Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
(ФГБОУ ВО «АмГУ»)

Институт компьютерных и инженерных наук
Кафедра математического анализа и моделирования
Направление подготовки: 01.04.02 – «Прикладная математика и информатика»
Направленность (профиль) образовательной программы – «Математическое и программное обеспечение информационных систем»

УТВЕРЖДАЮ
И.о. зав. кафедрой
_____Н.Н. Максимова
«_____»_____2025 г.

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

на тему: «Методы машинного обучения для диагностики заболевания диабетом»

Исполнитель
студент группы 3101ом _____ В.Т. Колесников
(подпись, дата)

Научный руководитель
доцент, канд. физ.-мат. наук _____ Н.Н. Максимова
(подпись, дата)

Руководитель научного
содержания программы
магистратуры
доцент, канд. физ.-мат. наук _____ Н.Н. Максимова
(подпись, дата)

Нормоконтроль
ассистент _____ В.О. Салмиянов
(подпись, дата)

Рецензент
доцент, канд. тех. наук _____ С.Г. Самохвалова
(подпись, дата)

Благовещенск 2025

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
(ФГБОУ ВО «АмГУ»)

Институт компьютерных и инженерных наук
Кафедра математического анализа и моделирования

УТВЕРЖДАЮ
И.о. зав. кафедрой
_____Н.Н. Максимова
«_____»_____2025 г.

З А Д А Н И Е

К магистерской диссертации студента Вадима Тимофеевича Колесникова

1. Тема магистерской диссертации: Методы машинного обучения для диагностики заболевания диабетом.

(утверждены приказом от 24.01.2025 г. № 162-уч)

2. Срок сдачи студентом законченной проекта: 24 июня 2025 года.

3. Исходные данные к магистерской диссертации: набор данных «Diabetes prediction», отчеты по производственной практике (научно-исследовательской работе) за 1-3 семестры, отчет по преддипломной практике, среда разработки Google Colab.

4. Содержание выпускной квалификационной работы (перечень подлежащих разработке вопросов): обзор истории развития искусственного интеллекта и его современное состояние, обзор постановок задач машинного обучения, обзор набора данных; модель логистической регрессии, модель наивного Байеса, нейронные сети.

5. Перечень материалов приложения (наличие чертежей, таблиц, графиков, схем, программных продуктов, иллюстративного материала и т.п.): результаты работы программ в виде изображений, результаты работы программ в виде таблиц.

6. Консультанты по магистерской диссертации: рецензент – Самохвалова С.Г.,

доцент кафедры информационной безопасности, ФГБОУ ВО «АмГУ», канд. техн. наук, доцент; нормоконтроль – Салмиянов В.О., ассистент.

7. Дата выдачи задания: 03.02.2025 г.

Руководитель магистерской диссертации: Максимова Надежда Николаевна, доцент, канд. физ.-мат. наук, доцент

Задание принял к исполнению (03.02.2025): _____ В.Т. Колесников

РЕФЕРАТ

Магистерская диссертация содержит 91 с., 76 рисунков, 12 таблиц, 3 приложений, 25 источников.

ДИАБЕТ, НАБОР ДАННЫХ, МАШИННОЕ ОБУЧЕНИЕ, ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ, PYTHON, БИНАРНАЯ КЛАССИФИКАЦИЯ, ЛОГИСТИЧЕСКАЯ РЕГРЕССИЯ, НАИВНЫЙ БАЙЕС, НЕЙРОННАЯ СЕТЬ.

В работе решается задача проектирования модели машинного обучения для прогнозирования диабета у пациентов на основе их истории болезни и демографической информации, а также оценка работоспособности полученных моделей.

Цель магистерской диссертации – создание модели машинного обучения для прогнозирования заболевания – диабет.

Для достижения цели были поставлены следующие задачи:

- ознакомление с историей развития искусственного интеллекта и его современным состоянием;

- осуществить анализ постановок задач машинного обучения, а также метрик оценки качества моделей;

- провести предобработку набора данных «Diabetes prediction»;

- построить и обучить модели для прогнозирования диабета классическими методами, оценить их качества;

- построить и обучить модели для прогнозирования диабета при помощи нейронной сети, оценить ее качество;

- сравнить полученные результаты всех моделей и выбрать оптимальный вариант;

В качестве языка программирования используется Python, а средой разработки является Google Colab.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Машинное обучение (МО) представляет собой область искусственного интеллекта, направленную на разработку алгоритмов и статистических моделей, позволяющих компьютерным системам выполнять задачи без явного программирования. Вместо жёстко заданных инструкций системы опираются на выявление закономерностей в данных и построение логических выводов на их основе.

Алгоритмы машинного обучения применяются для анализа больших объёмов информации, с целью автоматического извлечения скрытых паттернов и принятия обоснованных решений.

В последние годы распространение данных и достижения в вычислительных возможностях стали катализаторами быстрого роста технологий машинного обучения в здравоохранении. Интеграция МО в патологию и медицину открыла новые возможности для повышения точности диагностики болезней, например, таких, как диабет.

Диабет – хроническое заболевание, которое возникает либо в случаях, когда поджелудочная железа не вырабатывает достаточное количество инсулина, либо, когда организм не может эффективно использовать вырабатываемый инсулин. Инсулин – это гормон, регулирующий уровень глюкозы в крови. Распространенным следствием неконтролируемого диабета является гипергликемия, или повышенный уровень содержания глюкозы (сахара) в крови, со временем приводящий к серьезному повреждению многих систем организма, особенно нервов и кровеносных сосудов. Ранняя диагностика заболеваний и своевременное проведение профилактических мероприятий способны значительно снизить вероятность развития осложнений и повысить качество жизни пациентов. Вместе с тем, традиционные методы диагностики зачастую характеризуются высокими временными и финансовыми затратами, а также не всегда обеспечивают достаточный уровень точности.

Учитывая серьезность заболевания, а также развитие технологий машинного обучения, можно сказать, что предсказание диабета при помощи искусственного интеллекта является актуальной темой.

Объектом исследования является набор данных «Diabetes prediction dataset», содержащем информацию о пациентах и наличии у них диабета.

Предмет исследования – модели машинного обучения для данного предсказания заболевания.

Целью работы является создания моделей машинного обучения для прогнозирования диабета у пациентов на основе их истории болезни и демографической информации, а также оценка работоспособности полученных моделей.

Для достижения цели в ходе написания магистерской диссертации необходимо выполнить ряд поставленных **задач**:

- ознакомится с историей развития искусственного интеллекта и его современным состоянием;

- провести анализ постановок задач машинного обучения, а также метрик оценки качества моделей;

- осуществить предобработку набора данных «Diabetes prediction»;

- построить и обучить модели для прогнозирования диабета классическими методами, оценить их качества;

- построить и обучить модели для прогнозирования диабета при помощи нейронной сети, оценить ее качество;

- сравнить полученные результаты всех моделей и выбрать оптимальный вариант;

Теоретическая значимость магистерской диссертации заключается в создании основной кодовой базы, которая может быть доработана или изменена в будущем.

В качестве **практической значимостью** данной работы, использование модели может быть полезно для медицинских работников при диагностике и выявлении заболеваний у пациентов, которые могут подвергаться риску разви-

тия диабета. Это способствует составлению более точного индивидуального курса лечения.

Подобные исследования с применением алгоритмов машинного обучения и нейронных сетей помогают медицинским работникам при постановке диагноза, а анализ влияния различных признаков на результат – выявлять негативные факторов, а также более точно выстраивать тактику лечение больных пациентов.

Структура работы

Диссертация состоит из введения, четырех глав, заключения, списка использованных источников и приложений.

В первой главе рассматриваются основные понятия машинного обучения.

Вторая глава посвящена методам машинного обучения, а также метрикам оценки качества моделей.

В третьей главе приводится моделирование прогнозирования диабета на классических методах.

В четвертой главе описано создание модели диагностики заболевания на основе нейронной сети.

Заключение содержит основные выводы и направления дальнейшего исследования.

По результатам работы **опубликованы** 2 тезисов докладом и 1 статья [23-25]. Результаты работ докладывались на научно-методических семинарах кафедры математического анализа и моделирования и трех **научных конференциях**:

- XXXIII и XXXIV научные конференции «День науки» (Амурский государственный университет, 2024 и 2025 гг.);

- Национальная научная конференция «Far East Math 2024» (Тихоокеанский государственный университет).

1 ВВЕДЕНИЕ В ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ

В условиях массового роста больших данных и передовых технологий, таких как Интернет вещей, объем и сложность современных наборов данных затрудняют их анализ с помощью традиционных инструментов бизнес-аналитики (BI). И машинное, и глубокое обучение стали иметь решающее значение для решения новых задач в нескольких отраслях.

Искусственный интеллект (ИИ) охватывает совокупность технологий и методов, позволяющих автоматизировать выполнение задач, которые традиционно требуют участия человека – таких как распознавание образов, принятие решений, обработка естественного языка и прогнозирование. На современном этапе развития науки и технологий ИИ активно внедряется во все сферы жизни, становясь важной составляющей цифровой инфраструктуры и неотъемлемой частью функционирования современного общества.

По своей сути машинное обучение – это подмножество искусственного интеллекта, которое позволяет компьютерным системам учиться на данных и улучшать свою производительность с течением времени без явного программирования для каждой задачи. Вместо того чтобы полагаться на фиксированный набор правил, курируемых экспертами-людьми, алгоритмы машинного обучения выводят правила и шаблоны непосредственно из больших наборов данных. В отличие от экспертных систем, которые требуют ручного обновления знаний даже при незначительных изменениях в предметной области, модели машинного обучения обладают способностью динамически адаптироваться к новым данным. Это позволяет им формировать прогнозы и принимать обоснованные решения в условиях ранее не встречавшихся ситуаций [5].

В области машинного обучения существует множество методов, каждый из которых имеет свой собственный набор преимуществ, недостатков и идеальных вариантов использования. От деревьев решений и опорных векторных машин до случайных лесов и наивных байесовских классификаторов — набор алгоритмов, имеющихся в распоряжении специалиста по данным, широк. Однако

среди них нейронные сети выделяются своими непревзойденными возможностями в обработке сложных и многомерных данных. Их способность автоматически изучать сложные шаблоны и представления делает их исключительно универсальными и мощными, особенно для таких задач, как распознавание изображений и речи, обработка естественного языка и даже игры. В отличие от других алгоритмов машинного обучения, которые могут испытывать трудности со сложностью и масштабом таких проблем, нейронные сети преуспевают в них, часто обеспечивая превосходную производительность.

Нейронная сеть состоит из слоев взаимосвязанных узлов или «нейронов», вдохновленных нейронной структурой человеческого мозга. Каждый нейрон получает входные данные, обрабатывает их с помощью взвешенной суммы и функции активации и передает результат нейронам в следующем слое. Эти веса корректируются в процессе обучения с помощью техники, известной как обратное распространение, которая минимизирует ошибку между предсказанными и фактическими результатами. Архитектура нейронной сети может значительно различаться: некоторые сети имеют только один слой нейронов, в то время как другие имеют несколько слоев, известных как глубокие нейронные сети. Взаимосвязи между нейронами, связанные веса и метод корректировки этих весов вносят вклад в способность сети обучаться и делать прогнозы или решения.

1.1 История развития искусственного интеллекта

В течение столетий, предшествовавших 1950-м годам, появилось несколько философских и логических концепций, которые легли в основу теорий искусственного интеллекта. Древнегреческие философы оказали большое влияние на западную культуру, имея идеи о сущности сознания, человеческой мысли и обучении. В течение сотен лет эти концепции развивались и в конечном итоге стали более сосредоточенными на возможности машин обрести способность к обучению и на искусственном интеллекте, поскольку технологии все больше интегрировались в человеческую жизнь.

Хронология искусственного интеллекта восходит к 1763 году, когда Томас Байес разработал структуру вероятности событий, названную байесовской референцией, которая послужила ведущим подходом для машинного обучения. В начале 1900-х годов появились первые изображения роботов в популярных медиа по всему миру, таких как фильмы «Метрополис» и «Гакутенсоку».

Основываясь на этих ранних корнях, ниже представлена краткая хронология основных событий, произошедших в период быстрого развития искусственного интеллекта за последние несколько десятилетий:

Ниже приведены основные даты развития ИИ [11].

1950 год – Алан Тьюринг опубликовал работу «Вычислительные машины и разум», в которой он представил концепцию теста, позже получившего название «тест Тьюринга», и заложил основы для дальнейшего развития искусственного интеллекта.

1956 год – Джон Маккарти, Марвин Минский, Натаниэль Рочестер и Клод Шеннон предложили термин «искусственный интеллект» в рамках заявки на проведение семинара в Дартмутском колледже, который впоследствии был признан ключевым событием в становлении этой области исследований.

1959 год – Артур Сэмюэл впервые использовал термин «машинное обучение» в своей работе, описав возможность программирования компьютеров таким образом, чтобы они могли превзойти своих создателей в процессе выполнения определённых задач, таких как игра в шашки.

1965 год – Эдвард Фейгенбаум, Брюс Бьюкенен, Джошуа Ледерберг и Карл Джерасси разработали первую экспертную систему Dendral, предназначенную для помощи химикам-органикам в идентификации структуры неизвестных органических соединений.

1981 год – Дэнни Хиллис спроектировал параллельные вычислительные архитектуры, специально предназначенные для задач искусственного интеллекта. Их структура стала прообразом современных графических процессоров (GPU).

1988 год – Питер Браун и его коллеги представили статью «Статистический подход к языковому переводу», которая стала основой для развития одного из самых популярных направлений в области машинного перевода.

2000 год – Исследовательская группа из Монреальского университета опубликовала работу «Нейронно-вероятностная языковая модель», в которой был предложен метод моделирования естественного языка с использованием нейронных сетей прямого распространения.

2009 год – Раджат Райна, Ананд Мадхаван и Эндрю Нг опубликовали статью «Крупномасштабное глубокое обучение без учителя с использованием графических процессоров», в которой была продемонстрирована эффективность использования GPU для обучения больших нейронных сетей.

2023 год – Компания OpenAI объявила о выпуске мультимодельной языковой модели GPT-4, способной обрабатывать как текстовые, так и графические входные данные.

1.2 Области искусственного интеллекта

Алгоритмы машинного обучения – это форма ИИ, которая учится непосредственно на исторических данных для выявления закономерностей. В результате их можно использовать для принятия решений и точных прогнозов при использовании дополнительных наборов данных.

Приложения машинного обучения обучаются в интерактивном процессе, который снабжает модель помеченными данными. Затем алгоритм делает прогнозы и учится, как адаптировать эти данные для повышения точности, хотя для извлечения признаков требуется вмешательство человека – процесс, который определяет ключевые особенности из необработанных данных, которые алгоритм должен использовать для классификации данных [2].

Глубокое обучение – это разновидность машинного обучения, которая использует искусственные нейронные сети и огромные объемы данных для анализа данных и генерации результатов таким образом, чтобы имитировать работу человеческого мозга. Глубокое обучение также называют глубокими нейронными сетями, поскольку оно обычно использует три или более слоев

нейронных сетей для выполнения математических операций над данными и извлечения функций.

В глубоком обучении алгоритмы создаются аналогично алгоритмам машинного обучения, но имеют гораздо больше уровней, которые увеличивают сложность и абстракцию. Компьютерная модель учится классифицировать данные непосредственно из изображений, текста или звука без первоначального извлечения признаков, которое требуется при машинном обучении [5].

1.3 Основные задачи, решаемые методами машинного обучения и искусственными нейронными сетями

1. Задача регрессии

Регрессия в машинном обучении относится к контролируемому методу обучения, где цель состоит в том, чтобы предсказать непрерывное числовое значение на основе одного или нескольких независимых признаков. Он находит связи между переменными, чтобы можно было делать прогнозы. В регрессии присутствуют два типа переменных [13]:

Зависимая переменная (целевая): переменная, которую нужно предсказать, например, цена дома.

Независимые переменные (признаки): входные переменные, влияющие на прогноз, например, местоположение, количество комнат.

Проблема регрессионного анализа работает с выходной переменной, которая является действительным или непрерывным значением.

Регрессию можно классифицировать по различным типам в зависимости от количества предикторов и характера связи между переменными:

1) Простая линейная регрессия

Линейная регрессия – одна из самых простых и широко используемых статистических моделей. Она предполагает, что существует линейная связь между независимыми и зависимыми переменными. Это означает, что изменение зависимой переменной пропорционально изменению независимых переменных. Например, прогнозирование цены дома на основе его размера.

2) Множественная линейная регрессия

Множественная линейная регрессия расширяет простую линейную регрессию, используя несколько независимых переменных для прогнозирования целевой переменной. Например, прогнозирование цены дома на основе нескольких характеристик, таких как размер, местоположение, количество комнат и т. д.

3) Полиномиальная регрессия

Полиномиальная регрессия используется для моделирования нелинейных отношений между зависимой переменной и независимыми переменными. Она добавляет полиномиальные члены в модель линейной регрессии для охвата более сложных отношений. Например, когда нужно предсказать нелинейную тенденцию, такую как рост населения с течением времени.

4) Регрессия гребня и лассо

Регрессия гребня и лассо – это регуляризованные версии линейной регрессии, которые помогают избежать переобучения, штрафуя большие коэффициенты. Когда есть риск переобучения из-за слишком большого количества признаков, мы используем этот тип алгоритмов регрессии.

5) Регрессия опорных векторов (SVR)

SVM – это тип алгоритма, который используется для задач классификации, но его также можно использовать для задач регрессии. SVR работает путем поиска гиперплоскости, которая минимизирует сумму квадратов остатков между прогнозируемыми и фактическими значениями.

6) Регрессия дерева решений

Дерево решений использует древовидную структуру для принятия решений, где каждая ветвь дерева представляет решение, а листья представляют результаты. Например, для прогнозирования поведения клиентов на основе таких характеристик, как возраст, доход и т.д.

7) Регрессия случайного леса

Случайный лес – это ансамблевый метод, который строит несколько деревьев решений, и каждое дерево обучается на отдельном подмножестве обучающих данных. Окончательный прогноз делается путем усреднения прогнозов

всех деревьев. Например, отток клиентов или данные о продажах с использованием этого.

2. Задача классификации

Классификация обучает машину сортировать вещи по категориям. Она учится, просматривая примеры с метками (например, письма с пометкой «спам» или «не спам»). После обучения модель может решить, к какой категории относятся новые элементы, например, определить, является ли новое письмо спамом или нет. Например, модель классификации может быть обучена на наборе данных изображений, помеченных как собаки или кошки, и ее можно использовать для прогнозирования класса новых и невиданных изображений как собак или кошек на основе их характеристик, таких как цвет, текстура и форма.

Существуют различные типы задач классификации в зависимости от того, с каким количеством категорий (или классов) работают и как они организованы. Основные типы классификации [9]:

1) Бинарная классификация

Это самый простой вид классификации. В бинарной классификации цель состоит в том, чтобы отсортировать данные по двум различным категориям. Например, система, которая сортирует электронные письма на спам и не спам. Она работает, рассматривая различные характеристики электронного письма, такие как определенные ключевые слова или данные отправителя, и решает, является ли оно спамом или нет. Она выбирает только между этими двумя вариантами.

2) Многоклассовая классификация

Здесь, вместо двух категорий, данные должны быть отсортированы по более, чем двум категориям. Модель выбирает ту, которая лучше всего соответствует входным данным. Например, система распознавания изображений, которая сортирует изображения животных по таким категориям, как кошка, собака и птица.

По сути, машина анализирует особенности изображения (форму, цвет или текстуру) и выбирает, какое животное с наибольшей вероятностью изображено на картинке, основываясь на полученном обучении.

3) Многомарковая классификация

В многомарковой классификации один фрагмент данных может принадлежать нескольким категориям одновременно. В отличие от многоклассовой классификации, где каждая точка данных принадлежит только одному классу, многомарковая классификация позволяет точкам данных принадлежать нескольким классам. Система рекомендаций фильмов может помечать фильм как боевик и как комедию. Система проверяет различные характеристики (например, сюжет фильма, актеры или теги жанра) и присваивает несколько меток одному фрагменту данных, а не только одной.

3. Задача кластеризации

Задача группировки точек данных на основе их сходства друг с другом называется Кластеризацией или Кластерным анализом. Этот метод определяется в ветви неконтролируемого обучения, которая направлена на получение информации из немаркированных точек данных [16].

Например, набор данных о покупательских привычках клиентов. Кластеризация может помочь сгруппировать клиентов со схожим покупательским поведением, которое затем можно использовать для целевого маркетинга, рекомендаций по продуктам или сегментации клиентов.

Типы методов кластеризации

1) Центроидная кластеризация (методы разбиения)

Кластеризация на основе центроида организует точки данных вокруг центральных векторов (центроидов), которые представляют кластеры. Принадлежность каждой точки данных определяется на основе близости к одному из центроидов кластеров. В качестве меры сходства в подобных алгоритмах, как правило, применяются евклидово расстояние, манхэттенское расстояние или обобщённое расстояние Минковского.

Исходное пространство данных разбивается на фиксированное количество кластеров, каждый из которых представлен центральным вектором — центроидом. При вычислении близости входной вектор приравнивается к центроиду, с которым он демонстрирует минимальное различие, и далее относится к соответствующему кластеру.

Одним из ключевых ограничений алгоритмов, основанных на центроидах, является необходимость предварительного задания количества кластеров (параметра k). Это значение может быть определено как эвристически, так и с использованием формальных методов, например, метода локтя. Несмотря на указанный недостаток, такие методы остаются наиболее распространёнными в задачах кластерного анализа благодаря своей вычислительной эффективности и относительной простоте реализации.

Наиболее известные алгоритмы кластеризации на основе центроидов:

- алгоритм K-средних (K-means);
- алгоритм кластеризации на основе K-медоидов (K-medoids).

2) Кластеризация на основе плотности (методы на основе моделей)

Методы кластеризации, основанные на плотности, определяют кластеры как регионы с высокой концентрацией объектов, отделённые друг от друга областями с низкой плотностью. В отличие от центроидных алгоритмов, методы данного типа не требуют предварительного задания количества кластеров и менее чувствительны к начальным условиям. Основные особенности подхода:

- способность выделять кластеры произвольной формы;
- устойчивость к наличию шума и выбросов в данных;
- эффективное разделение кластеров различных размеров и структур;
- применимость к наборам данных, содержащим перекрывающиеся или неоднородные по форме кластеры;
- возможность работы с разреженными и плотными участками пространства признаков;
- использование локальной оценки плотности для выявления разнообразных морфологических структур кластеров.

