МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет	*Кафедра*
вычислительной техники	«Вычислительная техника»

Направление подготовки: 09.04.01 «Информатика и вычислительная техника»

Магистерская программа: «Вычислительные машины, комплексы, системы и сети»

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

на тему

ПРОГРАММА ПРОГНОЗИРОВАНИЯ ПОСЛЕОПЕРАЦИОННЫХ ОСЛОЖНЕНИЙ МОЧЕКАМЕННОЙ БОЛЕЗНИ

Магистрант _____ Кузнецов Роман Николаевич
(подпись, дата) (ФИО полностью)

Руководитель 07.06.23 Коннов Н.Н.
(подпись, дата) (фамилия, инициалы)

Нормоконтролёр 7.06.23 Бычков А.С.
(подпись, дата) (фамилия, инициалы)

Рецензент зав. кафедрой программирования 07.06.23 Мартынюк А.И.
(должность, место работы) (подпись, дата) (фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 02.06.2023 № 14)

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

Работа защищена с отметкой отлично (протокол заседания ГЭК от 9.06.23 № 6)

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

Пенза 2023

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

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
_____М.А. Митрохин
«29» ноября 2021г.

ЗАДАНИЕ
НА МАГИСТЕРСКУЮ ДИССЕРТАЦИЮ

1. Магистрант Кузнецов Роман Николаевич гр. 21ВВВм1 факультета ВТ

2.Тема работы «Программа прогнозирования послеоперационных осложнений мочекаменной болезни»

Тема утверждена приказом ПГУ № 924/о от "29" ноября 2021г.

3. Руководитель магистерской диссертации _ к.т.н., доцент Коннов Н. Н.__

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

Назначение разработки: осуществление краткосрочного прогнозирования наличия послеоперационных осложнений при мочекаменной болезни, отталкиваясь от наличия или отсутствия у диагностируемого человека определённых факторов, влияющих на риск наличия или отсутствия осложнений после операции.

Разработать экспертную систему для прогнозирования послеоперационных осложнений. В качестве пользовательского интерфейса использовать чат-бот Телеграмм.

5. Перечень подлежащих разработке вопросов

1. Анализ состояния исследований и разработок диагностических систем
2. Исследование методов прогнозирования осложнений при мочекаменной болезни
3. Классификация и прогноз послеоперационных осложнений при мочекаменной болезни
4. Проектирование программного продукта
5. Разработка программного продукта

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1.	Анализ состояния исследований и разработок диагностических систем	10%	21.09.2021-21.11.2021	
2.	Исследование методов прогнозирования осложнений при мочекаменной болезни	10%	22.11.2021-22.01.2022	
3.	Классификация и прогноз послеоперационных осложнений при мочекаменной болезни	20%	23.01.2022-23.04.2022	
4.	Проектирование программного продукта	20%	24.04.2022-29.01.2023	
5.	Разработка программного продукта	30%	30.01.2023-30.03.2023	
6.	Оформление выпускной работы	10%	31.03.2023-31.05.2023	

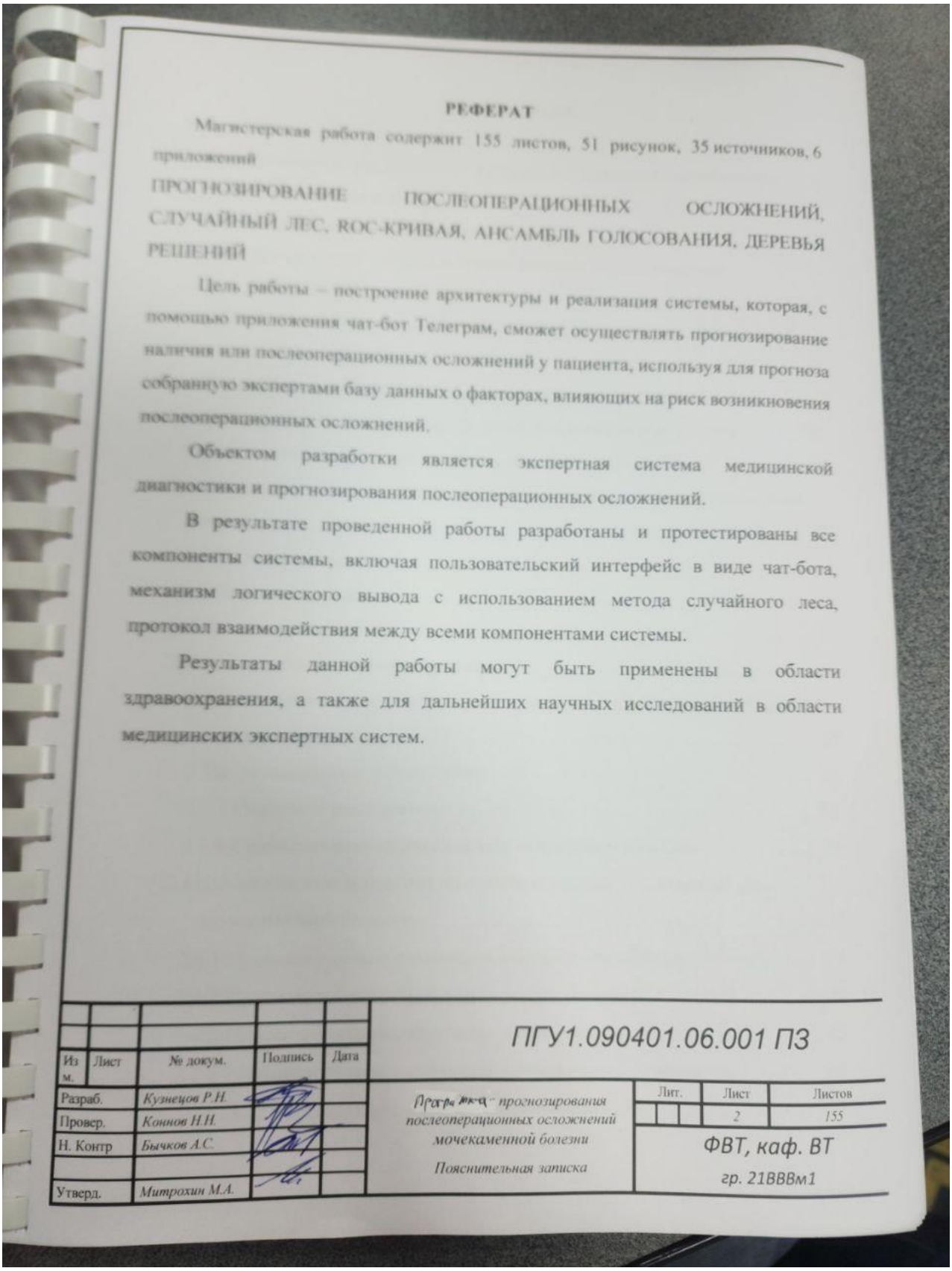
Дата выдачи задания "21" сентября 2023 г.

Руководитель магистерской диссертации _____ Коннов Н. Н.
(подпись, дата)

Задание к исполнению принял магистрант _____ Кузнецов Р. Н.
(подпись, дата)

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

3



РЕФЕРАТ

Магистерская работа содержит 155 листов, 51 рисунок, 35 источников, 6 приложений

ПРОГНОЗИРОВАНИЕ ПОСЛЕОПЕРАЦИОННЫХ ОСЛОЖНЕНИЙ, СЛУЧАЙНЫЙ ЛЕС, ROC-КРИВАЯ, АНСАМБЛЬ ГОЛОСОВАНИЯ, ДЕРЕВЬЯ РЕШЕНИЙ

Цель работы – построение архитектуры и реализация системы, которая, с помощью приложения чат-бот Телеграм, сможет осуществлять прогнозирование наличия или послеоперационных осложнений у пациента, используя для прогноза собранную экспертами базу данных о факторах, влияющих на риск возникновения послеоперационных осложнений.

Объектом разработки является экспертная система медицинской диагностики и прогнозирования послеоперационных осложнений.

В результате проведенной работы разработаны и протестированы все компоненты системы, включая пользовательский интерфейс в виде чат-бота, механизм логического вывода с использованием метода случайного леса, протокол взаимодействия между всеми компонентами системы.

Результаты данной работы могут быть применены в области здравоохранения, а также для дальнейших научных исследований в области медицинских экспертных систем.

Изм.	Лист	№ докум.	Подпись	Дата	ПГУ1.090401.06.001 ПЗ		
Разраб.		Кузнецов Р.Н.			Прогнозирование послеоперационных осложнений мочекаменной болезни	Лит.	Лист 2 / Листов 155
Провер.		Коннов Н.Н.					
Н. Контр		Бычков А.С.			Пояснительная записка	ФВТ, каф. ВТ гр. 21ВВВм1	
Утверд.		Митрохин М.А.					

СОДЕРЖАНИЕ

ВВЕДЕНИЕ 5
1 Анализ состояния исследований и разработок диагностических систем . . . 8
1.1 Анализ лабораторных методов диагностики 9
1.2 Анализ формальных методов диагностики 10
1.2.1 Диагностика с использованием методов математической статистики 12
1.2.2 Диагностика с использованием методов построения экспертных систем 13
1.3 Статистические методы в обработке экспертных данных. 14
1.4 Обзор отечественных и зарубежных диагностических систем 15
1.5 Постановка задачи диагностики состояния объекта 16
2 Исследование методов прогнозирования осложнений при мочекаменной болезни 17
2.1 Постановка задачи 17
2.2 Описание данных 18
2.2.1 Анализ эмпирического распределения 19
2.3 Отбор информативных признаков 24
2.3.1 Определение достоверности совпадений и различий характеристик зависимостей 24
2.3.2 Исследование зависимостей 27
2.3.3 Снижение размерности 33
2.3.4 Сравнение статистических методов выбора анализов 54
2.4 Классификация и прогноз послеоперационных осложнений при мочекаменной болезни 56
2.4.1 Прогнозирование с помощью метода к-ближайших соседей 58
2.4.2 Прогнозирование с помощью дерева решений 60
2.4.3 Прогнозирование с помощью логистической регрессии 65
2.4.4 Прогнозирование с помощью метода опорных векторов 68
2.4.5 Прогнозирование с помощью нейронных сетей 71

2.4.6 Прогнозирование с помощью алгоритма Случайный лес 82
2.5 Измерение качества классификации 85
3 Проектирование программного продукта 91
3.1 Компонентная архитектура 92
3.1.1 Однокомпонентная архитектура 93
3.1.2 Многокомпонентная архитектура 93
3.2 Архитектура программного обеспечени 95
3.3 Интерфейс Телеграмм бота 98
4 Разработка программного продукта 102
4.1 Выбранные технологии разработки 102
4.1.1 Анализ платформы для разработки. 102
4.1.2 Телеграмм Bot API 107
4.1.3 Среда разработки 109
4.2 Регистрация чат-бота Телеграмм 110
4.2.1 Реализация клиентской части чат-бота 112
ЗАКЛЮЧЕНИЕ 118
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 118
ПРИЛОЖЕНИЕ А Гистограммы значений лабораторных анализов 122
ПРИЛОЖЕНИЕ Б График нормального распределения вероятности 125
ПРИЛОЖЕНИЕ В Факторный анализ 128
ПРИЛОЖЕНИЕ Г Листинг программы 131
ПРИЛОЖЕНИЕ Д Минимальные и максимальные значения показателей крови 144
ПРИЛОЖЕНИЕ Е Презентация 145

ВВЕДЕНИЕ

Мочекаменная болезнь (МКБ) - одно из самых распространенных заболеваний, занимающее третье место после сердечно-сосудистых заболеваний и сахарного диабета. Взрослое население России и Европы страдает МКБ в 10-20% случаев [1-8]. Причиной болезни мочевыводящих путей у взрослых являются камни в мочевом пузыре, от которых страдают 20 миллионов лиц трудоспособного возраста в Северной Америке. МКБ встречается у каждого пятого мужчины и каждой третьей женщины. Около 2,5 млн операций на мочевыводящих путях проводятся ежегодно в мире, радикальная цистэктомия занимает второе место после аппендэктомии. Значительное увеличение числа операций связано с диагностикой на поздних стадиях, когда консервативные методы лечения малоэффективны или не применяются. Однако радикальная цистэктомия может привести к временной или стойкой утрате трудоспособности. Осложнения после открытых операций на мочевыводящих путях связаны с длительностью пребывания пациента в стационаре, что приводит к росту экономических затрат на лечение больных.

В настоящее время диагностика развивающихся осложнений в до- и послеоперационном периоде основана на клинико-лабораторных и биохимических исследованиях, которые не всегда точно отражают тяжесть патологического процесса и вероятность развития осложнений. Нет достоверных показателей, которые бы служили критерием прогноза течения послеоперационного периода, особенно у больных с МКБ [1-5].

Изменения состояния пациента наиболее ярко проявляются по исследованию крови, но это отражает лишь грубые нарушения функции органов и систем и такими методами не удается уловить большую часть функциональных нарушений мочевыводящих путей, создающих неблагоприятный фон для осложнений [1-5].

В случаях возникновения осложнений мочекаменной болезни показано оперативное лечение по жизненным показаниям. Увеличивается риск для жизни и здоровья пациента, значительно увеличивается время пребывания в

стационаре после операции, более тяжелый болевой синдром, вероятность возникновения послеоперационных осложнений выше, большой послеоперационный рубец, инфекционные осложнения. Из этого следует – хирургическое лечение неосложненной МКБ должно проводиться вовремя, до развития указанных состояний. Для достижения этой цели необходима большая работа по учету, единообразной регистрации послеоперационных осложнений, выявлению значимых факторов риска осложнений, разработке профилактических мероприятий по снижению уровня осложнений. Для этого необходимо выявление надежных прогностических параметров послеоперационных осложнений.

Основная цель исследования заключается в разработке модели диагностики объектов, которая опирается на нейронную сеть и алгоритмы функционирования интеллектуальной системы поддержки принятия решений.

В процессе решения данной задачи были выполнены следующие шаги:

- разработка методов подготовки данных для обучения нейронных сетей,
- выделение информативных признаков из экспериментальных данных,
- изучение и создание архитектур и алгоритмов для обучения нейронных сетей, а также разработка информационной модели интеллектуальной системы поддержки принятия решений,
- реализация программных средств обеспечения имитационного моделирования разрабатываемых алгоритмов и моделей для решения задач поддержки принятия решений,
- применение полученных теоретических результатов для построения модели имитации принятия решения при диагностике мочекаменной болезни.

При решении этих задач использованы результаты исследований больных с мочекаменной болезнью, полученных в ГУЗ «Пензенская областная клиническая больница им. Н. Н. Бурденко».

Основные методы исследования базируются на математическом аппарате теории нечетких множеств и нейронных сетей, теории вероятностей и математической статистики, теории информации и теории принятия решений,

методах вычислительной математики и математического программирования. Для практических исследований использована среды Goggle Colab, Matlab.

Настоящая магистерская диссертация состоит из введения, трёх разделов, заключения, списка использованных источников и приложений.

Первый раздел содержит существующие системы.

Во втором разделе представлены исследования методов классификации.

Третий раздел посвящен описанию работы разработанного кода.

1 Анализ состояния исследований и разработок диагностических систем

Диагностирование медицинских состояний – это задача классификации, где диагноз может содержать информацию о болезни или причинах ее возникновения. Классификация заболеваний может быть представлена в виде дерева, где каждый диагноз является терминальной вершиной, и к диагнозу можно подойти с помощью ответов на вопросы из каждой вершины. Однако, при постановке диагноза, мы не всегда можем получить прямые ответы, поэтому структура классификации болезней может только ограниченно помочь в диагностировании. Главная задача заключается в поиске диагностических признаков и правил на основе статистического материала.

Задачи диагностики заключаются в определении мер, направленных на отлаживание работы всех составляющих элементов системы и способов их реализации. Так как диагностирование – это исследовательский процесс, то на него автоматически переносятся все основные требования, предъявляемые к любому исследованию. Во-первых, оно должно быть основанным на достоверных первоисточниках; во-вторых, объективным, т.е. должна быть выработана процедура оценки и составления программы исследования объекта, учитывающая три обязательных условия: кто проводит исследование; где оно проводится и каковы его базовые параметры; в-третьих, должна быть обеспечена необходимая точность.

В общем виде диагностика обычно предполагает сравнение состояния выбранного объекта с какой-либо нормой или эталоном с тем, чтобы выявить наличие отклонений и определить их характер – положительный или отрицательный.

Алгоритмы для классификации и распознавания заболеваний в медицине используют три гипотезы. Первая гипотеза утверждает, что близкие описания в пространстве признаков соответствуют близким диагнозам при полном описании. Вторая гипотеза предполагает, что существует функциональный способ разделения классов с помощью линейной или

квадратичной разделимости. Третья гипотеза предполагает существование диагностического сочетания признаков, которые часто встречаются в одном конкретном классе. После предварительной диагностики принимаются решения о необходимости дальнейшего исследования объектов.

1.1 Анализ лабораторных методов диагностики

Существующие медицинские источники свидетельствуют о том, что заболевания мочевого пузыря имеют свои корни в раннем детстве. У людей с отношением к болезни мочевого камня в семье имеется риск возникновения этого заболевания в 2–4 раза выше. У родственников первой линии родства 56,4% детей всех возрастов с холелитиазом были выявлены камни в мочевом пузыре в результате обследования. Кроме того, известно о возможности развития мочекаменной болезни еще до рождения [5].

Факторы риска, способствующие развитию МКБ, включают в себя принадлежность к женскому полу (из-за уровня эстрогенов), возраст, ожирение, быструю потерю веса, беременность, расовую принадлежность, наследственность, сахарный диабет, болезнь Крона, цирроз печени, гемолитические анемии, гиперлипидемию, питание в «западном» стиле, прием клофибрата или эстрогенов, нарушения микробиоценоза кишечника и моторно-эвакуаторной функции кишечника (кишечная брадиаритмия, запоры), и др. факторы. Однако прием витамина С, умеренное употребление алкоголя, употребление кофе, орехов, физические упражнения могут действовать как факторы, предотвращающие развитие МКБ [6, 7].

Современное представление о том, как образуются камни в мочевом пузыре, основано на многих исследованиях. Они показали, что МКБ возникает из-за повреждения гепатоцитов, что приводит к образованию нестабильной мочи и последующей нуклеации кристаллов моногидрата холестерина, которые затем кристаллизуются в муциновом матриксе на фоне гипомоторной дисфункции мочевого пузыря. Этот процесс стал основой для создания классификации МКБ, которая рекомендуется использовать в клинической практике [8]:

– I стадия – начальная, или предкаменная,

– II стадия – формирование мочевыводящих путей,

– III стадия – хронический рецидивирующий калькулезный цистит,

– IV стадия – осложнения.

Авторы классификации отмечают, что развитие МКБ не всегда происходит по представленным стадиям, так как осложнения могут возникать на любой из них, включая первую стадию. Когда есть подозрение на мочекаменную болезнь, основным методом диагностики после сбора анамнеза и общего обследования является ультразвуковое исследование (УЗИ). Однако, типы камней в целом нельзя дифференцировать с помощью УЗИ. Например, кальцификация может быть выявлена только с помощью обзорной рентгенографии или компьютерной томографии (КТ).

Компьютерная томография с оценкой плотности по шкале Хаунсфилда является наиболее современным методом диагностики при МКБ. Обзорная рентгенография может выявлять конкременты только при плотности от +350 до +400 единиц Хаунсфилда, что может привести к ошибочному определению пероральной литолизы, так как плотность камней, подходящих для этого метода, не превышает +100 единиц. Для определения тактики лечения пациента с МКБ проводятся необходимые лабораторные исследования, включая клинический анализ крови, анализ мочи, копрограмму, оценку состояния микробиоты толстой кишки, биохимический анализ крови и, при необходимости, эзофагогастродуоденоскопию.

1.2 Анализ формальных методов диагностики

Методы прогнозирования осложнений после операции включают в себя широкий спектр технологий и подходов, в том числе клинические, иммунологические, молекулярные и генетические маркеры.

Клинические методы включают в себя оценку риска на основе факторов, таких как возраст, наличие хронических заболеваний, наличие инфекционных заболеваний, состояние пациента перед операцией. Различные шкалы и инструменты используются для оценки риска осложнений, таких как шкалы

Американского общества анестезиологии (ASA) и класификация Помери (POSSUM).

Иммунологические методы основаны на оценке функции иммунной системы. Например, прогностические маркеры, такие как общее количество лимфоцитов, отношение Т-клеток к Б-клеткам, отношение CD4 / CD8-клеток и уровень цитокинов, могут быть использованы для прогнозирования риска осложнений после операции.

Молекулярные маркеры включают в себя генетические варианты, связанные с болезнями и наследственными факторами. Например, полиморфизм в гене TNF-α может привести к повышению риска осложнений после операции.

Генетические маркеры могут быть использованы для оценки генетических факторов, связанных с болезнями и наследственными предрасположенностями. Например, генетические маркеры, связанные с нарушением метаболизма лекарств, могут использоваться для прогнозирования риска развития осложнений, связанных с анестезией.

Многие эти методы могут использоваться для определения риска осложнений у пациентов в зависимости от их индивидуальных характеристик и состояния здоровья, что может повысить качество здравоохранения и снизить количество осложнений после операций. Однако, для их применения требуется квалифицированный медицинский персонал и современное оборудование для диагностики.

Формальные методы медицинской диагностики являются эффективным средством для решения сложных проблем диагностики, позволяющих принимать обоснованные решения на основе точных вычислений. В данном обзоре рассматриваются основные формальные методы, используемые в медицинской диагностике.

Один из основных методов - решающие деревья. Этот метод используется для классификации объектов по ряду признаков. В медицинской диагностике этот метод может использоваться для классификации пациентов

на основе клинических данных. Решающие деревья используются для построения моделей, которые могут в дальнейшем использоваться в процессе диагностики.

Другой метод – метод байесовых сетей. Этот метод основан на вероятностной модели, которая позволяет оценить вероятность наличия заболевания у пациента на основе клинических данных и результатов исследований. Метод байесовых сетей широко используется в медицине, особенно в области онкологии.

Также используются методы искусственных нейронных сетей. Это математические модели, которые содержат нейроны, похожие на те, которые есть в нашем мозге. Их можно использовать для обучения компьютерных систем диагностике на основе исторических данный, а также в качестве прогностических инструментов, позволяющих определять, какие пациенты находятся в высоком риске заболевания.

Наряду с этими методами можно выделить методы математической логики, которые применяются для определения правил, с помощью которых может быть обеспечена автоматическая диагностика. В медицинской диагностике эти методы используются для разработки экспертных систем.

Также стоит отметить методы кластерного анализа, которые позволяют выявлять группы пациентов схожих диагностических признаков, профилей риска, что может существенно повысить эффективность диагностики.

Таким образом, формальные методы медицинской диагностики имеют широкий спектр применения и используются для различных задач, связанных с диагностикой в медицине. Они позволяют повысить точность и эффективность диагностики, а также улучшить результаты лечения пациентов.

1.2.1 Диагностика с использованием методов математической статистики

Методы математической статистики используются для анализа экспериментальных данных, включая описательную статистику, сравнение

выборок, сокращение размерности признаков, выявление зависимостей между переменными и классификацию. В задачах диагностики дискриминантный анализ может быть применен для определения классов объектов с помощью обучающих выборок. Кластерный анализ и группировка используются для выделения классов, а регрессионный анализ для исследования зависимости диагноза от входных факторов. Метод на основе байесовской вероятности позволяет рассчитать условную вероятность диагностических решений на основе априорной и условной вероятности наблюдения признаков.

Но возникают сложности в организации обработки больших объемов информации, а также необходимости учитывать качественные аспекты информации. Кроме того, структурная сложность объектов диагностики не учитывается, а полученные диагностические результаты не всегда могут быть доступно объяснены пользователям [9-13].

1.2.2 Диагностика с использованием методов построения экспертных систем

В настоящее время информационное обеспечение различных медицинских технологий становится все более важным. Применение современных информационных технологий является ключевым фактором для развития большинства отраслей знания и практической деятельности. Разработка и внедрение информационных систем становятся одной из наиболее важных задач.

С использованием новейших технологий можно значительно улучшить представление о течении заболевания путем создания математических моделей. Однако, из-за высокой информационной загрузки в сфере медицины, появляется сложность в выборе между полным анализом данных и компьютерной поддержкой принятия решений. В этой ситуации медицинские экспертные системы могут прийти на помощь врачам, помогая им не только проверять диагностические предположения, но и получать консультации в сложных диагностических случаях [9-10].

«Инженерия знаний» — это область исследований, которая занимается формализацией способов представления знаний и созданием экспертных систем. Термин введен Е. Фейгенбаумом, который считает, что этот подход является ключевым для решения трудных прикладных проблем, требующих знаний экспертов. Экспертные системы используются для решения неформализованных проблем, таких как задачи, которые не могут быть представлены в числовой форме, имеют неоднозначные и неточные данные, не имеют четко определенной целевой функции и не имеют однозначного алгоритмического решения.

В медицинских задачах, где огромный объем многомерных и запутанных клинических данных могут быть противоречивыми, все эти свойства являются типичными. Экспертные системы могут использоваться для решения задач диагностики, дифференциальной диагностики, прогнозирования, выбора стратегии и тактики лечения [11-12].

Экспертные системы обычно включают несколько основных компонентов, таких как базу знаний, рабочую память, механизм вывода решения, объяснительный компонент, компонент приобретения знаний и пользовательский интерфейс. Согласно известной литературе [14-18], экспертные системы функционируют в двух основных режимах: приобретения знаний и решения задач.

В режиме приобретения знаний эксперты вносят свои знания в базу знаний ЭС, что позволяет автоматизировать процесс принятия диагностических решений. В режиме решения задач пользователи получают диагностическое решение и, при необходимости, объяснение к нему в процессе взаимодействия с экспертной системой [19].

1.3 Статистические методы в обработке экспертных данных

В настоящее время все большую важность приобретает правильное использование статистических методов в планировании медицинских исследований, с целью минимизации случайных ошибок в научных исследованиях.

Статистика в медицине является одним из инструментов анализа экспериментальных данных и клинических наблюдений, и используется для диагностики, классификации и поисков новых закономерностей.

Важно отметить, что выборка, использованная в исследовании, должна быть репрезентативной для генеральной совокупности, которую она представляет. Поэтому при планировании исследования необходимо сформулировать четкую цель и строго соблюдать критерии включения и исключения в статистический анализ.

1.4 Обзор отечественных и зарубежных диагностических систем

Медицинская компьютерная система – это любая система, которая обеспечивает компьютерную реализацию задач медицинского назначения. Такая система, как правило, представляется совокупностью программ, предназначенных для решения задач определенного типа и объединяемых общим интерфейсом. Здесь под типами задач подразумеваются как прикладные задачи, так и типы обработки информации, основывающиеся на определенных структурах и способах хранения, и представлениях данных.

К числу медицинских систем (МС) могут быть отнесены медицинские информационные системы, системы обработки данных (СОД), системы управления в здравоохранении; диагностические системы, информационно-справочные системы и т.п.

Обзор отечественных и зарубежных диагностических медицинских систем является сложной и многоступенчатой задачей, поскольку на рынке представлено множество продуктов разных производителей и разных типов.

Одними из наиболее известных отечественных диагностических медицинских систем являются:

- система компьютерной томографии "Тоскар-1",
- ЭКГ система компьютерной диагностики "Кардиоденс",
- радиологический компьютерный комплекс "Регион",
- система компьютерной диагностики "Магнитотерапия",
- диагностическая система "ДИФРАКТ".

Среди зарубежных систем можно выделить следующие:

- система компьютерной томографии "Siemens Somatom Definition Flash" (Германия),

- рентгеновский компьютерный комплекс "GE Discovery XR656" (США),

- эхокардиограф "Philips EPIQ 7" (Нидерланды),

- система интраоперативной нейронавигации "Brainlab VectorVision" (Германия),

- система компьютерной томографии "Toshiba Aquilion Lightning" (Япония).

Каждая из этих систем имеет свои особенности и преимущества, и выбор конкретной системы зависит от многих факторов, таких как цена, функциональность, надежность, скорость и точность диагностики, а также от потребностей и требований клиники или больницы, которая будет использовать данную систему.

1.5 Постановка задачи диагностики состояния объекта

Одна из задач диагностики заболеваний заключается в определении наличия или отсутствия осложнений. Для этого используются данные, характеризующие состояние больного, которые называются признаками. Задача постановки диагноза заключается в определении принадлежности больного к тому или иному типу заболевания на основе имеющихся признаков. Если множества признаков для разных диагнозов не пересекаются, то задача решается построением разделяющей поверхности, и диагноз устанавливается однозначно.

В случае, когда множества признаков пересекаются, построение разделяющей поверхности также требуется, но диагноз определяется с некоторой вероятностью. Для диагностики нескольких заболеваний используется аналогичный метод.

2 Исследование методов прогнозирования осложнений при мочекаменной болезни

В данной главе рассматривается исследование методов прогнозирования послеоперационных осложнений при мочекаменной болезни, включающее проверку соответствия лабораторных анализов крови нормальному закону распределения, исследование применения способа заполнения пропусков в медицинских данных, отбор информативных признаков с использованием корреляционного анализа и метода выделения маркеров, а также применение метода деревьев решений и различных архитектур искусственных нейронных сетей для выбора лучшей модели прогнозирования осложнений.

2.1 Постановка задачи

Для прогнозирования послеоперационных осложнений у пациентов с мочно-каменной болезнью была создана группа из 109 историй болезни, полученных отделением торакальной хирургии клинической больницы имени Н.Н. Бурденко. Среди них 63 случая были без осложнений, а 46 случаев осложнены. Для прогнозирования использовались 15 показателей, основанных на общем и биохимическом анализах крови, а также на длительности оперативных вмешательств и пола пациента. Полный перечень анализов включал гемоглобин, эритроциты, цветовой показатель, лейкоциты, нейтрофилы палочкоядерные и сегментоядерные, лимфоциты, моноциты, скорость оседания эритроцитов, общий белок, общий билирубин, амилазу, мочевину, креатинин, глюкозу, а также указание на пол и длительность операции (в минутах). Минимальные и максимальные значения каждого показателя также были указаны Приложении Д.

Перед тем как применять комплекс методов и алгоритмов предобработки данных необходимо выполнить оценку качества данных с целью выявления наиболее характерных проблем и уровня их сложности, а также для выработки стратегии предобработки. Качество данных означает пригодность данных для конкретных целей, задач и методов анализа. В результате такой оценки, возможно, будет принято решение отказаться от

предобработки некоторых данных, поскольку это не приведет к желаемым результатам. Кроме того, применение некоторых методов предобработки может ухудшить данные, а предварительная оценка качества данных позволит по возможности избежать этого. Для оценки качества данных использовалась аналитическая платформа Deductor Academic 5.3. Для этого в систему была загружена группа наблюдений, состоящая из 109 историй болезни пациентов. Результаты оценки качества данных представлены на рисунке 2.1.

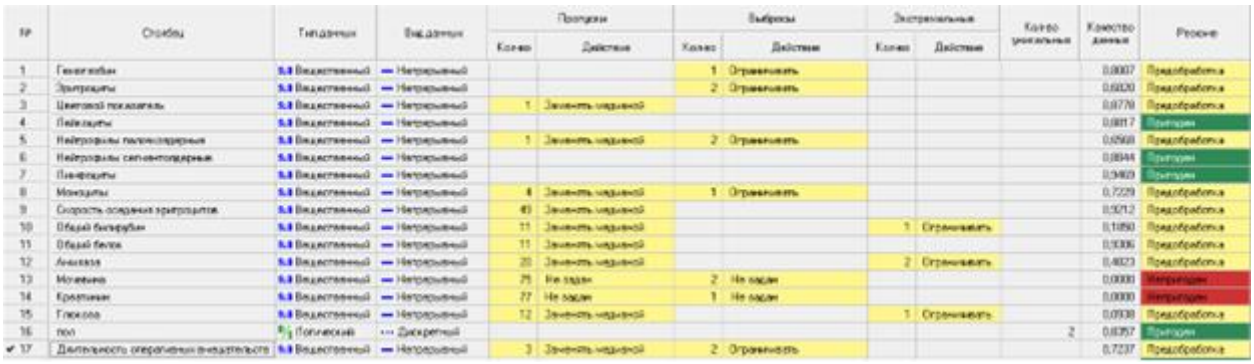


Рисунок 2.1 – Результаты оценки качества данных

Из рисунка 2.1 видно, что два показателя необходимо удалить из обучающей выборки, так как они не пригодны для дальнейшей обработки из-за отсутствия более 50% данных анализов по всем пациентам. Из рисунка Рисунок 2.1 видно, что 9 показателей содержат пропуски, которые необходимо заполнить, 5 показателей содержит выбросы и 3 показателя имеют экстремальные значения. После полной проверки осталось 15 показателей: гемоглобин, эритроциты, цветовой показатель, лейкоциты, нейтрофилы палочкоядерные, нейтрофилы сегментоядерные, лимфоциты, моноциты, скорость оседания эритроцитов, общий белок, общий билирубин, амилаза, глюкоза, пол, длительность оперативных вмешательств.

2.2 Описание данных

Одной из главных составляющих анализа данных является описательная статистика (дескриптивная статистика), которая предоставляет краткую числовую и графическую характеристику изучаемого явления. Получить точное значение параметра популяции (среднее значение, медиану, долю) невозможно за исключением случаев, когда все члены популяции участвуют в

исследовании. Однако можно оценить значение параметра популяции по выборке, при этом точность оценки зависит от методики измерения, объема и представительности выборки и биологической вариации. Показатели описательной статистики можно классифицировать на несколько групп, включая показатели положения, разброса, асимметрии и графические представления результатов, такие как гистограмма и частотная диаграмма.

2.2.1 Анализ эмпирического распределения

Общие характеристики генеральных совокупностей связаны с тем, как переменная (признак) распределена, т.е. насколько часто встречаются определенные значения в абсолютном или относительном выражении. В свою очередь, выборка характеризуется выборочным эмпирическим распределением значений признака.

Выборочное эмпирическое распределение представляет собой частотное распределение выборки по категориям значений признака. Оно помогает определить, какие значения признака встречаются чаще или реже в выборке, а также позволяет проанализировать отклонения от теоретического распределения. Однако следует учитывать, что выборочное распределение не является точной копией генерального распределения и может отличаться от него. Поэтому, при анализе результатов исследования на основе выборки необходимо учитывать такие факторы, как размер и репрезентативность выборки, выбранный метод и схема отбора образцов, а также учитывать возможную ошибку выборки.

Также при анализе результатов на основе выборки необходимо понимать, что статистические выводы, делаемые на основе выборки, могут иметь определенную степень неопределенности и риски ошибки. Эта неопределенность зависит от объема выборки и уровня значимости выбранного статистического теста. Аналитикам необходимо учитывать эту неопределенность, чтобы принимать более точные и обоснованные решения на основе анализа выборки. В целом, понимание выборочного эмпирического распределения и его отношений к генеральному распределению может помочь

сделать осмысленные выводы на основе анализа выборки, что является важным инструментом в многих областях, таких как медицина, экономика и социология.

В идеале свойства выборки должны максимально близко и без искажений отражать свойства генеральной совокупности, а эмпирическое распределение приближаться к распределению и генеральной совокупности. Анализ распределений направлен на выявление закономерности изменения частот в зависимости от значений варьирующего признака и анализ различных характеристик изучаемого распределения. Прежде, чем приступить к вычислению специальных статистических показателей, необходимо для исходной совокупности провести проверку на соответствие закону распределения медицинских анализов.

