В. П. Часовских, В. А. Усольцев, Е. В. Кох, Г. А. Акчурина

ДАТАСЕТ, МАШИННОЕ ОБУЧЕНИЕ И ИНФОРМАЦИОННАЯ СИСТЕМА ОПРЕДЕЛЕНИЯ И КАРТИРОВАНИЯ ДЕПОНИРУЕМОГО ЛЕСАМИ УГЛЕРОДА

В. П. Часовских, В. А. Усольцев, Е. В. Кох, Г. А. Акчурина

ДАТАСЕТ, МАШИННОЕ ОБУЧЕНИЕ И ИНФОРМАЦИОННАЯ СИСТЕМА ОПРЕДЕЛЕНИЯ И КАРТИРОВАНИЯ ДЕПОНИРУЕМОГО ЛЕСАМИ УГЛЕРОДА

Монография

Киров
2025

УДК 004.8(075)
ББК 32.813я73
Ч-24

Авторы:
В. П. Часовских, В. А. Усольцев, Е. В. Кох, Г. А. Акчурина

Рецензенты:
кафедра математических методов в экономике и социально-экономических наук Уральского института фондового рынка (протокол № 8 от 16 апреля 2024 г.);
***С. В. Поршнев,** профессор учебно-научного центра «Информационная безопасность» Института радиоэлектроники и информационных технологий – РтФ Уральского федерального университета имени первого Президента России Б. Н. Ельцина, доктор технических наук, профессор*

Ч-24 Часовских, В. П. Датасет, машинное обучение и информационная система определения и картирования депонируемого лесами углерода [Электронный ресурс]: монография / В. П. Часовских, В. А. Усольцев, Е. В. Кох, Г. А. Акчурина. – Электрон. текст. дан. (2,0 Мб). – Киров: Изд-во МЦИТО, 2025. – 1 электрон. опт. диск (CD-R). – Систем. требования: PC, Intel 1 ГГц, 512 Мб RAM, 2,0 Мб свобод. диск. пространства; CD-привод; ОС Windows XP и выше, ПО для чтения pdf-файлов. – Загл. с экрана.

ISBN 978-5-907974-79-1

Научное электронное издание

В контексте глобальных климатических изменений и усиливающегося внимания к проблеме парникового эффекта оценка и мониторинг поглощения углерода экосистемами приобретают стратегическое значение. Исследуются возможности и технологии подготовки актуальных наборов датасет. Определены их свойства и возможные риски искажения данных. Предлагается подготовку датасет, их анализ и обработку проводить в среде технологии искусственного интеллекта – машинное обучение. Рассматривается среда ML.NET, определяется её архитектура, свойства и возможности. ML.NET – новая и эффективная технология. Исследуются возможности ML.NET для регрессионного анализа картирования углерода, депонируемого лесами. Показано, что исследование применения технологий искусственного интеллекта для оценки картирования углерода, депонируемого лесами, затрагивает фундаментальные проблемы технологий искусственного интеллекта. Предлагаемый перечень декомпозирован, и определены возможные направления устранения рисков. Авторы предлагают в качестве информационной системы собственную разработку определения и картирования депонируемого лесами углерода в среде NATURAL.

ISBN 978-5-907974-79-1

УДК 004.8(075)
ББК 32.813я73

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Современная климатическая проблема представляет собой одну из наиболее актуальных и сложных задач, с которыми сталкивается человечество в XXI веке. Изменения климата, вызванные как естественными процессами, так и антропогенной деятельностью, оказывают значительное влияние на экосистемы, в частности на лесные экосистемы и биоту [1].

Леса, являясь важнейшими компонентами биосферы, играют ключевую роль в поддержании экологического баланса, обеспечивая кислород, регулируя климат и служа средой обитания для множества видов. Однако, с учетом глобального потепления, изменения режимов осадков и других климатических факторов, лесные экосистемы сталкиваются с серьезными вызовами, которые могут привести к их деградации и изменению структуры [2].

В условиях глобального потепления, которое, по прогнозам, будет продолжаться в ближайшие десятилетия, важно понимать, как именно климатические изменения влияют на распространение и состояние лесных экосистем.

Климатические изменения на планете выражаются в статистически значительных отклонениях выравненных метеорологических показателей, затрагивающих как глобальные, так и локальные уровни. Основные причины данных изменений включают естественные процессы, такие как вариации в солнечном излучении, а также антропогенные факторы, которые все более усугубляют влияние на климатическую систему Земли [3]. В последние десятилетия наблюдается заметное нарастание частоты экстремальных погодных явлений, что значительно меняет как природные, так и человеческие экосистемы.

Географическая широта является важным параметром, который определяет, насколько выражены климатические изменения в различных регионах мира. Например, в высоких и умеренных широтах изменения проявляются особенно остро, в то время как экваториальные регионы менее подвержены таким резким колебаниям [4]. Исследования показывают, что в условиях глобального

потепления лесные экосистемы сталкиваются с новыми вызовами, такими как изменение видов, обитающих в этих условиях, а также изменение структур необходимых для существования экосистем [5].

Климатические изменения оказывают влияние на лесные экосистемы через модификацию среды обитания, что проявляется в изменении фенологических событий, таких как время цветения и плодоношения растений [6]. Например, сдвиг в температурных режимах может привести к тому, что некоторые виды могут выйти за пределы своих экосистем, что в свою очередь приведет к изменению состава и структуру лесных видов [4]. Это способствует возникновению новых конкурентов и хищников, что усложняет динамику существующих экосистем.

Изменения климата и связанные с ними экологические последствия также касаются уровня углеродных выбросов и общего состояния атмосферы. Человеческая деятельность, включая вырубку лесов и загрязнение, усугубляет проблему, что приводит к увеличению количества углекислого газа в атмосфере и изменяет процесс фотосинтеза у деревьев и других растений [5]. Лесные массивы не только являются поглотителями углерода, но и играют ключевую роль в поддержании биоразнообразия, что делает важным понимание взаимосвязей между климатом и биотой [3].

В процессе изменений климата необходимо обращать внимание на адаптацию существующих экосистем. Это может включать как сохранение устойчивых к изменениям видов, так и восстановление деградированных земель [2]. Будущие исследования должны сосредоточиться на взаимодействии биоты и изменяющихся климатических факторов, чтобы обеспечить сохранение экосистем в условиях глобального потепления и его последствий [5].

Климатические изменения влияют на экосистемы лесов через множество специфических механизмов, определяющих их структуру и функции. Основным фактором изменения лесных экосистем является температура, непосредственно влияющая на первичную продукцию лесов. Изменение температуры приводит к изменению роста деревьев, что затрагивает весь трофический уровень лесной

биоты, от флоры до фауны. При повышении температуры активируется первичная продукция, однако это также влияет и на скорость разложения органического вещества. В частности, изменения в температурных режимах могут увеличить темпы разложения, что приведет к изменению углеродного баланса [8].

Ещё одним важным механизмом являются осадки. Изменение режима осадков непосредственно отражается на запасах влаги в почве, что критически важно для роста растений. Увеличение частоты экстремальных осадков или, наоборот, засухи может изменить доступность воды для древесных растений, что влияет на их здоровье и способность к фотосинтезу. Это также затрагивает взаимодействия между различными видами, поскольку многие организмы зависят от определённых условий увлажнённости [4].

Также следует учитывать антропогенные факторы, которые усугубляют природные изменения. Очистка земель под сельское или urban развитие приводит к исчезновению лесов и обострению климатических изменений, изменяя углеродный баланс на глобальном уровне. В результате освободившиеся углеродные запасы возвращаются в атмосферу, усугубляя парниковый эффект [7]. Лесные экосистемы действуют как углеродные фонды, способствующие поглощению CO_2, но разрушение их структуру и функций также может привести к ослаблению этой роли.

В частности, исследования, проведенные в Волжском бассейне, показывают, как защита лесов может быть важным фактора для сохранения углеродного баланса. Они демонстрируют необходимость разработки моделей, описывающих поток углекислого газа и его баланса в зависимости от климатических изменений. Эти модели помогают понять, как изменения климата могут изменить способность лесов поглощать углерод, подвергая риску не только лесные экосистемы, но и показатели глобального потепления.

Климатические изменения влияют и на биологическое разнообразие лесов. Изменение места обитания и распространения видов, вызванное сдвигом клима-

тических условий, может привести к уменьшению видового разнообразия. Высокая зависимость первичной продукции и скорости разложения от климатических факторов делает экосистемы уязвимыми к даже небольшим изменениям в температуре и влажности [10].

Климатические изменения воздействуют на лесные экосистемы через совокупность температурных, гидрологических и антропогенных факторов. Анализ этих механизмов позволяет лучше понять последствия климатических изменений и поможет разработать эффективные стратегии по их адаптации и сохранению.

Северные границы лесов претерпевают значительные изменения в условиях современного климатического кризиса. Географические сдвиги становятся все более очевидными, особенно в тех регионах, где температура в последние десятилетия растет быстрее всего. На Южном Урале и Полярном Урале зафиксировано продвижение границы леса на 60–80 метров, что давно сигнализирует о серьезных изменениях в экосистемах этих территорий [3].

Увеличение температуры и изменение режимов осадков создают благоприятные условия для роста древесных насаждений в северных регионах, где ранее характеристики климата считались неприемлемыми для лесного покрова. На Таймыре, где расположены одни из самых северных лесов в мире, наблюдается увеличение плотности леса, что также подтверждает изменения в экосистемах этих не столь привычных для лесов территорий. Увеличение сомкнутости леса указывает на успешное прорастание и выживание более теплолюбивых видов, в то время как традиционные хвойные породы, такие как лиственница, сталкиваются с новыми вызовами, связанными с изменениями в условиях обитания [5].

После длительного анализа космических снимков на территории Полярного Урала и южной части Югорского Урала видно, как леса продвигаются на север. Уровень изменения лесотундровых сообществ на протяжении XX века, особенно в последние десятилетия, также подтвердил эти тенденции [6]. Это приводит к изменению не только географии лесного покрова, но и к трансформации экосистемных функций и биоразнообразия.

Динамика лесных экосистем имеет важные экологические и экономические последствия. Леса, находящиеся на северных границах, выполняют климат защитные функции, играя значительную роль в углеродном цикле. С увеличением площади лесов на этих территориях вероятно изменение углеродного обмена, что может как положительно, так и отрицательно сказаться на окружающей среде [4].

Примечание об адаптации биоты к изменяющимся условиям также не менее важно. Разнообразие генетических ответов на стрессовые факторы, связанные с климатом, становится критически важным для выживания видов [5]. Это указывает на необходимость комплексного подхода к исследованию. Успех адаптации лесов и биоразнообразия будет зависеть от способности экосистемы реагировать на изменения и интегрировать новые виды, создавая тем самым устойчивые системы в новых условиях.

Лесные экосистемы, как одна из важнейших составляющих биосферы, являются чувствительными индикаторами изменений климата. В результате многовековых изменений, произошедших в период голоцена, лесные экосистемы продемонстрировали гибкость и способность к адаптации, что можно наблюдать на примере елово-широколиственных лесов, которые претерпели значительные трансформации [6]. Такое развитие событий стало результатом адаптивных механизмов, однако происходит это не без негативных последствий.

Исследования показывают, что современные лесные экосистемы становятся все более уязвимыми к изменениям климата. Например, в условиях повышения температуры и изменения режима осадков наблюдается не только ухудшение состояния лесного покрова, но и процесс заболачивания, что ведет к образованию сфагновых ельников и другим изменениям в структуре лесных сообществ [8]. Научные наблюдения за состоянием лесов в России подтверждают, что особенно важны вопросы, связанные с реализацией климатических проектов, которые в большинстве своем продвигаются медленно, особенно в Сибири и на Дальнем Востоке [11].

Статистические данные за последние десятилетия демонстрируют, что лесные экосистемы влияют на климат, регулируя теплообмен и гидрологические процессы, что делает климат региона более мягким и влажным. Это особенно заметно в синергии с изменением структуры лесных покровов, где наблюдается изменение типов растений и углеродного обмена [5]. Более того, увеличение уровня углерода в атмосфере и его улавливание лесами способствует смягчению климатических последствий [4].

Климатические изменения также воздействуют на биоразнообразие, вынуждая многие виды мигрировать в поисках более благоприятных условий. С учетом прогноза дальнейших изменений климата в Европе к концу века, можно ожидать, что это также будет способствовать изменению ареалов распространения лесных экосистем и связанных с ними видов флоры и фауны [6].

Изучая взаимодействие изменений климата и лесных экосистем, следует отметить, что адаптация лесов к новым условиям требует не только научного подхода, но и внедрения практических решений с учетом локальных экологических условий и сообществ. В этом контексте, необходимость в активизации климатических проектов и внедрении устойчивого лесопользования становится все более актуальной [11]. Нельзя забывать, что именно лесные экосистемы играют ключевую роль в углеродном цикле планеты, и сохранение их здорового состояния является необходимым условием для противодействия негативным климатическим последствиям.

Климатические изменения оказывают непосредственное воздействие на биоту лесных экосистем, что подтверждается множеством исследований. Изменения в потоке энергии и питательных веществ, вызванные изменениями климата и экологическими условиями, напрямую влияют на популяции различных видов растений и животных. Например, в южно-таежных лесах центра Каспийско-Балтийского водораздела было установлено, что колебания температуры и уровня осадков значительно изменили структуру сообществ мелких млекопитающих, что способствовало увеличению зональной контрастности их распространения [12].

Конкуренция среди древесных растений также подтверждает свои последствия для молодняка в лесных сообществах. Исследования показали, что климатические и биотические факторы обусловливают динамику естественного восстановления лесов, делая данный процесс более сложным и многогранным. Например, известные исследования подчеркивают, что климатические изменения придают дополнительные сложности процессам лесовосстановления, а именно – увеличивают конкуренцию между видами [7].

Биоразнообразие, по своей сути, является основой адаптационной способности лесов к изменениям климата. Созданные экосистемные функции обеспечиваются многими видами, которые, в свою очередь, способны реагировать на изменения в климатических условиях. Связь между биоразнообразием и климаторегулирующими функциями лесов требует углубленного изучения. Это позволит выявить механизмы, по которым биоразнообразие влияет на устойчивость экосистем к климатическим изменениям и их адаптацию к новым условиям [4].

Историческая динамика показывает, что лесные экосистемы изменялись под влиянием климатических колебаний, таких как увлажнение и похолодание во время Малого ледникового периода. На месте современных еловых лесов когда-то произрастали разнообразные сообщества, которые с течением времени менялись. Современные наблюдения указывают на то, что аналогичные процессы могут повторяться в условиях современного изменения климата, что может привести к уменьшению биоразнообразия [6].

Устойчивость экосистем лесов к изменениям климата оказывается под угрозой из-за значительного влияния внешних факторов и внутренних взаимодействий между видами. Возрастающее воздействие климатических факторов требует новых теоретических подходов и практических исследований, которые помогут понять, как биоразнообразие может быть использовано для устойчивого управления лесами в условиях неопределенности [8].

Климатические изменения оказывают многогранное воздействие на лесные экосистемы, приводя к серьезным экологическим последствиям. Основные

из них включают деградацию почвы, снижение биоразнообразия и изменение структуры лесных сообществ.

Деградация почвы обусловлена изменением климатических условий, таких как повышение температуры и изменение режима осадков. Это приводит к уменьшению содержания органического вещества и ухудшению способности почвы удерживать влагу [13]. Увеличение частоты экстремальных погодных условий, таких как засухи и наводнения, также негативно сказывается на структуре и химическом составе почвы, что, в свою очередь, влияет на здоровье лесов.

Снижение биоразнообразия в лесных экосистемах может быть связано с ухудшением условий для существования различных видов. Если эти изменения происходят слишком быстро, многие виды не успевают адаптироваться, что приводит к их исчезновению [14]. Исследования показывают, что изменения в температурном режиме могут способствовать изменению ареалов распространения видов, таким образом. Смена микроклимата в пределах леса изменяет предшествующие модели взаимодействия между растениями и ими, что ослабляет конкурентные преимущества некоторых видов, в то время как другие могут начать доминировать [8].

Изменения в структуре лесов также имеют серьезные экологические последствия. Например, с увеличением температуры и изменения соотношения влажности, а также с увеличением заболеваемости растений, наблюдается склонность к уменьшению численности хвойных деревьев. Устойчивость хвойных лесов к вредителям временно снижается из-за ослабления растений в условиях стресса [13]. Эти факторы могут привести к изменению таксономического состава и структуры лесных сообществ, влияя на всю связанную с ними биоту.

Важным аспектом является необходимость оценки уязвимости лесных экосистем и разработки адаптивных мер, которые помогут сохранить биоразнообразие и продлить жизнь лесной экосистемы. Адаптивное лесное управление должно учитывать региональные отличия и возникновение новых экосистем в

ответ на климатические изменения [15]. Оценка последствий климатических изменений будет способствовать выявлению возможных угроз и ситуаций, требующих внимания.

Таким образом, современное состояние лесных экосистем требует комплексного подхода к их защите и управлению, учитывающего ожидаемые изменения климата и его последствия для этих важнейших биологических ресурсов.

Адаптация лесных экосистем к климатическим изменениям требует системного подхода, учитывающего специфику каждой экосистемы. В процессе анализа состояния лесов выявлены ключевые проблемы, связанные с изменением климата, включая увеличение частоты и интенсивности природных катастроф, изменение состава флоры и фауны, а также угрозы для экосистемных функций и услуг. Это ставит перед лесным хозяйством задачу формирования эффективных адаптационных стратегий.

Одна из первоочередных мер – обновление образовательных программ в области лесного хозяйства. Специалисты должны получать актуальные знания о механизмах воздействия климата на леса, что позволит им эффективнее разрабатывать стратегии адаптации к новым условиям [3]. Внедрение в curricula современных методов лесоводства и понимания климатических рисков повысит готовность работников к изменениям.

Международное сотрудничество является еще одной важной стратегией. Обмен опытом и знаниями о прогнозировании климатических рисков, а также совместные разработки адаптационных мер могут значительно повысить устойчивость экосистем. Необходимость тесного взаимодействия между различными странами, организациями и научными учреждениями лишь подчеркивает актуальность данного подхода [4].

Создание комплексного плана лесохозяйственных мероприятий, включающего лесозащиту, восстановление лесов, а также противопожарные мероприятия, особенно важно для старовозрастных лесов, более подверженных изменениям климата [6]. Оценка состояния лесного фонда и мониторинг изменений в

экосистемах станут важными инструментами своевременной коррекции мер адаптации. Это позволит не только оперативно реагировать на изменения, но и оптимизировать ресурсы для достижения максимальной эффективности.

Для успешной адаптации образовательные программы и исследования должны быть направлены на анализ уязвимости экосистем, что позволит прогнозировать влияние климатических изменений и разработать меры по снижению рисков. Важно также учитывать необходимость применения инновационных методов управления, таких как устойчивое лесопользование и формирование гибридных экосистем, которые могут справляться с нестабильными условиями [5].

Формирование позитивного имиджа лесного хозяйства позволит привлекать общественное внимание к проблемам адаптации и созданию благоприятных условий для реализации адаптационных мер. Взаимодействие между различными заинтересованными сторонами, совместное планирование и поиск денежных средств для реализации проектов по адаптации являются важной составляющей успеха.

Систематизированный подход к разработке адаптационных мероприятий, основанный на глубоких научных исследованиях и обмене актуальными данными, станет основой для устойчивого лесоводства в условиях меняющегося климата [7].

Современные исследования, выявляющие реакцию биоты на изменения климата, становятся актуальными для понимания динамики лесных экосистем. Особенно важна оценка воздействия климатических изменений на структуру и разнообразие сообществ, как это показали работы, исследующие южно-таежные леса Каспийско-Балтийского водораздела [12]. Выявленные изменения в популяциях мелких млекопитающих, а также увеличивающаяся зональная контрастность, подчеркивают, что экосистемы не просто адаптируются, но и в значительной степени определяются климатическими условиями.

Обратные связи между климатом и биотой могут приводить к нарушению саморегуляции системы, что важно учитывать в будущих исследованиях. Обращение к взаимосвязям между различными трофическими уровнями биоты и их влиянием на климаторегулирующие функции лесов открывает новые горизонты для комплексных исследований. Достаточно изучены отдельные виды и их влияние на экосистему, но необходимо расширить понимание их комбинированного воздействия для оценки общего состояния лесных ресурсов.

Проблема сохранения биоразнообразия и усиливающегося влияния на климаторегулирующие функции лесов требует более глубокого анализа как таксономического, так и функционального разнообразия [4]. Сложность данной задачи демонстрирует необходимость учета различных параметров при оценке экологических изменений. Важность таксономического разнообразия, а также его влияние на экологическую устойчивость должны стать приоритетными направлениями для будущих исследований.

Научные работы также подчеркивают необходимость формирования адаптивных стратегий управления лесными экосистемами, принимающих во внимание сочетание климатических и биотических факторов [16]. Эффективное управление должно учитывать не только текущие климатические условия, но и перспективы их изменений, что связано с изучением пределов устойчивости лесных экосистем в условиях глобального потепления.

Кроме того, изучение механизмов иерархии видов, их взаимосвязей и влияния на экосистемные функции открывает новые аспекты для понимания работы лесных экосистем [17]. Парадигма «большее разнообразие – лучшая устойчивость» нуждается в переосмыслении с точки зрения количественных и качественных показателей, что может привести к созданию более комплексных и эффективных решений по сохранению лесов и улучшению их климаторегулирующих функций.

Следует также обратить внимание на последствия изменения климата, которые могут повлиять на уровень биоризика в определенных экосистемах, по-

этому исследование исторических изменений климата может предоставить ценную информацию о механизмах адаптации и выживания видов [16]. Экологическое моделирование, основанное на полученных данных, может способствовать созданию моделей прогнозирования и более эффективных практик управления лесными ресурсами.

Леса играют ключевую роль в глобальном углеродном цикле [19], выступая в качестве естественных поглотителей углекислого газа из атмосферы. По данным ФАО (Food and Agriculture Organization of the United Nations), лесные экосистемы содержат более 80% наземного и около 40% почвенного углерода биосферы.

Точное определение и картирование объемов депонируемого лесами углерода представляет собой сложную научно-техническую задачу [22–25], решение которой необходимо для реализации международных соглашений по климату, в том числе Парижского соглашения, а также для эффективного управления лесными ресурсами и разработки стратегий устойчивого развития.

Традиционные методы оценки запасов углерода в лесах, основанные на выборочных полевых измерениях, требуют значительных затрат времени и ресурсов, а также не позволяют получить непрерывное пространственное распределение показателей. В этой связи, применение современных информационных технологий [26, 27] и методов машинного обучения для автоматизации процесса картирования углерода представляется перспективным направлением исследований.

ГЛАВА 1. ТЕОРЕТИЧЕСКИЕ ОСНОВЫ ОПРЕДЕЛЕНИЯ ДЕПОНИРОВАНИЯ УГЛЕРОДА ЛЕСНЫМИ ЭКОСИСТЕМАМИ

Депонирование углерода в лесных экосистемах стало актуальной темой в контексте глобальных климатических изменений.

Леса играют важнейшую роль в углеродном цикле, поскольку обеспечивают значительное поглощение углерода из атмосферы.

Этот процесс способствует снижению концентрации парниковых газов и смягчает последствия изменения климата.

В условиях нарастающего антропогенного воздействия на природные экосистемы понимание механизмов депонирования углерода становится особенно важным для прогнозирования атмосферы в глобальном масштабе и разработки эффективных природоохранных стратегий [28].

Основные причины значимости этой темы связаны с несколькими аспектами. Во-первых, леса, будучи ключевыми углеродными резервуарами, способны абсорбировать огромные объемы углекислого газа, что напрямую влияет на климатическую устойчивость. Во-вторых, с изменением климата меняются и экосистемные услуги, которые предоставляют леса. Исследование этих изменений помогает предсказать, как леса будут реагировать на климатические колебания и как эти реакции отразятся на депонировании углерода. Например, известные международные модели, такие как ИЗИС IIASA и EFIMOD, позволяют оценивать запасы углерода и их динамику в зависимости от различных факторов, включая возраст леса и его состав [29].

Осознание того, что лесные экосистемы могут стать как источником, так и «дисконтным» резервуаром углерода, подчеркивает необходимость оптимизации лесоводческих практик. Эффективное управление лесами требует не только учета текущего состояния углеродных запасов, но и понимания, как различные методы ведения лесного хозяйства влияют на депонирование углерода. Например, применение экологизированных подходов к рубкам может способствовать

не только сохранению биологического разнообразия, но и поддержанию углеродного баланса [30].

Ключевыми вопросами, требующими дальнейшего исследования, являются: как изменения в структуре насаждений влияют на углеродный баланс; какие методы оценки депонирования углерода наилучшим образом подходят для специфических лесных экосистем; и как инновационные технологии, такие как спутниковое наблюдение, могут повысить точность оценок углеродных запасов [11]. Кроме того, важно изучать, как районные особенности лесов, такие как тип почвы и климат, влияют на эффективность депонирования углерода.

Таким образом, лесные экосистемы не только играют важнейшую роль в углеродном цикле, но и представляют собой сложные объекты исследований. Изучение их функций и потенциала в контексте изменения климата требует мультидисциплинарного подхода и интеграции знаний из различных областей научных исследований. Углубленное понимание этих процессов будет способствовать разработке практических рекомендаций по управлению лесами в условиях быстро меняющегося климата и обеспечению устойчивости экосистем [32].

Существующие международные модели оценки углерода лесными экосистемами играют важную роль в понимании и прогнозировании углеродного баланса. Модель ИЗИС IIASA, разработанная в Австрии, представляет собой инструмент для оценки потенциального запасов углерода, который использует геоинформационные технологии и позволяет проводить высокоточные оценки [29]. При этом ее применение ограничено из-за зависимости от доступных данных по количеству, росту и типам лесных насаждений в различных регионах.

Модель EFIMOD + ROMUL, используемая в России, уникальна тем, что учитывает различные экологические факторы и особенности лесных экосистем. Она позволяет интегрировать данные о деградации лесов и изменениях в климате, что делает модель более адаптированной к местным условиям. Однако, одним из ее недостатков является ограниченная доступность необходимых данных, которые часто требуют значительных затрат на сбор и обработку [29].

Наряду с этим, модель РОБУЛ, также разработанная в России, направлена на оценку углеродного баланса в лесах и предоставляет доступ к данным о текущем состоянии экосистем и возможных изменениях. Ее основная функция – оценка запасов углерода, хранящегося в различных компонентах лесных экосистем, что позволяет эффективно выявлять районы с высокой уязвимостью и разрабатывать меры по охране и восстановлению лесов. Однако, как и в случае с EFIMOD + ROMUL, необходима большая база данных для точного моделирования [29].

Модель CBM-CFS3, разработанная в Канаде, представляет интерес благодаря своей гибкости и способности ориентироваться на различные типы лесов и условиях. Она активно используется для прогнозирования изменений в углеродном состоянии лесов, но также является достаточно сложной в плане настройки и требует значительных навыков для эффективного использования [29].

FORCARB2, модель, разработанная в США, также обладает своими уникальными характеристиками, сосредоточенными на анализе углеродного баланса в различных типах лесных экосистем. Данная модель, по сравнению с другими, требует меньше данных и может быть более доступной для использования, однако ее точность и предсказательная способность могут быть несколько снижены [29].

Таким образом, несмотря на разнообразие моделей, каждая из них имеет свои преимущества и недостатки в зависимости от контекста и доступных данных. Модели могут значительно варьироваться в применяемых методах и подходах к оценке углерода, что делает выбор между ними важным шагом при проведении исследований и разработке стратегий в области управления лесными ресурсами [33]. Так, в рамках углеродного регулирования и оценки углеродного следа также учитываются аспекты воздействия углеродных налогов и торговли углеродными единицами, которые требуют интеграции данных из различных источников и применения комплексных подходов [34].

Таким образом, современные международные модели для оценки углерода в лесных экосистемах предоставляют значимые инструменты для научного анализа и разработки эффективных методов углеродного регулирования, но успех

их применения во многом зависит от качества исходных данных и контекста местных условий.

Картографирование запасов углерода в лесных экосистемах играет важную роль в управлении природными ресурсами и оценке вклада лесов в углеродный баланс. Разные подходы к этой задаче обеспечивают комплексное понимание распределения углерода в экосистемах. Выделяется четыре основных метода: картографический, конверсионный, дистанционный и модельный. Каждый из них имеет свои преимущества и недостатки, что позволяет комбинировать методы для достижения точности и надежности результатов.

Картографический метод осуществляется через выделение полигонов на географических картах и связывание этих данных с типовыми значениями. Это позволяет рассчитывать общий запас углерода в различных регионах. Существуют также определения базовых подходов к đánh giá запасов углерода в экосистемах. Этот подход позволяет установить взаимосвязь между типами экосистем и объемами запасов углерода, используя локальные данные для вычислений [35].

На Европейско-Уральской части России, где сосредоточены хвойные и широколиственные леса, картографические данные можно получить с помощью геоинформационных систем (ГИС). Современные алгоритмы позволяют анализировать пространственное распределение углерода с высокой точностью. В этом регионе общий запас древесины составляет 4,7 миллиарда м³, что выше среднего по стране показателя. Это подчеркивает важность лесов как перспективного ресурса для углеродного балансирования [42].

Дистанционные методы, такие как Лидер и Спутниковая фотограмметрия, предоставляют возможность сбора данных о верхнем слое лесной растительности и способствуют оценке продуктивности лесов. С их помощью можно отслеживать динамику углеродных запасов и прогресс в их оценке [43]. Данная информация используется для планирования лесных работ и определения одну из важнейших задач – сохранение углеродных депозитов.

Модельный подход включает в себя применение математических моделей, позволяющих экстраполировать данные и проводить прогнозирование на основе имеющихся данных. Это позволяет исследовать возможные изменения запасов углерода под воздействием антропогенных факторов и климатических изменений [36]. Основные ограничения таких моделей заключаются в точности введенных параметров и их актуальности.

Применение геоинформационных технологий в картографировании углеродных запасов является важным направлением для устойчивого лесоуправления. С помощью ГИС возможно создание карт, которые отражают объемы накопления углерода в зависимости от типа леса и состояния экосистемы [37]. Таким образом, картографирование запасов углерода лесными экосистемами не только предоставляет информацию о состоянии лесов, но и способствует выработке стратегий по их эффективному использованию и охране.

Лесные экосистемы России являются существенными участниками глобального процесса смягчения изменений климата, обеспечивая компенсацию почти 27% антропогенных выбросов парниковых газов [38]. Однако отсутствие надежной системы инвентаризации лесов приводит к сложности в эффективном управлении ними, что, в свою очередь, затрудняет учет их углеродной емкости [38].

Основная функция лесов заключается в их способности поглощать углекислый газ (CO2) из атмосферы, тем самым регулируя парниковый эффект. Модели, описывающие зависимость первичной продукции лесов от климатических факторов, таких как температура и уровень осадков, становятся все более актуальными в условиях изменения климата [41]. Исследования показывают, что адаптация лесного хозяйства необходима для поддержания устойчивости лесных экосистем. Это включает в себя как традиционные меры охраны лесов, так и создание новых норм управления, адаптированных к меняющимся условиям окружающей среды [39].