3) Кластеризация на основе связности (иерархическая кластеризация)

Кластеризация на основе связности представляет собой метод, формирующий иерархическую структуру кластеров на основе попарных расстояний между элементами. Для представления результатов используется дендрограмма – древовидная диаграмма, отражающая последовательность объединения кластеров.

Алгоритм начинает работу с того, что каждый объект исходного набора рассматривается как отдельный кластер. На каждом шаге производится объединение наиболее близких пар кластеров в соответствии с выбранной метрикой сходства или различия. Процесс продолжается до тех пор, пока все элементы не будут объединены в один общий кластер, находящийся на вершине иерархии.

Данный подход позволяет получить многоуровневое представление о структуре данных и является полезным инструментом для анализа сложных зависимостей между объектами.

4) Кластеризация на основе распределения

Методы кластеризации на основе распределения предполагают, что данные порождаются смесью вероятностных распределений, таких как нормальное (гауссовское), пуассоновское и другие. Задача заключается в оценке параметров этих распределений для выделения групповых структур в данных.

Основные характеристики подхода:

- каждый кластер моделируется собственным законом распределения;
- принадлежность объекта к кластеру определяется вероятностными весами;
- по сравнению с расстояниемными методами (например, K-средних), данный подход обеспечивает более гибкое представление о формах, размерах и плотности кластеров.

Такие методы особенно актуальны при работе с данными, имеющими естественную статистическую природу – например, сенсорными, финансовыми или биомедицинскими. Наиболее распространённым представителем данного

класса является модель смеси гауссовых распределений (Gaussian Mixture Model , GMM).

5) Нечеткая кластеризация

Нечеткая кластеризация относится к мягким методам группировки, позволяя каждому объекту одновременно принадлежать нескольким кластерам с различной степенью уверенности. Основные принципы:

- каждой точке данных присваиваются коэффициенты принадлежности, принимающие значения в диапазоне [0, 1] для каждого кластера;
- суммарное значение степеней принадлежности объекта ко всем кластерам обычно равно единице;
- такие коэффициенты отражают вероятность или степень принадлежности точки конкретному кластеру.

Этот подход широко используется в задачах, где чёткое разделение на кластеры невозможно или нецелесообразно, например, при анализе неопределённых или пересекающихся групп.

4. Задача уменьшения размерности

Снижение размерности – это процесс уменьшения количества признаков (или измерений) в наборе данных с сохранением как можно большего количества информации.

Существует несколько методов снижения размерности:

1) Анализ главных компонент (PCA)

Он работает путем преобразования многомерных данных в пространство с меньшей размерностью, при этом максимизируя дисперсию (или разброс) данных в новом пространстве. Это помогает сохранить наиболее важные закономерности и взаимосвязи в данных.

Приоритет отдается направлениям, в которых данные различаются сильнее всего (поскольку больше различий = больше полезной информации).

2) Сингулярное разложение (SVD)

Метод, используемый в линейной алгебре для разложения матрицы на три более простые матрицы, что упрощает анализ и обработку.

3) Линейный дискриминантный анализ (LDA)

LDA работает путем определения линейной комбинации признаков, которая разделяет или характеризует два или более классов объектов или событий. LDA делает это путем проецирования данных с двумя или более измерениями в одно измерение, чтобы их можно было легче классифицировать. Поэтому этот метод иногда называют уменьшением размерности. Эта универсальность гарантирует, что LDA может использоваться для задач классификации многоклассовых данных, в отличие от логистической регрессии, которая ограничена бинарной классификацией. Таким образом, LDA часто применяется для улучшения работы других алгоритмов классификации обучения, таких как дерево решений, случайный лес или опорные векторные машины (SVM).

1.4 Современное состояние средств искусственного интеллекта

Технологии ИИ достигли значительных успехов в различных областях, включая обработку естественного языка, распознавание изображений и автоматизацию. Эти разработки привели к созданию интеллектуальных систем, которые могут выполнять задачи, которые когда-то считались исключительно человеческими возможностями.

Текущее состояние искусственного интеллекта характеризуется существенным ростом его принятия среди крупных компаний. Согласно последнему отчету Artificial Intelligence Index, принятие ИИ среди крупных компаний выросло на впечатляющие 47% [3].

Этот всплеск внедрения подчеркивает растущее признание потенциала ИИ для преобразования различных отраслей и улучшения бизнес-операций. Крупные компании все чаще используют технологии ИИ для получения конкурентных преимуществ, оптимизации процессов и принятия решений на основе данных.

Искусственный интеллект добился значительных успехов в последние годы, преобразовав различные отрасли и аспекты повседневной жизни. Текущее состояние ИИ:

1) Достижения в области машинного обучения: основа ИИ, машинное обучение, значительно продвинулась вперед. Такие алгоритмы, как глубокое обучение и обучение с подкреплением, позволили системам ИИ обрабатывать огромные объемы данных и со временем улучшать свою производительность. Эти улучшения привели к прорывам в распознавании изображений, обработке естественного языка и автономных транспортных средствах.

2) Обработка естественного языка (NLP): NLP, подмножество ИИ, достигло впечатляющих вех. Такие модели, как GPT-4 (Generative Pre-trained Transformer 4), могут генерировать текст, похожий на человеческий, делая чат-ботов, создание контента и языковой перевод более эффективными и естественными.

3) Автономные системы: автономные системы на базе ИИ достигли успехов в таких областях, как беспилотные автомобили и дроны. Такие компании, как Tesla и Waymo, тестируют беспилотные автомобили на дорогах общего пользования, демонстрируя потенциал ИИ для революционных преобразований в сфере транспорта.

4) Революция в здравоохранении: ИИ играет ключевую роль в здравоохранении. От обнаружения заболеваний до открытия лекарств алгоритмы ИИ помогают врачам точнее диагностировать заболевания и ускоряют процессы разработки лекарств.

5) Персонализированный опыт: ИИ улучшает пользовательский опыт, персонализируя контент и рекомендации. Стриминговые сервисы, платформы социальных сетей и сайты электронной коммерции используют алгоритмы ИИ для адаптации предложений на основе предпочтений и поведения пользователя.

6) Проблемы этики и предвзятости: по мере того, как ИИ становится все более распространенным, проблемы этики и предвзятости выходят на первый план. Обеспечение справедливости, прозрачности и отсутствия дискриминационных предвзятостей в системах ИИ остается проблемой, которую отрасль активно решает.

7) Ограничения и проблемы: Хотя ИИ достиг значительного прогресса, он все еще сталкивается с проблемами. Системы ИИ могут испытывать трудности в ситуациях, для которых они не были специально обучены, и могут не иметь способностей к здравому смыслу. Кроме того, потребление энергии при обучении больших моделей ИИ вызывает все большую озабоченность.

8) ИИ в творчестве: ИИ делает успехи и в творческих областях. Искусство, музыка и литература, созданные ИИ, привлекают внимание, стирая границы между человеческим и машинным творчеством.

9) Бизнес-интеграция: ИИ интегрируется в различные отрасли, от финансов до розничной торговли. Компании используют ИИ для таких задач, как предиктивная аналитика, обнаружение мошенничества и оптимизация цепочки поставок.

10) Исследования и инновации: Исследования в области ИИ продолжаются, ученые и инженеры постоянно расширяют границы того, чего может достичь ИИ. По мере развития технологий в ближайшем будущем ожидаются новые прорывы.

Хотя ИИ достиг значительного прогресса в различных областях, важно прояснить его текущий статус по отношению к Теории разума.

На данный момент ИИ не достиг полностью состояния Теории разума. Хотя системы ИИ могут обрабатывать огромные объемы данных, распознавать закономерности и даже заниматься сложной обработкой естественного языка, им не хватает истинного понимания человеческих эмоций, намерений и сознания. Текущие системы ИИ преуспевают в узких задачах, но испытывают трудности с обобщением или пониманием контекста за пределами своих обучающих данных.

1.5 Этические аспекты применения систем искусственного интеллекта

Искусственный интеллект по мере своего развития, вызывает серьезные этические опасения относительно его использования, владения, ответственно-

сти и долгосрочных последствий для человечества. Ниже представлен обзор некоторых наиболее острых этических проблем, связанных с ИИ сегодня.

1) Предвзятость и дискриминация.

Системы искусственного интеллекта обучаются на больших объемах данных, в которых могут быть отражены существующие в обществе предубеждения. В результате эти систематические смещения могут быть воспроизведены или даже усилены алгоритмами ИИ, что приводит к несправедливым или дискриминационным решениям в ключевых областях, таких как трудоустройство, кредитование, правосудие и распределение ресурсов.

Например, если компания внедряет ИИ для автоматического анализа резюме при подборе кандидатов на вакансию, система, как правило, обучается на исторических данных о прошлых наймах. Однако если эти данные содержат гендерные, расовые или иные виды предвзятости, алгоритм может усвоить и воспроизводить такие стереотипы, исключая из рассмотрения кандидатов, не соответствующих сложившимся шаблонам найма.

2) Прозрачность и подотчетность.

Многие системы искусственного интеллекта функционируют в режиме «черного ящика», то есть их внутренние процессы принятия решений недоступны или слабо интерпретируемы для внешнего наблюдателя. Это создает трудности в обеспечении прозрачности и подотчетности, поскольку заинтересованные стороны часто не имеют четкого представления о том, каким образом система пришла к тому или иному выводу. В таких критических областях, как здравоохранение или автономные транспортные средства, прозрачность имеет жизненно важное значение для выяснения того, как принимаются решения, и кто несет за них ответственность. Прояснение ответственности особенно важно, когда системы ИИ допускают ошибки или причиняют вред, гарантируя возможность принятия соответствующих корректирующих мер.

3) Творчество и Право собственности.

Когда художник завершает картину, он владеет ею. Но когда человек-создатель создает произведение цифрового искусства, вводя текстовую под-

сказку в систему ИИ, запрограммированную отдельным лицом или организацией, все не так ясно. Кому принадлежит созданное ИИ искусство? Кто может его коммерциализировать? Кто подвержен риску нарушения?

4) Социальная манипуляция и дезинформация.

Фейковые новости, дезинформация и искажение информации являются обычным явлением в политике, конкурентном бизнесе и многих других областях. Алгоритмы ИИ могут использоваться для распространения этой дезинформации, манипулирования общественным мнением и усиления социальных разногласий. Например, такие технологии, как deepfakes, которые способны генерировать реалистичный, но сфабрикованный аудиовизуальный контент, представляют значительные риски для вмешательства в выборы и политической стабильности.

5) Конфиденциальность, безопасность и наблюдение.

Эффективность ИИ часто зависит от доступности больших объемов персональных данных. По мере расширения использования ИИ возникают опасения относительно того, как эта информация собирается, хранится и используется.

2 МЕТОДЫ МАШИННОГО ОБУЧЕНИЯ

2.1 Основные понятия моделирования на данных

Моделирование данных представляет собой процесс создания визуального представления всей информационной системы или её отдельных компонентов с целью определения связей между элементами данных и структурами.

Цель моделирования заключается в демонстрации типов данных, используемых и хранимых в системе, взаимосвязей между ними, способов группировки и организации информации, а также форматов и атрибутов данных.

Как и при любом проектировании, разработка баз данных и информационных систем начинается с высокоуровневого обобщённого представления и постепенно переходит к более детализированному и конкретному. Модели данных обычно классифицируются на три категории, различающиеся степенью абстракции. Процесс моделирования, как правило, начинается с концептуальной модели, затем переходит к логической и завершается физической моделью [15].

Типы моделей данных:

1) Концептуальные модели данных.

Предоставляет общее описание состава системы и её функциональных характеристик. Такие модели разрабатываются на этапе сбора начальных требований к проекту. Они включают классы сущностей, их атрибуты, ограничения, отношения между сущностями, а также учитывают требования к безопасности и целостности данных. (рис. 1).

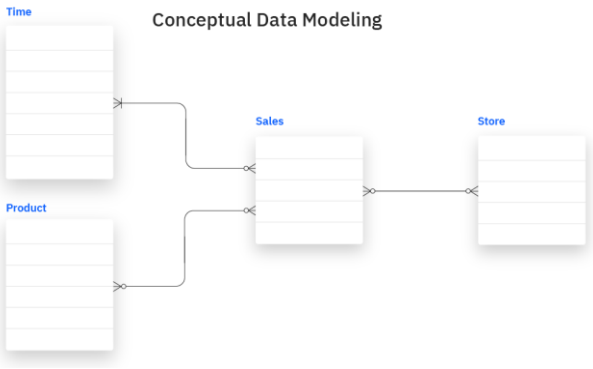


Рисунок 1 – Концептуальные модели данных

2) Логические модели данных.

Отличается меньшей степенью абстракции и содержит более подробное описание понятий и связей предметной области. Для описания используется одна из стандартизированных систем обозначений моделирования данных. Логическая модель определяет атрибуты данных, включая их типы и длины, а также связи между сущностями. На этом уровне не рассматриваются технические аспекты реализации. В гибких методологиях разработки (например, DevOps) этот этап часто пропускается. Однако логическое моделирование может быть полезным в регламентированных средах и для проектов, ориентированных на данные, таких как проектирование хранилищ данных или разработка систем отчетности. (рис. 2).

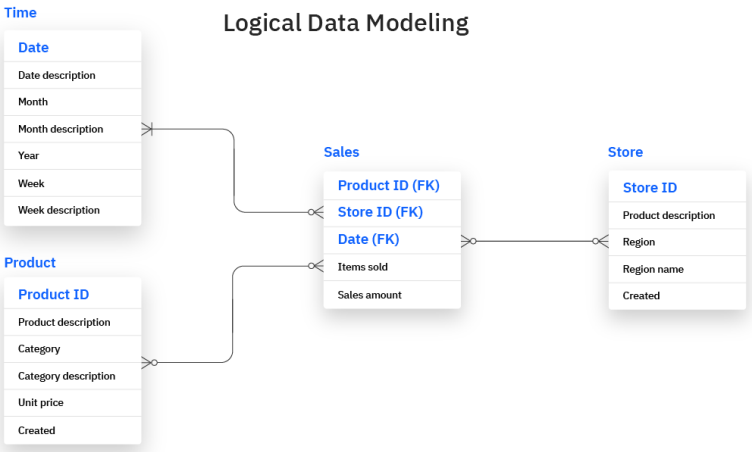


Рисунок 2 – Логические модели данных

3) Физические модели данных.

Является наиболее детализированной и отражает реальное представление данных в системе хранения. Она служит основой для реализации реляционной базы данных и включает таблицы, первичные и внешние ключи, а также связи между сущностями. Кроме того, физическая модель может содержать параметры, специфичные для конкретной системы управления базами данных (СУБД), такие как настройки производительности. (рис. 3).

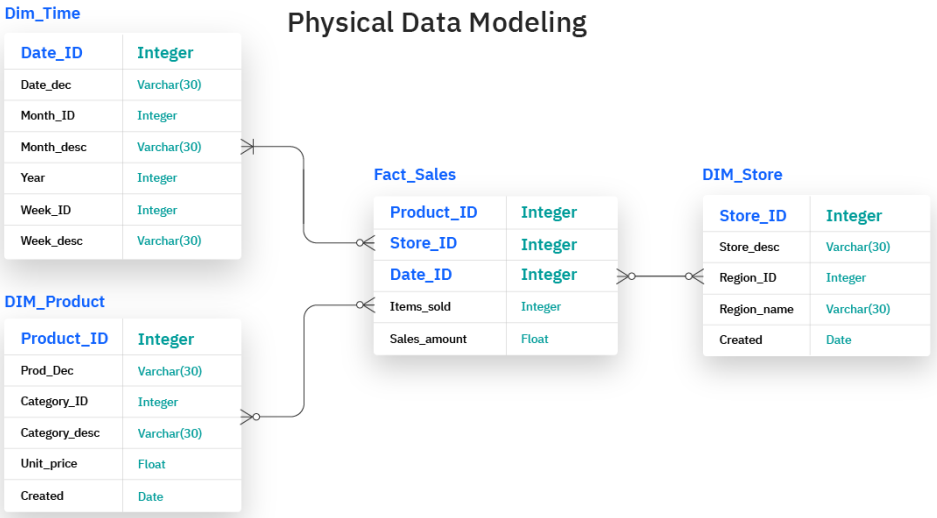


Рисунок 3 – Физические модели данных

Развитие моделей данных происходило параллельно с эволюцией систем управления базами данных, при этом сложность моделей возрастала в соответствии с растущими потребностями организаций в эффективном хранении и обработке информации.

4) Иерархические модели данных.

Описывает связи типа «один ко многим» в виде древовидной структуры. Каждая запись имеет одного родителя (корневой элемент), связанного с одной или несколькими дочерними записями. Эта модель была реализована в IBM Information Management System (IMS), выпущенной в 1966 году и получившей широкое распространение, особенно в банковской сфере. Несмотря на уступку современным подходам по эффективности, иерархическая модель продолжает использоваться в системах XML и геоинформационных системах (ГИС).

5) Реляционные модели данных.

Впервые предложена исследователем IBM Э. Ф. Коддом в 1970 году. Реляционный подход остаётся одним из самых популярных в корпоративных системах. Он не требует знания физических характеристик хранилища, поскольку данные структурируются в виде таблиц, что позволяет значительно упрощать работу с информацией и снижать уровень сложности баз данных.

2.2 Классические методы машинного обучения

Классическое машинное обучение (Classical Machine Learning, ML) — это совокупность методов и подходов к анализу данных, которые позволяют строить модели на основе решения типовых задач, повторяющихся в обучающих выборках [6].

1) Контролируемое обучение (обучение с учителем).

Этот подход предполагает участие разработчика, который отвечает за разметку обучающих данных, а также за определение правил и границ функционирования алгоритмов. Обучение осуществляется на основе набора данных, для которых заранее известны целевые значения. Основные задачи обучения с учителем включают:

- классификацию — определение категории, к которой относится объект;
- регрессию — прогнозирование числового значения на основе входных признаков.

Алгоритмы контролируемого обучения обучаются с использованием маркированных данных, что означает, что входные данные помечены правильным выходом. Цель этих алгоритмов – изучить сопоставление входов с выходами, что позволяет предсказывать выход для новых данных. Обычные алгоритмы контролируемого обучения включают:

– Линейная регрессия: используется для прогнозирования непрерывных результатов. Она моделирует связь между зависимой переменной и одной или несколькими независимыми переменными путем подгонки линейного уравнения к наблюдаемым данным.

– Логистическая регрессия: используется для задач бинарной классификации (например, прогнозирование результатов «да/нет»). Оценивает вероятности с помощью логистической функции.

– Деревья решений: эти модели прогнозируют значение целевой переменной, изучая простые правила принятия решений, выведенные из характеристик данных.

– Случайные леса: совокупность деревьев решений, обычно используемых для классификации и регрессии, повышающих точность модели и контроль переобучения.

2) Обучение без учителя (Unsupervised Learning).

Данный подход не предполагает прямого контроля со стороны разработчика, а желаемые выходные значения заранее не определены. Алгоритм самостоятельно выявляет скрытые закономерности и структуры в данных. Для обучения используются неразмеченные наборы данных, то есть данные, не сопровождённые целевыми метками.

К основным задачам обучения без учителя относятся:

- кластеризация — группировка объектов на основе сходства их признаков;
- поиск ассоциативных правил — выявление взаимосвязей между элементами в наборах данных;
- обнаружение аномалий — выявление редких или отклоняющихся от нормы объектов.

Алгоритмы неконтролируемого обучения используются с наборами данных без маркированных ответов. Цель здесь — вывести естественную структуру, присутствующую в наборе точек данных. Распространенные методы неконтролируемого обучения включают:

– Кластеризация: такие алгоритмы, как K-средние, иерархическая кластеризация и DBSCAN, группируют набор объектов таким образом, что объекты в одной группе более похожи друг на друга, чем на объекты в других группах.

– Ассоциация: эти алгоритмы находят правила, описывающие большие фрагменты ваших данных, такие как анализ потребительской корзины.

– Анализ главных компонент: статистическая процедура, которая использует ортогональное преобразование для преобразования набора наблюдений возможно коррелированных переменных в набор значений линейно некоррелированных переменных.

– Автокодировщики: особый тип нейронной сети, используемый для обучения эффективному кодированию немаркированных данных.

3) Обучение с частичным привлечением учителя (Semi-Supervised Learning)

Этот подход представляет собой гибрид контролируемого и неконтролируемого обучения, сочетающий их основные преимущества. Процесс обучения условно разделяется на два этапа:

- Инициализация модели на основе размеченных данных – на начальном этапе модель обучается распознавать ключевые признаки и закономерности с использованием небольшого объема размеченных данных;
- Самостоятельное обучение на неразмеченных данных – после начальной настройки модель применяет полученные знания для анализа и обобщения на основе большого объема неразмеченных данных.

Таким образом, алгоритм использует ограниченное количество разметки в качестве отправной точки и далее самостоятельно развивает свои предсказательные способности. Этот метод особенно эффективен в условиях, когда разметка данных затруднена или требует значительных ресурсов.

4) Обучение с подкреплением. Алгоритмы обучения с подкреплением учатся принимать последовательность решений. По своей сути обучение включает агента, который принимает решения в динамической среде для достижения набора целей, стремясь максимизировать кумулятивные вознаграждения. В отличие от традиционных парадигм машинного обучения, где модели обучаются на фиксированном наборе данных, агенты RL обучаются на постоянной обратной связи и совершенствуются по мере взаимодействия со своей средой.

Обучение с подкреплением способствует созданию высокоперсонализированных решений, таких как адаптивные системы рекомендаций, которые развиваются вместе с поведением пользователя. Его способность оптимизировать распределение ресурсов приносит пользу различным отраслям, что приводит к экономии средств и повышению эффективности. Масштабируемость обучения

с подкреплением позволяет применять его в различных областях, от простых задач до сложных систем, без существенных изменений.

