.24459328,  1.12302895]]),
132    1
202    0
196    0
75     1
176    0
Name: target, dtype: int64)
```

Рисунок 17 – Нормализованная выборка

Иногда размер выборки может заметно повлиять на скорость обучения модели. Так как не все модели быстро проходят обучение, при этом также требуется производить настройку параметров, стоит текущую полную выборку скопировать и преобразовать в уменьшенную. В данной выборке меньшего размера будет находиться лишь часть объектов оригинальной

выборки, отобранных путем семплирования. Данный подход позволяет сократить время обучения модели в процессе её настройки, что позволит быстрее достичь желаемых результатов работы алгоритма обучения.

```
sample_fraction = 0.3
X_train_reduced, _, y_train_reduced, _ = train_test_split(
    X_train, y_train,
    train_size=sample_fraction,
    random_state=42)
print(X_train_reduced.shape)
print(X_train_reduced[0:5,:])
```

В данном коде приведен пример того, как можно поделить уже имеющуюся обучающую выборку на выборку меньшего размера. В качестве соотношения был выбран коэффициент 0.3. Другими словами уменьшенная обучающая выборка будет представлять собой лишь 30% от оригинального набора, но при этом сохранит основные черты. Изначальная обучающая выборка содержала 80% от всего набора данных, а сокращенная содержит 30% от этого значения (Рисунок 18)

```
(72, 13)
[[-1.69473524 -1.46841752  1.00257707  0.36416545 -0.50756498
  0.89896224  0.1029038  -0.69663055 -0.89686172 -0.64911323
 -0.51292188]
 [-0.37126932 -1.46841752  1.00257707  0.47839125  1.19307747
 -1.00583187 -0.33440125 -0.69663055  0.39718162  0.97635214
 -0.51292188]
 [ 0.06988599  0.68100522 -0.93851463  0.02148802  2.06272417
  0.89896224 -0.7717063  1.43548113  0.13837295 -0.64911323
 1.12302895]
 [-1.80502406  0.68100522  1.00257707  0.36416545 -1.37721168
  0.89896224  1.0212444  -0.69663055 -0.89686172  0.97635214
 -0.51292188]
 [ 0.18017482 -1.46841752  0.03203122  0.47839125  0.92252071
 -1.00583187  0.1466343  -0.69663055  0.22464251 -0.64911323
 -0.51292188]]
```

Рисунок 18 – Уменьшенная выборка (72 строки)

Как можно заметить, набор нормализованных данных был заметно уменьшен. Однако благодаря методам семплирования он сохранил идентичность изначальной выборки

4.2 Реализация моделей, их обучение и оценка

Теперь, когда больше не осталось никаких препятствий, можно приступить к практическому этапу обучения модели. Данный этап подразумевает вызов методов машинного обучения упомянутых в предыдущих главах.

Логистическая регрессия – самый простой во многих планах алгоритм. Однако этого алгоритма вполне достаточно для выполнения задачи с сохранением высокого уровня точности.

```
logr = LogisticRegression()
logr.fit(X_train_reduced, y_train_reduced)

y_pred_log = logr.predict(X_test)

print(classification_report(y_test, y_pred_log))

cm_log = confusion_matrix(y_test, y_pred_log)
sns.heatmap(cm_log, annot=True, cmap="Reds")
plt.title("Confusion Matrix – Logistic Regression")
plt.show()
```

Данный код уже конкретно предназначен для оценки качества работы модели искусственного интеллекта. В качестве классификатора выбирается стандартная логистическая регрессия. Далее вызывается метод обучения модели на заранее подготовленной уменьшенной выборке и совершается предсказание значений тестовой выборки.

	precision	recall	f1-score	support
0	0.89	0.86	0.88	29
1	0.88	0.91	0.89	32
accuracy			0.89	61
macro avg	0.89	0.88	0.88	61
weighted avg	0.89	0.89	0.89	61

Рисунок 19 – Отчет работы модели



Рисунок 20 – Матрица ошибок

Для того чтобы понять, что отобразил отработавший код, потребуется немного разобраться в значении всех характеристик, указанных на Рисунок 19 и Рисунок 20. Для более понятного представления отобразим характеристики отчета в виде таблицы, в которой будут подробно описаны все пункты (Таблица 5).

Таблица 5 – Описание отчета работы модели

Характеристика	Описание
precision (точность предсказания)	Указывает процент верно предсказанных заболеваний для каждого из возможных исходов.
recall (полнота)	Доля верно предсказанных примеров среди всех объектов класса, принадлежащих к данному случаю. В отличие от точности оценивает процент выявленных случаев.

Продолжение таблицы 5

Характеристика	Описание
F1 (F мера)	Гармоническое среднее значение между точностью и полнотой, полезно при необходимости дать одну общую оценку модели.
support (поддержка)	Общее число объектов указанного класса в выборке.
accuracy (общая точность)	Доля всех верных предсказаний по всем классам в целом.
macro avg (общее среднее)	Среднее арифметическое основных метрик: тонкость, полнота и F мера.
weighted avg (взвешенное среднее)	Среднее значение метрик с использованием весов классов. Учитывается возможный дисбаланс выборки.

Теперь, когда стало понятно, как интерпретировать каждую из указанных метрик, стоит обратить внимание на матрицу ошибок модели или матрицу спутанности. Данная матрица читается довольно легко, если знать принцип.

Вертикальные ряды отвечают за то, какой выбор сделала модель в результате своей работы. Строки матрицы же отвечают за реальную принадлежность объектов к конкретным классам. Таким образом, обученный алгоритм верно оценил 25 случаев отсутствия заболевания у пациентов и 3 случая заболеваний раскрыть не смогла.

Благодаря матрице спутанности можно довольно быстро и наглядно определить насколько сильно модель ошибается при классификации заболевания у пациентов.

По результатам тестового прогона модели можно увидеть, что в целом модель научилась определять болезнь с довольно высокой точностью. Полнота в 91% показывает, что модель способна выявить довольно большую долю людей с сердечными заболеваниями, тем самым дав рекомендацию к срочному прохождению мед обследования.

Метод опорных векторов – метод определяющий гиперплоскость для разграничения объектов:

```
from sklearn.svm import SVC
clf = SVC(kernel='rbf', random_state=42,
class_weight='balanced')
clf.fit(X_train_reduced, y_train_reduced)

y_pred = clf.predict(X_test)

print(classification_report(y_test, y_pred))

cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, cmap="Reds")
plt.title("Confusion Matrix")
plt.show()
```

Данный код вызывает классификатор, использующий метод опорных векторов. Результаты работы отображены далее (Рисунок 21, Рисунок 22)

	precision	recall	f1-score	support
0	0.89	0.86	0.88	29
1	0.88	0.91	0.89	32
accuracy			0.89	61
macro avg	0.89	0.88	0.88	61
weighted avg	0.89	0.89	0.89	61

Рисунок 21 – Метрики работы модели

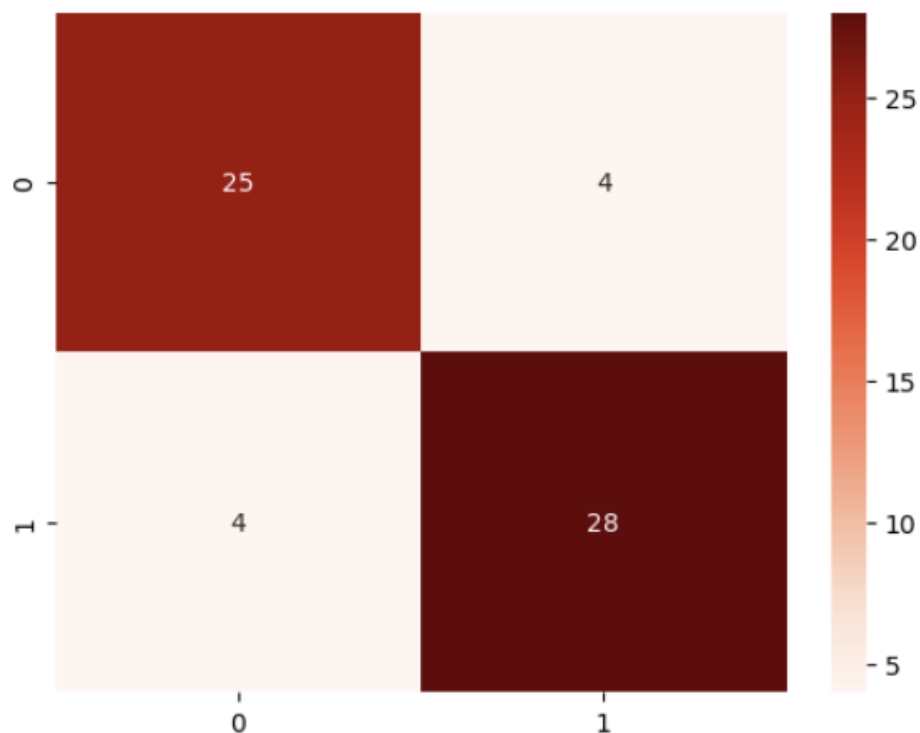


Рисунок 22 – Матрица ошибок

Можно заметить, что данный метод в целом хорошо себя проявляет. Он показывает хорошие результаты работы, не сильно отличающиеся от метода линейной регрессии. Высокие показатели полноты показывают хорошую способность модели выявлять болезнь у пациентов.

Следующим в списке моделей является случайный лес. Данная модель представляет собой ансамбль деревьев решений, спустя некоторое количество попыток удалось подобрать параметры:

```
from sklearn.ensemble import RandomForestClassifier
clf = RandomForestClassifier(n_estimators=50, max_depth=4,
random_state=42)

clf.fit(X_train_reduced, y_train_reduced)
y_pred = clf.predict(X_test)
print(classification_report(y_test, y_pred))

cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, cmap="Reds")
plt.title("Confusion Matrix")
plt.show()
```

Данный код задает параметры работы классификатора и отображает результат работы ансамбля деревьев решений.

	precision	recall	f1-score	support
0	0.89	0.86	0.88	29
1	0.88	0.91	0.89	32
accuracy			0.89	61
macro avg	0.89	0.88	0.88	61
weighted avg	0.89	0.89	0.89	61

Рисунок 23 – Оценочные метрики

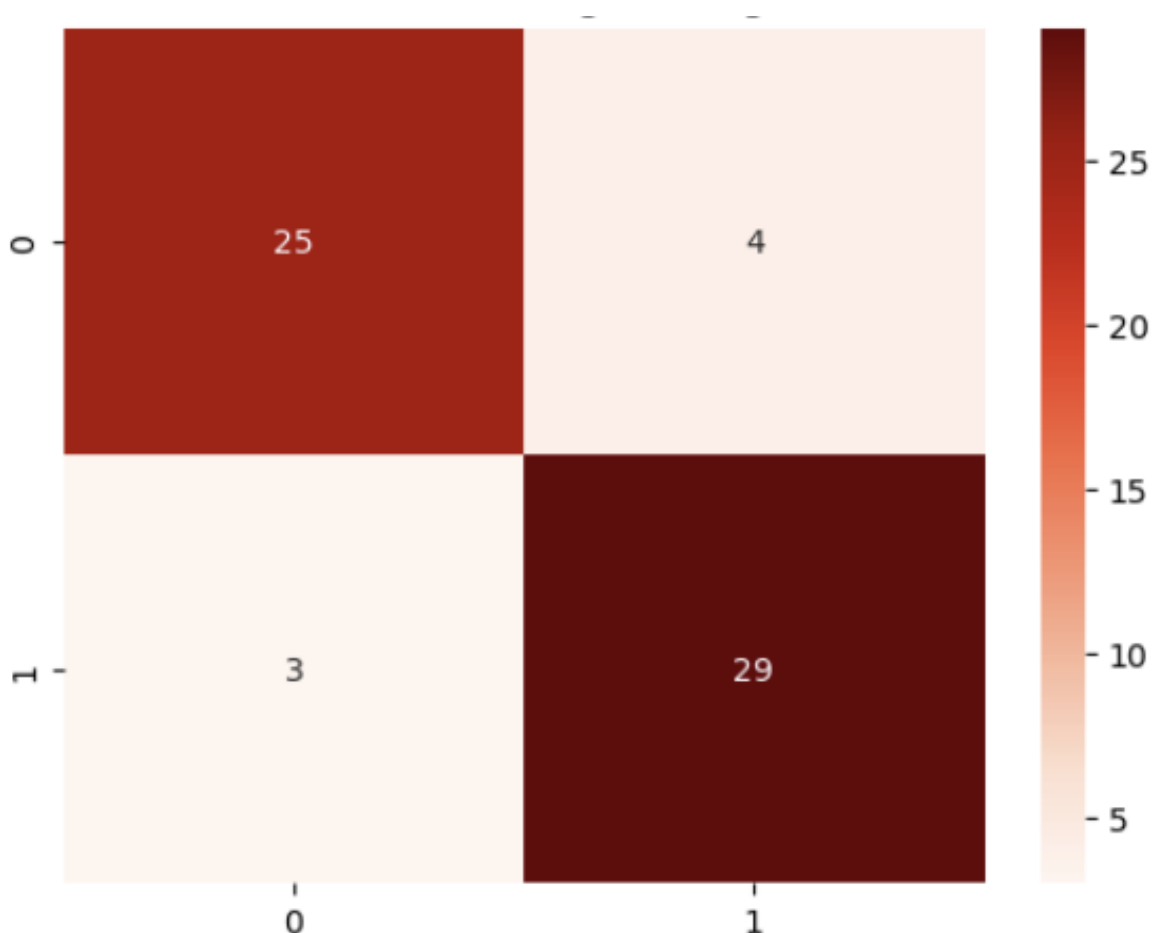


Рисунок 24 – Матрица ошибок

На Рисунок 23 и Рисунок 24 отображены результаты работы модели. Способность обученного классификатора выявлять болезнь осталась примерно той же, однако теперь удалось уменьшить число ложных срабатываний, что позволит реже беспокоить пациентов. Однако для достижения достаточной точности требуется использовать довольно большое количество деревьев, что потенциально может стать проблемой.

Самый непростой выбор модели для обучения – многослойный перцептрон. Ключевой особенностью данной модели является высокая точность, однако в противовес ей идет крайне сложная, непредсказуемая настройка. Выбор количества слоев в какой-то момент начинает происходить по принципу угадывания, а изменение размерности скрытого слоя каждый раз может привести к неожиданным результатам. Очень часто в попытке сделать, слегка лучше, происходило неожиданное ухудшение работы. Ключевой особенностью данного подхода можно назвать непредсказуемость и высокий потенциал.