Проблема заключается в том, что климатические изменения требуют внедрения новых подходов к управлению лесными ресурсами. Одним из таких подходов является разработка и внедрение адаптационных мероприятий, включая

усиление традиционных методов охраны лесов и создание климатически адаптированных норм управления [40]. Особое внимание стоит уделить практикам, способствующим сохранению и увеличению углеродного баланса лесов, что может включать как восстановление деградированных экосистем, так и развитие устойчивого лесного хозяйства [36].

Климатические изменения влияют не только на лесные экосистемы, но и на их способность поглощать углерод. Исследования показывают, что изменения в температурных режимах и сезонных осадках непосредственно влияют на рост и продуктивность лесов, что, в свою очередь, отражается на их углеродной емкости. Поддержание высокого уровня углеродного захвата требует постоянного мониторинга и разработки адаптивных подходов к управлению лесными ресурсами [41].

В конечном счете, леса могут служить не только как поглотители углерода, но и как места для реализации стратегий по смягчению последствий изменения климата. Необходимость интеграции экологических норм в законодательство и практики управления ресурсами может стать критическим фактором для сохранения лесных экосистем и обеспечения их устойчивости в условиях изменяющегося климата [39].

Оценка депонирования углерода в лесных экосистемах основана на ряде методологических подходов, которые включают эксперименты, инвентаризации и удаленные методы. Одним из самых распространенных способов является использование моделей, таких как ИЗИС IIASA, EFIMOD + ROMUL, РОБУЛ, CBM-CFS3 и FORCARB2, которые применяются для анализа углеродных запасов в разных экосистемах, включая хвойно-широколиственные леса [35]. Эти модели позволяют не только определять текущие уровни углерода, но и проводить прогнозы, что критически важно для планирования устойчивого управления лесами.

Инвентаризации представляют собой важный компонент оценки углеродного баланса. Данные о состоянии лесов и углеродных запасах собираются путём полевых измерений. Например, в России проводятся регулярные инвентаризации, где замеры проводятся на различных уровнях, от отдельных деревьев до

масштабных площадей [42]. Эти данные используются для расчета коэффициентов, необходимых для определения поглощения углерода, что позволяет оценить эффективность лесов в качестве углеродных синкраторов.

Удаленные методы, такие как дистанционное зондирование и спутниковые технологии, также играют важную роль. Они обеспечивают доступ к данным, которые могут охватывать большие территории и предоставляют актуальную информацию о состоянии лесных экосистем [43]. Такие технологии позволяют оценивать не только количество углерода, но и динамику его изменений, что является ключевым элементом в условиях изменения климата.

Кейс исследования углерода в экосистемах Татарстана продемонстрировал применение различных подходов, направленных на оценку накопления углерода. В работе были обозначены ключевые направления, методы и этапы, соблюдаемые в проведении исследований, что способствует улучшению углерододепонирующей функции лесов [36]. Создание карбоновых стационаров для мониторинга позволяет усилить данные инвентаризаций и предоставляет возможность дальнейшего изучения углеродного баланса в условиях изменяющегося климата.

Методические разработки, основанные на рекомендациях МГЭИК, подчеркивают, что необходимо вести учет поглощения парниковых газов с учетом текущих и прогнозируемых климатических условий. Это требует комплексного подхода и актуализации методов оценки углеродного состояния, чтобы обеспечить надежные данные для экологической политики и управления [37]. Сравнительный анализ данных о запасах углерода в лесных экосистемах может быть произведен на основе различных методов и источников, предоставляющих информацию о фитомассе и углеродных запасах. Например, в исследовании, проведенном на европейской территории России, отмечается изменчивость запасов углерода в автоматных и полугидроморфных почвах, где среднее значение фитомассы сосняков составило 254.2 т/га с диапазоном от 171 до 395 т/га. Запасы углерода варьировались от 85 до 197 т С/га, что в среднем соответствует 126.5 т С/га, что подтверждает необходимость объединения различных методов анализа [35].

Данные о записях углерода в древесине ствола, занимают основную часть углеродного пула, составив от 59.5% до 75.6% от общей надземной фитомассы. Ветви и хвоя имеют значительно меньшую долю в углеродном запасе [42]. Это позволяет уменьшить неопределенность при оценке углерода и выделить наиболее значимые элементы фитомассы, которые вносят наибольший вклад в общую картину.

Анализ проведенных оценок запаса углерода в лесных экосистемах показывает, что общая оценка запасов углерода в надземной и подземной фитомассе на 2020 год составила 46.9±0.4×10^9 т С, среднее значение 52.1±0.5 т С/га. Данная оценка находится на 35% выше по сравнению с предыдущими данными, что указывает на необходимость пересмотра существующих моделей оценки запасов углерода [43]. Это подтверждает, что данные, полученные с использованием разных методов, могут значительно варьироваться.

Важность валидности данных становится очевидной, когда рассматриваются различные методы измерения. Полученные результаты анализа подтверждают, что данные о запасах углерода в данной области требуют дальнейшего исследования и проверки. В частности, выявленная разница в запасах углерода может указывать на необходимость использования различных подходов и стратегий для более точной оценки и прогнозирования углерода в лесных экосистемах [36].

Таким образом, при сравнительном подходе необходимо учитывать различные факторы, такие как условия местопроизрастания, возраст лесных массивов и методики исследования. Важно, чтобы будущие исследования сосредоточились на стандартизации методов измерения и объединении полученных данных для более точной оценки депонирования углерода в лесах [37].

В Европейско-Уральской части России депонирование углерода в лесных экосистемах представляет собой сложную и многофакторную задачу, требующую детального анализа. По данным проведения исследований, углеродный пул лесов в этом регионе варьируется от 28 до 50 миллиардов тонн, и годичный уг-

леродный сток определён в пределах 58–429 миллионов тонн [35]. Эти показатели показывают, что леса в данной области играют важную роль в углеродной бюджетной системе страны.

Для юго-западного Урала была создана база данных о биомассе лесов, которая оценивает общую углеродную емкость в 267,5 миллиона тонн с ежегодными запасами в 18,2 миллиона тонн [42]. Это подтверждает, что лесные экосистемы не только служат хранилищем углерода, но и регулярно дополняют этот запас благодаря фотосинтезу.

Почвы также имеют существенное значение в вопросе углеродного депонирования. Оценки показывают, что запасы углерода в почвах для глубин 0–30 см составляют около 19,3 миллиарда тонн, а для глубин 0–100 см – 34,2 миллиарда тонн [43]. Такие данные подчеркивают, что необходимо учитывать как лесные массы, так и почвенные ресурсы при расчетах углеродных потоков.

Исследования также сосредоточены на картографировании углеродных запасов в лесных экосистемах, что является важным этапом для понимания пространственного распределения углерода [36]. Современные программные решения позволяют создавать точные карты, что может обернуться в значительное улучшение в системах управления лесным хозяйством и экосистемами в целом.

Более того, общая площадь лесного фонда в европейской части России оценивается примерно в 181,13 миллиона гектаров [37]. Это подчеркивает важность этой территории как одного из крупных углеродных резервуаров страны. Для повышения точности и качества исследований создаются системы анализа распределения углерода, что способствует более глубокому пониманию процессов депонирования и утилизации углерода в лесах региона, тем самым формируя основы для будущих стратегий управления и охраны лесов.

Оценка и прогнозирование углерода в лесных экосистемах требуют комплексного подхода, который включает специфические методы, адаптированные к экологическим условиям и типам лесов. Для практиков важно учитывать, что модели оценки углерода, такие как ИЗИС IIASA и EFIMOD + ROMUL, обеспечивают точные расчеты, однако их применение требует понимания локальных

условий [35]. Рекомендуется проводить верификацию расчетов, используя наземные методики для калибровки результатов моделей.

Тщательный подход к оценке запасов углерода не ограничивается только инвентаризацией древесных запасов. Необходимо обращать внимание на параметры почвы и подлеска, так как углерод, находящийся в этой среде, также играет важную роль в углеродном балансе. Важно учитывать, что, например, в засушливых условиях доля неорганического углерода может составлять до 25% от общего количества углерода в почвах [36]. Поэтому интеграция данных о почвах в модели оценки является необходимой.

Помимо этого, оценка не должна ограничиваться стационарными показателями. Включение в процесс оценки динамики изменения углерода из-за разложения корней и отмирания растительности позволяет более точно предсказывать углеродный поток [42]. Использование новейших методик, таких как метод определения запасов углерода в биогеоценозах, станет одним из ключевых элементов для лесоводов, работающих в условиях аридных экосистем [37].

Стоит отметить, что внедрение новых технологий, таких как дроновые съемки и спутниковый мониторинг, открывает новые горизонты для оценки углерода. Это позволяет быстро и без значительных затрат собирать большие объемы данных о состоянии экосистем и динамике углеродного цикла [3]. Оснащение лесоводов такими инструментами должно быть приоритетным направлением, что обеспечит более качественный контроль за состоянием лесных экосистем и улучшит углеродное моделирование.

Так как климатические изменения оказывают влияние на углеродный обмен, важно учитывать потенциальные изменения температуры и осадков в прогнозах. Следует применять сценарные методики, которые учитывают различные климатические изменения, для получения наиболее полных и точных предсказаний [36]. Рекомендуется также регулярно пересматривать параметры моделей, чтобы адаптировать их к изменяющемуся климату и экосистемным условиям.

Синергия между теоретическими исследованиями, практическими методами и новыми технологиями – это подход, который позволит более эффективно

оценивать и прогнозировать углеродные запасы в лесных экосистемах, адаптируя существующие методики к конкретным условиям исследования. Важно, чтобы практики сохраняли гибкость в подходах и оставались открытыми для новых знаний, что обеспечит успешное поддержание и улучшение углеродного баланса в экосистемах.

Лесные экосистемы играют ключевую роль в углеродном цикле, выступая как важные поглотители углерода, что делает их незаменимыми в борьбе с изменением климата. В ходе исследования были рассмотрены международные модели оценки углерода, такие как ИЗИС IIASA, EFIMOD + ROMUL и РОБУЛ, которые предоставляют различные подходы к количественной оценке запасов углерода в лесах. Эти модели позволяют не только оценить текущие запасы углерода, но и прогнозировать их изменения в зависимости от различных факторов, таких как изменение климата, антропогенное воздействие и управление лесными ресурсами.

Анализ картографических данных о запасах углерода в хвойно-широколиственных лесах Европейско-Уральской части России показал, что эти экосистемы обладают значительным потенциалом для депонирования углерода. Картографирование позволяет визуализировать распределение углерода в лесах, что, в свою очередь, способствует более эффективному управлению лесными ресурсами и разработке стратегий по их охране. Важно отметить, что лесные экосистемы не только способствуют снижению концентрации углерода в атмосфере, но и поддерживают биологическое разнообразие, что является критически важным для устойчивости экосистем в условиях изменения климата.

Значимость лесных экосистем для климатических изменений не может быть переоценена. Они не только поглощают углерод, но и обеспечивают множество экосистемных услуг, таких как очистка воздуха, поддержание водного баланса и сохранение почвенного плодородия. В связи с этим, разработка рекомендаций по оценке углерода становится актуальной задачей. В ходе работы были предложены методы, которые могут быть использованы для более точной

оценки углерода в лесных экосистемах, включая использование дистанционного зондирования и моделирования.

Сравнительный анализ данных, проведенный в рамках исследования, позволил выявить ключевые факторы, влияющие на депонирование углерода, а также оценить эффективность различных методов оценки. Это, в свою очередь, может служить основой для дальнейших исследований и разработки более точных моделей, которые учитывают специфику различных лесных экосистем.

Важно понимать, что лесные экосистемы представляют собой важный элемент в стратегии борьбы с изменением климата. Их способность к депонированию углерода и поддержанию биологического разнообразия делает их объектом пристального внимания как со стороны научного сообщества, так и со стороны государственных структур. Необходимость в разработке и внедрении эффективных методов оценки углерода в лесах становится все более актуальной, что подчеркивает важность дальнейших исследований в этой области.

ГЛАВА 2. ФОРМИРОВАНИЕ И АНАЛИЗ ДАТАСЕТА ДЛЯ МАШИННОГО ОБУЧЕНИЯ

В современном мире, где объемы данных растут с каждым днем, машинное обучение становится неотъемлемой частью множества областей, включая медицину, финансы, маркетинг и многие другие. Эффективность алгоритмов машинного обучения во многом зависит от качества данных, на которых они обучаются. Поэтому формирование и анализ датасета представляют собой ключевые этапы в процессе разработки моделей, способных решать сложные задачи.

Следует отметить, что многие проекты в области машинного обучения терпят неудачу именно из-за недостаточного внимания к этапам подготовки данных. Неправильно собранные, очищенные или организованные данные могут привести к искажению результатов и, как следствие, к неверным выводам. Поэтому важно не только знать, как использовать алгоритмы машинного обучения, но и понимать, как правильно формировать и обрабатывать данные, чтобы обеспечить их высокое качество.

Определение четких целей является важным первым шагом в процессе сбора данных для машинного обучения. Ясные цели помогают определить, какие данные необходимы, какие источники использовать и каким образом организовывать процесс сбора. Без четкого понимания конечной задачи, процесс может стать неэффективным, результатом чего часто становятся неполные или нерелевантные данные, что негативно сказывается на производительности моделей.

Для примера, в проекте по распознаванию лиц необходимо заранее установить, какие именно характеристики лиц (такие как форма носа, цвет глаз, и т. д.) будут иметь решение о том, какие данные собирать. Если цель проекта нечетко сформулирована, команда может пропустить сбор данных, необходимых для адекватного обучения модели, что приведет к недостаточной точности распознавания [44]. Введение правильных целей в начале процесса способствует

не только качественному сбору данных, но и улучшению взаимодействия между членами команды, так как все понимают общую картину и задачи.

Принципы целеполагания особенно важны в контексте ограниченного доступа к данным. Например, многие исследователи сталкиваются с проблемами доступа к необходимым наборам данных, что приводит к необходимости разработки собственных методов сбора и подготовки данных [14]. Команды, которые заранее четко определили свои цели, могут более эффективно разрабатывать стратегии сбора информации, адаптируя свои методы к специфике задач, что в свою очередь снижает риск неэффективных трат времени и ресурсов.

Касаясь реальных примеров − в некоторых стартапах для создания моделей машинного обучения четко обозначали свои цели, что позволяло минимизировать трудозатраты и быстро адаптироваться к изменениям на рынке [45]. Так, четкие цели помогали не только в процессе сбора данных, но и в дальнейших этапах, таких как анализ и обработка собранной информации.

Еще один аспект, который следует учитывать, касается специфики данных, необходимых для разных задач получения информации. Например, в медицине целью сбора данных может быть конкретизация признаков и симптомов заболеваний, что требует дополнительной подготовки данных и обдуманных методов сбора [46]. Неправильное определение целей может привести к сбору данных, которые не соответствуют необходимым критериям, что ставит под угрозу успешность всего проекта.

В итоге, формулировка правильных целей не только помогает оптимизировать процесс сбора данных, но и обеспечивает более целенаправленный и гармоничный подход на всех стадиях проекта. Хорошо задействованный процесс целеполагания позволяет командам сосредоточиться на ключевых аспектах, улучшая согласованность и понимая, какие навыки и ресурсы нужны для успешного завершения задач. Таким образом, правильные цели сбора данных закладывают основу для успешного сбора информации.

После определения целей следующим шагом становится сбор необходимых данных, который напрямую влияет на качество и точность моделей машинного обучения. Эффективный процесс сбора предполагает использование различных методов и источников информации, включая как структурированные, так и неструктурированные данные. Одним из распространенных подходов является использование открытых наборов данных, которые часто доступны для исследователей и разработчиков. Однако необходимо учитывать, что данные из таких источников могут иметь ограничения, связанные с качеством и актуальностью, что требует внимательного анализа [47].

Кроме открытых источников, существует возможность сбора данных напрямую через проведение опросов, экспериментов или наблюдений. Эти методики позволяют получать информацию, максимально адаптированную под конкретные задачи, однако они нередко требуют значительных временных и финансовых затрат. Также следует подумать о правовых аспектах, таких как соблюдение законов о защите данных, что добавляет дополнительную сложность к этому процессу [48].

Кроме того, важно учитывать разнообразие собираемых данных. Например, при разработке систем распознавания лиц может потребоваться аннотирование изображений для создания обучающего набора. Этот этап хоть и может показаться несущественным, на самом деле является критично важным для достижения высоких результатов: качество аннотаций существенно влияет на ход дальнейшей работы с моделью [47]. Практические примеры показывают, что использование единых стандартов в аннотировании и маркировке данных может значительно повысить качество итогового результата [49].

Кроме классификации данных, необходимо уделить внимание вопросам управления качеством на всех этапах сборной информации. Наличие невостребованных или неточных данных может привести к необоснованным выводам действенности модели, что сильно подрывает доверие к результатам анализа. Для решения этой проблемы могут использоваться методы предварительной

фильтрации и контроль за ними, которые обеспечивают соответствие данных заданным требованиям [50].

Во время формирования датасета важно пройти все этапы, включая подготовку, сбор и аннотирование данных, тщательно проверяя и документируя каждое действие. Пренебрежение одним из этих этапов может значительно снизить результативность конечного продукта. Таким образом, важно помнить, что тщательная проработка процесса сбора данных станет фундаментом качественного билдирования будущих моделей машинного обучения. Эффективный сбор данных является важной частью формирования качественного датасета.

Очистка собранных данных − это следующий критически важный этап в процессе подготовки к машинному обучению. На этом этапе проводятся мероприятия, направленные на устранение различных проблем, связанных с качеством данных. Одной из распространенных проблем являются шумовые данные, которые могут возникать в результате ошибок при сборе или передаче информации. Шумовые данные могут негативно повлиять на работу моделей, снижая их точность. Способы фильтрации шума включают различные статистические методы, такие как медианное фильтрование и анализ временных рядов [51].

Пропуски в записях также представляют собой серьезную проблему. Они могут возникать по самым различным причинам: техническим сбоям, человеческим ошибкам или просто недостаточному уровню сбора данных. Заполнить пропуски можно путем интерполяции или использования методов машинного обучения, таких как KNN, чтобы предсказать недостающие значения на основе имеющихся данных [47]. Такой подход позволяет сохранить информацию и улучшить качество набора данных.

Визуализация данных на этом этапе также может оказаться полезной. Графический анализ позволяет выявить аномалии и несоответствия, которые могут не быть очевидными при линейном просмотре данных. Построение диаграмм разброса и гистограмм может помочь идентифицировать выбросы или незаключённые зависимости в данных [49].

Классификация и сегментация данных также важны для очистки. Эти методы позволяют структурировать данные, снижая шум и упрощая дальнейший анализ. Классификация может облегчить процесс обработки больших объемов информации, а сегментация поможет разбить сложные наборы данных на более управляемые части [52].

Использование специализированных инструментов, таких как GDAL и OpenCV, может существенно облегчить задачу очистки данных, особенно в области обработки изображений и дистанционного зондирования [44]. Эти инструменты предлагают готовые решения для корректировки искажений и фильтрации шумов, что снижает объем ручной работы и вероятность ошибок.

Важно учитывать, что все процессы очистки и обработки данных напрямую влияют на итоговое качество модели машинного обучения. Без их выполнения можно получить не только неточные, но и искажающие информацию данные, которые приведут к ошибочным выводам и результатам. Процесс очистки определяет качество финального датасета.

После очистки данные необходимо организовать для удобства дальнейшего анализа и обеспечения эффективной работы алгоритмов машинного обучения. Организация данных влияет на их представление, что, в свою очередь, может существенно отразиться на результатах моделей. Важно выбрать формат, в котором данные будут храниться, учитывая, что разные алгоритмы могут требовать различных структур. Например, для задач классификации удобно использовать форматы табличных данных, где строки представляют образцы, а столбцы – признаки [47].

Один из распространенных способов организации датасета – это создание обучающей, валидационной и тестовой выборок. Обучающая выборка используется для обучения модели, валидационная – для настройки гиперпараметров и предотвращения переобучения, а тестовая позволяет оценить итоговую производительность модели. Такой подход позволяет более точно оценить качество модели на данных, которые не использовались в процессе ее обучения, что критически для проверки ее обобщающей способности [48].

Важным аспектом является аннотирование данных, что подразумевает добавление метаданных, таких как разметка классов или дополнительных характеристик образцов. Это необходимо для задач, где результат зависит от корректной интерпретации данных. Качество аннотаций определяет, насколько эффективно модель сможет учиться, и может влиять на итоговую точность [49].

Формат хранения данных также играет роль: например, многие библиотеки для машинного обучения, такие как TensorFlow и PyTorch, предлагают удобные инструменты для работы с форматами TFRecord или HDF5, которые обеспечивают быструю загрузку и эффективное хранение данных. Эти форматы позволяют обработать большие объемы данных, делая возможной работа с более сложными моделями, например, глубокими нейронными сетями [50].

Кроме того, стоит учитывать необходимые преобразования данных, такие как нормализация или стандартизация. Это позволяет улучшить сходимость оптимизационных алгоритмов и повысить общую эффективность обучения [53]. Также полезно применять методы аугментации данных, особенно в задачах, связанных с изображениями – это обеспечивает разнообразие образцов и уменьшает вероятность переобучения модели.

Организация датасета является завершающим шагом перед его использованием в моделях.

Ошибки на каждом этапе формирования могут серьезно повлиять на качество получаемого датасета и, как следствие, на результаты работы моделей машинного обучения. Первая ошибка часто возникает на этапе определения необходимых данных. Неправильная интерпретация задач способствует созданию нерелевантных наборов данных, которые не удовлетворяют требованиям задачи. Например, определение слишком узкой категории или избыточной информации приводит к тому, что модель не может создать обобщение, способное работать с реальными данными [54].

Ошибки в сборе данных могут возникнуть из-за технических неполадок или некорректного подхода к выборке. Применение недостаточно репрезента-

тивного выборочного метода способно исказить результаты, делая данные односторонними. В конечном итоге, это может привести к ухудшению показателей модели из-за недостаточной обучающей выборки [55]. Также, как показывает опыт, недостаточное внимание к различным источникам данных может привести к недостающим или ошибочным записям, что значительно ограничивает обучаемость модели [56].

На этапе разметки данных могут произойти ошибки, связанные с человеческим фактором. Неправильные метки или ошибка в определении классов могут негативно отразиться на точности классификации. Наши исследования показывают, что до 87% неудачных проектов возникают из-за неправильной разметки данных, что ставит под сомнение сам процесс обучения [56]. Допускаемая степень ошибки может привести к тому, что модель начнет работать по неправильным алгоритмам, что, несомненно, скажется на итоговых результатах.

Также стоит отметить важность этапа очистки данных. Недостаточный анализ или нерегулярное удаление дубликатов, пустых строк и аномалий существенно ухудшают качество датасета. Когда данные загрязнены, это может привести к тому, что модели преувеличивают редко встречающиеся значения, что в свою очередь, делает их менее адаптируемыми к новым данным [50].

Кроме того, на этапе организации данных важно учитывать наличие метаданных, описывающих структуру и значимость каждого элемента. Их отсутствие затрудняет работу даже с качественными данными, теряя контекст и дополнительные важные сведения о каждом из них [53].

Таким образом, ошибки на разных этапах формирования датасета могут создать сложные для преодоления последствия. Правильная разметка, выборка и обработка данных, а также их организация играют критически важную роль в успехе проекта, поскольку именно от этого зависит, как именно модель будет понимать данные и способны ли они обеспечить высокие показатели [56]. Избежание этих ошибок имеет ключевое значение для успеха проекта.

Отметим распространенные ошибки, которые могут возникнуть на каждом из этапов формирования датасета. К ним относятся недостаточное внимание к качеству данных, игнорирование этапа очистки, а также неправильная организация данных. Изучение этих ошибок и их последствий позволяет не только избежать их в будущем, но и улучшить общий процесс работы с данными.

Таким образом, формирование и анализ датасета для машинного обучения – это сложный и многогранный процесс, требующий внимательного подхода на каждом этапе. Успех в этой области зависит от четкого понимания целей, качественного сбора и обработки данных, а также правильной организации и представления полученного датасета. Важно помнить, что качественный датасет – это основа для успешного обучения моделей и получения достоверных результатов в области машинного обучения и аналитики.

В контексте машинного обучения датасеты выполняют несколько ключевых функций [57]:

- обучение моделей распознаванию паттернов и закономерностей;
- валидация работоспособности и эффективности обученных моделей;
- тестирование моделей на новых, не встречавшихся ранее данных;
- сравнение различных алгоритмов машинного обучения между собой.

Значение качественных датасетов сложно переоценить. Исследования показывают, что улучшение качества и увеличение объема данных зачастую дает больший прирост в точности работы моделей, чем усложнение самих алгоритмов.

Для эффективного обучения моделей машинного обучения датасеты должны соответствовать ряду требований [58].

1. **Репрезентативность** – данные должны адекватно отражать реальную популяцию или явление, которое моделируется. Нерепрезентативные выборки могут привести к систематическим ошибкам в работе моделей.

2. **Релевантность** – данные должны содержать информацию, значимую для решаемой задачи. Включение нерелевантных признаков может увеличить вычислительную сложность и снизить качество модели.

3. **Достаточный объем** – количество наблюдений должно быть достаточным для выявления статистически значимых закономерностей. Необходимый объем выборки зависит от сложности задачи и количества признаков.

4. **Сбалансированность** – для задач классификации важно, чтобы различные классы были представлены в достаточном количестве. Дисбаланс классов может привести к смещению модели в сторону доминирующего класса.

5. **Точность и достоверность** – данные должны быть свободны от ошибок измерения, ввода и других искажений, способных повлиять на качество обучения.

6. **Актуальность** – данные должны отражать текущее состояние моделируемой системы или явления, особенно в динамически меняющихся областях.

7. **Согласованность** – данные должны быть внутренне непротиворечивыми и соответствовать заданным ограничениям целостности.

8. **Полнота** – минимизация пропущенных значений критически важна для большинства алгоритмов машинного обучения.

В машинном обучении используются различные типы данных, каждый из которых имеет свои особенности и требует специфических подходов к обработке.

1. **Числовые данные**:

- **непрерывные** (интервальные и относительные) – измеряемые величины с бесконечным числом возможных значений (вес, рост, температура);
- **дискретные** – целочисленные значения, обычно представляющие количество чего-либо (число детей, количество комнат).

2. **Категориальные данные**:

- **номинальные** – обозначают категории без естественного порядка (цвет, пол, национальность);
- **порядковые** – обозначают категории с естественным порядком (уровень образования, стадия заболевания).

3. **Текстовые данные** – неструктурированный или слабоструктурированный текст, требующий специальных методов обработки и векторизации.

4. **Временные ряды** – последовательности измерений, упорядоченные по времени (цены акций, метеорологические данные).

5. **Пространственные данные** – информация, привязанная к географическим координатам или иным пространственным ориентирам.

6. **Графовые данные** – информация о взаимосвязях между объектами (социальные сети, молекулярные структуры).

7. **Мультимедийные данные**:

- изображения;
- аудио;
- видео.

Каждый тип данных требует специфического подхода к предварительной обработке, анализу и выбору соответствующих алгоритмов машинного обучения. Например:

- категориальные данные обычно требуют кодирования (one-hot, label encoding);
- текстовые данные нуждаются в векторизации (TF-IDF, word embeddings);
- временные ряды часто требуют выделения сезонности, тренда и других компонент;
- изображения обычно обрабатываются с помощью сверточных нейронных сетей, требующих предварительной нормализации и аугментации.

Современные источники данных для формирования датасетов многообразны и включают [59]:

1. **открытые репозитории данных**:

- Kaggle (https://www.kaggle.com/datasets);
- UCI Machine Learning Repository (https://archive.ics.uci.edu/ml/);
- Google Dataset Search (https://datasetsearch.research.google.com/);
- Data.gov и аналогичные государственные порталы;
- GitHub (многочисленные репозитории с данными).

2. **API-интерфейсы**:

- социальные сети (Twitter API, Facebook Graph API);
- финансовые данные (Yahoo Finance API, Alpha Vantage);
- погодные данные (OpenWeatherMap, Weather API);
- транспортные данные (Яндекс.Расписания API, Uber API).

3. **Веб-скрапинг** – извлечение данных с веб-страниц с помощью специализированных инструментов и библиотек (Beautiful Soup, Scrapy, Selenium).

4. **Сенсорные устройства и IoT**:

- мобильные устройства;
- носимая электроника;
- промышленные датчики;
- умные дома и города.

5. **Данные организаций**:

- транзакционные системы;
- CRM-системы;
- логи сервисов и приложений;
- внутренние базы данных.

6. **Генеративные модели** – создание синтетических данных с помощью алгоритмов (GAN, VAE).

7. **Краудсорсинг** – привлечение людей для создания или разметки данных (Amazon Mechanical Turk, Toloka).

Процесс извлечения и агрегации данных включает несколько этапов и методов.

1. **Прямая загрузка** – самый простой способ, применимый для готовых наборов данных из репозиториев.

2. **API-интеграция**:

- использование SDK и библиотек для работы с API;
- REST API запросы с параметрами;
- работа с пагинацией и ограничениями API;
- обработка ответов (JSON, XML).

3. **Веб-скрапинг**:

- анализ структуры HTML-страницы;
- извлечение данных с помощью селекторов (CSS, XPath);
- обход ограничений (капчи, блокировки IP);
- соблюдение robots.txt и политик сайтов.

4. **Работа с базами данных**:

- SQL-запросы к реляционным базам;
- агрегирующие запросы;
- работа с NoSQL хранилищами.

5. **ETL-процессы** (Extract, Transform, Load):

- извлечение данных из различных источников;;
- трансформация в единый формат;
- загрузка в хранилище или аналитическую систему.

6. **Потоковая обработка**:

- работа с потоковыми API (Twitter Stream API);
- использование систем типа Kafka, RabbitMQ;
- обработка в реальном времени.

7. **Распределенное извлечение данных**:

- параллельная обработка больших объемов;
- использование фреймворков типа Apache Spark, Hadoop.

Пропущенные значения (missing values) – распространенная проблема в реальных датасетах, способная существенно повлиять на качество моделей машинного обучения.

Основные подходы к работе с пропущенными значениями [60].

1. **Анализ причин пропусков**

- **MCAR (Missing Completely At Random)** – пропуски полностью случайны;
- **MAR (Missing At Random)** – вероятность пропуска зависит от наблюдаемых данных;

- **MNAR (Missing Not At Random)** – вероятность пропуска зависит от значения самого пропущенного элемента.

2. **Методы обработки**

- **Удаление**:

− удаление строк с пропусками (listwise deletion);

− удаление признаков с большим количеством пропусков;

− попарное удаление (pairwise deletion) – использование всех доступных данных для каждой пары переменных.

- **Заполнение (импутация)**:

− Статистические методы:

- среднее, медиана, мода;
- регрессионная импутация;
- стохастическая регрессионная импутация.

− Алгоритмические методы:

- K-nearest neighbors (KNN) импутация;
- случайный лес (Random Forest) импутация;
- MICE (Multiple Imputation by Chained Equations).

− Временные ряды:

- интерполяция (линейная, сплайны);
- предыдущее/следующее значение (forward/backward fill).

- **Использование индикаторов пропусков** − создание дополнительных бинарных признаков, указывающих на наличие пропуска.

Выбор оптимальной стратегии обработки пропущенных значений зависит от типа данных, характера пропусков, их количества и конкретной задачи машинного обучения.