Для анализа связи между возможными значениями переменной и частотой их наблюдений в выборке используется эмпирическое распределение, представленное в виде статистического ряда распределения. Для построения такого ряда в выборке необходимо иметь достаточно много наблюдений - обычно несколько десятков или больше, которые группируются по интервалам. Выбор количества интервалов m для группировки может быть выполнен с использованием формулы Стерджесса [20]:

$$m = 1 + 3{,}322\, lg\, N$$

где N– объем выборки.

Подготовка данных для статистического ряда распределения выполнялась в следующем порядке:

– в зависимости от числа наблюдений N рассчитываем число интервалов ряда m,

– определяем размах вариационного ряда $R = xmin_{max}$,

– рассчитываем длину интервала $h = R/m$,

– определяем границы и средние точки интервалов,

– подсчитываем частоту наблюдений, частость и накопленные частоту и частость для каждого интервала.

Гистограмма была построена на основе статистического ряда распределения, который включал в себя данные о характеристиках выборки. По ней были сделаны предварительные выводы о соответствии эмпирического распределения нормальному распределению. Гистограмма была построена при помощи программы StatSoft STATISTICA 10.0 и содержала информацию о частоте наблюдений в каждом интервале ряда показателей, отображенных по горизонтали. По вертикали же отложены значения наблюдений в соответствующем интервале.

По результатам графического анализа, представленным в приложении А, было установлено, что Гемоглобин, Лимфоциты и Общий белок имеют распределение, близкое к нормальному, в то время как остальные показатели не подчиняются данному распределению.

Для дальнейшего исследования были вычислены статистические показатели, описывающие положение экспериментальных данных на числовой оси для 13 показателей общего и биохимического анализа крови и 2 дополнительных показателя из обучающего множества. Полученные результаты представлены на рисунке 2.2

№	Название	Максимум	Минимум	Среднее значение	Медиана	Мода
1.	Гемоглобин	173	69	137,77	140	146
2.	Эритроциты	5,1	2,5	4,40	4,5	4,6
3.	Цветовой показатель	1	0,82	0,93	0,93	0,93
4.	Лейкоциты	20,5	3	8,59	7,5	6,4
5.	Нейтрофилы палочкоядерные	43	1	8,73	6	2
6.	Нейтрофилы сегментоядерные	80	39	62,69	64	69
7.	Лимфоциты	47	2	23,91	23	23
8.	Моноциты	15	1	3,85	3	3
9.	Скорость оседания эритроцитов	64	3	21,57	15	15
10.	Общий билирубин	722,5	8,5	34,78	15,2	15
11.	Общий белок	85	57,5	70,17	69	68
12.	Амилаза	146,3	8,5	27,96	27	27
13.	Глюкоза	62	3,8	6,76	6	5,3
14.	Длительность оперативных вмешательств	240	30	77,11	65	60
15.	Пол	1	0	0,73	1	1

Рисунок 2.2 – Максимум и минимум элементов выборки, среднее значение, медиана, мода

Указанные меры центральной тенденции данных (среднее, медиана, мода) сообщают о нормальности распределения признака. При нормальном распределении все три показателя более или менее совпадают, а при асимметричном распределении — нет. Из рисунка 2.2 видно, что меры центральной тенденции данных совпадают у таких показателей как цветовой показатель, лимфоциты, моноциты, амилаза. Это говорит о том, что эмпирическое распределение данных показателей близкое к нормальному.

Для применения описательной статистики необходимо учитывать тип данных и параметры распределения, которые характеризуются асимметрией и эксцессом, которые являются показателями отклонения формы распределения от нормальной формы [21]. На рисунке 2.3 представлены значения асимметрии и эксцесса, полученные в результате анализа пациентов.

№	Название	Асимметрия	Эксцесс
1.	Гемоглобин	-0,87	1,64
2.	Эритроциты	-1,81	4,59
3.	Цветовой показатель	-0,38	-0,05
4.	Лейкоциты	1,04	0,40
5.	Нейтрофилы палочкоядерные	1,84	4,11
6.	Нейтрофилы сегментоядерные	-0,51	0,04
7.	Лимфоциты	0,05	-0,81
8.	Моноциты	1,35	2,50
9.	Скорость оседания эритроцитов	1,50	1,33
10.	Общий билирубин	7,36	63,92
11.	Общий белок	0,30	-0,18
12.	Амилаза	4,06	19,74
13.	Глюкоза	9,16	90,81
14.	Длительность оперативных вмешательств	2,05	5,28
15.	Пол	-1,07	-0,86

Рисунок 2.3 – Значения эксцесса и асимметрии анализов пациентов

Результаты, отраженные на рисунке 2.3, свидетельствуют о том, что в большинстве случаев значения эксцесса у пациентов превышают нормальный уровень в 0, указывая на их несоответствие нормальному закону распределения и более высокую точку максимума. Из рисунка 2.3 видно, что многие из анализов у пациентов имеют смещение относительно среднего значения, отраженное также на графиках нормального распределения

вероятности (Приложение Б). Данный график выявляет отклонения на основе коэффициента асимметрии и эксцесса.

Критерий Шапиро-Уилка W является наиболее эффективным для определения нормальности, так как он обладаем большей мощностью по сравнению с альтернативными критериями проверки нормальности.

$$W = \frac{S^2}{\sum_{i=1}^{n}(X_i - \bar{X})^2},$$

Критерий Шапиро-Уилка основан на оптимальной линейной несмещённой оценке дисперсии S к её обычной оценке методом максимального правдоподобия.

$$S = \sum_k a_k \left[X_{(n-1-k)} - X_{(k)}\right],$$

где индекс k изменяется от 1 до $n/2$ или от 1 до $(n-1)/2$ при четном и нечетном n соответственно.

Коэффициенты a_k приведены в стандарте [22, 23].

Для выполнения теста Шапиро-Уилка выборка должна быть объемом не меньше 3 и не больше 5000. Нулевая гипотеза H_0 теста Шапиро–Уилка заключается в том, что случайная величина, выборка которой известна, распределена по нормальному закону. Альтернативная гипотеза H_1 заключается в том, закон распределения не является нормальным. Полученные данные представлены на рисунке 2.4.

№	Название	Вероятность справедливости гипотезы
1.	Гемоглобин	0,0011
2.	Эритроциты	0,0000
3.	Цветовой показатель	0,0040
4.	Лейкоциты	0,0000
5.	Нейтрофилы палочкоядерные	0,0000
6.	Нейтрофилы сегментоядерные	0,0241
7.	Лимфоциты	0,0119
8.	Моноциты	0,0000
9.	Скорость оседания эритроцитов	0,0000
10.	Общий билирубин	0,0000
11.	Общий белок	0,0537
12.	Амилаза	0,0000
13.	Глюкоза	0,0000
14.	Длительность оперативных вмешательств	0,0000
15.	Пол	0,0000

Рисунок 2.4. – Значения вероятности теста Шапиро-Уилка анализов пациентов

Поскольку вероятности справедливости гипотезы H_0 оказалась меньше уровня значимости ($\alpha = 0{,}05$), мы принимаем, что она не верна.

Из этого следует, что для обработки данных доступны не все методы статистического анализа. Будем применять непараметрические методы, которые позволяют исследовать данные без каких-либо допущений о характере распределения переменных, в том числе при нарушении требования нормальности распределения.

2.3 Отбора информативных признаков

В медицинских исследованиях изучаются множество связей, но только некоторые из них имеют статистическую и клиническую значимость. Нельзя заранее определить, сколько ценных связей будет найдено. Поэтому важно правильно отобрать признаки для исследования, потому что это может помочь понять механизмы проблемы. Это может быть даже более важным, чем быстрая обработка данных или повышение точности классификации.

2.3.1 Определению достоверности совпадений и различий характеристик зависимостей

В данном разделе описаны общие методы определения достоверности совпадений и отличий характеристик изучаемых объектов с целью выявления самых информативных признаков.

Типовой задачей анализа данных в медико-биологических исследованиях является установление совпадений или различий характеристик экспериментальной и контрольной группы. Для этого формулируются статистические гипотезы [24, 25]:

- гипотеза об отсутствии различий (так называемая нулевая гипотеза),

- гипотеза о значимости различий (так называемая альтернативная гипотеза).

В целях принятия решений по выбору гипотезы (нулевой или альтернативной) необходимо вычислить эмпирическое значение критерия, которое затем сравнивается с критическим значением (известным, как правило, таблично). Критические значения обычно указываются для нескольких уровней значимости (обычно 0,05, 0,01 и 0,001), но в медико-биологических исследованиях чаще всего используется уровень значимости 0,05 с максимальной допустимой ошибкой в 5%. Если эмпирическое значение критерия меньше или равно критическому, то принимается альтернативная гипотеза, в противном случае - нулевая гипотеза. Используя непараметрический метод Вилкоксона-Манна-Уитни для проверки гипотезы о совпадении характеристик двух групп, данный критерий оперирует не абсолютными значениями элементов, а результатами их парных сравнений [25, 26].

Условия применимости метода: анализируемые признаки должны быть количественными или порядковыми. Проверяется гипотеза о равенстве средних рангов (в отличие от t-критерия, где проверяется гипотеза о равенстве средних значений).

Для определения достоверности совпадений и различий в экспериментальных данных, измеренных в шкале отношений, применяется алгоритм, основанный на критерии Вилкоксона-Манна-Уитни.

Сначала составляется единый ранжированный ряд N из обеих сопоставляемых выборок.

$$N = n_1 + n_2,$$

где n_1 – количество элементов в первой выборке,

n_2 – количество элементов во второй выборке.

Затем этот ряд разделяется на два и подсчитываются суммы рангов T_x, соответствующие каждой выборке. Большей из двух сумм соответствует выборка с большим количеством элементов.

Затем по формуле определяется значение U-критерия Манна-Уитни.

$$U = n_1 \cdot n_2 + \frac{n_x\left(n_x + 1\right)}{2} - T_x$$

где n_1 — количество элементов в первой выборке,

n_2 — количество элементов во второй выборке,

T_x— сумма рангов, соответствующую выборке с n_x элементами.

По таблице находится критическое значение критерия для избранного уровня статистической значимости. Если значение U-критерия меньше табличного, то признается наличие существенного различия между уровнем признака в рассматриваемых выборках. Результаты применения этого критерия для сравнения выборок больных с осложнениями и больных без осложнений представлены на рисунке 2.5.

Результаты применения критерия Вилкоксона-Манна-Уитни для сравнения двух выборок больных с осложнениями и больных без осложнений представлены на рисунке 2.5:

№	Название	Статистика U Манна-Уитни	Статистика W Уилкоксона	Z	Уровень значимости	Гипотеза
1.	Гемоглобин	1235,00	2316,00	-1,314	0,19	нулевая гипотеза
2.	Эритроциты	1299,50	2380,50	-0,928	0,35	нулевая гипотеза
3.	Цветовой показатель	1215,00	2296,00	-1,441	0,14	нулевая гипотеза
4.	Лейкоциты	1265,00	2146,00	1,553	0,011	альтернативная гипотеза
5.	Нейтрофилы палочкоядерные	1860,00	2941,00	2,529	0,011	альтернативная гипотеза
6.	Нейтрофилы сегментоядерные	1365,50	2446,50	-0,513	0,608	нулевая гипотеза
7.	Лимфоциты	1108,00	2189,00	-2,094	0,036	альтернативная гипотеза
8.	Моноциты	1508,00	2589,00	0,367	0,713	нулевая гипотеза
9.	Скорость оседания эритроцитов	1543,00	2624,00	0,615	0,538	нулевая гипотеза
10.	Общий билирубин	1939,50	3020,50	3,015	0,003	альтернативная гипотеза
11.	Общий белок	1372,00	2453,00	-0,474	0,636	нулевая гипотеза
12.	Амилаза	1678,50	2759,50	1,417	0,157	нулевая гипотеза
13.	Глюкоза	1613,00	2694,00	1,009	0,313	нулевая гипотеза
14.	Длительность оперативных вмешательств	1883,00	2964,00	2,672	0,008	альтернативная гипотеза
15.	Пол	1298,50	2379,50	-1,206	0,228	нулевая гипотеза

Рисунок 2.5 – Результат применения критерия Вилкоксона-Манна-Уитни

Проанализировав результаты представленные на рисунке 2.5 можно сделать вывод о том, что средние значения анализов больных с осложнениями и больных без осложнений значимо различаются у следующих показателей: нейтрофилы палочкоядерные, лимфоциты, общий билирубин, длительность оперативных вмешательств, амилаза. Поэтому эти факторы следует включить в выборку для системы прогнозирования.

2.3.2 Исследование зависимостей

Одной из важных задач медицинского исследования является получение связи между факторами, воздействующими на организм, и параметром-откликом на это воздействие, а также моделирование этого параметра в зависимости от действующего фактора. Связь между переменными

величинами может быть функциональной или статистической (корреляционной).

Функциональная связь между переменными, при которой каждому значению одной величины соответствует строго определенное значение другой, то есть $Y = F(X_1, X_2, \ldots, X_n)$. В медико-биологических исследованиях сталкиваются с функциональной связью очень редко, поскольку объекты этих исследований имеют большую индивидуальную изменчивость. С другой стороны, характеристики биологических объектов зависят, как правило, от комплекса большого числа сложных взаимосвязей и не могут быть сведены к отношению двух или трех факторов.

Статистическая связь означает, что значение одной переменной изменяется в среднем в зависимости от того, какие значения принимает другая. Очень часто рассматривается как функциональная зависимость со случайной ошибкой, то есть:

$$Y = F(X_1, X_2, \ldots, X_n) + \varepsilon_1,$$

где $F(X_1, X_2, \ldots, X_n)$ – функция, описывающая зависимость Y от совокупности независимых переменных $X_1, X_2, \ldots, X_n$;

ε – некоторая случайная ошибка.

Известно, что сумма константы и случайной величины является случайной величиной. В связи с этим значения Y, рассчитанные по указанной формуле, будут вследствие добавления случайной величины ε также случайными величинами.

Следует отметить, что в случае биологических факторов тот или иной характер связи сохраняется, как правило, только в определенном интервале изменений признаков. За пределами этого интервала связь может ослабнуть, стать прямо противоположной по направлению либо совсем исчезнуть.

Корреляционная связь представляет собой частный случай статистической связи $M(Y|X = x) = \bar{y}(x)$, то есть математическое ожидание

переменной Y, при условии, что случайная величина X принимает значение x.

В ходе анализа корреляционной связи решаются четыре взаимосвязанных задачи

1) установление направления и формы корреляционной связи,

2) оценка плотности связи,

3) оценка репрезентативности статистических оценок взаимосвязей, полученных по выборочным данным (величина ошибки, уровень значимости),

4) установление величины детерминации (доли взаимовлияния) коррелируемых факторов.

Статистическая связь описывается с помощью различных методов, таких как: методы корреляционного, регрессионного и дисперсионного анализа.

В задачах, которые решаются дисперсионным анализом, присутствует отклик числовой природы, на который воздействует несколько переменных, имеющих номинальную природу. Так как в работе данные не отвечают нормальному распределению, результаты применения дисперсионного анализа оказываются искаженными и могут быть неправильно интерпретированы в дальнейшем. В связи с этим в данной работе этот метод не рассматривался.

Для выявления зависимостей между переменными и результатом (наличие – отсутствие осложнений) был проведен корреляционный анализ. Так как данные не подчиняются нормальному закону распределения, целесообразно использовать непараметрические коэффициенты связи – коэффициенты ранговой корреляции Спирмена и Кендэла.

Метод ранговой корреляции Спирмена может использоваться при данных, измеренных в интервальной шкале, потому что он не является параметрическим и отражает тенденцию к изменению переменных в одном направлении. Этот метод обозначается, как и определяется путем сравнения рангов, то есть номеров значений, которые упорядочиваются для

сравниваемых переменных. Формула для вычисления коэффициента Спирмена имеет вид [27, 28, 29]:

$$\rho = 1 - \frac{6\sum d^2}{n(n^2-1)},$$

где $\sum d^2$ – сумма квадратов разностей рангов;

n – число парных наблюдений.

Важно отметить, что коэффициент корреляции Спирмена можно использовать для оценки связи между переменными, независимо от их распределения благодаря тому, что все детали, связанные с конкретными распределениями, исчезают при замене значений переменных на их ранги в выборке. Кроме того, этот коэффициент менее чувствителен к выбросам, что является еще одним важным качеством при анализе экспериментальных данных.

Коэффициент ранговой корреляции Кендалла, определяемый по следующей формуле [27, 28, 29], применяется для выявления связи между количественными или качественными показателями, если они могут быть упорядочены.

$$\tau = \frac{4\sum_{i=1}^{n-1} R_i}{n(n-1)} - 1,$$

где n — объем выборки (число сопоставляемых пар);

R_i — число рангов во втором вариационном ряду, бóльших, чем данное ранговое число и расположенных ниже него.

Достоверность коэффициентов корреляции Спирмена и Кендэлла оценивается на основе рассчитанного t-критерия Стьюдента:

$$t = \frac{|r_{xy}|}{\sqrt{\frac{1-r^2{}_{xy}}{n-2}}},$$

где $\sqrt{\frac{1-r^2{}_{xy}}{n-2}}$ — средняя квадратичная ошибка коэффициента корреляции;

r_{xy} — коэффициент корреляции.

После расчета значений t-критерия производится их сравнение со значением критического значения, соответствующего уровню значимости. Если уровень значимости равен 0,05, то при коэффициенте корреляции ниже этого значения (т.е. при p >0,05), он считается незначимым. Аналогично, при уровне значимости 0,01 коэффициент корреляции считается значимым при p <0,01.

Исследование было проведено при помощи статистического пакета StatSoft STATISTICA 10.0, так как большинство переменных не подчиняются нормальному распределению. Для сравнения разных распределений данных были вычислены все коэффициенты корреляции. Результаты этих вычислений представлены на рисунках 2.6–2.7, которые отражают связь между различными переменными и наличием или отсутствием осложнений.

Эти таблицы позволяют оценить степень взаимосвязи между показателями и наличием осложнений. Например, на рисунке 2.6 отражает связь между различными показателями анализов (гемоглобин, лейкоциты, тромбоциты) и наличием осложнений. Видно, что есть показатели, у которых высокий коэффициент корреляции, что указывает на сильную связь между ними и наличием осложнений. Например, уровень гемоглобина в крови имеет коэффициент корреляции -0,126, что говорит о незначимости показателя.

Таким образом, анализ коэффициентов корреляции, проведенный при помощи StatSoft STATISTICA 10.0, позволил оценить степень взаимосвязи между показателями и наличием осложнений. Полученные результаты могут быть использованы для дальнейшего исследования и определения факторов, влияющих на наличие осложнений у пациентов.

№	Название	Критерий Спирмена	t-критерия Стьюдента	Значимость коэффициента корреляции
1.	Гемоглобин	-0,126	1,318	незначим
2.	Эритроциты	-0,089	0,927	незначим
3.	Цветовой показатель	-0,138	1,449	незначим
4.	Лейкоциты	0,255	0,735	незначим
5.	Нейтрофилы палочкоядерные	0,253	2,709	значим
6.	Нейтрофилы сегментоядерные	-0,049	0,511	незначим
7.	Лимфоциты	-0,201	2,127	значим
8.	Моноциты	0,035	0,366	незначим
9.	Скорость оседания эритроцитов	0,059	0,614	незначим
10.	Общий билирубин	0,290	3,135	значим
11.	Общий белок	-0,045	0,472	незначим
12.	Амилаза	0,336	2,423	значим
13.	Глюкоза	0,097	1,009	незначим
14.	Длительность оперативных вмешательств	0,257	2,752	значим
15.	Пол	-0,116	1,209	незначим

Рисунок 2.6 – Итоги корреляционного анализа Спирмена

Из представленной таблицы видно, что коэффициенты корреляции оказались статистически значимыми и, кроме того, свидетельствуют о корреляции с результатом у следующих показателей: Амилаза, Нейтрофилы палочкоядерные, Общий билирубин, Длительность оперативных вмешательств.

№	Название	Критерий Кендалла	t-критерия Стьюдента	Значимость коэффициента корреляции
1.	Гемоглобин	-0,104	1,088	незначим
2.	Эритроциты	-0,077	0,804	незначим
3.	Цветовой показатель	-0,117	1,227	незначим
4.	Лейкоциты	0,202	1,138	незначим
5.	Нейтрофилы палочкоядерные	0,205	2,169	значим
6.	Нейтрофилы сегментоядерные	-0,041	0,427	незначим
7.	Лимфоциты	-0,267	2,868	значим
8.	Моноциты	0,031	0,320	незначим
9.	Скорость оседания эритроцитов	0,052	0,544	незначим
10.	Общий билирубин	0,241	2,567	значим
11.	Общий белок	-0,038	0,395	незначим
12.	Амилаза	0,114	2,191	значим
13.	Глюкоза	0,081	0,841	незначим
14.	Длительность оперативных вмешательств	0,216	2,298	значим
15.	Пол	-0,116	1,209	незначим

Рисунок 2.7 – Итоги корреляционного анализа Кендалла

Из представленной таблицы видно, что коэффициенты корреляции оказались статистически значимыми и, кроме того, свидетельствуют о

корреляции с результатом у следующих показателей: амилаза, нейтрофилы палочкоядерные, лимфоциты, общий билирубин, длительность оперативных вмешательств.

Проанализировав рисунки 2.6–2.7 можно сделать вывод о том, что наиболее коррелированными с результатом анализами являются: лейкоциты, нейтрофилы палочкоядерные, общий билирубин, длительность оперативных вмешательств. Эти факторы следует включить в выборку для системы прогнозирования.

2.3.3 Снижение размерности

Проблема отбора параметров решается в большой области знаний, под названием Dimensionality Reduction (уменьшение размерности). В свою очередь, эта область знаний делится на две части Feature Extraction (преобразование параметров) и Feature Selection (выбор параметров) (рисунок 2.8).

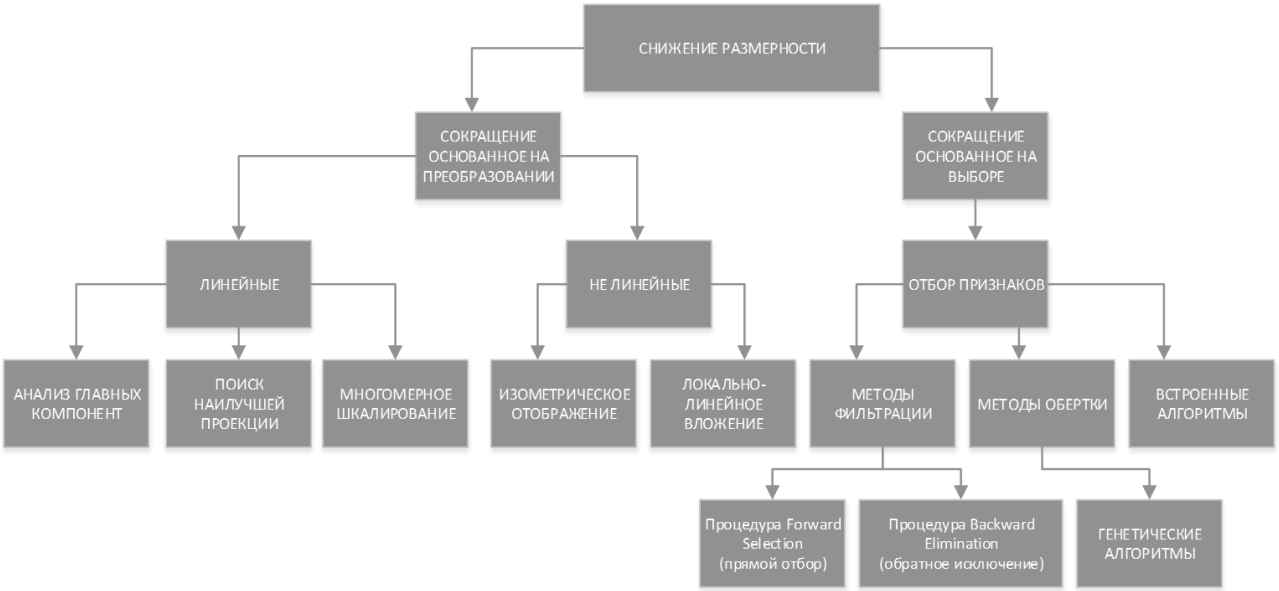


Рисунок 2.8 – Таксономия подходов сокращения измерения

В практических задачах затраты получение и обработку данных могут быть высокими, так что отбор признаков может существенно уменьшить эти затраты. Он также позволяет избежать проблемы переобучения, что улучшает точность модели. Поиск оптимального числа признаков может быть нетривиальной задачей, но существуют различные методы отбора признаков,

которые могут помочь выбрать самые важные и информативные параметры. Кроме того, отбор признаков может помочь упростить модель и сделать её более интерпретируемой. Таким образом, выбор оптимального набора признаков является важным шагом при решении задач машинного обучения и может существенно повлиять на качество модели.

Кроме того, одним из главных преимуществ методов отбора признаков является то, что они могут улучшить обобщающую способность модели. Это означает, что модель, построенная на основе отобранных признаков, будет лучше работать на новых, ранее не виденных её ранее данных. Это очень важно при работе с реальными данными, которые по своей природе всегда несколько шумные и содержат некоторую степень неопределенности.

С другой стороны, проводя отбор признаков, стоит помнить о том, что удаление признаков может привести к потере части данных и информации, которая может быть важна для решения задачи. Поэтому при выборе метода отбора признаков необходимо учитывать конкретные особенности задачи и данных, на которых она будет решаться.

В целом можно сказать, что методы отбора признаков являются важным инструментом в задачах машинного обучения. Они помогают уменьшить сложность моделей, улучшить обобщающую способность, упростить интерпретацию результатов и сократить затраты на получение и обработку данных. Поэтому при выборе подходящего метода отбора признаков стоит учитывать как их преимущества, так и возможные ограничения и недостатки каждого из них.

2.3.3.1 Feature Extraction

Подход Feature Extraction включает следующие методы: метод главных компонент, факторного анализа позволяют, не отбрасывая конкретные входы, учитывать лишь наиболее значимые комбинации их значений. Основными их целями являются:

- снижение размерности признакового пространства путем выявления статистически значимых линейных комбинаций нескольких признаков (анализ главных компонент),

- изучение взаимосвязи признаков путем выявления факторов, определяющих дисперсию признаков (факторный анализ).

Метод факторного анализа позволяет обработать большое количество переменных, связанных с имеющимися наблюдениями. В ходе анализа переменные, имеющие сильную корреляцию между собой, объединяются в составе факторов. В результате этого процесса получаются комплексные признаки, которые содержат гораздо больше информации, чем отдельные переменные. Целью факторного анализа является нахождение таких факторов, которые наиболее полно объясняют наблюдаемые связи между переменными.

Эта методика помогает свести к минимуму случайную составляющую и получить более достоверную информацию об исходных признаках и группах, которые исследуются. Также факторный анализ помогает выявить скрытые переменные и факторы, которые отвечают за наличие линейных связей между исследуемыми переменными.

Факторный анализ является важным инструментом в машинном обучении и статистическом анализе данных. Он может применяться в широком спектре областей, таких как социология, психология, маркетинг, экономика и многие другие.

Однако стоит иметь в виду, что факторный анализ не всегда приводит к уменьшению количества признаков. Иногда он может выделить большое количество факторов, каждый из которых будет содержать всего несколько признаков. Кроме того, в ходе исследований с помощью факторного анализа необходимо учитывать множество факторов, таких как размер выборки, надежность данных и т.д.

Рассмотрим модель факторного анализа [30]. Пусть задана система переменных $X_1, X_2, \ldots, X_n$. Значения переменных или признаков $X_1, X_2, \ldots, X_n$ известны для каждого из N предприятий (объектов). Представим исходную

информацию в виде матрицы $X = x_{ij}$ размерности $(n \times N)$. Каждая строка состоит из значений одного показателя для каждого из N объектов исследования. Предполагается, что каждый элемент этой матрицы x_{ij} является результатом воздействия некоторого числа m гипотетических общих факторов и одного характерного фактора

$$x_{ij} = a_{j1}f_{1i} + a_{j2}f_{2i} + \ldots + a_{jn}f_n + \ldots + a_{jm}f_{mi} + d_j v_{ji}$$

где a_{jr} – весовой коэффициент $j-$*й* переменной на $r-$*м* общем факторе или нагрузка $j-$*й* переменной на $r-$*м* общем факторе;

f_{ri} – значение $r-$го общего фактора на $i-$*м* объекте исследования;

d_j – нагрузка или весовой коэффициент $j-$*й* переменной на $j-$*м* характерном факторе;

v_{ji} – значение $j-$го характерного фактора на $i-$*м* объекте исследования;

$j = 1, \ldots, n;\ i = 1, \ldots, N;\ r = 1, \ldots, m;\ m \ll n.$.

Так как массив данных $X = x_{ij}$ представляет величины различной размерности, то для того, чтобы перейти к безразмерным величинам, пронормируем элементы массива.

$$y_{ji} = \left(x_{ji} - \bar{X}_j\right) / S_j$$

где $\bar{X}_j$– выборочное среднее $j-$*й* переменной (признака);

S_j – выборочная дисперсия $^{j-\text{й}}$ переменной. После этих преобразований получим:

$$y_{ij} = a_{j1}f_{1i} + a_{j2}f_{2i} + \ldots + a_{jn}f_n + \ldots + a_{jm}f_{mi} + d_j v_{ji}$$

где a_{jm} – неизвестные коэффициенты, называемые факторными нагрузками;

$d_j v_{ji}$ – называется остатком (невязкой), или остаточным специфическим фактором.

Задача состоит в том, чтобы оценить a_{jm} некоторым оптимальным образом.

Если в качестве критерия оптимальности используют минимум расхождения между ковариационной матрицей исходных признаков и той, которая получается после оценивания факторных нагрузок (мера «расхождения» двух матриц, в данном случае есть просто евклидова норма их разности), то приходят к методу главных компонент.

Метод анализа главных факторов используется для определения оптимальной близости между исходными корреляциями признаков и корреляциями, полученными в модели после оценки нагрузок. Он включает в себя метод главных компонент, который позволяет получить несколько новых показателей, являющихся линейными комбинациями исходных показателей, полученных в результате эксперимента.

Этот метод помогает сократить количество переменных, объединив их в несколько главных компонент, каждая из которых отражает значимую часть исходной информации. Главные компоненты также могут быть использованы для более эффективного анализа данных и выделения латентных факторов, которые могут быть связаны между собой.

Кроме того, метод главных компонент может использоваться для уменьшения шума в данные и устранения линейно зависимых переменных. Он также может помочь в определении наиболее значимых признаков, которые можно использовать в дальнейшем анализе.

Также следует учитывать, что использование метода главных компонент может привести к потере информации, поскольку признаки объединяются в несколько групп. Кроме того, метод главных компонент может быть

чувствителен к выбору модели и условиям эксперимента, таким как выборка, количество признаков и обработка выбросов.

Для более точного анализа и правильного выбора метода следует учитывать характеристики данных, цель исследования и другие факторы, которые могут повлиять на результаты анализа. Необходимо также тестировать различные методы и оценивать их эффективность для каждого конкретного набора данных и задачи [30].

Еще одним фактором, который следует учитывать при использовании метода главных компонент, является интерпретируемость полученных результатов. Хотя главные компоненты могут помочь сократить количество переменных и улучшить качество анализа данных, их интерпретация может быть сложной из-за линейной комбинации признаков. В этом случае может потребоваться дополнительный анализ для того, чтобы точнее определить, какие конкретно признаки влияют на полученные результаты.

В целом, использование метода главных компонент для снижения размерности данных имеет ряд преимуществ, таких как повышение эффективности анализа, уменьшение шума в данных и выделение значимых признаков. Однако, необходимо учитывать, что этот метод также имеет некоторые ограничения и не всегда является оптимальным решением для всех типов данных и задач. Для достижения наилучших результатов рекомендуется использовать несколько методов анализа данных и сравнить их эффективность и точность работы.

Метод главных компонент применяет преобразование координат в исходном пространстве признаков $x_1, \ldots, x_p$, которое приводит к созданию новой системы координат $y_1, \ldots, y_p$, содержащей ортонормированные линейные комбинации признаков.

$$\begin{cases} y_j(x) = w_{1j}(x_1 - m_1) + \ldots + w_{pj}(x_p - m_p); \\ \sum_{i=1}^{p} w_{ij}^2 = 1 \qquad (j = \overline{1,p}); \\ \sum_{i=1}^{p} w_{ij} w_{ik} = 0 \qquad (j,k = \overline{1,p}, j \neq k); \end{cases}$$

где m_i — математическое ожидание признака x_i .

Метод главных компонент выбирает линейные комбинации исходных признаков с наибольшей дисперсией, чтобы создать новую систему координат в пространстве признаков. Эта система координат ориентирована по направлениям наибольшей вытянутости эллипсоида рассеивания объектов исследуемой выборки. Первая главная компонента имеет наибольшую дисперсию, а последующие главные компоненты выбираются так, чтобы они были некоррелированы с предыдущими главными компонентами и имели наибольшую дисперсию. Вычисление коэффициентов главных компонент выполняется с использованием собственных векторов и собственных чисел корреляционной матрицы.

Метод главных компонент позволяет увидеть, какие признаки являются существенными и объясняют большую часть изменений, а какие не являются. Решение об остановке процедуры выделения факторов принимается на основании критерия Кайзера и критерия каменистой осыпи.

При использовании критерия Кайзера, если фактор не выделяет дисперсию, эквивалентную, по крайней мере, дисперсии одной переменной, то он опускается [32].

Критерий каменистой осыпи является графическим методом [32]. Собственные значения (факторную нагрузку) можно изобразить в виде простого графика (рисунок 2.9).

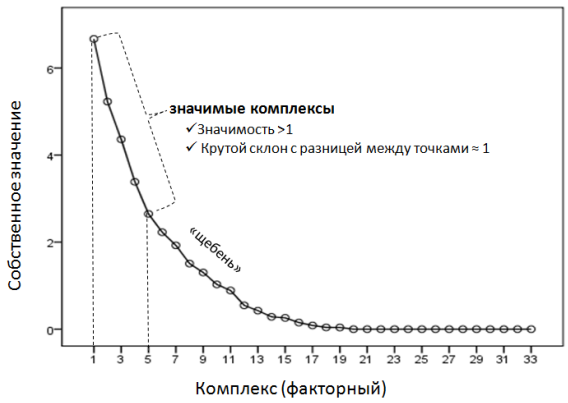


Рисунок 2.9 – Точечная диаграмма значимости факторных моделей

На графике компонентополагающее число определяется как место, где график убывания собственных значений имеет крутой склон с расстоянием между точками примерно равным единице и собственное значение больше единицы. Незначимые компоненты расположены на максимально замедленной части кривой, где расстояние между точками меньше единицы, а собственное значение компонента меньше единицы, образуя так называемый «щебень».

Критерий Кайзера и критерий каменистой осыпи являются распространенными методами выбора количества факторов, но они могут сохранять слишком много или слишком мало факторов. Важно также определить, сколько факторов могут быть содержательно интерпретированы на практике. Для этого исследуются решения с разным количеством факторов, после чего выбираются наиболее осмысленные и клинически значимые.

Поэтому при выборе наиболее «осмысленного» факторного комплекса необходимо учитывать не только критерии Кайзера и каменистой осыпи, но и результаты анализа в совокупности с конкретными целями исследования. Для этого можно оценить объясняющую способность выбранного факторного комплекса относительно всех исходных переменных, а также рассмотреть специфические характеристики выбранного факторного комплекса и его соответствие целям исследования.