2.3 Подготовка данных для обучения

Подготовка данных имеет решающее значение в любом проекте машинного обучения, поскольку она напрямую влияет на производительность и точность вашей модели.

Подготовка данных для машинного обучения – это процесс очистки, преобразования и организации необработанных данных в формат, понятный алгоритмам машинного обучения. В машинном обучении алгоритм учится на предоставленных ему данных и может эффективно обучаться только в том случае, если эти данные чистые и полные.

Без хорошо подготовленных данных даже самые передовые алгоритмы могут давать неточные или вводящие в заблуждение результаты.

2.3.1 Источники данных

В качестве источников данных могут выступать:

Данные API;

Данные электронных таблиц;

Данные Веб-сайтов;

Базы данных SQL;

Другим важным аспектом является тип собираемой вами информации. С точки зрения науки о данных все различные формы данных делятся на три большие группы: структурированные, полуструктурированные и неструктурированные.

1) Структурированные данные – это хорошо организованная информация, намеренно размещенная в таблицах со столбцами и строками. Распространенными примерами являются списки инвентаря или электронные таблицы, фиксирующие транзакции и их атрибуты.

2) Полуструктурированные данные не так строго отформатированы, как табличные, но при этом сохраняют идентифицируемые элементы – такие как

теги и другие маркеры – которые упрощают поиск. Форматы, принадлежащие к этой категории, включают файлы JSON, CSV и XML.

3) Неструктурированные данные не имеют схемы, то есть не имеют предопределенной структуры и хранятся в своем собственном формате. Это включает изображения, текстовые документы, видео- и аудиофайлы. Обычно неструктурированные данные легко получить, но с ними трудно работать. Для их обработки требуется много места для хранения и передовые, ресурсоемкие методы машинного обучения, такие как обработка естественного языка (NLP) или распознавание изображений

2.3.2 Предобработка данных

Предварительная обработка данных – это процесс оценки, фильтрации, обработки и кодирования данных, чтобы алгоритм машинного обучения мог их понять и использовать полученный результат. Основная цель предварительной обработки данных – устранить проблемы с данными, такие как пропущенные значения, улучшить качество данных и сделать данные полезными для целей машинного обучения.

В качестве предварительной обработки данных используется следующее:

1) Кодирование данных. Нечисловые данные могут вызывать ошибки у модулей машинного обучения. Чтобы избежать проблем в дальнейшем, данные следует упорядочить в числовом виде. Для решение этой задачи необходимо преобразовать все текстовые значения в числовую форму.

2) Устранение дублирующих записей. Дубликаты могут приводить модель к смещению, что в следствии вызовет переобучение.

3) Масштабирование и нормализация. Зависимые и независимые переменные изменяются по отдельным шкалам, или одна из них изменяется линейно, а другая – экспоненциально. Нормализация и масштабирование помогают модифицировать данные таким образом, чтобы компьютеры могли извлекать значимую связь между этими переменными.

Корреляция данных. Корреляция – это мера связи между переменными. Она измеряет, в какой степени одна переменная зависит от изменения другой

переменной. Это позволяет определять наиболее правильные факторы для исследования и построения модели.

2.3.3 Разделение данных на тренировочный, проверочный и тестовый наборы

Основной задачей машинного обучения является разработка и исследование алгоритмов, способных обучаться на основе данных и делать прогнозы или принимать решения. Такие алгоритмы строят математические модели, основываясь на входных данных, и используют их для анализа новых примеров.

Для построения и оценки моделей данные, как правило, разделяются на несколько наборов: обучающий, проверочный и тестовый. Каждый из этих наборов выполняет свою функцию на разных этапах процесса обучения.

Обучающий набор (Training Set).

Обучающий набор используется для построения и настройки модели. Алгоритм анализирует эти данные, чтобы определить зависимости и подстроить свои параметры. Размер обучающего набора должен быть достаточным для получения устойчивых закономерностей, но не слишком большим, чтобы избежать переобучения – ситуации, когда модель запоминает обучающие примеры, но теряет способность к обобщению на новых данных.

Чтобы модель могла корректно обрабатывать новые, ранее не виденные данные, обучающая выборка должна быть репрезентативной относительно всей совокупности данных. Это позволяет предотвратить систематическое смещение и повысить обобщающую способность модели.

Проверочный набор (Validation Set).

Проверочный набор служит для оценки производительности модели в ходе обучения и для настройки её гиперпараметров. Он позволяет выявлять такие проблемы, как переобучение, и принимать корректирующие меры. Анализ модели на проверочных данных даёт представление о том, насколько хорошо она способна обобщать информацию, а также помогает выбрать наиболее эффективную конфигурацию модели.

Тестовый набор (Test Set).

Тестовый набор предназначен для финальной оценки качества обученной модели. Он представляет собой независимую выборку, на которой модель никогда ранее не обучалась, что позволяет получить объективную оценку её способности работать с новыми данными.

Сохранение тестового набора отдельно от обучающих и проверочных данных обеспечивает достоверное измерение обобщающей способности модели. Оценка на тестовой выборке показывает, насколько точно модель усвоила закономерности и может ли она давать корректные прогнозы вне контекста обучения.

Эта структура разделения данных на обучающий, проверочный и тестовый наборы является стандартной практикой в машинном обучении и играет ключевую роль в построении надёжных и эффективных моделей.

2.4 Задачи бинарной классификации: постановка задачи и метрики качества

Бинарная классификация – это классификация с категориальной выходной переменной, которой может принимать только два значения. Чаще всего это такая переменная принимает значения 1 и 0. Бинарные классификационные модели являются более понятными и интерпретируемыми.

Ключевые компоненты моделей бинарной классификации:

1) Переменные признаков или входные переменные – это атрибуты экземпляров, подаваемых в модель машинного обучения. Они могут включать набор данных, который охватывает числовые значения, категориальные данные. Качество этих переменных значительно влияет на производительность модели. Переменные признаков также можно считать обучающими данными. Качество обучающих наборов данных влияет на результаты и оценку точности, которые будет выдавать модель машинного обучения. Например, в модели бинарной классификации, прогнозирующей спам-письма, признаки могут включать длину письма, наличие определенных ключевых слов и адрес отправителя.

2) Целевые переменные, или метки, – это классы, нужно предсказать. В двоичной классификации каждый экземпляр принадлежит к одному из двух классов. Например, целевой переменной для обнаружения спама будет то, является ли электронное письмо спамом (1) или нет (0). Целевые переменные можно считать результатами обучения и статистического анализа модели машинного обучения. Чем более качественные данные вводятся, тем лучше будут результаты обучения, особенно при применении к новым данным. Если результат теста содержит много ложноположительных или ложноотрицательных результатов, то необходима переоценка переменных признаков. Ключевым моментом является избежание высокого уровня положительных и высокого уровня отрицательных результатов, поскольку это гарантирует хороший результат теста. Точный результат теста необходим для возможности интерпретировать информацию, которую предоставляет модель машинного обучения. При глубоком обучении важно отслеживать уровень истинных положительных результатов и уровень ложных положительных результатов. Ложноотрицательные результаты могут искажать результаты. Истинные положительные и истинные отрицательные результаты –это метрики для измерения, поскольку они дают уверенность в том, что будут правильные результаты.

В основе оценки качества классификационных моделей лежит статистика результатов классификации обучающих примеров.

Для оценки качества классификационных моделей в задачах машинного обучения широко используется матрица ошибок (confusion matrix) — это таблица, отражающая соответствие между предсказанными моделью и фактическими значениями. Матрица содержит четыре основные комбинации прогнозов и реальных результатов.

В данной матрице различают следующие типы исходов:

True Positive (TP) — истинно положительный результат: модель правильно предсказала положительный исход.

False Negative (FN) — ложноотрицательный результат: модель предсказала отрицательный исход, однако на самом деле он был положительным (ошибка второго рода).

False Positive (FP) — ложноположительный результат: модель предсказала положительный исход, но на самом деле он был отрицательным (ошибка первого рода).

True Negative (TN) — истинно отрицательный результат: модель правильно предсказала отрицательный исход.

Матрица ошибок позволяет детально проанализировать эффективность модели и служит основой для расчёта различных метрик качества классификации, таких как точность, полнота, F-мера и др. [6]

На рисунке 4 представлена графическая интерпретация матрицы ошибок с обозначением всех указанных выше показателей.

1. *Accuracy* – доля правильных ответов алгоритма.

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}. \quad (1)$$

Если задача с неравными классами, то данная метрика бесполезна.

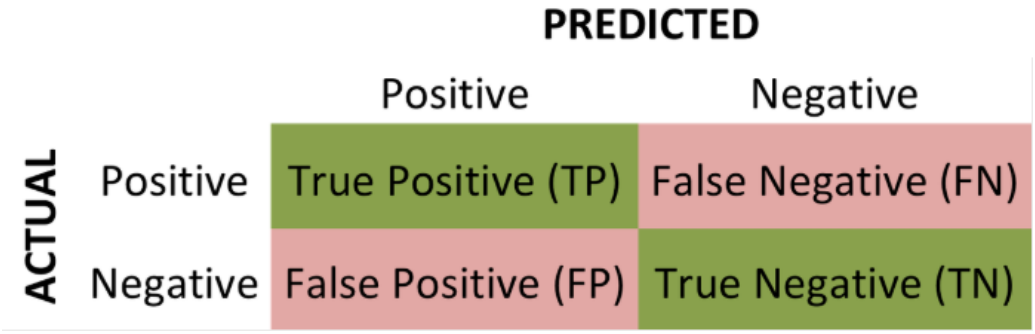


Рисунок 4 – Матрица путаницы

2. *Precision (точность)* – доля правильных ответов модели в пределах класса, т.е. доля объектов, действительно принадлежащих данному классу, относительно всех объектов, которые система отнесла к этому классу.

$$precision = \frac{TP}{TP + FP}. \quad (2)$$

3. *Specificity (специфичность)* – полный аналог точности, но только при предсказании объекта отрицательного класса.

$$specificity = \frac{TN}{TN + FN}. \quad (3)$$

4. *Recall (полнота)* – доля истинно положительных классификаций, она показывает, какая доля объектов, реально относящихся к положительному классу, предсказана верно.

$$recall = \frac{TP}{TP + FN}. \quad (4)$$

5. *F1-мера* – среднее гармоническое precision и recall. Учитывает распределение классов, однако сложно интерпретируемая.

$$F_1 = \frac{2 \cdot precision \cdot recall}{precision + recall}. \quad (5)$$

6. ROC-кривая (Receiver Operating Characteristic curve) представляет собой график, отражающий зависимость между долей истинно положительных результатов (True Positive Rate , TPR) и долей ложноположительных результатов (False Positive Rate , FPR) при варьировании порога классификации. Данный инструмент широко используется для оценки качества бинарных классификаторов.

Кривая строится путём изменения порога принятия решения модели и вычисления соответствующих значений TPR и FPR на тестовой выборке. ROC-кривая позволяет визуализировать компромисс между чувствительностью модели (способностью находить все положительные объекты) и её специфичностью (способностью корректно отсеивать отрицательные объекты).

AUC (Area Under Curve) — это количественная мера качества классификатора, равная площади под ROC-кривой. Значение AUC варьируется от 0 до 1: чем выше значение AUC, тем лучше модель различает классы. Идеальная модель имеет AUC = 1, тогда как случайное угадывание соответствует AUC = 0.5.

На рисунке 5 представлена ROC-кривая с указанием соответствующего значения AUC, что наглядно демонстрирует эффективность классификатора.

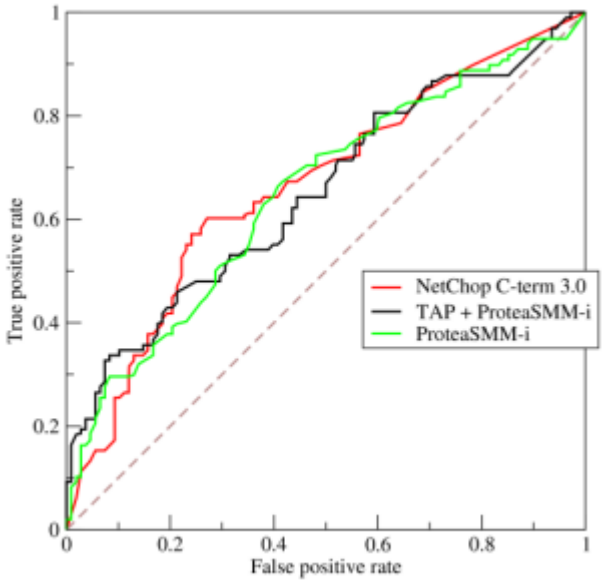


Рисунок 5 – Кривая ошибок

2.5 Описание применяемых методов

В качестве алгоритмов для диагностики диабета будут использованы следующие модели:

1) Наивный байесовский классификатор;

2) Логистическая регрессия;

3) Нейронная сеть.

2.5.1 Наивный байесовский классификатор

Байесовский классификатор – простой вероятностный классификатор, базирующийся на теореме Байеса. Для классифицируемого объекта вычисляются функции правдоподобия каждого из классов, по ним вычисляются апостериорные вероятности классов. Объект относится к тому классу, для которого апостериорная вероятность максимальна [14].

Вероятностная постановка задачи классификации:

$$p(x,y) = p(y) \cdot p(x \mid y), \tag{1}$$

где

X – множество объектов;

Y – конечное множество имён классов;

$X \times Y$ – вероятностное пространство с плотностью распределения.

Априорная вероятность классов:

$$P_y = p(y). \tag{2}$$

Функциями правдоподобия классов:

$$P_y = p(x) = p(x \mid y). \tag{3}$$

Формула (8) показывает вероятность того, что имеется *x*, если классификатор показал *y*.

Наивный метод Байеса представляют собой набор алгоритмов обучения с учителем, основанный на применении теоремы Байеса с «наивным» предположением условной независимости между каждой парой признаков при условии значение переменной класса.

Алгоритм байесовского классификатора:

$$a(x) = \arg\max_{y \in Y} \left(\ln \lambda_y P_y^{'} + \sum_{i=1}^{n} \ln P_{yi}^{'}(\xi_i) \right), \tag{4}$$

где λ_y – означает величина потери при отнесении объекта класса у к данному классу.

Достоинства метода:

1) простота реализации;

2) низкие вычислительные затраты при обучении и классификации;

3) когда признаки почти независимы (в тех редких случаях), наивный байесовский классификатор близок к оптимальному;

4) достаточно малое количество данных для обучения, оценки параметров и классификации.

Недостатки метода – низкое качество классификации в общем случае.

Виды классификаторов наивного Байеса:

1) Гауссовский наивный Байес (naive_bayes.GaussianNB) – работает с непрерывными атрибутами и характеристиками данных, которые следуют распределению Гаусса по всему набору данных.

2) Наивный Байес Бернулли (naive_bayes.BernoulliNB) – Обычно он используется, когда данные являются бинарными, и моделирует возникновение

признаков с использованием распределения Бернулли. Он используется для классификации бинарных признаков, таких как «Да» или «Нет», «1» или «0», «Истина» или «Ложь».

3) Категориальный наивный Байес (naive_bayes.CategoricalNB) – реализует категориальный наивный алгоритм Байеса для категориально распределенных данных. Он предполагает, что каждая функция, которая описывается индексом *i*, имеет свое категориальное распределение.

4) Дополнение наивного Байеса (naive_bayes.ComplementNB) – использует статистику из комплементарного каждого класса для вычисления весов модели.

5) Полиномиальный наивный Байес(naive_bayes.MultinomialNB) – реализует наивный алгоритм Байеса для мультиномиально распределенных данных и является одним из двух классических вариантов наивного алгоритма Байеса, используемых в классификации текстов.

2.5.2 Логистическая регрессия

1) логистическая регрессия – это расчет, используемый для прогнозирования двоичного результата: либо истина, либо ложь [3].

Логистическую регрессию можно использовать для классификации наблюдений с использованием различных типов данных, и она позволяет легко определить наиболее эффективные переменные, используемые для классификации.

Логистическая регрессия предсказывает вероятность принадлежности объекта к одному из двух классов. Она использует логистическую функцию для преобразования прогнозируемых значений между нулем и единицей. Этот алгоритм устанавливает связь между входными переменными и целевой переменной. Логистическая регрессия нашла свое применение в маркетинговых исследованиях, медицинском анализе и банковском деле для понимания взаимосвязей переменных и прогнозирования их вероятных результатов, таких как болезнь или покупки клиентов. Она помогает отвечать на вопросы, касающиеся вероятности событий.

Логистическая регрессия дает вероятностные значения, лежащие между 0 и 1. На рисунке 5 показана логистическая функция:

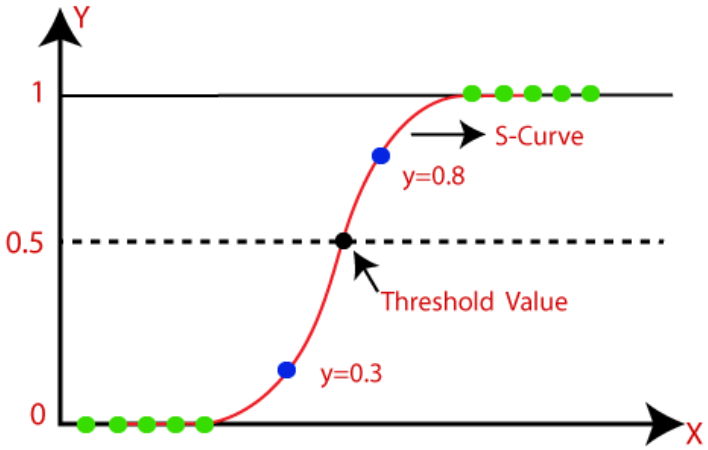


Рисунок 5 – Логистическая регрессия

Вводится так называемая зависимая переменная y, которая принимает одно из двух возможных значений – обычно обозначаемых как 0 (событие не произошло) и 1 (событие произошло). Для предсказания значения этой переменной используется множество независимых переменных (также известных как признаки , предикторы или регрессоры) – вещественных величин x_1, x_2, ..., x_n, На основе наблюдаемых значений этих признаков строится модель, предназначенная для оценки вероятности принятия зависимой переменной того или иного значения.

Существует три типа моделей логистической регрессии, которые определяются на основе категориального ответа:

1) *Бинарная логистическая регрессия*: в этом подходе ответ или зависимая переменная является дихотомической по своей природе, то есть имеет только два возможных результата (например, 0 или 1). Некоторые популярные примеры ее использования включают прогнозирование того, является ли электронное письмо спамом или нет, или является ли опухоль злокачественной или нет. В логистической регрессии это наиболее часто используемый подход, и в

более общем смысле это один из наиболее распространенных классификаторов для бинарной классификации.

2) *Мультиномиальная логистическая регрессия*: в этом типе модели логистической регрессии зависимая переменная имеет три или более возможных результата; однако эти значения не имеют определенного порядка. Например, киностудии хотят предсказать, какой жанр фильма, скорее всего, посмотрит зритель, чтобы более эффективно продвигать фильмы. Мультиномиальная логистическая регрессионная модель может помочь студии определить силу влияния возраста, пола и статуса знакомств человека на тип фильма, который он предпочитает. Затем студия может сориентировать рекламную кампанию определенного фильма на группу людей, которые, скорее всего, пойдут его смотреть.

3) *Порядковая логистическая регрессия*: Этот тип модели логистической регрессии используется, когда переменная ответа имеет три или более возможных результата, но в этом случае эти значения имеют определенный порядок. Примеры порядковых ответов включают шкалы оценок от 1 до 5.

В рамках машинного обучения логистическая регрессия принадлежит к семейству контролируемых моделей машинного обучения. Она также считается дискриминативной моделью, что означает, что она пытается различать классы (или категории). В отличие от генеративного алгоритма, такого как наивный байес, она не может, как следует из названия, генерировать информацию, такую как изображение, класса, который она пытается предсказать (например, изображение кошки).

Логистическая регрессия также может быть склонна к переобучению, особенно когда в модели имеется большое количество переменных-предикторов. Регуляризация обычно используется для штрафования параметров с большими коэффициентами, когда модель страдает от высокой размерности.

2.5.3 Нейронная сеть

Нейронная сеть представляет собой вычислительную модель или программу машинного обучения, имитирующую принципы обработки информации,

характерные для человеческого мозга. Она основана на взаимодействии элементов, называемых нейронами, которые функционируют подобно биологическим нейронам, обеспечивая распознавание паттернов, оценку альтернатив и принятие решений.

Формально нейронная сеть представляет собой совокупность соединённых между собой нейронов через синапсы – связи, передающие сигналы от одного нейрона к другому (рис. 6).

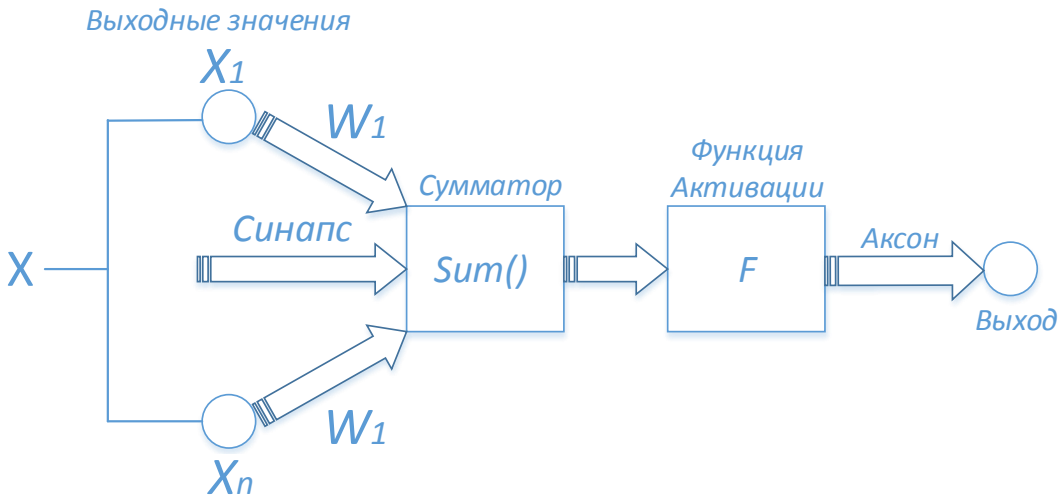


Рисунок 6 – Модель искусственного нейрона

Нейроны (Xi) – базовые вычислительные единицы, являющиеся упрощённой моделью биологического нейрона. Каждый нейрон принимает входные сигналы от предыдущих нейронов, выполняет над ними определённое преобразование и передаёт результат дальше по сети.