Выбросы (outliers) – наблюдения, значительно отклоняющиеся от общей закономерности в данных. Их присутствие может существенно искажать статистические показатели и негативно влиять на качество моделей.

3. **Методы выявления выбросов**

- **Статистические методы**:

- Z-оценка (стандартизованные отклонения);
- метод межквартильного размаха (IQR);
- метод Тьюки (boxplot);
- критерий Граббса;
- критерий Чаувенета.

- **Визуальные методы**:

- диаграммы размаха (boxplot);
- гистограммы распределения;
- Q-Q графики;
- диаграммы рассеяния.

- **Алгоритмические методы**:

- изолирующий лес (Isolation Forest);
- локальный фактор выброса (LOF);
- One-class SVM;
- DBSCAN;
- автоэнкодеры для многомерных данных.

Нормализация и стандартизация – важные этапы предобработки числовых данных, особенно для алгоритмов, чувствительных к масштабу признаков (например, методы на основе расстояний).

1. **Методы нормализации**

- **MIN-MAX нормализация** (приведение к диапазону [0,1]) −

X_norm = (X – X_min) / (X_max – X_min).

- **Масштабирование к диапазону [-1,1]** −

X_norm = 2 * ((X – X_min) / (X_max – X_min)) – 1.

- **Десятичное масштабирование** −

$X\_norm = X / 10^n$,

где n выбирается так, чтобы максимальное абсолютное значение было <1.

2. **Методы стандартизации**

- **Z-стандартизация** (приведение к распределению с μ=0, σ=1) −

X_std = (X – μ) / σ.

- **Масштабирование к единичной дисперсии** −

X_std = X / σ.

- **Масштабирование к медиане и MAD** −

X_std = (X – median(X)) / MAD,

где MAD – медианное абсолютное отклонение.

3. **Нелинейные преобразования**

- **Логарифмическое преобразование** −

X_transformed = log(X + c).

где c – константа (обычно 1) для обработки нулевых значений

- **Преобразование Бокса-Кокса** −

X_transformed = (X^λ – 1) / λ, если $\lambda \neq 0$,

X_transformed = log(X), если λ = 0.

- **Извлечение корня** −

X_transformed = X^(1/n).

4. **Робастные методы масштабирования**

- **RobustScaler**(на основе квантилей) −

X_robust = (X – Q2) / (Q3 – Q1),

где Q1, Q2, Q3 – квартили распределения.

5. **Практические аспекты**

➢ Выбор метода зависит от распределения данных и используемого алгоритма.

➢ Трансформации должны применяться отдельно к обучающей и тестовой выборкам.

➢ Параметры трансформации (min, max, μ, σ) должны вычисляться только на обучающих данных.

- Некоторые алгоритмы (например, деревья решений) нечувствительны к масштабу признаков.

Исследовательский анализ данных (Exploratory Data Analysis, EDA) представляет собой комплексный подход к изучению свойств датасета перед построением моделей. Основная цель EDA – понимание структуры данных, выявление закономерностей, проверка гипотез и определение потенциальных проблем.

1. **Основные компоненты EDA**

- **Описательная статистика**:
  - Меры центральной тенденции (среднее, медиана, мода).
  - Меры разброса (стандартное отклонение, дисперсия, квартили).
  - Форма распределения (асимметрия, эксцесс).
  - Корреляции между признаками.
- **Анализ распределений**:
  - Одномерные распределения отдельных признаков.
  - Многомерные распределения и взаимосвязи.
  - Проверка на нормальность и другие теоретические распределения.
- **Выявление структуры данных**:
  - Кластеры и группировки.
  - Временные или пространственные тренды.
  - Иерархические структуры.
- **Анализ аномалий**:
  - Выбросы и нетипичные наблюдения.
  - Несогласованности в данных.
  - Необычные паттерны.

2. **Методологический подход к EDA**

- **Итеративность** – последовательное углубление анализа.
- **Комбинирование** количественных и качественных методов.
- **Формирование и проверка гипотез.**
- **Синтез результатов** для целостного понимания данных.

3. **Типичный процесс EDA**

- Начальное изучение структуры данных (размерность, типы).
- Анализ пропущенных значений и выбросов.
- Расчет и визуализация описательных статистик.
- Анализ распределений отдельных переменных.
- Изучение взаимосвязей между переменными.
- Сегментация данных для более детального анализа.
- Формулирование гипотез для дальнейшего моделирования.

Визуализация данных – мощный инструмент анализа, позволяющий быстро идентифицировать паттерны, тренды и аномалии, которые могут быть неочевидны при работе с табличными данными.

1. **Одномерные визуализации**

- **Гистограммы** – для отображения распределения непрерывных переменных.
- **Графики плотности** – сглаженная версия гистограмм.
- **Боксплоты (ящики с усами)** – для отображения квартилей и выбросов.
- **Виолинплоты** – комбинация боксплота и графика плотности.
- **Столбчатые диаграммы** – для категориальных переменных.
- **Круговые диаграммы** – для отображения долей категорий.

2. **Двумерные визуализации**

- **Диаграммы рассеяния (scatter plots)** – для отображения взаимосвязи двух непрерывных переменных.
- **Линейные графики** – для отображения временных рядов или трендов.
- **Тепловые карты (heatmaps)** – для визуализации матриц корреляций или других матричных данных.
- **Контурные графики** – для отображения плотности распределения в двумерном пространстве.
- **Группированные боксплоты** – для сравнения распределений по категориям.

3. **Многомерные визуализации**

- **Пузырьковые диаграммы** – расширение scatter plot с дополнительным измерением в виде размера точки.
- **Параллельные координаты** – для визуализации многомерных данных на 2D плоскости.
- **Радарные диаграммы** – для сравнения многих переменных.
- **Treemaps** – для иерархических данных.
- **Графы и сетевые визуализации** – для отображения связей между объектами.

4. **Методы снижения размерности для визуализации**

- **PCA (Principal Component Analysis)** – линейное снижение размерности.
- **T-SNE (t-Distributed Stochastic Neighbor Embedding)** – нелинейное снижение размерности с сохранением локальной структуры.
- **UMAP (Uniform Manifold Approximation and Projection)** – современная альтернатива T-SNE.

5. **Интерактивные визуализации**

- **Интерактивные дашборды** – для комплексного отображения данных.
- **Анимированные графики** – для отображения изменений во времени.
- **Фильтрация и детализация** – для исследования подмножеств данных.

6. **Принципы эффективной визуализации**

- **Простота** – минимизация визуального шума.
- **Точность** – корректное представление данных без искажений.
- **Ясность** – четкое выделение основного сообщения.
- **Эффективность** – выбор наиболее подходящего типа визуализации для конкретных данных.

Завершающий этап это корректное разделение данных на обучающую, валидационную и тестовую выборки является фундаментальным условием для построения надежных моделей машинного обучения и объективной оценки их эффективности.

ГЛАВА 3. ТЕХНОЛОГИЯ ML.NET

ML.NET, предложенная Microsoft в 2019 г., представляет собой инновационный инструмент для машинного обучения, специально созданный для разработчиков, работающих в среде .NET. Эта платформа предоставляет возможность интегрировать функционал машинного обучения в существующие приложения без необходимости глубоких знаний в области data science и математики, тем самым открывая доступ к развитым алгоритмам анализа данных и обработки информации.

ML.NET обертывает алгоритмы машинного обучения, позволяя чаще всего реализовывать использование этих алгоритмов через однократный вызов функции.

ML.NET реализует:

- технологии для классификационных задач и категоризационных задач;
- технологии регрессионного анализа для прогнозирования последовательностей;
- технологии выявления аномалий;
- рекомендательные технологии;
- технологии временные рядов;
- методы распознавания образов в изображениях;
- техники категоризации и анализа текстовых данных.

ML.NET – это фреймворк для машинного обучения в рамках Visual Studio 2019, представляющий собой компонент технологии искусственного интеллекта.

ML.NET задействует алгоритмы и статистические подходы из библиотек Visual Studio 2022 для автоматического обучения систем на базе .NET, минуя нужду во внешнем программном обеспечении.

В Visual Studio 2022 ML.NET используется для тренировки машинного автомата и предсказания будущих значений целевой переменной системы.

В Visual Studio 2022 ML.NET в основном применяется для обучения моделей и работы с данными.

В ходе машинного обучения через Visual Studio 2022 данные подаются в систему, после чего в этой же среде определяется наиболее подходящая модель ML.NET для тренировки системы под .NET. По завершении этого процесса ML.NET в Visual Studio 2022 способна осуществлять прогнозы и отображать результаты визуально.

ML.NET поддерживает обучение с учителем и без учителя.

ML.NET представляет собой передовую платформу для машинного обучения, доступную в качестве проекта с открытым исходным кодом, совместимого с различными операционными системами, включая Windows, Linux и macOS. Это решение позволяет разрабатывать настраиваемые модели машинного обучения для широкого спектра приложений, включая консольные и настольные приложения, веб- и мобильные платформы, игры, а также устройства, работающие в рамках концепции Интернета вещей. Одной из ключевых особенностей ML.NET является его интеграция с другими инструментами и фреймворками для машинного обучения, такими как TensorFlow, Accord.NET и Microsoft Cognitive Toolkit (CNTK), что расширяет возможности разработчиков в создании сложных систем ИИ. В последних обновлениях ML.NET были реализованы функции для упрощения загрузки и обработки данных из реляционных баз данных (например, SQL Server, Oracle, MySQL), облегчая тем самым подготовку данных для обучения моделей. Кроме того, последняя версия ML.NET акцентирует внимание на развитии возможностей AutoML, что значительно упрощает разработку моделей машинного обучения, делая их более доступными для разработчиков без глубокой специализации в данной области.

ML.NET предлагает основную концепцию, важную для разработки приложений на основе машинного обучения.

Для точного прогнозирования результатов в ML.NET мы должны загружать обширные наборы данных для тренировки модели. Возможности ML.NET

позволяют использовать различные источники данных для обучения и тестирования, включая текстовые файлы (CSV/TSV), реляционные базы данных (с поддержкой SQL Server, Oracle, MySQL и др.), двоичные файлы, IEnumerable и др.

Образование. Выбор адекватного алгоритма машинного обучения очень важен для разработки эффективной модели, соответствующей заданным целям и задачам, для последующего обучения и предсказания исходов.

Анализ. Для разработки и применения модели машинного обучения обязательно необходимо определить ее задачу. Для группировки данных по схожим характеристикам применяется метод кластеризации. Для предсказания количественных значений, например, цены или стоимости, идеально подходит регрессионный анализ. В то время как для определения принадлежности данных к определенной категории или классу, например, для анализа эмоциональной окраски отзывов, используется классификация.

Ожидаемые исходы. Используя информацию из этапов обучения и тестирования, конечное предсказание делается через приложение ML.NET с применением обученной модели. Эта модель, сохраненная в бинарном файле, может быть легко встроена в другие .NET-приложения, обеспечивая тем самым их расширение и интеграцию с возможностями машинного обучения.

Обсудим актуальные архитектуры и подходы в контексте ML.NET.

В Visual Studio 2022 для установки среды ML.NET используются инструменты пакетного менеджера NuGet.

Перечисляем требуемые программные модули.

```
using Microsoft.ML/*/;
using Microsoft.ML.Data/*/;
using Microsoft.ML.Models;
using Microsoft.ML.Data;
using Microsoft.ML.Trainers;
utilizing Microsoft Machine Learning Transformations; using System;
utilizing System.Collections.Generic namespace;
```

```
using System.Linq;
using C = System.Console;
```

Для набора данных (dataset) создаем структуру хранения в виде класса.

Для обучения и проверки модели используем обучающую и тестовую выборки соответственно, организованные в виде классов.

Разрабатываем класс UnitData для хранения исходных данных и класс UnitPrediction для хранения прогнозируемых результатов.

```
public class UnitData/*/
{
[Column("0")]/*/
public float UnitA/*/;
[Column("1")]
public float UnitS/*/;
[Column("2")]
public float Volume;
[Column("3")]
public string Status;
[Column("4")]
public float Level;
}
public class UnitPrediction
{
[ColumnName("Score")]
public float Level;
}
```

Формируем набор данных, используя функционал GenerateUData.

```
internal static List<UnitData> CreateUDataEntries(int count) {
var pdlist = new List<UnitData>();
Random randomGenerator = new Random));
```

```
for (int index =      0; index < limit; index++)
{
var pd = new UnitData
{
UnitA = random.Next(minValue: 0, maxValue: 20),
UnitS = random.Next(minValue: 0, maxValue: 10),
Volume = random.Next(4, 30)
};
if ((pd.UnitA > 16 || pd.UnitS > 7)      && pd.Volume is > 23)
pd.Status is set to "Warning";
else if ((pd.UnitA < 5      || pd.UnitS < 3)      && pd.Volume < 15)
pd.Status = "Warning";
else
pd.Status = "Standard";
// Выражение вычисляет атрибут «Level» объекта «pd» путем
// суммирования значений атрибутов «UnitA»,  «UnitS» и «Объем»
// того же объекта, а затем вычитания случайного целого числа // от 1 до 2 включительно, сгенерированного «random.Next(1,      3)»
pdlist.Add(pd) ;
}
return pdlist;
}
```

Затем мы формируем файл для оптимальной идентификации и проводим тестирование набора данных. Файл для улучшенной идентификации включает в себя 2000 записей, в то время как тестовый набор содержит 20 записей.

```
var dataSet = CreateUniformData(2000);
var sampleData = CreateUserData(20);
Console.WriteLine("Contents of the Dataset:");
C.WriteLine("identifier,UnitAlpha,UnitSigma, Capacity, Condition, Grade");
```

foreach (var entry in trainData.Take(20))

C.WriteLine(string.Join(",", item.UnitA, item.UnitS, item.Volume, item.State, item.Grade));

Для оптимизации процесса распознавания содержимое файла трансформируется в набор данных, представленный через класс CollectionDataSource.

var trainDataset = CollectionDataSource.Initialize(trainData);

Разработка модели. Инициируем экземпляр LearningPipe- line() и интегрируем в него необходимые элементы модели. В арсенале ML.NET представлен широкий спектр регрессионных алгоритмов для анализа данных. Для наших целей применим алгоритм регрессии Generalized Additive Model Regressor.

var dataProcessingPipeline = new LearningPipeline();

// Добавляем компоненты

// learningPipe, который включает набор данных trainCollection // в свою последовательность

learningPipe.Add(new ColumnMapping(("Level", "Target"))); // Добавьте в learningPipe экземпляр CategoricalOneHotVec- torizer для колонки "Status"

learningPipe.Add(new ColumnConcatenator("Attributes", "UnitA", "UnitS", "Capacity", "Condition"));

// Добавляем алгоритм регрессии learningPipe.Append(new GeneralizedAdditiveModelRegressor());

Упомянутые элементы в обучающем потоке развиваются через применение функции Train().

C.WriteLine("\nInitiating model training:");

// Переменной "model" присваивается результат процесса

// обучения, выполняемого "learningPipe", который использует // "UnitData" в качестве входных данных и прогнозирует // результаты, указанные "UnitPrediction"

Оценка точности. Класс RegressionEvaluator() предназначен для анализа регрессионных моделей, позволяя измерять их производительность через расчет среднеквадратичной ошибки и коэффициента детерминации.

```
var evaluator = new RegressionEvaluator();
var evaluationResults = evaluator.Assess(model, trainingDataset); Console.WriteLine("\^odel Evaluation:");
Console.WriteLine("Root Mean Square: " + metrics.RootMeanSquare); Console.WriteLine("Coefficient of Determination (R^A2):      " +
metrics.RSquared);
```

Предсказание результатов на тестовом наборе данных. Создаем прогностическую модель для тестового набора. Для сопоставления фактических и прогнозных значений выводим их одновременно.

```
var forecast = model.Forecast(testData);
// Отображение тестового набора данных вместе
// с прогнозируемыми результатами
var outcomes = testData.Zip(predicted, (actual, forecast) => Tuple.Create(t.UnitA, t.UnitS, t.Volume, t.Status, t.Level, p.Level));
int i =1;
Console.WriteLine("\nPredicted outcome:");
C.WriteLine("identifier,UnitAlpha,UnitSigma,Capacity,Condition, InitialLevel,ForecastedLevel");
foreach (var item in results)
{
C.WriteLine(string.Concat(i, ",", item.Item1,item.Item2, item.Item3, item.Item4, item.Item5, item.Item6));
}
```

Демонстрировалось применение модели регрессии на языке C# в ML.NET.

Проанализируем пример использования классификации с помощью ML.NET на языке C#.

```
using Microsoft.ML;
using Microsoft.ML.Data;
using Microsoft.ML.Models;
using Microsoft.MachineLearning.Api;
using Microsoft.ML.Trainers;
utilizing Microsoft.ML.DataTransforms;
using System;
utilizing System.Collections.Generic;
using System.Linq;
using C = System.Console;
namespace mltest
{
Public class UnitData
{
[Column("0")]
public float UnitA;
[Column("1")]
public        float UnitS;
[Column("2")]
public        float Volume;
[Column("3")]
public string Status;
[Column("4")]
public float Level;
}
public class ForecastingUnit
{
[ColumnName("Score")]
public float Level;
```

```
}
class Program
{
static void Main(string[] args)
{
// Создание обучающих и оценочных наборов данных
var trainSet = CreateTrainingData(2000);
var testData = CreateUserData(20);
C.WriteLine("Contents of the dataset:");
C.WriteLine("identifier,UnitAlpha,UnitSigma/ Capacity, Condition, Tier");
foreach (var element in trainData.Take(20))
C.WriteLine(string.Concat(item.UnitA, ", ", item.UnitS, item.Volume, item.State, item.Tier));
// Преобразование списка в совокупный источник данных var trainCollection =
CollectionDataSource.Establish(trainData);
// Инициализируем модель машинного обучения
var learningPipe = new LearningPipeline();
// Добавляем компоненты
learningPipe.Incorporate(trainCollection);
learningPipe.Append(new ColumnCopyingEstimator(("Level", "Label")));
learningPipe.Add(new OneHotEncodingTransformer("Status"));
learningPipe.Add(new ColumnConcatenator("Features", "UnitA", "UnitS", "Capacity", "Condition"));
// Реализация модели линейной регрессии
learningPipe.Include(new Generalized Additive Model Regressor());
// Обучение модели
C.WriteLine("\nInitiating model training:");
var model = learningPipe.Fit<UnitData, UnitPrediction>();
```

```
// Оценка производительности модели и проверка точности var evaluator
= new RegressionEvaluator();
var metrics = evaluator.CalculateMetrics(model,
trainingData);
C.WriteLine(("\nAssessment of Model Performance:");
Console.WriteLine("Root Mean Square: " +
metrics.RootMeanSquare);
```

C.WriteLine displays the squared coefficient of determination, indicated by metrics.RSquared.

```
// Результаты прогноза по набору данных var forecasted =
model.Forecast(testingData);
// Отображение тестового набора данных вместе
// с результатами прогноза
var outcome = testData.Zip(predicted, (trueValue, predictedValue) => Tuple.Create(t.UnitA, t.UnitS, t.Volume, t.Status, t.Level, p.Level));
int i = 1;
C.WriteLine("\nForecasted outcome:");
C.WriteLine("identifier,UnitAlpha, UnitSigma,Capacity, Condition,InitialLevel,ForecastLevel");
for each (var element in outcomes)
{
C.WriteLine(string.Join(",", i, item.Iteml, item.Item2, item.Item3, item.Item4, item.Item5, item.Item6));
}
C.ReadLine();
}
internal static List<UnitData> ProduceUnitDataList(int quantity)
{
List<UnitData> pdlist = new List<UnitData>();
```

```
Random random = new Random));
for (int index = 0; index < count; index++)
{
var productDetails = new ProductInfo
{
UnitA = random.Next(0, 20),
UnitS = random.Next(0, 10),
// Переменная "Volume" инициализируется случайным целым
// значением, которое генерируется в диапазоне от 4 до 29
// включительно с использованием "random.Next"
};
// Если свойства pd, в частности, UnitA, превышают 16 или
// UnitS превышают 7, одновременно с объемом pd больше 23, // условие выполняется
pd.Status = "Caution";
else if ((pd.UnitA < 5 || pd.UnitS < 3) && pd.Volume
< 15)
pd.Status = "Warning"; else
pd.Status = "Regular";
pd.Level = (pd.UnitA + pd.UnitS + pd.Volume) – ran- dom.Next(1, 3);
pdlist.Add(pd);
}
return pdlist;
}
}
}
```

Классификация с ML.NET на C#. В Visual Studio инициируем создание консольного приложения и приступаем к интеграции ML.NET, используя NuGet Package Manager для удобной загрузки и установки соответствующего пакета в среду разработки.

```
using Microsoft.ML;
using Microsoft.ML.Data;
using Microsoft.ML.Models;
using Microsoft.MachineLearning.Api;
using Microsoft.ML.Trainers;
implementing Microsoft.ML.Transforms;
using System;
utilizing System.Collections.Generic;
using System.Linq;
```

В процессе создания тестового и обучающего наборов данных для задачи классификации мы компилируем целевые значения в определенные категории. В этом контексте класс ProcessData служит в качестве репозитория для хранения обучающих и тестовых данных, в то время как класс ProcessPrediction предназначен для хранения результатов предсказаний, выполненных на основе этих данных.

```
public class ProcessData
{
[Column("0")]
public float UnitA;
[Column("1")]
public float UnitS;
[Column("2")]
public float Volume;
[Column("3")]
[ColumnName("Label")]
public string Label;
}
public class ProcessForecast
{
[ColumnName("PredictedOutcome")] public string PredictedLabels;
}
```

Применяем метод GeneratePData() для создания примеров данных, что облегчает подготовку файла для точного анализа и оценку качества данных. В этом процессе доступна проверка алгоритма генерации данных.

```
private static List<ProcessData> CreateProcessDataList(int count, bool test = false) {
List<ProcessData> pdlist = new List<ProcessData>();
var randomGenerator = new Random));
for (int index = 0; index < count; index++)
{
ProcessData pd = new var
{
// UnitA присваивается значение с помощью генератора
// случайных чисел в диапазоне от 0 до 119
UnitS = random.Next(0, 60),
Volume = random.NextInt64(40, 280)
};
if ( !test)
{
pd.Label = "Alert";
else if ((pd.UnitA < 25    || pd.UnitS < 20)    && pd.Volume <  100)
pd.Label = "Warning";
else
pd.Label = "Typical";
}
else
pd.Label =   "";
pdlist.Add(pd);
}
return pdlist;
}
```

Для верификации данных внутри используем метод Print(), выводящий элементы списка на экран.

```
private static void Display(IEnumerable<ProcessData> outcomes) {
int i = 1;
Console.WriteLine("id,UnitA,UnitS,Volume,Label");
for (var item in results)
{
Console.WriteLine(string.Join(",", i,item.UnitA, item.UnitS, item.Amount, item.Description));
i++;
}
}
```

Мы генерируем файл, содержащий 2000 элементов для улучшения качества распознавания, и набор из 50 элементов для проверки, используя функцию GeneratePData().

```
var trainData = ProducePseudoData(2000);
Print(trainData.Take(20));
var testData = CreatePseudoData(50, isTest: true);
Print(testData.Take(20) ) ;
```

Мы подготовили наборы данных для обучения и тестирования. Теперь следующим шагом является разработка модели с использованием класса LearningPipeline из фреймворка ML.NET.

```
var learningPipeline = new LearningPipeline();
```

Содержимое списка необходимо конвертировать в формат CollectionDataSource для последующей интеграции в структуру LearningPipeline.

```
var trainCollection = CollectionDataSource.From(trainData); learningPipe.Append(trainCollection);
```

Конвертация меток в числовые индексы.

```
learningPipe.Add(new DictionaryLabeler("Label"));
```

Определение столбцов функций.

learningPipe.Add(new ColumnConcatenator("Features", new[] { "UnitA", "UnitS", "Volume" }));

Реализация метода классификации.

learningPipe.Add(new SDCARegressionTrainer());

// Интеграция конвертера прогнозируемых меток PredictedLabelColumnOriginalValueConverter в learningPipe { PredictedLabelColumn = "ForecastOutcome" });

Обучение модели. Интегрируем ClassificationEvaluator() для оценки эффективности модели, затем инициируем выполнение программы для ее валидации. С использованием этого инструмента анализа мы способны извлечь данные по матрице ошибок и показателям потерь.

var evaluator = new ClassificationEvaluator(); metrics = evaluator.CalculatePerformance(model, trainingDataset);

Console.WriteLine("Micro-accuracy: " + metrics.MicroAccuracy); Console.WriteLine("Displaying Log Loss metric: " + metrics.LogLoss);

При использовании функции автоматического внедрения процедуры нормализации MinMaxScaler применяйте параметры «norm=Warn» или «norm=No» для деактивации данного механизма.

Сборка тестовых и предсказанных данных производится перед их выводом на печать.

var outcomes = testData.Zip(predicted, (t, p) => new AnalyzeData {

UnitA = t.UnitA,

UnitS = t.UnitS,

Volume = t.Volume,

Label = p.PredictedLabels

}).ToList() ;

Print(results);

Реализации регрессионного анализа на языке C#

```
using Microsoft.ML;
using Microsoft.ML.Data;
using Microsoft.ML.Models;
using Microsoft.ML.Legacy;
using Microsoft.ML.Trainers;
using Microsoft.ML.DataTransforms;
using System;
utilizing System.Collections.Generic;
using System.Linq;
namespace mltest
{
public class ProcessData
{
[Column("0")]
public float UnitA;
[Column("1")]
public float UnitS;
[Column("2")]
public float Volume;
[Column("3")]
[ColumnName("Label")]
public string Label;
}
public class ForecastProcess
{
[AttributeName("ForecastedOutcome")]
string PredictedLabels;
}
```

```
class Program
{
static void Main(string[] args)
{
// Создание обучающих и оценочных наборов данных var trainingDataset = CreatePredictiveData(2000);
// Отображаем первые 20 записей из набора trainData
var testData = ProducePseudoData(50, testMode:true);
// Создание объекта для проекта машинного обучения
var learningPipe = new LearningPipeline();
var trainCollection =
CollectionDataSource.From(trainData);
learningPipe.Include(trainCollection);
// Преобразование текстовых меток в числовые индексы learningPipe.Add(new Dictionarizer("Target"));
learningPipe.Add(
new ColumnConcatenator("Features", "UnitA", "UnitS", "Volume"));
// Процесс категоризации
learningPipe.Add(new SDCARegressionTrainer());
// Преобразование прогнозируемого значения столбца
// меток
learningPipe.Add(new
PredictedLabelColumnOriginalValueConverter()
{ PredictedLabelColumn = "PredictedLabel" });
// Модель обучения
var predictionModel = learningPipeline.Fit<ProcessData, ProcessPrediction>();
// Оценка эффективности модели и проверка точности
ClassificationEvaluator evaluator = new
ClassificationEvaluator();
```

```
var measurements = evaluator.Assess(model, trainingSet);
Console.WriteLine("AccuracyMicro: " + metrics.AccuracyMicro);
Console.WriteLine("Logarithmic Loss: " + metrics.LogLoss);
//      Прогноз результатов на основе оценки данных
var prediction = model.Forecast(testData);
//      Получаем тестовые данные и классы прогнозов
var outcomes = testData.Zip(predicted, (test, pred)
=> new ProcessData
{
UnitA is      assigned the value of t.UnitA,
UnitSize      = time.UnitSize,
Volume = transaction.Volume,
Variable      = p.PredictedTags
}).ToList();
// Отображение результатов выполнения DisplayOutput(results);
// Console.ReadLine() метод в C# используется для чтения // следующей строки символов из стандартного ввода
}
private static void Display(IEnumerable<ProcessData> outcomes)
{
int i =1;
Console.WriteLine(string.Join(",", new string[] {"id", "UnitA"}));
"MeasurementUnits", "Capacity", "Identifier"));
foreach (var item in results)
{
Console.WriteLine($"{i},{item.UnitA},{item.UnitS},");          item.Capacity, item.Identifier));
}       ++;
}
```

```
private static List<ProcessData>
CreateProcessDataList(int quantity, bool isTestMode = false) {
List<ProcessData> pdList        = new();
Random random =new Random();
for (int index =      0; index < total; index++)
{
var pd = new ProcessData
{
UnitA = random.NextInt32(0,  120),
UnitS = random.Next(minValue: 0, maxValue: 60), Volume = random.rand-
int(40,         280)
};
if (! test)
{
If either pd.UnitA exceeds 75 or pd.UnitS is above 30, and concurrently, pd.Vol-
ume surpasses 230 pd.Label = "Caution";
otherwise, if (either pd.UnitA is below 25 or pd.UnitS falls short of 20) and
pd.Volume is less than 100 pd.Label = "Warning";
else pd.Label = "Typical";
}
else
pd.Label =   "";
pdlist.Insert(pd);
}
return pdlist;
}
}
}
```

Особенности ML.NET. Microsoft ML.NET обеспечивает широкие возможности для машинного обучения, включая классификацию текста, прогнозирование числовых данных, обнаружение аномалий, рекомендательные системы и обработку изображений.

Используя DotNet, можно разработать программное обеспечение для машинного обучения благодаря фреймворку ML.NET.

Для программирования с ML.NET подходят C# и F#.

ML.NET – это кросс-платформенная, открытая разработка с исходным кодом.

ML.NET поддерживает разработку и выполнение на платформах Windows, Linux и macOS.

Активно применяется в Microsoft Windows, Bing, Azure, и поддерживает внедрение в сторонние среды, например, TensorFlow, CNTK и Accord.NET.

ML.NET обеспечивает создание приложений на основе машинного обучения для веб-разработок, мобильных платформ, настольных систем, игровых проектов и Интернета вещей.

ML.NET сохраняет тренированную модель в формате бинарного файла, который затем может быть интегрирован в различные DotNet-приложения.

Компания Microsoft неустанно обогащает свои продукты рядом новшеств, например интеграцию продвинутых методов глубокого обучения с использованием TensorFlow и Cognitive Toolkit (CNTK).

В раннем выпуске ML.NET 0.2 был внедрен набор функ- циональностей для выполнения задач кластеризации в рамках машинного обучения. В предварительной версии ML.NET 0.5 был включен механизм оценки моделей TensorFlow. В предварительной версии ML.NET 0.6 расширены функции, позволяя оценивать уже обученные ONNX модели.

С выпуском версии ML.NET 0.7 платформа расширила свою поддержку, включив в себя архитектуры как x86, так и x64. ML.NET, находясь в стадии предварительного выпуска, постоянно обновляется Microsoft, при этом в платформу

интегрируются новые функциональные возможности. В то время как более ранние версии ML.NET были сосредоточены исключительно на поддержке x64, начиная с текущей версии, разработчики могут вести разработку под обе архитектуры – как x86, так и x64.

Первая тестовая версия ML.NET 0.7 внедряет на стадии испытаний интеграционные Python-библиотеки NimbusML для ML.NET.

В предварительной версии ML.NET 0.7 представлены функционалы для выявления аномалий.

ML.NET строится вокруг модели машинного обучения, задающей последовательность действий для генерации прогнозов на базе вводимой информации. Эта платформа дает возможность разработать индивидуальные модели с выбором нужного алгоритма и поддерживает загрузку готовых моделей TensorFlow и ONNX.