Кроме того, не всегда имеет смысл использовать метод главных компонент в качестве инструмента статистического анализа. Этот метод является линейным и не позволяет учитывать возможные нелинейные

зависимости между переменными. Кроме того, он не всегда эффективен при работе с большим количеством переменных или в случаях, когда требуется сохранить набор специфических характеристик каждой переменной.

Таким образом, метод главных компонент является полезным инструментом, который может помочь уменьшить размерность пространства признаков и выделить наиболее важные компоненты, объясняющие наибольшую дисперсию. Однако его использование следует ограничивать специфическими свойствами каждого исследования и анализировать результаты метода в сочетании с другими методами статистического анализа.

Целью данного исследования является экспериментальная проверка возможности использования факторного анализа и метода главных компонент для отбора информативных признаков. Для проведения исследований использовался статистический пакет StatSoft STATISTICA 10.0.

Для исследования факторного анализа было взято 15 показателей (гемоглобин, эритроциты, цветовой показатель, лейкоциты, нейтрофилы палочкоядерные, нейтрофилы сегментоядерные, лимфоциты, моноциты, скорость оседания эритроцитов, общий белок, общий билирубин, амилаза, глюкоза, пол, длительность оперативных вмешательств). Обработано 109 пациентов и 109 пациентов принято для дальнейших вычислений.

На предварительном этапе исходные переменные, на основе которых будут отыскиваться факторы, стандартизуются.
Под стандартизацией понимается вычитание из каждого значения переменной ее среднего значения с последующим делением полученного результата на среднеквадратическое отклонение этой переменной. После стандартизации среднее значение каждой исходной переменной становится равным нулю, а дисперсия - единице. Общая дисперсия стандартизированных данных равна сумме дисперсий переменных, т. е. числу переменных. В нашем случае мы имеем 15 переменных, каждая из которых имеет дисперсию 1, то дисперсия данных равна 15. Это наибольшая изменчивость, которая потенциально может быть выделена с помощью метода главных компонент.

Для выделения факторов был использован метод главных компонент, который работает с первоначальной матрицей корреляции.

Приложение В таблица В.1. содержит информацию о том, какая часть дисперсии каждой из учитываемых в анализе переменных может быть объяснена предложенной факторной моделью. Например, мы видим, что показатель «Гемоглобин», объясняется данной факторной моделью на 92% (это максимальное значение), тогда как показатель «Длительность оперативных вмешательств» только на 7%. Эта таблица может быть полезна для того, чтобы исключить слишком «непредсказуемые» переменные из анализа, тем самым повысив общую прогностическую успешность факторной модели. В нашем случае это «Моноциты», «Скорость оседания эритроцитов», «Амилаза», «Глюкоза», «Длительность оперативных вмешательств» и «Пол», так как эти переменные объясняются менее чем на 50%.

Каждый последующий фактор объясняет все меньше дисперсии, как видно из приложения В таблицы В.2, где приведены результаты, включая номера и объясняемые дисперсии для каждого фактора. Первый фактор объясняет 21,97% общей дисперсии, второй - 15,52%, а остальные факторы содержат не более 9% общей дисперсии. Таблица также содержит собственные значения только для шести факторов, оставленных в соответствии с выбранным критерием Кайзера, которые вместе объясняют 37,49% дисперсии исходных данных.

Это говорит о том, что использование только первых нескольких факторов может быть достаточно для объяснения значительной части дисперсии. Однако, при выборе наиболее подходящего факторного комплекса необходимо учитывать не только долю объясненной дисперсии, но и другие характеристики, такие как специфичность каждого фактора и его соответствие целям исследования.

Например, если исследование направлено на изучение определенного аспекта данных, может потребоваться использование факторов, которые не

имеют высокой доли объясненной дисперсии, но являются более специфичными для этого аспекта.

Таким образом, метод главных компонент представляет собой мощный инструмент для уменьшения размерности пространства признаков и выделения наиболее важных компонент, но не должен использоваться сам по себе, а рассматриваться в сочетании с другими методами статистического анализа и учитывать специфику конкретного исследования.

Таблица В.3 представляет собой матрицу факторных нагрузок, из которой видно, с каким из факторов наиболее сильно коррелирует та или иная переменная. Матрица компонентов в первом наиболее значимом компоненте с факторной нагрузкой 3,296 показывает сильную прямую ассоциацию с тремя показателями (гемоглобин – векторная нагрузка переменной 0,896, эритроциты – 0,845, цветовой показатель – 0,736). Второй компонент сильно ассоциирует с показателем (общий билирубин – 0,703, общий белок – 0,668, амилаза – 0,633, глюкоза – 0,539). Остальные менее значимые компоненты. Такую модель сложно интерпретировать.

График каменной осыпи (рисунок 2.10) также позволяет определить количество эффективных факторов в модели.

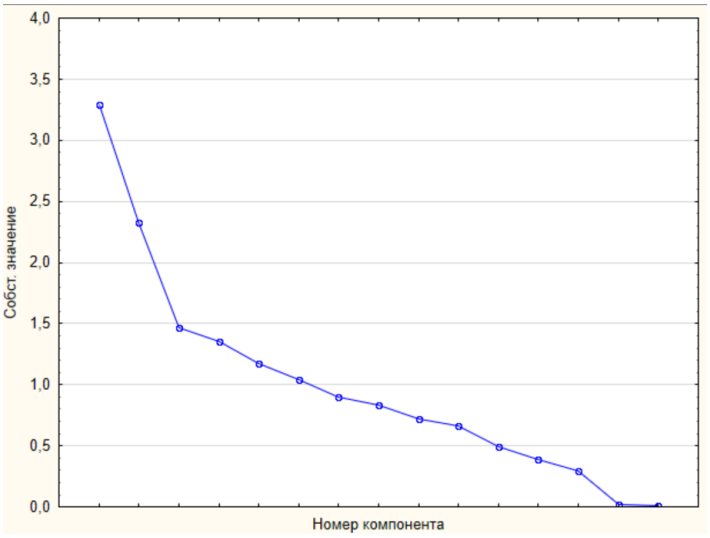


Рисунок 2.10 – График каменистой осыпи (факторный анализ)

График каменной осыпи (рисунок 2.10) также позволяет определить количество эффективных факторов в модели. При движении вправо на этом

графике собственные значения поначалу резко падают. Когда это падение прекращается, и кривая делает изгиб в сторону менее резкого снижения, тест осыпи Кеттелла говорит отбросить все дальнейшие компоненты после того, с которого начинается изгиб. В исследовании критерий Кайзера приводит к 2 компонентам, однако критерий осыпи позволяет ограничиться 3 компонентами. Это видно и из таблицы В.2, но на графике это представлено более наглядно.

Учитывая то, что на основе таблицы полной объясненной дисперсии (таблица В.2) и особенно графика каменистой осыпи (рисунок 2.9) было найдено факторное решение, теперь на первый план выходит проблема интерпретируемости. Следующим шагом стало вращение, которое помогает представить результаты в более удобном виде и обычно облегчает интерпретацию факторов.

Из приложения В таблицы В.4 видно, что в первом наиболее значимом компоненте с факторной нагрузкой 3,296 показывает сильную прямую ассоциацию с четырьмя показателями (гемоглобин – векторная нагрузка переменной 0,941, эритроциты – 0,859, цветовой показатель – 0,806, общий белок –0,544). Второй компонент сильно ассоциирует с показателем (общий билирубин – 0,521, лимфоциты – 0,863, лейкоциты – 0,821, нейтрофилы палочкоядерные – 0,711).

Таким образом, для достижения наилучшей корреляции между факторами и исходными переменными, оси необходимо повернуть так, чтобы каждый фактор максимально коррелировал с одними переменными и минимально коррелировал с другими переменными. В данном исследовании был выбран метод варимакс вращения, который позволяет упростить интерпретацию факторов, минимизируя число переменных с высокими нагрузками на каждый фактор. Это позволяет сократить количество переменных, с которыми нужно работать, и сделать результаты исследования более доступными и понятными.

Также стоит упомянуть, что выбор метода вращения зависит от конкретного исследования и целей, которые ставятся перед исследователем. Например, если целью является выделение компонент, которые лучше всего описывают данные, то может быть использован метод главных факторов. А если целью является выделение компонент, которые наилучшим образом дают интерпретацию структуры данных, то может быть использован метод максимальной вероятности.

В любом случае, выбор наиболее подходящего метода вращения осей относится к важным этапам факторного анализа. Он напрямую влияет на качество результатов и их интерпретируемость. При выборе метода необходимо учитывать не только цель исследования, но и специфику данных, размер выборки, а также другие факторы, которые могут повлиять на выбор.

Таким образом, на основе выяснения степени соответствия всех показателей были определены следующие два фактора:

1) фактор 1 (Ф-1),

2) фактор 2 (Ф-2).

Проанализировав результаты расчета факторного анализа, можно выделить следующие факторы для включения их в выборку для системы прогнозирования: гемоглобин, эритроциты, цветовой показатель, общий белок, общий билирубин, лимфоциты, лейкоциты, нейтрофилы палочкоядерные.

Также были проведены исследования метода главных компонент для отбора информативных признаков. Для проведения исследований использовался статистический пакет пакета StatSoft STATISTICA 10.0.

Для исследования метода главных компонент было взято 15 показателей. Обработано 109 пациентов и 109 пациентов принято для дальнейших вычислений. График каменистой осыпи представлен на рисунке 2.11. Данный график идентичен графику, построенному по методу факторного анализа. Данный метод показал те же результаты что и факторный анализ.

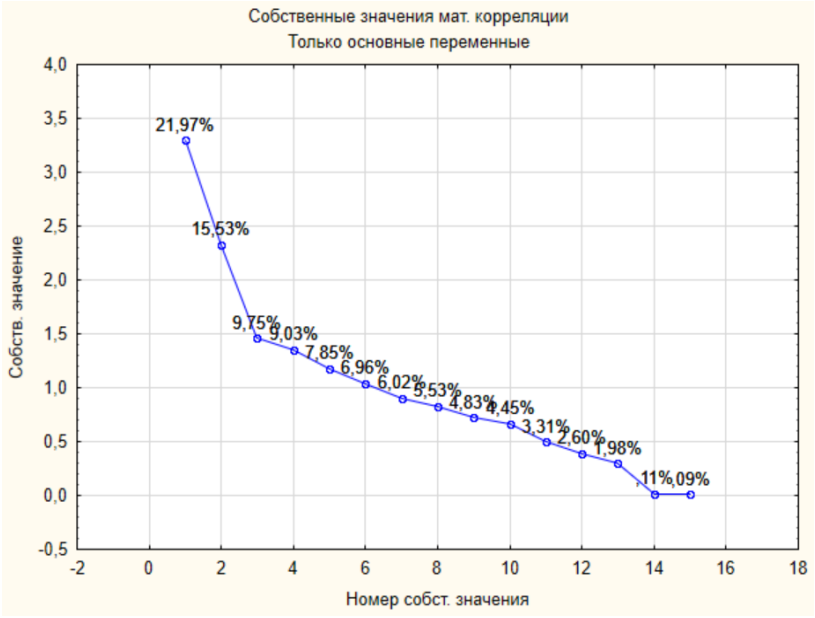


Рисунок 2.11 – График каменистой осыпи (метод главных компонент)

Таблицы, представленные в Приложении В для факторного анализа соответствуют и методу главных компонент, в связи с этим дальнейшие исследования метода главных компонент нецелесообразны.

2.3.3.2 Feature Selection

Основная цель метода отбора переменных (Feature Selection) состоит в том, чтобы определить минимальное подмножество признаков от проблемной области, сохраняя соответственно высокую точность в представлении оригинала. Чаще всего используемые алгоритмы отбора информативных признаков (Feature Selection), можно условно разделить на два класса — «обертки» (wrappers) и «фильтры» (filters). Существуют три основных класса алгоритмов отбора признаков (Feature Selection) – фильтры (filters), обертки (wrappers) и встроенные алгоритмы (embedded algorithms) [33].

Метод отбора переменных (Feature Selection) является одним из наиболее используемых методов в анализе данных и машинном обучении. Он относится к методам уменьшения размерности данных и заключается в том, чтобы выбрать определенный набор переменных из полного набора данных, которые будут использоваться для анализа или обучения модели.

Существуют различные подходы к методу отбора переменных в зависимости от поставленных задач и данных. Одни из наиболее распространенных методов включают следующие:

- отбор на основе корреляции: данный метод заключается в том, чтобы выбрать набор переменных, которые имеют наиболее сильную корреляцию с искомой переменной (например, зависимой переменной в задаче регрессии),

- отбор на основе статистических тестов: данный метод использует статистические тесты, такие как T-тест или ANOVA, для оценки важности переменных в задаче классификации или регрессии,

- рекурсивный отбор, основанный на модели. Данный метод использует отдельную модель, например, случайный лес, решающее дерево или логистическую регрессию, чтобы оценить важность каждой переменной и последовательно удалять наименее важные переменные до тех пор, пока не достигнут заданный размер набора переменных.

Кроме того, также используются комбинированные методы, которые используют два или более из упомянутых методов одновременно.

Выбор наиболее подходящего метода отбора переменных зависит от конкретной задачи, типа данных и количества переменных, а также от желаемой простоты и интерпретируемости статистических моделей. Некоторые методы могут быть более эффективны в одних случаях, а другие - в других. Однако, методы отбора переменных играют важную роль в сокращении размерности данных и повышении качества модели, особенно в условиях большого количества переменных и ограниченных ресурсов вычислительной мощности.

Кроме того, метод отбора переменных может помочь избежать проблему переобучения (overfitting) при построении моделей. Если в модель попадут переменные, влияние которых не существенно для исходного датасета, то модель может «переобучиться», то есть стать слишком сложной и необобщающейся.

С другой стороны, некоторые методы отбора переменных могут приводить к потере информации и важных переменных, поэтому необходимо балансировать между оставлением достаточного количества переменных, чтобы модель была достаточно информативной, и уменьшением размерности данных, чтобы снизить сложность модели.

Метод отбора переменных также может помочь улучшить интерпретируемость модели и повысить ее практическую применимость. Если модель использует только наиболее важные переменные в задаче, то результаты модели становятся более понятными и интерпретируемыми, что может упростить принятие решений.

В целом, метод отбора переменных играет важную роль в анализе данных и машинном обучении, помогая повысить качество моделей и улучшить их интерпретируемость. Однако, при выборе метода отбора переменных следует учитывать конкретные задачи, тип данных и количество переменных, чтобы достичь наилучшего результата.

Forward Selection и Backward Elimination — это два из классических методов отбора переменных, которые применяются для выбора оптимального набора переменных.

Метод Forward Selection начинается с выбора одной переменной, которая наилучшим образом объясняет зависимую переменную. Затем новые переменные добавляются по одной за раз на основе того, как много они могут внести в объяснение зависимой переменной (например, большой коэффициент корреляции или малый p-value). Процедура продолжается до тех пор, пока добавление дополнительных переменных не приводит к значимому улучшению качества модели.

Напротив, метод Backward Elimination начинается с модели, содержащей все переменные, и затем последовательно удаляет одну переменную за раз, начиная с той, которая наименее важна для модели. Процедура продолжается до тех пор, пока удаление дополнительных переменных не приводит к значимому ухудшению качества модели.

Оба метода имеют свои преимущества и недостатки. Преимущества Forward Selection включают то, что он обычно быстрее, чем Backward Elimination, потому что он добавляет переменные по очереди. De преимуществ Backward Elimination — это то, что он рассматривает все переменные в начале и отбрасывает менее важные от начала до конца, что может привести к выбору наилучшей модели быстрее. Кроме того, Backward Elimination склонен к избавлению от шума, который, может быть, присутствует в начальной модели.

Однако оба метода могут привести к недооценке модели, если не берутся во внимание важные переменные, которые не были включены или отброшены, чтобы удовлетворить условиям метода.

В целом, выбор между методом Forward Selection и Backward Elimination зависит от конкретной задачи, типа данных и количества переменных в датасете. В некоторых случаях лучший выбор, может быть, комбинирование этих методов, чтобы получить оптимальную модель.

Целью данного исследования является экспериментальная проверка возможности использования метода Forward Selection и метода Backward Elimination для отбора информативных признаков. Было взято 15 показателей (гемоглобин, эритроциты, цветовой показатель, лейкоциты, нейтрофилы палочкоядерные, нейтрофилы сегментоядерные, лимфоциты, моноциты, скорость оседания эритроцитов, общий белок, общий билирубин, амилаза, глюкоза, пол, длительность оперативных вмешательств). Обработано 109 пациентов и 109 пациентов принято для дальнейших вычислений.

Для проведения исследования использовалась функция последовательного выбора признаков, на ее основе разработан сценарий (fsa.m), использующий функцию sequentialfs пакета Statistics and Machine Learning Toolbox системы Matlab. Обращение к функции имеет следующий вид:

`[inmodel,history]=sequentialfs(fun,X,y,`*`param1`*`,`*`val1`*`,...).`

Функция fun является функцией дескриптором, которая определяет критерий, используемый для выбора показателей и определения точки

останова. Входными данными для функции sequentialfs являются матрица независимых переменных X размерностью 15*109 и вектор зависимой переменной y размерностью 1*109. Строка матрицы X соответствует наблюдениям; столбцы матрицы соответствуют показателям. Столбец вектора y соответствует целевому классу для каждого наблюдения в матрице X. Начиная с пустого набора показателей, sequentialfs создает кандидата в подмножество путем последовательного добавления каждого из показателей, которые пока не были выбраны. Для каждого кандидата в подмножество, sequentialfs выполняет кросс-проверку, неоднократно вызывая функцию fun с различными обучающими и тестовыми подмножествами x и y следующим образом:

```
criterion = fun(XTRAIN,ytrain,XTEST,ytest).
```

В функцию fun передавались обучающие и тестовые выборки, полученные случайным выбором из входных данных, 50% для обучающей и 50% для тестовой выборки. Каждый раз при вызове этой функции возвращается скалярное значение критерия. Функция fun использует XTRAIN и ytrain для обучения модели, затем предсказывает значения XTEST используя эту модель, и возвращает некоторую меру расстояния, или потери, тех прогнозируемых значений от ytest. В перекрестной проверке для заданного набора кандидатов sequentialfs суммирует значения, возвращаемые fun и делит эту сумму на общее число тестовых наблюдений. Затем он использует среднее значение для оценки каждого кандидата в подмножество.

После вычисления средних значений критерия для каждого кандидата в подмножество sequentialfs выбирает кандидата в подмножество, который минимизирует среднее значение критерия. Этот процесс продолжается до тех пор, пока добавление новых кандидатов не минимизирует критерий. В данной работе минимизируется сумма ошибок классификации. Классификация выполнялась с использованием функции дискриминантного анализа:

```
class = classify(sample,training,group).
```

Данная функция классифицирует каждую строку данных в выборочной матрице sample на одну из групп, представленных в обучающей матрице training. sample и training должны быть матрицами с одинаковым числом столбцов. group это группирующая переменная для матрицы training. Ее однозначно определяемые значения определяют группы, каждый элемент определяет ту группу, к которой принадлежит соответствующая строка матрицы training. group может быть категориальной переменной (переменной, определенной на шкале наименований), числовым вектором, строковым массивом или строковой матрицей. Матрицы training и group должны иметь одинаковое число строк. Выход class указывает группу, к которой приписывается каждая строка матрицы sample и имеет тот же тип, что и group.

Эксперименты с методом Backward Elimination показали, что данный метод отобрал только один показать: Общий билирубин (рисунок 2.12).

```
Initial columns included:  none
Columns that can not be included:  none
Step 1, added column 10, criterion value 0.311927
Final columns included:  10

fs =

     0     0     0     0     0     0     0     0     0     1     0     0     0     0     0


history =

      In: [0 0 0 0 0 0 0 0 0 1 0 0 0 0 0]
    Crit: 0.3119
```

Рисунок 2.12 – Результат работы алгоритма обратного исключения переменных

На рисунке показаны шаги уменьшения значения критерия, при котором новая переменная будет добавлена в модель. Конечный результат представляет собой уменьшенную модель только с одной из пятнадцати переменных: столбец 10. Эти переменные указаны в логическом векторе fs, возвращаемом sequentialfs. Это показатель – Общий билирубин.

Эксперименты с методом Forward Selection показали, что данный метод отобрал восемь показателей: Лейкоциты, Нейтрофилы палочкоядерные, Нейтрофилы сегментоядерные, Лимфоциты, Общий билирубин, Амилаза, Длительность оперативных вмешательств (рисунок 2.13).

```
Initial columns included:  all
Columns that must be included:  none
Step 1, used initial columns, criterion value 0.33945
Step 2, removed column 3, criterion value 0.321101
Step 3, removed column 11, criterion value 0.293578
Step 4, removed column 2, criterion value 0.293578
Step 5, removed column 8, criterion value 0.293578
Step 6, removed column 15, criterion value 0.284404
Step 7, removed column 13, criterion value 0.284404
Step 8, removed column 1, criterion value 0.284404
Step 9, removed column 9, criterion value 0.284404
Step 10, removed column 12, criterion value 0.266055
Final columns included:  4 5 6 7 10 14
fs =

     0     0     0     1     1     1     1     0     0     1     0     0     0     1     0


history =

      In: [9x15 logical]
    Crit: [0.3394 0.3211 0.2936 0.2936 0.2936 0.2844 0.2844 0.2844 0.2844 0.2660]
```

Рисунок 2.13 – Результат работы алгоритма прямого отбора переменных

На следующем этапе исследования проводился регрессионный анализ для дополнительного уточнения выводов алгоритмов относительно выбора наиболее значимых независимых переменных.

Сложные системы не могут быть ограничены единственной моделью на основе экспериментальных данных. Для каждого объекта, рассматриваемого как черный ящик, существует бесконечное количество уравнений с одинаковыми или почти одинаковыми внешними проявлениями. Однако в логике научных исследований требуется выбрать одну или несколько математических моделей с оптимальной структурой. Для решения этой задачи генерируются наборы уравнений различной сложности, а затем отбираются лучшие по заданному критерию.

В данной работе была использована математическая модель множественной линейной регрессии, которая представляется в виде:

$$y_k = \theta_{1k}x_{1ki} + \theta_{jk}x_{jk} + \ldots + \theta_{ml}x_{ml} + \varepsilon_k, i = \overline{1,n}, j = \overline{1,m};\ k = \overline{1,l}$$

где y_k – значение зависимой переменной в k -м слое наблюдения;

x_{ml} – известное значение независимой переменной в k-м слое m-строке наблюдения;

θ_{jk} – неизвестный коэффициент при i-ой объясняющей переменной;

ε_k – случайная составляющая.

Цель множественной линейной регрессии - оценить влияние каждой независимой переменной на зависимую переменную при условии влияния всех других независимых переменных.

Для нахождения оптимального набора переменных был использован регрессионный анализ, который позволил определить значимость каждой независимой переменной и их влияние на зависимую переменную. Этот анализ позволил утверждать, какие переменные следует включить в модель, а какие исключить. Он также способствовал более точному уточнению выводов относительно выбора наиболее значимых независимых переменных и созданию более эффективной модели.

Для расчетов по регрессионной модели был выбран пакет Statistics and Machine Learning Toolbox системы Matlab. При использовании этого пакета была применена функция шаговой регрессии:

```
stepwise(X,y,inmodel,alpha).
```

Функция позволяет получить в интерактивном режиме регрессионную модель для зависимой переменной y от независимых переменных – столбцов матрицы X. Зависимая переменная y задается как вектор. Число элементов вектора y должно быть равно количеству строк в матрице X. Входной аргумент inmodel позволяет управлять начальным множеством факторов, включенным в регрессионную модель. Элементы вектора inmodel являются номерами факторов, т.е. номерами столбцов матрицы X. Входной аргумент alpha задает уровень значимости для расчета границ доверительного интервала коэффициентов регрессионной модели. alpha служит для проверки гипотезы о статистической значимости каждого фактора в регрессионной модели.

В качестве независимых переменных для анализа были использованы переменные, отобранные при помощи алгоритмов прямого отбора и обратного исключения. В качестве зависимой переменной использовался показатель наличия или отсутствия послеоперационных осложнений (рисунки 2.14, 2.15).

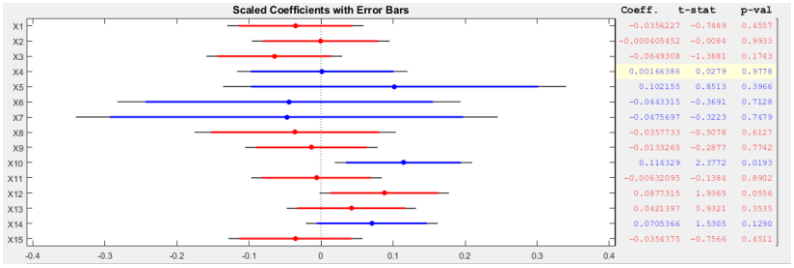


Рисунок 2.14 – Результат работы шаговой регрессии на данных, полученных от алгоритма прямого отбора

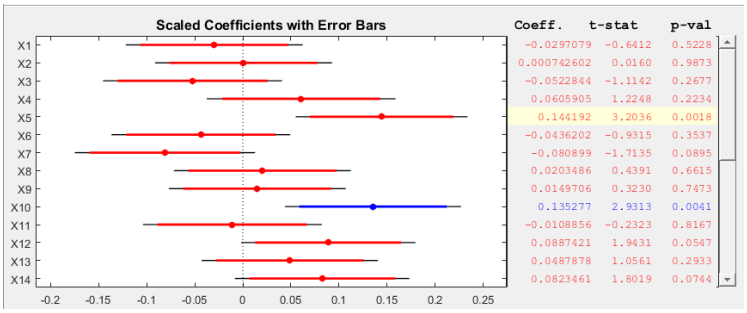


Рисунок 2.15 – Результат работы шаговой регрессии на данных, полученных от алгоритма обратного исключения переменных

В ходе экспериментов моделью регрессии на данных, полученных от алгоритма прямого отбора, было распознано на обучающей выборке 54,02%, на тестовой – 45,45%. Моделью множественной линейной регрессии на данных, полученных от алгоритма обратного исключения, было распознано на обучающей выборке 18,39%, на тестовой – 04,55 %. Также для каждой модели был рассчитан коэффициент детерминации (R^2). R2 является показателем соответствия модели и данных (чем ближе R2 к 1, тем лучше модель соответствует данным). Модель множественной линейной регрессии на данных, отобранных при помощи алгоритма прямого отбора, имеет R2 равный 0,69, что означает, что около 69% изменчивости значений переменной может быть объяснено моделью, а 31% остаются необъясненными. Это означает, что модель объясняет значительную часть исходной изменчивости, но все еще остается значительный объем изменчивости, который не может быть объяснен моделью.

Это может указывать на то, что некоторые другие факторы, не учтенные в модели, также могут влиять на зависимую переменную. При этом стоит отметить, что значение R2 всегда должно рассматриваться в сочетании с

другими параметрами модели, такими как значимость коэффициентов и нормальность распределения остатков, что поможет более полно оценить адекватность модели.

Моделью множественной линейной регрессии на данных, полученных от алгоритма обратного исключения, имеет R^2 равный 0,10, таким образом только 10% исходной изменчивости может быть объяснено данной моделью. Эксперименты показали, что метод прямого отбора лучше отслеживает взаимосвязи между переменными и его целесообразнее использовать далее.

2.3.4 Сравнение статистических методов выбора анализов

На рисунке 2.16 приведены результаты статистического исследования обучающей выборки различными методами, для того чтобы выбрать наиболее информативный набор входных анализов. В данной таблице знаком «+» обозначена информативность анализа в каждом методе.

№	Название	Критерий Вилкоксона-Манна-Уитни	Корреляционный анализ	Факторный анализ	Алгоритм прямого отбора переменных
1.	Гемоглобин			+	
2.	Эритроциты			+	
3.	Цветовой показатель			+	
4.	Лейкоциты			+	
5.	Нейтрофилы палочкоядерные	+	+	+	+
6.	Нейтрофилы сегментоядерные				+
7.	Лимфоциты	+		+	+
8.	Моноциты				
9.	Скорость оседания эритроцитов				
10.	Общий билирубин	+	+	+	+
11.	Общий белок			+	
12.	Амилаза	+	+	+	+
13.	Глюкоза				
14.	Длительность оперативных вмешательств	+	+		+
15.	Пол				

Рисунок 2.16 – Результат исследования данных

Для дальнейшей диагностики выбираются те анализы, информативность которых подтвердилась большинством методов. По результатам анализа рисунка 2.16 можно сделать вывод, что наиболее информативными признаками являются: нейтрофилы палочкоядерные, лимфоциты, общий билирубин, длительность оперативных вмешательств, амилаза. Это позволило

сократить число анализ с 15 до 5. Это позволило, из множества разнообразных показателей выбрать их минимальное число, достаточное для прогнозирования послеоперационных осложнений при мочекаменной болезни.

2.4 Классификация и прогноз послеоперационных осложнений при мочекаменной болезни

Проведя анализ данных и выявив зависимости между переменными, можно проанализировать соотношение групп переменных и попытаться прогнозировать значения одних переменных в зависимости от значений других. Для этого необходимо выполнить следующие задачи: описание данных, установление сходства и отличий между ними, количественный и качественный анализ зависимостей между переменными и выделение наиболее существенных переменных. В результате этого анализа можно будет прогнозировать значения одних переменных, используя значения других переменных.

Такой анализ может быть полезен для многих задач, например, для прогнозирования заболеваний. Важно понимать, что успешное прогнозирование значений переменных зависит не только от выбранных переменных, но и от правильной настройки модели, выбора методов анализа и стратегии обработки данных. Правильно проведенный анализ позволит получить точный и полезный прогноз, который может быть использован для принятия врачебных решений или других областях деятельности.

При использовании многомерных методов анализа для прогнозирования в медицине возникают серьезные ограничения, связанные с принципами линеаризации, на которых основаны эти методы. В частности, такие методы требуют большой выборки исследуемых объектов для каждого анализируемого признака или переменной (около 10 объектов на каждую переменную). Например, если анализируется 10 биохимических показателей крови, то для каждого из них необходимо иметь около 100 пациентов для достаточной точности анализа. Это требование применимо не только к

линейным методам аппроксимации, таким как линейная регрессия, но и к методам дискриминантного и факторного анализа. Важно учитывать эти ограничения при использовании многомерных методов анализа для решения задач в медицине.

Более того, при использовании многомерных методов анализа в медицине необходимо учитывать также и другие факторы, которые могут существенно влиять на точность прогнозирования. Например, возможность появления случайных факторов, таких как ошибки измерения, может привести к получению неточных результатов. А также необходимо учитывать качество данных, их достоверность, степень нормализации и скоррелированности с другими переменными. Важно также учитывать специфику медицинских исследований, таких как большое варьирование геномных данных, необходимость использования статистических методов для тестирования гипотез и многие другие аспекты, которые должны быть приняты во внимание при использовании многомерных методов анализа в медицине.

Кроме того, при использовании многомерных методов анализа в медицине необходимо учитывать также и контекст, в котором проводятся исследования, так как разные области медицины могут требовать разных методов анализа. Например, при анализе медицинских данных, связанных с заболеваниями сердечно-сосудистой системы, могут использоваться методы, такие как регрессионный анализ, факторный анализ и другие, которые помогают выявить взаимосвязи между различными факторами риска и развитием заболевания. В то же время, при анализе данных, связанных с онкологическими заболеваниями, могут применяться более сложные методы, такие как дискриминантный анализ и анализ выживаемости, которые позволяют оценить вероятность развития заболевания и выживаемости пациентов.

Таким образом, при использовании многомерных методов анализа в медицине необходимо учитывать все возможные ограничения и факторы,

которые могут влиять на достоверность результатов и принимать во внимание контекст и специфику конкретного медицинского исследования.

2.4.1 Прогнозирование с помощью метода к-ближайших соседей

Метод k-ближайших соседей (k Nearest Neighbors, или kNN) – популярный алгоритм классификации, который используется в разных типах задач машинного обучения. Наравне с деревом решений это один из самых понятных подходов к классификации.

Допустим, мы сталкиваемся с задачей классификации, где имеется m наблюдений, каждому из которых соответствует запись в таблице, и необходимо определить класс для новой записи. Для решения этой задачи на первом шаге алгоритма следует выбрать оптимальное число k – количество ближайших соседей. Если k = 1, то результат может быть неточным, так как класс будет определяться только по самому близкому объекту. С другой стороны, при выборе слишком большого значения k, могут быть упущены многие локальные особенности.

Далее, на втором шаге, необходимо найти k записей с наименьшим расстоянием до вектора признаков нового объекта (поиск соседей). Однако, для расчета расстояния необходимо выбрать правильную функцию, удовлетворяющую определенным правилам:

$d(x, y) \geq 0, d(x, y) = 0$ тогда и только тогда, когда $x = y$

$d(x, y) = d(y, x)$

$d(x, z) \leq d(x, y) + d(y, z)$

где x, y, z – векторы признаков сравниваемых объектов.

Для упорядоченных значений атрибутов находится Евклидово расстояние:

$$D_E \sqrt{\sum_{i}^{n} (x_i - y_i)}$$

Евклидова метрика (евклидово расстояние, или же Euclidean distance) – метрика в евклидовом пространстве, расстояние между двумя точками евклидова пространства, вычисляемое по теореме Пифагора. Проще говоря, это наименьшее возможное расстояние между точками A и B. Хотя евклидово расстояние полезно для малых измерений, оно не работает для больших измерений и для категориальных переменных. Недостатком евклидова расстояния является то, что оно игнорирует сходство между атрибутами. Каждый из них рассматривается как полностью отличный от всех остальных.

Данный алгоритм реализован в Google Colab. Прогнозирование проводилось на анализах, полученных ранее с помощью статистических методов. Классификатор к-ближайших соседей проводится для зависимой переменной Оценка (наличие осложнений – 1, отсутствие осложнений – 0) и 5 предикторов (Амилаза, Нейтрофилы палочкоядерные, Лимфоциты, Общий билирубин, Длительность оперативных вмешательств).

На основе вышеописанного алгоритма был разработан сценарий, реализующий выполнение логистической регрессии. Результаты эксперимента представлены на рисунке 2.17.

```
The accuracy score for method is: 57.599999999999994%
The f1 score for method is: 30.0%
The precision score for method is: 100.0%
The recall score for method is: 17.599999999999998%
```

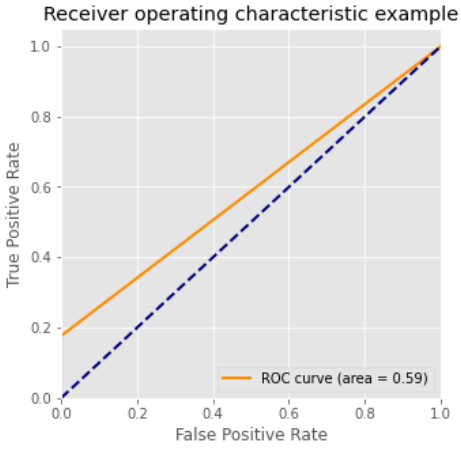


Рисунок 2.17 – Результат работы к-ближайших соседей

Данный алгоритм показал точность диагностики равную 57,5%, и площадь под ROC-кривой 0,59. Данный результат является неудовлетворительным для диагностики в области медицины. В данный процент ошибки входит лишь ошибка второго рода, это когда классификатор показал отсутствие осложнений, хотя на самом деле они есть. Это видно из матрицы неточностей (confusion matrix) на рисунке 2.18.