Существуют следующие типы нейронов:

- входные нейроны – получают исходные данные из внешней среды;
- скрытые нейроны – осуществляют преобразование информации внутри сети; они могут получать сигналы как от входных, так и от других скрытых нейронов.

После формирования входного слоя каждому входному сигналу присваивается вес, который отражает степень его влияния на выход модели. Более высокие веса указывают на большую значимость соответствующего признака.

Далее значения входов умножаются на свои веса и суммируются. Полученная сумма передается в функцию активации, которая определяет, будет ли активирован данный нейрон. Если значение суммы превышает заданный порог, нейрон «срабатывает» и передаёт сигнал на следующий слой сети.

Этот процесс последовательной передачи данных от входного слоя к выходному определяет тип нейронной сети как сеть прямого распространения (feedforward neural network) , в которой информация движется строго в одном направлении –от входа к выходу, без обратных связей.

Выходные нейроны (получают значения в виде вероятности того или иного действия). Функция, описывающая нейрон, приведена в формуле (5):

$$X_i = \sum\left(W_{i-1}X_{i-1}\right) + W_0 \tag{5}$$

где:

W_0 – смещение;

W_{i-1} – вес от предыдущих нейронов;

X_i – значение текущего нейрона;

X_{i-1} – значение предыдущего нейрона;

Синапсы W_i – веса искусственной нейронной сети.

Веса представляют собой параметры модели, подлежащие обучению, и определяют значимость соответствующего входного признака для формирования выходного значения. Каждый вес отражает силу связи между нейронами.

Сумматор – это функциональный элемент нейрона, в котором вычисляется взвешенная сумма входных сигналов. Формально она рассчитывается как сумма произведений входных значений на соответствующие им веса.

Аксон – представляет собой выходной сигнал нейрона, который передается на вход следующего слоя сети и записывается в соответствующий нейрон выходного слоя.

Функция активации – это математическая функция, применяемая к выходу сумматора. Она вносит нелинейность в модель, что позволяет нейронной сети обучаться сложным зависимостям и эффективно обрабатывать нелинейно

разделимые данные. Без функции активации поведение сети было бы эквивалентно линейной регрессии вне зависимости от количества слоёв.

Функция активации определяет, будет ли нейрон активирован, путём сравнения взвешенной суммы входов с заданным пороговым значением (смещением). Это обеспечивает способность модели принимать нелинейные решения и делать точные прогнозы.

Обучение нейронной сети заключается в постепенной корректировке весовых коэффициентов, связывающих нейроны между собой. Эта процедура выполняется на основе заданных гиперпараметров модели, таких как скорость обучения, количество эпох и структура сети.

Типовая нейронная сеть состоит из трёх типов слоёв:

- входной слой , принимающий исходные данные;
- один или несколько скрытых слоёв, выполняющих преобразование информации;
- выходной слой, предоставляющий результат работы сети.

Каждый нейрон внутри сети имеет собственные веса и пороговое значение. Если выходное значение нейрона превышает установленный порог, он активируется и передаёт сигнал дальше по сети. В противном случае передача сигнала на следующий слой не происходит.

Современные нейронные сети могут иметь от миллионов до более чем миллиарда параметров для настройки с помощью обратного распространения. Им также требуется большой объем обучающих данных для достижения высокой точности, что означает, что сотни тысяч или миллионов входных образцов должны быть пропущены как через прямой, так и через обратный проход. Поскольку нейронные сети создаются из большого количества идентичных нейронов, они по своей природе высокопараллельны. Этот параллелизм естественным образом отображается на графические процессоры, которые обеспечивают значительное ускорение вычислений по сравнению с обучением только на CPU.

Нейронные сети могут быть разделены на различные типы в зависимости от их структуры и области применения. Ниже приведены основные виды архитектур, используемых в машинном обучении.

1) Персептрон.

Персептрон является одной из первых моделей искусственной нейронной сети, предложенной Фрэнком Розенблаттом в 1958 году. Это простая сеть без скрытых слоёв, предназначенная для решения задач линейной классификации.

2) Нейронные сети прямого распространения (Feedforward Neural Networks) или многослойный персептрон (MLP).

Эти сети состоят из входного слоя, одного или нескольких скрытых слоёв и выходного слоя. В отличие от оригинального персептрона, современные MLP используют нелинейные функции активации, такие как сигмоида, ReLU и др., что позволяет им решать сложные, нелинейные задачи.

Данные подаются на вход сети и последовательно проходят через все слои до выходного. Такие модели применяются в различных областях, включая компьютерное зрение и обработку естественного языка, являясь базовой архитектурой для более сложных типов сетей.

3) Свёрточные нейронные сети (Convolutional Neural Networks, CNN).

CNN похожи на сети прямого распространения, но специально разработаны для обработки данных с явной топологической структурой, таких как изображения. Они широко используются в задачах распознавания образов, анализа изображений и компьютерного зрения.

Основной особенностью CNN являются свёрточные слои, которые применяют операции свёртки для выделения признаков из входных данных. Эти операции основаны на принципах линейной алгебры, в частности, умножении матриц, и позволяют эффективно находить локальные закономерности в изображениях.

4) Рекуррентные нейронные сети (Recurrent Neural Networks, RNN).

RNN характеризуются наличием обратных связей, что позволяет им сохранять информацию о предыдущих входных данных. Это делает их особенно

эффективными при работе с последовательностями — например, с временными рядами.

RNN часто применяются в задачах прогнозирования, таких как предсказание цен на фондовом рынке или объемов продаж. Они способны учитывать контекст и временные зависимости, что критично для обработки речи, текста и других последовательностей.

Нейронные сети обучаются на больших объёмах данных, постепенно повышая точность своих предсказаний. После завершения обучения и настройки гиперпараметров они становятся мощным инструментом в области компьютерных наук и искусственного интеллекта.

С их помощью можно быстро выполнять задачи классификации и кластеризации, в то время как аналогичные задачи вручную могли бы занимать часы. Например, автоматическое распознавание речи или изображений может быть выполнено за считанные минуты. Одним из известных примеров использования нейронных сетей является поисковый алгоритм Google, который использует ИИ для улучшения качества результатов поиска.

Преимущества нейронных сетей:

1) Адаптивность. Они могут учиться и принимать самостоятельные решения.

2) Параллельная обработка. Большие сети могут обрабатывать несколько входов одновременно.

3) Отказоустойчивость. Даже если часть сети выйдет из строя, вся сеть все равно сможет функционировать.

Каковы ограничения нейронных сетей:

1) Зависимость от данных. Для эффективного функционирования им требуется большой объем данных.

2) Непрозрачная природа. Часто их называют «черными ящиками», поскольку сложно понять, как они принимают конкретные решения.

3) Переобучение. Иногда они могут запоминать данные, а не учиться на них.

Хотя все модели глубокого обучения (deep learning) относятся к классу нейронных сетей, не каждая нейронная сеть может быть отнесена к глубокому обучению. Термин глубокое обучение применяется к архитектурам с тремя и более слоями, включая входной, выходной и как минимум один скрытый слой.

Такие сети разработаны таким образом, чтобы имитировать работу человеческого мозга, обеспечивая возможность извлечения сложных закономерностей из больших объемов данных. В отличие от однослойных нейронных сетей, способных давать лишь приблизительные прогнозы, добавление дополнительных скрытых слоёв позволяет значительно повысить точность моделей за счёт построения более абстрактных представлений данных.

Таким образом, ключевым фактором, определяющим переход от простой нейронной сети к глубокой, является её глубина – количество слоёв, участвующих в обработке информации.

3 ПРОГНОЗИРОВАНИЕ ЗАБОЛЕВАЕМОСТИ ДИАБЕТОМ КЛАССИЧЕСКИМИ МЕТОДАМИ

3.1 Набор данных «Diabetes prediction»

Набор данных «Diabetes Prediction» содержит медицинские и демографические сведения о пациентах, а также информацию об их диабетическом статусе — положительном или отрицательном. В набор входят такие признаки, как возраст, пол, индекс массы тела (ИМТ), наличие гипертонии, сердечно-сосудистых заболеваний, история курения, уровень гликированного гемоглобина HbA1c и концентрация глюкозы в крови [5].

Данный датасет был разработан индийским специалистом по обработке данных Мохаммедом Мустафой.

Этот набор данных широко используется для построения и тестирования моделей машинного обучения, предназначенных для прогнозирования риска развития диабета у пациентов на основе их анамнеза и демографических характеристик. Это может быть полезно для медицинских работников при выявлении пациентов, которые могут подвергаться риску развития диабета, и при разработке индивидуальных планов лечения. Кроме того, набор данных может использоваться исследователями для изучения взаимосвязи между различными медицинскими и демографическими факторами и вероятностью развития диабета.

Характеристики набора:

Пол относится к биологическому полу человека, который может повлиять на его восприимчивость к диабету;

Возраст является важным фактором, поскольку диабет чаще диагностируется у пожилых людей. В наборе данных возраст варьируется от 0 до 80 лет.

Гипертония – это заболевание, при котором артериальное давление в артериях постоянно повышено. Он имеет значения 0 или 1, где 0 означает, что у них нет гипертонии, а 1 означает, что у них есть гипертония.

Заболевания сердца – еще одно заболевание, связанное с повышенным риском развития диабета. Он имеет значения 0 или 1, где 0 указывает на отсутствие сердечно-сосудистых заболеваний, а 1 означает, что у них есть сердечно-сосудистые заболевания.

История курения также считается фактором риска развития диабета и может усугубить осложнения, связанные с диабетом. В нашем наборе данных есть 5 категорий: «не в настоящее время», «бывший», «Нет информации», «текущий», «никогда и никогда».

ИМТ (индекс массы тела) – это показатель количества жира в организме, основанный на весе и росте. Более высокие значения ИМТ связаны с более высоким риском развития диабета. Диапазон ИМТ в наборе данных составляет от 10,16 до 71,55. ИМТ менее 18,5 – недостаточный вес, 18,5–24,9 – нормальный, 25–29,9 – избыточный вес, 30 и более – ожирение.

Уровень HbA1c (гемоглобин A1c) – это показатель среднего уровня сахара в крови человека за последние 2–3 месяца. Более высокие уровни указывают на больший риск развития диабета. Чаще всего более 6,5% уровня HbA1c указывает на диабет.

Уровень глюкозы в крови относится к количеству глюкозы в кровотоке в данный момент времени. Высокий уровень глюкозы в крови является ключевым индикатором диабета.

Диабет является прогнозируемой целевой переменной: значения 1 указывают на наличие диабета, а 0 – на отсутствие диабета.

Статистические графики представлены на рисунках 7-12.

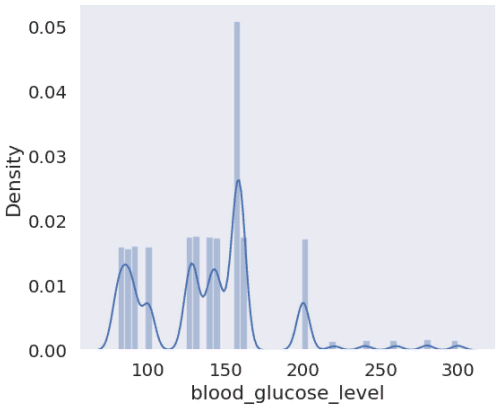


Рисунок 7 – Статистика уровня глюкозы в крови

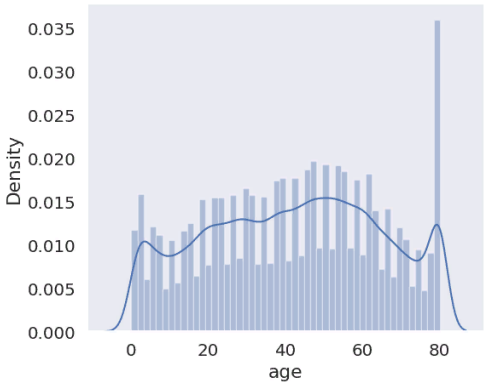


Рисунок 8 – Статистика возрастов

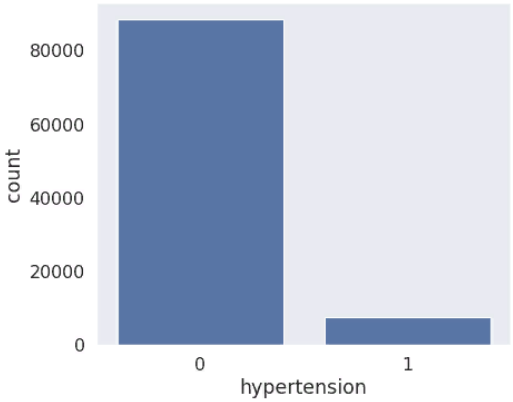


Рисунок 9 – Статистика гипертонии

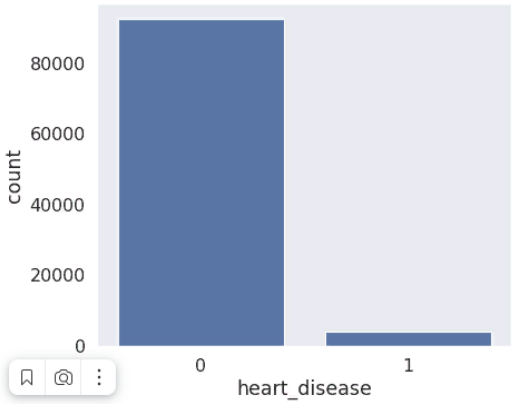


Рисунок 10 – Статистика болезни сердца

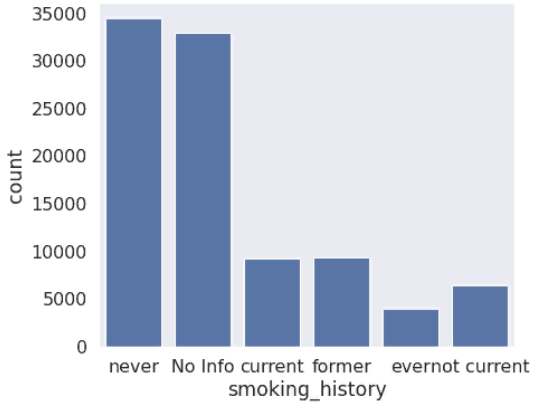


Рисунок 11 – Статистика по курению

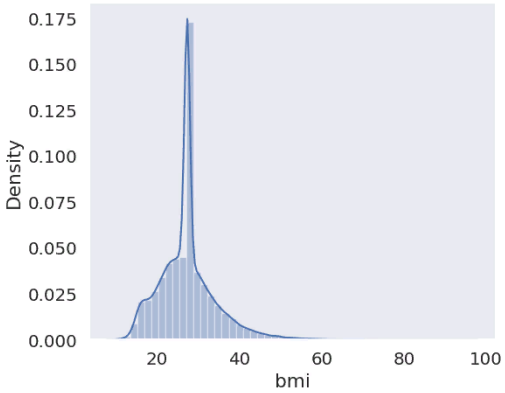


Рисунок 12 – Статистика ИМТ

3.2 Прогнозирование диабета методом наивного Байеса

Алгоритм прогнозирование диабета методом наивного Байеса:

1. Для прогнозирования понадобятся следующие библиотеки:

Numpy – это математическая и научная вычислительная библиотека с открытым исходным кодом для задач программирования на Python. Название NumPy является сокращением от Numerical Python. Библиотека NumPy предлагает набор высокоуровневых математических функций, включая поддержку многомерных массивов, маскированных массивов и матриц. NumPy также включает в себя различные логические и математические возможности для этих массивов, такие как манипуляция формой, сортировка, выбор, линейная алгебра, статистические операции, генерация случайных чисел и дискретные преобразования Фурье.

Pandas – это пакет Python с открытым исходным кодом, который наиболее широко используется для задач нализа данных и машинного обучения. Он создан на основе другого пакета Numpy, который обеспечивает поддержку многомерных массивов. Как один из самых популярных пакетов обработки данных, Pandas хорошо работает со многими другими модулями науки о данных в экосистеме Python и обычно включается в каждый дистрибутив Python, от тех, которые поставляются с вашей операционной системой, до коммерческих дистрибутивов поставщиков, таких как ActivePython от ActiveState.

Matplotlib – кроссплатформенная библиотека визуализации данных и графического построения графиков (гистограмм, диаграмм рассеяния, столбчатых диаграмм и т. д.) для Python и его числового расширения NumPy. Таким образом, она предлагает жизнеспособную альтернативу MATLAB с открытым исходным кодом. Разработчики также могут использовать API (интерфейсы прикладного программирования) matplotlib для встраивания графиков в приложения с графическим интерфейсом.

Sklearn – это библиотека анализа данных с открытым исходным кодом и золотой стандарт для машинного обучения (ML) в экосистеме Python. Ключевые концепции и функции включают:

Seaborn – это мощная библиотека визуализации данных Python, созданная на основе Matplotlib. Она предоставляет высокоуровневый интерфейс для рисования привлекательных и информативных статистических графиков. Seaborn

особенно хорошо подходит для визуализации сложных наборов данных, что делает ее фаворитом среди специалистов по работе с данными и аналитиков. Она упрощает процесс создания эстетически приятных и информативных графиков, которые необходимы для исследования и представления данных в областях ИИ, машинного обучения и науки о данных.

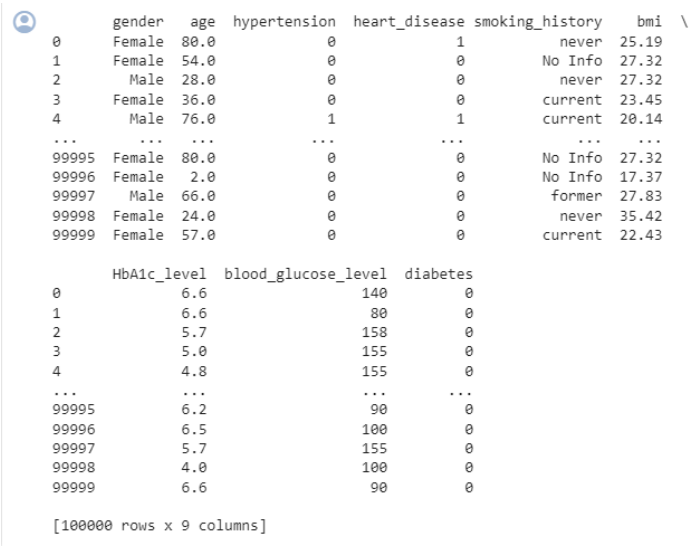
Warnings – Модуль в Python предоставляет способ управления обработкой предупреждений в скрипте Python. Он позволяет разработчикам выдавать предупреждающие сообщения, чтобы оповестить пользователей о потенциальных проблемах или неожиданном поведении в их коде. Он обычно используется, когда метод помечен как устаревший, но может также применяться для предупреждения о неожиданных условиях выполнения.

Tensorflow – это фреймворк с открытым исходным кодом для машинного обучения (ML) и искусственного интеллекта (AI), разработанный Google Brain. Он был разработан для облегчения разработки моделей машинного обучения, в частности моделей глубокого обучения, предоставляя инструменты для их легкого создания, обучения и развертывания на разных платформах.

Keras – это высокоуровневый API глубокого обучения, который упрощает процесс создания глубоких нейронных сетей. Изначально разработанный как независимая библиотека, Keras теперь тесно интегрирован в TensorFlow в качестве его официального высокоуровневого API. Он поддерживает несколько внутренних движков, включая TensorFlow, Theano и Microsoft Cognitive Toolkit (CNTK).

2. Загрузка набора данных.

На рисунке 13 представлен вывод данных



```
       gender   age  hypertension  heart_disease smoking_history    bmi  \
0      Female  80.0             0              1           never  25.19
1      Female  54.0             0              0         No Info  27.32
2        Male  28.0             0              0           never  27.32
3      Female  36.0             0              0         current  23.45
4        Male  76.0             1              1         current  20.14
...       ...   ...           ...            ...             ...    ...
99995  Female  80.0             0              0         No Info  27.32
99996  Female   2.0             0              0         No Info  17.37
99997    Male  66.0             0              0          former  27.83
99998  Female  24.0             0              0           never  35.42
99999  Female  57.0             0              0         current  22.43

       HbA1c_level  blood_glucose_level  diabetes
0              6.6                  140         0
1              6.6                   80         0
2              5.7                  158         0
3              5.0                  155         0
4              4.8                  155         0
...            ...                  ...       ...
99995          6.2                   90         0
99996          6.5                  100         0
99997          5.7                  155         0
99998          4.0                  100         0
99999          6.6                   90         0

[100000 rows x 9 columns]
```

Рисунок 13 – Вывод набора данных

3. Выполняется кодирование столбцов 'gender' и 'smoking_history при помощи метода get_dummies() [20] (рисунок 14). Далее кодированные столбцы объединяются с другими столбцами (Рисунок 15).

	gender_Male	gender_Other	smoking_history_current	smoking_history_ever	smoking_history_former	smoking_history_never
0	False	False	False	False	False	True
1	False	False	False	False	False	False
2	True	False	False	False	False	True
3	False	False	True	False	False	False
4	True	False	True	False	False	False

Рисунок 14 – Кодирование столбцов 'gender' и 'smoking_history

	gender_Male	gender_Other	smoking_history_current	smoking_history_ever	smoking_history_former	smoking_history_never
0	False	False	False	False	False	True
1	False	False	False	False	False	False
2	True	False	False	False	False	True
3	False	False	True	False	False	False
4	True	False	True	False	False	False

Рисунок 15 – Объединение кодированных столбцов с другими столбцами

4. Разделение датасета на обучающий и тестовый.

С помощью функции train_test_split() набор данных разделяется на обучающий и тестовый [21]. В качестве настроек параметров значение test_size принимается равным 0.2, а значение random_state – 10.