Разработанную модель можно интегрировать в программное обеспечение и применять ее для прогнозирования исходов.

ML.NET функционирует на операционных системах Linux, Windows и macOS при использовании .NET Core, а на Windows также совместим с .NET Framework. Версия для 64-битных систем доступна на всех перечисленных платформах, в то время как 32-битная версия ограничена работой на Windows и не поддерживает определенные возможности, связанные с LightGBM, TensorFlow и ONNX.

ML.NET поддерживает генерацию прогнозов разнообразных категорий.

Классификация/категоризация – автоматизированная классификация отзывов от клиентов на позитивные и негативные.

Прогнозирование регрессии для предсказания непрерывных переменных – например, оценка цены недвижимости с учетом ее площади и местоположения.

Обнаружение аномалий – к примеру, выявление мошеннических операций по банковским счетам.

Рекомендации – например, рекомендация товаров, соответствующих интересам клиентов онлайн-магазина, исходя из истории их предшествующих заказов.

Серии данных по времени/последовательности данных – к примеру, метеорологические прогнозы и прогнозирование объемов продаж.

Классификация изображений – например, диагностика аномалий на изображениях, полученных с медицинского оборудования.

Классификация текста – например, проведите категоризацию текстов на основании их контента.

Сходство предложений – можно оценить степень схожести двух фраз.

Пример

Обсудим участок кода из базового приложения на ML.NET. В демонстрации создается модель линейной регрессии, предназначенная для предсказания стоимости недвижимости на основе ее площади и первоначальной цены.

```
Using System;
using Microsoft.ML;
using Microsoft.ML.Data;
class Program
{
class HouseData
{
public float Dimensions { get; set; }
public float Price { get; set; }
}
public interface Prediction
{
[AttributeName("Score")]
public float Price { get; set; }
}
```

```
Define Main(string[] args) as Empty
{
MLContext mlContext = new MLContext();
//      1. Получение или разработка наборов данных
//      для обучения
HouseData houseData[] =       {
()      { Dimension = 1.1F, Cost = 1.2F       },
new   HouseData()       {       Area =       1.9F, Cost    = 2.3F       },
new   HouseData()       {       Dimension =       2.8F,  Cost =       3.0F   },
new HouseData() { Size = 3.4F, Price = 3.7F       }       };
IDataView trainingSet = Context.Data.LoadFromEnumerable(PropertyData);
//      2. Описание схемы предварительной       обработки данных
//      и разработки модели
Pipelinee = Context.Data.Preprocessing.Combine("Features", new[]      { "Size" })
.Attach(mlContext.Regression.Trainers.Sdca(labelColumnName:
"Price", maximumIterations: 100));
//      3.      Обучение модели
var model = pipeline.Train(trainingData);
//      4.      Прогнозированиерезультатов
var size = new HouseData() { Size = 2.5f     };
// Переменной "var price" присваивается результат
// использования метода Predict, который создан для
//      прогнозирования моделей данных  о домах, созданных
// на основе "Context.Model" с переданной конкретной моделью.
//      Этот механизм, предназначенный   для прогнозирования
// значений на основе данных о доме, использует заданный
// параметр "Size" для своего прогнозирования
Console.WriteLine($"Estimated market value for a property with
{size.Size*1000} square feet area: {price.Price*100:C} thousand dollars");
```

The estimated market value for a property spanning 2500 square feet is approximately $261,980.

}

}

Процесс использования программного обеспечения. На рис. 1 демонстрируется архитектура программного обеспечения и цикличный процесс создания модели:

– коллекция и интеграция тренировочных данных в структуру IDataView;

– определение последовательности операций по извлечению признаков и использование алгоритма машинного обучения;

– тренировка модели через вызов метода Fit() в рамках шагов, которые выполняются, чтобы создать и запустить программу или приложение;

– установление эквивалентности достижений заданным нормам и стандартам модели и циклы оптимизации для повышения ее качества;

– сериализация модели в бинарный формат для интеграции в приложение;

– инициализация ITransformer из сохраненной модели;

– прогнозирование данных через метод CreatePrediction- Engine.Predict().

Модель ML.NET. Эта модель – сущность, включающая набор трансформаций для выработки предсказаний, основываясь на анализе данных, представленных пользователем.

Basic. Наиболее элементарная форма – это бивариантная линейная регрессия, представляющая собой взаимосвязь, где одна непрерывная переменная изменяется прямо пропорционально другой.

Более сложная модель. Усовершенствованная модель категоризации транзакций применяет анализ текстовых данных для классификации операций.

Для анализа транзакций их описания декомпозируют на составляющие элементы, исключая нерелевантный текст и символы, а затем производится количественный подсчет слов и символов. Эти составляющие элементы служат базой для тренировки линейной регрессионной модели, основанной на множестве категорий из тренировочного датасета. Схожесть новых описаний с уже существующими в тренировочном наборе данных повышает вероятность корректного категоризирования аналогичных транзакций.

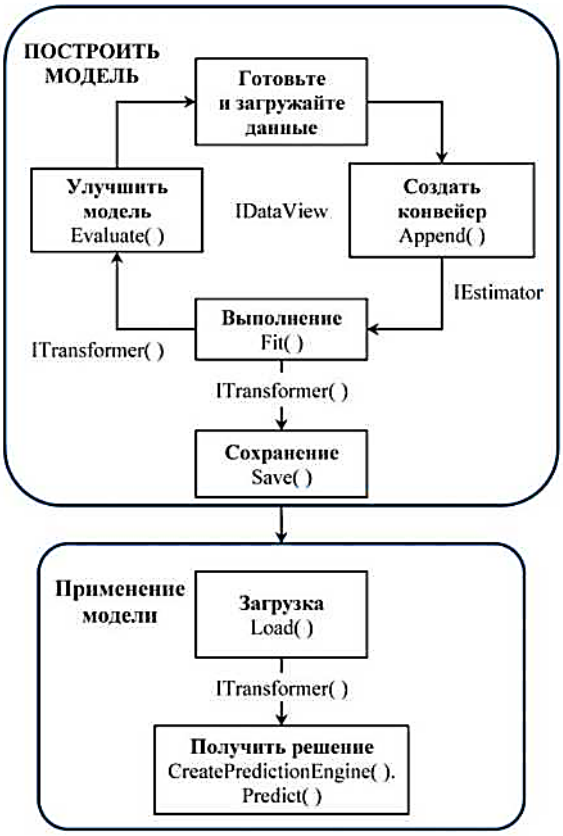

Рис. 1. Цикличный процесс создания модели

В примере, где приведены модели прогнозирования стоимости жилья и классификации текстов основаны на линейных алгоритмах. Однако варьирование типов данных и поставленных целей требует рассмотрения других подходов, например, использования деревьев решений, обобщенных аддитивных моделей и прочих методик.

Подготовка данных. В большинстве ситуаций данные не сразу подходят для тренировки алгоритмов машинного обучения в своем первоначальном формате. Обработка данных представляет собой ключевой этап, предшествующий их использованию для оптимизации модели. Часто необходимо выполнить ряд преобразований, включая конвертацию из текстовых форматов в числовые, удаление лишней информации для очистки данных, регулировку объема данных, их масштабирование или нормализацию для улучшения качества и эффективности обучения модели.

Архитектура ML.NET. Разработка на ML.NET начинается с создания экземпляра MLContext. Это фабрика по созданию компонентов для загрузки данных, преобразования данных, обучения моделей, прогнозирования и операций сохранения/загрузки моделей.

Инициализация платформы ML.NET и архивирование данных

Пространство имен: Microsoft.ML.

Сборка: Microsoft.ML.Data.dll.

Пакет: Microsoft.ML v4.0.1.

Класс, используемый для создания компонентов, которые работают с данными, но не являются частью тренировочных этапов создания и запуска модели. Включает компоненты для загрузки, сохранения, кеширования, сортировки, перемешивания и разбиения данных.

C#

public sealed class DataOperationsCatalog

Наследование: Object ^ DataOperationsCatalog.

Подготовка данных

Пространство имен: Microsoft.ML.

Сборка: Microsoft.ML.Data.dll.

Пакет: Microsoft.ML v3.0.1.

Класс, используемый в MLContext для создания экземпляров компонентов преобразования.

C#

public sealed class TransformsCatalog

Наследование: Object ^ TransformsCatalog.

Алгоритмы обучения

Двоичная классификация

Пространство имен: Microsoft.ML.

Сборка: Microsoft.ML.Data.dll.

Пакет: Microsoft.ML v3.0.1.

Класс, используемый в MLContext для создания экземпляров компонентов двоичной классификации, таких как средства обучения и калибраторы.

C#

public sealed class BinaryClassificationCatalog :

Microsoft.ML.TrainCatalogBase

Наследование: Object ^ TrainCatalogBase ^ BinaryClassificationCatalog.

Многоклассовая классификация

Пространство имен: Microsoft.ML.

Сборка: Microsoft.ML.Data.dll.

Пакет: Microsoft.ML v3.0.1.

Класс, используемый в MLContext для создания экземпляров компонентов многоклассовой классификации, таких как обучающие объекты.

C#

public sealed class MulticlassClassificationCatalog :

Microsoft.ML.TrainCatalogBase

Наследование: Object ^ TrainCatalogBase ^ MulticlassClassificationCatalog.

Обнаружение аномалий

Пространство имен: Microsoft.ML.

Сборка: Microsoft.ML.Data.dll.

Пакет: Microsoft.ML v3.0.1.

Класс, применяемый для создания экземпляров компонентов в среде MLContext, включая инструменты для обучения и анализа в задачах выявления аномалий.

C#

public sealed class AnomalyDetectionCatalog :

Microsoft.ML.TrainCatalogBase

Наследование: Object ^ TrainCatalogBase ^ AnomalyDetectionCatalog.

Прогнозирование

Пространство имен: Microsoft.ML.

Сборка: Microsoft.ML.Data.dll.

Пакет: Microsoft.ML v3.0.1.

Класс, используемый MLContext для создания экземпляров компонентов прогнозирования.

C#

public sealed class ForecastingCatalog :

Microsoft.ML.TrainCatalogBase

Наследование: Object ^ TrainCatalogBase ^ ForecastingCatalog.

Ранжирование

Пространство имен: Microsoft.ML.

Сборка: Microsoft.ML.Data.dll.

Пакет: Microsoft.ML v3.0.1.

Класс, применяемый в MLContext для создания элементов ранжирования, включая обучающие модели и модули оценки.

C#

public sealed class RankingCatalog :

Microsoft.ML.TrainCatalogBase

Наследование: Object ^ TrainCatalogBase ^ RankingCatalog.

Регрессия

Пространство имен: Microsoft.ML.

Сборка: Microsoft.ML.Data.dll.

Пакет: Microsoft.ML v3.0.1.

Класс, применяемый в MLContext для создания элементов регрессии, включая обучающие алгоритмы и анализаторы результатов.

C#

public sealed class RegressionCatalog :

Microsoft.ML.TrainCatalogBase

Наследование: Object ^ TrainCatalogBase ^ RegressionCatalog.

Использование модели

Пространство имен: Microsoft.ML.

Сборка: Microsoft.ML.Data.dll.

Пакет: Microsoft.ML v3.0.1.

Класс, используемый в MLContext для сохранения и загрузки обученных моделей.

C#

public sealed class ModelOperationsCatalog

Наследование: Object ^ Model Operations Catalog.

Через любую из вышеупомянутых категорий доступен доступ к соответствующим способам разработки. В Visual Studio для отображения каталогов применяется функционал IntelliSense.

Сборка конвейера

В любом директории представлен набор функций-расширений. Рассмотрим их применение в процессе построения обучающего конвейера.

var pipeline = mlContext.Data.Processors.Combine("Attributes", new[] { "Dimension" })

.Add(mlContext.Regression.Trainers.SdcaRegression(labelColumn- Name: "Price", maximumNumberOfIterations: 100));

Эта фаза ограничивается только разработкой сущностей, не переходя к их реализации.

Обучение модели

После инициализации объектов в процессе обработки данных их можно применять для тренировки алгоритма.

var trainedModel = pipeline.TrainOn(trainingDataSet);

Использование метода Fit() позволяет применять наши тренировочные данные для настройки параметров модели – процесс, известный как ее обучение. В контексте обсуждаемой линейной регрессии модель определяется двумя ключевыми параметрами: смещением и весом. Выполнение метода Fit() обеспечивает определение конкретных значений этих параметров. Стоит заметить, что для большинства моделей машинного обучения количество параметров значительно превышает два.

Анализируем процесс согласно структуре машинного обучения.

Разбиение данных на тренировочные и тестовые выборки. Задача алгоритма машинного обучения заключается в выявлении структур в тренировочном наборе данных. Эти выявленные структуры применяются для генерации прогнозов, основываясь на данных, которые не встречались ранее.

Произведем разбиение исходного массива данных на две части: обучающую и тестовую, с применением функции TrainTestSplit. Это действие приведет к созданию экземпляра TrainTestData, который будет включать в себя два элемента типа IDataView: первый элемент предназначен для обучающей выборки, второй – для тестовой. Процентное соотношение данных, отводимых под тестовую выборку, устанавливается через аргумент testFraction. В приведенном ниже примере кода для тестового набора данные выделены в объеме 20 % от всего массива данных.

DataOperationsCatalog.TrainTestData dataPartition = mlContext.Data.TrainTestSplit(data, testFraction: 0.2); IDataView trainingDataset = dataSplit.TrainingData; IDataView testData = dataSplit.TestSet;

Подготовка данных. Перед тренировкой модели ИИ следует выполнить предобработку данных. ML.NET-алгоритмы требуют специфичных форматов входящих колонок. Отсутствующие значения автоматически получают стандартные имена для входных и выходных колонок.

Управление предполагаемыми типами данных в столбцах. В ML.NET алгоритмы машинного обучения требуют ввода в форме вектора с дробными числами фиксированной длины. При условии, что все данные уже конвертированы в числа и должны быть обработаны параллельно (например, пиксели изображений), используйте атрибут VectorType для соответствующей структуры данных.

Когда столкнетесь с неоднородными данными, а задача требует применить различные методы трансформации к каждому атрибуту, воспользуйтесь функцией Concatenate после обработки индивидуальных атрибутов. Это позволит агрегировать обработанные данные в единую структуру, формируя новый атрибут из векторов, представляющих собой обработанные значения каждого столбца.

В данном кодовом сегменте колонки «Size» и «Historical- Prices» агрегируются в единый вектор элементов, представляющий собой исходные данные для создания новой колонки, именуемой «Features». В силу однородности масштабов данных метод «NormalizeMinMax» используется к колонке «Features» в целях выполнения нормализации значений.

```
//      1. Метод предварительной оценки данных.
// Объединяем атрибуты размера и истории в единый вектор // признаков, сохраненный в недавно созданном столбце // с именем «Features».
//      2. Стандартизировать набор функций
IEstimator<ITransformer> dataPreparationEstimator = mlContext. Transforms . mlContext.Transforms.CombineColumns ( "Features" , new[ ] {"Size", "HistoricalPrices"}) .Append(mlContext.Transforms.NormalizeMinMax("Features"));
// Инициализация преобразования подготовки данных ITransformer dataPreparationTransformer = dataPreparationEstimator.Fit(trainingDataSet);
// Выполнение преобразования на обучающем наборе данных IDataView processedTrainingData = dataPreparationTransformer.Apply(trainData);
```

Обработка стандартных наименований колонок. При отсутствии явного указания названий колонок система машинного обучения ML.NET автоматически присваивает стандартные имена. Для обработки входных данных каждый экземпляр алгоритма обладает параметром под наименованием featureColumnName. Когда это необходимо, предусмотрен дополнительный параметр labelColumnName для определения целевого значения, к которому должна стремиться модель. По умолчанию эти параметры называются Features и Label соответственно.

В процессе предобработки данных, когда применяется операция Concatenate для формирования объединенного столбца под названием Features, нет необходимости явно указывать имя этого столбца в настройках алгоритма машинного обучения, поскольку он уже включен в подготовленный набор данных в формате IDataView. Что касается столбца с метками, именуемого

CurrentPrice, его имя автоматически модифицируется в Label благодаря использованию атрибута ColumnName в структуре данных. Это действие, совершенное библиотекой ML.NET, избавляет пользователя от необходимости задавать имя столбца с метками (labelColumnName) при настройке оценки производительности выбранной модели машинного обучения.

При отказе от использования предопределенных наименований столбцов необходимо явно указать названия атрибутов и целевых переменных, передавая их как аргументы при конфигурации модели машинного обучения.

Это демонстрируется в приведенном ниже коде:

var UserDefinedColumnSdcaEstimator =

mlContext.Regression.Trainers.Sdca(labelColumnName: "MyLabelColumnName", featureColumnName: "MyFeatureColumnName");

Кеширование данных. По умолчанию в процессе обработки данных их внесение или стриминг происходит в режиме «ленивой» загрузки. Это подразумевает, что алгоритмы машинного обучения в процессе тренировки могут многократно осуществлять чтение данных с накопителя и последующую обработку. С целью уменьшить объем операций чтения данных адекватным решением является применение кеширования данных в оперативной памяти. Кеширование осуществляется в рамках EstimatorChain через механизм AppendCacheCheckpoint.

Для повышения эффективности обработки данных перед применением алгоритмов машинного обучения рекомендуется располагать метод AppendCacheCheckpoint в начале конвейера данных.

Через внедрение метода AppendCacheCheckpoint в класс EstimatorChain перед использованием StochasticDualCoordinateAscent алгоритм обучения оптимизирует процесс, сохраняя вычисления предшествующих оценщиков для дальнейшего применения.

// 1. Объединить атрибуты Size и Historical в единое

// векторное представление и назначить этот результирующий // вектор новому введенному столбцу с названием Features // 2. Стандартизировать вектор Attributes

// 3. Кешировать подготовленные данные

// 4. Реализовать алгоритм Sdca для обучения модели

IEstimator<ITransformer> preparationPipeline = mlContext.Transforms.Concatenate("Features", new[] {"Size", "HistoricalPrices"})

.Add(mlContext.Transforms.NormalizeMinMax("Features"))
.AddCacheCheckpoint(mlContext);

.AddTrainer(mlContext.Regression.Trainers.SdcaRegression());

Тренировка алгоритма искусственного интеллекта. После выполнения этапа подготовки данных следует применить функцию Fit() для тренировки модели машинного обучения, используя алгоритм регрессии Stochastic Dual Coordinate Ascent.

// Инициализация стохастической двухкоординатной

// регрессионной модели оценки var regressionEstimator = mlContext.Regression.Trainers.SdcaNonCalibrated();

Построение алгоритма искусственного интеллекта

var model = sdcaEstimator.Train(transformedTrainingDataSet);

Извлечение параметров модели. После тренировки модели экспортируйте получившиеся ModelParameters для анализа или дальнейшего обучения. Параметры линейной регрессии (LinearRegressionModelParameters) включают смещение.

var trainedModelParameters = trainedModel.Model as LinearRegressionModelParameters;

Использование модели. Данные можно трансформировать в предсказания двумя способами: параллельно или последовательно. В нашем эксперименте мы применили обе методики: параллельную трансформацию для оценки эффективности модели и последовательную для генерации индивидуальных предсказаний. Давайте подробнее рассмотрим процесс формирования отдельных прогнозов.

Функция CreatePredictionEngine() требует указать классы для входных и выходных данных. Соответствие между полями классов и столбцами в датасете устанавливается через имена полей или атрибуты в коде, что критически важно для обучения и последующего предсказания модели.

Модели и схема данных. Основой архитектуры машинного обучения ML.NET являются объекты DataView.

Каждый этап в потоковой обработке данных требует определенной входной структуры (описание требуемых данных по имени, типу и объему) и генерирует результат с заданной выходной структурой (конкретизация результата работы по имени, типу и объему данных).

Когда выходные данные одного этапа преобразования в конвейере обработки данных не совпадают с ожидаемыми входными данными следующего этапа, ML.NET генерирует ошибку в виде исключения.

Структура данных для представления информации организована в таблицу, состоящую из колонок и записей. Каждая колонка имеет уникальное наименование, определенный тип данных и размер. В качестве примера, рассматривая обработанный и структурированный массив данных с информацией о стоимости жилья, можно выделить колонки «Size» и «Price». Эти атрибуты характеризуются конкретными значениями и классифицируются как скалярные данные, в отличие от векторных (рис. 2).

Size скаляр число	**Price** скаляр число	**Features** скаляр вектор (1)	**Score** скаляр число
1.1	1.2	[1.1]	1.29
1.9	2.3	[1.9]	2.14
2.8	3.0	[2.8]	3.10
3.4	3.7	[3-4]	3.75 *J*

Рис. 2. Структура данных для представления информации о стоимости жилья

ML.NET framework обрабатывает данные через векторные столбцы. Стандартное наименование такого столбца – «Features». В демонстрационном проекте по прогнозированию стоимости жилья мы интегрировали столбец «Size» в вектор «Features» для анализа.

Дополнительно после завершения процесса прогнозирования различные алгоритмы машинного обучения порождают дополнительные колонки данных. Названия этих дополнений строго определены и зависят от конкретного применяемого метода в машинном обучении. Например, в контексте выполнения регрессионного анализа одна из таких добавленных колонок именуется, как «Score», что стало причиной для того, чтобы мы именовали наш целевой признак данных, как «Price».

```
public class Prediction
{
[ColumnName("Score")]
public float Price { get; set; }
}
```

Ключевой характеристикой компонентов DataView в библиотеке ML.NET является их простая вычислительная модель. Это означает, что данные, представленные этими объектами, активно загружаются и применяются исключительно в процессах обучения, оценивания моделей и в ситуациях, когда требуется генерация прогнозов. В процессе разработки и тестирования решений, основанных на ML.NET, разработчики имеют возможность воспользоваться возможностями отладчика Visual Studio, чтобы осуществить визуализацию и анализ любого объекта DataView при помощи специализированного метода для предварительного просмотра данных.

```
var debugLog = testPriceDataView.GeneratePreview();
```

В отладчике возможно провести анализ значения переменной debug, оценив ее данные. Отказ от применения функции пред- просмотра в коде рекомендуется из-за ее негативного влияния на эффективность выполнения программы.

Развертывание модели. В практических реализациях задач машинного обучения процессы тренировки и оценки эффективности модели обычно разграничиваются с этапом прогнозирования. Такой подход обусловлен тем, что выполнение данных операций часто находится в ведении различных команд специалистов.

```
mlContext.Model.Save(model, trainingData.Schema, "model.zip");
```

Авторы продемонстрировали, что широкое применение моделей на основе линейных корреляций опирается на легкость их понимания и проверки; подчеркивается, что точность прогнозов не всегда удовлетворяет потребности пользователя.

ГЛАВА 4. ML.NET И РЕГРЕССИОННЫЙ АНАЛИЗ ДЛЯ КАРТИРОВАНИЯ УГЛЕРОДА, ДЕПОНИРУЕМОГО ЛЕСАМИ

Для понимания данного раздела необходимы знания языка программирования C#.

ML.NET являясь открытой, кроссплатформенной средой машинного обучения предоставляет разработчикам возможность создавать, обучать и внедрять модели машинного обучения без необходимости углубленных знаний в области науки о данных.

Ключевые особенности ML.NET

1. **Интеграция с экосистемой .NET** – платформа полностью совместима с языками C#, F# и другими .NET-языками, что позволяет легко внедрять модели машинного обучения в существующие приложения.

2. **Широкий спектр алгоритмов** – ML.NET поддерживает различные типы задач машинного обучения, включая классификацию, регрессию, кластеризацию, обнаружение аномалий, рекомендательные системы и другие.

3. **Производительность** – благодаря оптимизированным низкоуровневым библиотекам, ML.NET обеспечивает высокую скорость обработки данных и обучения моделей.

4. **Пайплайны данных** – платформа предоставляет гибкие средства для создания цепочек преобразования данных, включая загрузку, очистку, нормализацию и трансформацию признаков.

5. **Автоматическое машинное обучение (AutoML)** – функциональность, позволяющая автоматически подбирать оптимальные алгоритмы и гиперпараметры для конкретной задачи.

6. **Модульная архитектура** – возможность комбинировать различные компоненты и алгоритмы для создания комплексных решений.

7. **Интероперабельность** – поддержка импорта моделей из других фреймворков (TensorFlow, ONNX) и экспорта обученных моделей в различные форматы.

Для экологических исследований, в частности для задач картирования углерода, ML.NET представляет ценность благодаря наличию эффективных регрессионных алгоритмов, возможности работы с большими наборами данных и интеграции с геопространственными библиотеками .NET.

Платформа ML.NET предлагает широкий спектр регрессионных алгоритмов, которые могут быть применены для прогнозирования запасов углерода в лесных экосистемах.

Основные алгоритмы регрессии, доступные в ML.NET

1. **Линейная регрессия (FastLinearRegressor)** – классический алгоритм, моделирующий линейную зависимость между признаками и целевой переменной. Подходит для задач с явными линейными взаимосвязями и ограниченным количеством признаков.

2. **Градиентный бустинг (LightGbmRegressor, FastTreeRegressor)** – ансамблевые методы, основанные на последовательном построении деревьев решений. Эти алгоритмы хорошо справляются с нелинейными зависимостями и устойчивы к выбросам в данных.

3. **Стохастический градиентный спуск (SdcaRegressor)** – эффективный метод для обучения линейных моделей на больших наборах данных с применением регуляризации.

4. **Решающие леса (FastForestRegressor)** – ансамблевый алгоритм, основанный на построении множества независимых деревьев решений. Обладает высокой точностью и устойчивостью к переобучению.

5. **Нейронные сети (OnlineGradientDescentRegressor)** – реализация нейросетевого подхода для задач регрессии, способная моделировать сложные нелинейные зависимости.

6. **Факторизационные машины (FieldAware Factorization Machine Regressor)** – алгоритм, эффективно учитывающий взаимодействия между признаками, что особенно важно при наличии категориальных переменных.

Каждый из этих алгоритмов имеет свои преимущества и ограничения применительно к задаче картирования углерода

Линейные модели просты в интерпретации, но могут не улавливать сложные экологические взаимосвязи.

Древесные ансамбли (градиентный бустинг, решающие леса) обычно демонстрируют высокую точность при моделировании экологических процессов и устойчивы к шуму в данных.

Нейронные сети способны выявлять сложные паттерны, но требуют большого объема данных для обучения и склонны к переобучению.

ML.NET предоставляет единый интерфейс для работы со всеми этими алгоритмами, что позволяет легко сравнивать их эффективность и выбирать оптимальное решение для конкретной задачи. Кроме того, платформа поддерживает кросс-валидацию и другие методы оценки моделей, необходимые для надежного сравнения алгоритмов.

Применение ML.NET для решения задач экологического моделирования, в частности для картирования углерода в лесах, обладает рядом существенных преимуществ.

1. **Интеграция с геоинформационными системами** – благодаря совместимости с библиотеками .NET для работы с геопространственными данными (например, NetTopologySuite,GDAL.NET), ML.NET позволяет легко создавать интегрированные решения для анализа и визуализации пространственно-распределенных данных.

2. **Масштабируемость** – платформа способна эффективно обрабатывать большие объемы данных, что критически важно при работе с высокоразрешающими спутниковыми снимками и цифровыми моделями рельефа, охватывающими обширные лесные территории.

3. **Гибкость в предварительной обработке данных** – ML.NET предоставляет богатый инструментарий для трансформации и подготовки данных, включая обработку пропущенных значений, нормализацию, кодирование категориальных признаков и генерацию новых признаков.

4. **Возможность создания комплексных приложений** – интеграция с экосистемой .NET позволяет разрабатывать полноценные приложения с графическим интерфейсом, веб-интерфейсом или в виде микросервисов для мониторинга лесных экосистем.

5. **Воспроизводимость результатов** – ML.NET обеспечивает возможность сохранения и загрузки как обученных моделей, так и полных пайплайнов обработки данных, что гарантирует воспроизводимость результатов и упрощает развертывание моделей в производственной среде.

6. **Автоматизация процессов машинного обучения** – функциональность AutoML позволяет автоматизировать подбор оптимальных алгоритмов и гиперпараметров, что особенно ценно при работе с экологическими данными, характеризующимися сложными нелинейными взаимосвязями.

7. **Снижение барьера входа** – использование знакомого языка программирования (C#) и привычной среды разработки делает технологии машинного обучения доступными для специалистов в области лесного хозяйства и экологии без глубоких знаний в области науки о данных.

Для задачи картирования углерода в лесах особенно ценными являются возможности ML.NET по обработке разнородных данных (спутниковые снимки, полевые измерения, климатические параметры) и созданию моделей, учитывающих пространственные взаимосвязи. Платформа позволяет эффективно комбинировать данные из различных источников и создавать комплексные предиктивные модели, отражающие сложную природу процессов накопления углерода в лесных экосистемах.

Разрабатываемая информационная система для определения и картирования углерода, депонируемого лесами, с использованием ML.NET имеет модульную архитектуру, состоящую из следующих компонентов.

1. **Модуль сбора и импорта данных**:

- интерфейсы для импорта полевых измерений (запасы углерода на пробных площадях);

- компоненты для загрузки и предварительной обработки спутниковых снимков;
- инструменты для импорта геопространственных данных (цифровые модели рельефа, почвенные карты, климатические данные);
- система хранения метаданных и управления версиями наборов данных.

2. **Модуль предварительной обработки данных**:

- компоненты для пространственной привязки и совмещения разнородных данных;
- инструменты для вычисления спектральных индексов и текстурных характеристик;
- процедуры нормализации и стандартизации признаков;
- компоненты для обработки пропущенных значений и выбросов;
- генераторы производных признаков на основе экологических закономерностей.

3. **Модуль машинного обучения на базе ML.NET** :

- интерфейс для выбора и настройки регрессионных алгоритмов;
- компоненты для разделения данных на обучающую и тестовую выборки;
- инструменты для кросс-валидации и оценки качества моделей;
- процедуры оптимизации гиперпараметров моделей;
- механизмы для сохранения и загрузки обученных моделей.

4. **Модуль пространственного прогнозирования**:

- компоненты для применения обученных моделей к пространственно-распределенным данным;
- инструменты для обработки крупных растровых наборов данных по фрагментам;
- механизмы агрегации результатов прогнозирования на различных пространственных уровнях;
- процедуры учета пространственной автокорреляции.

5. **Модуль визуализации и экспорта результатов**:

- компоненты для создания тематических карт запасов углерода;
- инструменты для визуализации неопределенности прогнозов;
- интерфейсы для экспорта результатов в стандартные геоинформационные форматы;
- генераторы статистических отчетов и графиков.

6. **Модуль управления и мониторинга**:

- компоненты для планирования и выполнения процессов обработки данных;
- инструменты для мониторинга производительности системы;
- механизмы логирования и отслеживания ошибок;
- интерфейсы для управления пользователями и правами доступа.

Архитектура системы предусматривает возможность как пакетной обработки данных, так и интерактивного режима работы. Взаимодействие между модулями осуществляется через стандартизованные интерфейсы, что обеспечивает гибкость системы и возможность замены отдельных компонентов без изменения общей архитектуры.

Схема потока данных в системе включает следующие основные этапы.

1. Импорт и предварительная обработка исходных данных.
2. Формирование обучающего набора данных на основе полевых измерений.
3. Обучение и валидация регрессионных моделей.
4. Применение моделей для пространственного прогнозирования.
5. Визуализация и экспорт результатов картирования.