```
from sklearn.neural_network import MLPClassifier
clf = MLPClassifier(
    hidden_layer_sizes=(10,2),
    max_iter=10000,
    random_state=42,
    early_stopping=False
)

clf.fit(X_train_reduced, y_train_reduced)

y_pred = clf.predict(X_test)

print(classification_report(y_test, y_pred))

cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, cmap="Reds")
plt.title("Confusion Matrix")
plt.show()
```

Данный код демонстрирует применение многослойного перцептрона. Как можно заметить, данный код отличается от других вариантов применения моделей. В большинстве случаев достаточно оставить гиперпараметры нетронутыми или лишь немного их изменить, однако в данном случае требуется более тонкая настройка. В проанализированном мною источнике [14] автор уделяет большое внимание важности настройки ключевых параметров модели.

Результаты работы модели представлены на Рисунок 25 и Рисунок 26.

	precision	recall	f1-score	support
0	0.89	0.86	0.88	29
1	0.88	0.91	0.89	32
accuracy			0.89	61
macro avg	0.89	0.88	0.88	61
weighted avg	0.89	0.89	0.89	61

Рисунок 25 – Метрики модели

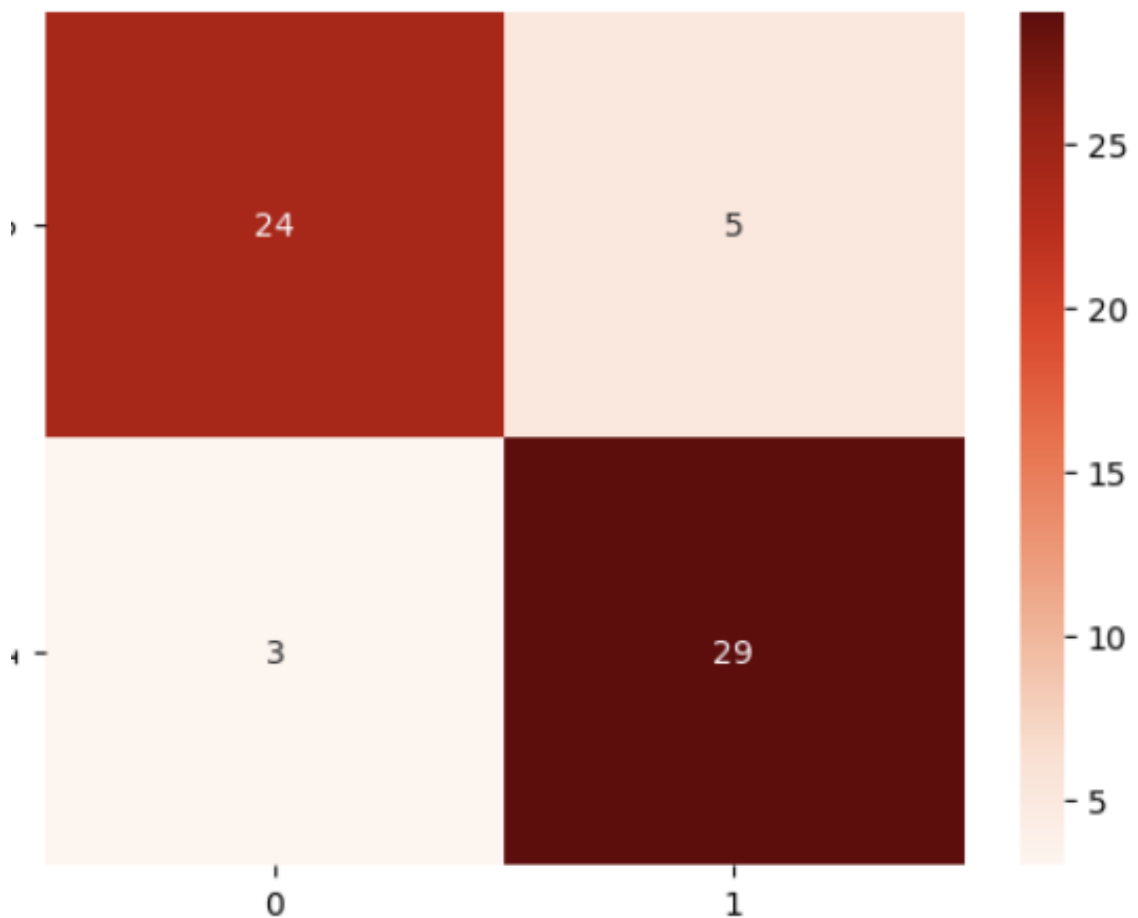


Рисунок 26 – Матрица ошибок

Достигаемая точность оказалась неплохая, но результат не соответствует приложенным усилиям, если сравнивать с работой иных моделей. Чаще всего это обосновывается недостаточной обработкой данных со стороны программиста или неправильной настройкой модели в связи с недостатком опыта.

Для того чтобы подвести итог по сравниваемым алгоритмам, сведем все выводы в одну таблицу результатов оценки моделей (Таблица 6).

Таблица 6 – Общая оценка моделей

Модель	F1-score	Полнота (выявление болезни)	Особенности применения
Случайный лес	0.89	0.91	После настройки числа деревьев и проведения нескольких прогонов для подбора глубины деревьев удалось добиться хорошего результата
Метод опорных векторов	0.87	0.88	В целом качество работы было немного слабее остальных вариантов, однако метод показал себя рабочим
Логистическая регрессия	0.89	0.91	Судя по всему, данный метод хорошо подошел для решения задачи, можно увидеть одни из самых высоких показателей оценки
Многослойный перцептрон	0.87	0.91	Очень сложный для настройки алгоритм, потребовалось больше десяти попыток, чтобы достичь результата хоть сколько-то приближенного к тому, что выдала модель логистической регрессии с первого раза. При этом разница во времени обучения была заметна невооруженным глазом.

4.3 Применение модели в конечном приложении

После оценки результаты работы каждой модели было решено использовать модель логистической регрессии, которая хорошо себя зарекомендовала как самая устойчивая модель, не требующая сложной настройки. Показатели точности этой модели оказались высоки и не уступают свои конкурентам.

Для взаимодействия с пользователем было сформировано простое графическое приложение, которое загружает обученные модели. Оно получает ввод пользователя, после чего выдает результат предсказания на основе обученной модели.

Для того чтобы использовать нашу модель в приложении, её нужно экспортировать в файл. Делается это именно потому, что в ином случае модель пришлось бы каждый раз заново обучать при каждом запуске приложения, что не только нелогично, но и не всегда возможно.

Для сохранения модели в файл воспользуемся модулем JobLib. Данный модуль позволяет сохранить объекты в виде набора данных в файле, после чего восстановить их заново уже в другой среде. Данный процесс называется сериализацией и десериализацией соответственно.

Сохраним нашу модель и препроцессор данных, производящий нормализацию. Нормализатор также необходимо сохранить, ведь масштабированные данные не хранят в себе информацию об изначальной выборке, из-за чего невозможно будет нормализовать пользовательский ввод и привести данные к формату, читаемому моделью.