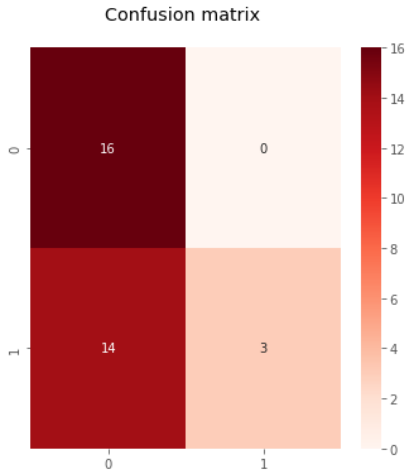


Рисунок 2.18 – Визуализатор матрица неточности

В нашей работе нет больных с отсутствием послеоперационных осложнений который метод определил как больных с осложнениями (гипердиагностика), 14 больных с наличием послеоперационных осложнений метод определил как с отсутствием осложнений (гиподиагностика).

При анализе медицинских данных важно не только учитывать процент ошибок, но также и процент гипердиагностики и гиподиагностики сети, что особенно важно для медицинских работников. Для этого рассчитывается количество ошибок первого и второго рода. Ошибка первого рода, или гипердиагностика, может произойти, если анализ крови покажет наличие заболевания, хотя человек на самом деле здоров. Ошибка второго рода, или гиподиагностика, может произойти, если анализ крови покажет отсутствие заболевания, хотя человек на самом деле болен.

2.4.2 Прогнозирование с помощью дерева решений

Метод деревьев решений представляет собой эффективный инструмент обработки медико-биологических данных, что было подтверждено

применением данного метода для прогнозирования послеоперационных осложнений у пациентов с мочно-каменной болезнью.

Данный метод позволяет создать древовидную модель, которая может быть использована для классификации пациентов с высокой точностью. Благодаря этому медицинские работники могут более точно определять риски послеоперационных осложнений у пациентов с мочно-каменной болезнью и принимать более обоснованные решения относительно дальнейшего лечения. Кроме того, метод деревьев решений может быть применен для обработки и анализа других медико-биологических данных, что позволяет улучшить качество диагностики и повысить эффективность лечения различных заболеваний.

Основная идея метода заключается в том, чтобы построить древовидную структуру решений на основе набора данных, чтобы предсказывать целевую переменную. Каждый узел дерева представляет собой тест на значение одного из признаков набора данных, а каждая ветвь дерева соответствует возможному результату теста. Каждый конечный узел дерева, называемый листом, представляет собой прогнозируемое значение целевой переменной (рисунок 2.19).

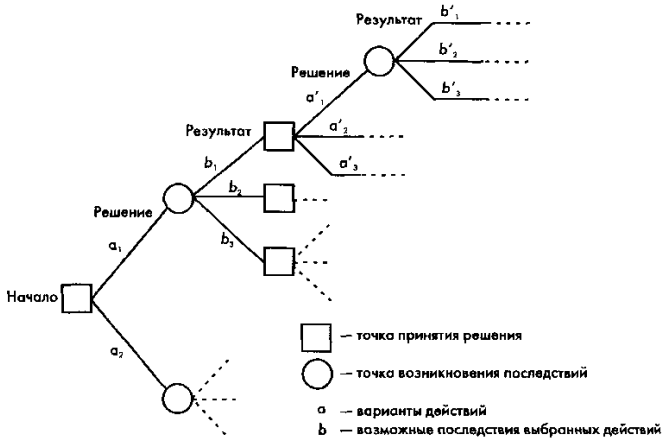


Рисунок 2.19 – Структура дерева решений

Построение дерева решений начинается с корня, который представляет собой все доступные данные. Затем алгоритм выбирает признак, который наилучшим образом разделяет данные на две группы по значению этого признака. Процесс разделения продолжается до тех пор, пока все листья

дерева не будут содержать только данные из одного класса, или пока не будет достигнут заданный критерий остановки.

Дерево решений может использоваться как для классификации, так и для регрессии. В случае классификации каждый лист дерева определяет класс, к которому относится соответствующее наблюдение. В случае регрессии лист дерева определяет значение целевой переменной. Метод деревьев решений имеет множество преимуществ, в том числе простоту и удобство интерпретации полученных результатов, а также возможность обрабатывать наборы данных со множеством признаков.

Одним из основных преимуществ метода деревьев решений является его простота и удобство использования. Дерево решений можно использовать для моделирования большого числа переменных без необходимости знать математические формулы или сложные алгоритмы. Кроме того, дерево решений очень быстро обучается и способно работать с большими объемами данных.

Еще одним преимуществом деревьев решений является возможность их интерпретации. Дерево решений представляет собой графическую модель, которую легко можно понять и объяснить другим. Дерево решений дает возможность увидеть, какие признаки наиболее важны для принятия решения, что помогает лучше понимать процесс принятия решений.

Также метод деревьев решений имеет высокую точность прогнозирования и хорошую устойчивость к выбросам и шумам в данных. Однако, недостатком метода деревьев решений является возможность переобучения модели. Дерево может стать слишком сложным и чувствительным к шумам в данных, что снижает точность предсказания.

В целом, метод деревьев решений является эффективным инструментом для решения задач классификации и регрессии в различных областях, в том числе в медицинских приложениях.

Для иллюстрации возможностей выявления скрытых иерархических закономерностей применительно к данным медико-биологической природы

было исследовано влияние переменных, характеризующих состояние больного при мочно-каменной болезни, на послеоперационные осложнения.

Для построения деревьев классификации, предназначенных для прогнозирования послеоперационных осложнений, были использованы данные, полученные в результате обследования и лечения пациентов в отделениях торакальной хирургии областной клинической больницы имени Н.Н. Бурденко. Параметры, определяющие текущее состояние пациента, включали такие факторы, как количество лейкоцитов, палочкоядерных нейтрофилов, лимфоцитов, общий уровень билирубина и длительность оперативного вмешательства.

Для построения деревьев классификации была использована обучающая выборка, включающая 109 наблюдений. Минимальное количество примеров в узле, необходимое для создания нового узла, было выбрано равным 2, что составляет примерно 1% от общего объема обучающей выборки. Это позволило избежать появления недостоверных правил при слишком малом количестве примеров и почти полного отсутствия правил при слишком большом количестве.

Для оценки качества построенных деревьев классификации была использована тестовая выборка, составляющая 45% от исходной выборки. В ходе построения деревьев были использованы инструменты Google Colab.

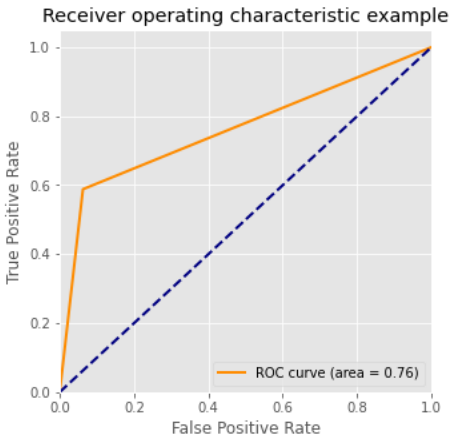
Деревья классификации были построены с целью прогнозирования послеоперационных осложнений у пациентов. Деревья позволяют определять правила, которые соответствуют определенным характеристикам пациента. Например, если у пациента уровень лейкоцитов выше определенного значения, то это может указывать на риск возникновения осложнений после операции.

При построении деревьев классификации было использовано разбиение выборки на обучающую и тестовую. Обучающая выборка использовалась для обучения деревьев, а тестовая - для оценки качества построенных деревьев.

Для каждого дерева был определен пороговый уровень ошибки, который указывал на качество прогнозирования послеоперационных осложнений деревом. Более точные деревья имели меньший уровень ошибки и могли использоваться для прогнозирования возникновения осложнений после операции с большей уверенностью.

В результате использования деревьев классификации удалось прогнозировать появление послеоперационных осложнений с высокой точностью. Это позволяет своевременно принимать меры для предотвращения возможных осложнений и улучшения исходов операции.

На рисунке 2.20 приведен результат ROC-анализа.




```
The accuracy score for method is: 75.8%
The f1 score for method is: 71.39999999999999%
The precision score for method is: 90.9%
The recall score for method is: 58.8%
```

Рисунок 2.20 – Результат построения дерева решений

На рисунке видно, что площадь под кривой составила 0,76, это говорит о достаточно высокой точности классификатора. Точность классификации составляет примерно 75,8%, это говорит о том что данный метод не достаточно хорошо определяет классы.

На рисунке 2.21 представлена матрица неточности. Она отображает число (или процент) правильно и неправильно распознанных распознанных примеров.

В нашей работе 1 больной с отсутствием послеоперационных осложнений дерево решений определило как больных с осложнениями (гипердиагностика), 7 больных с наличием послеоперационных осложнений дерево решений определило как с отсутствием осложнений (гиподиагностика).

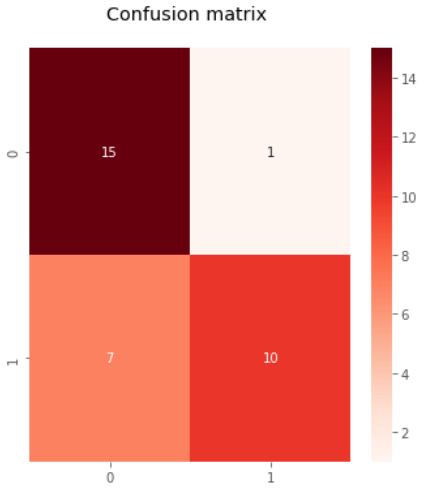


Рисунок 2.21 – Визуализатор матрица неточности

В данном исследовании разница между ошибками первого и второго не существенна, это является недостатком данного метода. Ошибка данного метода составила – 75,8%. Данный результат является неудовлетворительным для диагностики в области медицины.

2.4.3 Прогнозирование с помощью логистической регрессии

Прогнозирование с помощью логистической регрессии является одним из методов анализа данных и прогнозирования вероятности возникновения событий. Логистическая регрессия используется для моделирования связи между зависимой переменной и одной или несколькими независимыми переменными.

Данная методика может быть использована для прогнозирования послеоперационных осложнений. Для этого необходимо использовать данные об исходах операций, а также информацию о состоянии пациента перед операцией. Логистическая регрессия может использоваться для определения того, какие факторы оказывают наибольшее влияние на вероятность возникновения осложнений после операции.

Прежде чем применять логистическую регрессию, необходимо провести предварительный анализ данных, а именно, оценить качество данных, проверить наличие выбросов и пропусков, а также определить корреляцию между переменными.

Следующим шагом является построение модели логистической регрессии на основе имеющихся данных. Для этого используется алгоритм нахождения коэффициентов, который позволяет оценить влияние каждой из независимых переменных на вероятность возникновения осложнений после операции.

$$P = \frac{1}{1+e^{-y}},$$

где P — вероятность того, что произойдет интересующее событие;

e — основание натуральных логарифмов;

y — значение стандартного уравнения линейной регрессии.

После построения модели логистической регрессии необходимо провести ее проверку на тестовой выборке для оценки ее точности и качества прогнозирования. Если модель оказывается недостаточно точной, то следует провести ее дальнейшую настройку при помощи методов оптимизации.

Таким образом, прогнозирование с помощью логистической регрессии - это эффективный инструмент для определения вероятности возникновения послеоперационных осложнений и позволяет принимать своевременные меры для предотвращения их возникновения.

Данный алгоритм реализован в Google Colab. Прогнозирование проводилось на анализах, полученных ранее с помощью статистических методов. Классификатор Логистическая регрессия проводится для зависимой переменной Оценка (наличие осложнений – 1, отсутствие осложнений – 0) и 5 предикторов (Амилаза, Нейтрофилы палочкоядерные, Лимфоциты, Общий билирубин, Длительность оперативных вмешательств).

На основе выше описанного алгоритма был разработан сценарий, реализующий выполнение логистической регрессии. Результаты эксперимента представлены на рисунке 2.22.

```
The accuracy score for method is: 54.50000000000001%
The f1 score for method is: 28.599999999999998%
The precision score for method is: 75.0%
The recall score for method is: 17.599999999999998%
```

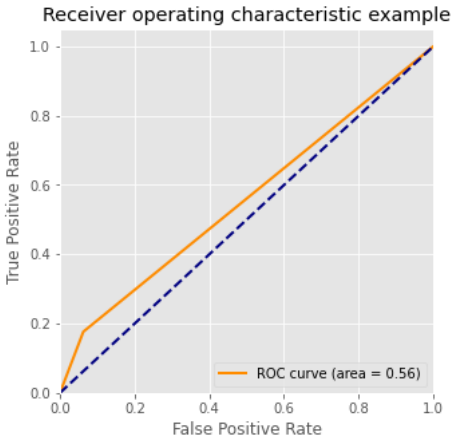


Рисунок 2.22 – Результат работы логистической регрессии

Данный алгоритм показал точность диагностики равную 54,5%, и площадь под ROC-кривой 0,56. Данный результат является неудовлетворительным для диагностики в области медицины. В данный процент ошибки входит лишь ошибка второго рода, это когда классификатор показал отсутствие осложнений, хотя на самом деле они есть. Это видно из матрицы неточностей (confusion matrix) на рисунке 2.23.

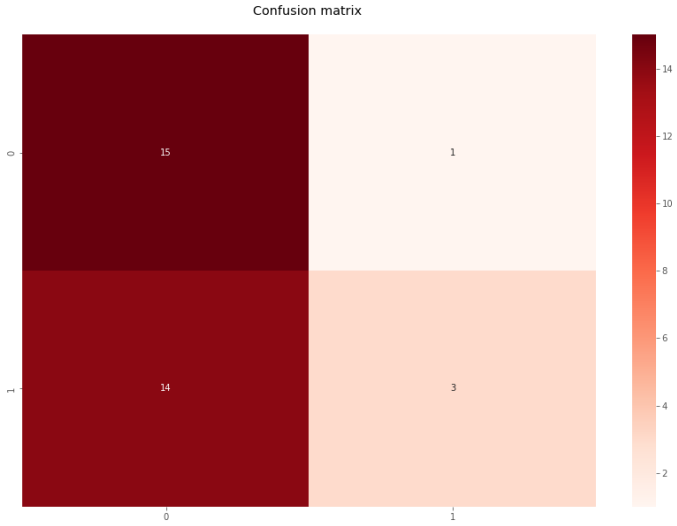


Рисунок 2.23 – Визуализатор матрица неточности

Однако, следует учитывать, что логистическая регрессия может быть ограничена в применении в случае наличия нелинейных связей или непрерывных переменных. В таких случаях могут использоваться другие методы анализа данных, например, машинное обучение.

Таким образом, прогнозирование с помощью логистической регрессии является одним из наиболее распространенных методов анализа данных в

области медицины. Этот метод позволяет прогнозировать различные медицинские показатели, что может существенно помочь в профилактике и лечении заболеваний.

2.4.4 Прогнозирование с помощью метода опорных векторов

Прогнозирование с помощью метода опорных векторов (SVM) является одним из методов машинного обучения и может использоваться для решения широкого круга задач, включая прогнозирование в медицинской сфере.

Метод опорных векторов основан на построении гиперплоскости в n-мерном пространстве, которая максимально разделяет данные разных классов. В случае прогнозирования SVM используется для построения модели, которая может предсказывать значения зависимой переменной на основе значений независимых переменных (рисунок 2.24).

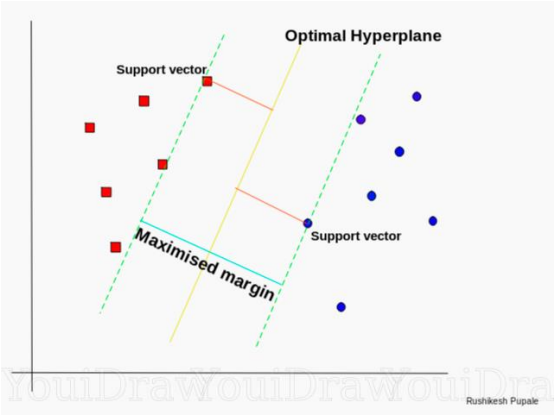


Рисунок 2.24 – Метод опорных векторов

Для прогнозирования медицинских параметров с помощью SVM, сначала необходимо собрать данные, содержащие информацию об исходах лечения и основных медицинских параметрах, которые могут влиять на результаты лечения.

Затем осуществляется предварительная обработка данных, включающая в себя очистку данных, выборку признаков и масштабирование данных. Важно отметить, что для использования SVM необходимо иметь признаки, которые линейно разделимы.

После предварительной обработки данных модель SVM строится на основе обучающей выборки. Она может использоваться для прогнозирования значений зависимой переменной для новых наблюдений.

$$a(x) = sign\left(\sum_{j=1}^{n} w_j x^j - w_0\right) = sign(\langle w, x\rangle - w_0)$$

где $x = (x^1, \ldots, x^n)$ – признаковое описание объекта *x*;

$w = (w^1, \ldots, w^n) \in \mathbb{R}^n -$ вектор;

$\langle w, x\rangle = w_0$ – уравнение, описывающее гиперплоскость, разделяющую классы в пространстве $\mathbb{R}^n$.

Одним из преимуществ метода опорных векторов является высокая точность прогнозирования, особенно если данные хорошо обработаны и содержат значимые признаки. Кроме того, SVM может использоваться для решения задач классификации, регрессии и обнаружения выбросов.

Однако, следует учитывать, что метод опорных векторов может быть громоздким и требовательным к вычислительным ресурсам, особенно при работе с большими объемами данных. Кроме того, он может страдать от проблемы переобучения в случае, если модель обучена на слишком малом количестве данных или не адаптирована к новым данным.

Таким образом, прогнозирование с помощью метода опорных векторов является эффективным способом прогнозирования медицинских параметров, который может быть использован для решения широкого круга задач в медицинской сфере.

Для применения метода опорных векторов в медицинской сфере можно использовать его для решения задач, таких как:

- диагностика заболеваний. SVM может быть использован для диагностики различных заболеваний на основе анализа медицинских данных пациентов,

- прогнозирование эффективности лечения. Данный метод может быть использован для прогнозирования результатов лечения пациентов на основе начальной клинической картины и других медицинских параметров,

- определение риска развития заболеваний. SVM может быть использован для определения риска развития заболеваний на основе медицинской истории, общего состояния пациента и других рисковых факторов,

- прогнозирование смертности. SVM может быть использован для прогнозирования смертности на основе медицинской истории, общего состояния пациента и других медицинских параметров.

Однако, следует отметить, что SVM может быть чувствителен к выбросам в данных и требователен к значительным объемам данных для эффективного прогнозирования.

Данный алгоритм реализован в Google Colab. Прогнозирование проводилось на анализах, полученных ранее с помощью статистических методов. Классификатор метод опорных векторов проводится для зависимой переменной Оценка (наличие осложнений – 1, отсутствие осложнений – 0) и 5 предикторов (Амилаза, Нейтрофилы палочкоядерные, Лимфоциты, Общий билирубин, Длительность оперативных вмешательств).

На основе вышеописанного алгоритма был разработан сценарий, реализующий выполнение метода опорных векторов. Результаты эксперимента представлены на рисунке 2.25.

```
The accuracy score for method is: 63.6%
The f1 score for method is: 53.800000000000004%
The precision score for method is: 77.8%
The recall score for method is: 41.199999999999996%
```

Рисунок 2.25 – Результат работы метода опорных векторов

Данный алгоритм показал точность диагностики равную 63,6%, и площадь под ROC-кривой 0,64. Данный результат является неудовлетворительным для диагностики в области медицины. Это видно из матрицы неточностей (confusion matrix) на рисунке 2.26.

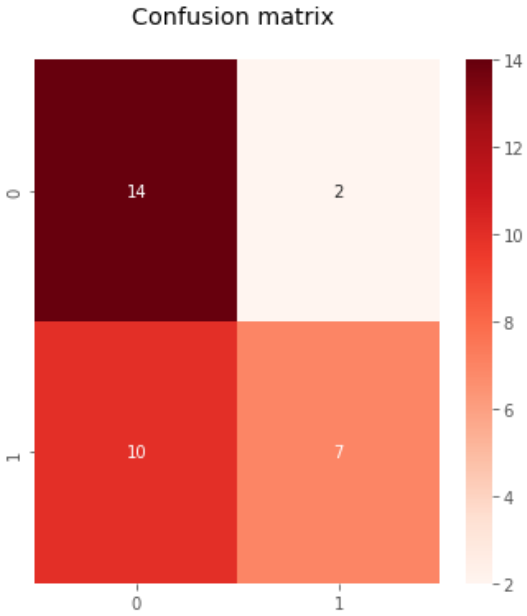


Рисунок 2.26 – Визуализатор матриц неточности

В данный процент ошибки входит лишь ошибка второго рода, это когда классификатор показал отсутствие осложнений, хотя на самом деле они есть.

2.4.5 Прогнозирование с помощью нейронных сетей

Прогнозирование с помощью нейронных сетей является одним из методов машинного обучения, который может использоваться для решения многих задач в различных сферах, включая медицину.

Нейронные сети являются моделями машинного обучения, которые моделируют работу мозга и используются для обработки сложных и неструктурированных данных. Нейронные сети работают на основе связей между нейронами, которые соединены между собой и формируют слои.

Для прогнозирования медицинских параметров с помощью нейронных сетей необходимо собрать данные, содержащие информацию об исходах лечения и медицинских параметрах, которые могут влиять на результаты лечения.

Затем осуществляется предварительная обработка данных, включающая в себя очистку данных, выборку признаков и масштабирование данных. Важно отметить, что для использования нейронных сетей, данные должны быть готовы для обучения модели.

После предварительной обработки данных модель нейронной сети строится на основе обучающей выборки. Она может использоваться для прогнозирования значений зависимой переменной для новых наблюдений.

Одним из преимуществ прогнозирования с помощью нейронных сетей является способность моделировать физиологические процессы, которые могут быть сложны для анализа с помощью других методов машинного обучения. Кроме того, нейронные сети могут использоваться для решения задач классификации, регрессии и кластеризации.

Однако, следует учитывать, что нейронные сети могут быть чувствительны к переобучению, особенно если модель обучена на слишком малом количестве данных. Кроме того, они могут страдать от проблемы «черного ящика», когда процесс принятия решений внутри модели сложно интерпретировать.

Таким образом, прогнозирование с помощью нейронных сетей является эффективным способом прогнозирования медицинских параметров, который может быть использован для решения широкого круга задач в медицинской сфере.

Для проведения исследования применялись нейронные сети различной конфигурации. Исследования проводились в пакете Neural Network Toolbox системы MATLAB. Одна из исследуемых нейронных сетей была настроена на решение задачи классификации, а именно прогнозирования послеоперационных осложнений у пациентов с МКБ. На вход нейронной сети подавались пять отобранных медицинских показателей, а на ее выходе формировался признак «наличие-отсутствие» послеоперационных осложнений.

Алгоритм Левенберга-Марквардта (LM) имеет самую быструю сходимость. Это преимущество особенно значимо, если требуется высокая точность обучения. Алгоритмы методов Ньютона и секущих плоскостей занимают следующее место для нейронных сетей умеренных размеров. Алгоритм BFGS [34] требует значительного объема памяти для хранения матрицы Гессе, но при этом значительно превосходит по быстродействию алгоритмы метода сопряженных градиентов. Алгоритм SCG не требуют использования процедур одномерного поиска и предъявляют незначительные требования к памяти. Он работает достаточно быстро и может быть рекомендованы для решения задач большой размерности.

Экспериментальные исследования решено было начинать с алгоритма LM. Потом эксперименты проводились с алгоритмом BFGS. Алгоритм SCG также характеризуется высоким быстродействием и предъявляет относительно малые требования к объему используемой памяти, он также был использован в экспериментах.

Алгоритм Левенберга-Марквардта (Levenberg-Marquardt Algorithm) — это метод численной оптимизации, используемый в решении проблем нелинейной регрессии или минимизации функции Хи-квадрат в некоторых методах анализа данных. Он широко используется в обработке изображений, компьютерном зрении, распознавании речи и других областях науки.

Алгоритм Левенберга-Марквардта представляет собой комбинацию методов градиентного спуска и гаусс-ньютоновского метода. Он позволяет быстро находить оптимальные значения параметров модели, используя поправки на каждой итерации на основе последовательного увеличения коэффициента регуляризации (фактора затухания) [34].

Основная идея алгоритма заключается в том, чтобы на каждой итерации использовать метод градиентного спуска, который обеспечивает быстрый процесс снижения функции Хи-квадрат, а затем переходить на метод гаусс-ньютоновского метода, который обеспечивает более точные оценки параметров.

В целом, алгоритм Левенберга-Марквардта является одним из наиболее эффективных методов для решения задач нелинейной оптимизации. Однако, он может оказаться неэффективным при решении некоторых сложных задач оптимизации, а также может потребовать значительных вычислительных ресурсов.

Один из примеров применения алгоритма Левенберга-Марквардта — это задача обучения нейронных сетей. Нейронные сети могут быть использованы для решения многих задач, таких как классификация, регрессия, и распознавание образов. Алгоритм Левенберга-Марквардта часто используется для обучения нейронных сетей, так как он позволяет достичь высокой точности прогнозирования и сократить количество итераций обучения.

Одним из главных преимуществ алгоритма Левенберга-Марквардта является его способность быстро находить оптимальные значения параметров модели и достигать высокой точности. Кроме того, алгоритм обеспечивает контроль за регуляризацией и предотвращает чрезмерное увеличение значений параметров.

Алгоритм Левенберга-Марквардта использует специальную стратегию для оценки матрицы Гессе.

$$H \cong J^T J,$$

а градиент рассчитан по формуле:

$$g = J^T e,$$

где $J = \frac{\partial J}{\partial W}$ – матрица Якоби производных функционала ошибки по настраиваемым параметрам;

e – вектор ошибок сети.

Алгоритм SCG (Scaled Conjugate Gradient) — это метод численной оптимизации, предложенный Моллером в 1993 году, который используется в решении задач нелинейной оптимизации и обучения нейронных сетей. Он основан на методе сопряженных градиентов, который является одним из самых быстрых методов оптимизации [33].

Основная идея алгоритма SCG заключается в том, чтобы автоматически выбирать оптимальный размер шага на каждой итерации оптимизации, исходя из текущего значения производной функции. Это позволяет быстро достичь оптимального значения функции за меньшее количество итераций, чем другие методы оптимизации.

Алгоритм SCG использует два различных параметра масштабирования - один для размера шага и другой для скорости обучения. Первый параметр используется для ограничения размера шага на каждой итерации, чтобы избежать переобучения и продолжения движения в направлении, которое уже было исследовано. Второй параметр используется для контроля градиентного спуска, т.е. скорости, с которой алгоритм принимает решения на каждой итерации.

В целом, алгоритм SCG является очень эффективным методом оптимизации, который может использоваться для решения различных задач, таких как обучение нейронных сетей, классификация, регрессия и распознавание образов. Его преимущества включают быструю скорость сходимости, автоматический подбор размера шага и скорости обучения, и легкость в реализации.

При выборе метода оптимизации был учтен фактор не положительности матрицы Гессе, чтобы избежать сложностей в алгоритме. Формулы обновления коэффициентов для этого метода, следующие:

$$w_{k+1} = w_k + a_k p_k,$$

$$a_k = \frac{\mu_k}{\delta_k}, \mu_k = -p_k^T E_k^{'}, \delta_k = p_k^T S_k,$$

$$S_k = E^{''}(w_k) p_k \frac{E^{'}(w_k + \delta_k p_k) - E^{'}(w_k)}{\delta_k} + \lambda_k p_k$$

где $k = 1,2,\ldots,N$, a_k – размер шага;

p_k – сопряженный вектор;

E_k – вектор ошибки;

δ_k – матрица Гессе, $\delta_k = \frac{\delta}{|p_k|}$;

λ_k – параметр масштабирования матрицы Гессе.

BFGS (Broyden-Fletcher-Goldfarb-Shanno) — это квазиньютоновский метод безусловной оптимизации, который позволяет эффективно и быстро находить оптимальные решения второго порядка. Он относится к классу методов оптимизации второго порядка, так как использует информацию о вторых производных функции:

$$argmin\, f\,(x).$$

Основным преимуществом метода BFGS является то, что он позволяет быстро сходимость к оптимальному решению без необходимости явного вычисления вторых производных функции. Вместо этого он оценивает их, используя информацию о градиентах функции.

BFGS использует квазиньютоновскую формулу, которая позволяет вычислять новую оценку матрицы Гессе на каждой итерации. Эта формула напоминает формулу обратного распространения ошибки, используемую в нейронных сетях, и является ключевой особенностью метода BFGS.

$$f(p + x_k) \approx f(x_k) + \nabla f^T(x_k) p + \frac{1}{2} p^T H_k p,$$

где H – матрица Гессе функционала f в точке x;

∇f – вектор градиента функционала.

В целом, метод BFGS является эффективным и точным методом оптимизации, который может использоваться в различных областях науки и техники. Он обладает свойствами устойчивости, быстрой скорости сходимости и не требует больших объемов памяти для хранения матрицы Гессе [35].

Метод BFGS обеспечивает быстрое и точное решение задач оптимизации, но имеет также и некоторые недостатки. В частности, он не гарантирует глобальную сходимость, а также может быть чувствителен к выбору начального приближения и соотношению масштабов между переменными.

Цель данного исследования состоит в сравнении нескольких алгоритмов обучения на основе данных, собранных с использованием различных архитектур нейронных сетей. В ходе экспериментов были протестированы различные варианты количества нейронов в скрытых слоях. Для определения эффективности алгоритмов обучения использовалась вероятность распознавания, определенная по результатам тестирования. Прогнозирование считалось верным, если значение выхода сети было больше 0,8.

Для дополнительной оценки качества классификации был применен ROC-анализ и рассчитан показатель AUC [67]. AUC представляет собой площадь под ROC-кривой и дает численное представление ее качества. Чем выше значение AUC, тем более качественным считается классификатор.

Проблема переобучения (overfitting) является одной из наиболее распространенных проблем при обучении нейронных сетей. Она возникает, когда модель обучается на обучающих данных вплоть до того, что она начинает "запоминать" их вместо того, чтобы обобщать полученный опыт. В результате переобученная модель не сможет правильно обрабатывать новые данные, что делает ее бессмысленной.

Для борьбы с переобучением можно использовать различные техники, такие как:

- регуляризация: добавление дополнительных членов в функцию ошибки для ограничения параметров модели,

- дропаут (dropout): исключение случайно выбранных нейронов из обучения на каждой итерации,

- увеличение количества обучающих данных: большее количество данных может помочь модели обобщать опыт лучше,

- ранняя остановка (early stopping): остановка обучения, когда производительность модели начинает ухудшаться на наборе проверочных данных.

Кроме того, в процессе обучения нейронных сетей можно использовать различные стратегии кросс-валидации, такие как:

- перекрестная проверка (cross-validation): разделение данных на несколько частей для обучения модели на одной части и проверки на других,

- отложенная выборка (hold-out validation): разделение данных на две части, одна из которых используется для обучения, а другая - для проверки,

- K-fold кросс-валидация: разделение данных на K "сложений", в каждом из которых K-1 сложение используется для обучения и 1 сложение - для проверки.

Также можно использовать аугментацию данных (data augmentation) для расширения набора данных и улучшения качества модели. Аугментация данных заключается в создании дополнительных обучающих примеров за счет различных преобразований и добавления шума к существующим данным.

В целом, правильный выбор методов и техник обучения может максимально снизить вероятность переобучения модели и помочь с достижением более высокой производительности в решении задач машинного обучения.

Для оценки эффективности обучения искусственных нейронных сетей использован общепринятый стандартный показатель - среднеквадратичная ошибка (СКО) обучения.

$$СКО = \frac{1}{N}\sum_{i=1}^{N}(e_i)^2 = \frac{1}{N}\sum_{i=1}^{N}(d_i - y_i)^2,$$

где N – объем выборки (число примеров в обучающем множестве;

e_i – ошибка сети;

d_i – желаемая величина выхода;

y_i – реально полученные на сети значения для каждого примера i.

Исследования, по прогнозированию послеоперационных осложнений проводились на статистическом материале больных мочекаменной болезнью отделения торакальной хирургии областной клинической больницы им. Н.Н. Бурденко. База исходных данных содержит клинико-лабораторные показатели 109 пациентов, из них 63 случая без осложнения, 46 осложненных случаев.

По результатам статистического анализа были отобраны наиболее информативные признаки: Лейкоциты, Нейтрофилы палочкоядерные, Лимфоциты, Общий билирубин, Длительность оперативных вмешательств. Это позволило сократить число анализов с 15 до 5.

Результаты экспериментов с использованием алгоритма обучения LM представлены на рисунках 2.27-2.30 представлены характеристики качества работы архитектур нейронных сетей, а также функции активации.

Архитектура	Распознанных		AUC	Функция ошибки	Ф-я актив. Скрытых нейр.	Ф-я актив. выходных нейр.
	Правильно (%)	Неправильно (%)				
5-3-1	74,30	25,7	0,44	Сум. квадр.	Гиперболическая	Гиперболическая
5-11-1	57,8	42,2	0,48	Сум. квадр.	Гиперболическая	Логистическая
5-9-1	77,9	22,1	0,45	Сум. квадр.	Гиперболическая	Софтмакс
5-15-1	75,1	22,9	0,44	Сум. квадр.	Гиперболическая	Логистическая
5-10-1	65,10	34,9	0,50	Сум. квадр.	Гиперболическая	Гиперболическая
5-13-1	72,5	27,5	0,39	Сум. квадр.	Гиперболическая	Софтмакс
5-17-1	73,4	26,6	0,37	Сум. квадр.	Гиперболическая	Гиперболическая

Рисунок 2.27 – Результаты обучения нейронных сетей с пятью входным нейронами (алгоритм LM)

Из рисунка 2.27 видно, что при обучении нейронных сетей с использованием алгоритма обучения Левенберга-Марквардта удачной является архитектура 5-9-1. Ошибка данной сети составила 21,1%.

Архитектура	Распознанных		AUC	Функция ошибки	Ф-я актив. Скрытых нейр.	Ф-я актив. выходных нейр.
	Правильно (%)	Неправильно (%)				
5-3-1	42,2	57,8	0,009	Сум. квадр.	Гиперболическая	Гиперболическая
5-11-1	67	33	0,51	Сум. квадр.	Логистическая	Логистическая
5-9-1	69,8	30,2	0,67	Сум. квадр.	Гиперболическая	Софтмакс
5-15-1	58,7	41,3	0,28	Сум. квадр.	Гиперболическая	Логистическая
5-10-1	66,1	33,9	0,46	Сум. квадр.	Гиперболическая	Гиперболическая
5-13-1	65,1	34,9	0,30	Сум. квадр.	Логистическая	Софтмакс
5-17-1	69,7	30,3	0,45	Сум. квадр.	Логистическая	Логистическая

Рисунок 2.28 – Результаты обучения нейронных сетей с пятью входным нейронами (алгоритм BFGS)

Наиболее удачная архитектура нейронной сети является 5-9-1 (рисунок 2.28). Данная архитектура обучалась с использованием алгоритма BFGS, функцией активации скрытых нейронов являлся гиперболический тангенс, а функцией активации выходных нейронов являлась softmax. Ошибка сети составила 23,9%.