На рисунках 16-19 представлены обучающие и тестовые данные.

	gender	age	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	blood_glucose_level
93378	2	49.0	0	0	0	33.72	3.5	140
89781	1	27.0	0	0	5	25.68	4.0	85
5466	1	45.0	0	0	1	21.96	6.2	140
18541	1	63.0	0	0	2	27.32	6.1	126
61935	1	22.0	0	0	0	23.96	4.0	140
...	...	...	...	...	...	...	...	...
496	2	70.0	0	0	1	27.32	6.5	130
49177	2	47.0	0	0	3	29.24	5.8	155
28155	2	57.0	0	0	1	36.24	5.7	140
92602	2	55.0	1	0	0	36.82	5.8	145
62692	2	3.0	0	0	0	27.32	5.8	90

70000 rows × 8 columns

Рисунок 16 – Набор X_train

	gender	age	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	blood_glucose_level
6287	2	62.0	1	0	0	28.22	6.2	145
46255	1	66.0	1	1	0	36.60	5.7	300
23851	1	49.0	0	0	4	35.47	4.8	159
70923	2	20.0	0	0	5	19.56	6.1	90
75435	2	64.0	0	0	0	25.12	4.0	200
...	...	...	...	...	...	...	...	...
57209	1	58.0	1	0	4	20.42	6.6	100
29238	1	29.0	0	0	1	27.32	6.2	126
38369	1	28.0	0	0	0	27.32	8.8	240
78970	1	58.0	0	0	0	26.57	6.5	126
36838	1	23.0	0	0	0	27.32	4.5	126

30000 rows × 8 columns

Рисунок 17 – Набор X_test

```
93378    0
89781    0
5466     0
18541    0
61935    0
        ..
496      0
49177    0
28155    0
92602    0
62692    0
Name: diabetes, Length: 70000, dtype: int64
```

Рисунок 18 – Набор y_train

```
6287     1
46255    1
23851    0
70923    0
75435    0
        ..
57209    0
29238    0
38369    1
78970    0
36838    0
Name: diabetes, Length: 30000, dtype: int64
```

Рисунок 19 – Набор y_test

5. Построение и обучение модели с использованием функции MultinomialNB().

Выполнение диагностики модели с помощью матрицы неточности:

array([[26435, 1059],

[1798, 708]])

Результат диагностики приведен на рисунке 20.

```
количество истинно отрицательных предсказаний: 26435
количество истинно положительных предсказаний: 708
количество ложно отрицательных предсказаний: 1798
количество ложно положительных предсказаний: 1059
```

Рисунок 20 – Диагностика модели

Добавляем метрики качества. Результат метрик приведены на рисунке 21.

```
Accuracy (доля правильных ответов): 90.47666666666667%
Precision (точность): 40.0679117147708%
Recall (полнота): 28.252194732641662%
F1: 33.13831032061783%
```

Рисунок 21 – Метрики качества

6. Построение ROC-кривой.

ROC-кривая отображает процент истинных положительных результатов, предсказанных моделью, поскольку отсечка вероятности прогнозирования снижается с 1 до 0. Чем выше площадь под кривой, тем точнее модель может предсказывать результаты.

ROC-кривая представлена на рисунке 22. Как видно из кривой, показатель AUC составляет 0.74. Аналогично алгоритм проводится для остальных функций, их значения AUC приведены в таблицы 1.

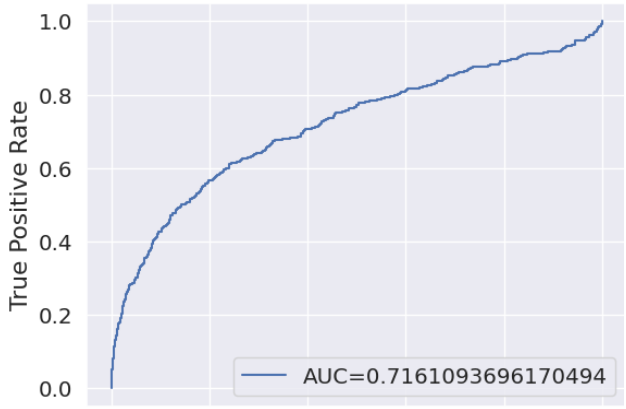


Рисунок 22 – ROC-кривая

Таблица 1 – Показатели AUC

Функция	Показатель AUC
MultinomialNB()	0.72
GaussianNB()	0.92
ComplementNB()	0.72
BernoulliNB()	0.70

Таким образом, самой точной оказалась функция GaussianNB().

3.3 Прогнозирование диабета методом логистической регрессии

Алгоритм прогнозирование диабета методом Логистической регрессии:

1. Импорт необходимых библиотек:

Numpy

Pandas

Matplotlib

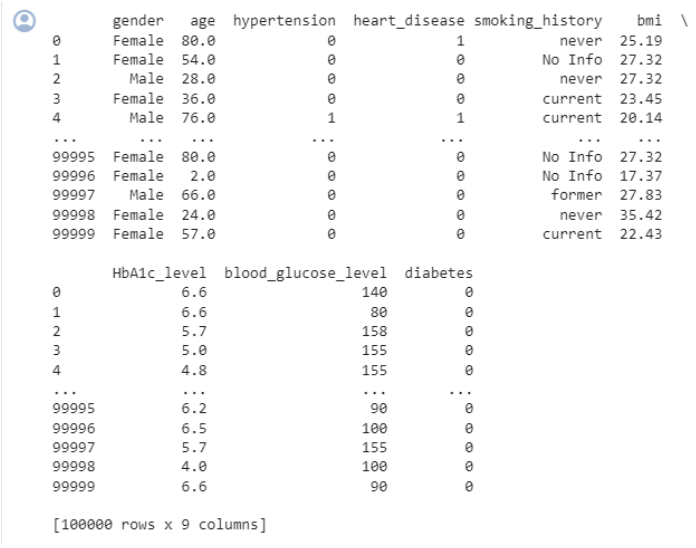
Sklearn

Seaborn

Warnings

2. Загрузка набора данных.

На рисунке 22 представлен вывод данных.



	gender	age	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	blood_glucose_level	diabetes
0	Female	80.0	0	1	never	25.19	6.6	140	0
1	Female	54.0	0	0	No Info	27.32	6.6	80	0
2	Male	28.0	0	0	never	27.32	5.7	158	0
3	Female	36.0	0	0	current	23.45	5.0	155	0
4	Male	76.0	1	1	current	20.14	4.8	155	0
...	...	...	...	...	...	...	...	...	...
99995	Female	80.0	0	0	No Info	27.32	6.2	90	0
99996	Female	2.0	0	0	No Info	17.37	6.5	100	0
99997	Male	66.0	0	0	former	27.83	5.7	155	0
99998	Female	24.0	0	0	never	35.42	4.0	100	0
99999	Female	57.0	0	0	current	22.43	6.6	90	0

[100000 rows x 9 columns]

Рисунок 23 – Вывод набора данных

3. Выполняется кодирование столбцов 'gender' и 'smoking_history при помощи метода get_dummies().

4. Разделение датасета на обучающий и тестовый.

С помощью функции train_test_split() набор данных разделяется на обучающий и тестовый. В качестве настроек параметров значение test_size принимается равным 0.2, а значение random_state – 10.

На рисунках 24-27 представлены обучающие и тестовые данные.

	gender	age	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	blood_glucose_level
93378	2	49.0	0	0	0	33.72	3.5	140
89781	1	27.0	0	0	5	25.68	4.0	85
5466	1	45.0	0	0	1	21.96	6.2	140
18541	1	63.0	0	0	2	27.32	6.1	126
61935	1	22.0	0	0	0	23.96	4.0	140
...	...	...	...	...	...	...	...	...
496	2	70.0	0	0	1	27.32	6.5	130
49177	2	47.0	0	0	3	29.24	5.8	155
28155	2	57.0	0	0	1	36.24	5.7	140
92602	2	55.0	1	0	0	36.82	5.8	145
62692	2	3.0	0	0	0	27.32	5.8	90

70000 rows × 8 columns

Рисунок 24 – Набор X_train

	gender	age	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	blood_glucose_level
6287	2	62.0	1	0	0	28.22	6.2	145
46255	1	66.0	1	1	0	36.60	5.7	300
23851	1	49.0	0	0	4	35.47	4.8	159
70923	2	20.0	0	0	5	19.56	6.1	90
75435	2	64.0	0	0	0	25.12	4.0	200
...	...	...	...	...	...	...	...	...
57209	1	58.0	1	0	4	20.42	6.6	100
29238	1	29.0	0	0	1	27.32	6.2	126
38369	1	28.0	0	0	0	27.32	8.8	240
78970	1	58.0	0	0	0	26.57	6.5	126
36838	1	23.0	0	0	0	27.32	4.5	126

30000 rows × 8 columns

Рисунок 25 – Набор X_test

```
93378    0
89781    0
5466     0
18541    0
61935    0
        ..
496      0
49177    0
28155    0
92602    0
62692    0
Name: diabetes, Length: 70000, dtype: int64
```

Рисунок 26 – Набор y_train

```
6287     1
46255    1
23851    0
70923    0
75435    0
        ..
57209    0
29238    0
38369    1
78970    0
36838    0
Name: diabetes, Length: 30000, dtype: int64
```

Рисунок 27 – Набор y_test

5. Построение и обучение модели с использованием Логистической регрессии.

Выполнение диагностики модели с помощью матрицы неточности:

array([[17390, 144],

[630, 1066]])

Добавляем метрики качества. Результат метрик приведены на рисунке 28.

```
Accuracy (доля правильных ответов): 95.97503900156006%
Precision (точность): 88.09917355371901%
Recall (полнота): 62.85377358490566%
F1: 73.36545079146595%
```

Рисунок 28 – Метрики качества

6. Выводим отчет о классификации

Отчет представлен на рисунке 29.

	precision	recall	f1-score	support
0	0.97	0.99	0.98	17534
1	0.88	0.63	0.73	1696
accuracy			0.96	19230
macro avg	0.92	0.81	0.86	19230
weighted avg	0.96	0.96	0.96	19230

Рисунок 29 – Отчет о классификации

7. Построение ROC-кривой.

ROC-кривая представлена на рисунке 30. Как видно из кривой, показатель AUC составляет 0.96.

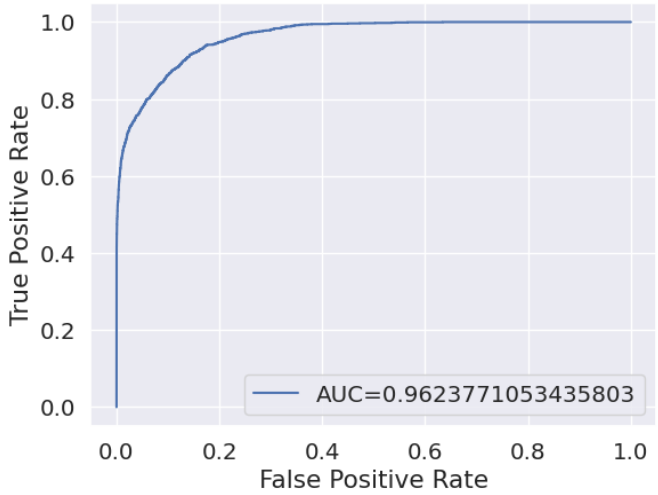


Рисунок 30 – ROC-кривая

3.4 Оценка важности признаков для логистической регрессии

На рисунке 31 представлен график влияния признаков на прогнозирование методом логистической регрессии.

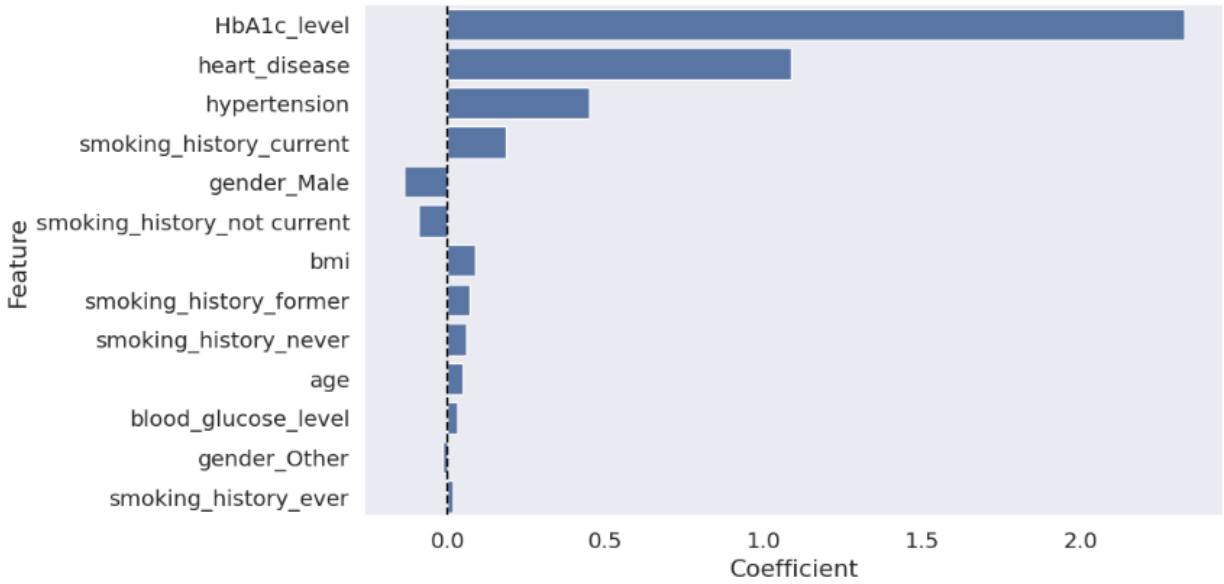


Рисунок 31 – Влияние признаков

Как видно из графика, наиболее важные признаки это blood_glucose_level, HbA1c_level, bmi, age. Наименее – gender, smoking_history

4 НЕЙРОННЫЕ СЕТИ ДЛЯ ПРОГНОЗИРОВАНИЯ ДИАБЕТА

4.1 Описание архитектуры нейронной сети и настройка гиперпараметров

Архитектура нейронной сети представляет собой структуру и организацию искусственной нейронной сети, которая представляет собой вычислительную модель, основанную на человеческом мозге.

Архитектура нейронной сети относится к структурированному расположению узлов (нейронов) и слоев, которые определяют, как искусственная нейронная сеть обрабатывает и обучается на данных. Конструкция ИНС влияет на ее способность изучать сложные шаблоны и эффективно выполнять задачи. Роль архитектуры в нейронных сетях:

1. Проектирование под конкретную задачу;
2. Способность изучать сложные закономерности;
3. Эффективность;
4. Оптимизация;
5. Обобщение модели.

Вывод наилучшего варианта архитектуры проводится экспериментально при манипуляции с настройками гиперпараметрами. Опираясь экспименты с архитектурой и [10], лучший вариант представлен в таблице 2.

Таблица 2 – Выбранная архитектура нейронной сети

Слой	Количество нейронов	Функция активации
Входной	8	-
Скрытый 1	128	relu
Скрытый 2	256	
Скрытый 3	512	
Скрытый 4	1024	
Скрытый 5	2048	
Скрытый 6	4096	
Выходной	1	sigmoid

Выбора оптимальных гиперпараметров проводится также экспериментально. Манипуляции будут проводится с такими параметрами, как функция оптимизации, метрика качества, а также наличие или отсутствие процедкры SMOTE. Варианты представлены в таблицах 3-11, а результаты на рисунках 30-74.

Таблица 3 – Настройки гиперпараметров нейронной сети (Вариант 1)

Гиперпараметр	Значение/настройка
SMOTE	**False**
Доля тестовой выборки	0.2
Функция ошибки	«binary_crossentropy»
Функция оптимизации	**«adam»**
Метрика качества	**«recall»**
Размер подвыборки	300
Доля валидационной выборки	0.2
Количество эпох	40

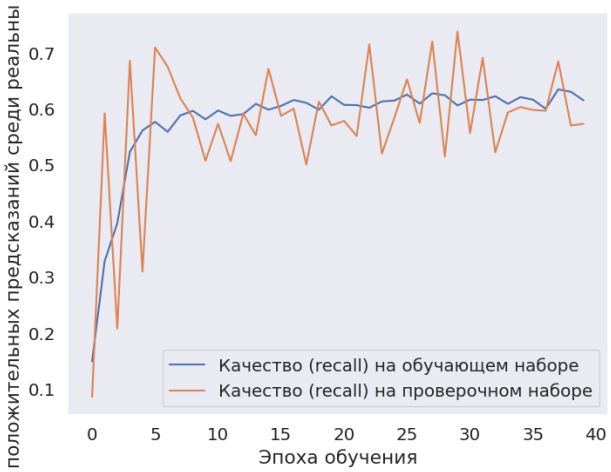


Рисунок 32 – Доля верных положительных предсказаний среди предсказанных положительных

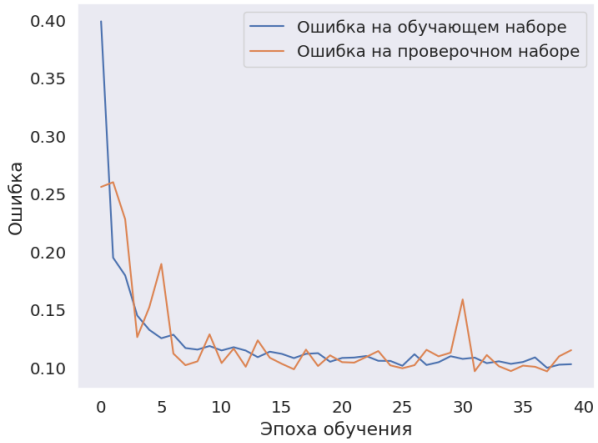


Рисунок 33 – Ошибка на обучающем и проверочном наборах

	precision	recall	f1-score	support
0	0.96	1.00	0.98	17534
1	1.00	0.55	0.71	1696
accuracy			0.96	19230
macro avg	0.98	0.78	0.85	19230
weighted avg	0.96	0.96	0.96	19230

Рисунок 34 – Полный отчет о классификации

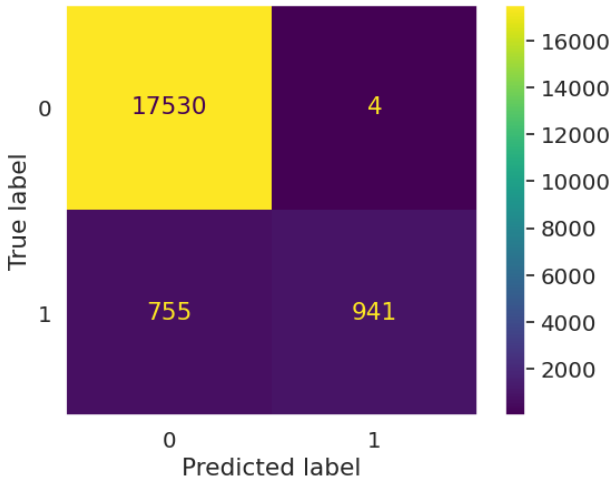


Рисунок 35 – Матрица путаницы

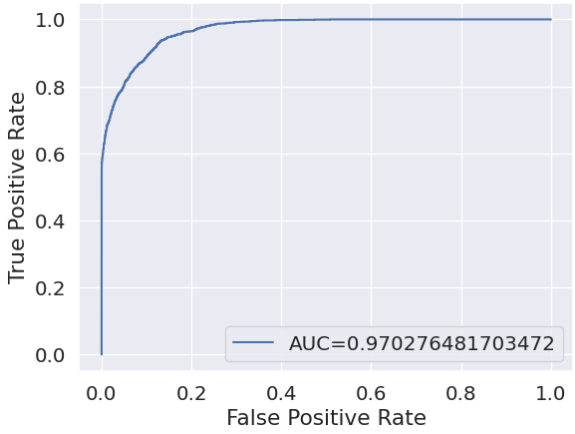


Рисунок 36 – Кривая ошибок

Таблица 4 – Настройки гиперпараметров нейронной сети (Вариант 2)

Гиперпараметр	Значение/настройка
SMOTE	**False**
Доля тестовой выборки	0.2
Функция ошибки	«binary_crossentropy»
Функция оптимизации	**«nadam»**
Метрика качества	**«recall»**
Размер подвыборки	300
Доля валидационной выборки	0.2
Количество эпох	40

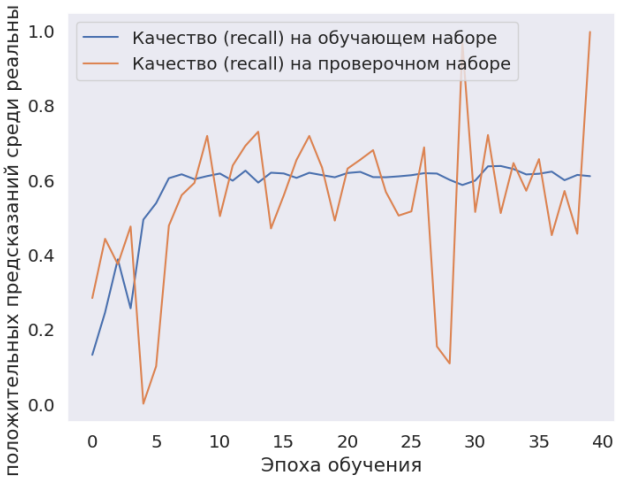


Рисунок 37 – Доля верных положительных предсказаний среди предсказанных положительных

Рисунок 38– Ошибка на обучающем и проверочном наборах

```
601/601 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step
              precision    recall  f1-score   support

           0       1.00      0.64      0.78     17534
           1       0.21      0.99      0.35      1696

    accuracy                           0.67     19230
   macro avg       0.61      0.82      0.57     19230
weighted avg       0.93      0.67      0.74     19230
```