```
import joblib
logr = LogisticRegression()
logr.fit(X_train, y_train)
y_pred_log = logr.predict(X_test)

joblib.dump(logr, 'logistic_model.pkl')
joblib.dump(scaler, 'scaler.pkl')
```

Данный код производит обучение модели уже на полной выборке. Так как ранее использовалось обучение модели на сокращенной выборке в угоду

скорости обучения, нужно обучить модель заново, но уже на полной выборке.

В результате работы данного кода были сформированы два файла, которые и представляют собой сохраненную модель и предпроцессор данных (Рисунок 27).

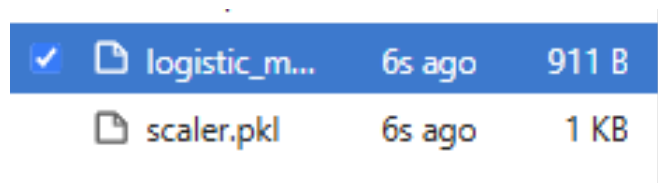


Рисунок 27 – Сохраненная модель

После того как модель сохранена, можно начать её применение. В ходе анализа и сравнения различных методов реализации графического интерфейса в прочитанных мною источниках [15, 16, 17] мною была выбрана библиотека tkinter, встроенная в Python по умолчанию. Данная библиотека является старым и надежным инструментом для создания графических приложений. Здесь весь интерфейс строится по простой системе окон и виджетов, их наполняющих.

Библиотека tkinter предлагает простой для использования набор инструментов. Визуальная составляющая библиотеки не выделяется красотой и яркостью, однако это компенсируется простотой разработки и настройки. Средствами данной библиотеки, можно построить простую и удобную систему окон пользовательского интерфейса приложения.

```
import numpy as np
from tkinter import ttk, messagebox
import joblib
import tkinter as tk

model = joblib.load('logistic_model.pkl')
scaler = joblib.load('scaler.pkl')
```

```
field_names = ['sex', 'cp', 'exang', 'slope', 'ca', 'thal']
```

В данном коде производится базовая инициализация данных приложения. Загружаются модели и предпроцессор. Определяются имена

полей признаков, которые мы будем использовать для предсказания. Это нужно для сопоставления их с полями из предпроцессора.

```
entries = []
for i, label in enumerate(labels):
    tk.Label(root, text=label).pack(pady=5)
    entry = tk.Entry(root)
    entry.pack(pady=5)
    entries.append(entry)

entry_sex, entry_cp, entry_exang, entry_slope, entry_ca,
entry_thal = entries

tk.Button(root, text="Получить Результат",
command=Get_Result, bg='lightblue').pack(pady=15)

result_label = tk.Label(root, text="", font=('Courier New',
12, 'bold'))
result_label.pack(pady=10)

root.mainloop()
```

Данный код формирует основные поля ввода в интерфейсе пользователя. Подписи к ним содержат пояснения, поэтому здесь указываться не будут, так как важности не несут. В конце, когда вид приложения сформирован, создается окно. После ввода данных формируется вход в модель, происходит масштабирование входных данных. Этот набор значений уже подается в модель и запускается процесс предсказания болезни. Данный код реализует процесс предсказания:

```
feature_names = ['sex', 'cp', 'exang', 'slope', 'ca',
'thal']
features_df = pd.DataFrame([user_input],
columns=feature_names)

print(features_df)
Fscaled = scaler.transform(features_df)
pred = model.predict(Fscaled)[0]

if pred == 1:
    messagebox.showwarning(
        "Результат обследования",
        "Результат обследования предупреждает о выявлении
заболевания!\n\n"
        "Срочно обратитесь к врачу, чтобы получить необходимое
лечение."
    )
```