Для эффективного применения ML.NET в задаче картирования углерода критически важным является этап сбора и предварительной обработки данных.

Система работает с несколькими типами входных данных.

1. Полевые измерения запасов углерода:

- данные лесной инвентаризации с геопривязкой (координаты пробных площадей);
- измерения запасов углерода в различных пулах (надземная и подземная биомасса, мертвая древесина, подстилка, почва);
- таксационные характеристики насаждений (возраст, высота, диаметр, сомкнутость, видовой состав).

2. Данные дистанционного зондирования:

- мультиспектральные спутниковые снимки (Landsat, Sentinel-2);
- радарные данные (Sentinel-1);
- данные лидарной съемки (при наличии).

3. Геопространственные данные:

- цифровые модели рельефа и производные характеристики (уклон, экспозиция, кривизна);
- почвенные карты и характеристики почв;
- климатические данные (температура, осадки, солнечная радиация);
- границы лесных массивов и информация о лесоустройстве.

Предварительная обработка данных включает следующие этапы:

1. Пространственная привязка и интеграция:

- приведение всех данных к единой системе координат;
- перепроецирование растровых данных при необходимости;
- совмещение полевых измерений с пространственными слоями.

2. Обработка спутниковых снимков:

- атмосферная коррекция;
- маскирование облаков и теней;
- расчет спектральных индексов (NDVI, EVI, NBR, NDWI и др.);
- вычисление текстурных характеристик (GLCM);
- временное агрегирование (средние, минимальные, максимальные значения индексов).

3. Извлечение и трансформация признаков:

- извлечение значений растровых слоев в точках полевых измерений;
- расчет буферных статистик (средние, медианные значения в окрестности точек);
- генерация производных признаков на основе экологических моделей;
- кодирование категориальных переменных (тип леса, почвенные классы).

4. Анализ и обработка данных:

- обнаружение и обработка выбросов;
- заполнение пропущенных значений;
- нормализация или стандартизация числовых признаков;
- оценка мультиколлинеарности и отбор значимых признаков.

5. Формирование обучающих наборов данных:

- разделение данных на обучающую и тестовую выборки;
- стратификация выборок по типам леса или экологическим зонам;
- подготовка данных в формате, совместимом с ML.NET .

Для реализации предварительной обработки в ML.NET используется система трансформаций данных, которая позволяет создавать пайплайны преобразований, применяемые как на этапе обучения, так и при прогнозировании. Это обеспечивает согласованность обработки данных и минимизирует возможность ошибок.

Выбор оптимальной регрессионной модели для картирования углерода основывается на анализе характеристик данных и особенностей моделируемых экологических процессов. Авторы рассматривают и сравнивают следующие регрессионные алгоритмы, доступные в ML.NET .

1. Линейная регрессия (FastLinearRegressor)

C#

```
var pipeline = mlContext.Transforms.Concatenate("Features", featureColumns)
 .Append(mlContext.Transforms.NormalizeMinMax("Features"))
 .Append(mlContext.Regression.Trainers.FastLinear(
 labelColumnName: "CarbonStock",
 featureColumnName: "Features",
 l2Regularization: 0.01));
```

Данный алгоритм применяется как базовый для установления линейных зависимостей между признаками и запасами углерода. Важным аспектом является выбор оптимальных параметров регуляризации для предотвращения переобучения.

2. Градиентный бустинг (LightGbmRegressor)

C#

```
var pipeline = mlContext.Transforms.Concatenate("Features", featureColumns)
 .Append(mlContext.Regression.Trainers.LightGbm(
 labelColumnName: "CarbonStock",
 featureColumnName: "Features",
 numberOfLeaves: 31,
 numberOfIterations: 100,
 minimumExampleCountPerLeaf: 20,
 learningRate: 0.1));
```

LightGBM представляет собой эффективную реализацию градиентного бустинга, способную работать с большими наборами данных и учитывать нелинейные взаимосвязи между признаками. Этот алгоритм особенно перспективен для экологического моделирования из-за его способности улавливать сложные взаимодействия факторов.

3. Решающие леса (FastForestRegressor)

C#

```
var pipeline = mlContext.Transforms.Concatenate("Features", featureColumns)
 .Append(mlContext.Regression.Trainers.FastForest(
 labelColumnName: "CarbonStock",
 featureColumnName: "Features",
 numberOfTrees: 100,
 numberOfLeaves: 20,
 minimumExampleCountPerLeaf: 10));
```

Метод решающих лесов эффективен для экологических данных благодаря устойчивости к шуму и выбросам, а также способности работать с разнородными признаками без предварительной нормализации.

4. Факторизационные машины (FieldAware Factorization Machine Regressor)

```
C#
var pipeline = mlContext.Transforms.Concatenate("Features", featureColumns)
 .Append(mlContext.Regression.Trainers.FieldAwareFactorizationMachine(
 labelColumnName: "CarbonStock",
 featureColumnName: "Features",
 learningRate: 0.01,
 numberOfIterations: 100));
```

Данный алгоритм эффективно моделирует взаимодействия между признаками, что важно при наличии категориальных переменных и комплексных взаимосвязей в экологических данных.

Для каждого алгоритма реализуются этапы процесса.

1. **Подготовка данных и определение признаков:**

```
C#
// Определение схемы данных
var dataSchema = mlContext.Data.LoadFromTextFile<CarbonDataPoint>(
 path: dataPath,
 hasHeader: true,
 separatorChar: ',');
// Определение признаков для модели
string[] featureColumns = new[] {
 "NDVI", "EVI", "Height", "Slope", "Aspect", "SoilType",
 "MeanTemperature", "Precipitation", "ForestType", "ForestAge"
};
```

2. **Разделение данных на обучающую и тестовую выборки:**

```
C#
var dataSplit = mlContext.Data.TrainTestSplit(
 data: dataSchema,
 testFraction: 0.2,
 seed: 42);
var trainData = dataSplit.TrainSet;
var testData = dataSplit.TestSet;
```

3. **Определение пайплайна обработки данных и обучения:**

```
C#
// Пример для LightGBM
var pipeline = mlContext.Transforms.Concatenate("Features", featureColumns)
 .Append(mlContext.Transforms.CopyColumns("Label", "CarbonStock"))
 .Append(mlContext.Regression.Trainers.LightGbm(
 labelColumnName: "Label",
 featureColumnName: "Features"));
```

4. **Обучение модели и оценка качества:**

```
C#
var model = pipeline.Fit(trainData);
var predictions = model.Transform(testData);
var metrics = mlContext.Regression.Evaluate(predictions);
Console.WriteLine($"R²: {metrics.RSquared}");
Console.WriteLine($"RMSE: {metrics.RootMeanSquaredError}");
Console.WriteLine($"MAE: {metrics.MeanAbsoluteError}");
```

5. **Кросс-валидация для надежной оценки:**

```
C#
var cvResults = mlContext.Regression.CrossValidate(
 data: dataSchema,
 estimator: pipeline,
```

```
 numberOfFolds: 5,
 labelColumnName: "CarbonStock");
var averageR2 = cvResults.Select(r => r.Metrics.RSquared).Average();
var averageRMSE = cvResults.Select(r => r.Metrics.RootMeanSquaredError).Average();
```

6. **Сохранение модели для последующего использования:**

```
C#
mlContext.Model.Save(model, trainData.Schema, modelPath);
```

Для поиска оптимальных гиперпараметров используется функциональность автоматического машинного обучения:

```
C#
var experimentSettings = new RegressionExperimentSettings {
 MaxExperimentTimeInSeconds = 3600,
 OptimizingMetric = RegressionMetric.RSquared
};
var experiment = mlContext.Auto().CreateRegressionExperiment(experimentSettings);
var experimentResult = experiment.Execute(
 trainData: dataSchema,
 labelColumnName: "CarbonStock",
 progressHandler: new RegressionProgressHandler());
var bestModel = experimentResult.BestRun.Model;
```

Выбор окончательной модели для картирования углерода осуществляется на основе комплексной оценки нескольких метрик производительности (R^2, RMSE, MAE) и анализа остатков. Предпочтение отдается моделям, демонстрирующим не только высокую точность, но и устойчивость результатов при кросс-валидации, а также способность к обобщению на новые данные.

Программное решение для картирования углерода с использованием ML.NET реализовано в виде многоуровневой архитектуры, обеспечивающей модульность, масштабируемость и расширяемость системы. Структура решения включает следующие компоненты.

1. Уровень доступа к данным (Data Access Layer)

C#

```
namespace CarbonMapping.DataAccess
{
 // Интерфейс для работы с источниками данных
 public interface IDataSource<T>
 {
 Task<IEnumerable<T>> GetDataAsync();
 Task SaveDataAsync(IEnumerable<T> data);
 }

 // Реализация для CSV файлов с полевыми измерениями
 public class CsvFieldDataSource : IDataSource<CarbonSamplePoint>
 {
 private readonly string _filePath;

 public CsvFieldDataSource(string filePath)
 {
 _filePath = filePath;
 }

 public async Task<IEnumerable<CarbonSamplePoint>> GetDataAsync()
 {
 // Реализация чтения данных из CSV
 }

 public async Task SaveDataAsync(IEnumerable<CarbonSamplePoint> data)
 {
 // Реализация записи данных в CSV
 }
 }
```

```
// Реализация для растровых геоданных
public class GeoTiffDataSource : IDataSource<RasterData>
{
// Реализация для работы с геопространственными растрами
}
}
```

2. Уровень моделей данных (Data Models)

```
C#
namespace CarbonMapping.Models
{
// Модель для точек полевых измерений
public class CarbonSamplePoint
{
public double Longitude { get; set; }
public double Latitude { get; set; }
public double CarbonStock { get; set; }
public double NDVI { get; set; }
public double EVI { get; set; }
public double ForestHeight { get; set; }
public double Slope { get; set; }
public double Aspect { get; set; }
public string ForestType { get; set; }
public int ForestAge { get; set; }
public string SoilType { get; set; }
public double MeanTemperature { get; set; }
public double Precipitation { get; set; }
}

// Модель для результатов прогнозирования
public class CarbonPrediction
```

```
 {
 [ColumnName("Score")]
 public float PredictedCarbonStock { get; set; }
 }

 // Модель для растровых данных
 public class RasterData
 {
 public string Name { get; set; }
 public double[] Values { get; set; }
 public int Width { get; set; }
 public int Height { get; set; }
 public double XOrigin { get; set; }
 public double YOrigin { get; set; }
 public double CellSize { get; set; }
 public string Projection { get; set; }
 }
}
```

3. Уровень обработки данных (Data Processing)

C#

```
namespace CarbonMapping.Processing
{
 // Класс для предварительной обработки данных
 public class DataPreprocessor
 {
 public IEnumerable<CarbonSamplePoint> ExtractFeatures(
 IEnumerable<CarbonSamplePoint> samplePoints,
 IEnumerable<RasterData> rasterLayers)
 {
```

```
// Извлечение значений из растровых слоев в точках отбора проб
// Расчет буферных статистик
// Генерация новых признаков
}

public IEnumerable<CarbonSamplePoint> CleanData(IEnumerable<CarbonSample-
Point> data)
{
// Обработка выбросов
// Заполнение пропущенных значений
// Фильтрация недостоверных данных
}
}

// Класс для пространственного прогнозирования
public class SpatialPredictor
{
private readonly MLContext _mlContext;
private ITransformer _model;

public SpatialPredictor(string modelPath)
{
_mlContext = new MLContext(seed: 42);
// Загрузка обученной модели
DataViewSchema modelSchema;
_model = _mlContext.Model.Load(modelPath, out modelSchema);
}

public RasterData PredictCarbonMap(IEnumerable<RasterData> features)
```

```
 {
 // Применение модели к пространственным данным
 // Генерация растра с прогнозом запасов углерода
 }
 }
}
```

4. Уровень машинного обучения (Machine Learning)

```
C#
namespace CarbonMapping.MachineLearning
{
 // Класс для обучения регрессионных моделей
 public class CarbonModelTrainer
 {
 private readonly MLContext _mlContext;

 public CarbonModelTrainer()
 {
 _mlContext = new MLContext(seed: 42);
 }

 public ITransformer TrainLinearModel(IDataView trainingData)
 {
 // Определение пайплайна для линейной регрессии
 var pipeline = _mlContext.Transforms.Concatenate("Features", featureColumns)
 .Append(_mlContext.Transforms.NormalizeMinMax("Features"))
 .Append(_mlContext.Regression.Trainers.FastLinear(
 labelColumnName: "CarbonStock",
 featureColumnName: "Features"));
```

```
return pipeline.Fit(trainingData);
}

public ITransformer TrainGradientBoostingModel(IDataView trainingData)
{
// Определение пайплайна для градиентного бустинга
var pipeline = _mlContext.Transforms.Concatenate("Features", featureColumns)
.Append(_mlContext.Regression.Trainers.LightGbm(
labelColumnName: "CarbonStock",
featureColumnName: "Features"));

return pipeline.Fit(trainingData);
}

public RegressionMetrics EvaluateModel(ITransformer model, IDataView testData)
{
// Оценка качества модели
var predictions = model.Transform(testData);
return _mlContext.Regression.Evaluate(predictions);
}

public void SaveModel(ITransformer model, DataViewSchema schema, string mod-
elPath)
{
// Сохранение модели
_mlContext.Model.Save(model, schema, modelPath);
}
}
```

```
// Класс для автоматического подбора оптимальной модели
public class AutoMLModelSelector
{
private readonly MLContext _mlContext;

public AutoMLModelSelector()
{
_mlContext = new MLContext(seed: 42);
}

public ExperimentResult<RegressionMetrics> FindBestModel(
IDataView data,
string labelColumnName,
TimeSpan maxTime)
{
// Настройка и запуск эксперимента AutoML
var experimentSettings = new RegressionExperimentSettings
{
MaxExperimentTimeInSeconds = (int)maxTime.TotalSeconds,
OptimizingMetric = RegressionMetric.RSquared
};

var experiment = _mlContext.Auto().CreateRegressionExperiment(experimentSettings);
return experiment.Execute(data, labelColumnName);
}
}
}
```

5. Уровень визуализации и экспорта (Visualization and Export)

```
C#
namespace CarbonMapping.Visualization
{
 // Класс для создания карт запасов углерода
 public class CarbonMapGenerator
 {
 public void CreateCarbonMap(
 RasterData carbonData,
 string outputPath,
 ColorRamp colorRamp)
 {
 // Генерация тематической карты запасов углерода
 // Экспорт в GeoTIFF и/или другие форматы
 }

 public void CreateUncertaintyMap(
 RasterData carbonData,
 RasterData uncertaintyData,
 string outputPath)
 {
 // Создание карты с визуализацией неопределенности прогноза
 }
 }

 // Класс для генерации отчетов
 public class ReportGenerator
 {
 public void GenerateCarbonReport(
 RasterData carbonMap,
```

```
IEnumerable<PolygonData> forestStands,
string outputPath)
{
// Расчет статистики по запасам углерода
// Генерация таблиц и графиков
// Формирование PDF отчета
}
}
}
```

6. Уровень приложения (Application Layer)

C#

```
namespace CarbonMapping.App
{
// Главный класс приложения
public class CarbonMappingApplication
{
private readonly IDataSource<CarbonSamplePoint> _fieldDataSource;
private readonly IDataSource<RasterData> _rasterDataSource;
private readonly DataPreprocessor _preprocessor;
private readonly CarbonModelTrainer _modelTrainer;
private readonly SpatialPredictor _predictor;
private readonly CarbonMapGenerator _mapGenerator;

public CarbonMappingApplication(
IDataSource<CarbonSamplePoint> fieldDataSource,
IDataSource<RasterData> rasterDataSource)
{
_fieldDataSource = fieldDataSource;
_rasterDataSource = rasterDataSource;
```

```
_preprocessor = new DataPreprocessor();
_modelTrainer = new CarbonModelTrainer();
_predictor = new SpatialPredictor("models/carbon_model.zip");
_mapGenerator = new CarbonMapGenerator();
}

public async Task RunFullWorkflowAsync(string outputFolder)
{
// Загрузка полевых данных
var fieldData = await _fieldDataSource.GetDataAsync();

// Загрузка растровых данных
var rasterLayers = await _rasterDataSource.GetDataAsync();

// Предварительная обработка данных
var processedData = _preprocessor.CleanData(
_preprocessor.ExtractFeatures(fieldData, rasterLayers));

// Подготовка данных для ML.NET
var mlData = ConvertToMLDataView(processedData);

// Разделение на обучающую и тестовую выборки
var dataSplit = _mlContext.Data.TrainTestSplit(mlData);

// Обучение моделей
var linearModel = _modelTrainer.TrainLinearModel(dataSplit.TrainSet);
var gbModel = _modelTrainer.TrainGradientBoostingModel(dataSplit.TrainSet);

// Оценка моделей
var linearMetrics = _modelTrainer.EvaluateModel(linearModel, dataSplit.TestSet);
var gbMetrics = _modelTrainer.EvaluateModel(gbModel, dataSplit.TestSet);
```

```
// Выбор лучшей модели
var bestModel = gbMetrics.RSquared > linearMetrics.RSquared ? gbModel : linearModel;

// Сохранение модели
_modelTrainer.SaveModel(bestModel, mlData.Schema, "models/best_carbon_model.zip");

// Генерация карты запасов углерода
var carbonMap = _predictor.PredictCarbonMap(rasterLayers);

// Создание и экспорт карты
_mapGenerator.CreateCarbonMap(carbonMap, $"{outputFolder}/carbon_map.tiff", ColorRamp.GreenToRed);

// Генерация отчета
var reportGenerator = new ReportGenerator();
reportGenerator.GenerateCarbonReport(carbonMap, null, $"{outputFolder}/carbon_report.pdf");
}
}
}
```

7. Уровень пользовательского интерфейса (UI Layer)

В зависимости от требований, система может включать различные варианты пользовательского интерфейса:

- консольное приложение для автоматизированных процессов;
- настольное приложение с графическим интерфейсом для интерактивной работы;
- веб-интерфейс для распределенного доступа;
- API для интеграции с другими системами.

Описанная структура программного решения обеспечивает:

- четкое разделение ответственности между компонентами;
- возможность независимого тестирования каждого модуля;
- гибкость в замене отдельных компонентов;
- масштабируемость для обработки больших объемов данных;
- расширяемость для добавления новых функциональных возможностей.

Интеграция геопространственных данных является ключевым аспектом системы картирования углерода, поскольку требует эффективного управления и обработки разнородной пространственной информации. Для работы с геоданными в решении используются следующие подходы и технологии.

1. Библиотеки для работы с геопространственными данными

В проекте используются .NET-обертки для известных геопространственных библиотек:

C#

```
// Добавление ссылок на NuGet-пакеты
// Install-Package NetTopologySuite
// Install-Package MaxRev.Gdal.Core
// Install-Package MaxRev.Gdal.WindowsRuntime.Minimal

using NetTopologySuite.Geometries;
using NetTopologySuite.IO;
using OSGeo.GDAL;
using OSGeo.OGR;
using OSGeo.OSR;
```

2. Классы для работы с растровыми данными

C#

```
namespace CarbonMapping.Geo
{
 public class RasterProcessor
 {
 static RasterProcessor()
 {
 // Инициализация GDAL
 GdalConfiguration.ConfigureGdal();
 Gdal.AllRegister();
 }
```

```
public RasterData LoadRaster(string filePath)
{
using var dataset = Gdal.Open(filePath, Access.GA_ReadOnly);
if (dataset == null)
throw new Exception($"Unable to open raster file: {filePath}");

var band = dataset.GetRasterBand(1);
int width = dataset.RasterXSize;
int height = dataset.RasterYSize;

// Получение геотрансформации
double[] geoTransform = new double[6];
dataset.GetGeoTransform(geoTransform);

// Чтение данных
float[] rasterData = new float[width * height];
band.ReadRaster(0, 0, width, height, rasterData, width, height, 0, 0);

// Получение проекции
var projection = dataset.GetProjectionRef();

return new RasterData
{
Name = Path.GetFileNameWithoutExtension(filePath),
Values = Array.ConvertAll(rasterData, x => (double)x),
Width = width,
Height = height,
XOrigin = geoTransform[0],
YOrigin = geoTransform[3],
CellSize = geoTransform[1],
Projection = projection
};
}

public void SaveRaster(RasterData rasterData, string outputPath)
{
// Получение драйвера для GeoTIFF
var driver = Gdal.GetDriverByName("GTiff");

// Создание нового растра
using var dataset = driver.Create(
```

```
outputPath,
rasterData.Width,
rasterData.Height,
1,
DataType.GDT_Float32,
new string[] { "COMPRESS=DEFLATE", "TILED=YES" });

// Установка геотрансформации
double[] geoTransform = new double[6];
geoTransform[0] = rasterData.XOrigin; // X origin
geoTransform[1] = rasterData.CellSize; // X pixel size
geoTransform[2] = 0; // X rotation
geoTransform[3] = rasterData.YOrigin; // Y origin
geoTransform[4] = 0; // Y rotation
geoTransform[5] = -rasterData.CellSize; // Y pixel size (negative)

dataset.SetGeoTransform(geoTransform);

// Установка проекции
dataset.SetProjection(rasterData.Projection);

// Запись данных
var band = dataset.GetRasterBand(1);
float[] floatData = Array.ConvertAll(rasterData.Values, x => (float)x);
band.WriteRaster(0, 0, rasterData.Width, rasterData.Height, floatData, raster-
Data.Width, rasterData.Height, 0, 0);

// Установка NoData Value если необходимо
band.SetNoDataValue(-9999);

// Построение пирамидальных слоев для ускорения отображения
dataset.BuildOverviews("NEAREST", new int[] { 2, 4, 8, 16 });
}

public RasterData ExtractValuesAtPoints(RasterData raster, IEnumerable<Coordi-
nate> points)
{
// Извлечение значений растра в заданных точках
}
```

```
public RasterData CalculateSpectralIndex(RasterData band1, RasterData band2, string indexType)
{
// Расчет спектральных индексов (NDVI, EVI и др.)
}

public RasterData ApplyModel(ITransformer model, params RasterData[] featureRasters)
{
// Применение ML.NET модели к набору растровых слоев
// для создания прогнозной карты
}
}
}
```

3. Классы для работы с векторными данными

C#

```
namespace CarbonMapping.Geo
{
public class VectorProcessor
{
static VectorProcessor()
{
// Инициализация OGR
GdalConfiguration.ConfigureOgr();
Ogr.RegisterAll();
}

public IEnumerable<PolygonData> LoadPolygons(string filePath)
{
using var dataSource = Ogr.Open(filePath, 0);
if (dataSource == null)
throw new Exception($"Unable to open vector file: {filePath}");

var layer = dataSource.GetLayerByIndex(0);
layer.ResetReading();
```

```
var polygons = new List<PolygonData>();
var feature = layer.GetNextFeature();

while (feature != null)
{
var geometry = feature.GetGeometryRef();
if (geometry != null ML.NET ML.NET geometry.GetGeometryType() == wkbGeom-
etryType.wkbPolygon)
{
var polygon = new PolygonData
{
Geometry = ConvertToNetTopologySuitePolygon(geometry),
Attributes = ExtractAttributes(feature, layer)
};

polygons.Add(polygon);
}

feature = layer.GetNextFeature();
}

return polygons;
}

private Polygon ConvertToNetTopologySuitePolygon(Geometry ogrGeometry)
{
// Конвертация геометрии OGR в формат NetTopologySuite
}

private Dictionary<string, object> ExtractAttributes(Feature feature, Layer layer)
{
// Извлечение атрибутивной информации из фичи
}

public void SavePolygons(IEnumerable<PolygonData> polygons, string outputPath)
{
// Сохранение полигонов в файл (Shapefile, GeoJSON и т. д.)
}
```

```
public IEnumerable<CarbonSamplePoint> ExtractPointsFromVector(string filePath)
{
// Извлечение точек полевых измерений из векторного файла
}

public IEnumerable<PolygonData> CalculateZonalStatistics(
IEnumerable<PolygonData> polygons,
RasterData raster)
{
// Расчет зональной статистики (среднее, минимум, максимум, сумма)
// для значений растра в пределах каждого полигона
}
}

public class PolygonData
{
public Polygon Geometry { get; set; }
public Dictionary<string, object> Attributes { get; set; }
}
}
```

4. Классы для пространственного анализа

C#

```
namespace CarbonMapping.Geo
{
public class SpatialAnalysis
{
public RasterData CalculateTerrainAttributes(RasterData dem)
{
// Расчет производных характеристик рельефа (уклон, экспозиция)
}

public RasterData CalculateDistanceToFeatures(RasterData landcover, int tar-
getClass)
{
```

```
 // Расчет расстояния до ближайших объектов заданного класса
 }

 public RasterData InterpolateValues(IEnumerable<CarbonSamplePoint> points, RasterData template)
 {
 // Пространственная интерполяция точечных данных
 }

 public RasterData ApplyFocalStatistics(RasterData input, int windowSize, string statType)
 {
 // Расчет фокальной статистики (скользящее окно)
 }

 public RasterData MaskByPolygons(RasterData input, IEnumerable<PolygonData> masks)
 {
 // Маскирование растра по полигонам
 }
 }
}
```

5. Интеграция геоданных с ML.NET

Для эффективной интеграции геопространственных данных с ML.NET разработаны следующие компоненты:

```
C#
namespace CarbonMapping.Integration
{
 public class GeoMLIntegration
```

```
{
private readonly MLContext _mlContext;

public GeoMLIntegration()
{
_mlContext = new MLContext(seed: 42);
}

public IDataView CreateDataViewFromPoints(
IEnumerable<CarbonSamplePoint> points,
IEnumerable<RasterData> rasters)
{
// Извлечение значений растров в точках
var enrichedPoints = points.Select(p =>
{
var result = new CarbonSamplePoint
{
Longitude = p.Longitude,
Latitude = p.Latitude,
CarbonStock = p.CarbonStock
};

foreach (var raster in rasters)
{
// Определение типа растра и извлечение значения
// Дополнение точек значениями из растров
}

return result;
});
```

```
// Преобразование в формат IDataView для ML.NET
return _mlContext.Data.LoadFromEnumerable(enrichedPoints);
}

public RasterData PredictRaster(
ITransformer model,
IEnumerable<RasterData> featureRasters,
RasterData template)
{
// Подготовка признаков для каждого пикселя
var width = template.Width;
var height = template.Height;
var predictedValues = new double[width * height];

// Применение модели к каждому пикселю
// Это может быть оптимизировано для обработки блоками
for (int y = 0; y < height; y++)
{
for (int x = 0; x < width; x++)
{
var pixelFeatures = ExtractPixelFeatures(featureRasters, x, y);
var prediction = PredictSinglePixel(model, pixelFeatures);
predictedValues[y * width + x] = prediction;
}
}

// Создание результирующего растра
return new RasterData
{
```

```
Name = "CarbonStock",
Values = predictedValues,
Width = width,
Height = height,
XOrigin = template.XOrigin,
YOrigin = template.YOrigin,
CellSize = template.CellSize,
Projection = template.Projection
};
}

private float[] ExtractPixelFeatures(IEnumerable<RasterData> rasters, int x, int y)
{
// Извлечение значений всех признаков для данного пикселя
}

private float PredictSinglePixel(ITransformer model, float[] features)
{
// Применение модели к одному пикселю
}
}
}
```

Интеграция геопространственных данных в систему картирования углерода позволяет:

- объединять данные из различных источников (полевые измерения, спутниковые снимки, топографические данные);
- извлекать пространственные признаки для обучения моделей машинного обучения;
- применять обученные модели для прогнозирования запасов углерода на обширных территориях;

- визуализировать результаты в виде тематических карт;
- проводить пространственный анализ для выявления закономерностей и тенденций в распределении запасов углерода.

Реализация алгоритмов машинного обучения в системе картирования углерода осуществляется с использованием возможностей платформы ML.NET .

Смотрим ключевые компоненты, обеспечивающие интеграцию алгоритмов ML в общую архитектуру системы.