Архитектура	Распознанных		AUC	Функция ошибки	Ф-я актив. Скрытых нейр.	Ф-я актив. выходных нейр.
	Правильно (%)	Неправильно (%)				
5-3-1	60,6	39,4	0,38	Сум. квадр.	Гиперболическая	Гиперболическая
5-11-1	64,2	35,8	0,24	Сум. квадр.	Логистическая	Логистическая
5-9-1	68,8	31,2	0,43	Сум. квадр.	Гиперболическая	Софтмакс
5-15-1	59,6	40,4	0,34	Сум. квадр.	Гиперболическая	Логистическая
5-10-1	63,3	36,7	0,46	Сум. квадр.	Гиперболическая	Гиперболическая
5-13-1	61,5	38,5	0,42	Сум. квадр.	Логистическая	Софтмакс
5-17-1	51,4	48,6	0,41	Сум. квадр.	Логистическая	Логистическая

Рисунок 2.29 – Результаты обучения нейронных сетей с пятью входным нейронами (алгоритм SCG)

Из рисунка 2.29 видно, что при обучении на тех же архитектурах нейронных сетей, но с использованием алгоритма обучения SCG наиболее удачная архитектура 5-9-1.

Алгоритм обучения	Ошибка (%)
Алгоритм LM	22,1
Алгоритм BFGS	30,2
Алгоритм SCG	31,2

Рисунок 2.30 – Обобщенные результаты обучения нейронных сетей

Согласно полученным результатам, можно заключить, что наиболее подходящей архитектурой нейронной сети является 5-9-1, использующая алгоритм LM и функцию активации скрытых нейронов - гиперболический тангенс, а функция активации выходных нейронов - softmax. Однако, учитывая процент ошибки при прогнозировании, можно предположить, что применение нейронных сетей в качестве технологии для прогнозирования послеоперационных осложнений у больных МКБ не обеспечивает достаточно высокую точность.

На основе вышеописанного алгоритма был разработан сценарий, реализующий выполнение нейронной сети архитектуры 5-9-1 с

использованием алгоритма LM, функцией активации скрытых нейронов являлся гиперболический тангенс, а функцией активации выходных нейронов являлась softmax. Результаты эксперимента представлены на рисунке 2.31.

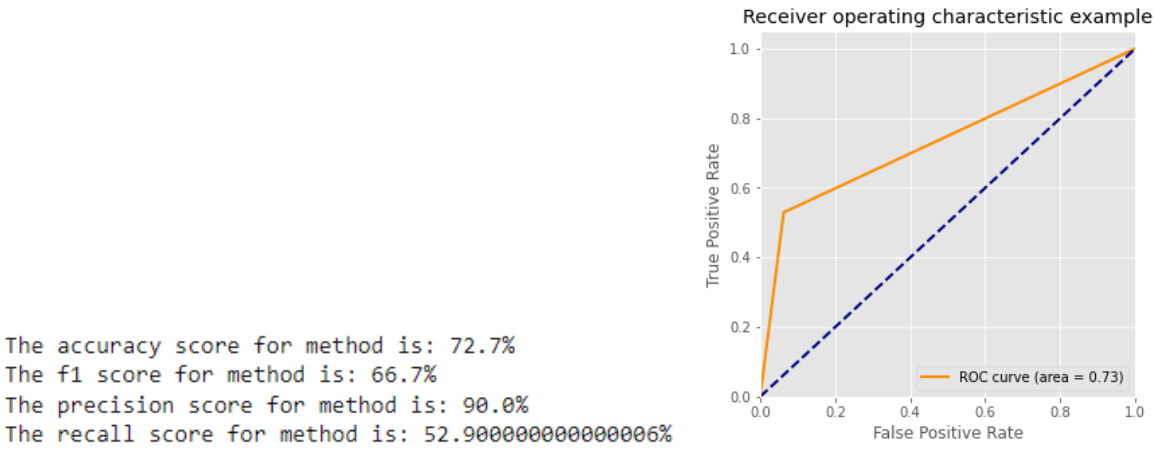


Рисунок 2.31 – Результат работы нейронной сети

Данный алгоритм показал точность диагностики равную 72,87%, и площадь под ROC-кривой 0,73. Данный результат является недостаточно точен.

В данный процент ошибки входят как ошибки первого рода, так и ошибки второго рода. Это видно из матрицы неточностей (confusion matrix) на рисунке 2.32.

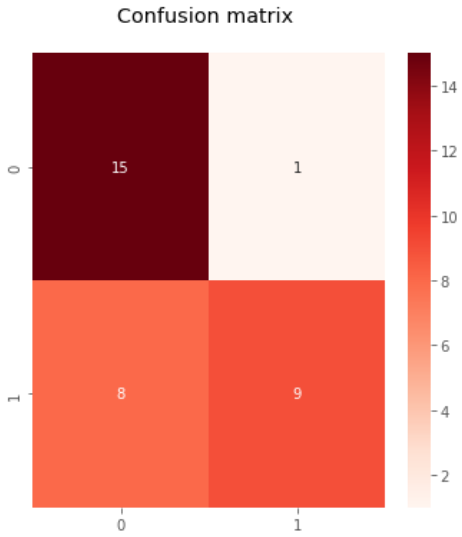


Рисунок 2.32 – Визуализатор матрица неточности

2.4.6 Прогнозирование с помощью алгоритма Случайный лес

Дерево решений может оказаться очень большим, со многими уровнями, и его считать придется очень долго. Это не эффективно. Можно ограничить

глубину дерева (число уровней), но тогда пострадает качество решения задачи. Выход - использовать много меньших деревьев. Пусть одно из них не сможет точно решить задачу, но их совокупность может быть точнее. Если 90 из 100 деревьев сообщают о классе №1, а остальные о классе №2, то можно поверить большинству и в окончательном решении принять класс №1. Получается что не одно дерево, а их множество (совокупность, комитет, объединение и пр. синонимы) будет принимать решение. Тогда отдельное дерево можно сделать попроще, с меньшей глубиной.

Но правило толпы, что выбор наиболее популярного решения будет более правильный и точный, как и в нашей жизни с голосованиями, работает далеко не всегда. Важно, чтобы голосующие были как можно меньше связаны между собой, чтобы не впадать в общие заблуждения. Так же и для деревьев, надо чтобы деревья в этом множестве были не совсем похожи друг на друга. Этого можно добиться (частично) если деревья создавать на случайно выбранных подмножествах примеров (возможно с повторениями) и случайно выбранных атрибутах.

Алгоритм случайный лес (random forest) представляет собой метод машинного обучения, который используется для решения задач классификации, регрессии и кластеризации.

Основным принципом алгоритма случайный лес является построение ансамбля деревьев решений, которые обучаются на различных подмножествах обучающих данных и выборках признаков. Каждый дерево решений работает на основе определенного подмножества данных и признаков, что позволяет избежать переобучения и улучшить обобщающую способность модели.

Для построения случайного леса необходимо выполнить следующие три шага:

1) разбить исходный набор данных на несколько случайных поднаборов размера n,

2) для каждого поднабора выполнить построение дерева решений на основе того, какое условие разделит данные наиболее эффективно (например, на основе критерия Джини или энтропии),

3) выбрать наиболее часто встречающееся предсказание каждого дерева и сделать итоговое предсказание на основе голосования.

Преимущества алгоритма случайный лес:

- высокая точность классификации или регрессии,
- устойчивость к шуму и выбросам в данных,
- способность обрабатывать большие объемы данных,
- возможность оценки важности признаков.

Недостатки алгоритма случайный лес:

- высокая вычислительная сложность в случае использования большого количества деревьев,
- затруднения в интерпретации результатов (при большом количестве признаков).

Тем не менее, благодаря своей эффективности и высокой точности, алгоритм случайный лес широко используется в задачах классификации, регрессии и кластеризации, включая обработку изображений, анализ текстовых данных, рекомендательные системы, медицинские и финансовые исследования и другие области.

Данный алгоритм реализован в Google Colab. Прогнозирование проводилось на анализах, полученных ранее с помощью статистических методов. Классификатор случайный лес проводится для зависимой переменной Оценка (наличие осложнений – 1, отсутствие осложнений – 0) и 5 предикторов (Амилаза, Нейтрофилы палочкоядерные, Лимфоциты, Общий билирубин, Длительность оперативных вмешательств).

На основе вышеописанного алгоритма был разработан сценарий, реализующий выполнение случайного леса. Результаты эксперимента представлены на рисунке 2.33.

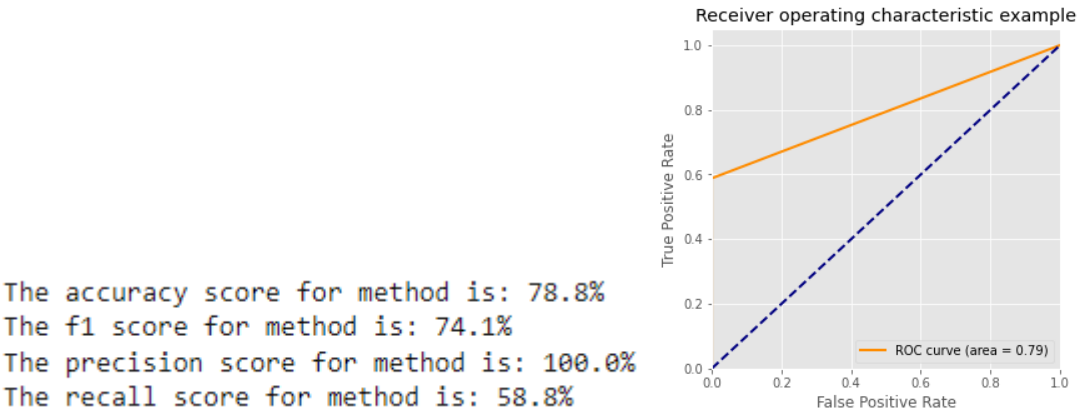


Рисунок 2.33 – Результат работы алгоритма случайный лес

Данный алгоритм показал точность диагностики равную 78,8%, и площадь под ROC-кривой 0,79. Данный результат является удовлетворительным для диагностики в области медицины.

В данный процент ошибки входит лишь ошибка первого рода, это когда классификатор показал наличие осложнений, хотя на самом деле их нет. Это видно из матрицы неточностей (confusion matrix) на рисунке 2.34.

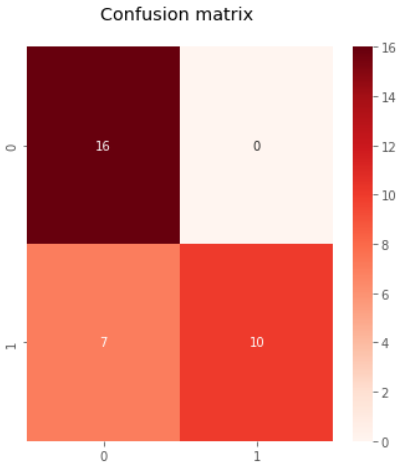


Рисунок 2.34 – Визуализатор матрица неточности

2.5 Измерение качества классификации

Подавая на вход классификатора контрольные выборки, не участвовавшие в обучении классификатора, но принадлежность которых к классам известна, можно оценить эффективность (качество) работы исследуемого классификатора.

Для оценки качества классификации рассмотренных алгоритмов воспользуемся -мерой.

F-мера складывается из полноты и точности и вычисляется для каждого класса в отдельности, а затем, если нужна общая -мера, берётся среднее значение по всем классам [67].

Полнота по классу:

$$r_i = \frac{T_i}{M_i}, i = \overline{1, K},$$

где T_i – число объектов, правильно приписанных к классу i;

M_i – общее число объектов класса i, имеющихся в тестируемых данных;

K– число классов.

Полнота системы представляет собой процент объектов, принадлежащих к определенному классу и найденных классификатором в тестовой выборке по отношению к общему количеству документов этого класса в выборке.

Точность по классу:

$$p_i = \frac{T_i}{C_i}, i = \overline{1, K},$$

где C_i – общее число объектов, приписанных к классу i (если $C_i = 0$, то считается $p_i = 0$).

Точность системы по конкретному классу определяется как отношение числа правильно классифицированных объектов данного класса к общему числу объектов, которые система отнесла к этому классу.

Необходим отметить, что точность и полнота не зависят от соотношения размеров классов. Даже если объектов положительного класса на порядки

меньше, чем объектов отрицательного класса, данные показатели будут корректно отражать качество работы алгоритма.

Это способ разбить объекты на четыре категории в зависимости от комбинации истинного ответа и ответа алгоритма рисунок 2.35. Через элементы этой таблицы можно, например, выразить долю правильных ответов:

Категория i		Экспертная оценка	
		Положительная	Отрицательная
Оценка системы	Положительная	TP	FP
	Отрицательная	FN	TN

Рисунок 2.35 – Таблицы неточности

Для заданного класса в таблице содержится информация о том, сколько раз система приняла верное и неверное решение по объектам этого класса. Обозначения используемых показателей, следующие:

- TP - количество объектов, которые система верно классифицировала как принадлежащие к данному классу,

- TN - количество объектов, которые система верно классифицировала как не принадлежащие к данному классу,

- FP - количество объектов, которые система неверно классифицировала как принадлежащие к данному классу,

- FN - количество объектов, которые система неверно классифицировала как не принадлежащие к данному классу.

Таким образом, значения TP, TN, FP и FN являются важными для оценки точности и полноты работы классификатора и составляют основу для вычисления метрик производительности.

Тогда, точность и полнота определяются следующим образом:

$$r = \frac{TP}{TP + FN}, p = \frac{TP}{TP + FP}.$$

Для оценки качества работы алгоритма предлагается использовать F-мера на основе метрик точности и полноты. Она представляет собой гармоническое среднее между точностью и полнотой. Она стремится к нулю, если точность или полнота стремится к нулю.

F-мера:

$$F = 2\frac{p * r}{p + r}.$$

В ходе проведенного исследования рассматривались три статистических метода классификации:

- дискриминантный анализ,
- деревья решений,
- байесовский классификатор.

Каждый из этих методов имеет свои достоинства и недостатки, поэтому была проведена их сравнительная аналитика. Дискриминантный анализ — это метод, основанный на разбиении выборки на группы по классам и построении оптимальной разделяющей гиперплоскости между группами. Деревья решений основаны на последовательных вопросах, которые позволяют разделить выборку на подмножества с наиболее похожими объектами. Байесовский классификатор — это метод, основанный на теореме Байеса и вероятностных характеристиках классов, которые позволяют определить принадлежность объекта к определенному классу на основе анализа его признаков.

Путем сравнительного анализа этих методов выявлены их сильные и слабые стороны, что позволило выбрать наиболее подходящий метод для решения конкретной задачи. Более того, некоторые аспекты были доработаны и улучшены для увеличения точности и производительности системы классификации. Например, была использована комбинация деревьев решений

и байесовского классификатора, что позволило улучшить точность классификации и сократить время обработки больших объемов данных.

По каждому методу были рассчитаны метрики полноты и точности. Данные метрики представлены на рисунке 2.36.

```
The accuracy score for LogReg is: 54.50000000000001%
Mean Absolute Error: 0.45454545454545453
Mean Absolute Percentage Error: 136472715980924.55
Mean Squared Error: 0.45454545454545453
Root Mean Squared Error: 0.674199862463242
R2 Score: -0.8198529411764706
None
The accuracy score for KNN is: 57.599999999999994%
Mean Absolute Error: 0.42424242424242425
Mean Absolute Percentage Error: 0.42424242424242425
Mean Squared Error: 0.42424242424242425
Root Mean Squared Error: 0.6513389472789296
R2 Score: -0.6985294117647058
None
The accuracy score for DTC is: 75.8%
Mean Absolute Error: 0.24242424242424243
Mean Absolute Percentage Error: 136472715980924.33
Mean Squared Error: 0.24242424242424243
Root Mean Squared Error: 0.49236596391733095
R2 Score: 0.02941176470588236
The accuracy score for SVM is: 63.6%
Mean Absolute Error: 0.36363636363636365
Mean Absolute Percentage Error: 272945431961848.6
Mean Squared Error: 0.36363636363636365
Root Mean Squared Error: 0.6030226891555273
R2 Score: -0.4558823529411764
None
The accuracy score for PPN is: 69.69999999999999%
The accuracy score for MLP is: 72.7%
Mean Absolute Error: 0.2727272727272727
Mean Absolute Percentage Error: 136472715980924.36
Mean Squared Error: 0.2727272727272727
Root Mean Squared Error: 0.5222329678670935
R2 Score: -0.09191176470588247
None
The accuracy score for RandomForest is: 78.8%
Mean Absolute Error: 0.21212121212121213
Mean Absolute Percentage Error: 0.21212121212121213
Mean Squared Error: 0.21212121212121213
Root Mean Squared Error: 0.4605661864718383
R2 Score: 0.15073529411764708
```

Рисунок 2.36 – Метрики полноты и точности

Из рисунка видно, что удовлетворительный результат прогнозирования для диагностики в области медицины лучше всего себя показал метод Случайный лес, его точность составила 78,8%.

График оценки точности методов приведен на рисунке 2.37.

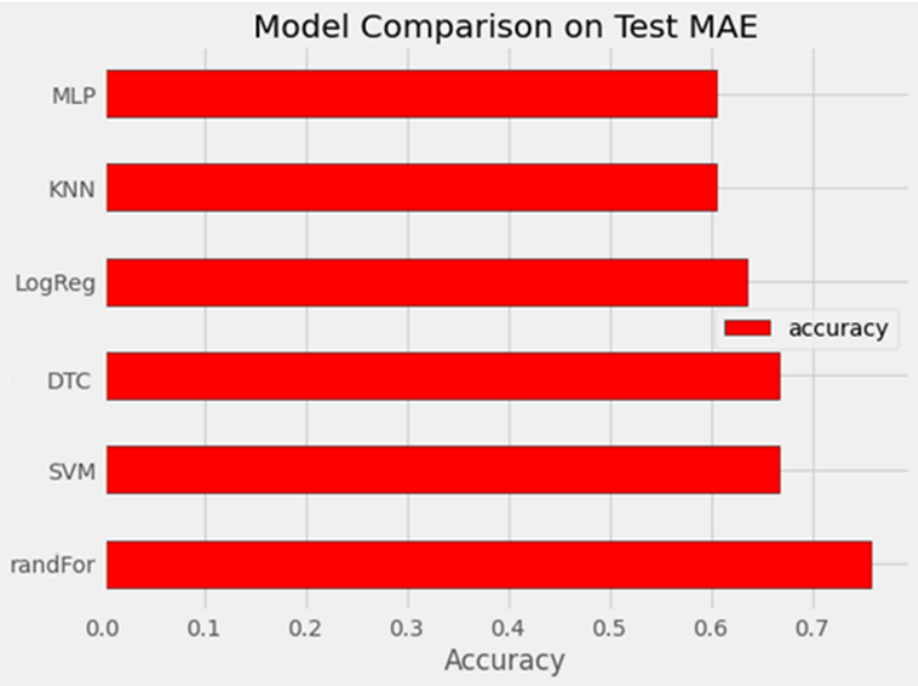


Рисунок 2.37 – График оценки точности методов

В связи с этим для дальнейшей реализации целесообразно использовать обученную модель с применением Случайного леса.

3 Проектирование программного продукта

Чат-бот - программа, которая может взаимодействовать с пользователем через текстовые сообщения и отвечать на вопросы или выполнять определенные задачи. Он использует искусственный интеллект и алгоритмы обработки естественного языка для понимания и анализа сообщений, которые получает от пользователя. Чат-боты часто используются в бизнесе, маркетинге и образовании для автоматизации общения с клиентами и учениками, а также для предоставления информации и услуг. Они могут быть интегрированы в мессенджеры, социальные сети или веб-сайты, что позволяет пользователю легко и удобно получать необходимые ответы и решать свои проблемы.

Чат-боты могут быть настроены на разные задачи, включая ответы на часто задаваемые вопросы, обслуживание клиентов, заказ товаров и услуг, проведение опросов и даже обучение языковым навыкам. Кроме того, чат-боты могут использоваться для мониторинга и анализа данных, сбора обратной связи и определения тенденций и предпочтений пользователей.

Реализация чат-ботов может осуществляться как с использованием программирования с нуля, так и с использованием готовых платформ и библиотек, которые предоставляют шаблоны и инструменты для создания чат-ботов. Среди таких платформ можно выделить DialogFlow от Google, IBM Watson, Amazon Lex и Microsoft Bot Framework.

Чат-боты становятся все более популярными и востребованными в современном мире, благодаря своей универсальности, доступности и экономичности. Они помогают упростить и автоматизировать множество бизнес- и повседневных задач, что позволяет сэкономить время и ресурсы, а также повысить качество обслуживания клиентов и пользователей.

На основании проведенных исследований на данную тематику были сформулированы следующие функциональные требования для бота:

- возможность загрузки модели обучения,
- возможность загрузки выборки,
- возможность проведения тестирования,

- просмотр точности модели,

- просмотр состояния пациента.

Разработка архитектуры чат-бота включает восемь этапов:

1) создание веб-клиента, который представляет собой серверное приложение, получающее обновления от Telegram через webhook и передающее их обработчику обновлений,

2) организация хранения выборки для обучения модели и ее тестирования с использованием репозитория GitHub,

3) идентификация пользователей, осуществляемая через уникальный идентификатор чата,

4) взаимодействие с чат-ботом, имеется возможность отправки сообщений, чтобы взаимодействовать с пользователями,

5) прием сообщений, который осуществляется через механизм приемников,

6) работа с кнопками, добавляемыми в сообщения для обработки нажатий кнопок был добавлен класс CallbackQuerryItem в проект Entities,

7) обработчик обновлений, который осуществляет обработку приходящих от клиента обновлений через класс CommonHandler в проекте UpdateHandlers,

8) бизнес-логика, необходимая для обеспечения требуемой функциональности бота.

3.1 Компонентная архитектура

Компонентная архитектура является методом проектирования и разработки систем и заключается в разбиении дизайна на функциональные или логические компоненты с определенными интерфейсами, содержащими методы, события и свойства. Это обеспечивает более высокий уровень абстракции, чем объектно-ориентированное развитие, и не фокусируется на протоколах связи или общем состоянии.

Существует два подхода к реализации программного обеспечения:

- однокомпонентная архитектура,

- многокомпонентная архитектура.

В первом случае всё приложение разрабатывается как единое целое, а во втором - как набор связанных компонентов. При многокомпонентной архитектуре компоненты могут быть разработаны и использованы повторно в разных системах, что способствует более эффективной разработке программного обеспечения.

3.1.1 Однокомпонентная архитектура

Однокомпонентная архитектура — это подход к разработке программного обеспечения, при котором все части приложения разрабатываются и собираются как единое целое. В данном случае есть один основной модуль, обеспечивающий взаимодействие с пользователем и выполнение основных функций приложения.

Однокомпонентная архитектура является простой и быстрой для разработки, но приложения, разработанные таким образом, не могут модифицироваться или расширяться без изменения всего приложения. Это может привести к риску нарушения работоспособности системы.

Также, при использовании однокомпонентной архитектуры для больших проектов может быть сложно поддерживать код и проводить его тестирование, так как весь код находится в одном монолитном блоке.

3.1.2 Многокомпонентная архитектура

Многокомпонентная архитектура — это подход к разработке программного обеспечения, при котором приложение делится на несколько компонентов, каждый из которых отвечает за определенную функциональность. Компоненты соединяются между собой через интерфейсы, которые определяют правила взаимодействия компонентов.

Многокомпонентная архитектура позволяет легко разрабатывать и расширять приложения. Каждый компонент может быть разработан, протестирован и поддерживаться независимо друг от друга. При необходимости добавления новой функциональности достаточно разработать новый компонент и соединить его с остальной частью системы.

Более того, компоненты могут быть переиспользованы в различных проектах, что существенно сокращает время и затраты на разработку новых приложений.

Однако многокомпонентная архитектура требует большего количества времени и ресурсов на проектирование и разработку приложения, так как необходимо определить и разработать компоненты, их интерфейсы и методы взаимодействия.

Ещё одним преимуществом многокомпонентной архитектуры является возможность проведения параллельной разработки компонентов. Разные разработчики могут заниматься разработкой разных компонентов одновременно. Это может существенно ускорить разработку и снизить риски ошибок, поскольку разные компоненты будут создаваться независимо друг от друга.

Кроме того, многокомпонентная архитектура обеспечивает лучшую масштабируемость и управляемость приложения, так как каждый компонент может быть изменен или заменен без воздействия на другие компоненты.

Недостатки многокомпонентной архитектуры включают более высокий уровень сложности в сравнении с однокомпонентной архитектурой, так как необходимо размышлять о том, как разбить целое приложение на компоненты и каким образом соединить их. Кроме того, процесс тестирования приложения в многокомпонентной архитектуре может быть более трудоемким, поскольку нужно проводить тестирование каждого компонента и их взаимодействия.

Данная архитектура графически представлена на рисунке 3.1.

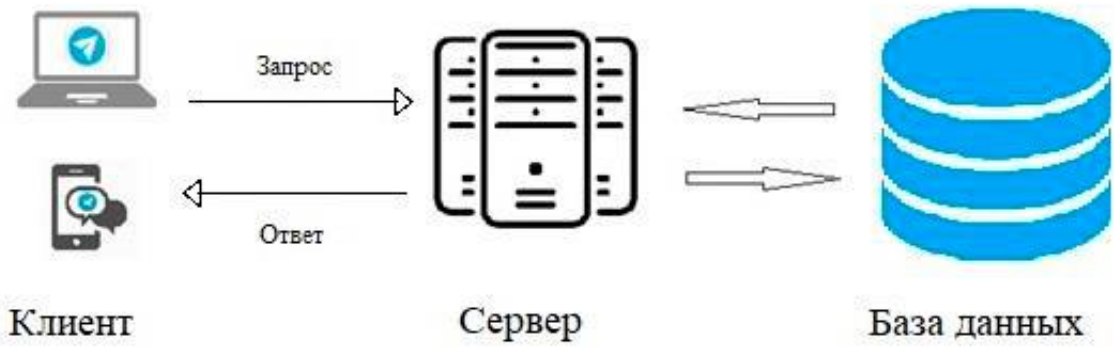


Рисунок 3.1 – Трехуровневая компонентная архитектура

3.2 Архитектура программного обеспечения

Проектирование выбранной трехуровневой архитектуры будет выполнено с использованием сервис-ориентированного подхода, который основывается на создании распределенного программного обеспечения, состоящего из отдельных сервисов. Этот подход обладает рядом преимуществ, включая возможность быстро адаптироваться к меняющимся требованиям информационного мира, благодаря независимости каждого сервиса и возможности их легкой замены.

Кроме того, сервис-ориентированный подход позволяет лучше масштабировать приложение, поскольку каждый сервис может быть масштабирован независимо от других. Это снижает риски сбоев и обеспечивает лучшую производительность системы в целом.

Однако, использование сервис-ориентированного подхода также может привести к возникновению некоторых сложностей, связанных с координацией и управлением сервисами, а также с поддержкой необходимых инфраструктурных компонентов.

В целом, выбранная архитектура является современной и эффективной, и должна обеспечить достаточную гибкость и масштабируемость для реализации поставленных требований и задач.

Веб-сервисы SOAP и REST являются наиболее распространенными веб-сервисами на сегодняшний день. Веб-сервисы SOAP основываются на протоколе XML и представляют стандартизированный подход к обмену сообщениями между приложениями через интернет. Они поддерживаются большинством языков программирования и платформ.

REST сервисы, в свою очередь, являются более легковесными и простыми по сравнению с SOAP. Они основываются на протоколах HTTP и JSON, что делает их более гибкими и быстрыми в работе.

Каждый из подходов имеет свои преимущества и недостатки и подходит для разных сценариев использования. SOAP сервисы часто используются в больших корпоративных системах, где необходима высокая точность и

надежность обмена сообщениями, а REST сервисы чаще используются в более масштабируемых, распределенных системах для быстрого и легкого доступа к данным и ресурсам.

Также стоит отметить, что REST сервисы являются более популярными в создании API (Application Programming Interface) для веб-приложений и мобильных приложений, поскольку они более просты в использовании и позволяют создавать более гибкие и удобные для разработчиков API. Однако, в некоторых случаях SOAP может быть более удобным выбором, например, если требуется высокая безопасность и целостность данных.

Несмотря на то, что выбор между SOAP и REST может быть зависеть от конкретных потребностей проекта, все больше разработчиков предпочитают использовать REST сервисы благодаря их простоте и более широкой поддержке в современных технологиях.

API RESTful — это подход к разработке веб-сервисов, основанных на архитектуре REST. RESTful API является гибким, масштабируемым и эффективным способом создания API для веб-приложений и мобильных приложений. Он основывается на использовании URL-адресов для доступа к ресурсам и HTTP-методов для выполнения операций с ними.

Одним из главных преимуществ RESTful API является его простота и легкость в использовании. Он позволяет разработчикам быстро и легко создавать API, обеспечивая доступ к ресурсам через широко используемые протоколы, такие как HTTP. Это сокращает время разработки и упрощает интеграцию с другими приложениями.

Кроме того, RESTful API обеспечивает хорошую гибкость и масштабируемость благодаря использованию URL-адресов и HTTP-методов, которые могут быть легко изменены и адаптированы к меняющимся требованиям проекта. Он также обеспечивает более простое управление состоянием клиента и сервера, что упрощает отладку и разработку.

В целом, RESTful API является широко используемым и эффективным подходом к созданию API для веб-приложений и мобильных приложений, и он стал стандартом в индустрии разработки веб-сервисов.

Некоторые примеры представлены на рисунке 3.2.

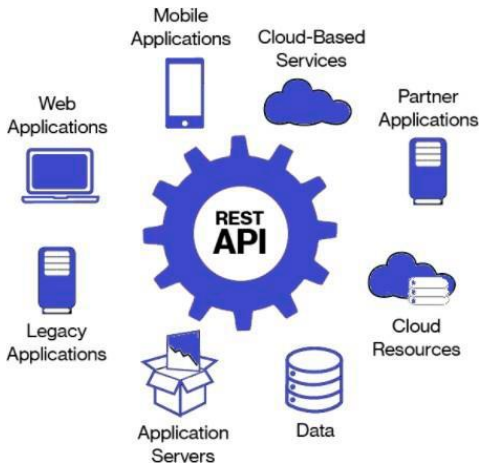


Рисунок 3.2 – Визуальное представление REST

Методы HTTP GET, POST, PUT и DELETE используются в RESTful API для выполнения операций CRUD (Create, Read, Update, Delete) с ресурсами.

Метод GET используется для получения информации о ресурсе. Он не меняет данные на сервере и используется только для чтения информации.

Метод POST используется для создания нового ресурса на сервере с помощью отправки данных клиентом.

Метод PUT используется для обновления информации о ресурсе на сервере. Он может использоваться для создания нового ресурса, если на сервере нет такого ресурса.

Метод DELETE используется для удаления ресурса на сервере.

Вместе с этими основными методами HTTP, RESTful API также может использовать другие методы HTTP, такие как PATCH для частичного обновления ресурса или HEAD для получения метаданных о ресурсе.

Каждый из методов HTTP в RESTful API представляет определенную операцию, связанную с CRUD, и используется для работы с ресурсами на сервере. Как правило, клиенты RESTful API используют определенный метод HTTP для выполнения нужной операции с ресурсом.

Данный чат-бот содержит две основные компоненты. Пользовательский интерфейс, который предоставляет пользователям информацию о прохождении модели машинного обучения, а также предоставляет возможность классификации пациентов и определения наличия или отсутствия послеоперационных осложнений.

У системы Телеграмм бота имеется также модуль по созданию модели машинного обучения, в который входят функции по подготовке данных: заполнение пропусков, оценка статистических показателей, отбор информативных показателей, исследования различных методов машинного обучения и оценка их точности.

В данном случае была построена архитектура двух модулей по отдельности:

– ориентированный на пользователя модуль,

– модуль создания модели машинного обучения.

3.3 Интерфейс Телеграмм бота

Интерфейс Телеграм-бота представляет собой взаимодействие пользователя с ботом через приложение Telegram. Это может быть как текстовый чат, так и использование inline кнопок, клавиатур, аудио- и видеоплееров, изображений и других элементов интерфейса.

Пользователь может отправлять команды боту, вводить текстовые сообщения или использовать inline кнопки для взаимодействия с ботом. Бот, в свою очередь, может отвечать текстовыми сообщениями, отправлять изображения и другие медиа-файлы, предоставлять ссылки на веб-ресурсы и выполнять другие операции.

Интерфейс Телеграм-бота может быть настроен и доработан в соответствии с конкретными потребностями пользователей и типом задач, которые бот должен решать. Например, для бота магазина онлайн-торговли интерфейс может включать каталог товаров и процесс оформления заказа, а для бота погоды - информацию о погоде в введенном пользователем городе и предоставление рекомендаций по одежде.

Телеграм-боты используются в различных сферах, включая бизнес, образование, развлечения, здравоохранение и др. Их интерфейс обычно разрабатывается с учетом специфики деятельности и потребностей пользователей, чтобы обеспечить максимально простое, удобное и эффективное взаимодействие с ботом.

Одним из преимуществ использования Телеграм-бота является то, что пользователь может взаимодействовать с ботом в любое время и из любой точки мира, где есть доступ к интернету. Также, чат с ботом может быть анонимным, что может быть важным для некоторых пользователей.

Интерфейс Телеграм-бота разрабатывается с использованием Телеграм API, которое предоставляет разработчику доступ к возможностям платформы Telegram для создания ботов. API содержит различные методы для работы с сообщениями, медиа-файлами, кнопками, клавиатурами, а также возможности для анализа текста и использования искусственного интеллекта.

Одной из задач разработчика Телеграм-бота является создание удобного и дружелюбного интерфейса, чтобы пользователи могли легко и быстро получать нужную им информацию, запускать нужные команды или проводить другие операции. Разработчик также может использовать аналитику для получения информации о поведении пользователей и улучшения интерфейса в соответствии с этими данными.

Таким образом, интерфейс Телеграм-бота представляет собой способ взаимодействия пользователя с ботом через приложение Telegram и может быть настроен для выполнения различных функций в зависимости от потребностей пользователей и типа задач, которые бот должен решать.

Разницу этих двух способов демонстрирует рисунок 3.3.

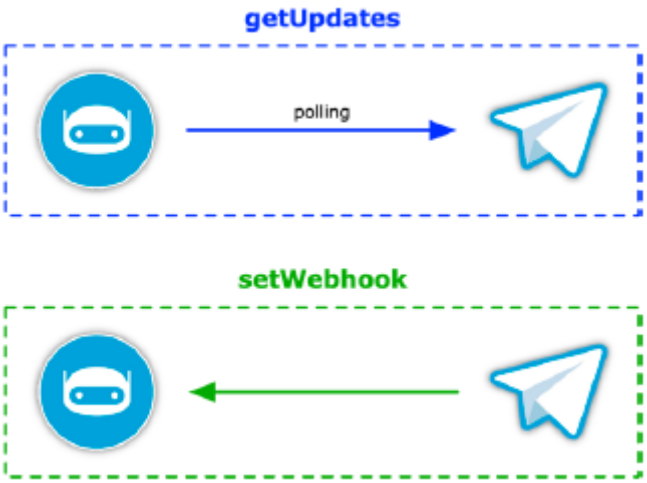


Рисунок 3.3 – Способы взаимодействия с ботом

В состав программного обеспечения может входить и веб-сервер, и Телеграм-бот. В зависимости от задач, которые должны решаться, возможна различная комбинация программных компонентов.