Рисунок 39 – Полный отчет о классификации

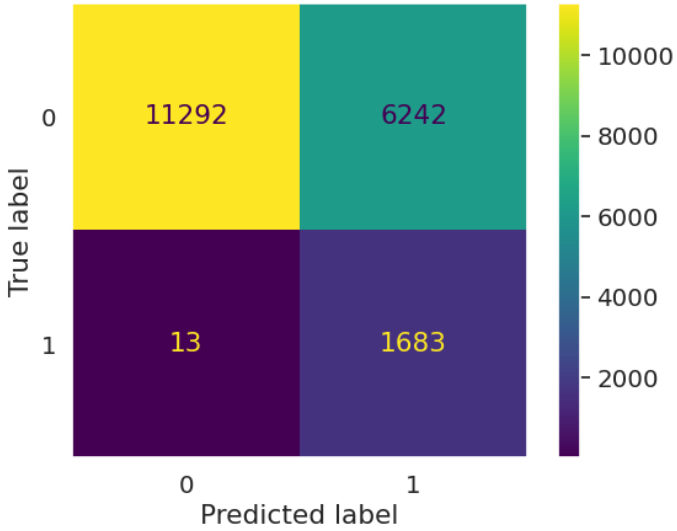


Рисунок 40 – Матрица путаницы

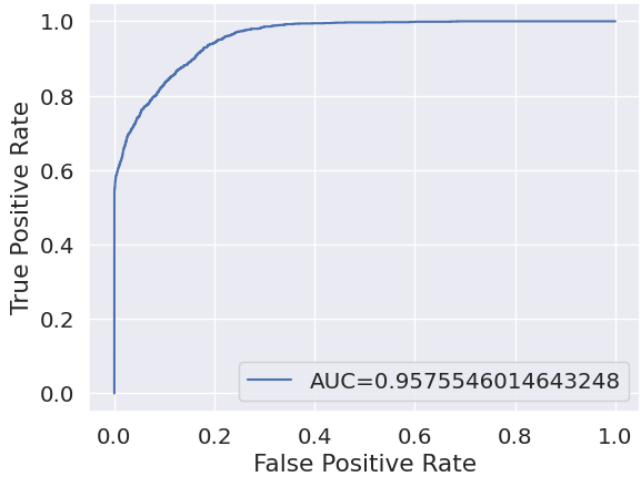


Рисунок 41 – Кривая ошибок

Таблица 5 – Настройки гиперпараметров нейронной сети (Вариант 3)

Гиперпараметр	Значение/настройка
SMOTE	**False**
Доля тестовой выборки	0.2
Функция ошибки	«binary_crossentropy»
Функция оптимизации	**«adam»**
Метрика качества	**«accuracy»**
Размер подвыборки	300
Доля валидационной выборки	0.2
Количество эпох	40

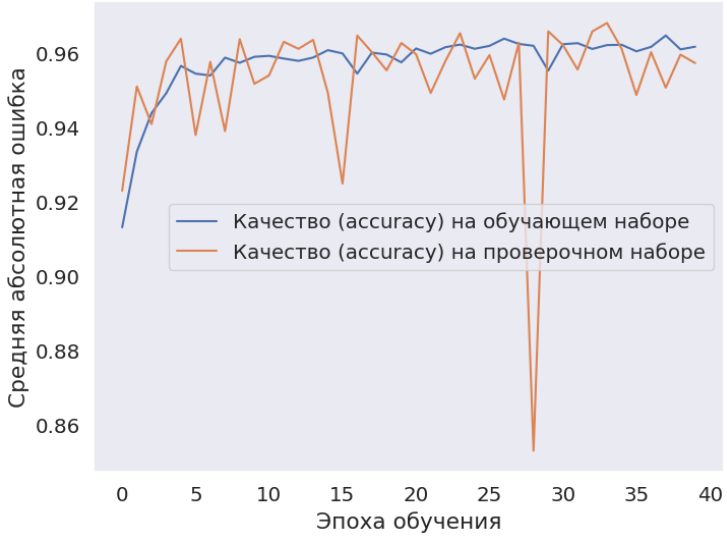


Рисунок 42 – Доля верных положительных предсказаний среди предсказанных положительных

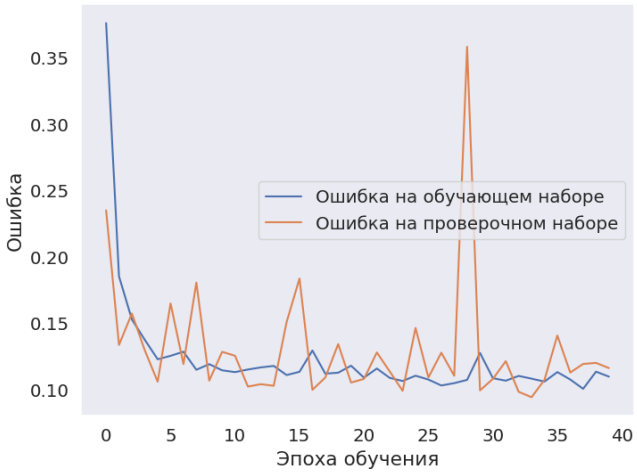


Рисунок 43 – Ошибка на обучающем и проверочном наборах

	precision	recall	f1-score	support
0	0.95	1.00	0.98	17534
1	1.00	0.49	0.66	1696
accuracy			0.95	19230
macro avg	0.98	0.74	0.82	19230
weighted avg	0.96	0.95	0.95	19230

Рисунок 44 – Полный отчет о классификации

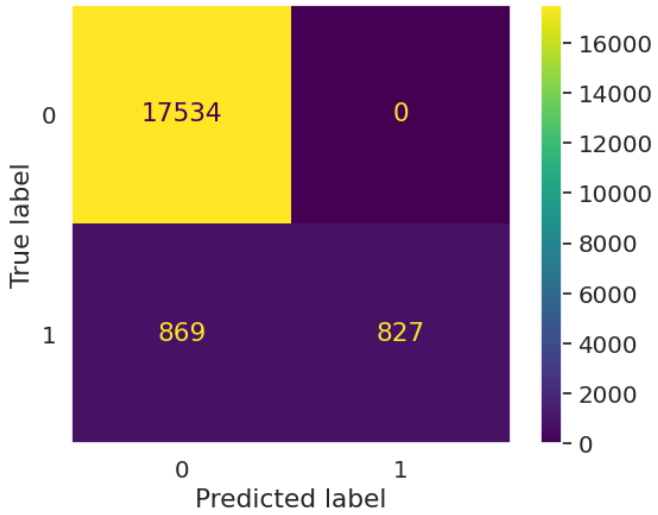


Рисунок 45 – Матрица путаницы

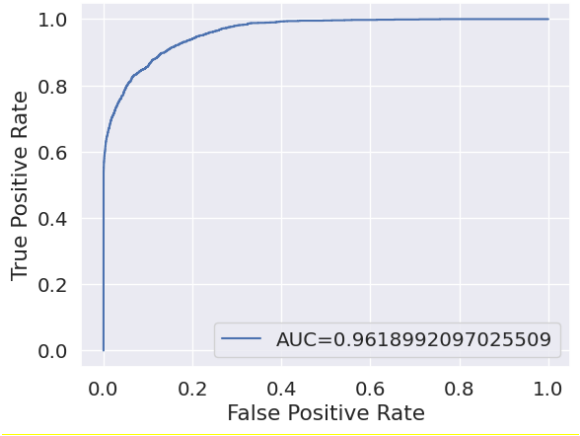


Рисунок 46– Кривая ошибок

Таблица 6 – Настройки гиперпараметров нейронной сети (Вариант 4)

Гиперпараметр	Значение/настройка
SMOTE	**False**
Доля тестовой выборки	0.2
Функция ошибки	«binary_crossentropy»
Функция оптимизации	**«nadam»**
Метрика качества	**«accuracy»**
Размер подвыборки	300
Доля валидационной выборки	0.2
Количество эпох	40

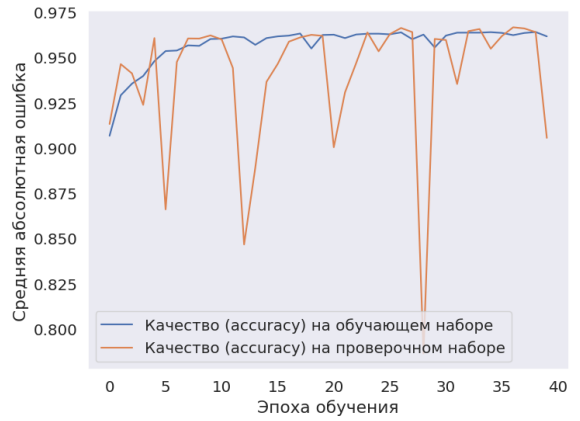


Рисунок 47 – Доля верных положительных предсказаний среди предсказанных положительных

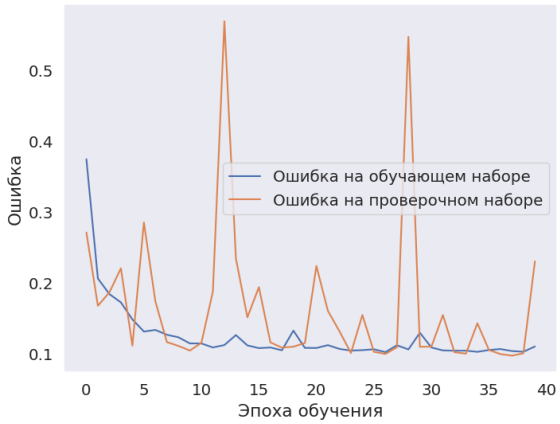


Рисунок 48 – Ошибка на обучающем и проверочном наборах

	precision	recall	f1-score	support
0	0.99	0.91	0.94	17534
1	0.47	0.87	0.61	1696
accuracy			0.90	19230
macro avg	0.73	0.89	0.78	19230
weighted avg	0.94	0.90	0.91	19230

Рисунок 49 – Полный отчет о классификации

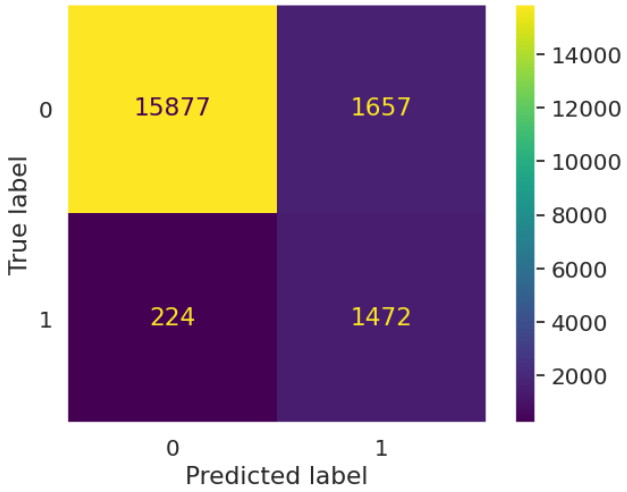


Рисунок 50 – Матрица путаницы

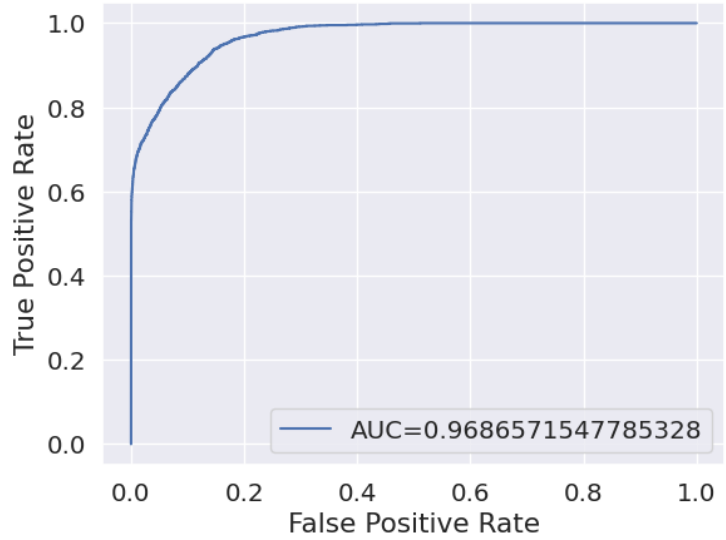


Рисунок 51 – Кривая ошибок

Таблица 7 – Настройки гиперпараметров нейронной сети (Вариант 5)

Гиперпараметр	Значение/настройка
SMOTE	**False**
Доля тестовой выборки	0.2
Функция ошибки	«binary_crossentropy»
Функция оптимизации	**«lion»**
Метрика качества	**«recall»**
Размер подвыборки	300
Доля валидационной выборки	0.2
Количество эпох	40

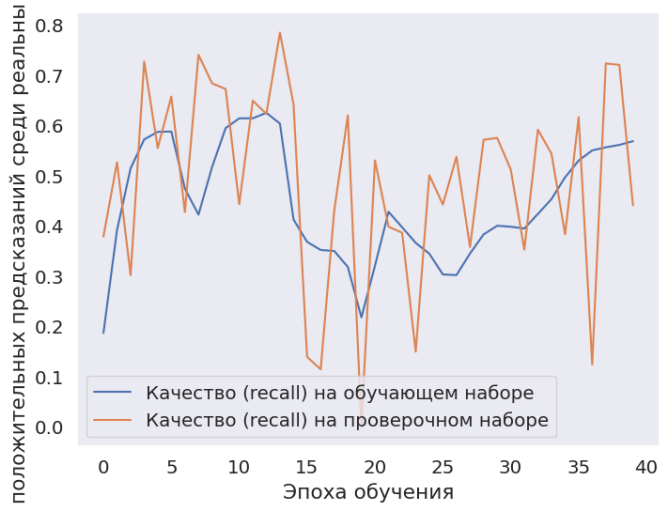


Рисунок 52 – Доля верных положительных предсказаний среди предсказанных положительных

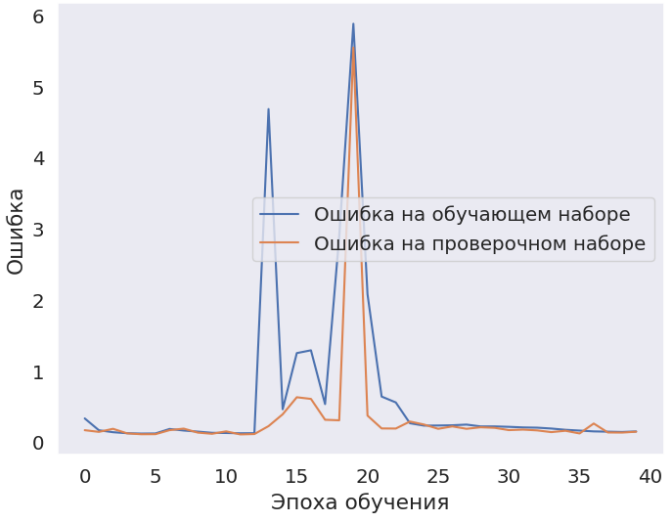


Рисунок 53 – Ошибка на обучающем и проверочном наборах

	precision	recall	f1-score	support
0	0.95	1.00	0.97	17534
1	0.99	0.43	0.60	1696
accuracy			0.95	19230
macro avg	0.97	0.72	0.79	19230
weighted avg	0.95	0.95	0.94	19230

Рисунок 54 – Полный отчет о классификации

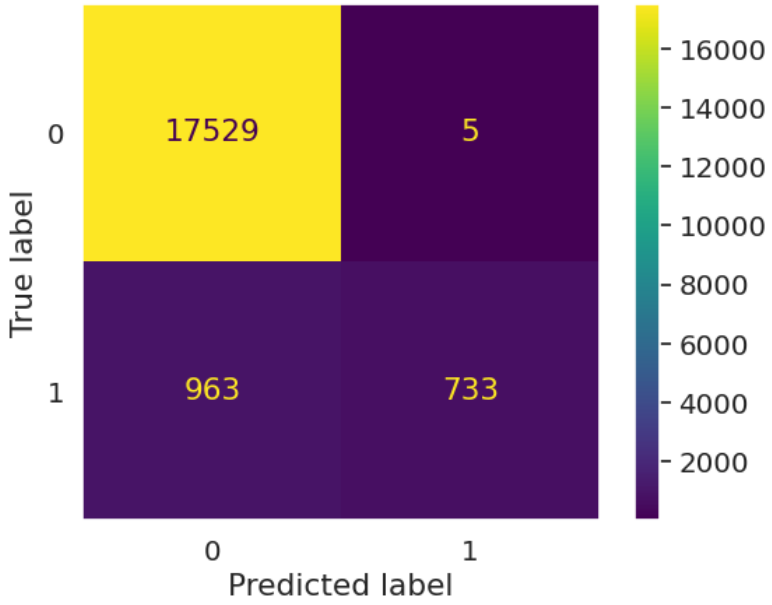


Рисунок 55 – Матрица путаницы

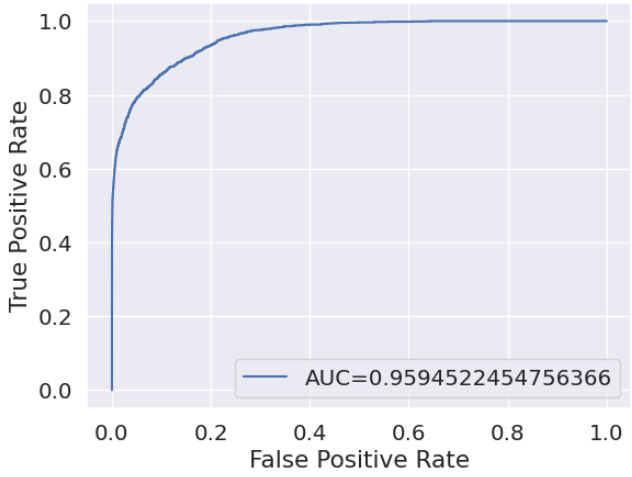


Рисунок 56 – Кривая ошибок

Таблица 8 – Настройки гиперпараметров нейронной сети (Вариант 6)

Гиперпараметр	Значение/настройка
SMOTE	**True**
Доля тестовой выборки	0.2
Функция ошибки	«binary_crossentropy»
Функция оптимизации	**«lion»**
Метрика качества	**«recall»**
Размер подвыборки	300
Доля валидационной выборки	0.2
Количество эпох	40

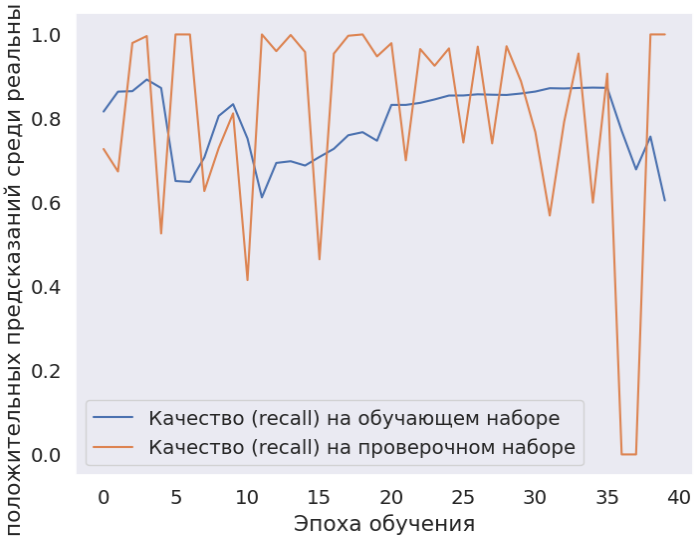


Рисунок 57 – Доля верных положительных предсказаний среди предсказанных положительных

Рисунок 58 – Ошибка на обучающем и проверочном наборах

0	1.00	0.01	0.01	17533
1	0.50	1.00	0.67	17533
accuracy			0.50	35066
macro avg	0.75	0.50	0.34	35066
weighted avg	0.75	0.50	0.34	35066

Рисунок 59 – Полный отчет о классификации

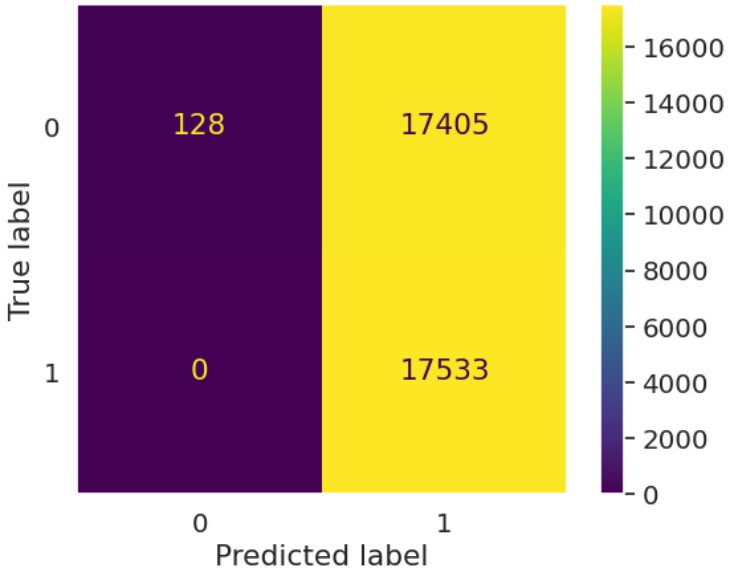


Рисунок 60 – Матрица путаницы

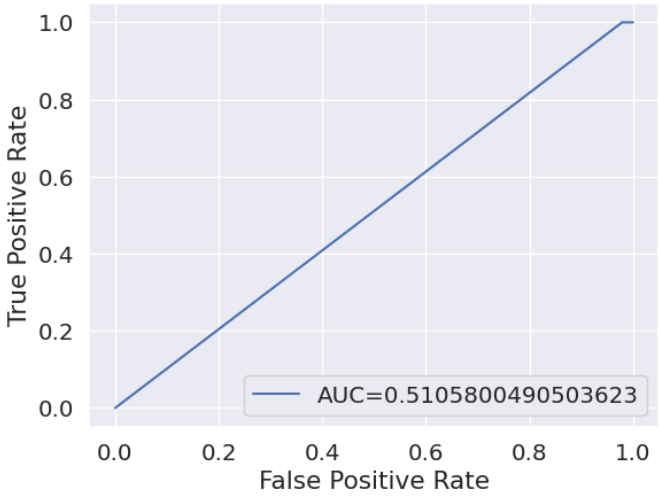


Рисунок 61 – Кривая ошибок

Таблица 9 – Настройки гиперпараметров нейронной сети (Вариант 7)