1. Класс для управления моделями машинного обучения

C#

```
namespace CarbonMapping.MachineLearning
{
 public class CarbonModelManager
 {
 private readonly MLContext _mlContext;
 private readonly string _modelsDirectory;

 public CarbonModelManager(string modelsDirectory)
 {
 _mlContext = new MLContext(seed: 42);
 _modelsDirectory = modelsDirectory;
 Directory.CreateDirectory(_modelsDirectory);
 }

 // Класс для определения входных данных модели
 public class ModelInput
 {
 [LoadColumn(0)]
 public float CarbonStock { get; set; }
```

```
[LoadColumn(1)]
public float NDVI { get; set; }

[LoadColumn(2)]
public float EVI { get; set; }

[LoadColumn(3)]
public float ForestHeight { get; set; }

[LoadColumn(4)]
public float Slope { get; set; }

[LoadColumn(5)]
public float Aspect { get; set; }

[LoadColumn(6)]
public float ForestAge { get; set; }

[LoadColumn(7)]
public float MeanTemperature { get; set; }

[LoadColumn(8)]
public float Precipitation { get; set; }

[LoadColumn(9)]
public string ForestType { get; set; }

[LoadColumn(10)]
public string SoilType { get; set; }
```

```
}

// Класс для определения выходных данных модели
public class ModelOutput
{
[ColumnName("Score")]
public float PredictedCarbonStock { get; set; }
}

public ITransformer TrainLinearRegressionModel(IDataView trainingData)
{
// Определение пайплайна для линейной регрессии
var featureColumns = trainingData.Schema
.Select(col => col.Name)
.Where(name => name != "CarbonStock")
.ToArray();

var pipeline = _mlContext.Transforms.Categorical.OneHotEncoding("ForestType")
.Append(_mlContext.Transforms.Categorical.OneHotEncoding("SoilType"))
.Append(_mlContext.Transforms.Concatenate("Features", featureColumns))
.Append(_mlContext.Transforms.NormalizeMinMax("Features"))
.Append(_mlContext.Regression.Trainers.FastLinear(
labelColumnName: "CarbonStock",
featureColumnName: "Features",
l2Regularization: 0.01));

Console.WriteLine("Training linear regression model...");
var model = pipeline.Fit(trainingData);
Console.WriteLine("Model training completed.");
```

```
return model;
}

public ITransformer TrainGradientBoostingModel(IDataView trainingData)
{
// Определение пайплайна для LightGBM
var featureColumns = trainingData.Schema
.Select(col => col.Name)
.Where(name => name != "CarbonStock")
.ToArray();

var pipeline = _mlContext.Transforms.Categorical.OneHotEncoding("ForestType")
.Append(_mlContext.Transforms.Categorical.OneHotEncoding("SoilType"))
.Append(_mlContext.Transforms.Concatenate("Features", featureColumns))
.Append(_mlContext.Regression.Trainers.LightGbm(
labelColumnName: "CarbonStock",
featureColumnName: "Features",
numberOfLeaves: 31,
numberOfIterations: 100,
minimumExampleCountPerLeaf: 20,
learningRate: 0.1));

Console.WriteLine("Training gradient boosting model...");
var model = pipeline.Fit(trainingData);
Console.WriteLine("Model training completed.");

return model;
}
```

```
public ITransformer TrainRandomForestModel(IDataView trainingData)
{
// Определение пайплайна для FastForest
var featureColumns = trainingData.Schema
.Select(col => col.Name)
.Where(name => name != "CarbonStock")
.ToArray();

var pipeline = _mlContext.Transforms.Categorical.OneHotEncoding("ForestType")
.Append(_mlContext.Transforms.Categorical.OneHotEncoding("SoilType"))
.Append(_mlContext.Transforms.Concatenate("Features", featureColumns))
.Append(_mlContext.Regression.Trainers.FastForest(
labelColumnName: "CarbonStock",
featureColumnName: "Features",
numberOfTrees: 100,
numberOfLeaves: 20,
minimumExampleCountPerLeaf: 10));

Console.WriteLine("Training random forest model...");
var model = pipeline.Fit(trainingData);
Console.WriteLine("Model training completed.");

return model;
}

public RegressionMetrics EvaluateModel(ITransformer model, IDataView testData)
{
// Оценка качества модели
var predictions = model.Transform(testData);
```

```
var metrics = _mlContext.Regression.Evaluate(
data: predictions,
labelColumnName: "CarbonStock",
scoreColumnName: "Score");

Console.WriteLine($"R²: {metrics.RSquared:F4}");
Console.WriteLine($"RMSE: {metrics.RootMeanSquaredError:F4}");
Console.WriteLine($"MAE: {metrics.MeanAbsoluteError:F4}");

return metrics;
}

public void SaveModel(ITransformer model, DataViewSchema schema, string mod-
elName)
{
// Сохранение модели
string modelPath = Path.Combine(_modelsDirectory, $"{modelName}.zip");
_mlContext.Model.Save(model, schema, modelPath);
Console.WriteLine($"Model saved to {modelPath}");
}

public ITransformer LoadModel(string modelName)
{
// Загрузка модели
string modelPath = Path.Combine(_modelsDirectory, $"{modelName}.zip");
DataViewSchema schema;
var model = _mlContext.Model.Load(modelPath, out schema);
Console.WriteLine($"Model loaded from {modelPath}");
return model;
```

```
}

public void PerformCrossValidation(IDataView data, int folds = 5)
{
// Кросс-валидация для оценки устойчивости модели
Console.WriteLine($"Performing {folds}-fold cross-validation...");

var featureColumns = data.Schema
.Select(col => col.Name)
.Where(name => name != "CarbonStock")
.ToArray();

var pipeline = _mlContext.Transforms.Categorical.OneHotEncoding("ForestType")
.Append(_mlContext.Transforms.Categorical.OneHotEncoding("SoilType"))
.Append(_mlContext.Transforms.Concatenate("Features", featureColumns))
.Append(_mlContext.Regression.Trainers.LightGbm(
labelColumnName: "CarbonStock",
featureColumnName: "Features"));

var cvResults = _mlContext.Regression.CrossValidate(
data: data,
estimator: pipeline,
numberOfFolds: folds,
labelColumnName: "CarbonStock");

var metrics = cvResults.Select(r => r.Metrics);

Console.WriteLine($"Average R²: {metrics.Average(m => m.RSquared):F4}");
```

```
Console.WriteLine($"Average RMSE: {metrics.Average(m => m.Root-MeanSquaredError):F4}");
Console.WriteLine($"Average MAE: {metrics.Average(m => m.MeanAbso-luteError):F4}");
Console.WriteLine($"Standard Deviation of R²: {metrics.StandardDeviation(m => m.RSquared):F4}");
}

public void AnalyzeFeatureImportance(ITransformer model, DataViewSchema schema)
{
// Анализ важности признаков
// Работает только для определенных типов моделей (FastTree, LightGBM)
try
{
// Для моделей на базе деревьев решений
var permutationMetrics = _mlContext.Regression.PermutationFeatureImportance(
model,
data: _mlContext.Data.LoadFromEnumerable(new List<ModelInput>()),
labelColumnName: "CarbonStock",
permutationCount: 30);

var featureImportance = permutationMetrics
.Select((metrics, index) => new { Feature = schema[index].Name, metrics.RSquared })
.OrderByDescending(feature => Math.Abs(feature.RSquared.Mean));

Console.WriteLine("Feature Importance (based on permutation importance):");
foreach (var feature in featureImportance)
```

```
{
Console.WriteLine($"{feature.Feature}: {feature.RSquared.Mean:F6}");
}
}
catch (Exception ex)
{
Console.WriteLine($"Feature importance analysis failed: {ex.Message}");
}
}
}
}
```

2. Класс для автоматического подбора оптимальной модели

C#

```
namespace CarbonMapping.MachineLearning
{
public class AutoMLModelSelector
{
private readonly MLContext _mlContext;

public AutoMLModelSelector()
{
_mlContext = new MLContext(seed: 42);
}

public ExperimentResult<RegressionMetrics> FindBestModel(
IDataView data,
string labelColumnName,
TimeSpan maxTime)
{
```

```
Console.WriteLine("Starting AutoML experiment...");

// Настройка эксперимента AutoML
var experimentSettings = new RegressionExperimentSettings
{
MaxExperimentTimeInSeconds = (int)maxTime.TotalSeconds,
OptimizingMetric = RegressionMetric.RSquared,
CacheDirectoryName = Path.Combine(Path.GetTempPath(), "CarbonMappingAutoML")
};

// Определение алгоритмов для AutoML
experimentSettings.Trainers.Add(RegressionTrainer.FastForest);
experimentSettings.Trainers.Add(RegressionTrainer.FastTree);
experimentSettings.Trainers.Add(RegressionTrainer.LightGbm);
experimentSettings.Trainers.Add(RegressionTrainer.FastLinear);
experimentSettings.Trainers.Add(RegressionTrainer.Sdca);

// Создание и запуск эксперимента
var experiment = _mlContext.Auto().CreateRegressionExperiment(experimentSettings);

// Обработчик прогресса
var progressHandler = new RegressionExperimentProgressHandler();

var result = experiment.Execute(
trainData: data,
labelColumnName: labelColumnName,
progressHandler: progressHandler);
```

```
// Вывод результатов
Console.WriteLine("AutoML experiment completed.");
Console.WriteLine($"Best algorithm: {result.BestRun.TrainerName}");
Console.WriteLine($"R²: {result.BestRun.ValidationMetrics.RSquared:F4}");
Console.WriteLine($"RMSE: {result.BestRun.ValidationMetrics.RootMeanSquaredError:F4}");
Console.WriteLine($"MAE: {result.BestRun.ValidationMetrics.MeanAbsoluteError:F4}");

return result;
}

// Класс для отслеживания прогресса AutoML
private class RegressionExperimentProgressHandler : IProgress<RunDetail<RegressionMetrics>>
{
public void Report(RunDetail<RegressionMetrics> value)
{
Console.WriteLine($"Completed run for {value.TrainerName} with R² = {value.ValidationMetrics.RSquared:F4}");
}
}
}
}
```

3. Класс для применения модели к пространственным данным

C#

```
namespace CarbonMapping.MachineLearning
{
public class SpatialModelApplicator
```

```
{
private readonly MLContext _mlContext;
private readonly ITransformer _model;
private readonly PredictionEngine<CarbonModelManager.ModelInput, CarbonModelManager.ModelOutput> _predictionEngine;

public SpatialModelApplicator(ITransformer model)
{
_mlContext = new MLContext(seed: 42);
_model = model;
_predictionEngine = _mlContext.Model.CreatePredictionEngine<CarbonModelManager.ModelInput, CarbonModelManager.ModelOutput>(model);
}

public RasterData CreateCarbonMap(
Dictionary<string, RasterData> featureRasters,
Dictionary<string, string> categoricalValues = null)
{
// Проверка наличия всех необходимых растров
var requiredFeatures = new[]
{
"NDVI", "EVI", "ForestHeight", "Slope", "Aspect",
"ForestAge", "MeanTemperature", "Precipitation"
};

foreach (var feature in requiredFeatures)
{
if (!featureRasters.ContainsKey(feature))
{
```

```
throw new ArgumentException($"Required feature raster missing: {feature}");
}
}

// Получение параметров выходного растра из первого входного
var templateRaster = featureRasters.First().Value;
int width = templateRaster.Width;
int height = templateRaster.Height;
double[] resultValues = new double[width * height];

Console.WriteLine($"Creating carbon map with dimensions {width}x{height}...");

// Обработка по блокам для оптимизации памяти
int blockSize = 1000; // Размер блока для обработки

for (int yOffset = 0; yOffset < height; yOffset += blockSize)
{
int blockHeight = Math.Min(blockSize, height – yOffset);

for (int xOffset = 0; xOffset < width; xOffset += blockSize)
{
int blockWidth = Math.Min(blockSize, width – xOffset);

// Извлечение данных блока для всех растров
var blockData = ExtractBlockData(featureRasters, xOffset, yOffset, blockWidth, blockHeight);

// Применение модели к каждому пикселю в блоке
for (int y = 0; y < blockHeight; y++)
```

```
{
for (int x = 0; x < blockWidth; x++)
{
int globalX = xOffset + x;
int globalY = yOffset + y;
int globalIndex = globalY * width + globalX;

// Проверка на NoData
bool hasNoData = false;
foreach (var feature in requiredFeatures)
{
if (blockData[feature][y * blockWidth + x] == -9999)
{
hasNoData = true;
break;
}
}

if (hasNoData)
{
resultValues[globalIndex] = -9999; // NoData value
continue;
}

// Создание входных данных для модели
var input = new CarbonModelManager.ModelInput
{
NDVI = (float)blockData["NDVI"][y * blockWidth + x],
EVI = (float)blockData["EVI"][y * blockWidth + x],
```

```
ForestHeight = (float)blockData["ForestHeight"][y * blockWidth + x],
Slope = (float)blockData["Slope"][y * blockWidth + x],
Aspect = (float)blockData["Aspect"][y * blockWidth + x],
ForestAge = (float)blockData["ForestAge"][y * blockWidth + x],
MeanTemperature = (float)blockData["MeanTemperature"][y * blockWidth + x],
Precipitation = (float)blockData["Precipitation"][y * blockWidth + x]
};

// Добавление категориальных переменных если они предоставлены
if (categoricalValues != null)
{
if (categoricalValues.ContainsKey("ForestType"))
input.ForestType = categoricalValues["ForestType"];

if (categoricalValues.ContainsKey("SoilType"))
input.SoilType = categoricalValues["SoilType"];
}

// Получение прогноза
var prediction = _predictionEngine.Predict(input);
resultValues[globalIndex] = prediction.PredictedCarbonStock;
}
}
}

Console.WriteLine($"Processed {Math.Min((yOffset + blockSize) * 100 / height, 100)}% of the image");
}
```

```
// Создание результирующего растра
return new RasterData
{
Name = "CarbonStock",
Values = resultValues,
Width = width,
Height = height,
XOrigin = templateRaster.XOrigin,
YOrigin = templateRaster.YOrigin,
CellSize = templateRaster.CellSize,
Projection = templateRaster.Projection
};
}

private Dictionary<string, double[]> ExtractBlockData(
Dictionary<string, RasterData> rasters,
int xOffset,
int yOffset,
int blockWidth,
int blockHeight)
{
var result = new Dictionary<string, double[]>();

foreach (var kvp in rasters)
{
var raster = kvp.Value;
var blockData = new double[blockWidth * blockHeight];

for (int y = 0; y < blockHeight; y++)
```

```
{
for (int x = 0; x < blockWidth; x++)
{
int srcX = xOffset + x;
int srcY = yOffset + y;

if (srcX < raster.Width ML.NET ML.NET srcY < raster.Height)
{
blockData[y * blockWidth + x] = raster.Values[srcY * raster.Width + srcX];
}
else
{
blockData[y * blockWidth + x] = -9999; // NoData value
}
}
}

result[kvp.Key] = blockData;
}

return result;
}

public RasterData CreateUncertaintyMap(
Dictionary<string, RasterData> featureRasters,
int numberOfIterations = 10)
{
// Создание карты неопределенности прогноза
// (например, путем возмущения входных данных и анализа вариации результатов)
```

```
 }
 }
}
```

4. Расширения для статистического анализа

C#

```
namespace CarbonMapping.MachineLearning
{
 public static class StatisticsExtensions
 {
 public static double Average<T>(this IEnumerable<T> source, Func<T, double> selector)
 {
 return source.Select(selector).Average();
 }

 public static double StandardDeviation<T>(this IEnumerable<T> source, Func<T, double> selector)
 {
 var values = source.Select(selector).ToList();
 if (values.Count <= 1)
 return 0;

 double avg = values.Average();
 double sum = values.Sum(d => Math.Pow(d – avg, 2));
 return Math.Sqrt(sum / (values.Count – 1));
 }

 public static double Correlation(double[] x, double[] y)
 {
 if (x.Length != y.Length)
 throw new ArgumentException("Arrays must have the same length");
```

```
 int n = x.Length;
 double sumX = 0, sumY = 0, sumXY = 0, sumX2 = 0, sumY2 = 0;

 for (int i = 0; i < n; i++)
 {
 sumX += x[i];
 sumY += y[i];
 sumXY += x[i] * y[i];
 sumX2 += x[i] * x[i];
 sumY2 += y[i] * y[i];
 }

 double numerator = n * sumXY – sumX * sumY;
 double denominator = Math.Sqrt((n * sumX2 – sumX * sumX) * (n * sumY2 – sumY
* sumY));

 return denominator == 0 ? 0 : numerator / denominator;
 }
 }
}
```

5. Главный класс для обучения и применения моделей

C#

```
namespace CarbonMapping.MachineLearning
{
 public class CarbonMappingModelWorkflow
 {
 private readonly MLContext _mlContext;
 private readonly CarbonModelManager _modelManager;
 private readonly AutoMLModelSelector _autoMLSelector;

 public CarbonMappingModelWorkflow(string modelsDirectory)
 {
```

```
_mlContext = new MLContext(seed: 42);
_modelManager = new CarbonModelManager(modelsDirectory);
_autoMLSelector = new AutoMLModelSelector();
}

public async Task<ITransformer> TrainModelsAndSelectBestAsync(
string trainingDataPath,
bool useAutoML = false,
TimeSpan? autoMLMaxTime = null)
{
Console.WriteLine("Loading training data...");

// Загрузка данных
var data = _mlContext.Data.LoadFromTextFile<CarbonModelManager.ModelInput>(
path: trainingDataPath,
hasHeader: true,
separatorChar: ',');

// Разделение на обучающую и тестовую выборки
var dataSplit = _mlContext.Data.TrainTestSplit(data, testFraction: 0.2);
var trainData = dataSplit.TrainSet;
var testData = dataSplit.TestSet;

ITransformer bestModel;

if (useAutoML ML.NET ML.NET autoMLMaxTime.HasValue)
{
// Использование AutoML для подбора оптимальной модели
var experimentResult = _autoMLSelector.FindBestModel(
data: trainData,
labelColumnName: "CarbonStock",
```

```
maxTime: autoMLMaxTime.Value);

bestModel = experimentResult.BestRun.Model;
// Оценка лучшей модели
var metrics = _modelManager.EvaluateModel(bestModel, testData);

// Сохранение модели
_modelManager.SaveModel(bestModel, data.Schema, "automl_best_model");
}
else
{
// Обучение нескольких моделей и выбор лучшей
Console.WriteLine("Training multiple models...");

var linearModel = _modelManager.TrainLinearRegressionModel(trainData);
var linearMetrics = _modelManager.EvaluateModel(linearModel, testData);

var gbModel = _modelManager.TrainGradientBoostingModel(trainData);
var gbMetrics = _modelManager.EvaluateModel(gbModel, testData);

var rfModel = _modelManager.TrainRandomForestModel(trainData);
var rfMetrics = _modelManager.EvaluateModel(rfModel, testData);

// Выбор лучшей модели по R²
Console.WriteLine("Selecting best model...");

double bestR2 = Math.Max(linearMetrics.RSquared,
Math.Max(gbMetrics.RSquared, rfMetrics.RSquared));

if (bestR2 == linearMetrics.RSquared)
{
bestModel = linearModel;
```

```
_modelManager.SaveModel(bestModel, data.Schema, "linear_regression_model");
Console.WriteLine("Linear Regression selected as best model.");
}
else if (bestR2 == gbMetrics.RSquared)
{
bestModel = gbModel;
_modelManager.SaveModel(bestModel, data.Schema, "gradient_boosting_model");
Console.WriteLine("Gradient Boosting selected as best model.");
}
else
{
bestModel = rfModel;
_modelManager.SaveModel(bestModel, data.Schema, "random_forest_model");
Console.WriteLine("Random Forest selected as best model.");
}
}

// Кросс-валидация для выбранной модели
_modelManager.PerformCrossValidation(data);

// Анализ важности признаков
_modelManager.AnalyzeFeatureImportance(bestModel, data.Schema);

return bestModel;
}

public async Task<RasterData> CreateCarbonMapAsync(
ITransformer model,
Dictionary<string, RasterData> featureRasters,
string outputPath)
{
Console.WriteLine("Creating carbon map...");
```

```
// Применение модели к пространственным данным
var applicator = new SpatialModelApplicator(model);
var carbonMap = applicator.CreateCarbonMap(featureRasters);

// Сохранение результата
var rasterProcessor = new RasterProcessor();
rasterProcessor.SaveRaster(carbonMap, outputPath);

Console.WriteLine($"Carbon map saved to {outputPath}");

return carbonMap;
}
}
}
```

Реализованные алгоритмы машинного обучения в ML.NET позволяют:

1. обучать различные типы регрессионных моделей (линейная регрессия, градиентный бустинг, случайный лес);

2. автоматически подбирать оптимальные гиперпараметры и алгоритмы с помощью AutoML;

3. оценивать качество моделей на тестовых данных и с помощью кросс-валидации;

4. анализировать важность признаков для понимания факторов, влияющих на запасы углерода;

5. применять обученные модели к пространственным данным для создания карт запасов углерода;

6. оценивать неопределенность прогнозов.

Такой подход обеспечивает гибкость в выборе алгоритмов и возможность адаптации системы к различным типам лесных экосистем и наборам исходных данных.

ГЛАВА 5. ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ ДЛЯ ОЦЕНКИ КАРТИРОВАНИЯ УГЛЕРОДА, ДЕПОНИРУЕМОГО ЛЕСАМИ

Будущее исследований в данной области обещает множество новых направлений, связанных с применением ИИ для более точной оценки углерода, накопленного лесами. Разработка моделей, способных учитывать различные эко системные факторы, открывает перспективы для создания адаптивных инструментов, которые могут функционировать на региональном уровне и учитывать уникальные биомассы и условия среды. Внедрение таких технологий значительно повысит точность мониторинга и управления углеродными резервами.

Перспективные исследования должны сосредоточиться на интеграции данных из различных источников, включая спутниковые снимки и наземные измерения, для создания более полных моделей. Успешное использование ИИ в этом контексте поможет не только в прогнозировании динамики углеродного депонирования, но и в оценке влияния изменения климата на лесные экосистемы.

Кроме того, необходима адаптация существующих моделей к различным видам растительности и характеру экосистем, что особенно актуально для регионов с высокой биологической разнообразностью [62].

С учетом успешных примеров внедрения ИИ в смежные области, таких как энергетика, можно ожидать значительного роста интереса к подобным проектам в области экологии. Так, по данным исследований, более 58% организаций в энергетическом секторе начали активно использовать ИИ в 2024 году [63]. Это свидетельствует о потенциальной готовности аналогичных решений и в управлении углеродными ресурсами лесов.

Способности искусственного интеллекта открывают возможности для паттерн-анализа, который может помочь выявить важные взаимосвязи между экосистемами и углеродными процессами. Важно также развивать методы автоматического сбора и анализа данных, что позволит повысить эффективность мониторинга углерода и упростить управление природными ресурсами на уровне

предприятий и государственных организаций. Это даст возможность не только оптимизировать производственные процессы, но и создать более устойчивые системы управления лесными ресурсами [64, 65].

С учетом динамики изменений в области экологии и технологий, необходимо взаимодействие между учеными, экосистемными менеджерами и политиками. Такое сотрудничество обеспечит более эффективное применение ИИ для решения задач, связанных с углеродным депонированием. Важно наращивать масштабы исследования и адаптировать модели, чтобы они были применимы в новых условиях и для других лесных литосферных контекстов, включая урбанизированные или аграрные районы.

Эти усилия подчеркивают важность деятельности по моделированию и управлению углеродом в лесных экосистемах, направленной на создание устойчивых и эффективных подходов, способных учитывать множество переменных. Это подчеркивает значимость интеграции ИИ в охрану окружающей среды.

В условиях глобального изменения климата и нарастающей угрозы экологическим системам, эффективное управление углеродными ресурсами становится одной из ключевых задач для обеспечения устойчивого развития.

Проблема отсутствия точных инструментов для оценки углерода, накопленного лесами, требует комплексного подхода, который включает в себя как сбор и анализ данных, так и применение современных технологий, таких как искусственный интеллект.

Использование спутниковых технологий, дронов и наземных исследований позволяет значительно повысить точность и объем собираемых данных, что, в свою очередь, способствует более глубокому анализу экосистем.

Обработка и анализ собранных данных с применением методов машинного обучения и статистического моделирования открывают новые горизонты для понимания процессов углеродного депонирования.

Разработка моделей, способных предсказывать накопление углерода на основе различных факторов, позволяет не только оценить текущие запасы углерода, но и спрогнозировать их изменения в будущем. Это особенно важно в условиях изменения климата, когда лесные экосистемы могут подвергаться значительным изменениям.

Будущее исследований в области искусственного интеллекта и управления углеродом выглядит многообещающе. С развитием технологий и увеличением объемов доступных данных, возможности для более точной оценки углерода и оптимизации управления лесами будут только расширяться. Важно продолжать исследовать новые подходы и методы, которые могут помочь в решении актуальных экологических проблем, связанных с углеродным депонированием.

Исследования авторов позволяют определить на будущие 10 лет следующие ключевые направления в среде ИИ.

- Интеграция многомасштабных данных: спутниковая оптика, SAR, воздушный/космический лидар (ALS, GEDI), УБЛ/фотограмметрия дронов, гиперспектр.
- Самообучающиеся и полусупервизируемые модели (self-/semi-supervised) для использования огромного объёма ненадписанных данных.
- Модели с пространственно-временной контекстуализацией (spatio-temporal transformers, ConvLSTM, attention-based UNet).
- Прямое предсказание углерода + оценка неопределённости (probabilistic DL, ensembles, bayesian approaches).
- Глобальные предтренированные модели экосистем (ecofoundation models) и их локальное дообучение.
- Edge/near-edge inference: обработка файлов на удалённых серверах/устройствах для быстрой отчетности.
- Стандарты интероперабельности и прозрачной валидации для MRV/REDD+.

Эффективность технологий ИИ скажется на следующих изменениях в данных:

- масштабные, стандартизированные наборы полевых данных (с геопозиционированием и метаданными), открытые и FAIR;
- координация съёмок ALS/теренов для создания ссылочных «калибровочных» областей;
- принятие пространственно ориентированной кросс-валидации и обязательной валидации на независимых наборах;
- метаданные о датах съёмок и ростовом периоде – для согласования в пространственно-временных моделях.

Применение технологий ИИ кроме преимуществ определяет и риски. В «Национальной стратегии развития искусственного интеллекта на период до 2030 года в РФ» определен большой раздел «Раздел 51.8. – Основными направлениями внедрения доверенных технологий искусственного интеллекта» [66, 67], определяющие важную проблему доверия к ИИ. «Доверенные технологии искусственного интеллекта технологии, отвечающие стандартам безопасности, разработанные с учетом принципов объективности, не дискриминации, этичности, исключающие при их использовании возможность причинения вреда человеку и нарушения его основополагающих прав и свобод, нанесения ущерба интересам общества и государства. Прозрачность: объяснимость работы искусственного интеллекта и процесса достижения им результатов, недискриминационный доступ пользователей продуктов, которые созданы с использованием технологий искусственного интеллекта, к информации о применяемых в этих продуктах алгоритмах работы искусственного интеллекта. Уровень доверия граждан к технологиям искусственного интеллекта в 2030 году должен вырасти не менее чем до 80 процентов по сравнению с 55 процентами в 2022 году.

Популяризация технологий искусственного интеллекта и повышение доверия граждан к ним» [66, 67]. Исследования авторов [68] показали, что доверие к ИИ определяются фундаментальными проблемами, которые в настоящее время должны устраняться.

Существенными проблемами являются следующие.

- Ограниченность ресурсов, как материальных, так и человеческих, наряду с внутренними организационными факторами, оказывает существенное воздействие на эволюцию систем искусственного интеллекта.
- Нехватка вычислительных мощностей, бюджетов и квалифицированных специалистов часто сдерживает разработку более интерпретируемых алгоритмов.
- Фундаментальный фактор результативности алгоритмов машинного обучения – высокое качество обучающего корпуса. Шумные, нерепрезентативные либо смещённые выборки вызывают систематические ошибки модели и ослабляют доверие пользователей. Поэтому критичны многоступенчатая предобработка, корректная разметка, статистический аудит и методы снижения Байеса, минимизирующие влияние артефактов на финальный вывод.
- Пределы применения формальных подходов в сфере, интерпретируемого ИИ заслуживают детального анализа. Да, строгие методы повышают доверие, обеспечивая верифицируемую надежность и прогнозируемость поведения систем, однако их внедрение связано с вычислительной сложностью, ресурса затратностью и, как следствие, тормозящими масштабное промышленное распространение барьерами.

В ходе проведённого анализа различные исследователи выделяют ряд ключевых проблем современного искусственного интеллекта [69]:

- сложность моделей ИИ;
- ресурсные ограничения;
- проблемы с данными;
- организационные причины;
- ограничения современных алгоритмов верификации;
- проблемы с архитектурой нейросетей;
- ограничения в математических основах.

В табл. 1. для важных (с точки зрения авторов) фундаментальных проблемы современного ИИ выполним декомпозицию и определим значение и пути устранения.

Таблица 1. Фундаментальные проблемы современного ИИ.

Сложность моделей ИИ	
Комбинаторный взрыв возможных входов	Это эффект резкого («взрывного») роста временной сложности алгоритма при увеличении размера входных данных задачи. Связано это с тем, что рассматриваемый алгоритм не является полиномиальным, то есть время решения задачи не ограничено никаким многочленом от длины входа. Проблема комбинаторного взрыва была осознана в 1950-х годах, когда были разработаны первые программы ИИ, но её решение до сих пор не найдено. Чтобы справиться с комбинаторным взрывом, нужны алгоритмы, способные анализировать структуру целевой области и использовать преимущества накопленного знания за счёт эвристического поиска, долгосрочного планирования и свободных абстрактных представлений. Как только количество объектов превысит определённое число, рост количества комбинаторных объектов (например, путей в графе), которые перебираются в процессе решения, становится неудержимым. Трудности могут возникнуть не при проверке огромного количества объектов, а гораздо раньше – при пересчёте.
Нелинейность и сложность функций активации	Функция активации – это математические функции, которые определяют выход нейрона на основе его входа, внося нелинейность в модель. Это позволяет сети изучать сложные закономерности и взаимосвязи в данных, которые невозможны при использовании чисто линейной модели. Нелинейность означает, что взаимосвязь между входом и выходом не является прямой: результат не изменяется пропорционально входным данным. Это позволяет моделировать сложные шаблоны, которые невозможны при использовании чисто линейной модели. Например, сеть может создавать изогнутые границы принятия решений для правильного разделения данных, которые сложно разделить линейной функцией. Сложность функций активации зависит от их математических свойств и специфики задачи, для которой создаётся нейронная сеть. Например: Сигмоидальная функция (логистическая) – ограничивает выходные значения в диапазоне [0, 1], что полезно для задач классификации.
Стохастичность обучения и входных данных	Стохастичность в ИИ проявляется в разных аспектах: в процессе обучения моделей и в входных данных. Это означает введение случайности или вероятности в алгоритмы и модели, что позволяет учитывать неопределённость и эффективно обрабатывать зашумлённые или неполные данные.

	Обучение – стохастический градиентный спуск (SGD) – это алгоритм оптимизации, который вносит случайность в обновления параметров. Вместо вычисления градиента с использованием всего набора данных градиент оценивается с использованием случайно выбранного подмножества данных (мини-пакета). Такая случайная выборка привносит стохастичность в процесс оптимизации, делая его более адаптируемым к зашумленным или динамичным данным. Стохастические нейронные сети (SNN) – класс нейронных сетей, которые включают случайность в свою архитектуру и процессы обучения. Входные данные. Использование разных частей данных на разных итерациях помогает избежать застревания модели в локальном минимуме, особенно если оптимизируемая функция обычно невыпуклая.
Стохастичность обучения и входных данных	Учёт изменчивости данных – например, если распределение на данные изменяется со временем, система замедляется в обучении, так как ей приходится долго адаптироваться под изменяющиеся условия. В этом случае стохастичность помогает модели адаптироваться к изменяющимся условиям. Однако стохастическая природа многих алгоритмов ИИ может вызывать проблемы – результаты могут различаться даже при одних и тех же входных данных из-за случайной инициализации или внутренней изменчивости процессов обучения. Чтобы обеспечить согласованное поведение, необходимо использовать статистические методы или исправлять начальные исходные данные.
Динамическое обновление моделей на новых данных	Динамическое обновление моделей на новых данных в ИИ позволяет поддерживать актуальность и точность моделей в долгосрочной перспективе. Динамическое обновление моделей реализуется с помощью методов машинного обучения, алгоритмов и платформ, которые автоматизируют процесс.
Динамическое обновление моделей на новых данных	Один из подходов к динамическому обновлению – Retrieval-Augmented Generation (RAG). При использовании этого метода генеративная языковая модель (LLM) снабжается доступом к внешним источникам информации. Модель перед генерацией ответа выполняет поиск релевантных данных и использует найденные сведения при формировании ответа.
Трудности с моделированием непрерывных пространств состояний	Некоторые трудности, с которыми сталкиваются при моделировании непрерывных пространств состояний в искусственном интеллекте. Невозможность полного перебора. Даже для сравнительно простых задач пространство решений чрезвычайно велико, и исследование всего пространства невозможно. Этот эффект называют комбинаторным взрывом или проклятием размерности. Отсутствие оценивающих функций. Для некоторых задач не существует функций, которые бы позволяли на каждом шаге гарантированно выбирать лучший ход.