Веб-сервер представляет собой программное обеспечение, которое позволяет обслуживать HTTP-запросы от пользователей и отдавать им соответствующие HTTP-ответы. Это может быть веб-сайт, который пользователь открывает в браузере, или серверное приложение с API, к которому обращается клиентское приложение или другой сервер.

Телеграм-бот, в свою очередь, является приложением, разработанным для платформы Telegram, которое обеспечивает взаимодействие с пользователями через чат в Telegram. Бот может обрабатывать команды от пользователей, отправлять им текстовые сообщения, медиа-файлы, использовать inline кнопки и т.д., в зависимости от функциональности бота.

Комбинируя веб-сервер и Телеграм-бот, можно создавать различные приложения, которые взаимодействуют с пользователем как через веб-интерфейс, так и через чат в Telegram. Например, это может быть онлайн-магазин, который позволяет пользователям совершать покупки через веб-интерфейс или чат с помощью Телеграм-бота, или сервис для заказа еды, который позволяет пользователям выбирать и заказывать еду через веб-интерфейс или чат с ботом в Telegram.

Другим примером комбинации веб-сервера и Телеграм-бота может быть приложение для онлайн-опросов. Пользователи могут заполнять опросы через веб-интерфейс или чат с ботом в Telegram, а результаты опросов могут быть собраны и анализированы на сервере.

Еще одним примером может быть система управления задачами или проектами, которая позволяет пользователям создавать и редактировать задачи через веб-интерфейс или чат с ботом в Telegram. Результаты управления задачами могут быть сохранены и обработаны на сервере.

Кроме того, комбинация веб-сервера и Телеграм-бота может быть использована в различных автоматизированных системах, таких как система мониторинга серверов, система отслеживания заказов, система управления контентом.

В целом, комбинация веб-сервера и Телеграм-бота позволяет обеспечить многоканальный доступ для пользователя, что делает приложения более удобными и гибкими. Каждый пользователь может выбрать наиболее удобный способ взаимодействия с приложением, будь это веб-интерфейс или чат в Telegram.

4 Разработка программного продукта

В данной главе будет рассмотрен весь процесс разработки программного продукта и выявлены возникшие проблемы и методы их решения.

4.1. Выбранные технологии разработки

4.1.1 Анализ платформы для разработки

Существует множество языков программирования, каждый из которых предназначен для решения определенных задач и имеет свой собственный набор операторов и синтаксис. Создание приложений с использованием искусственного интеллекта требует специальных технологий и навыков, а также глубоких исследований. Для успешной реализации идей связанных с применением ИИ, необходим язык программирования с богатым инструментарием и гибкостью. Python является именно таким языком, поэтому его широко используют при разработке проектов, связанных с искусственным интеллектом.

Python является отличным выбором для проектов, связанных с машинным обучением и искусственным интеллектом, так как он обладает рядом характеристик, которые помогают разработчикам работать над проектом с высокой производительностью и уверенностью на всех этапах, от разработки до поддержки. Python имеет простой и логичный синтаксис, гибкость и мультиплатформенность. Его библиотеки и фреймворки для машинного обучения и работы с искусственным интеллектом отличны, а сообщество разработчиков вокруг Python довольно обширное. Python является одним из самых популярных языков программирования в мире не только в области ИИ и машинного обучения.

Разработка алгоритмов машинного обучения и искусственного интеллекта является сложной задачей, которая требует много времени. Для того, чтобы разработчикам было легче находить оптимальные способы решения задач, им нужна надежная и хорошо структурированная среда разработки.

Python также предоставляет практически неограниченные возможности для создания пользовательских сценариев и автоматизации процессов. Это, в сочетании с возможностью быстрого прототипирования и тестирования, делает Python идеальным выбором для быстрого прототипирования и дальнейшей разработки ИИ-приложений и моделей машинного обучения.

Наконец, Python имеет богатую и широко используемую экосистему, которая включает в себя всевозможные инструменты, библиотеки и фреймворки, используемые для разработки искусственного интеллекта. Это позволяет разработчикам выбирать наиболее подходящие инструменты для конкретной задачи и использовать их в своей работе.

Таким образом, Python является надежным и эффективным средством для разработки ИИ-приложений и моделей машинного обучения благодаря своей простоте, гибкости и многочисленным возможностям, а также благодаря своей богатой экосистеме.

Python предоставляет обширный набор программных библиотек для решения распространенных задач в области искусственного интеллекта и машинного обучения благодаря его богатому стеку технологий. В этих библиотеках содержится заранее написанный код для решения различных задач. Некоторые из библиотек, используемых в области искусственного интеллекта и машинного обучения в Python, включают в себя:

- NumPy. Библиотека для работы с многомерными массивами и матрицами, которая предоставляет широкий набор математических функций для работы с массивами и эффективные алгоритмы для обработки больших объемов данных,

- Pandas. Библиотека для работы с данными, которая предоставляет инструменты для анализа и обработки табличных данных, которая позволяет легко загружать данные из различных источников, обрабатывать их и представлять в нужном виде,

- Matplotlib. Библиотека для визуализации данных, которая предоставляет широкий набор инструментов для создания различных

графиков и диаграмм, которая позволяет визуализировать данные в любом формате и стиле.

- Scikit-learn. Библиотека для машинного обучения, которая предоставляет инструменты для создания моделей машинного обучения, их обучения и тестирования, которая включает в себя широкий набор алгоритмов для классификации, регрессии, кластеризации и других задач машинного обучения.

- TensorFlow и PyTorch. Библиотеки для создания нейронных сетей и глубокого обучения, которые предоставляют различные инструменты для создания, обучения и тестирования нейронных сетей всех типов и сложности.

Это лишь некоторые из наиболее популярных библиотек для искусственного интеллекта и машинного обучения в Python. Все они предоставляют различные инструменты и функции для решения различных задач в этой области.

Сфера разработки искусственного интеллекта находится в процессе активного развития, и есть несколько языков программирования, которые имеют доминирующее положение в этой области. Некоторые языки программирования предоставляют отличные экосистемы для создания проектов с использованием искусственного интеллекта и машинного обучения. Среди них можно выделить:

Язык программирования R обычно используется для анализа и обработки данных с целью статистического анализа. Он также имеет некоторые пакеты и библиотеки для создания проектов, связанных с машинным обучением. Например, некоторые пакеты R, такие как Gmodels, Class, Tm и RODBC, предоставляют инструменты для работы с данными и создания моделей машинного обучения. Эти пакеты могут использоваться для создания проектов в области машинного обучения на языке R.

Некоторые другие пакеты R, которые можно использовать для создания проектов машинного обучения, включают в себя:

- caret. Это пакет R, который предоставляет стандартизованный интерфейс для работы с различными алгоритмами машинного обучения в R,

- H2O. Это открытая платформа машинного обучения, которая может использоваться как из R, так и из Python, которые предоставляет широкий набор алгоритмов машинного обучения и возможности для создания и проверки моделей,

- mxnet. Это фреймворк машинного обучения, который можно использовать как на R, так и на Python. Он предназначен для работы с большими объемами данных и поддерживает распределенное обучение и инференс,

- tensorflow. Это библиотека для машинного обучения, которую можно использовать как на R, так и на Python. Она предоставляет широкий выбор алгоритмов машинного обучения и инструменты для создания и обучения нейронных сетей.

Это только некоторые из пакетов R, которые можно использовать для работы с машинным обучением и создания проектов в этой области. Развитие машинного обучения и искусственного интеллекта все еще продолжается, и новые инструменты и библиотеки появляются постоянно.

R и Python — это два широко используемых языка программирования в области анализа данных, машинного обучения и искусственного интеллекта. Ни один язык не является идеальным для всех сценариев, а выбор конкретного языка может зависеть от требований проекта и предпочтений разработчика.

Несмотря на то, что R изначально был разработан для статистического анализа данных, на нём также можно реализовать машинное обучение. R обладает богатым набором библиотек и пакетов, которые упрощают работу с данными, и многие из них предоставляют широкий выбор алгоритмов машинного обучения.

Python также имеет большое сообщество разработчиков и богатую экосистему библиотек, которая делает его привлекательным для использования в области машинного обучения и искусственного интеллекта.

В особенности, в Python есть глубоко распространенные библиотеки для работы с данными, такие как NumPy, Pandas, Matplotlib и Scikit-learn, которые предоставляют различные инструменты для обработки и визуализации данных.

В целом, решение о том, какой язык использовать, может зависеть от опыта разработчика, доступности ресурсов, требований конкретного проекта и свойств данных, с которыми нужно работать. В любом случае, как R, так и Python предоставляют широкий набор инструментов для работы с данными и создания проектов в области машинного обучения и искусственного интеллекта.

Scala — это мультипарадигменный язык программирования, который был создан в 2003 году, и он сочетает в себе функциональное и объектно-ориентированное программирование. Язык Scala запущен на платформе Java и полностью совместим с Java. Он основан на принципе "одного языка для всего", что позволяет легко интегрировать его с существующими приложениями, написанными на Java. Разработчики могут использовать Java-фреймворки и библиотеки в Scala, а также использовать существующий Java-код в Scala-приложениях.

Scala имеет сильную статическую типизацию, что помогает обнаруживать ошибки на ранних этапах разработки приложений. Он также имеет мощный систему неизменяемых коллекций, которая позволяет создавать безопасный и эффективный код. Scala предоставляет удобный синтаксис для решения многих проблем, чтобы написание модульных и тестируемых приложений было легче и быстрее. Scala также имеет расширяемый тип-систему, которая позволяет создавать собственные типы данных и алгоритмы.

Scala часто используется для создания крупных, высокопроизводительных приложений, таких как фреймворки Akka и Play, а также фреймворк машинного обучения Apache Spark. Он подходит для

создания приложений, которые требуют высокой степени параллелизма, таких как рабочие процессы больших данных и веб-распределенные сервисы.

Scala — это язык, который предоставляет разработчикам большую свободу и мощь в создании высококачественного кода. Он имеет широкое сообщество разработчиков и готовых библиотек, которые могут помочь в создании качественных приложений.

Если нужна простота и быстрота разработки ML, то рекомендуется использовать Python. Это язык имеет глубоко распространенные библиотеки для машинного обучения, такие как Scikit-learn, TensorFlow, Keras и PyTorch, и широкую экосистему инструментов, которые позволяют упрощать работу с данными и визуализацией, а также создавать модели машинного обучения.

Проанализировав все недостатки и преимущества выбор, останавливается на языке программирования Python.

4.1.2 Телеграмм Bot API

Telegram Bot API, также известный как Telegram Bot Platform, - это инструмент для создания ботов для Telegram, которые могут коммуницировать с пользователями через мессенджер Telegram. Bot API предоставляет разработчикам мощный и гибкий фреймворк для создания ботов, включая возможность отправки сообщений, получения обновлений, управления разговорами.

Bot API включает в себя следующие основные функции:

- отправка и получение сообщений. Бот может отправлять текстовые, аудио, видео, фото и другие типы сообщений пользователям, а также получать сообщения от пользователей,

- Inline-режим. Бот может быть настроен на обработку запросов, которые пользователи отправляют через inline-режим,

- работа с клавиатурами. Боты могут использовать клавиатуры, чтобы помочь пользователям быстрее взаимодействовать с ними,

- управление пользователями. Бот может получать информацию о пользователях, подписывать и отписываться от каналов и групп, отправлять сообщения в групповые чаты и многое другое,

- работа с медиафайлами. Боты могут использовать Prism для конвертирования и сжатия файлов, а также для получения файлов из сообщений, которые отправляют пользователи.

Telegram Bot API предоставляет разработчикам широкие возможности для создания удобных и функциональных ботов для Telegram. Он также является доступным и простым в использовании, что позволяет разработчикам с легкостью начать создание своих ботов.

Для использования Telegram Bot API необходимо зарегистрировать бота в Telegram и получить API-ключ. Затем можно использовать HTTP-запросы для отправки и получения обновлений через Telegram Bot API.

Основные API-методы, которые могут быть использованы для создания бота в Telegram, включают следующие:

- sendMessage. Отправляет сообщение пользователю,

- getUpdates. Получает список обновлений для бота,

- sendPhoto/sendVideo/sendDocument/sendAudio. Отправляет фото, видео, документы и аудио пользователю,

- inlineQuery. Обработка запросов через inline-режим,

- editMessageText/editMessageReplyMarkup. Редактирует текст сообщения и клавиатуру,

- answerCallbackQuery. Отправляет ответ на нажатие кнопки в клавиатуре,

- getChat. Получает информацию о чате, в котором происходит общение.

Эти методы могут быть использованы для создания бота, который может отправлять сообщения, отправлять файлы, использовать встроенные функции, управлять клавиатурой и осуществлять многие другие функции.

Telegram Bot API также поддерживает Webhooks, которые могут использоваться для автоматического получения обновлений бота и создания более быстрого и надежного опыта работы с ботами.

В целом, Telegram Bot API является мощным инструментом для создания ботов для Telegram. Это позволяет разработчикам создавать удобные и функциональные боты, которые могут использоваться для широкого спектра задач, от простых сообщений до более сложных приложений, использующих машинное обучение и искусственный интеллект.

4.1.3 Среда разработки

Google Colaboratory, более известная как «Colab», — бесплатная платформа для блокнотов Jupyter. Кроме среды запуска блокнотов Python и R Colab позволяет совместно использовать свободный доступ к ограниченному количеству GPU и TPU.

Colaboratory, или просто Colab, позволяет писать и выполнять код Python в браузере. При этом:

- не требуется никакой настройки,
- бесплатный доступ к графическим процессорам,
- предоставлять доступ к документам другим людям очень просто.

Благодаря блокнотам Colab можно использовать в одном документе исполняемый код, форматированный текст, изображения, разметку HTML, набор LaTeX и не только. Блокноты Colab храниться на Google Диске.

Блокноты Colab – это блокноты Jupyter, которые размещены в сервисе Colab. Colab позволяет использовать для анализа и визуализации данных все возможности популярных библиотек Python. Например, в Colab используется библиотека numpy для генерации случайных данных, а также библиотека matplotlib для их визуализации.

В Colab можете импортировать набор данных изображения, сориентировать на него классификатор изображений и оценить модель с помощью нескольких строк кода. Код в блокнотах Colab исполняется на облачных серверах Google. Это означает, что можно использовать аппаратное

обеспечение Google, в том числе графические процессоры и TPU, независимо от мощности локальной машины. Нужен только браузер.

Colab активно используется в области машинного обучения, в том числе для:

- знакомства с TensorFlow,
- разработки и обучения нейронных сетей,
- экспериментов с TPU,
- распространения исследований в области ИИ,
- создания руководств.

4.2 Регистрация чат-бота Телеграмм

Для регистрации чат-бота Telegram нужно:

- найте в Telegram бота @BotFather и отправьте ему сообщение "/start",
- отправить ему новое сообщение с текстом "/newbot" и следуйте инструкциям,
- выбрать имя для вашего бота, заканчивающееся на "bot". Например, "MyAwesomeBot",
- выбрать уникальное имя пользователя для вашего бота, заканчивающееся на "bot". Например, "MyAwesomeBot123_bot",
- скопировать токен, который BotFather выдал вам после успешной регистрации вашего бота,

После регистрации вашего чат-бота вы можете использовать Telegram Bot API для создания и управления вашим ботом. Вам будет необходимо написать скрипт или приложение, которое будет использовать API вашего бота для отправки и получения сообщений, обрабатывать запросы.

Для использования Telegram Bot API необходимо выбрать язык программирования, который вы хотите использовать, и библиотеку для доступа к API. Существует множество библиотек для разных языков программирования, в том числе Python, JavaScript, Ruby, PHP, Java и другие.

Пример регистрации чат-бота представлен на рисунке 4.1.

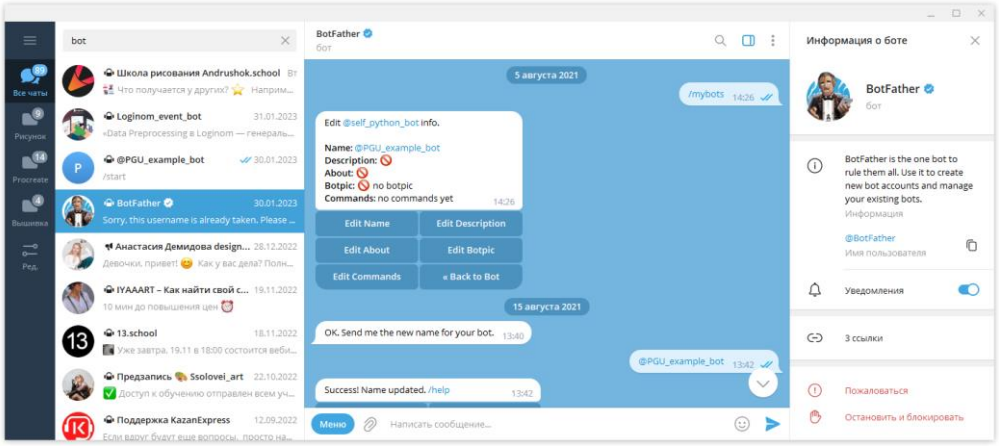


Рисунок 4.1 – Пример регистрации чат-бота

Если вы используете Python, то можете использовать библиотеку python-telegram-bot, которая является популярным и простым в использовании инструментом для создания чат-ботов Telegram на этом языке. Библиотека обладает всем необходимым функционалом, чтобы создать полноценного чат-бота Telegram.

Чтобы определить дополнительные параметры для вашего чат-бота, такие как иконка, приветственное сообщение, описание, а также удаление уже существующих чат-ботов, можно использовать следующие команды (см. рисунок 4.2).

Команда	Описание
/setname	Изменяет существующее имя
/setdescription	Присваивает текст, который будет отображаться при первом открытии чат-бота
/setabouttext	Присваивает текст в поле «О чат-боте»
/setuserpic	Присваивает выбранную картинку
/setcommands	Позволяет команд создать список доступных
/deletebot	Удаляет выбранного чат-бота

Рисунок 4.2 – Доступные команды для изменения чат-ботов

Кроме команд, которые позволяют изменять основные параметры чат-бота, существует ряд команд, которые выводят неизменяемые параметры (такие как токен) и позволяют присваивать значения другим параметрам, как показано на рисунке 4.3.

Команда	Описание
/token	Возвращает полученный ранее токену выбранного бота
/revoke	Аннулирует токен доступа к боту
/setinline	Включает или выключает возможность вызывать бота из других чатов
/setinlinegeo	Включает или выключает возможность передавать местоположение бота из другого чата
/setinlinefeedback	Позволяет получать информацию о количестве выбранных пользователями команд
/setjoingroup	Определяет может ли быть добавлен в групповые диалоги бот
/setprivacy	Включает режим конфиденциальности. В этом режиме бот получает, обрабатывает и отсылает обратно информацию отдельно для каждого пользователя в чате

Рисунок 4.3 – Доступные команды для дополнительной настройки чат-бота

Получив токен и настроив параметры на стороне Telegram, можно приступать к разработке программной части вашего чат-бота.

Токен – это метка, которая идентифицирует вашего чат-бота в системе Telegram и полезна для взаимодействия вашего скрипта с сервером Telegram по API. Чтобы использовать функции API вашего бота, необходимо передавать в вызов API соответствующие данные параметров, включая токен бота, как один из параметров. Поэтому токен является важным параметром, которым вы должны пользоваться при создании своего чат-бота.

4.2.1 Реализация клиентской части чат-бота

Для разработки программной части чат-бота можно использовать различные языки программирования, такие как Python, JavaScript, PHP и другие, а также соответствующие библиотеки для доступа к Telegram API.

Например, если вы используете Python, вы можете использовать библиотеку python-telegram-bot, которая облегчает создание чат-ботов Telegram.

Для начала работы нужно создать экземпляр бота с помощью токена, который получили при регистрации от Telegram. Для реализации поставленных целей потребуется четыре различные команды:

– загрузка выборки,

– отбор параметров,

– случайный лес,

– предсказание.

На Рисунке 4.4 представлено Меню чат-бота.

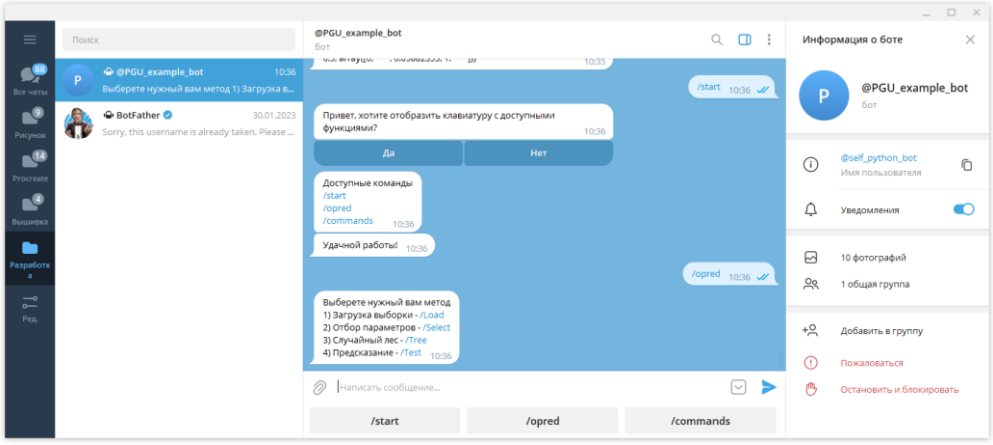


Рисунок 4.4 – Меню чат-бота

В мессенджере Telegram клавиатура представлена в виде объекта, а ее кнопки - в виде JSON строки. Документация Telegram Bot API утверждает, что каждая кнопка на клавиатуре должна содержать обязательный параметр - текст (text), а также может содержать до шести необязательных параметров: ссылку (url), данные обратного вызова (callback_data), возможность встроенного поиска (switch_inline_query), возможность вывода клавиатуры в текущем чате (switch_inline_query_current_chat), игровое описание кнопки (callback_game) и возможность покупки (pay). Пример реализации клавиатуры одного из меню чат-бота изображен на рисунке 4.5.

```
@bot.message_handler(commands=['start'])
def start(message):
    markup = types.InlineKeyboardMarkup()
    button1 = types.InlineKeyboardButton(text="Да", callback_data = 'yes')
    button2 = types.InlineKeyboardButton(text="Нет", callback_data = 'no')
    markup.add(button1, button2)
    bot.send_message(message.chat.id, 'Привет, хотите отобразить клавиатуру с доступными функциями?', reply_markup = markup)
```

Рисунок 4.5 – Реализация клавиатуры чат-бота

Затем мы перешли к реализации основного меню, которое позволяет пользователям выбрать конкретный этап работы с выборкой пациентов, который им необходим.

При выборе команды «Загрузка выборки - /Load» происходит загрузка выборки о пациентах с мочекаменной болезнью из репозитория. Пример реализации представлен на рисунке 4.6.

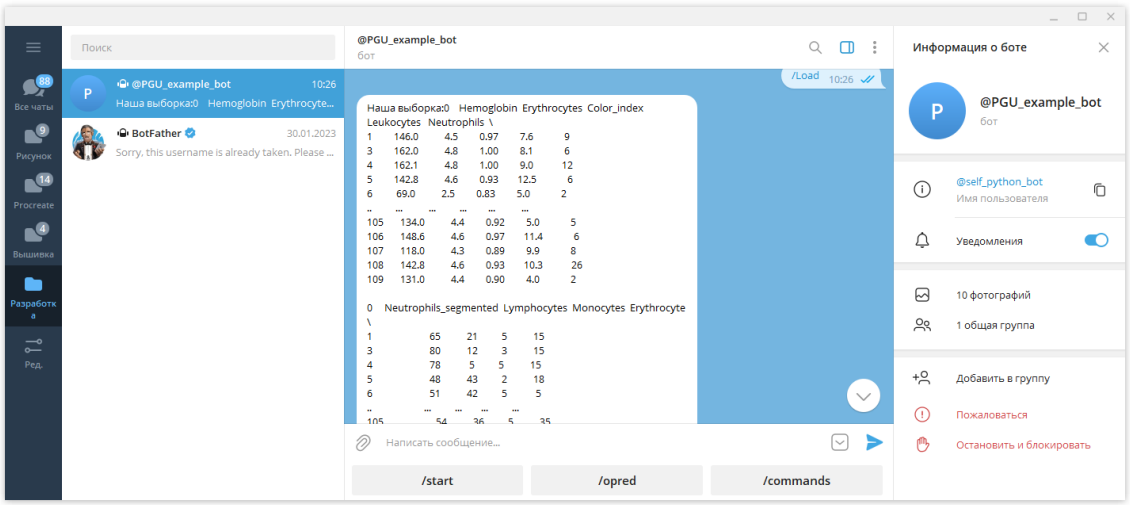


Рисунок 4.6 – Команды «Загрузка выборки - /Load» чат-бота

При выборе команды «Отбор параметров - /Select» происходит анализ выборки и с использованием метода KBest были отобраны пять информативных показателей и сформирована выборка для обучения и тестирования модели машинного обучения. Пример реализации представлен на рисунке 4.7.

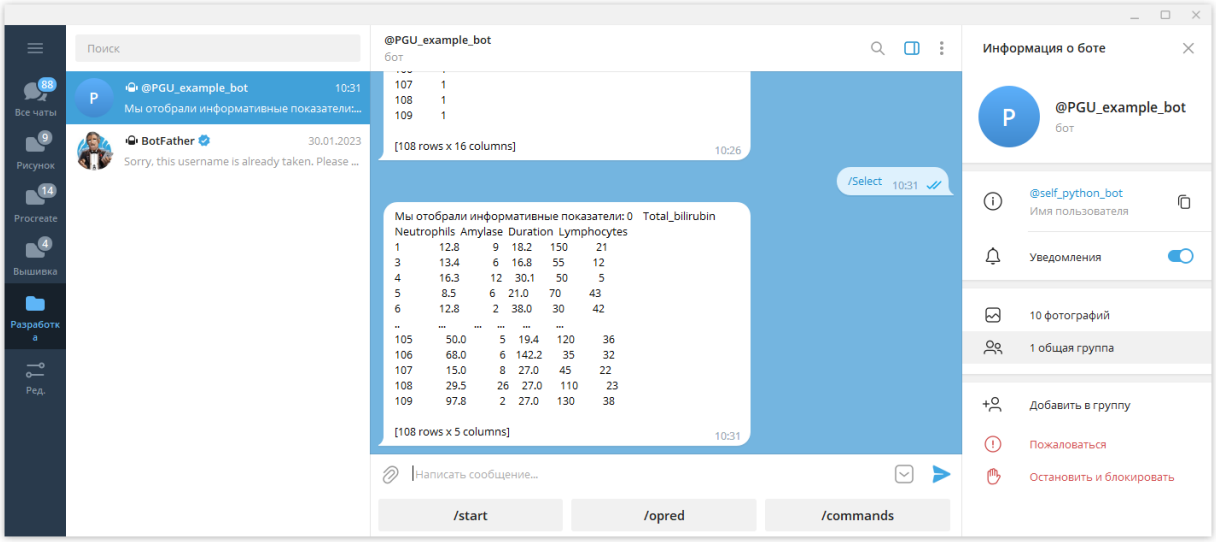


Рисунок 4.7 – Команды «Отбор параметров - /Select» чат-бота

При выборе команды «Случайный лес - /Tree» обучаем модель с использованием алгоритма Случайный лес, который показал наименьшую

ошибку при проведении экспериментов. Пример реализации представлен на рисунке 4.8.

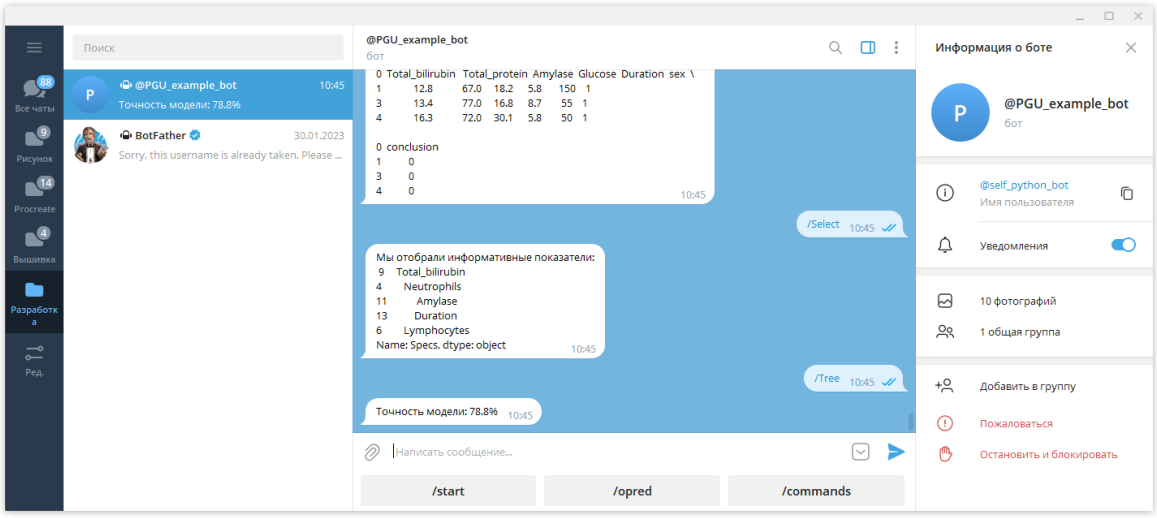


Рисунок 4.8 – Команды «Случайный лес - /Tree» чат-бота

При выборе команды «Предсказание - /Test» проводим классификацию пациентов уже на настроенной модели. Для определения наличия или отсутствия осложнений после операции необходимо ввести значения пяти показателей, которые были отобраны на предыдущих этапах. Пример реализации представлен на рисунке 4.9.

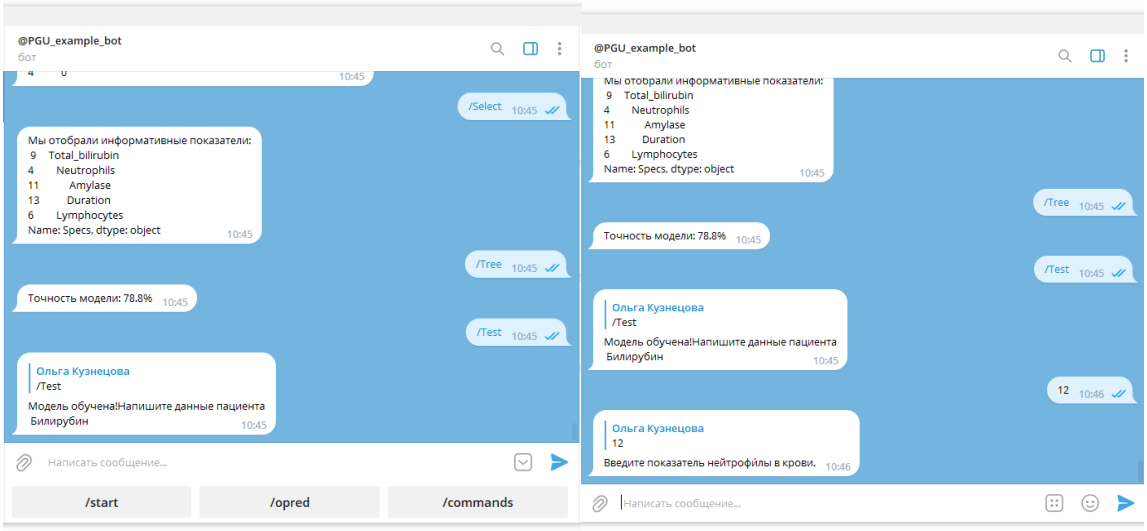


Рисунок 4.9 – Команды «Предсказание - /Test» чат-бота

На первом шаге ввели значение показателей Билирубин и Нейрофилы и Амилаза в крови. Пример реализации представлен на рисунке 4.10.

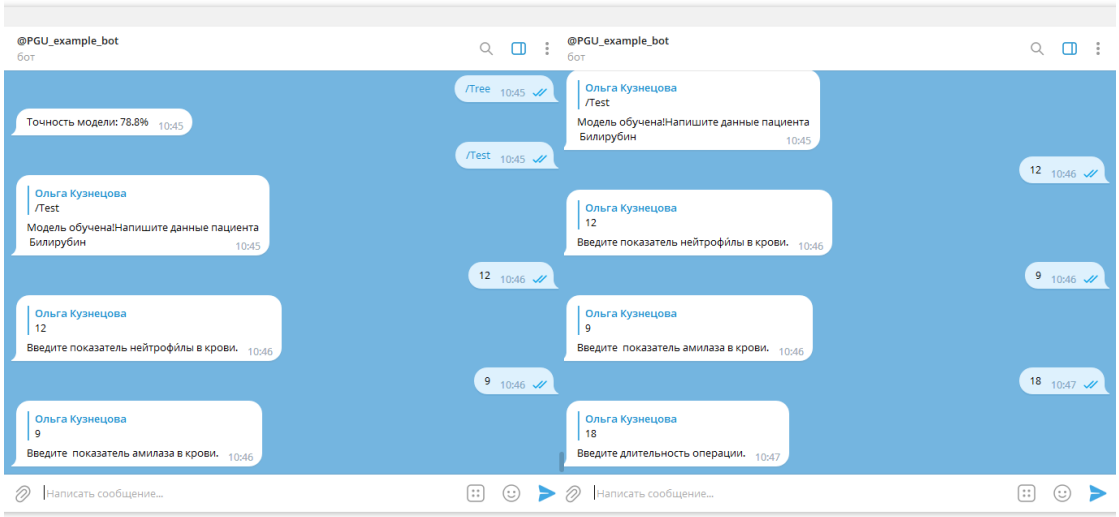


Рисунок 4.10 – Команды «Предсказание - /Test» чат-бота

На следующем шаге ввели значение показателей Лимфоциты в крови и Длительность операции. Пример реализации представлен на рисунке 4.11.

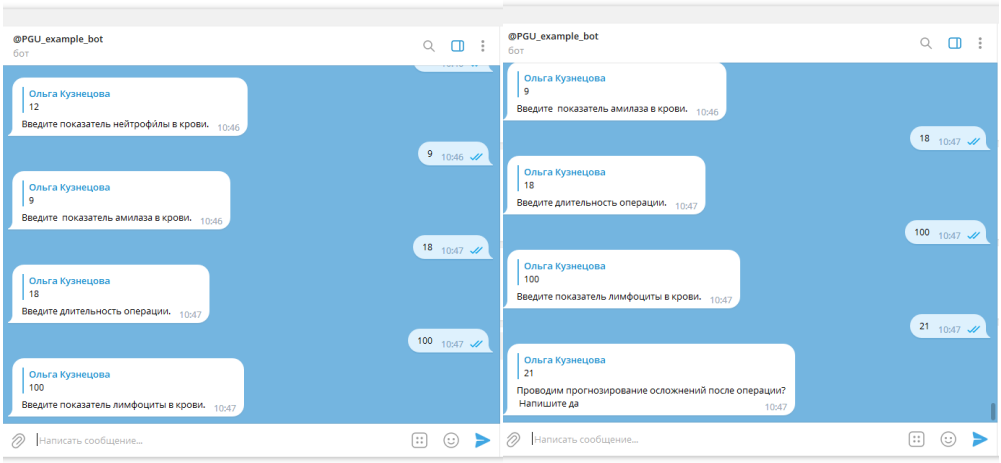


Рисунок 4.11 – Команды «Предсказание - /Test» чат-бота

На заключительном шаге чат-бот просит подтвердить введенные ранее данные и выводит прогноз. Пример реализации представлен на рисунке 4.12.