```

else:
    messagebox.showinfo(
        "Результат обследования",
        "Результат обследования не выявил заболевания.\n"
        "Однако может существовать вероятность ошибки,\n"
        "рекомендуется пройти консультацию у врача."
    )

```

Работа приложения демонстрируется далее (Рисунок 28, Рисунок 29).

Ключевую роль в коде играет именно приведение введенных пользователем данных

Рисунок 28 – Результат работы приложения, болезнь не выявлена

Так как в ранее предоставленных результатах тестирования качества моделей было указано, что точность предсказания модели приблизительно 90%, система даже в случае выдачи отрицательного результата диагностики

предупреждает пользователя о потенциальной возможности ошибки. Данный подход важен и будет соблюдаться всегда.

Интеллектуальная диагностика состояния здоровья (Раздел сердечных заболеваний)

Пол (0/1 Женский/ Мужской):

Тип боли в груди указанный в обследовании (0-3):

Наличие стенокардии при упражнениях (0/1 : Нет/Да):

Результат обследования наклона ST сегмента (1-3):

Количество окрашенных сосудов по анализам флюорографии (0-3):

Результат прохождения теста с таллием(1-4):

Результат обследования

⚠ Результат обследования предупреждает о выявлении заболевания!

Срочно обратитесь к врачу, чтобы получить необходимое лечение.

Рисунок 29 – Выявлено заболевание

В ходе тестирования приложения удалось проверить и качество работы самой модели, в большинстве случаев результат предсказания совпадал с данными из датасета. Так как конкретные данные обследований мне недоступны, было решено попробовать ввести некоторые строки тестовой выборки датасета. В большинстве случаев, приложение показывало верные диагнозы и выводило верный результат предсказания с рекомендациями.

На данный момент визуальная составляющая приложения является тестовой версией для отладки работы модели. В дальнейшем планируется использовать более современные и приятные глазу элементы интерфейса. Среди неудобств можно выделить невозможность сменить тип модели. Также поля ввода задаются статично, и в случае изменения состава выборки придется изменять код программы. Так что в будущем планируется расширение функционала модулей приложения, с целью упростить взаимодействие с ним как пользователя, так и разработчика.

ЗАКЛЮЧЕНИЕ

В ходе работы была выполнена разработка интеллектуальной системы диагностики медицинского состояния человека. Данная система способна по данным, введенным пользователем, выявлять потенциальные заболевания.

Программа позволяет загружать данные в систему, обрабатывать их от шума, и применять их для обучения модели. Модуль обучения моделей реализует возможность сравнивать различные модели, путем демонстрации ключевой статистики выполнения предсказаний.

Все выбранные методы показали высокую эффективность и могут быть применены для достижения более качественного результата. Каждый метод обладает своими достоинствами и недостатками, которые можно обойти при достаточном проявлении умений со стороны разработчика системы.

В перспективах развития системы возможно внедрение более качественных методов статистической обработки данных, и более гибких систем настройки моделей. В будущем планируется автоматизация процесса обучения, не требующая от разработчика ручной настройки параметров. Данное приложение не является финальной версией, и в будущем оно будет развиваться для достижения еще более лучших результатов.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Поряева Е.П., Евстафьева В.А., Искусственный интеллект в медицине // ВЕСТНИК НАУКИ И ОБРАЗОВАНИЯ №6(60). Часть 2 — С. 15—18.
2. Уролбой Абдуманнон угли Хусанов, Мейрбек Бахитбай угли Кудратиллаев, Бобирбек Норпулат угли Сиддилов, Саияра Балтабаевна Довлетова Искусственный интеллект в медицине // «Science and Education» Научный журнал . — Impact Factor 3.848. май 2023. — Глава 4. — С. 772—782.
3. Шахназаров Б. А. Применение технологий искусственного интеллекта при создании вакцин и иных объектов интеллектуальной собственности (правовые аспекты) // Актуальные проблемы российского права. — 2020. — Т. 15. — № 7. — С. 76—90.
4. Фершт В.М., Латкин А.П., Иванова В.Н. Современные подходы к использованию искусственного интеллекта в медицине // Территория новых возможностей. Вестник Владивостокского государственного университета экономики и сервиса. 2020. Т. 12, № 1. С. 121–130.
5. Манкибаев Б.С. Основные направления внедрения искусственного интеллекта в медицине // Журнал «Наука, образование и культура» — № 3 (37). — 2019. С. 69—71
6. Сааков Д. В. Улучшение производительности алгоритмов машинного обучения с несбалансированными данными // Экономика строительства. – 2023. – №. 4. – С. 73-77..
7. Клюковкин Г. К. Нейронные сети против классических ML-моделей: в каких случаях стоит усложнять архитектуру //Актуальные исследования. – 2024. – №. 37 (219).
8. Терещенко А. Метрики оценки качества вероятностей в бинарной классификации: опыт из ФинТеха // Хабр. 2024. 8 декабря. URL: <https://habr.com/ru/articles/864794/> (дата обращения: 04.05.2026).

9. Lapp, D. Heart Disease Dataset / David Lapp // Kaggle: AI & ML community: [сайт]. – 2019. – URL: <https://www.kaggle.com/datasets/johnsmith88/heart-disease-dataset> (дата обращения: 16.05.2026).

10. Чурилов А. А. Методы машинного обучения для прогнозирования позиций сайтов в поисковой выдаче //ЕВРАЗИЙСКИЙ НАУЧНЫЙ. – 2026. – С. 29.

11. Пальмов С. В., Поскиваткина А. А. Сравнение библиотек tkInter, PyQt, wxPython //Евразийское Научное Объединение. – 2020. – №. 11-2. – С. 122-126.

12. Сальникова К. В. Анализа массива данных с помощью инструмента визуализации «Ящик с усами» //Univsum: экономика и юриспруденция. – 2021. – №. 6 (81). – С. 11-17.

13. Синотова С. Л. и др. Сравнение эффективности различных методов отбора признаков для решения задачи бинарной классификации предсказания наступления беременности при проведении экстракорпорального оплодотворения //Моделирование, оптимизация и информационные технологии. – 2020. – Т. 8. – №. 3. – С. 2-3.

14. Зубков Г. А. Алгоритм оптимизации гиперпараметров многослойного персептрона для решения задачи прогнозирования технического состояния космического аппарата //Вестник Тверского государственного технического университета. Серия: Технические науки. – 2021. – №. 1. – С. 64-70.

15. Абишева А. А. ПРИЕМЕНИЕ TKINTER ГРАФИЧЕСКОГО ИНТЕРФЕЙСА КОМПОЗИЦИКА PYGUBU //Глобус: технические науки. – 2019. – №. 3 (27). – С. 5-10.

16. Прокопьева Н. В., Губко М. В., Андреева И. А. К вопросу о выборе библиотеки Python по работе с графическими интерфейсами для преподавания в рамках основной или дополнительной школьной программы //Научно-педагогический журнал " Учитель Алтая". – 2021. – №. 3. – С. 24-29.

17. Наркевич А. Н., Виноградов К. А. Выбор метода для статистического анализа медицинских данных и способа графического представления результатов //Социальные аспекты здоровья населения. – 2019. – №. 4 (68). – С. 9.

Приложение А

(обязательное)

Разработка интеллектуальной системы диагностики медицинского
состояния человека
Листинг программы

Модуль обработки данных:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as sts
%matplotlib inline
import statistics as st
import math
df=pd.read_csv('../heart.csv', delimiter=',')
cp = df['cp']
age = df['age']
fig, ax = plt.subplots()
ax.scatter(cp, age, c='red', alpha=0.3, edgecolors='none')

ax.grid(True)
plt.ylabel('age')
plt.xlabel('chest pain type')
plt.show()
trestbps = df['trestbps']
age = df['age']
fig, ax = plt.subplots()
ax.scatter(age, trestbps, c='red', alpha=0.37,
edgecolors='none')

ax.grid(True)
plt.xlabel('age')
plt.ylabel('trestbps')
plt.show()
x = [i for i in range(0, len(df.age))]
y = df.age

plt.xlabel('номер пациента')
plt.ylabel('возраст')
plt.title('Точечный график зависимости двух выборок x и y')
plt.grid(True)
plt.scatter(x, y, marker='.')
plt.show()
sns.heatmap(df.corr(method='spearman'), annot=True, cmap='RdYl
Gn', linewidths=0.2)
fig=plt.gcf()
fig.set_size_inches(12,12)
plt.show()

import matplotlib.pyplot as plt
import matplotlib.mlab as mlab
import matplotlib
plt.style.use('ggplot')
from matplotlib.pyplot import figure

%matplotlib inline
matplotlib.rcParams['figure.figsize'] = (12,8)
```

```

pd.options.mode.chained_assignment = None
plt.figure(figsize=(8,6))
plt.scatter(range(df.shape[0]), np.sort(df.target.values))
plt.xlabel('Количество', fontsize=12)
plt.ylabel('Заболевание', fontsize=12)
plt.show()
dfc['thal'] = dfc['thal'].replace(0, np.nan)
dfc['ca'] = dfc['ca'].replace(4, np.nan)

max_cols = 3
total_columns = dfc.shape[1]
rows = (total_columns + max_cols - 1) // max_cols

plt.figure(figsize=(15, rows * 6))

for idx, col in enumerate(dfc.columns):
    plt.subplot(rows, max_cols, idx + 1)
    plt.boxplot(dfc[col].dropna())
    plt.title(f'Boxplot: {col}')
    plt.ylabel('Значения')
    plt.xlabel(col)
    plt.xticks([])

plt.tight_layout(pad=3.0)
plt.show()

dfc_new = dfc.iloc[np.where(dfc['trestbps'].values <=
170)[0]]
dfc_new.shape

dfc_new['trestbps'].hist(bins=30)
dfc['trestbps'].hist(bins=30)
plt.figure(figsize=(4,3))
plt.boxplot(dfc_new['trestbps'])
plt.show()
plt.figure(figsize=(8,6))
plt.scatter(range(df.shape[0]), np.sort(df.chol.values))
plt.xlabel('Нормер', fontsize=12)
plt.ylabel('Холестерин', fontsize=12)
plt.show()
dfc_new = dfc.iloc[np.where(dfc['chol'].values <= 370)[0]]
dfc_new.shape

plt.figure(figsize=(12,4))
plt.hist(dfc_new['chol'], bins=30)
plt.xticks(np.arange(0, dfc_new['chol'].max() + 50, 25))
plt.show()

plt.figure(figsize=(3,6))
plt.boxplot(dfc_new['chol'])