Гиперпараметр	Значение/настройка
SMOTE	**True**
Доля тестовой выборки	0.2
Функция ошибки	«binary_crossentropy»
Функция оптимизации	**«nadam»**
Метрика качества	**«recall»**
Размер подвыборки	300
Доля валидационной выборки	0.2
Количество эпох	40

Рисунок 62 – Доля верных положительных предсказаний среди предсказанных положительных

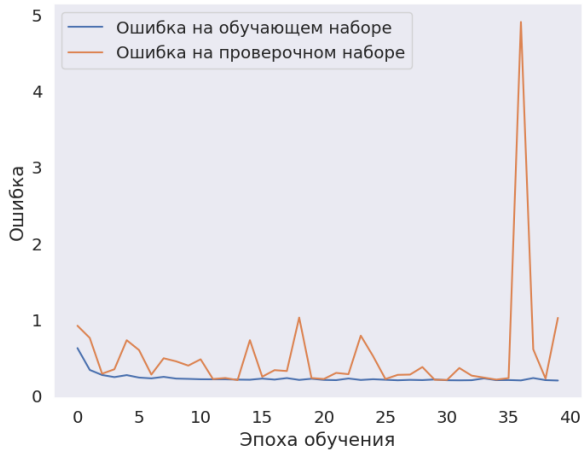


Рисунок 63 – Ошибка на обучающем и проверочном наборах

	precision	recall	f1-score	support
0	0.68	1.00	0.81	17533
1	1.00	0.53	0.69	17533
accuracy			0.77	35066
macro avg	0.84	0.77	0.75	35066
weighted avg	0.84	0.77	0.75	35066

Рисунок 64 – Полный отчет о классификации

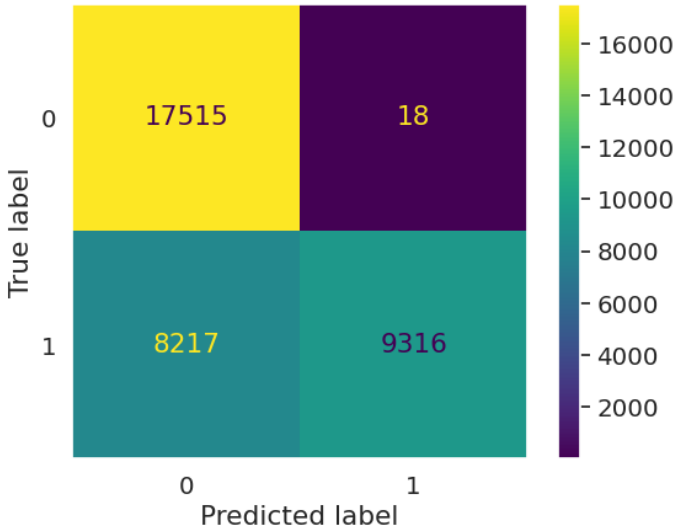


Рисунок 65 – Матрица путаницы

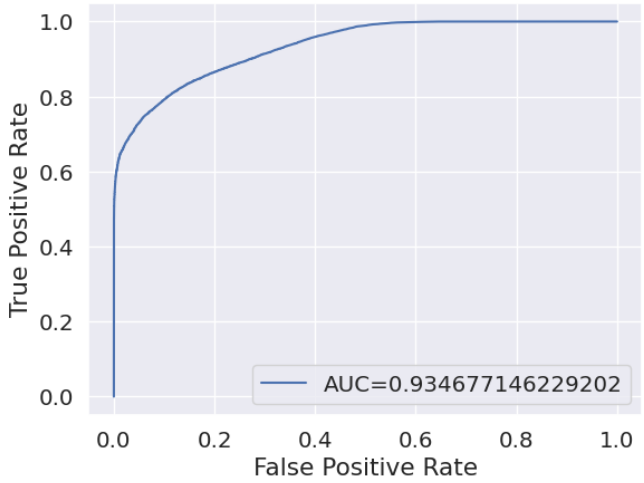


Рисунок 66 – Кривая ошибок

Таблица 10 – Настройки гиперпараметров нейронной сети (Вариант 8)

Гиперпараметр	Значение/настройка
SMOTE	**True**
Доля тестовой выборки	0.2
Функция ошибки	«binary_crossentropy»
Функция оптимизации	**«nadam»**
Метрика качества	**«accuracy»**
Размер подвыборки	300
Доля валидационной выборки	0.2
Количество эпох	40

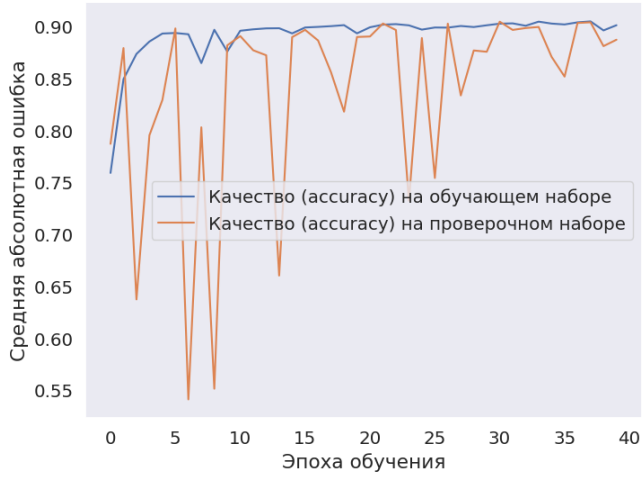


Рисунок 67 – Доля верных положительных предсказаний среди предсказанных положительных

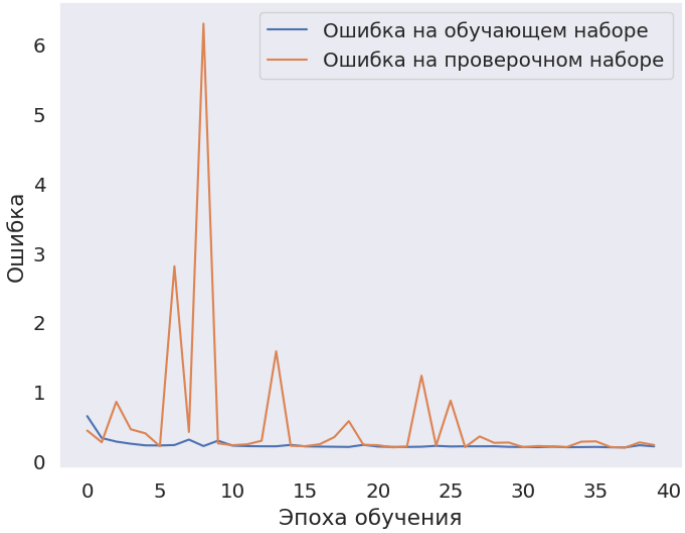


Рисунок 68 – Ошибка на обучающем и проверочном наборах

	precision	recall	f1-score	support
0	0.85	0.94	0.89	17533
1	0.94	0.83	0.88	17533
accuracy			0.89	35066
macro avg	0.89	0.89	0.89	35066
weighted avg	0.89	0.89	0.89	35066

Рисунок 69 – Полный отчет о классификации

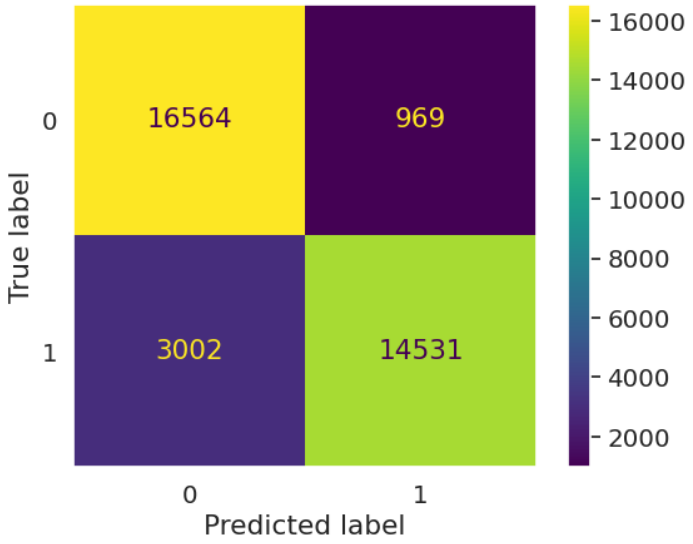


Рисунок 70 – Матрица путаницы

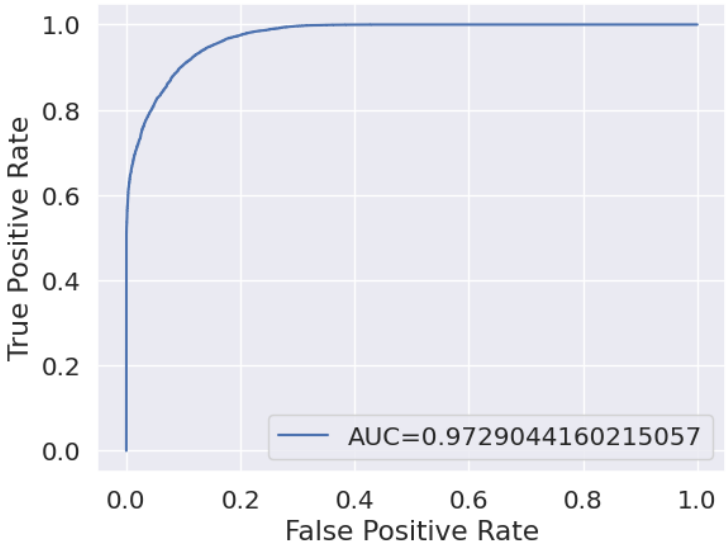


Рисунок 71 – Кривая ошибок

Таблица 11 – Настройки гиперпараметров нейронной сети (Вариант 9)

Гиперпараметр	Значение/настройка
SMOTE	**True**
Доля тестовой выборки	0.2
Функция ошибки	«binary_crossentropy»
Функция оптимизации	**«nadam»**
Метрика качества	**«precision»**
Размер подвыборки	300
Доля валидационной выборки	0.2
Количество эпох	40

Рисунок 72 – Доля верных положительных предсказаний среди предсказанных положительных

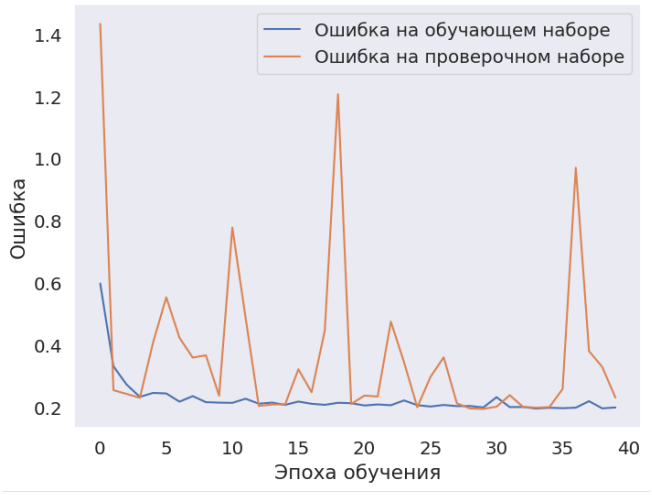


Рисунок 73 – Ошибка на обучающем и проверочном наборах

	precision	recall	f1-score	support
0	0.98	0.79	0.88	17533
1	0.83	0.98	0.90	17533
accuracy			0.89	35066
macro avg	0.90	0.89	0.89	35066
weighted avg	0.90	0.89	0.89	35066

Рисунок 74 – Полный отчет о классификации

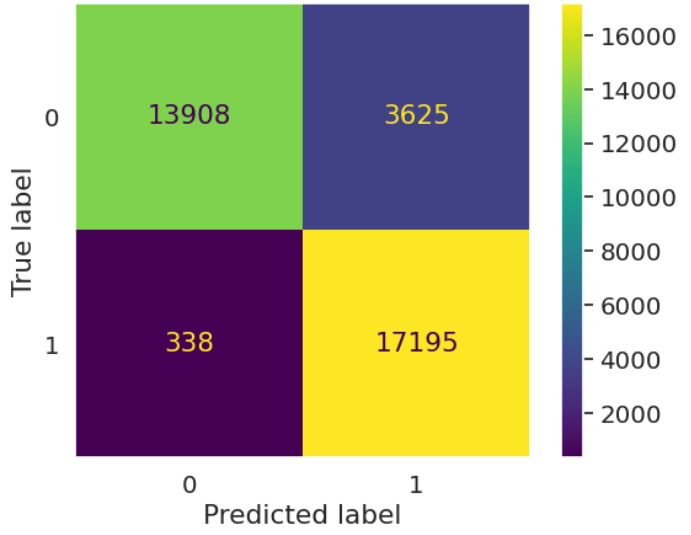


Рисунок 73 – Матрица путаницы

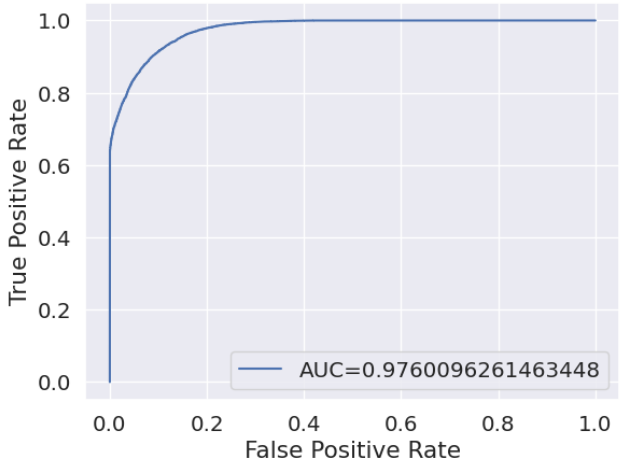


Рисунок 75 – Кривая ошибок

4.2 Анализ полученных результатов обучения

При анализе результатов необходимо опираться на показатели метрик качеств: accuracy, recall, precision, F1-score, ROC Curve. Их описание приведено в п. 2.2. Так, как в медицине недопустим ложноотрицательный диагноз, то следует в первую очередь особое внимание уделять показателю recall.

Для удобства результаты приведены в таблицу 12.

Таблица 12 – Сравнения результатов

	Выход модели	Метрика				
		precision	recall	f1-score	accuracy	ROC Curve
1	2	3	4	5	6	7
Вариант 1	0	0,96	1,00	0,98	0,96 0,96	0,970 0,970
	1	0,96	1,00	0,98		
Вариант 2	0	1,00	0,64	0.78	0,67	0,957
	1	0,21	0,99	0,55		
Вариант 3	0	0,95	1,00	0,98	0,95	0,961
	1	1,00	0,49	0,66		
Вариант 4	0	0,99	0,91	0,94	0,90	0,967
	1	0,47	0,87	0,61		

Продолжение таблицы 12

1	2	3	4	5	6	7
Вариант 5	0	0,95	1,00	0,97	0,95	0,959
	1	0,99	0,43	0,60		
Вариант 6	0	1,00	0,01	0,01	0,50	0,511
	1	0,50	1,00	0,67		
Вариант 7	0	0,68	1,00	0,81	0,77	0,935
	1	1,00	0,53	0,69		
Вариант 8	0	0,85	0,94	0,89	0,89	0,973
	1	0,94	0,83	0,88		
Вариант 9	0	0,98	0,79	0,88	0,89	0,976
	1	0,83	0,98	0,90	0,89	0,976

Лучшие результаты показали варианты 8 и 9.

4.3 Оценка важности признаков

На рисунке 76 представлен график влияния признаков.

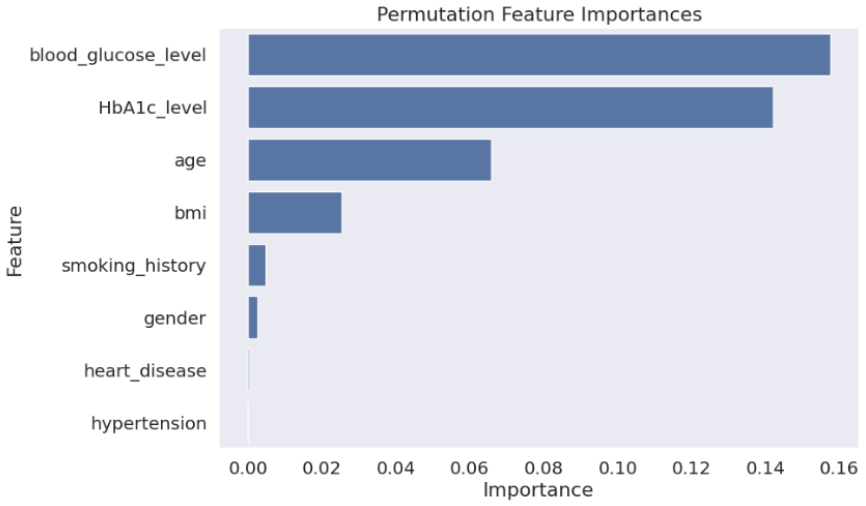


Рисунок 76 – Влияние признаков

Наиболее важные признаки по убыванию:

- blood_glucose_level – уровень глюкозы в крови
- HbA1c_level – уровень гликированного гемоглобина

- age – возраст пациента
- bmi – индекс массы тела
- hypertension – наличие гипертонии
- heart_disease – наличие сердечных заболеваний
- smoking_history – история курения
- gender – пол пациента
- SHAP подтвердил выводы Permutation Importance:
- Наибольший вклад в предсказание оказывают уровень глюкозы и HbA1c
- Также важны возраст и BMI
- Пол и история курения оказывают слабое влияние

ЗАКЛЮЧЕНИЕ

За последние 25 лет развитие машинного обучения сделало огромный шаг вперед, включая сферу медицины. Искусственный интеллект производит революцию в лечении диабета, прогнозируя риски и индивидуализируя планы лечения для улучшения результатов лечения пациентов. Машинное обучение в диагностике диабета обещает улучшение результатов лечения и безопасности жизни.

Методы ИИ показали многообещающие результаты в прогнозировании начала, диагностики и прогнозирования диабета. Однако для полного использования возможностей ИИ в здравоохранении необходимы непрерывные исследования и разработки. Текущие усилия должны решать такие проблемы, как конфиденциальность данных, предвзятость и ложные прогнозы, одновременно повышая точность и эффективность.

Кроме того, разработка этических принципов имеет решающее значение для ответственного и этичного использования ИИ в прогнозировании начала заболеваний. Это не только улучшит результаты лечения пациентов, но и будет способствовать продвижению ИИ в здравоохранении, тем самым принося пользу людям в глобальном масштабе.

В первой части диссертации рассмотрены: история ИИ, его основные направления и задачи. Ознакомление с текущим состоянием ИИ, а также его этическим аспектом.

Во второй части проведен анализ классических методов машинного обучения, постановок задач машинного обучения, а также метрик оценки качества моделей.

В третьей части осуществлена предобработка набора данных «Diabetes prediction», построена и обучена модель для прогнозирования диабета классическими методами, проведена оценка их качества.

В четвертой части построена и обучена модель для прогнозирования диабета при помощи нейронной сети, создано множество вариантов архитектур нейронной сети, проведен их анализ и выбран оптимальный вариант.

В процессе выполнения магистерской диссертации, создана основная кодовая база трех моделей (Наивный Байес, логистическая регрессия, нейронная сеть), которые могут быть модифицированы в дальнейшем, путем политики открытого исходного кода.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1 Андрейченко, А.Е. Разработка и валидация моделей машинного обучения, прогнозирующих риск госпитализации пациентов с сахарным диабетом в течение последующих 12 месяцев / А.Е. Андрейченко, А.Д. Ермак, Д.В. Гаврилов, Р.Э. Новицкий, А.В. Гусев // Сахарный диабет. – 2024. – Т. 27. – №2. – С. 142-157. – Режим доступа: https://doi.org/10.14341/DM13065

2 Вастрик [Электронный ресурс]: офиц. сайт. Режим доступа – https://vas3k.ru/blog/machine_learning/

3 Гусев, А.В. Искусственный интеллект в медицине и здравоохранении / А. В. Гусев, С. Л. Добриднюк // Информационное общество – Режим доступа: https://webiomed.ru/media/publications_files/iskusstvennyi-intellekt-v-meditsine-i-zdravookhranenii.pdf – 18.12.2024.

4 Климонтов, В.В. Искусственный интеллект в диабетологии / В. В. Климонтов, В. Б. Бериков, О. В. Сайк // Эндокринологический научный центр. – Режим доступа: https://doi.org/10.14341/DM12665 – 18.12.2024.

5 Кугаевских, А.В. Классические методы машинного обучения: учебное пособие / А.В. Кугаевских, Д.И. Муромцев, О.В. Кирсанова. – СПб: Университет ИТМО, 2022. – 53 с

6 Лаптев, Д.Н. Модель клинического прогнозирования сахарного диабета MODY типа у детей / Д. Н. Лаптев, Е. А. Сечко, Е. М. Романенкова, И. А. Еремина, О. Б. Безлепкина, В. А. Петеркова, Н. Г. Мокрышева // Сахарный диабет. – 2024. – Т. 27. – №1. – С. 33-40. – Режим доступа: https://doi.org/10.14341/DM13091

7 Лимановская, О. В. Основы машинного обучения: учебное пособие / О. В. Лимановская. – Екатеринбург: Уральского университета, 2020. – 88 с.

8 Флакс, П. Машинное обучение. Наука и искусство построения алгоритмов, которые извлекают знания из данных / П. Флакс. – Москва: Изд-во ДМК Пресс, 2015. – 400 с.

9 Потопахин, В.В. Романтика искусственного интеллекта / В.В. Потопахин – Москва: Изд-во ДМК Пресс, 2017. – 170 с.

10 Русяева, Н.В. Методы машинного обучения в дифференциальной диагностике сложно классифицируемых типов сахарного диабета / Н.В. Русяева, И.И. Голодников, И. В. Кононенко, Т. В. Никонова, М. В. Шестакова // Сахарный диабет. – 2023. – Т. 26. – №5. – С. 473-483. - Режим доступа: https://doi.org/10.14341/DM13070

11 Рашка, С. Python и машинное обучение / С. Рашка. – Москва: Изд-во ДМК Пресс, 2017. – 418 с.