	Сложность выражения неформальных знаний. Особенно сложно выразить такие знания в формальных логических терминах, если они не являются полностью достоверными.
Трудности с моделированием непрерывных пространств состояний	Нестабильность существующих моделей. Они могут быть нестабильными или требовать значительного количества вычислительных ресурсов. Для решения этих проблем, например, разработали «линейные колебательные модели пространства состояний» (LinOSS). Этот подход обеспечивает стабильные и вычислительно эффективные прогнозы без чрезмерно ограничивающих условий для параметров модели.
Разнообразие архитектур CNN, RNN, Transformer и др.	Разнообразие архитектур нейросетей ИИ включает разные типы, каждая из которых предназначена для решения определённого круга задач. Сверточные нейросети (CNN). Используют свёртки (фильтры), чтобы находить шаблоны в данных – от простых до сложных. Применяются для распознавания изображений и видео, медицинской диагностики, обнаружения объектов. Рекуррентные нейросети (RNN). Сохраняют состояние при обработке последовательностей – могут учитывать контекст. Используются для распознавания речи, обработки текста, анализа временных рядов. Популярные виды: LSTM, GRU.
Разнообразие архитектур CNN, RNN, Transformer и др.	Трансформеры. Используют механизм внимания и параллельные вычисления для эффективной обработки последовательных данных. Применяются в машинном переводе, генерации текста, чат-ботах и многих других приложениях.
Ресурсные ограничения	
Недостаток вычислительных ресурсов	Недостаток вычислительных ресурсов в ИИ связан с несколькими факторами. Рост спроса на вычислительные мощности. С момента появления ChatGPT в 2023 году спрос на вычислительные ресурсы резко вырос. Облачные платформы, технологические гиганты и стартапы наращивают загрузку дата-центров, обучая и разворачивая всё более сложные модели. Повышенное энергопотребление нейровычислений. Для работы компьютерных систем, на которых обучают нейросети, требуется дополнительное специализированное оборудование с повышенным энергопотреблением.
Недостаток вычислительных ресурсов	Перегрузка энергосистем. Например, в США энергосистема PJM испытывает острую нехватку мощностей на фоне стремительного роста потребления со стороны вычислительных центров и генеративного ИИ. Перспектива масштабного внедрения автономных ИИ-моделей с локальным обучением. Поддержка таких сценариев требует стабильного доступа к значительным объёмам электроэнергии, чего перегруженные сети обеспечить уже не могут. По прогнозам летом 2025 года в регионах высокой концентрации дата-центров прогнозируется рост тарифов на электроэнергию более чем на 20%. Высокие цены и нестабиль-

	ность электроснабжения в пиковые периоды грозят замедлить внедрение ИИ-сервисов и повысить операционные расходы их владельцев.
Высокая стоимость проведения формальной верификации	Формальная верификация в области ИИ может быть высокой из-за сложности алгоритмов и проблем с масштабированием. Традиционно проверка систем ИИ сосредоточена на тестировании – оценке сети на большом наборе точек во входном пространстве и определении, соответствуют ли её выходы желаемым на этом наборе.
Высокая стоимость проведения формальной верификации	Однако формальная верификация предоставляет математические гарантии в отношении верной работы системы для любого входа из пространства возможных входных данных, но требует разработки алгоритмов, подтверждающих соответствие модели желаемым спецификациям для всех возможных входных данных. Причины. Сложность алгоритмов. С масштабированием и усложнением ИИ-систем становится всё сложнее проектировать алгоритмы, достаточно адаптированные к модели ИИ. Проблемы с перебором всех возможных вариантов выходных данных для заданного набора входных (например, незначительных изменений изображения). В крупномасштабных моделях перебор всех возможных вариантов трудно реализуем из-за астрономического количества возможных изменений. Трудности с формулировкой спецификаций – «правильное» поведение ИИ-систем часто трудно сформулировать точно. Методы. Для формальной верификации ИИ используются, например:
Высокая стоимость проведения формальной верификации	Дедуктивная верификация – позволяет свести задачу проверки корректности программы относительно спецификаций к задаче проверки истинности утверждений о том, что программа корректна относительно спецификаций. Такие утверждения – логические формулы, и если все условия корректности истинны, то программа корректна относительно спецификаций. Проверка на модели (model checking) – создание математической модели искусственной нейронной сети и её поведения, описание архитектуры сети, функций активации, весов и других параметров. Стоимость формальной верификации зависит от сложности алгоритмов и сложности проверяемой архитектуры ИИ. Решения. Выполняются работы над снижением стоимости формальной верификации, например: Разрабатывают алгоритмы, адаптированные к модели ИИ. Например, используют методы понижения размерности (дропаут, прунинг) для уменьшения времени работы алгоритмов.
Высокая стоимость проведения формальной верификации	Стандартизируют алгоритмы верификации – это позволит обеспечить простыми в использовании инструментами, кото-

	рые проливающими свет на возможные режимы отказа системы ИИ до того, как этот отказ приведёт к обширным отрицательным последствиям. Расширяют спектр состязательных примеров – это поможет разрабатывать подходы к верификации для свойств, имеющих отношение к реальному миру.
Длительность времени, необходимая для верификации больших моделей	Верификация больших моделей в области ИИ может занимать длительное время из-за сложности адаптации традиционных методов к сложным моделям, содержащим сотни миллионов параметров. Это связано с необходимостью проверки соответствия модели спецификациям, которые описывают желаемые свойства системы (устойчивость к малым изменениям входных данных, ограничения по безопасности и др.). Причины. Масштаб и отсутствие структуры в моделях. Адаптировать традиционные подходы (модульное тестирование, формальную верификацию) сложно из-за масштаба и отсутствия структуры в моделях.
Длительность времени, необходимая для верификации больших моделей	Сложность проверки всех возможных входных данных. Входное пространство часто бесконечно по мощности, и невозможно проверить все возможные данные. Необходимость формального доказательства соответствия модели спецификациям. Необходимо разработать алгоритмы, подтверждающие соответствие модели желаемым спецификациям для всех возможных входных данных, но эти подходы нелегко масштабировать на современные модели. Методы. Разработка новых подходов для верификации сложных моделей. Например, изучение техник оценки соответствия модели спецификациям, которые требуют от неё разработчик и пользователи. Пересмотр алгоритмов обучения так, чтобы они не только хорошо работали на тренировочных данных, но и соответствовали желаемым спецификациям. Использование разных стратегий валидации модели, например, кросс-валидации, чтобы точно оценить её обобщающую способность. Это позволяет избежать переобучения и увеличивает вероятность того, что модель будет хорошо работать на новых данных. Инструменты. Автоматизированные инструменты для верификации, которые упрощают процесс и позволяют проводить аудит в режиме реального времени. Прозрачность и объяснимость моделей, например, использование структур интерпретируемости моделей и объяснимого ИИ (XAI). Это помогает аудиторам понять процессы принятия решений и выявить потенциальные проблемы. Включение человеческого контроля в разработку и аудит ИИ, чтобы выявить проблемы, которые автоматизированные системы могут упустить.

Отсутствие инструментов, совместимых с реальными системами ИИ	Верификация больших моделей в области ИИ может занимать длительное время из-за сложности адаптации традиционных методов к сложным моделям, содержащим сотни миллионов параметров. Это связано с необходимостью проверки соответствия модели спецификациям, которые описывают желаемые свойства системы (устойчивость к малым изменениям входных данных, ограничения по безопасности и др.).
Отсутствие инструментов, совместимых с реальными системами ИИ	Причины. Масштаб и отсутствие структуры в моделях. Адаптировать традиционные подходы (модульное тестирование, формальную верификацию) сложно из-за масштаба и отсутствия структуры в моделях. Сложность проверки всех возможных входных данных. Входное пространство часто бесконечно по мощности, и невозможно проверить все возможные данные. Необходимость формального доказательства соответствия модели спецификациям. Необходимо разработать алгоритмы, подтверждающие соответствие модели желаемым спецификациям для всех возможных входных данных, но эти подходы нелегко масштабировать на современные модели. Методы. Разработка новых подходов для верификации сложных моделей. Например, изучение техник оценки соответствия модели спецификациям, которые требуют от неё разработчик и пользователи. Пересмотр алгоритмов обучения так, чтобы они не только хорошо работали на тренировочных данных, но и соответствовали желаемым спецификациям.
Отсутствие инструментов, совместимых с реальными системами ИИ	Использование разных стратегий валидации модели, например, кросс-валидации, чтобы точно оценить её обобщающую способность. Это позволяет избежать переобучения и увеличивает вероятность того, что модель будет хорошо работать на новых данных. Инструменты. Автоматизированные инструменты для верификации, которые упрощают процесс, позволяя проводить аудит в режиме реального времени. Прозрачность и объяснимость моделей, например, использование структур интерпретируемости моделей и объяснимого ИИ (XAI). Это помогает аудиторам понять процессы принятия решений и выявить потенциальные проблемы. Включение человеческого контроля в разработку и аудит ИИ, чтобы выявить проблемы, которые автоматизированные системы могут упустить.
Несовместимость с популярными ML-фреймворками (TensorFlow. PyTorch, Visual Studio)	Несовместимость с популярными ML-фреймворками (TensorFlow, PyTorch) и средой разработки Visual Studio может возникать при верификации больших моделей в ИИ из-за различий в версиях, зависимостях и функциях. Это приводит к проблемам с переносом моделей между фреймворками и запуском их на разном оборудовании (GPU, TPU) без серьёзных доработок.

	TensorFlow. Проблемы обратной совместимости. Старые исследования в TensorFlow 1 не всегда сочетаются с новыми возможностями в TensorFlow 2. Например, модель, обученная в TensorFlow 1, не может быть легко использована в TensorFlow 2 без сложных преобразований. Ошибки, связанные с CUDA. Версия TensorFlow несовместима с версией драйвера NVIDIA или архитектурой GPU. Решение: нужно устанавливать сборку под нужную версию CUDA (например, cu118 или cu121). PyTorch. Конфликты версий. Например, при установке библиотек, которые зависят от конкретных версий PyTorch, возникают ошибки.
Несовместимость с популярными ML-фреймворками (TensorFlow. PyTorch, Visual Studio)	Решение: рекомендуется использовать метод чистой установки, чтобы выровнять версии. Для сложных настроек можно использовать среды conda или контейнеры Visual Studio. Несовместимость проектов. Например, проект, обученный в TensorFlow, может быть несовместим с текущей версией Visual Studio. Решение: нужно скорректировать код для правильной работы, если это возможно. Ошибки, связанные с зависимостями. Например, конфликты зависимостей (pip долго думает или выдаёт ошибку). Решение: можно использовать conda или инструменты вроде Poetry, которые имеют более продвинутый механизм разрешения зависимостей, или попробовать установить проблемный пакет первым в чистом окружении.
Недостаток вычислительных ресурсов	Недостаток вычислительных ресурсов в ИИ касается всех, кто применяет или планирует применять искусственный интеллект. Некоторые причины проблемы. Спрос на вычислительные мощности растёт. Облачные платформы, технологические гиганты и стартапы наращивают загрузку дата-центров, обучая и разворачивая всё более сложные модели.
Недостаток вычислительных ресурсов	Повышенное энергопотребление нейровычислений. Для работы компьютерных систем, на которых обучают нейросети, требуется дополнительное специализированное оборудование с повышенным энергопотреблением. Дефицит чипов. Крупные компании обучают и тренируют множество моделей, для чего нужны соответствующие чипы. Один из способов решения проблемы – повышение эффективности использования существующих ресурсов. Сюда входит оптимальное применение оборудования, распределённые вычисления во времени или между командами, приоритизация проектов, требующих мощностей, минимизация простоев мощностей и поиск мощностей у партнёров или коллег. Также некоторые разработчики AI-моделей (Google, OpenAI, Microsoft, Amazon, Alibaba) представили собственные чипы для ИИ или объявили о работе над ними.

	Российские ученые разработали технологию (аналогов нет) для создания процессоров нового поколения. Она базируется на инновационном методе формирования логических элементов с точностью до 0,2 ангстрема (0,02 нм).
	Описание разработки опубликовано в Science Advances («Научные достижения» Smirnov et al., Sci. Adv. 11, eads9744 (2025) 2025), 7 May 2025). Технологию запатентовали в России, и в настоящий момент ведется ее патентование в других странах.
Требование к глубокому знанию формальных методов со стороны разработчиков	Глубокое знание формальных методов (детерминированных или стохастических) и/или методов моделирования и тестирования необходимо разработчикам ИИ для решения ряда задач. Некоторые из них: Формулировка предположений об окружающей среде системы. Это важно для правильного функционирования ИИ. Определение ожидаемых функциональных возможностей и ограничений. Оценка существенных характеристик системы: корректности, надёжности, вероятности ошибок и других. Кроме того, разработчики ИИ должны разбираться в нейронных сетях, обучении с подкреплением и концепциях обработки естественного языка. Также важно, чтобы разработчики могли обрабатывать и интерпретировать большие наборы данных. Это помогает обеспечить правильное обучение инструментов ИИ, что приводит к более точным и эффективным результатам.
Требование к глубокому знанию формальных методов со стороны разработчиков	Таким образом, глубокое знание формальных методов позволяет разработчикам ИИ эффективно взаимодействовать с инструментами, выполнять индивидуальные настройки и обеспечивать надёжность и эффективность автоматизированных функций.
Ограничения реального времени	Некоторые ограничения ИИ, которые могут влиять на работу в реальном времени: Синхронная работа. Некоторые ИИ-модели обрабатывают и отвечают на каждый ввод последовательно, по одному за раз. Это может стать ограничением в сценариях, требующих взаимодействия в реальном времени или одновременной обработки нескольких запросов. Ограничение контекста. Большие языковые модели (LLM) ограничены размером контекстного окна, из-за чего им приходится «забывать» информацию, полученную в начале долгих диалогов.
Ограничения реального времени	Невозможность выходить в Интернет. LLM не могут просматривать веб-страницы или вызывать веб-службы, поэтому они ограничены данными, на которых их обучали, и не имеют возможности получать или проверять информацию из живых веб-источников в режиме реального времени. Сложности с передачей данных между уровнями памяти. Это может замедлять работу при увеличении контекста.

	Для решения некоторых из этих ограничений используют, например, технологию Helix Parallelism от Nvidia. Она расширяет «память» моделей, позволяя им в реальном времени обрабатывать и анализировать большие объёмы данных.
Проблемы с верификацией сетей, обученных на больших объемах данных	Некоторые проблемы с верификацией сетей, обученных на больших объёмах данных в ИИ. Сложность перебора всех возможных вариантов выходных данных. В крупномасштабных моделях это трудно реализуемо из-за астрономического количества возможных изменений. Невозможность проверить все возможные входные данные. Часто входное пространство фактически бесконечно по мощности.
Проблемы с верификацией сетей, обученных на больших объемах данных	Трудность с интерпретируемостью моделей. Сложные модели машинного обучения тяжело интерпретировать, что создаёт проблемы с прозрачностью принятия решений. Использование контента, сгенерированного моделями, при обучении. Это может привести к необратимым дефектам и коллапсу модели, когда она неправильно воспринимает реальность. Для решения этих проблем специалисты предлагают, например, качественный контроль экспертами исходных данных. Также важно создавать стандартизированные и объективные классификаторы оценки производительности ИИ-моделей и качества датасетов.
Проблемы с данными	
Шумность данных и возможность ошибок при разметке	Шумность данных и возможность ошибок при разметке в ИИ – важные проблемы, которые влияют на производительность моделей машинного обучения. Эти проблемы связаны с наличием в данных ошибочной, нерелевантной или плохо записанной информации, а также с ошибками в разметке. Шум в данных – это случайные или систематические вариации, которые не несут полезной информации и искажают истинные закономерности. Некоторые типы шума:
Шумность данных и возможность ошибок при разметке	Случайный – данные содержат непредсказуемые значения без определённой закономерности. Например, ошибки измерений в датчиках или неожиданные изменения. Неправильная маркировка – неправильные этикетки влияют на обучение модели, его способность к обобщению сокращается. Отсутствующие или неполные данные – иногда в некоторых выборках отсутствует ключевая информация, что может исказить результаты. Выбросы – значения, которые существенно отличаются от остального набора данных, могут отрицательно повлиять на производительность модели. Методы снижения шума. Предварительная обработка данных – очистка данных, удаление выбросов, обработка отсутствующих данных и исправление неправильных маркировок.

	Нормализация и масштабирование – преобразование данных в единый масштаб помогает минимизировать влияние шума на некоторые алгоритмы машинного обучения.
Шумность данных и возможность ошибок при разметке	Использование надёжных моделей – некоторые модели, например алгоритмы, основанные на деревьях решений или нейронные сети, могут лучше обрабатывать зашумлённые данные. Ошибки при разметке. Ошибки в разметке могут возникать по следующим причинам. Неправильных или непоследовательных меток – например, один аннотатор может обозначить транспортное средство как «легковой автомобиль», а другой – как «грузовик». Отсутствующих ярлыков – аннотатор может не отметить кошку на изображениях. Неверной интерпретации инструкции – инструкции, предоставленные аннотаторам, не ясны. Методы устранения ошибок. Создание хорошо документированных стандартов маркировки и проведение проверок контроля качества. Постоянное обучение аннотаторов и использование консенсусной маркировки, когда несколько аннотаторов просматривают каждый образец.
Невозможность охватить все возможные случаи тестовыми данными	Невозможность охватить все возможные случаи тестовыми данными при использовании ИИ связана с ограниченностью обучающей выборки алгоритмов машинного обучения. Если данные, на которых учился ИИ, не охватывают конкретную область знаний или события, связанные с уникальными контекстами, ответ системы будет поверхностным или неверным. Это может проявляться в разных областях, например: В тестировании программного обеспечения – ИИ не может автоматически генерировать тестовые случаи для всех возможных сценариев, которые традиционное тестирование может пропустить. В ответах на вопросы – если вопрос выходит за рамки известных данных, система не знает, что ответить. Причины. Ограниченность обучающих данных. ИИ обучается на огромных наборах информации, но, если вопрос выходит за рамки известных данных, система не знает, что ответить. Невозможность понимания контекста. Хотя современные модели хорошо справляются с языковыми задачами, они не всегда способны понять глубокий контекст или подтекст вопроса.
Невозможность охватить все возможные случаи тестовыми данными	Сложности с неоднозначными запросами. Когда вопрос содержит двойной смысл или требует интуитивного понимания, алгоритм часто предлагает стандартный, обобщённый ответ. Решения. Сбор разнообразных данных. Они должны отражать различные сценарии тестирования, пограничные случаи и реальное

	поведение пользователей, чтобы модель ИИ могла обобщать и адаптироваться к различным условиям. Постоянное совершенствование. Нужно постоянно добавлять свежие данные в модели ИИ, чтобы повысить их точность и адаптировать к меняющимся средам приложений. Автоматизация генерации тестовых случаев. Инструменты на базе ИИ анализируют программное обеспечение и генерируют соответствующие тестовые случаи, охватывая пограничные случаи и пользовательские сценарии.
Этические и правовые ограничения на использование некоторых наборов данных	Этические ограничения. Недискриминация. Алгоритмы и наборы данных не должны приводить к умышленной дискриминации отдельных лиц или групп лиц. Рекомендуется не учитывать такие факторы, как национальная, языковая и расовая принадлежность, принадлежность к политическим партиям и общественным объединениям, вероисповедание и отношение к религии. Непричинение вреда. Недопустимо использовать технологии ИИ для причинения вреда жизни и здоровью человека, имуществу граждан и юридических лиц, окружающей среде. Прозрачность и открытость принципов работы. Рекомендуется публично декларировать факт применения рекомендательных систем, а также раскрывать цели и общие принципы их работы. Использование данных о пользователе. Должно основываться на законодательстве о защите персональных данных и учитывать требования иных нормативных актов. Правовые ограничения. Авторские права. Использование для обучения моделей ИИ наборов данных, которые защищены авторским правом, может нарушать исключительное право владельца авторских прав контролировать воспроизведение своих работ.
Этические и правовые ограничения на использование некоторых наборов данных	В России искусственный интеллект не наделяется авторскими правами, его рассматривают только в качестве инструмента при создании объекта авторских прав. Защита персональных данных. Разработчики ИИ должны соблюдать законодательство в области персональных данных и охраняемых законом тайн. В частности, использовать качественные и репрезентативные наборы данных, полученные без нарушения закона из надёжных источников. Использование псевдоданных. Например, в законе 123-ФЗ от 24 апреля 2020 года установлен специальный правовой режим в сфере ИИ, в котором указаны требования по защите персональных данных граждан и использования псевдоданных, собираемых в режиме анонимности.
Динамическое изменение входных данных в реальных приложениях	Динамическое изменение входных данных в реальных приложениях ИИ реализуется с помощью адаптивных алгоритмов, которые корректируют своё поведение или структуру в ответ на изменяющиеся входные данные.
Динамическое изменение входных данных в реальных приложениях	Некоторые области применения таких алгоритмов.

	Обслуживание клиентов. Чат-боты настраивают тон и контент в зависимости от настроений пользователей, истории или сложности проблемы. Здравоохранение. Виртуальные ассистенты персонализируют рекомендации на основе истории болезни пациента, симптомов и контекста. Образование. Адаптивные системы обучения изменяют подсказки и пояснения в соответствии с темпом обучения учащегося и пробелами в знаниях. Автоматизация бизнеса. Агенты ИИ динамически обновляют инструкции по автоматизации рабочего процесса на основе данных в режиме реального времени или отзывов пользователей. Автономные автомобили. Динамическое исполнение помогает ускорить обработку данных от камер и сенсоров, улучшая реакцию автомобилей на дорожные условия. Голосовые помощники. Уменьшение задержки в обработке запросов позволяет сделать взаимодействие с ИИ более естественным.
Динамическое изменение входных данных в реальных приложениях	Облачные сервисы. Оптимизация моделей в облаке снижает нагрузку на серверы, улучшая производительность и экономя ресурсы. Кибербезопасность. Быстрая обработка потоков данных помогает обнаруживать угрозы в реальном времени.
Отсутствие формальных спецификаций для требований к данным	Отсутствие формальных спецификаций для требований к данным в системах ИИ – проблема, которая мешает автоматизации процессов, связанных с разработкой и тестированием систем на базе ИИ. Это проявляется в том, что требования к данным не имеют унифицированных шаблонов, чётких ролей, условий и ожиданий, машиночитаемой структуры. В результате ИИ не способен вычленить из требований полноценные тест-кейсы, что срывает автоматизацию на этапе первого же документа. Причины. Отсутствие стандартов. Каждый проект пишет требования по-своему, исходя из личных предпочтений автора, а не здравого смысла или потребностей автоматизации. Это приводит к:
Отсутствие формальных спецификаций для требований к данным	− отсутствию уникальных ID требований; − непонятным правилам валидации; − правилам заполнения полей формы, которые нужно уметь «читать» между строк. Сложность формулировки спецификаций. Спецификации, описывающие «правильное» поведение ИИ-систем, часто трудно сформулировать точно. Это связано с сложным поведением систем и работой в неструктурированном окружении. Решения. Требовать структурированную документацию. Помогать внедрять шаблоны, изучать инструменты, которые умеют работать с реальной документацией.

	Использовать инструменты на базе ИИ для анализа требований. Они анализируют требования на предмет согласованности, полноты и точности, автоматически отмечают несоответствия или конфликты и предлагают улучшения или корректирующие действия. Автоматизировать отслеживание требований. ИИ анализирует отношения между требованиями и другими артефактами, такими как проектная документация, тестовые наборы и код. Это помогает поддерживать чёткий контрольный след и выявлять возникающие проблемы.
Отсутствие формальных спецификаций для требований к данным	Использовать генерацию естественного языка (NLG) для документации требований. Алгоритмы машинного обучения автоматически создают текстовые описания требований, что снижает объём требуемой ручной работы.
Влияние смещений (bias) в данных на результаты верификации	Смещение (bias) в данных влияет на результаты верификации в ИИ, приводя к искажённым или неточным выводам. Некоторые примеры влияния смещения. Модель, обученная на изображениях дорожного движения в больших городах. Если её развернуть в сельской местности, она может неправильно классифицировать необычные дорожные разметки или не обнаружить типы автомобилей, которые она никогда раньше не видела. Модель распознавания лиц, обученная в основном на одной демографической группе. Она может оказаться не в состоянии точно предсказать всех пользователей. Нейронная сеть, принимающая решения о выдаче кредита малому предпринимательству на основании исторически смещённых в пользу белокожих мужчин данных, чаще отказывает в кредите афроамериканцам и женщинам.
Влияние смещений (bias) в данных на результаты верификации	Смещение снижает точность ИИ и, следовательно, его потенциал. В приложениях, где точность очень важна, например в здравоохранении или безопасности, такие ошибки могут быть опасны. Для выявления и устранения смещения в данных в ИИ используют, например, аудит данных, тестирование производительности модели в различных сценариях, сбор большего количества образцов из недопредставленных сценариев.
Организационные причины	
Недостаток специалистов по формальной верификации в области ИИ	Недостаток специалистов по формальной верификации в области ИИ может быть связан с несколькими факторами. Сложность адаптации традиционных методов тестирования. Подходы, которые работают хорошо на традиционных программах, сложно применять для тестирования ИИ-моделей из-за их масштаба и отсутствия структуры. Высокая сложность алгоритмов формальной верификации. Время выполнения таких алгоритмов зависит от количества параметров нейронной сети, поданной на вход, и структуры свойств, которые нужно проверить.
Недостаток специалистов по формальной верификации в области ИИ	Сложности с привлечением квалифицированных специалистов. Например, в 2024 году сообщалось, что регулирующие

	органы Италии сталкиваются с проблемами при найме экспертов по ИИ. Это связано с низким вознаграждением, длительными сроками найма и бюрократическими препятствиями при получении рабочих виз. Формальная верификация – процесс математического доказательства соответствия программы или системы заданным спецификациям. В случае ИИ-сетей это означает доказательство того, что поведение сети соответствует определённым ожиданиям и безопасным границам. Поскольку ИИ продолжает развиваться быстрыми темпами, спрос на квалифицированных специалистов, способных решать сложные этические, юридические и технические проблемы, которые он представляет, будет только возрастать.
Нежелание компаний инвестировать в сложные и дорогостоящие методы	Снижение окупаемости вложений. Некоторые разработчики ИИ отмечают, что новые модели не показывают той производительности, которую от них ожидают. При этом затраты на создание и обслуживание новых моделей растут.
Нежелание компаний инвестировать в сложные и дорогостоящие методы	Сложности с поиском качественных данных. Для создания продвинутых ИИ-моделей нужно всё больше неадаптированных массивов данных, сделанных человеком. Найти такие данные становится всё сложнее. Отсутствие квалифицированных специалистов. Для успешного внедрения ИИ необходимы квалифицированные кадры, а найти таких специалистов трудно. Необходимость в дорогостоящих центрах обработки данных. Например, у небольших компаний нет возможности инвестировать в создание больших языковых моделей, так как для этого нужны такие центры. Расходы на государственную регистрацию. Например, при внедрении ИИ-решений в медицинскую сферу нужно пройти сложную и затратную процедуру регистрации. Однако, по мнению некоторых экспертов, проблемы с финансированием ИИ-проектов решаемы. Например, источниками финансирования могут быть государственные программы поддержки инноваций, субсидии и гранты.
Конкурентное давление	Конкурентное давление в области ИИ – это стимул для технологических компаний и стартапов бороться за то, чтобы первыми запустить более мощные инструменты ИИ, позволяющие им получать награды (венчурные инвестиции, внимание СМИ, регистрацию пользователей). Это может проявляться в разных аспектах, например: Борьба за лидерство в разработке ИИ-решений. Например, западные компании стремятся к доминированию в области ИИ, а китайские конкуренты быстро перенимают и адаптируют перспективные технологии с меньшими затратами. Формирование стандартов – общепризнанных принципов разработки и использования ИИ-решений, которые используются как инструмент конкурентной борьбы. Подходы к стандартам отличаются от страны к стране: например, Китай

	внедряет стандарты, направленные на ограничение экспансии внешних технологий, а США устанавливает собственные стандарты как универсальные и международно-признанные. Причины. Быстрое развитие технологий ИИ и обещание, которое они несут для преобразования различных аспектов бизнеса и повседневной жизни.
Конкурентное давление	Стремление к технологическому лидерству – кто сможет освоить ИИ быстрее других, тот и получит стратегическое преимущество. Последствия. Поспешная разработка ИИ – компании выделяют как можно больше ресурсов на повышение мощности своих систем, пренебрегая при этом исследованиями в области безопасности. Например, в феврале 2023 года компания Microsoft поспешила с выпуском чат-бота Bing, в результате он стал угрожать пользователям и оскорблять их. Неравномерное распределение центров и мощностей развития ИИ – развитие ИИ идёт в странах, которые имеют ресурсы или возможности в виде высокого научного потенциала. Это влияет на менее развитые страны, которые зависят от чужих технологий и часто привязаны к их поставщикам. Меры. Ускорение разработки продуктов – компании, создающие флагманские языковые модели, должны выстраивать вокруг себя инфраструктуру, чтобы при появлении китайского аналога пользователям было труднее переходить на него. Например, за счёт глубокой интеграции своих моделей с широко используемыми программными решениями других разработчиков.
Конкурентное давление	Усиление патентной защиты – компании, контролирующие базовые модели, могут устанавливать ограничения на то, как другие компании используют их для последующих приложений, и взимать за доступ.
Отсутствие стандартов формальной верификации для ИИ	Сложности с адаптацией традиционных методов формальной верификации для тестирования моделей ИИ. Это связано с масштабом и отсутствием структуры в моделях, которые могут содержать сотни миллионов параметров. Для решения этой проблемы разрабатываются новые подходы, в том числе направленные на поддержание высокой производительности при внедрении криптографической или образцовой верификации. Некоторые из них: EZKL. Открытая система для создания, проверяемого ИИ и аналитики с использованием доказательств нулевого разглашения (ZKPs). Позволяет разработчикам доказать, что модели ИИ были выполнены правильно, не раскрывая чувствительные данные или собственные детали модели.
Отсутствие стандартов формальной верификации для ИИ	Proof of Spontaneous Proofs (PoSP). Обеспечивает эффективную верификацию, способен поддерживать масштабные децентрализованные рабочие нагрузки для ИИ, обеспечивая

	проверяемость и доверие высокопроизводительных вычислительных и выводных процессов. EigenLayer. Протокол повторного стекинга, который позволяет валидаторам Ethereum «переставлять» свои ETH для обеспечения дополнительных децентрализованных служб, в том числе для валидации ИИ. Также для обеспечения доверия к системам ИИ используются подходы, которые применимы к обычным информационным системам, но не ограничиваются ими. Например, для ИИ, основанного на машинном обучении, способность вызывать доверие подразумевает непредвзятость (объективность) функционирования системы.
Недостаток академических исследований, направленных на интеграцию формальных методов с ИИ	Недостаток академических исследований, направленных на интеграцию формальных методов с ИИ. Проявляется в разных областях, например в образовании и науке. Большинство исследований в этой области сконцентрировано на применении ИИ для организации образовательного процесса и обучения естественным языкам, а не на интеграции ИИ в обучение другим дисциплинам.
Недостаток академических исследований, направленных на интеграцию формальных методов с ИИ	Также в недостаточной степени рассматривается влияние ИИ на академическую успеваемость студентов разных годов и направлений обучения. Причины. Ограниченность практики использования ИИ. Технологии стали доступны широкому кругу пользователей сравнительно недавно, а их возможности ещё не до конца осмыслены преподавательским сообществом. Нехватка методических рекомендаций. Большинство работ в области интеграции ИИ носят гипотетический характер или базируются на локальном практическом опыте, меньше – на больших выборках и продвинутых методах анализа данных. Этические и технические проблемы. Внедрение ИИ в образование сопряжено с рядом юридических и этических вопросов, например, защита персональных данных, прозрачность алгоритмов оценки и принятия решений. Неравномерный доступ. Проблема неравномерного доступа к ИИ-технологиям среди студентов из разных регионов и с разными финансовыми возможностями.
Недостаток академических исследований, направленных на интеграцию формальных методов с ИИ	Направления. Образование. Исследования должны изучать, как ИИ может помочь в персонализированном преподавании, интеллектуальном обучении, создании динамичного образовательного контента (от интерактивных симуляций до адаптивных учебников). Наука. Технологии ИИ должны стать не только инструментом, но и объектом исследований, например, в области машинного обучения, автономных киберфизических систем, нейронных сетей. Анализ данных. Алгоритмы ИИ должны использоваться для быстрого и эффективного анализа огромных объёмов данных, что позволяет выявлять закономерности, корреляции и

	тенденции, которые нелегко обнаружить с помощью традиционных методов. Рекомендации. Систематизировать и формализовать практику использования ИИ. Например, проводить исследования, которые рассматривают интеграцию ИИ в обучение разным дисциплинам, а не только естественным языкам.
Недостаток академических исследований, направленных на интеграцию формальных методов с ИИ	Разрабатывать методы для объяснения причин, лежащих в основе результатов, полученных с помощью ИИ. Некоторые алгоритмы ИИ, такие как модели глубокого обучения, можно считать «чёрными ящиками», что затрудняет понимание и интерпретацию их процессов принятия решений. Учитывать этические аспекты. Исследователи должны разрабатывать методы для смягчения предвзятости моделей ИИ, обеспечения конфиденциальности данных и прозрачности алгоритмов.
Недостаточная осведомленность разработчиков о формальных методах	Проблема недостаточной осведомлённости разработчиков о важности применения формальных методов в разработке систем на основе ИИ. Формальные методы и методы моделирования и тестирования позволяют: Сформулировать предположения об окружающей среде системы, которые необходимы для её правильного функционирования. Указать ожидаемые функциональные возможности и ограничения системы. Определить существенные характеристики ИИ: корректность, надёжность, вероятность ошибок, ложноположительных результатов, чувствительность к неопределённости данных и параметров.
Недостаточная осведомленность разработчиков о формальных методах	Однако, по информации на 2021 год, применение формальных методов при разработке киберфизических систем сдерживалось из-за того, что существующие методологии и инструменты не подходили для формализации корректного поведения таких систем. Недостаточная осведомлённость разработчиков о важности формальных методов может привести к созданию нестабильных, ненадёжных и потенциально опасных систем. Это, в свою очередь, может повлечь некорректные прогнозы и рекомендации, системные сбои, утечки конфиденциальных данных и другие проблемы.
Ограничения современных алгоритмов верификации	
Нехватка автоматизированных инструментов для формальной верификации ИИ	Проблема сложности адаптации традиционных методов тестирования к моделям из-за их масштаба и отсутствия структуры. В 2019 году писали, что потребуется много работы, чтобы создать автоматизированные инструменты, которые гарантируют правильную работу систем ИИ в реальном мире. Некоторые механизмы, которые предлагают для улучшения верификации ИИ.