```

```

plt.show()
dfc.iloc[np.where(dfc['chol'].values > 370)[0]]
count = (dfc['chol'].values > 370).sum() / dfc.shape[0]
k=round(count*100, 2)
print('{} >> {}'.format('chol > 370', k))
dfc_new = dfc.iloc[np.where(dfc['chol'].values <= 370)[0]]
dfc_new.shape

plt.figure(figsize=(12,4))
plt.hist(dfc_new['chol'], bins=30)
#plt.xticks(np.arange(0, dfc_new['chol'].max() + 50, 25))
plt.show()

plt.figure(figsize=(3,6))
plt.boxplot(dfc_new['chol'])
plt.show()
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as sts
%matplotlib inline
import statistics as st
import math
import seaborn as sns
from IPython.display import display as disp
df=pd.read_csv('../heart.csv', delimiter=',')
X=df.iloc[:,0:13]
y=df.iloc[:,13]
from sklearn.linear_model import LogisticRegression
from sklearn.feature_selection import RFE

lr = LogisticRegression(
    class_weight = 'balanced',
    solver = 'lbfgs',
    random_state=42,
    n_jobs=-1,
    max_iter=50)

rfe = RFE(lr, n_features_to_select=7)
rfe=rfe.fit(X, y)

selector_ind = rfe.get_support()
df1 = X.iloc[:, selector_ind]
df1.head(5)
from sklearn.linear_model import LogisticRegression
from sklearn.feature_selection import RFE

lr = LogisticRegression(
    class_weight = 'balanced',
    solver = 'lbfgs',
    random_state=42,
    n_jobs=-1,

```

```

max_iter=50)

rfe = RFE(lr, verbose=1)
rfe=rfe.fit(X, y)

selector_ind = rfe.get_support()
df1 = X.iloc[:, selector_ind]
df1.head(5)
from sklearn.feature_selection import RFE
from sklearn.svm import SVC

clf = SVC(kernel='linear', random_state=42,
class_weight='balanced')

rfe = RFE(clf, verbose=1)
rfe=rfe.fit(X, y)

selector_ind = rfe.get_support()
df1 = X.iloc[:, selector_ind]
df1.head(5)
df2 = pd.concat([df1, y], axis=1)
df2.to_csv('../HeartNorm.csv', index=False)
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler,
label_binarize

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
X_train, X_test, y_train, y_test =
train_test_split(X_scaled, y, test_size=.2, random_state=42)
X_train.shape
sample_fraction = 0.3
X_train_reduced, _, y_train_reduced, _ = train_test_split(
    X_train, y_train,
    train_size=sample_fraction,
    random_state=42
)
print(X_train_reduced.shape)
print(X_train_reduced[0:5,:])
X_train[:5], y_train[:5]

```

Модуль обучения моделей:

```

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as sts
%matplotlib inline
import statistics as st
import math

```

```

import seaborn as sns
from IPython.display import display as disp
df=pd.read_csv('../heartNorm.csv', delimiter=',')
df.shape
X = df.iloc[:,0:6]
y = df.iloc[:, 6]
X
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler,
label_binarize

#Полная выборка
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
X_train, X_test, y_train, y_test =
train_test_split(X_scaled, y, test_size=.2, random_state=42)
X_train.shape
#Сокращенная выборка
sample_fraction = 0.3
X_train_reduced, _, y_train_reduced, _ = train_test_split(
    X_train, y_train,
    train_size=sample_fraction,
    random_state=42
)
X_train_reduced.shape

logr = LogisticRegression()
logr.fit(X_train_reduced, y_train_reduced)

y_pred_log = logr.predict(X_test)

print(classification_report(y_test, y_pred_log))

cm_log = confusion_matrix(y_test, y_pred_log)
sns.heatmap(cm_log, annot=True, cmap="Reds")
plt.title("Confusion Matrix – Logistic Regression")
plt.show()

import joblib

logr = LogisticRegression()
logr.fit(X_train, y_train)

y_pred_log = logr.predict(X_test)

joblib.dump(logr, 'logistic_model.pkl')
joblib.dump(scaler, 'scaler.pkl')

from sklearn.svm import SVC
clf = SVC(kernel='rbf', random_state=42,
class_weight='balanced')
clf.fit(X_train_reduced, y_train_reduced)

```

```

y_pred = clf.predict(X_test)

print(classification_report(y_test, y_pred))

cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, cmap="Reds")
plt.title("Confusion Matrix")
plt.show()

from sklearn.ensemble import RandomForestClassifier
clf = RandomForestClassifier(n_estimators=50, max_depth=4,
random_state=42)
clf.fit(X_train_reduced, y_train_reduced)

y_pred = clf.predict(X_test)

print(classification_report(y_test, y_pred))

cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, cmap="Reds")
plt.title("Confusion Matrix")
plt.show()

from sklearn.neural_network import MLPClassifier
clf = MLPClassifier(
    hidden_layer_sizes=(10,2),
    max_iter=10000,
    random_state=42,
    early_stopping=False
)
clf.fit(X_train_reduced, y_train_reduced)

y_pred = clf.predict(X_test)

print(classification_report(y_test, y_pred))

cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, cmap="Reds")
plt.title("Confusion Matrix")
plt.show()

```

Пользовательский модуль:

```

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as sts
%matplotlib inline
import statistics as st
import math
import seaborn as sns