12 Шинелёв, И.Н. Использование искусственных нейронных сетей в медицине / И.Н. Шинелев, И.Е. Тарасов // ИТ-Стандарт. – Режим доступа: https://itstd-journal.ru/wp-content/uploads/2021/04/Статья_6-Шинелев.pdf – 18.12.2024.

13 Фаустова, К.И. Нейронные сети: применение сегодня и перспективы развития / К.И. Фаустова // Территория науки. – 2017– №4. – с.83-87.

14 Федоров, М. Искусственный интеллект: международно-правовые коллизии и этические нормы / М. Федоров // Международная жизнь. – Апрель 2020.

15 Холодная, Е.В. Этические стандарты и регулирование искусственного интеллекта / Е. В. Холодная // Информационное право. – 2020. – № 3 (65). – с.12-17

16 Гудфеллоу, Я. Глубокое обучение / Я. Гудфеллоу. – Москва: Изд-во ДМК Пресс, 2017. – 652 с.

17 Brown Peter, Cocke John, Della Pietra Stephen, Della Pietra Vincent J., Jelinek Fredrick, Lafferty John, Mercer Robert, Roossin Paul Statistical Approach to Machine Translation / P. Brown, J. Cocke, S. Della Pietra , V. Della Pietra, F. Jeliek, J. Lafferty, R. Mercer, P. Roossin // Computational Linguistics. – 1990. – №16(2). – с. 79–85

18 Kaggle [Электронный ресурс]: офиц. сайт. Режим доступа: https://www.kaggle.com/datasets/iammustafatz/diabetes-prediction-dataset/ – 20.11.2023

19 ScikitLearn [Электронный ресурс]: офиц. сайт. Режим доступа: https://scikit-learn.org/stable/modules/naive_bayes.html – 15.11.2023.

20 Pandas [Электронный ресурс]: офиц. сайт. Режим доступа: пahttps://pandas.pydata.org/docs/reference/api/pandas.get_dummies.html – 15.11.2023.

21 ScikitLearn [Электронный ресурс]: офиц. сайт. Режим доступа: https://scikitlearn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html– 15.11.2023.

22 Zendesk [Электронный ресурс]: офиц. сайт. Режим доступа: https://www.zendesk.com/au/blog/machine-learning-and-deep-learning – 09.05.2024

Список опубликованных научных трудов

23 Колесников, В.Т. Диагностика диабета: метод логистической регрессии / В.Т. Колесников // «День науки»: материалы XXXIII научной конференции Амурского государственного университета (18 апреля 2024 г., Благовещенск). – Благовещенск: АмГУ, 2024. – С. 73-75.

24 Колесников, В.Т. Нейросетевой подход для прогнозирования заболевания диабетом / В.Т. Колесников, Н.Н. Максимова // Far East Math – 2024: материалы V национальной научной конференции (2–7 декабря 2024 г.) / Министерство науки и высшего образования Российской Федерации, Тихоокеанский государственный университет; редакционная коллегия: Е. Г. Агапова (ответственный редактор) [и др.]. – Хабаровск: Издательство ТОГУ, 2025. – С. 160-166.

25 Колесников, В.Т. Диагностика диабета: сравнение методы машинного обучения / В.Т. Колесников // «День науки»: материалы XXXIV научной конференции Амурского государственного университета (13 марта 2025 г., Благовещенск). – Благовещенск: АмГУ, 2025. – С.72-74.

ПРИЛОЖЕНИЕ А

«Наивный байесовский классификатор (прогнозирование диабета).ipynb»

Импортируем необходимые пакеты Python:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline

from sklearn import metrics

from numpy import *                                    # Импорт всего из
библиотеки
from sklearn. model_selection import train_test_split
from sklearn.naive_bayes import *

import seaborn as sns
sns.set(style='dark', font_scale=1.3)

import warnings
warnings.filterwarnings('ignore')
```

Загружаем данные:

```
df = pd.read_csv('/content/diabetes_prediction_dataset.csv')        # Загру-
жаем набор данных

print(df)
```

Разделим набор данных на обучающий и тестовый

```
X_train, X_test, y_train, y_test = train_test_split(df[['gender', 'age',
'hypertension', 'heart_disease', 'smoking_history', 'bmi', 'HbA1c_level',
'blood_glucose_level']], df['diabetes'], test_size = 0.3, random_state =
150)

#print(X_train, y_train)
#print(X_test, y_test)
```

Построим и обучим модель с использованием функции MultinomialNB()

```
model = MultinomialNB()                          # Для всех параметров оставим
значения по умолчанию
model.fit(X_train, y_train)                      # Обучаем модель на тренировоч-
ных данных
y_pred = model.predict(X_test)                   # Выполняем предсказание на
всем наборе данных
```

```
cnf_matrix = metrics.confusion_matrix(y_test, y_pred)
cnf_matrix
```

```
print('количество истинно отрицательных предсказаний:', cnf_matrix[0, 0])
print('количество истинно положительных предсказаний:', cnf_matrix[1, 1])
print('количество ложно отрицательных предсказаний:', cnf_matrix[1, 0])
print('количество ложно положительных предсказаний:', cnf_matrix[0, 1])
```

```
# определение метрик
y_pred_proba = model.predict_proba(X_test)[::, 1]
fpr, tpr, _ = metrics.roc_curve(y_test, y_pred_proba)
auc = metrics.roc_auc_score(y_test, y_pred_proba)

# построение ROC-кривой
plt.plot(fpr, tpr, label = 'AUC='+str(auc))
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend(loc = 4)
plt.grid()
plt.show()
```

Добавляем метрики качества.

```
print(f'Accuracy (доля правильных ответов): {metrics.accuracy_score(y_test,
y_pred)*100}%')

print(f'Precision (точность): {metrics.precision_score(y_test,
y_pred)*100}%')

print(f'Recall (полнота): {metrics.recall_score(y_test, y_pred)*100}%')

print(f'F1: {metrics.f1_score(y_test, y_pred)*100}%')
```

Выполним обучение с помощью метода GaussianNB()

```
model_G = GaussianNB()                          # Для всех параметров оставим
значения по умолчанию
model_G.fit(X_train, y_train)                   # Обучаем модель на трениро-
вочных данных
y_pred_G = model.predict(X_test)                # Выполняем предсказание на
всем наборе данных
```

```
cnf_matrix_G = metrics.confusion_matrix(y_test, y_pred_G)
cnf_matrix_G
```

```
print('количество истинно отрицательных предсказаний:', cnf_matrix_G[0, 0])
print('количество истинно положительных предсказаний:', cnf_matrix_G[1, 1])
```

```
print('количество ложно отрицательных предсказаний:', cnf_matrix_G[1, 0])
print('количество ложно положительных предсказаний:', cnf_matrix_G[0, 1])
```

```
# определение метрик
y_pred_proba_G = model_G.predict_proba(X_test)[::, 1]
fpr, tpr, _ = metrics.roc_curve(y_test, y_pred_proba_G)
auc = metrics.roc_auc_score(y_test, y_pred_proba_G)

# построение ROC-кривой
plt.plot(fpr, tpr, label = 'AUC='+str(auc))
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend(loc = 4)
plt.grid()
plt.show()
```

Выполним обучение с помощью метода ComplementNB()

```
model_C = ComplementNB()                              # Для всех параметров оста-
вим значения по умолчанию
model_C.fit(X_train, y_train)                        # Обучаем модель на трениро-
вочных данных
y_pred_C = model.predict(X_test)                     # Выполняем предсказание на
всем наборе данных
```

```
cnf_matrix_C = metrics.confusion_matrix(y_test, y_pred_C)
cnf_matrix_C
```

```
print('количество истинно отрицательных предсказаний:', cnf_matrix_C[0, 0])
print('количество истинно положительных предсказаний:', cnf_matrix_C[1, 1])
print('количество ложно отрицательных предсказаний:', cnf_matrix_C[1, 0])
print('количество ложно положительных предсказаний:', cnf_matrix_C[0, 1])
```

```
# определение метрик
y_pred_proba_C = model_C.predict_proba(X_test)[::, 1]
fpr, tpr, _ = metrics.roc_curve(y_test, y_pred_proba_C)
auc = metrics.roc_auc_score(y_test, y_pred_proba_C)

# построение ROC-кривой
plt.plot(fpr, tpr, label = 'AUC='+str(auc))
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend(loc = 4)
plt.grid()
plt.show()
```

ПРИЛОЖЕНИЕ Б
«Метод логистической регрессии (прогнозирование диабета).ipynb»

Импортируем необходимые пакеты Python:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline

from sklearn import metrics

from numpy import *                                    # Импорт всего из
библиотеки
from sklearn. model_selection import train_test_split
from sklearn.naive_bayes import *

import seaborn as sns
sns.set(style='dark', font_scale=1.3)

import warnings
warnings.filterwarnings('ignore')
```

Загружаем данные:

```
df = pd.read_csv('/content/diabetes_prediction_dataset.csv')        # Загру-
жаем набор данных

print(df)
```

Разделим набор данных на обучающий и тестовый

```
X_train, X_test, y_train, y_test = train_test_split(df[['gender', 'age',
'hypertension', 'heart_disease', 'smoking_history', 'bmi', 'HbA1c_level',
'blood_glucose_level']], df['diabetes'], test_size = 0.3, random_state =
150)

#print(X_train, y_train)
      #print(X_test, y_test)
```

Построим и обучим модель с использованием логистической регрессии:

```
model_log = LogisticRegression()                        # Для всех параметров
оставим значения по умолчанию
model_log.fit(X_train, y_train)                    # Обучаем модель на трени-
ровочных данных
y_pred_log = model_log.predict(X_test)                # Выполняем предсказа-
ние на всем наборе данных
```

Выполним диагностику модели

```
cnf_matrix_log = metrics.confusion_matrix(y_test, y_pred_log)
cnf_matrix_log

print(f'Accuracy (доля правильных ответов): {metrics.accuracy_score(y_test,
y_pred_log)*100}%')

print(f'Precision (точность): {metrics.precision_score(y_test,
y_pred_log)*100}%')

print(f'Recall (полнота): {metrics.recall_score(y_test, y_pred_log)*100}%')

print(f'F1: {metrics.f1_score(y_test, y_pred_log)*100}%')
```

Выводим отчет о классификации:

```
print(classification_report(y_test, y_pred_log))
```

Построим ROC-кривую

```
# определение метрик
y_pred_proba_log = model_log.predict_proba(X_test)[::, 1]
fpr_log, tpr_log, _ = metrics.roc_curve(y_test, y_pred_proba_log)
auc_log = metrics.roc_auc_score(y_test, y_pred_proba_log)

# построение ROC-кривой
plt.plot(fpr_log, tpr_log, label = 'AUC='+str(auc_log))
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend(loc = 4)
plt.grid()
plt.show()

coefficients = model_log.coef_[0]

feature_importance = pd.DataFrame({
    'Feature': X_train.columns,
    'Coefficient': coefficients
}).sort_values(by='Coefficient', key=abs, ascending=False)

print(feature_importance)

plt.figure(figsize=(10,6))
sns.barplot(x='Coefficient', y='Feature', data=feature_importance)
plt.title('Влияние признаков (коэффициенты логистической регрессии)')
plt.axvline(0, color='black', linestyle='--')
plt.show()
```

ПРИЛОЖЕНИЕ В
«Нейронная сеть (прогнозирование диабета).ipynb»

Импортируем необходимые пакеты Python:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline

from tensorflow.keras.models import Sequential    # Импорт последовательной модели из библиотеки моделей
from tensorflow.keras.layers import Dense, Dropout

from matplotlib.pyplot import figure

from sklearn import metrics

from numpy import *                                    # Импорт всего из библиотеки
from sklearn. model_selection import train_test_split
from sklearn.naive_bayes import *
#from sklearn.linear_model import LogisticRegression

import seaborn as sns
sns.set(style='dark', font_scale=1.3)

from imblearn.over_sampling import SMOTE   # imbalanced-learn (imblearn) — библиотека для борьбы с проблемами несбалансированных наборов данных; содержит несколько различных методов для проведения ресэмплинга (https://habr.com/ru/articles/461285/)
                                           # В SMOTE (Способ Передискретизации Синтезированных Меньшинств) создаются элементы в непосредственной близости от уже существующих в меньшем наборе.

#from sklearn.ensemble import RandomForestClassifier     # sklearn.ensemble - библиотека ансамблевых методов (https://scikit-learn.org/stable/modules/ensemble.html)
                                                        # RandomForestClassifier - один из ансаблевых методов

from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay, classification_report   #  classification_report в- для отображения текстовых отчетов по основным показателям классификации (точность, частота отзыва, значение F1 и др.)
import tensorflow as tf
import pickle                                  # реализует мощный алгоритм сериализации и десериализации объектов Python
```

```
import warnings
warnings.filterwarnings('ignore')
```

Загружаем данные:

```
df = pd.read_csv('/content/diabetes_prediction_dataset.csv')         # Загру-
жаем набор данных
print(df.shape)

df.head()
```

Статистическая обработка данных

```
df.duplicated().sum()
df.drop_duplicates(inplace=True)
display('Размер преобразованной таблицы:')
display(df.shape)
df.shape
df.describe()

from sklearn.preprocessing import LabelEncoder

labelencoder = LabelEncoder()

X = df[['gender', 'age', 'hypertension', 'heart_disease', 'smoking_history',
'bmi', 'HbA1c_level', 'blood_glucose_level']].copy()

X.loc[:, 'gender'] = labelencoder.fit_transform(X.loc[:, 'gender'])

X.loc[:, 'smoking_history'] = labelencoder.fit_transform(X.loc[:, 'smok-
ing_history'])

X.head()

X.corr()

sns.heatmap(X.corr())

y = df['diabetes']
print(y)

y.value_counts()
```

```
smote = SMOTE(sampling_strategy='minority')
X, y = smote.fit_resample(X, y)

y.value_counts()
```

Разделим набор данных на обучающий и тестовый:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=149, stratify=y)

((X_train.shape[0], X_train.shape[1]), (X_test.shape[0], X_test.shape[1]))
```

Создаем нейронную сеть

```
model = Sequential()
model.add(Dense(128, activation='relu', input_shape=(X_train.shape[1],)))
model.add(Dense(256, activation='relu'))
model.add(Dense(512, activation='relu'))
model.add(Dense(1024, activation='relu'))
model.add(Dense(2048, activation='relu'))
model.add(Dense(4096, activation='relu'))
model.add(Dense(1, activation='sigmoid')) #softmax, sigmoid

print(model.summary())

model.compile(optimizer='nadam', loss = 'binary_crossentropy', metrics=['
precision'])
# Check data types of X_train
print(X_train.dtypes)

# Convert any object columns to float32
for col in X_train.columns:
    if X_train[col].dtype == 'object':
        print(f"Converting column '{col}' with dtype 'object' to 'float32'")
        # Attempt to convert the column to numeric, coercing errors to NaN
        X_train[col] = pd.to_numeric(X_train[col], errors='coerce')
        # Fill any resulting NaN values (e.g., with the mean or median)
        X_train[col].fillna(X_train[col].mean(), inplace=True) # Or medi-
an(), depending on your data
        X_train[col] = X_train[col].astype('float32')
    else:
        # Ensure other numeric columns are also float32 for consistency with
Keras input
        X_train[col] = X_train[col].astype('float32')

# Do the same for X_test to ensure consistency for evaluation later
for col in X_test.columns:
    if X_test[col].dtype == 'object':
        print(f"Converting column '{col}' with dtype 'object' to 'float32'
for X_test")
        X_test[col] = pd.to_numeric(X_test[col], errors='coerce')
        X_test[col].fillna(X_test[col].mean(), inplace=True)
        X_test[col] = X_test[col].astype('float32')
    else:
        X_test[col] = X_test[col].astype('float32')
```

```
# Ensure y_train and y_test are also in a suitable format (e.g., numpy array
of float32)
y_train = y_train.astype('float32')
y_test = y_test.astype('float32')

# Now try fitting the model again
history = model.fit(X_train,
                    y_train,
                    batch_size = 300,
                    epochs=40,
                    validation_split=0.2,
                    verbose=1)
)

plt.figure(figsize=(8, 6))

plt.plot(history.history['recall'],
         label='Качество (recall) на обучающем наборе')
plt.plot(history.history['val_recall'],
         label='Качество (recall) на проверочном наборе')
plt.xlabel('Эпоха обучения')
plt.ylabel('Доля верных положительных предсказаний среди реальных положи-
тельных')
plt.legend()
plt.show()

plt.figure(figsize=(8, 6))

plt.plot(history.history['loss'],
         label='Ошибка на обучающем наборе')
plt.plot(history.history['val_loss'],
         label='Ошибка на проверочном наборе')
plt.xlabel('Эпоха обучения')
plt.ylabel('Ошибка')
plt.legend()
plt.show()

y_pred = model.predict(X_test).flatten()
#print(y_pred)


# Apply a threshold (e.g., 0.5) to convert probabilities to class labels
y_pred_classes = np.where(y_pred > 0.5, 1, 0)
#print(y_pred_classes)

# Now use y_pred_classes in classification_report
print(classification_report(y_test, y_pred_classes))

cm = confusion_matrix(y_test, y_pred_classes)
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()
```

```
plt.show()

y_pred_raw = model.predict(X_test)

# If your model's output layer is not a sigmoid/softmax,
# you'll need to apply sigmoid to get probabilities
if not isinstance(model.layers[-1], (tf.keras.layers.Activation,)): # Check
if last layer is Activation
    y_pred_proba = tf.sigmoid(y_pred_raw).numpy() # Convert to probabilities
    y_pred_proba = y_pred_proba[:, 0] # Extract probability of positive
class if needed
else:
    # If last layer is an activation layer, it's likely already outputting
probabilities
    y_pred_proba = y_pred_raw[:, 0] # Extract the probability for the posi-
tive class

# Now proceed with ROC curve calculations:
fpr, tpr, _ = metrics.roc_curve(y_test, y_pred_proba)
auc = metrics.roc_auc_score(y_test, y_pred_proba)
# ... (rest of your code)
# построение ROC-кривой
plt.plot(fpr, tpr, label = 'AUC='+str(auc))
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend(loc = 4)
plt.grid()
plt.show()

from sklearn.inspection import permutation_importance
from sklearn.metrics import make_scorer, accuracy_score
import numpy as np
import pandas as pd # Import pandas for handling DataFrame X

# Define a custom scoring function for binary classification that applies a
threshold
# This function will receive y_true and y_pred (predictions from the
*perturbed* X)
def custom_accuracy_scorer(y_true, y_pred, **kwargs):
    # y_pred received here are the raw predictions (probabilities) from the
model
    # after permutation has been applied to a column in X.

    # Apply a threshold (e.g., 0.5) to convert probabilities to binary la-
bels (0 or 1)
    # Ensure y_pred is treated as numpy array for np.where
    y_pred_classes = np.where(y_pred.flatten() > 0.5, 1, 0)

    # Calculate and return the accuracy score
    return accuracy_score(y_true, y_pred_classes)
```

```
# Create a scorer object using make_scorer.
# needs_estimator=True is generally not needed if the scorer doesn't call
estimator methods.
# However, keep it for robustness in case make_scorer requires it for cer-
tain scenarios.
# The issue was in the scorer trying to re-predict on the original X.
custom_scorer = make_scorer(custom_accuracy_scorer, needs_estimator=False) #
Set needs_estimator to False

# Ensure X_test is a numpy array or a supported type for permuta-
tion_importance
# Keras models usually expect numpy arrays or TensorFlow tensors.
# If X_test is a pandas DataFrame, convert it to a numpy array:
X_test_np = X_test.values

# Вычисляем важность признаков, используя custom_scorer
# Pass the numpy array X_test_np to permutation_importance
result = permutation_importance(model, X_test_np, y_test, scor-
ing=custom_scorer, n_repeats=5, random_state=42)
importances = result.importances_mean

# Create a DataFrame
feature_importance_df = pd.DataFrame({
    'Feature': X.columns, # Use the original column names from X
    'Importance': importances
}).sort_values(by='Importance', ascending=False)

# Визуализация
plt.figure(figsize=(10,6))
sns.barplot(x='Importance', y='Feature', data=feature_importance_df)
plt.title('Permutation Feature Importances')
plt.tight_layout()
plt.show()

# Установка библиотеки shap (если еще не установлена)
!pip install shap

import shap

# Подготовим фоновые данные для объяснения (например, первые 100 образцов)
# Convert background DataFrame to a numpy array as DeepExplainer expects it
background = X_train[:100].values

# Создадим Explainer
# Pass the background data to the DeepExplainer constructor
explainer = shap.DeepExplainer(model, background)
shap_values = explainer.shap_values(background)

# Summary plot
# Pass the numpy array background to the summary_plot as well
shap.summary_plot(shap_values, background, feature_names=X.columns)
```

```
# Create a DataFrame combining the features (X) and the target variable (y)
# X already contains the numerical/encoded versions of gender and smok-
ing_history
df_numeric = pd.concat([X, y], axis=1)

# Now calculate the correlation matrix on the DataFrame with only numerical
data
df_numeric.corr()['diabetes'].sort_values(ascending=False)

from sklearn.metrics import accuracy_score
import numpy as np
import pandas as pd # Ensure pandas is imported if not already

base_pred = model.predict(X_test)
base_acc = accuracy_score(y_test, (base_pred > 0.5))

feature_drop_impact = {}

for i in range(X.shape[1]):
    X_temp = X_test.copy()
    # Use .iloc for integer-position based indexing and assignment
    # Calculate the mean of the column using .iloc and assign it to the tem-
porary DataFrame
    X_temp.iloc[:, i] = X_test.iloc[:, i].mean()
    pred = model.predict(X_temp)
    new_acc = accuracy_score(y_test, (pred > 0.5))
    # Use X.columns[i] to get the feature name correctly
    feature_drop_impact[X.columns[i]] = base_acc - new_acc

# Выводим результаты
pd.Series(feature_drop_impact).sort_values(ascending=False).plot(kind='barh'
, title='Влияние удаления признаков на Accuracy')
plt.xlabel('Падение Accuracy')
plt.ylabel('Признак')
plt.gca().invert_yaxis()
plt.show()
```