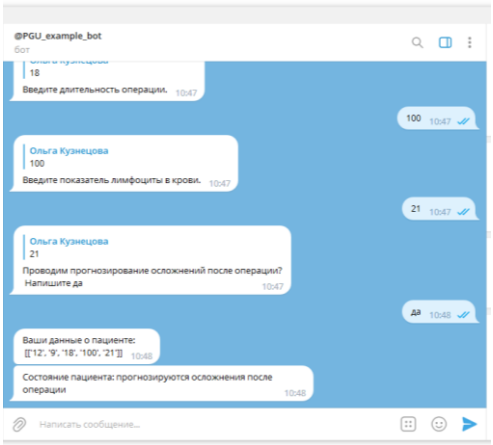


Рисунок 4.12 – Команды «Предсказание - /Test» чат-бота

На последнем этапе работы над чат-ботом пользователь заполняет необходимые данные, после чего бот запрашивает подтверждение введенной информации перед расчетом прогноза. Затем бот выводит прогноз на основе предоставленных данных и может предложить дополнительные рекомендации или действия пользователю. В зависимости от задач чат-бота этот шаг может быть изменен или дополнен другими функциональными возможностями.

Для применения чат-бота в медицинских целях, на последнем этапе бот может отправить информацию о консультанте, контакты клиники или организации, связанные с медициной, к дополнительному ознакомлению пользователей. Каждая задача и сфера применения имеют свою специфику и процесс работы над чат-ботом должен быть подстроен под нее.

ЗАКЛЮЧЕНИЕ

В процессе выполнения выпускной квалификационной работы были исследованы способы подготовки данных для обучения нейронных сетей: заполнение пропусков результатов анализов, проанализированы алгоритмы выделения информативных признаков из экспериментальных данных, исследована и разработана архитектура алгоритмов обучения нейронных сетей, разработка информационной модели интеллектуальной системы поддержки принятия решения при диагностике мочекаменной болезни.

В пояснительной записке выполнены следующие разделы: введение, теоретические сведения (анализ состояния исследования мочекаменной болезни, исследование методов отбора показателей, исследование методов прогнозирования), разработка программного кода. Произведена загрузка данных, обработка данных, создание нейронной сети, её обучение, тестирование и проверка.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Змачинская, И. М. Хронический холецистит : учеб.-метод. пособие / И. М. Змачинская, Т. Т. Ко- пать, М. К. Церех. – Минск : БГМУ, 2017. – 20 с.
2. Ивашкин В.Т., Полуэктова Е.А. Клиника и диагностика функциональных запоров // Лечащий врач. 2001. - № 5. - с. 26-29.
3. Chang JH et al: Role of magnetic resonance cholangiopancreatography for choledocholithiasis: analysis of patients with negative MRCP. Scand J Gastroenterol. 47(2):217-24, 2012.
4. Малаева, Е.Г. Гастроэнтерология : учеб. пособие для студентов субординатуры по терапии / Е.Г. Малаева. — Минск : Новое знание, 2016 — 333 с. : ил.
5. Ильченко А.А. Мочекаменная болезнь. М.: Анахарсис, 2004.- 200 с.
6. Гастроэнтерология: нац. рук. / под ред. В.Т. Ивашкина, Т.Л. Лапиной. М. : ГЭОТАР-Медиа, 2008. 704 с.
7. Гончарик, Т.А. Дифференциальная диагностика желтух: учеб.-метод. пособие / Т.А. Гончарик. Минск : БГМУ, 2009. 24 с.
8. Козловский, В.И. Лекции по факультетской терапии. В 2 т. Т. 2 /В.И. Козловский; Витеб. гос. мед. ун-т. 2-е изд., доп. и перераб. Витебск : ВГМУ, 2010. 306 с.
9. Жарко В.И., Цыбин А.К., Малахова И.В. и др. // Вопросы организации и информатизации здравоохранения. — 2006.— № 4. – С. 3—7.
10. Жарко В.И. // Мед. вестник. – 2008. – № 9 (843). – С. 2.
11. Андрейчиков А.В., Андрейчикова О.Н. Интеллектуальные информационные системы. – М.: Финансы и статистика, 2006.
12. Дюк В.А., Эмануэль В.Л. Информационные технологии в медико-биологических исследованиях. – СПб.: Питер, 2003.
13. Новиков, Д. А. Статистические методы в медико-биологическом эксперименте (типовые случаи) / Д. А. Новиков, В. В. Новочадов // Волгоград: Изд-во ВолГМУ, 2005. – 84 с.

14. Реброва, О. Ю. Статистический анализ медицинских данных. Применение пакета прикладных программ STATISTICA / О. Ю. Реброва. – М.: МедиаСфера, 2002. – 125 с.

15. Дедов, А. В. Построение математических моделей предварительного диагноза заболеваний печени на основе методов регрессионного анализа / А. В Дедов, Г. А. Попов // Вестник АГТУ. Серия: Управление, вычислительная техника и информатика. – 2014. – №4. – С. 124-136.

16. Мараховский, К.Ю. Инновации диагностической эндоскопии желудочно-кишечного тракта / К.Ю. Мараховский С.В. Белявская //ArsMedica. 2009. № 4. С. 30–35.

17. Диагностика заболеваний методами теории вероятностей / М. Л. Жмудяк, А. Н. Повалихин, А. В. Стребуков и др. – Барнаул: Изд-во АлтГТУ, 2006. – 168 с.

18. Кобринский, Б. А. Телемедицина и искусственный интеллект / Б. А. Кобринский // Новости искусственного интеллекта.- 2003. – №1. – С. 15-19.

19. Барсегян А. А., Куприянов М. С., Степаненко В. В., Холод И. И. - Методы и модели анализа данных: OLAP и Data Mining. СПб.: БХВ-Петербург, 2004. - 336 с: ил. 331 с.

20. Юнкеров В.И. Математико-статистическая обработка данных медицинских исследований / В.И. Юнкеров, С.Г. Григорьев // СПб.: ВМедА, 2002. – 266 с.

21. Муратова Е.А., Берестнева О.Г. Выявление скрытых закономерностей в социально- психологических исследованиях //Известия Томского политехнического университета. – 2003. –Т. 306. – № 3. – С. 97–102.

22. Баврина А.Ю., Мясников В.В. Автоматическая классификация, Самара: Изд-во Самар, гос. аэрокосм. ун-та, 2010. — 21 с

23. Наивный байесовский классификатор. – [Электронный ресурс]. – URL: http://bazhenov.me/blog/2012/06/11/naive-bayes.html (Дата обращения 26.05.2015).

24. Гнеденко Б.В., Хинчин А.Я. Элементарное введение в теорию вероятностей. М.: Едиториал УРСС, 2003. – 208 с.

25. Холлендер М., Вулф Д.А. Непараметрические методы статистики. М.: Финансы и статистика, 1983. – 518 с.

26. Тюрин Ю.Н., Макаров А.А. Статистический анализ данных на компьютере. — М.: ИНФРА-М, 2002. — 528 с.

27. Гилл Ф., Мюррей У., Райт М. Практическая оптимизация. — М.: Мир, 1985.

28. Паклин, Н.Б. Бизнес-аналитика: от данных к знаниям / Н.Б. Паклин, В.И. Орешков. – С.-Пб.: Питер, 2013.

29. Осовский, С. Нейронные сети для обработки информации: Пер. с польского И. Д. Рудинского. / С. Осовский. – М.: Финансы и статистика, 2004. – 344 с.

30. Халафян А.А. STATISTIC A 6. Статистический анализ данных. 3-е изд. Учебник — М: ООО «Бином-Пресс», 2007 г. — 512 с.

31. Воронин Г.В. Нейросетевые технологии и вопросы идентификации в медицинских исследования / Г.В. Воронин, Е.М. Пальцева // Вестник новых медицинских технологий. – 2009. – № 1. – С. 32–34.

32. Karpathy A, Li F. Deep visual-semantic alignments for generating image descriptions. Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2015

33. Zhang J, Zong C. Deep neural networks in machine translation: An overview. IEEE Intelligent Systems. 2015;30:16–25.

34. Silver D, Huang A, Maddison CJ, Guez A, Sifre L, et al. Mastering the game of Go with deep neural networks and tree search. Nature. 2016;529:484–489. [PubMed]

35. Bishop, C. M. Pattern Recognition and Machine Learning. Springer, New York, NY, 2006.

ПРИЛОЖЕНИЕ А

Гистограммы значений лабораторных анализов больных с мочекаменной болезнью для проверки данных на соответствие нормальному закону распределения

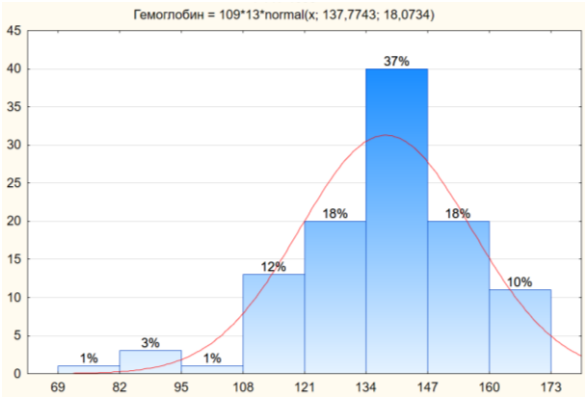


Рисунок А.1 – Гистограммы значений анализа. Гемоглобин

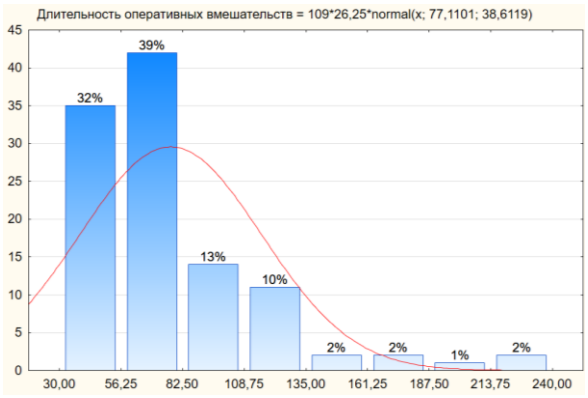


Рисунок А.2 – Гистограммы значений анализа. Длительность оперативных вмешательств

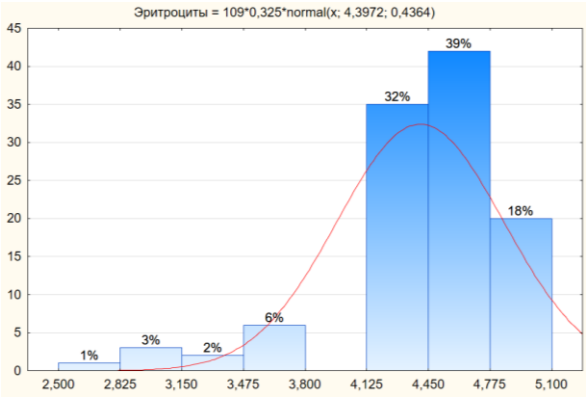


Рисунок А.3 – Гистограммы значений анализа. Эритроциты

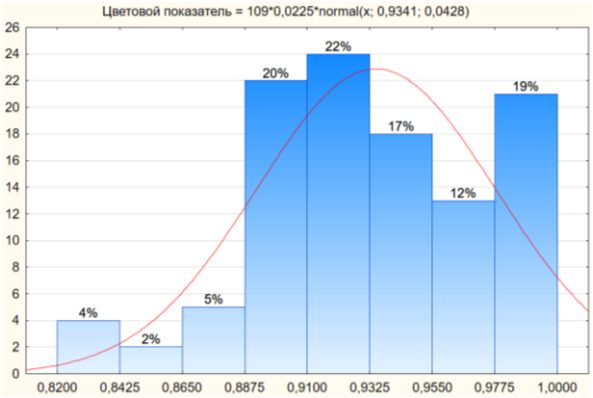


Рисунок А.4 – Гистограммы значений анализа. Цветовой показатель

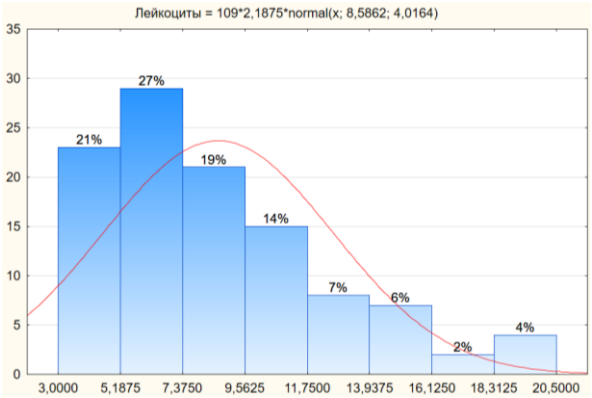


Рисунок А.5 – Гистограммы значений анализа. Лейкоциты

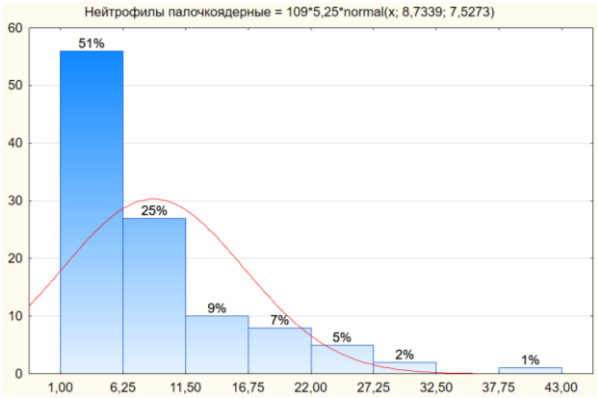


Рисунок А.6 – Гистограммы значений анализа. Нейтрофилы

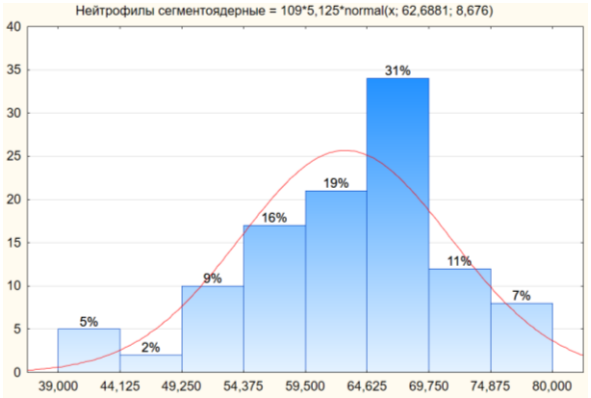


Рисунок А.7 – Гистограммы значений анализа. Нейтрофилы сегментоядерные

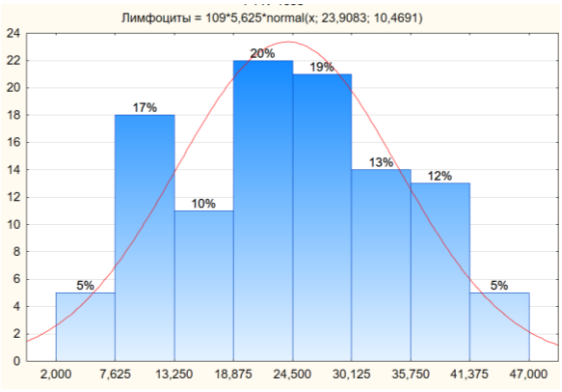


Рисунок А.8 – Гистограммы значений анализа. Лимфоциты

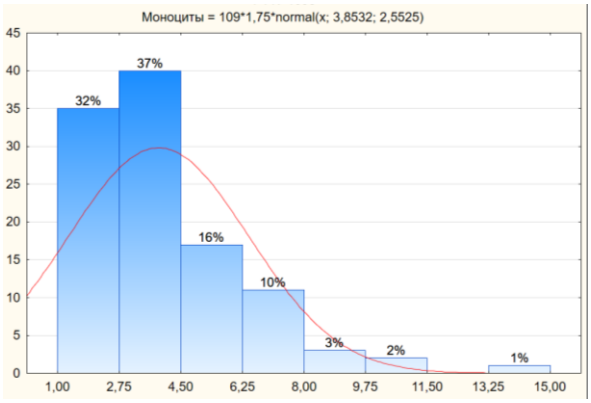


Рисунок А.9 – Гистограммы значений анализа. Моноциты

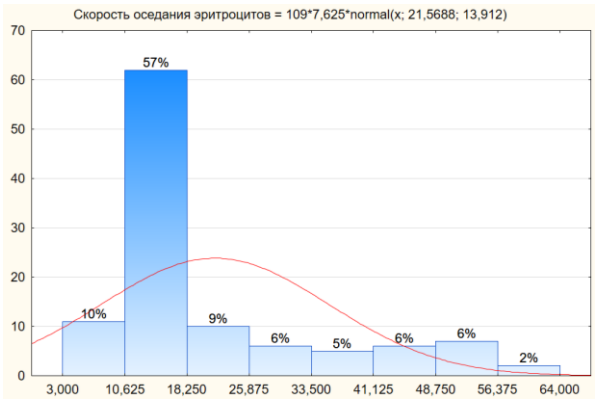


Рисунок А.10 – Гистограммы значений анализа. Скорость оседания эритроцитов

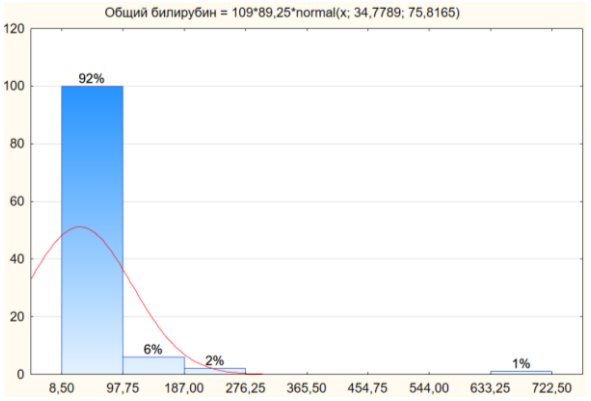


Рисунок А.11 – Гистограммы значений анализа. Общий билирубин

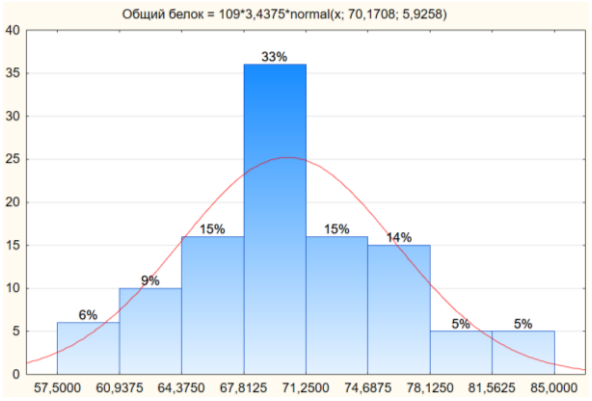


Рисунок А.12 – Гистограммы значений анализа. Общий белок

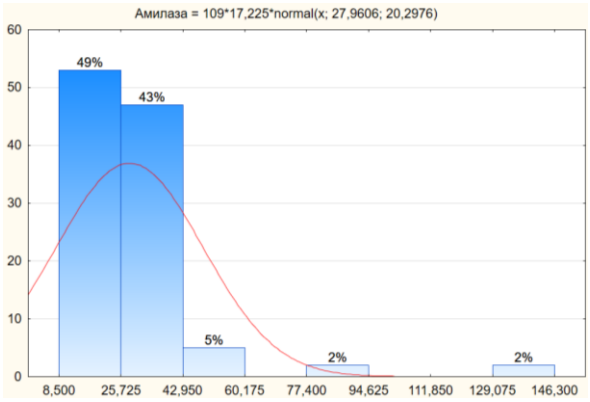


Рисунок А.13 – Гистограммы значений анализа. Амилаза

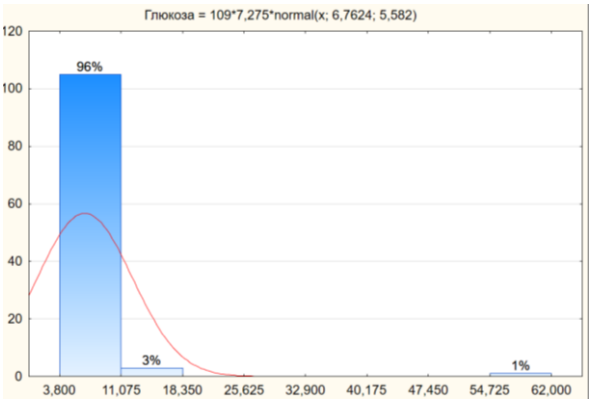


Рисунок А.14 – Гистограммы значений анализа. Глюкоза

ПРИЛОЖЕНИЕ Б

График нормального распределения вероятности

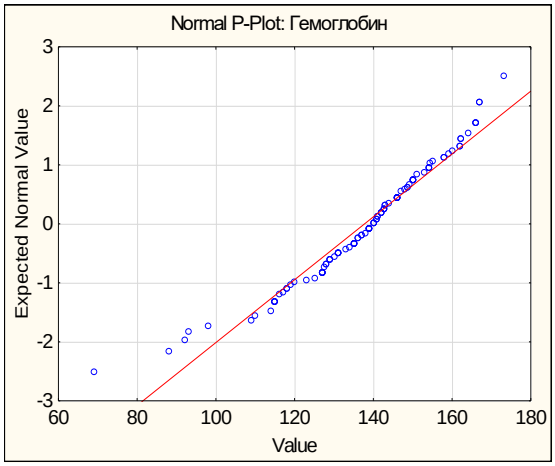


Рисунок Б.1 – График нормального распределения вероятности анализа. Гемоглобин

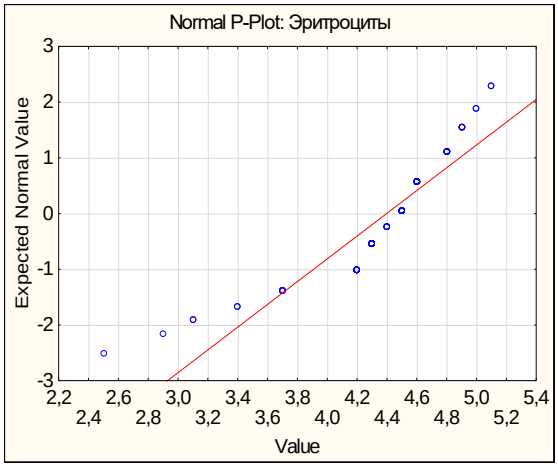


Рисунок Б.2 – График нормального распределения вероятности анализа. Эритроциты

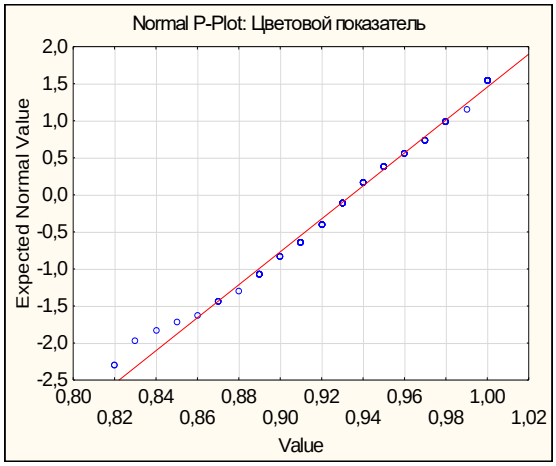


Рисунок Б.3 – График нормального распределения вероятности анализа. Цветовой показатель

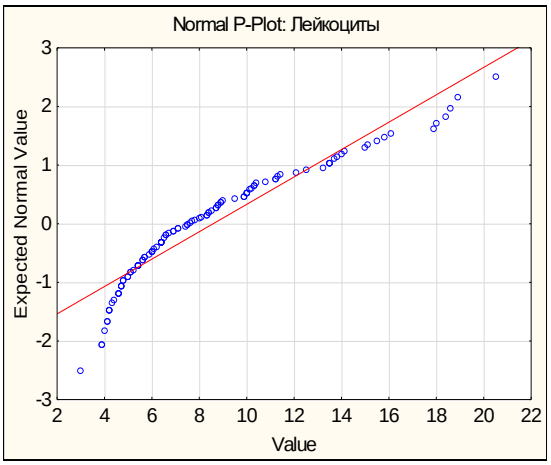


Рисунок Б.4 – График нормального распределения вероятности анализа. Лейкоциты

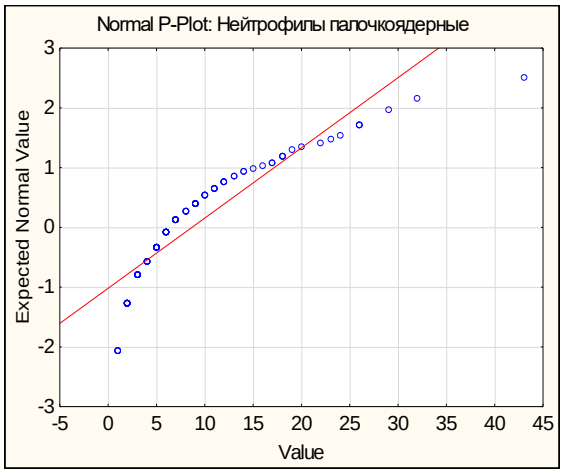


Рисунок Б.5 – График нормального распределения вероятности анализа. Нейтрофилы палочкоядерные

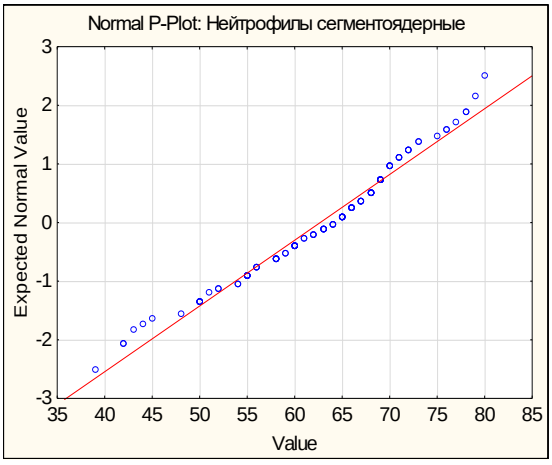


Рисунок Б.6 – График нормального распределения вероятности анализа. Нейтрофилы сегментоядерные

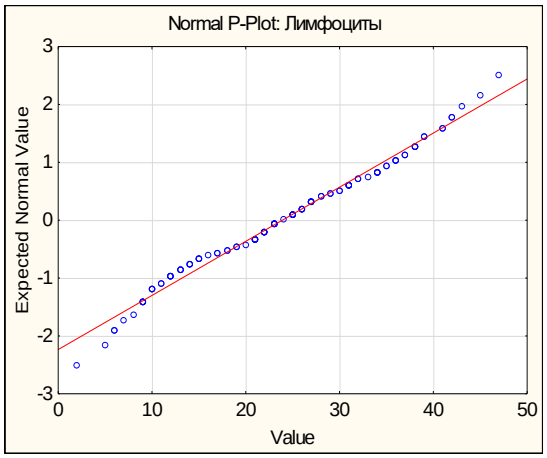


Рисунок Б.7 – График нормального распределения вероятности анализа. Лимфоциты

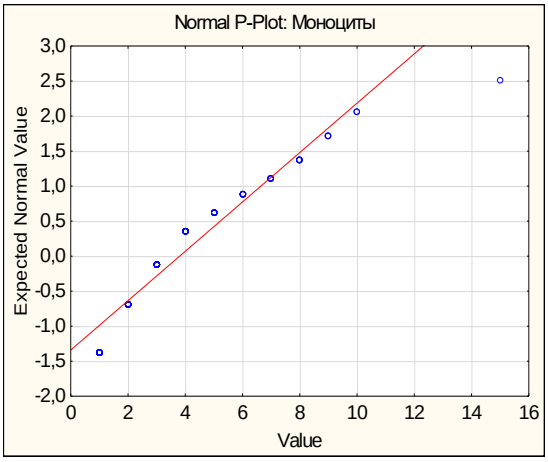


Рисунок Б.8 – График нормального распределения вероятности анализа .Моноциты

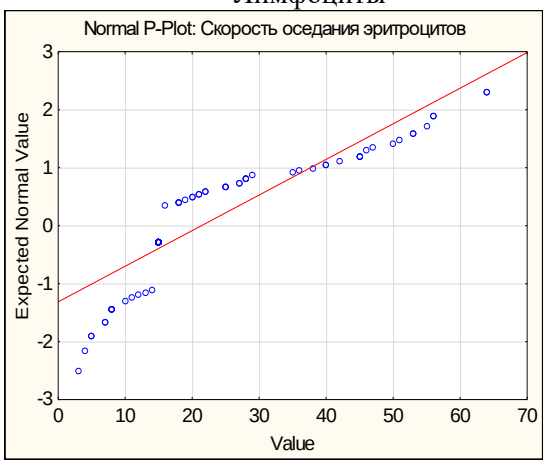


Рисунок Б.9 – График нормального распределения вероятности анализа. Скорость оседания эритроцитов

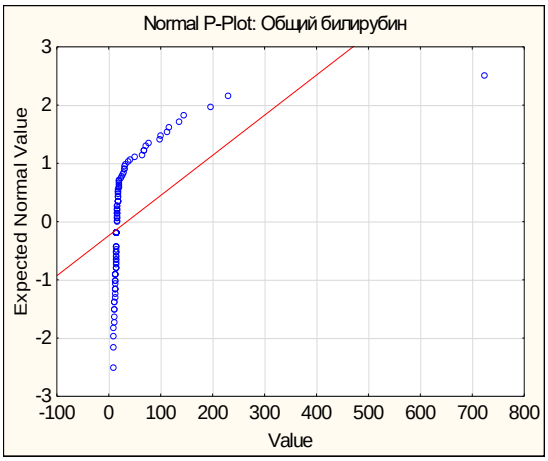


Рисунок Б.10 – График нормального распределения вероятности анализа. Общий билирубин

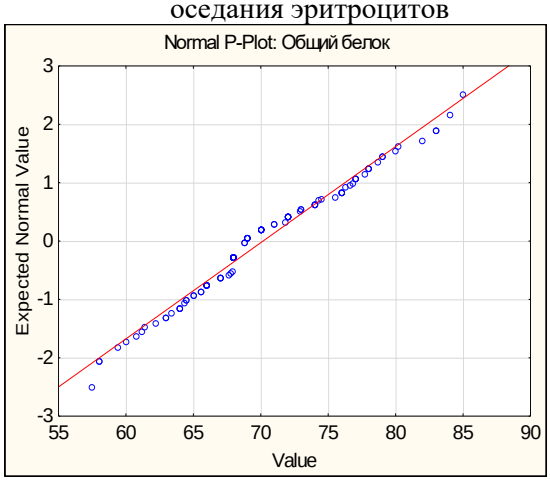


Рисунок Б.11 – График нормального распределения вероятности анализа. Общий белок

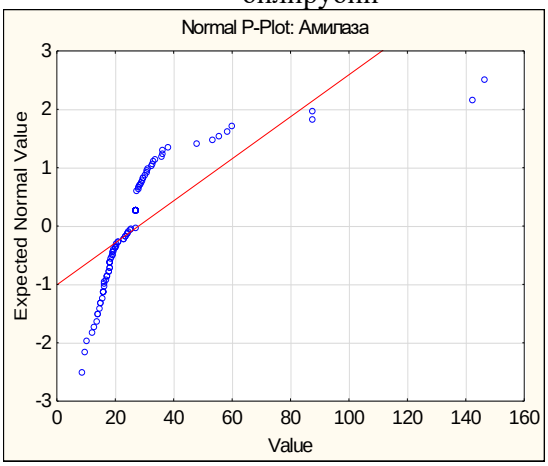


Рисунок Б.12 – График нормального распределения вероятности анализа. Амилаза

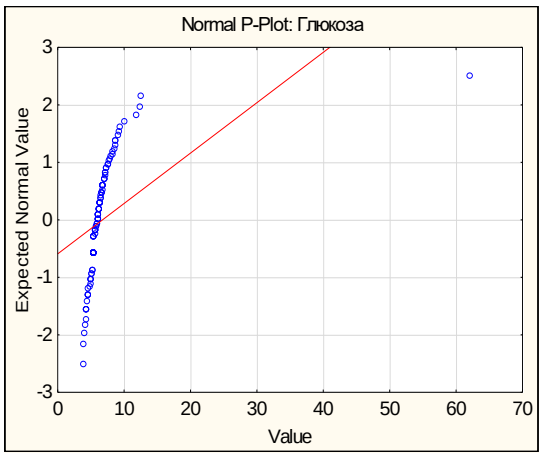


Рисунок Б.13 – График нормального распределения вероятности анализа. Глюкоза

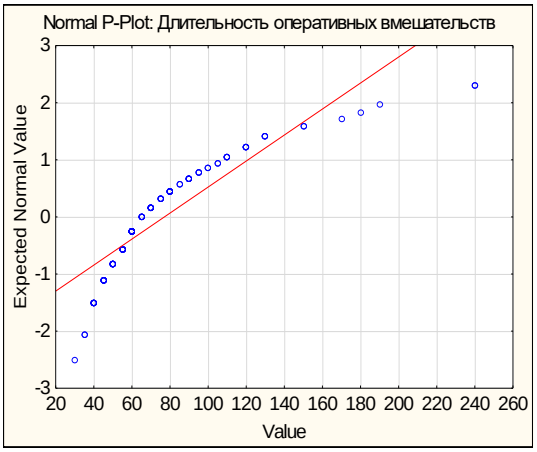


Рисунок Б.14 – График нормального распределения вероятности анализа. Длительность опреативных вмешательств

ПРИЛОЖЕНИЕ В

Факторный анализ

Таблица В.1 – Общности

	Начальная	Извлечение
Гемоглобин	1,000	0,921
Эритроциты	1,000	0,787
Цветовой показатель	1,000	0,660
Лейкоциты	1,000	0,677
Нейтрофилы палочкоядерные	1,000	0,509
Нейтрофилы сегментоядерные	1,000	0,515
Лимфоциты	1,000	0,748
Моноциты	1,000	0,098
Скорость оседания эритроцитов	1,000	0,226
Общий билирубин	1,000	0,506
Общий белок	1,000	0,598
Амилаза	1,000	0,008
Глюкоза	1,000	0,005
Длительность оперативных вмешательств	1,000	0,071
пол	1,000	0,096

Таблица В.2 – Дисперсия, объясненная последовательно выделяемыми факторами (компонентами)

Компонент	Начальные собственные значения			Извлечение суммы квадратов нагрузок			Ротация суммы квадратов нагрузок		
	Всего	% дисперсии	Суммарный %	Всего	% дисперсии	Суммарный %	Всего	% дисперсии	Суммарный %
1	3,296	21,972	21,972	3,296	21,972	21,972	3,020	20,136	20,136
2	2,329	15,526	37,498	2,329	15,526	37,498	2,604	17,362	37,498
3	1,463	9,752	47,249						
4	1,355	9,031	56,280						
5	1,177	7,849	64,129						
6	1,044	6,959	71,089						
7	,903	6,020	77,109						
8	,830	5,534	82,643						
9	,724	4,827	87,470						
10	,667	4,445	91,915						
11	,496	3,306	95,221						
12	,390	2,597	97,819						
13	,297	1,980	99,799						
14	,017	,111	99,910						
15	,013	,090	100,000						