```

```

from IPython.display import display as disp
import numpy as np
from tkinter import ttk, messagebox
import joblib
import tkinter as tk

model = joblib.load('logistic_model.pkl')
scaler = joblib.load('scaler.pkl')

field_names = ['sex', 'cp', 'exang', 'slope', 'ca', 'thal']

def Get_Result():
    try:
        user_input = [float(entry_sex.get()),
float(entry_cp.get()),float(entry_exang.get()),float(entry_slope
.get()),float(entry_ca.get()),float(entry_thal.get()),
]

        feature_names = ['sex', 'cp', 'exang', 'slope',
'ca', 'thal']
        features_df = pd.DataFrame([user_input],
columns=feature_names)

        print(features_df)

        Fscaled = scaler.transform(features_df)
        pred = model.predict(Fscaled)[0]

        if pred == 1:
            messagebox.showwarning(
                "Результат обследования",
                "Результат обследования предупреждает о
выявлении заболевания!\n\n"
                "Срочно обратитесь к врачу, чтобы получить
необходимое лечение."
            )
        else:
            messagebox.showinfo(
                "Результат обследования",
                "Результат обследования не выявил
заболевания.\n\n"
                "Однако может существовать вероятность
ошибки,\n"
                "рекомендуется пройти консультацию у врача."
            )

    except ValueError:
        messagebox.showerror("Ошибка не удалось прочитать
введенные данные", "Введите корректные числовые значения")
    except Exception as e:
        messagebox.showerror("Ошибка", f"Ошибка: {str(e)}")

```

```

root = tk.Tk()
root.title("Интеллектуальная диагностика состояния здоровья
(Раздел сердечных заболеваний)")
root.geometry("700x650")
labels = ['Пол (0/1 Женский/ Мужской):', 'Тип боли в груди
указанный в обследовании (0-3):', 'Наличие стенокардии при
упражнениях (0/1 : Нет/Да):',
          'Результат обследования наклона ST сегмента (1-
3):', 'Количество окрашенных сосудов по анализам флюорографии (0-
3):',
          'Результат прохождения теста с таллием(1-4):']

entries = []
for i, label in enumerate(labels):
    tk.Label(root, text=label).pack(pady=5)
    entry = tk.Entry(root)
    entry.pack(pady=5)
    entries.append(entry)

entry_sex, entry_cp, entry_exang, entry_slope, entry_ca,
entry_thal = entries

tk.Button(root, text="Получить Результат",
command=Get_Result, bg='lightblue').pack(pady=15)

result_label = tk.Label(root, text="", font=('Courier New',
12, 'bold'))
result_label.pack(pady=10)

root.mainloop()

```

Приложение Б

(обязательное)

Разработка интеллектуальной системы диагностики медицинского
состояния человека
Презентация

РАЗРАБОТКА ИНТЕЛЛЕКТУАЛЬНОЙ СИСТЕМЫ ДИАГНОСТИКИ МЕДИЦИНСКОГО СОСТОЯНИЯ ЧЕЛОВЕКА

Работу выполнил:
магистрант группы 24ВВИМ2

Савин Е.Д.

Научный руководитель:

к.т.н., Тимонин А.А

Пенза, 2026

Цели работы

Разработать систему, способную проводить диагностику состояния здоровья человека с применением технологий машинного обучения.

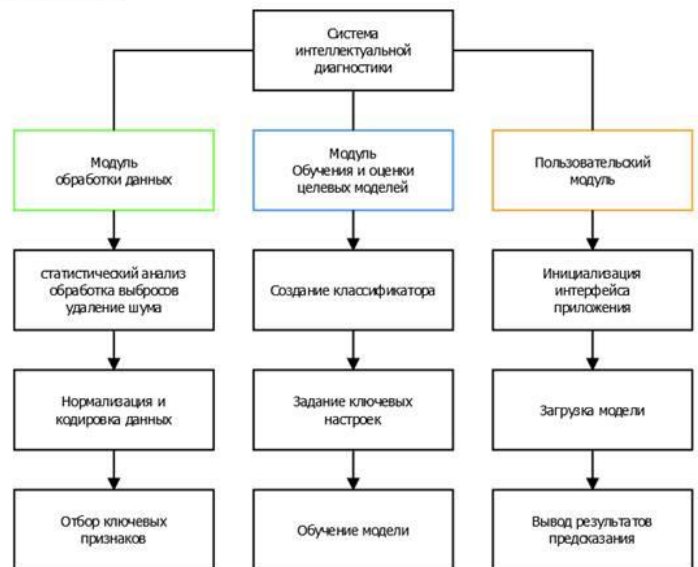
Объектом работы является система реализующая следующие возможности:

- загрузка и обработка медицинских данных
- обучение моделей искусственного интеллекта
- выдача результатов предсказания наличия заболеваний и рекомендаций пользователю

Структура разрабатываемой системы

Система состоит из трех основных модулей уникального назначения:

- Модуль предварительной обработки данных
- Модуль обучения моделей и оценки качества их работы
- Модуль взаимодействия с пользователем



Модуль предварительной обработки

Назначение модуля:

- Загрузка данных в корректном формате
- Очистка данных от пропусков и выбросов
- Выявление и отбор наиболее важных признаков
- Нормализация данных
- Формирование обучающей и тестовой выборок

Модуль обучения моделей

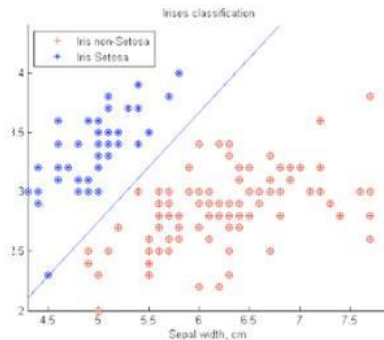
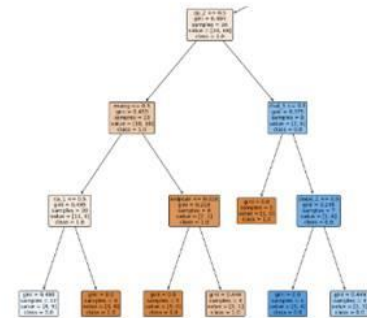
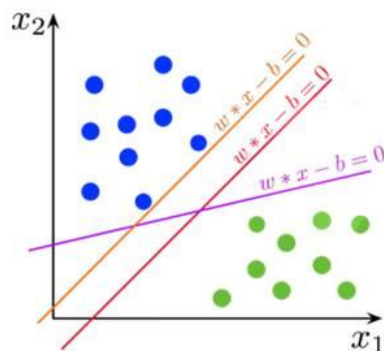
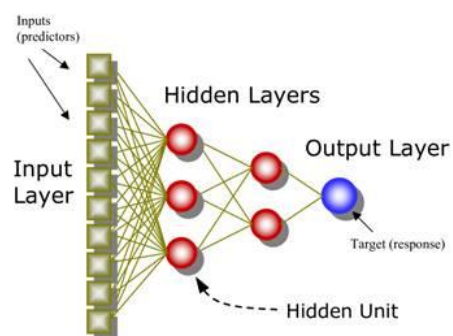
Функции модуля:

- Настройка ключевых параметров выбранных моделей
- Обучение моделей на отобранных данных
- Вывод результатов обучения моделей
- Оценка качества работы

Выбранные алгоритмы машинного обучения

- Логистическая регрессия
- Случайны лес
- Метод опорных векторов
- Многослойный персептрон

Особенности алгоритмов



Применяемые инструменты разработки

- Python
- Jupyter Lab
- Scikit Learn



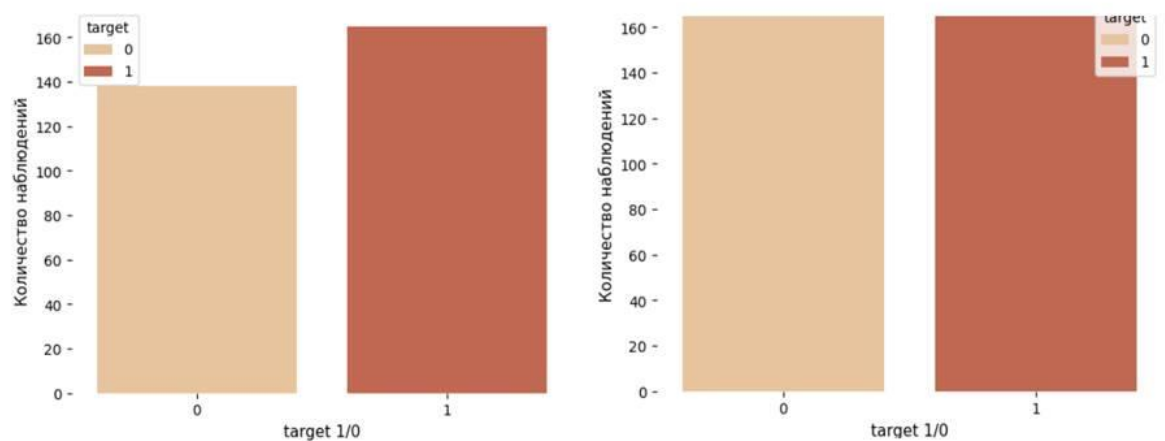
Процесс обработки данных

```
1:
```

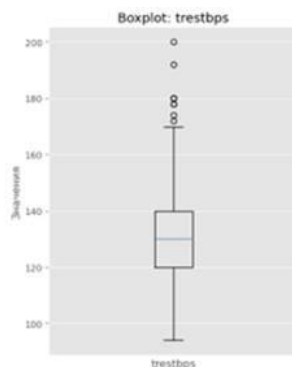
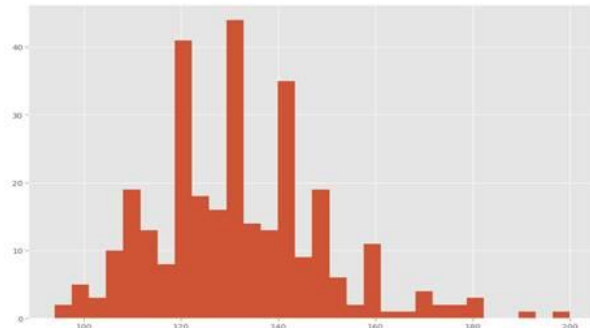
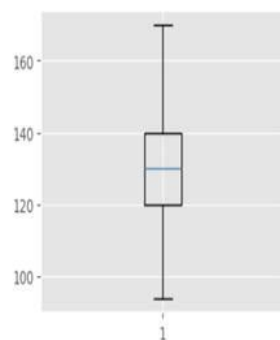
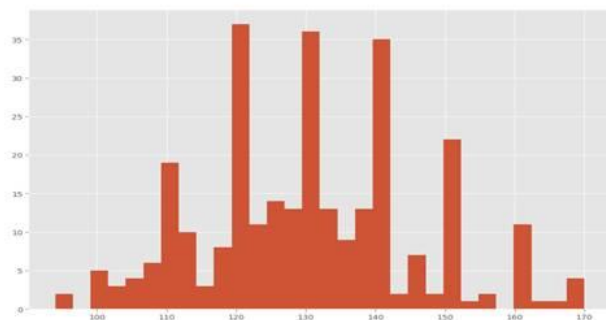
	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3	0
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3	0
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3	0
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3	0
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2	0

303 rows × 14 columns

Процесс обработки данных



Процесс обработки данных



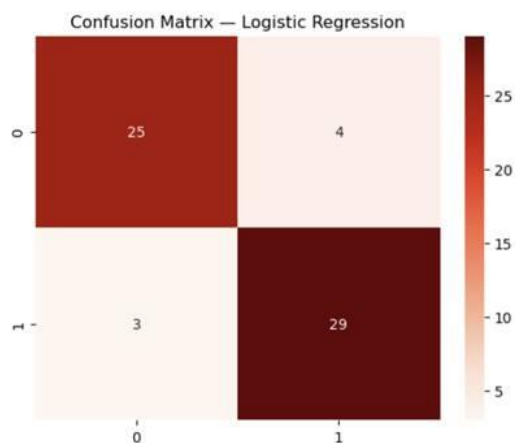
Отбор признаков

Fitting estimator with 13 features.
 Fitting estimator with 12 features.
 Fitting estimator with 11 features.
 Fitting estimator with 10 features.
 Fitting estimator with 9 features.
 Fitting estimator with 8 features.
 Fitting estimator with 7 features.

	cp	thalach	exang	oldpeak	ca	thal
0	3	150	0	2.3	0	1
1	2	187	0	3.5	0	2
2	1	172	0	1.4	0	2
3	1	178	0	0.8	0	2
4	0	163	1	0.6	0	2

Обучение моделей

	precision	recall	f1-score	support
0	0.89	0.86	0.88	29
1	0.88	0.91	0.89	32
accuracy			0.89	61
macro avg	0.89	0.88	0.88	61
weighted avg	0.89	0.89	0.89	61



Результаты обучения

Модель	F1-score	Полнота (выявление болезни)
Случайный лес	0.89	0.91
Метод опорных векторов	0.87	0.88
Логистическая регрессия	0.89	0.91
Многослойный персептрон	0.87	0.91

Пример взаимодействия с интерфейсом приложения

Интеллектуальная диагностика состояния здоровья (Раздел сердечных заболеваний)

Пол (0/1 Женский/ Мужской):
1

Тип боли в груди указанный в обследовании (0-3):
0

Наличие стенокардии при упражнениях (0/1 : Нет/Да):
1

Результат обследования наклона ST сегмента (1-3):
1

Количество окрашенных сосудов по анализам флюорографии (0-3):
1

Результат прохождения теста с таллием(1-4):
2

[Получить Результат](#)

Результат обследования

Результат обследования не выявил заболевания.
Однако может существовать вероятность ошибки, рекомендуется пройти консультацию у врача.

[ОК](#)

Заключение

Разработано приложение способное :

- Обработать медицинские данные
- Обучать модели и сравнивать результаты их работы
- Выявлять наличие заболеваний на основе пользовательского ввода

Версия системы не является финальной и подлежит дальнейшему развитию и доработке.