Таблица В.3 – Матрица компонентов

	Компонент	
	1	2
Гемоглобин	0,896	
Эритроциты	0,845	
Цветовой показатель	0,736	
Лейкоциты		
Нейтрофилы палочкоядерные		
Нейтрофилы сегментоядерные		
Лимфоциты		
Моноциты		
Скорость оседания эритроцитов		

Общий билирубин	-0,505	0,703
Общий белок		0,668
Амилаза		0,633
Глюкоза		0,539
Длительность оперативных вмешательств		
пол		

Таблица В.4 – Повернутая матрица компонентов

	Компонент	
	1	2
Гемоглобин	0,941	
Эритроциты	0,859	
Цветовой показатель	0,806	
Общий белок	0,544	
Скорость оседания эритроцитов		
Длительность оперативных вмешательств		
Лимфоциты		0,863
Лейкоциты		0,821
Нейтрофилы палочкоядерные		0,711
Общий билирубин		0,521
Нейтрофилы сегментоядерные		
пол		
Моноциты		
Амилаза		
Глюкоза		

ПРИЛОЖЕНИЕ Г

Листинг программы

```
import pandas as pd
import numpy as np
import seaborn as sns
import scipy.stats as sts
from sklearn.feature_selection import SelectKBest
from sklearn.ensemble import RandomForestClassifier
from sklearn.feature_selection import chi2
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split

from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.svm import SVC
from sklearn.linear_model import Perceptron
from sklearn.neural_network import MLPClassifier
from sklearn.ensemble import RandomForestClassifier



from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import f1_score
from sklearn.metrics import recall_score
from sklearn.metrics import precision_score
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score
from sklearn.metrics import auc

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

```
from matplotlib.pyplot import figure
from IPython.core.pylabtools import figsize
import itertools
from sklearn.datasets import make_friedman1
from sklearn.feature_selection import RFE
from sklearn.feature_selection import RFECV
from sklearn.ensemble import RandomForestClassifier,
AdaBoostClassifier
from sklearn.linear_model import RidgeCV
from sklearn.ensemble import StackingClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import VotingClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import cross_val_score
from sklearn.discriminant_analysis import
LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import mean_absolute_error,
mean_squared_error,mean_squared_log_error,
r2_score,mean_absolute_percentage_error

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

pd.options.mode.chained_assignment = None

 # Подключаем модуль для Телеграма
import telebot
# Импортируем типы из модуля, чтобы создавать кнопки
from telebot import types
import dummy_threading
# Загрузка выборки
df=pd.read_csv('/content/SPO_PGU/input.csv', header=None)

# Подготовка выборки
```

```
def preproc(df):
# загрузим данные
#будет установлена первая строка в качестве имен столбцов
dataframe
    df.columns = df.iloc[0]
    #удаляем 0-строку
    df=df.drop(df.index[[0,2]])
    #преобразуем к числовому формату
    df=df.iloc[0:108].apply(pd.to_numeric)
    x=df.iloc[:,0:15]
    y=df.iloc[:,-1]
    print(x)
    print(y)
    return df,x,y
def stat(df):
# описательная статистика для числовых данных
    print(df.describe())
    a=[]
    b=[]
    cols=list(df)
    for i in cols:
        kurt=sts.kurtosis(df[i], axis=0,fisher=True,bias=True)
        sk=sts.skew(df[i],axis=0, bias=True)
        print('Показатели ассиметрии и эксцесса для:
{}'.format(i))
        print(kurt,sk)
        a.append(kurt)
        b.append(sk)
    return a,b
def k_best(x,y):
    bestfeature=SelectKBest(score_func=chi2,k='all')
    fit=bestfeature.fit(x,y)
    df_score=pd.DataFrame(fit.scores_)
    df_column=pd.DataFrame(x.columns)
    feature_score=pd.concat([df_column,df_score],axis=1)
```

```
    feature_score.columns=['Specs', 'Score']
    feature_score=feature_score.sort_values(by='Score',
ascending=False)
    print('Значимость показателей')
    print(feature_score)
# выбираем показатели по важности
    f_s1=feature_score[feature_score['Score']>20]
    drop_list1 = f_s1['Specs']
    #drop_list1.head(7)
#feature_score
    x_1=x.loc[:,drop_list1]#3
    print('Используя метод фльтрации Хи2 для отбра показателей
мы получили следующий результат. Наиболее информативными
показателями являются:')
    print(drop_list1)
    print('Создадим новую выборку состоящую толко из этих
показателей.')
    print(x_1)
    return feature_score,x_1,f_s1,drop_list1

def scal(x,y):
#шкалируем весь набор данных
    scaler = MinMaxScaler(feature_range=(0,1))
#назначение показателейдля шклирования
    x_scaled = pd.DataFrame(scaler.fit_transform(x),
columns=x.columns)
# обучающая и тестовая выборки по полным данным
# деление на обучающую и тестовую выборки: 80 % - 20 %
    x_train, x_test, y_train, y_test =
train_test_split(x_scaled, y, test_size = 0.3)
    return x_train, x_test, y_train, y_test

df,x,y=preproc(df)
kurt,sk=stat(df)
feature_score,x_k,f_s1, drop_list1=k_best(x,y)
```

```
x_train, x_test, y_train, y_test=scal(x_k,y)

# Оценка качества классификации

def accuracy_model(y_test, model_pred):
# доля правильных ответов алгоритма: Точность = (истинное
положительное + истинно отрицательное значение) / всего
    acc = accuracy_score(y_test, model_pred)
    print(f"The accuracy score for method is:
{round(acc,3)*100}%")

# Оценка f1: оценку F1 можно интерпретировать как
средневзвешенное значение точности и запоминания, где оценка F1
достигает своего лучшего значения при 1 и худшего значения при
0.
    f1 = f1_score(y_test, model_pred)
    print(f"The f1 score for method is: {round(f1,3)*100}%")

# долю объектов, названных классификатором положительными и при
этом действительно являющимися положительными: если прогноз
«да», как часто он оказывается правильным? Точность = истинно
положительный / предсказанный да
    precision = precision_score(y_test, model_pred)
    print(f"The precision score for method is:
{round(precision,3)*100}%")

# показывает, какую долю объектов положительного класса из всех
объектов положительного класса нашел алгоритм: истинно
положительный показатель (чувствительность): когда на самом деле
да, как часто он дает предсказание «да»? Истинно положительный
процент = истинно положительный / фактический да
    recall = recall_score(y_test, model_pred)
    print(f"The recall score for method is:
{round(recall,3)*100}%")
```

```
    # матрица неточности

    cnf_matrix = confusion_matrix(y_test, model_pred)

    fig, ax = plt.subplots()
    ax.xaxis.set_label_position("top")
    plt.tight_layout()
    plt.title('Confusion matrix \n', y=1.1)
    sns.heatmap(pd.DataFrame(cnf_matrix), annot=True,cmap="Reds"
, fmt='g')

    # Compute ROC curve and ROC area for each class
    fpr = dict()
    tpr = dict()
    roc_auc = dict()
    fpr, tpr, thresholds = roc_curve(y_test, model_pred)
    roc_auc = auc(fpr, tpr)
    figsize(5, 5)
    plt.figure()
    lw = 2
    plt.plot(fpr, tpr, color='darkorange', lw=lw, label='ROC
curve (area = %0.2f)' % roc_auc)
    plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--
')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver operating characteristic example')
    plt.legend(loc="lower right")
    plt.show()
    return acc,f1,precision,recall,fpr

# Методы машинного обучения
def pred(vid, x_train, x_test,y_train):
```

```
    if vid == 'log':
        model = LogisticRegression().fit(x_train, y_train)
        pred = model.predict(x_test)
    elif vid == 'knn':
        model = KNeighborsClassifier(n_neighbors =
2).fit(x_train, y_train)
        pred = model.predict(x_test)
    elif vid == 'tree':
        model = DecisionTreeClassifier(criterion='entropy',
max_leaf_nodes=6, random_state=0).fit(x_train, y_train)
        pred = model.predict(x_test)
    elif vid == 'svm':
        model = SVC().fit(x_train, y_train)
        pred = model.predict(x_test)
    elif vid == 'mlp':
        model = MLPClassifier(activation='relu', solver='lbfgs',
alpha=1e-5, max_iter=1000, hidden_layer_sizes=(5, ),
random_state=1).fit(x_train, y_train)
        pred = model.predict(x_test)
    elif vid == 'for':
        model =
RandomForestClassifier(criterion='entropy',max_depth=2,
random_state=1).fit(x_train, y_train)
        pred = model.predict(x_test)
    else:
        print("Такого метода нет")
    return model, pred

#обучение на сокращенном наборе k-best
forst, pred7=pred('for', x_train, x_test, y_train)
acc6=accuracy_model(y_test,pred7)
print(f"The accuracy score for RandomForest is:
{round(accrf,3)*100}%")
print(evaluate_model(forst,x_test,y_test))
```

```
# Указываем токен
bot =
telebot.TeleBot('1937629580:AAGisRGqMOzSkHcqBzU_XI2Eh0XmKz9t-
uc')

# загрузка выборки
df=pd.read_csv('/content/SPO_PGU/input.csv', header=None)
#предобработка
df,x,y=preproc(df)
#отбор информативных показателей
feature_score,x_k,f_s1, drop_list1=k_best(x,y)
# разбиение выборки
x_train, x_test, y_train, y_test=scal(x_k,y)
 #обучение на сокращенном наборе k-best
forst, pred7=pred('for', x_train, x_test, y_train)
#точность модели
acc6=accuracy_score(y_test,pred7)
# переменные для ввода показателей
class Param:
  def _init_(self,bil,neutr,amyl,dur,lymph):
    self.bil=bil
    self.neutr=neutr
    self.amyl=amyl
    self.dur=dur
    self.lymph=lymph

par=Param()

@bot.message_handler(commands=['start'])
def start(message):
    markup = types.InlineKeyboardMarkup()
    button1 = types.InlineKeyboardButton(text="Да",
callback_data = 'yes')
    button2 = types.InlineKeyboardButton(text="Нет",
callback_data = 'no')
```

```
    markup.add(button1, button2)
    bot.send_message(message.chat.id, 'Привет, хотите отобразить
клавиатуру с доступными функциями?', reply_markup = markup)

@bot.message_handler(commands=['commands'])
def commands(message):
    bot.reply_to(message, "Доступные
команды\n/start\n/opred\n/commands")

@bot.message_handler(commands=['opred'])
def opred(message):
    bot.send_message(message.chat.id, 'Выберете нужный вам
метод\n1) Загрузка выборки - /Load\n2) Отбор параметров -
/Select\n3) Случайный лес - /Tree\n4) Предсказание - /Test')

@bot.message_handler(commands=['Load'])
#загрузка выборки
def Load(message):
    bot.send_message(message.chat.id,f"Наша выборка:\n
{df.head(3)}\n")

@bot.message_handler(commands=['Select'])
#отбор показателей
def Select(message):
    bot.send_message(message.chat.id,f"Мы отобрали информативные
показатели:\n {drop_list1} \n")

@bot.message_handler(commands=['Tree'])
# Случайный лес
def Tree(message):
    bot.send_message(message.chat.id,f"Точность модели:
{round(acc6,3)*100}% \n")

@bot.message_handler(commands=['Test'])
# тестирование
```

```
def Test(message):
    message = bot.reply_to(message,f"Модель обучена!Напишите
данные пациента\n {'Билирубин'} \n")
    bot.register_next_step_handler(message, get_name);
#следующий шаг – функция get_name

def get_name(message): #получаем анализы пациента
  try:
    bil=message.text
    if not bil.isdigit():
      message=bot.reply_to(message, "Показатель должен быть
числовым. \n Введите Билирубин")
      bot.register_next_step_handler(message, get_name)
      return
    par.bil=bil
    message = bot.reply_to(message, "Введите показатель
нейтрофи́лы в крови. \n")
    bot.register_next_step_handler(message, get_neut)
  except Exception as e:
    bot.reply_to(message, 'Что-то пошло не так. ☹')

def get_neut(message):
  try:
    neut=message.text
    if not neut.isdigit():
      message=bot.reply_to(message, "Показатель должен быть
числовым. \n Введите нейтрофи́лы в крови")
      bot.register_next_step_handler(message, get_neut)
      return
    par.neutr=neut
    message = bot.reply_to(message, "Введите  показатель амилаза
в крови. \n")
    bot.register_next_step_handler(message, get_amyl)
  except Exception as e:
    bot.reply_to(message, 'Что-то пошло не так. ☹')
```

```
def get_amyl(message):
  try:
    amyl=message.text
    if not amyl.isdigit():
      message=bot.reply_to(message, "Показатель должен быть
числовым. \n Введите показатель амилаза в крови")
      bot.register_next_step_handler(message, get_amyl)
      return
    par.amyl=amyl
    message = bot.reply_to(message, "Введите длительность
операции. \n")
    bot.register_next_step_handler(message, get_dur)
  except Exception as e:
    bot.reply_to(message, 'Что-то пошло не так. ☹')

def get_dur(message):
  try:
    dur=message.text
    if not dur.isdigit():
      message=bot.reply_to(message, "Показатель должен быть
числовым. \n Введите длительность операции.")
      bot.register_next_step_handler(message, get_dur)
      return
    par.dur=dur
    message = bot.reply_to(message, "Введите показатель
лимфоциты в крови. \n")
    bot.register_next_step_handler(message, get_lym)
  except Exception as e:
    bot.reply_to(message, 'Что-то пошло не так. ☹')

def get_lym(message):
  try:
    lym=message.text
    if not lym.isdigit():
```

```
      message=bot.reply_to(message, "Показатель должен быть
числовым. \n Введите лимфоциты в крови.")
      bot.register_next_step_handler(message, get_lym)
      return
    par.lymph=lym
    message = bot.reply_to(message, "Проводим прогнозирование
осложнений после операции? \n Напишите да")
    bot.register_next_step_handler(message, process_age_step)
  except Exception as e:
    bot.reply_to(message, 'Что-то пошло не так. ☹')

def process_age_step(message):
  if message.text=='да':
    data_in = [[par.bil,par.neutr,par.amyl,par.dur,par.lymph]]
    #test1=np.asarray(test)
    bot.send_message(message.chat.id,f"Ваши данные о пациенте:\n
{data_in} \n")
    pred9 = forst.predict(data_in)
    #bot.send_message(message.chat.id,f"пациенте:\n
{pred9.round(1)[0]} \n")
    if pred9.round(1)[0]==1:
      bot.send_message(message.chat.id,f"Состояние пациента:
{'прогнозируются осложнения после операции'} \n")
    else:
      bot.send_message(message.chat.id,f"Состояние пациента:
{'прогнозируются отсутствие осложнений после операции'} \n")
  else:
      message=bot.reply_to(message, "Вы не подтвердили
прогнозирование осложнений. \n Введите да.")
      bot.register_next_step_handler(message, process_age_step)

@bot.callback_query_handler(func = lambda call: True) #
Обрадотчик нажатия на кнопки
def answer(call):
    if call.data == 'yes':
```

```
        bot.send_message(call.message.chat.id, 'Доступные
команды\n/start\n/opred\n/commands')
        markup = types.ReplyKeyboardMarkup(resize_keyboard=True,
row_width=3)
        buttonA = types.KeyboardButton('/start')
        buttonC = types.KeyboardButton('/commands')
        buttonG = types.KeyboardButton('/opred')
        markup.row(buttonA, buttonG, buttonC)
        bot.send_message(call.message.chat.id, 'Удачной
работы!', reply_markup=markup)
    elif call.data == 'no':
        bot.send_message(call.message.chat.id, 'Ладно,
разбирайся сам')

# Запускаем постоянный опрос бота в Телеграме
bot.polling(none_stop=True, interval=0)
```

ПРИЛОЖЕНИЕ Д

Минимальные и максимальные значения показателей крови

1. гемоглобин (120 –160 г/л);
2. эритроциты (3,9·1012–5,5·1012 л);
3. цветовой показатель (0,86 – 1,1);
4. лейкоциты (4·109 –9·109 л);
5. нейтрофилы палочкоядерные (1 – 6%);
6. нейтрофилы сегментоядерные (45 – 70%);
7. лимфоциты (18 – 40%);
8. моноциты (2 – 9%);
9. скорость оседания эритроцитов (1– 15 мм/час);
10. общий белок (60 – 80 г/л);
11. общий билирубин (3 – 20 мкмоль/л);
12. амилаза (<300 МЕ/л)
13. мочевина (3,3 – 6,7 ммоль/л);
14. креатинин (60 – 120 мкмоль/л);
15. глюкоза (2,8 – 6 ммоль/л);
16. пол;
17. длительность оперативных вмешательств (мин.).

ПРИЛОЖЕНИЕ Е

Презентация

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Кафедра «Вычислительная техника»

Кузнецов Роман Николаевич

IT-СРЕДСТВА ПРОГНОЗИРОВАНИЯ ПОСЛЕОПЕРАЦИОННЫХ ОСЛОЖНЕНИЙ (ЖЕЛЧНО-КАМЕННОЙ БОЛЕЗНИ)

Научный руководитель к.т.н., доцент Коннов Н.Н.

1

Актуальность

Желчно-каменная болезнь – одно из самых распространенных заболеваний, занимающее третье место после сердечно-сосудистых заболеваний и сахарного диабета.

Взрослое население России и Европы страдает ЖКБ в 10-20% случаев.

Осложнения после открытых операций на желчевыводящих путях связаны с длительностью пребывания пациента в стационаре, что приводит к росту экономических затрат на лечение больных.

2

Цель, объект, методы исследования

Цель исследования:
заключается в разработке модели диагностики пациентов, которая опирается на методы машинного обучения и алгоритмы функционирования интеллектуальной системы поддержки принятия решений.

Объект исследования:
является экспертная система медицинской диагностики и прогнозирования послеоперационных осложнений.

Методы исследования:
Логистическая регрессия, к-ближайших соседей, деревья решений, случайный лес, нейронные сети.

3

Задачи

- разработка методов подготовки данных для обучения нейронных сетей,
- выделение информативных признаков из экспериментальных данных,
- изучение и создание архитектур и алгоритмов для обучения нейронных сетей, а также разработка информационной модели интеллектуальной системы поддержки принятия решений,
- реализация программных средств обеспечения имитационного моделирования разрабатываемых алгоритмов и моделей для решения задач поддержки принятия решений,
- применение полученных теоретических результатов для построения модели имитации принятия решения при диагностике желчнокаменной болезни.

4

Анализ лабораторных методов диагностики

- Клинические методы включают в себя оценку риска на основе факторов, таких как возраст, наличие хронических заболеваний, наличие инфекционных заболеваний, состояние пациента перед операцией.
- Иммунологические методы основаны на оценке функции иммунной системы.
- Молекулярные маркеры включают в себя генетические варианты, связанные с болезнями и наследственными факторами.
- Генетические маркеры могут быть использованы для оценки генетических факторов, связанных с болезнями и наследственными предрасположенностями.

5

Обзор отечественных и зарубежных диагностических систем

- система компьютерной томографии "Тоскар-1",
- ЭКГ система компьютерной диагностики "Кардиоденс",
- радиологический компьютерный комплекс "Регион",
- система компьютерной диагностики "Магнитотерапия",
- диагностическая система "ДИФРАКТ".

- эхокардиограф "Philips EPIQ 7" (Нидерланды),
- система интраоперативной нейронавигации " Brainlab VectorVision" (Германия),
- система компьютерной томографии "Toshiba Aquilion Lightning" (Япония).

6

Постановка задачи

Для прогнозирования послеоперационных осложнений у пациентов с желчно-каменной болезнью была создана группа из 109 историй болезни, полученных отделением торакальной хирургии клинической больницы имени Н.Н. Бурденко.

Среди них 63 случая были без осложнений, а 46 случаев осложнены. Для прогнозирования использовались 15 показателей, основанных на общем и биохимическом анализах крови, а также на длительности оперативных вмешательств и пола пациента .

Полный перечень анализов включал гемоглобин, эритроциты, цветовой показатель, лейкоциты, нейтрофилы палочкоядерные и сегментоядерные, лимфоциты, моноциты, скорость оседания эритроцитов, общий белок, общий билирубин, амилазу, мочевину, креатинин, глюкозу, а также указание на пол и длительность операции (в минутах) .

Описание данных

	Hemoglobin	Erythrocytes	Color_Index	Leukocytes	Neutrophils
count	109.000000	109.000000	109.000000	109.000000	109.000000
mean	137.774312	4.307248	0.934128	8.586239	8.733945
std	18.073422	0.436412	0.042757	4.016412	7.527286
min	69.000000	2.500000	0.820000	3.000000	1.000000
25%	128.000000	4.300000	0.910000	5.400000	3.000000
50%	140.000000	4.500000	0.930000	7.500000	6.000000
75%	149.000000	4.600000	0.970000	10.300000	11.000000
max	173.000000	5.100000	1.000000	28.500000	43.000000

	Neutrophils_segmented	Lymphocytes	Monocytes	Erythrocyte
count	109.000000	109.000000	109.000000	109.000000
mean	62.688073	23.908257	3.853211	21.568807
std	8.673554	10.409127	2.552507	13.912003
min	30.000000	2.000000	1.000000	3.000000
25%	58.000000	15.000000	2.000000	15.000000
50%	64.000000	23.000000	3.000000	15.000000
75%	69.000000	31.000000	5.000000	25.000000
max	80.000000	47.000000	15.000000	64.000000

	Total_bilirubin	Total_protein	Amylase	Glucose	Duration
count	109.000000	109.000000	109.000000	109.000000	109.000000
mean	34.778899	70.170826	27.960550	6.762385	77.110092
std	75.816534	5.925769	20.297628	5.581995	38.611872
min	8.500000	57.500000	8.500000	3.800000	30.000000
25%	14.000000	67.000000	18.200000	5.300000	55.000000
50%	15.200000	69.000000	27.000000	6.000000	65.000000
75%	19.500000	74.000000	28.000000	6.800000	90.000000
max	722.500000	85.000000	146.300000	62.000000	240.000000

	sex	conclusion
count	109.000000	109.000000
mean	0.733945	0.422018
std	0.443934	0.496163
min	0.000000	0.000000
25%	0.000000	0.000000
50%	1.000000	0.000000
75%	1.000000	1.000000
max	1.000000	1.000000

8

Значения эксцесса и асимметрии анализов пациентов

№	Название	Асимметрия	Эксцесс
1.	Гемоглобин	-0,87	1,64
2.	Эритроциты	-1,81	4,59
3.	Цветовой показатель	-0,38	-0,05
4.	Лейкоциты	1,04	0,40
5.	Нейтрофилы палочкоядерные	1,84	4,11
6.	Нейтрофилы сегментоядерные	-0,51	0,04
7.	Лимфоциты	0,05	-0,81
8.	Моноциты	1,35	2,50
9.	Скорость оседания эритроцитов	1,50	1,33
10.	Общий билирубин	7,36	63,92
11.	Общий белок	0,30	-0,18
12.	Амилаза	4,06	19,74
13.	Глюкоза	9,16	90,81
14.	Длительность оперативных вмешательств	2,05	5,28
15.	Пол	-1,07	-0,86

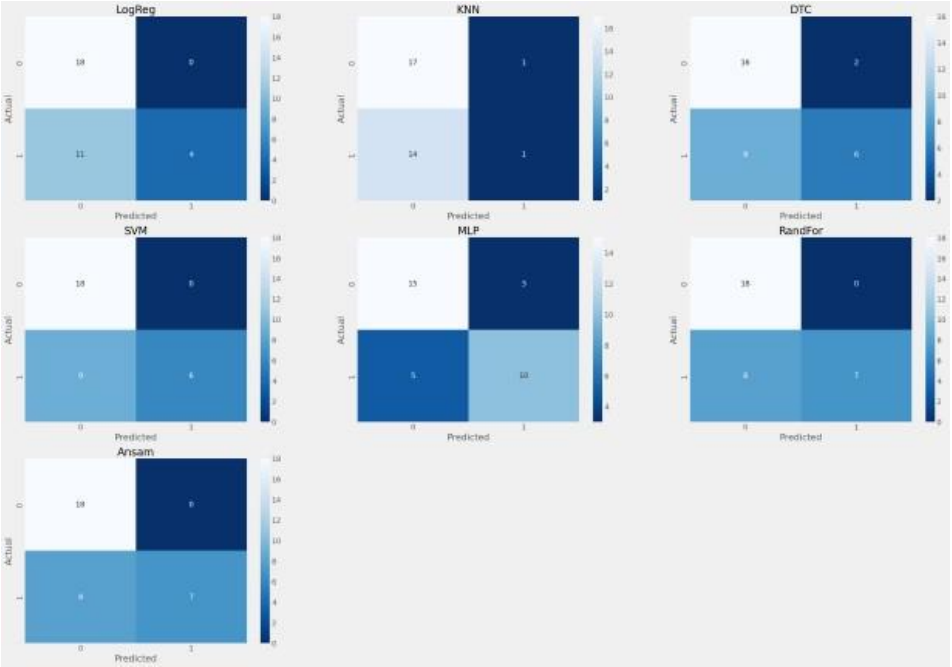
9

Отбора информативных признаков

№	Название	Критерий Вилкоксона-Манна-Уитни	Корреляционный анализ	Факторный анализ	Алгоритм прямого отбора переменных
1.	Гемоглобин			+	
2.	Эритроциты			+	
3.	Цветовой показатель			+	
4.	Лейкоциты			+	
5.	Нейтрофилы палочкоядерные	+	+	+	+
6.	Нейтрофилы сегментоядерные				+
7.	Лимфоциты	+		+	+
8.	Моноциты				
9.	Скорость оседания эритроцитов				
10.	Общий билирубин	+	+	+	+
11.	Общий белок			+	
12.	Амилаза	+	+	+	+
13.	Глюкоза				
14.	Длительность оперативных вмешательств	+	+		+
15.	Пол				

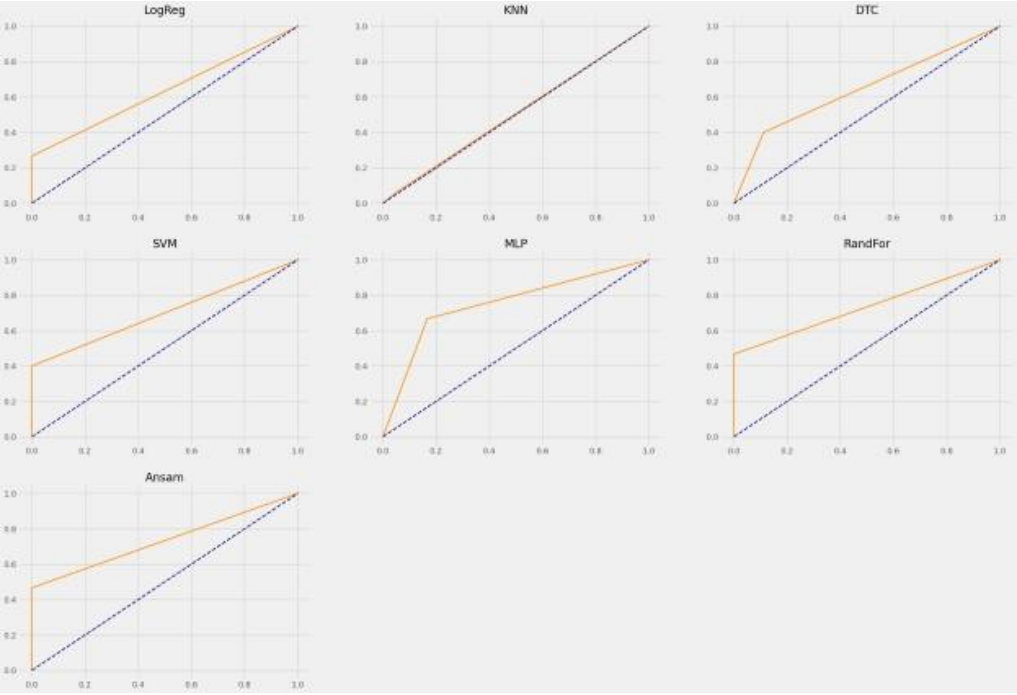
10

Классификация и прогноз послеоперационных осложнений при желчно-каменной болезни



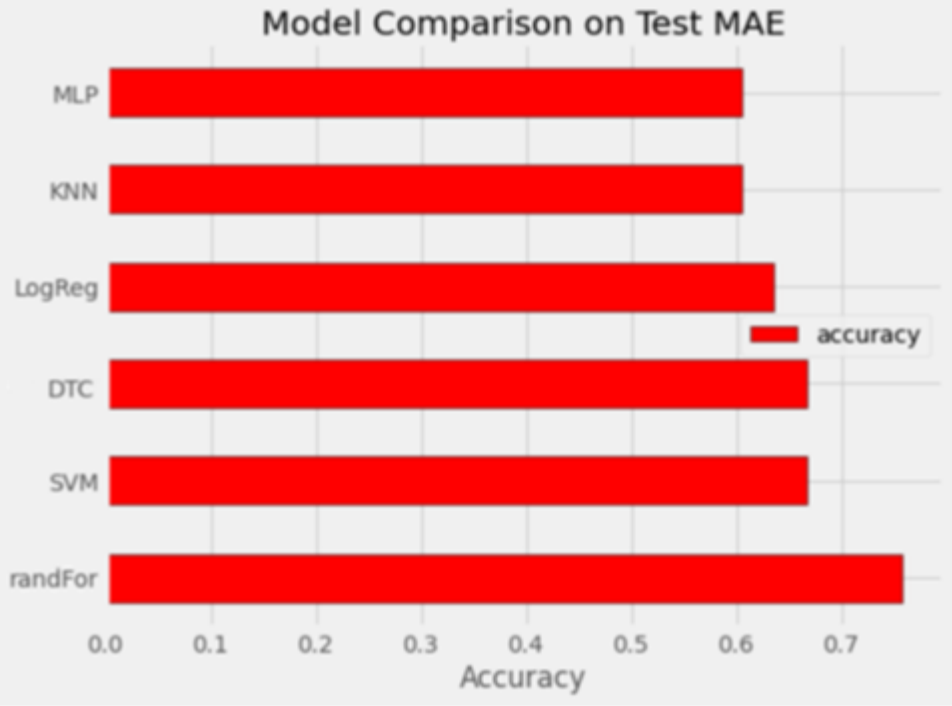
11

Классификация и прогноз послеоперационных осложнений при желчно-каменной болезни



12

График оценки точности методов



13

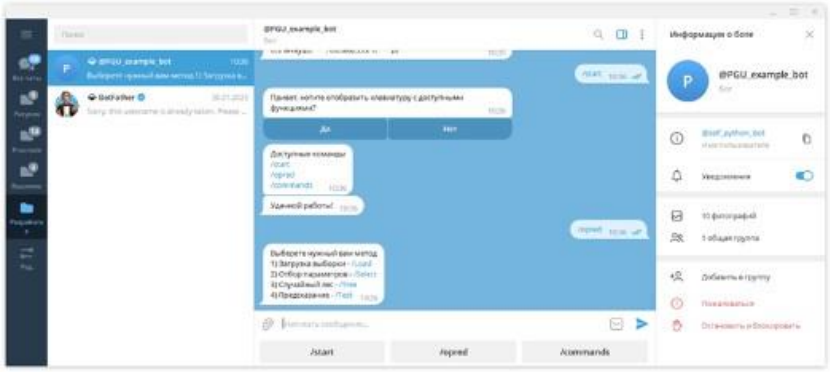
Разработка архитектуры чат-бота включает восемь этапов:

1) создание веб-клиента, который представляет собой серверное приложение, получающее обновления от Telegram через webhook и передающее их обработчику обновлений,
2) организация хранения выборки для обучения модели и ее тестирования с использованием репозитория GitHub,
3) идентификация пользователей, осуществляемая через уникальный идентификатор чата,
4) взаимодействие с чат-ботом, имеется возможность отправки сообщений, чтобы взаимодействовать с пользователями,
5) прием сообщений, который осуществляется через механизм приемников,
6) работа с кнопками, добавляемыми в сообщения для обработки нажатий кнопок был добавлен класс CallbackQuerryItem в проект Entities,
7) обработчик обновлений, который осуществляет обработку приходящих от клиента обновлений через класс CommonHandler в проекте UpdateHandlers,
8) бизнес-логика, необходимая для обеспечения требуемой функциональности бота.

14

Для реализации чат-бота потребуется четыре различные команды:

– загрузка выборки,
– отбор параметров,
– случайный лес,
– предсказание.



15

Загрузка выборки

16

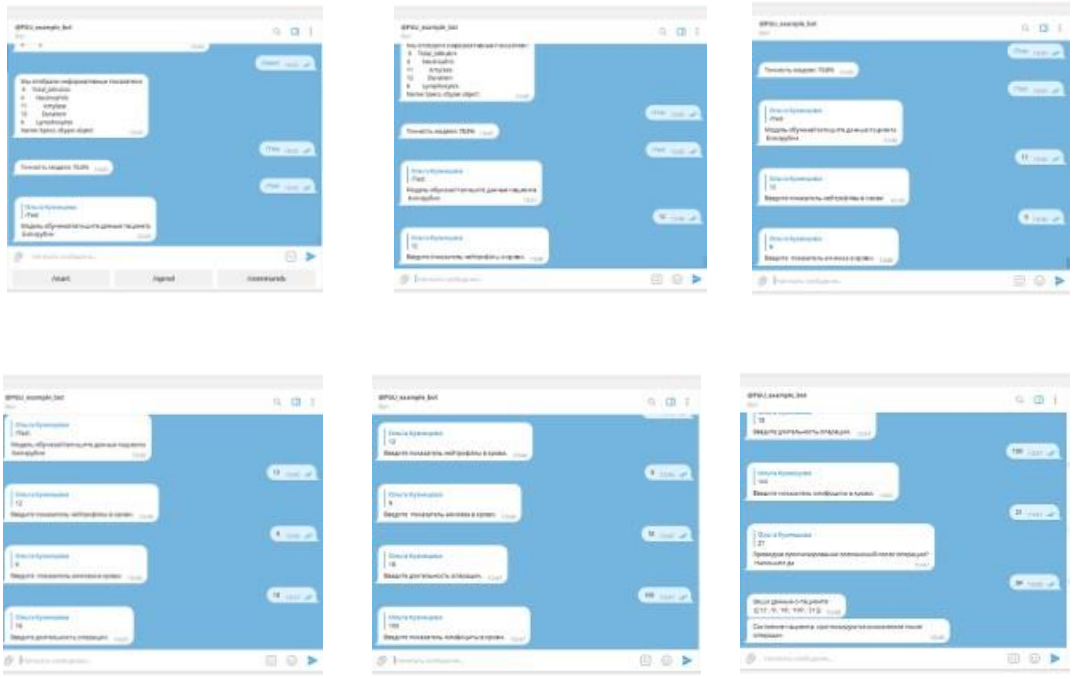
Отбор параметров

17

Случайный лес

18

Предсказание



19

Основные результаты и выводы

В процессе выполнения выпускной квалификационной работы были исследованы способы подготовки данных для обучения нейронных сетей: заполнение пропусков результатов анализов, проанализированы алгоритмы выделения информативных признаков из экспериментальных данных, исследована и разработана архитектура алгоритмов обучения нейронных сетей, разработка информационной модели интеллектуальной системы поддержки принятия решения при диагностике желчнокаменной болезни.

В пояснительной записке выполнены следующие разделы: введение, теоретические сведения (анализ состояния исследования желчнокаменной болезни, исследование методов отбора показателей, исследование методов прогнозирования), разработка программного кода. Произведена загрузка данных, обработка данных, создание нейронной сети, её обучение, тестирование и проверка.

20

СПАСИБО ЗА ВНИМАНИЕ!

21