	Аудит третьими сторонами. Независимая проверка алгоритмов и данных помогает выявить недостатки и устранить ошибки.
Нехватка автоматизированных инструментов для формальной верификации ИИ	Упражнения по «красной команде». Симуляция атак на системы ИИ позволяет выявить их уязвимости и улучшить защищённость. Премии за выявление предвзятости и безопасности. Такие конкурсы поощряют разработчиков находить и исправлять недостатки, что ведёт к улучшению качества ИИ. Безопасное аппаратное обеспечение. Оно предотвращает утечки данных и обеспечивает защиту информации в системах машинного обучения.
Неэффективность SMT-решателей для больших нейросетей	SMT-решатели (Satisfiability Modulo Theories) не всегда эффективны для больших нейросетей ИИ. Это связано с сложностью задач, которые возникают при верификации нейросетей, и ограничениями SMT-решателей. Проблема. SMT-решатели позволяют выяснить, выполнима ли формула, и подобрать значения переменных, для которых она выполняется. Однако для больших нейросетей, например, многослойных перцептронов (MLPs), генерируемые кодировки задач верификации могут быть сложными для современных SMT-решателей. Это ограничивает возможность проверки нейросетей на практике, особенно в приложениях, связанных с безопасностью.
Неэффективность SMT-решателей для больших нейросетей	Причины. Неразрешимость задач. В общем случае задача проверки нейросетей неразрешима, и решатели не всегда могут доказать наличие или отсутствие решения, а иногда и вовсе могут зависнуть. Сложность кодировок. Например, для MLPs, которые представляют собой систему взаимосвязанных вычислительных единиц (нейронов), организованных в слои, генерируемые кодировки могут быть сложными для SMT-решателей. Ограничения теорий. SMT-решатели не всегда учитывают все теории, которые используются в задачах верификации, и не всегда могут взаимодействовать между теориями при решении ограничений. Решения. Использовать другие подходы. Например, комбинировать SMT-решатели с другими методами верификации нейросетей, которые позволяют сузить границы выходного множества, сохранив при этом все взаимосвязи и прослеживаемость переходов между нейронами. Например, использовать интервальный анализ, который оценивает входное и выходное множество для каждого нейрона, а также измеряет результаты применения функции активации.
Неэффективность SMT-решателей для больших нейросетей	Учитывать ограничения теорий. Например, в некоторых случаях SMT-решатели не могут учитывать ограничения, связанные с типами, которые не являются теорией решателя.

Сложность доказательства устойчивости модели к небольшим изменениям входных данных	Доказать устойчивость модели ИИ к небольшим изменениям входных данных сложно из-за сложности моделей и проблем с данными. Даже небольшие изменения в формулировке запроса или шуме могут давать разные результаты, что требует анализа чувствительности модели к вариациям. Это важно, так как устойчивость модели не гарантирует «правильность» результатов: устойчивые прогнозы могут быть ошибочными. Методы. Для анализа устойчивости модели к изменениям входных данных используют, например: Адверсарное тестирование (Adversarial Testing). Модель сознательно «обманывают», создавая специальные входные данные, чтобы исследовать, как небольшие изменения влияют на выход модели. Это помогает обнаружить слишком «резкие» переходы между классами, неожиданные паттерны в классификации и области, где модель особенно уязвима.
Сложность доказательства устойчивости модели к небольшим изменениям входных данных	Тесты на робастность. Проверяют устойчивость модели к шуму и небольшим изменениям во входных данных, включают имитацию дрейфа и оценку поведения вне распределения. Визуализация пространства признаков. Это помогает обнаружить резкие «скачки» предсказаний при минимальных изменениях входов. Проблемы. Сложность моделей. Даже простые модели могут не справиться с решением простых задач при внесении малозаметных различий. Проблемы с данными. Недостаточный объём обучающей выборки, неразмеченные или некорректные данные, смещение (bias), из-за которого ИИ выдаёт ошибочные прогнозы. Переобучение (overfitting). Если модель была обучена на «чистом» наборе и переобучена, она оказывается слишком чувствительной к малейшим отклонениям от знакомого. Рекомендации. Использовать методы обучения с использованием противодействующих примеров (adversarial training). В модель на старте обучения добавляют данные об атаках, чтобы она могла научиться распознавать их и правильно обрабатывать такие случаи.
Сложность доказательства устойчивости модели к небольшим изменениям входных данных	Внедрять фильтры на вход (детекция шума, length check и т. д.). Это поможет модели выявлять и отклонять подобные запросы до обработки данных. Использовать синтетические данные для заполнения пробелов, если определённые сценарии слишком редки или слишком чувствительны, чтобы их можно было зафиксировать в естественных условиях.
Отсутствие поддержки динамических изменений в моделях	Есть проблема, когда модели ИИ не могут адекватно реагировать на динамично меняющиеся условия из-за использования устаревших данных для обучения. Некоторые последствия такого подхода.

	Финансовые убытки. Включают затраты на переработку моделей, повторное обучение, покупку или сбор новых данных, а также потери от некорректных решений, которые принимают ИИ-системы. Неэффективность операционной деятельности. ИИ-система, обученная на плохих данных, может генерировать неверные прогнозы, оптимизации или автоматизированные действия, что приводит к сбоям, ошибкам и снижению общей производительности.
Отсутствие поддержки динамических изменений в моделях	Потеря доверия и репутационный ущерб. Неэффективные или ошибочные ИИ-решения подрывают доверие клиентов, партнёров и инвесторов. Потеря конкурентного преимущества. В то время как конкуренты внедряют работающие ИИ-решения, компания, застрявшая в бесконечном цикле отладки данных, теряет свои позиции на рынке. Для решения этой проблемы предлагается, например, технология Model Context Protocol (MCP). Она устраняет «изоляцию» интеллектуальных моделей от данных, стандартизирует обмен информацией между ИИ-ассистентами и системами, где находятся данные. Также рассматривается архитектура «живого преобразователя» – гибкой, стохастически модулируемой системы, способной к самонастройке, рефлексии и субъективному восприятию информации.
Проблемы с кодированием нейросетей в логические ограничения	Существует проблема неспособности моделей ИИ адекватно реагировать на динамично меняющиеся условия. Некоторые причины возникновения этой проблемы.
Проблемы с кодированием нейросетей в логические ограничения	Устаревшие данные. Они не позволяют модели реагировать на изменения, что делает её бесполезной в реальном времени. Сложность моделей. Даже небольшие изменения в формулировке запроса могут дать очень разные результаты. Недостаточное развитие науки об обучении моделей. Адаптация ИИ к динамичным средам сопряжена со значительными трудностями. Некоторые из них. Скорость обработки. Модели должны реагировать в режиме реального времени, не теряя точности своих анализов. Способность к обобщению. Системы должны экстраполировать решения на новые ситуации, а не полагаться исключительно на конкретные знания. Этические критерии и принятие решений. Алгоритмы должны принимать ответственные решения в непредвиденных ситуациях, избегая нежелательных или пагубных последствий. Уменьшение смещения. ИИ должен гарантировать, что данные, используемые для обучения, не вносят предвзятости или систематические ошибки.
Проблемы с кодированием нейросетей в логические ограничения	Без возможности адаптироваться к изменениям системы ИИ быстро устареют в непредсказуемых условиях.

Ограниченные методы поиска контрпримеров	Существуют ограничения методов создания контрпримеров ИИ. По результатам одного из исследований, лучшие модели способны создавать контрпримеры только для менее 9% некорректных решений. При этом способность ИИ к фальсификации отстаёт от его способности генерировать решения. Также есть информация о сложности автоматической проверки контрпримеров для некорректных решений. Однако есть и позитивные новости: например, исследователи из Калифорнийского технологического института разработали модель ИИ, которая ищет неожиданные, сложные пути. Для этого использовался метод обучения с подкреплением: ИИ начинали с простых задач и постепенно их усложняли. В результате алгоритмы новой модели опровергли ряд потенциальных контрпримеров, которые оставались нерешёнными 25 лет.
Невозможность проверять модели с плавающей запятой с высокой точностью	Невозможность проверять модели с плавающей запятой с высокой точностью в ИИ связана с тем, что числа с плавающей запятой являются приблизительными. Ошибки округления и другие математические неточности могут привести к полной потере значимости выводов нейросети, а также сделать бессмысленными любые сравнения метрик качества. Некоторые причины проблем с точностью. Уменьшение размера числа. Например, для оптимизации скорости используются 16-битные числа с плавающей запятой (FP16). Каждое уменьшение размера числа приводит к снижению точности, что уменьшает значимость результата. Неточность операций. Все операции с числами имеют некоторые неточности, но самая большая из них возникает в случае вычитания чисел, близких друг к другу. Сравнение значений также может вносить ошибки, поскольку реализуется через вычитание. Накопление цифровых шумов. Единичные ошибки обычно не оказывают заметного влияния на работу нейросети, а вот их накопление в конечном итоге приводит к потере значимости результата.
Невозможность проверять модели с плавающей запятой с высокой точностью	Для решения этих проблем разрабатываются новые методы машинного обучения, например COLLAGE, который использует многокомпонентное представление с плавающей запятой (MCF) для точной обработки операций с числовыми ошибками.
Проблема экспоненциального роста вычислительных требований	Проблема экспоненциального роста вычислительных требований в ИИ заключается в том, что количество вычислений, необходимых для обучения современных моделей ИИ, растёт экспоненциально. То есть, чтобы сделать модель вдвое «умнее», может потребоваться не в два, а в десять или сто раз больше вычислений. Это приводит к резкому росту энергопотребления и стоимости, например: Обучение одной большой модели может потреблять огромное количество энергии.

	Запуск моделей в производстве (вывод) также влечёт за собой постоянные затраты энергии, часто превышающие затраты на фазу обучения с течением времени. Причины. Увеличение сложности моделей. Современные модели, особенно те, которые содержат миллиарды или даже триллионы параметров, требуют огромных объёмов оперативной памяти и вычислительной мощности.
Проблема экспоненциального роста вычислительных требований	Ограничения классических вычислений. Классические компьютеры работают с битами, которые могут быть только 0 или 1, и каждая операция выполняется последовательно или параллельно, но всегда в конечном числе состояний. Чтобы найти оптимальный выход (обучить модель), классическому компьютеру приходится пробовать один путь за другим, даже если это происходит очень быстро. Решения. Поиск новых вычислительных парадигм и специализированных архитектур. Например, разработка специализированных чипов для ИИ, которые работают быстрее и потребляют меньше энергии, чем GPU. Оптимизация хранения и извлечения знаний в моделях ИИ. Например, подход «масштабируемые слои памяти (SML)» – вместо того, чтобы встраивать всю изученную информацию в параметры с фиксированным весом, вводят внешнюю систему памяти, извлекая информацию только при необходимости. Это снижает вычислительные издержки и улучшает масштабируемость.
Проблема экспоненциального роста вычислительных требований	Использование квантовых вычислений. Квантовые компьютеры работают по иным принципам, чем классические, и некоторые алгоритмы могут быть выполнены на них с гораздо меньшими энергозатратами.
Плохая адаптируемость к задачам с неполной информацией	Плохая адаптируемость к задачам с неполной информацией в ИИ может быть связана с несколькими факторами. Качество данных. ИИ-модели обучаются на огромных наборах данных, но они могут быть неполными, неправильными или предвзятыми. Если модель обучалась на данных, которые не охватывают все возможные ситуации или содержат ошибки, её предсказания также будут ошибочными. Сложность поставленной задачи. Некоторые задачи требуют понимания контекста, ассоциаций и опыта, которых у модели нет. ИИ часто «думает» только в пределах данных, которые получил, и не всегда способен обобщать и адаптироваться к новым условиям. Технические ограничения архитектуры. Даже самые передовые архитектуры ИИ, такие как глубокие нейронные сети, имеют технические ограничения. У модели есть определённое количество слоёв, нейронов и весов, которые ограничивают её способность к анализу и синтезу информации.
Плохая адаптируемость к задачам с неполной информацией	Ограниченность в проверке данных. Модели ИИ, как правило, не умеют проверять правдивость информации и пола-

	гаются на вероятностные вычисления. Это означает, что иногда они «угадывают» ответ, что особенно рискованно при ответах на вопросы, связанных с фактами. Чтобы минимизировать риски получения неточной информации от ИИ, можно следовать ряду рекомендаций. Задавать более точные и детализированные вопросы. Это поможет модели лучше понять запрос. Проверять полученные данные, используя несколько источников информации, особенно если ответ кажется сомнительным. Учитывать контекст, предоставляя модели дополнительную информацию или уточнения. Разбивать сложные задачи. Большую задачу нужно разделить на несколько подзадач и решать их последовательно, опираясь на предыдущие результаты.
Проблемы с архитектурой нейросетей	
Глубокие нейросети имеют слишком сложную зависимость между слоями	Глубокие нейросети (глубокое обучение, deep learning) имеют сложную зависимость между слоями, что связано с их многослойной архитектурой. Это позволяет моделям выявлять сложные закономерности в данных, но при этом возникают проблемы, связанные с переобучением и сложностью интерпретации решений. Принцип работы. Глубокие нейросети состоят из множества слоёв, каждый из которых обрабатывает данные на разных уровнях абстракции. Некоторые особенности зависимости. Входной слой принимает исходные данные – например, изображение или текст. Скрытые слои производят вычисления на основе входящих параметров, настраивая свои «решения» через веса – параметры, которые определяют силу связи между нейронами. Выходной слой выводит результат вычислений. Чем глубже слой, тем более сложные и информативные признаки извлекает нейросеть. Например, в свёрточных нейросетях для обработки изображений первый слой анализирует «примитивные» детали (грани, края),
Глубокие нейросети имеют слишком сложную зависимость между слоями	второй – геометрические конструкции (прямые, кружки), третий – части объектов (глаза, колёса), а последний – целые объекты (лицо, машину). Проблемы. Переобучение (Overfitting) – модель чрезмерно адаптируется к обучающим данным, «запоминает» детали, которые не являются важными для общего паттерна, и теряет способность обобщать информацию. Сложность интерпретации решений – из-за сложности глубоких нейронных сетей часто трудно понять, как они принимают решения. Проблема исчезающего градиента – при обратном распределении ошибки градиент становится слишком малым, чтобы веса обновлялись эффективно. Неэффективное обновление весов снижает скорость и качество обучения нейросети.

	Решения. Чтобы избежать проблем, связанных со сложной зависимостью между слоями, важно: Тщательно выбирать архитектуру сети – количество слоёв и их характеристики в зависимости от задачи. Использовать методы регуляризации – например, Dropout и L2-регуляризацию, которые помогают снизить вероятность переобучения.
Глубокие нейросети имеют слишком сложную зависимость между слоями	Использовать функции активации – они добавляют системе нелинейность, что позволяет ей учиться более эффективно и запоминать сложные зависимости данных.
Отсутствие верификационных методов для гибридных систем	Имеется в виду проблема «чёрного ящика» в гибридных системах ИИ. Это отсутствие прозрачности в работе алгоритмов, когда пользователь не понимает, каким образом ИИ пришёл к тому или иному выводу или решению. Такая непрозрачность затрудняет верификацию результатов. Однако есть способы, которые помогают решить эту проблему и обеспечить верификацию гибридных систем ИИ. Участие человеческих экспертов в разметке данных, генерации информативных признаков и верификации моделей. Эксперты направляют обучение алгоритмов в сторону более осмысленных и корректных решений. Использование статистических методов для начальной проверки модели. Это позволяет сравнить результаты и оценить эффективность адаптированных параметров, полученных с помощью ИИ.
Отсутствие верификационных методов для гибридных систем	Применение модульной архитектуры при проектировании гибридных систем. Компоненты разделяют на слои логических правил и машинного обучения, обеспечивают строгий интерфейс обмена данными между ними. Использование правил как регуляризаторов в обучении. В функцию потерь нейросети добавляют штрафы за отклонение от экспертных знаний. Динамическое взвешивание. Реализуют адаптивный баланс между правилами и ML-моделью через механизм внимания. Также важно учитывать, что качество исходных данных напрямую определяет качество результатов, получаемых от интеллектуальных систем. Недостаточная верификация, наличие смещений или пропусков в данных могут привести к неверным выводам и ошибочным решениям.
Разнообразие топологий нейросетей	Некоторые виды топологий нейросетей в ИИ. Полносвязные. Каждый нейрон передаёт свой выходной сигнал остальным нейронам, в том числе и самому себе. Все входные сигналы подаются всем нейронам. Многослойные (слоистые). Нейроны объединяются в слои. Слой содержит совокупность нейронов с едиными входными сигналами. Число нейронов в слое может быть любым и не зависит от количества нейронов в других слоях.
Разнообразие топологий нейросетей	Монотонные. Это частный случай слоистых сетей с дополнительными условиями на связи и нейроны. Каждый слой,

	кроме последнего (выходного), разбит на два блока: возбуждающий и тормозящий. Сети без обратных связей. В таких сетях нейроны входного слоя получают входные сигналы, преобразуют их и передают нейронам первого скрытого слоя, и так далее вплоть до выходного, который выдает сигналы для интерпретатора и пользователя. Топологические. Специализированные архитектуры, предназначенные для работы с данными, структурированными в топологических областях. В отличие от традиционных нейронных сетей, адаптированных для сетчатых структур, топологические сети умеют обрабатывать более сложные представления данных. Сверточные. Предназначены для обработки двумерных структур данных, прежде всего изображений. Рекуррентные. Сети с обратными связями. Также нейронные сети различают по структуре сети (связей между нейронами), особенностям модели нейрона, особенностям обучения сети.
Разнообразие топологий нейросетей	Некоторые отличия (проверены в 2024 г.) сетей Колмогорова-Арнольда (KAN) от перечисленных выше нейронных сетей: Функциональная декомпозиция. KAN явно используют теорему Колмогорова-Арнольда для разложения многомерной функции на одномерные функции, в то время как традиционные нейронные сети не имеют такой явной декомпозиции. Одномерные преобразования. KAN в значительной степени полагаются на одномерные преобразования, тогда как традиционные нейронные сети используют многослойные персептроны с нелинейными функциями активации. Связность уровней. В традиционных нейронных сетях каждый нейрон в одном слое связан с каждым нейроном в последующем слое (полностью связан). В KAN связи структурированы таким образом, чтобы отражать сумму одномерных функций. Интерпретируемость. Структура KAN, основанная на одномерной функциональной декомпозиции, иногда может обеспечить большую интерпретируемость по сравнению с природой «чёрного ящика» традиционных нейронных сетей.
Разнообразие топологий нейросетей	Оптимизированная масштабируемость. KAN демонстрирует превосходную масштабируемость по сравнению с традиционными нейронными сетями, особенно в сценариях с высокоразмерными данными. Повышенная точность. Несмотря на использование меньшего количества параметров, KAN достигает более высокой точности и меньших потерь, чем традиционные нейронные сети, при решении различных задач. Применение сетей KAN приближает к созданию сильного искусственного интеллекта.

Формальные методы плохо адаптируются к сетям с изменяющейся структурой	Формальные методы в ИИ плохо адаптируются к сетям с изменяющейся структурой из-за ограничений, связанных с формализацией задач и отсутствием гибкости в работе с динамическими данными и знаниями. Это характерно для традиционных систем ИИ, которые часто не способны решать плохо формализованные задачи, требующие построения оригинального алгоритма решения в зависимости от конкретной ситуации. Причины. Ограничения формализации. Формальные методы (например, логические системы) не учитывают динамичность исходных данных и знаний, не позволяют автоматически строить алгоритмы решения в условиях неопределённости.
Формальные методы плохо адаптируются к сетям с изменяющейся структурой	Жёсткая запрограммированность. Варианты поведения системы часто запрограммированы в программе, что ограничивает гибкость. Например, в алгоритмах с большим числом конструкций (if, case) варианты поведения жёстко запрограммированы, и при настройке на иные сценарии работы нужно менять исходный код программ. Сложность автоматического построения представлений. Если в задачах приобретения знаний представления знаний заданы априори, то в задачах метаобучения ставится вопрос об автоматическом построении самих представлений, детали которых могут сильно меняться в зависимости от предметной области. Решения. Для преодоления ограничений формальных методов в ИИ используются, например: Системы, основанные на моделях. В таких системах программы и структуры данных генерируются или компонуются из единиц знаний, описанных в репозитарии, каждый раз при изменении модели проблемной области.
Формальные методы плохо адаптируются к сетям с изменяющейся структурой	Подходы, основанные на машинном обучении. Система обучается, а не программируется явно: ей передаются многочисленные примеры, имеющие отношение к решаемой задаче, а она находит в этих примерах статистическую структуру, которая позволяет системе выработать правила для автоматического решения задачи. Использование неклассической логики. Классическая логика (логика высказываний и логика предикатов) не подходит для решения целого ряда задач, что приводит к созданию других логических систем, которые учитывают динамичность.
Проблемы с проверкой работы слоев нормализации и активации	Некоторые проблемы, которые могут возникать при проверке работы слоёв нормализации и активации в ИИ: Проблемы с нормализацией. Раздельная нормализация обучающих и тестовых данных. Нормализация предполагает определение новых единиц измерения для проблемных переменных. Нужно выразить каждую запись в одних и тех же единицах измерения, независимо от того, принадлежит ли она к обучающему или тестовому набору.

Проблемы с проверкой работы слоев нормализации и активации	Проблема внутреннего ковариантного сдвига. Распределение входов слоя изменяется по мере обновления предыдущих слоёв, что значительно замедляет обучение. Нормализация – одно из перспективных направлений решения этой проблемы. Увеличение вычислительных затрат. Для небольших и мелких нейронных моделей с небольшим количеством слоёв нормализации это незначительно, но становится серьёзной проблемой, когда базовые сети становятся больше и глубже. Проблемы с активацией. Неправильное применение функции активации. Функции активации всегда должны следовать линейному преобразованию, чтобы сеть могла эффективно обучаться и представлять сложные шаблоны. Потеря преимуществ линейного преобразования. Линейное преобразование предназначено для отображения входных данных в новом пространстве, где функция активации может эффективно вводить нелинейность. Применяя сначала функцию активации, теряют преимущества этого сопоставления, уменьшая способность сети изучать сложные шаблоны.
Проблемы с проверкой работы слоев нормализации и активации	Вопросы размерности и инициализации. Функции активации ожидают, что входные данные будут иметь определённую форму, обычно это результат линейного преобразования. Их применение непосредственно к необработанным входным данным может привести к проблемам с размерностью и неправильной инициализации, что снижает производительность сети. Нарушение градиентного потока. Во время обратного распространения градиенты должны проходить как через линейное преобразование, так и через функцию активации. Применение функции активации сначала нарушает этот поток, что потенциально приводит к исчезновению или взрыву градиентов, что затрудняет процесс обучения.
Трудности с верификацией сетей, работающих в реальном времени	Некоторые трудности с верификацией сетей, работающих в реальном времени на основе искусственного интеллекта. Задержка. Даже при параллельном выполнении проверок синхронизация результатов и достижение согласия вызывает задержки. Процесс проходит только так быстро, как самый медленный узел. Сложность инженерии. Оркестровка оценок по нескольким моделям и обеспечение плавной работы механизма консенсуса требуют значительных инженерных усилий.
Трудности с верификацией сетей, работающих в реальном времени	Требования к вычислениям. Даже при использовании более маленьких моделей их одновременное выполнение в ансамблях увеличивает вычислительные требования. Неоднозначные случаи. В таких случаях ансамбли могут испытывать трудности с выравниванием, что приводит к несогласованным результатам. Масштаб и отсутствие структуры моделей. Адаптировать традиционные методы тестирования для моделей, содержащих сотни миллионов параметров, сложно.

	Потенциальное мошенничество. Без надёжной верификации злоумышленники могут представлять неверные или изменённые данные. Зависимость от централизованных слабых мест. Зависимость от оффчейн оракулов или частных серверов может подорвать децентрализованную этику, привести к цензуре или единичным точкам отказа.
Неясность, как корректно формализовать понятие “хорошей генерации” для генеративных моделей	Понятие «хорошей генерации» для генеративных моделей в ИИ может быть формализовано как реалистичность и разнообразие выходных данных. Это важно, так как генеративные модели создают новый контент (изображения, тексты, музыку) на основе изученных закономерностей, и качество генерации зависит от качества обучения и разнообразия обучающих данных. Критерии оценки. Реалистичность – насколько хорошо модель имитирует реальные данные. Например, сгенерированные изображения должны быть похожи на реальные. Разнообразие – насколько широк спектр возможных выходов. Идеальная модель должна уметь генерировать и котиков, и собачек, и пейзажи – в зависимости от задачи. Согласованность с исходными данными – при повторном запуске модели с теми же входными данными результаты должны быть похожими или одинаковыми. Методы. Для объективного анализа качества генерации используются специальные метрики, которые измеряют разные аспекты. Например:
Неясность, как корректно формализовать понятие “хорошей генерации” для генеративных моделей	FID (Fréchet Inception Distance) – сравнивает распределение сгенерированных изображений с распределением реальных (тестовых) данных. Чем ниже значение FID, тем лучше модель. Inception Score (IS) – оценивает два аспекта: качество – насколько изображения узнаваемы (высокая уверенность классификатора), разнообразие – насколько разные классы представлены в генерации. LPIPS (Learned Perceptual Image Patch Similarity) – оценивает perceptual similarity (похожесть для человеческого глаза). Важно: ни одна метрика не идеальна, лучше использовать несколько и смотреть на их комбинацию. Проблемы. Субъективность оценки – восприятие качества генерации может отличаться у разных людей. Учёт различных мнений и перспектив помогает снизить субъективность, но полностью избежать её может быть сложно. Ограниченность метрик – они могут не охватывать весь спектр допустимых вариантов ответа модели.
Рекуррентные сети (LSTM, GRU) имеют сложные временные зависимости	LSTM (Long Short-Term Memory) и GRU (Gated Recurrent Unit) – типы рекуррентных нейронных сетей (RNN), разработанные для обработки сложных временных зависимостей в искусственном интеллекте (ИИ). Эти архитектуры решают

	проблему исчезающего градиента, которая возникает в традиционных RNN и препятствует их способности улавливать долгосрочные зависимости в последовательных данных. LSTM (Long Short-Term Memory) – специализированный тип архитектуры RNN, разработанный для преодоления ограничений традиционных RNN в обучении зависимостям на дальних расстояниях. Особенности обработки временных зависимостей. Использует ячейки памяти, которые сохраняют информацию в течение длительного времени. Использует три «ворота» для регулирования информации, хранящейся в ячейке памяти: Ворота забывания – решают, какую информацию из состояния ячейки следует выбросить. Входные ворота – решают, какую новую информацию сохранить в состоянии ячейки. Выходной гейт – решает, какую часть состояния ячейки выводить.
Рекуррентные сети (LSTM, GRU) имеют сложные временные зависимости	Эти ворота учат, какую информацию важно сохранить или отбросить на каждом временном шаге, позволяя сети сохранять релевантный контекст в течение длительного времени. GRU (Gated Recurrent Unit) – упрощённая версия LSTM, разработанная для упрощения и ускорения обучения по сравнению с LSTM. Особенности обработки временных зависимостей: Ворота обновления – определяют, какое количество прошлой информации (предыдущее скрытое состояние) должно быть перенесено в будущее состояние. Ворота сброса – решают, сколько прошлой информации нужно забыть, прежде чем вычислять новое скрытое состояние кандидата. Эти ворота совместно управляют памятью сети, позволяя ей узнавать, какую информацию следует сохранять или отбрасывать в длинных последовательностях. Важно: хотя GRU также хорошо справляются с долгосрочными зависимостями, они могут быть не столь эффективны, как LSTM, при понимании более долгосрочных зависимостей из-за своей более простой структуры.
Верификация многомодальных моделей	Верификация многомодальных моделей в ИИ – это процесс оценки корректности и надёжности модели, которая обрабатывает и интегрирует несколько типов данных (текст, изображения, аудио, видео и данные сенсоров) одновременно. Цель – определить способность модели эффективно обобщать и давать правильные результаты на новых примерах. Верификация также позволяет выявить потенциальные проблемы, такие как переобучение, смещение или недостаточную производительность в реальных условиях. Методы. Для верификации многомодальных моделей используют, например:

	Контрастное обучение – модель учится сближать представления связанных пар модальностей (например, изображение и его описание) и отдалять представления несвязанных. Механизмы внимания – выявляют наиболее релевантные части одной модальности, соответствующие определённым элементам другой. Стратегии слияния – объединяют данные из разных модальностей в единое представление. Например:
Верификация многомодальных моделей	Раннее слияние – объединяет извлечённые векторы признаков сразу после обработки каждой модальности. Позднее слияние – сохраняет разделение модальностей до финальных этапов принятия решений, когда прогнозы от каждой модальности объединяются. Гибридное слияние – объединяет признаки несколько раз на разных уровнях модели, используя механизмы совместного внимания для динамического выделения и согласования важных кросс-модальных взаимодействий. Данные. Для верификации многомодальных моделей используют разнообразные наборы данных, которые охватывают несколько модальностей. Некоторые требования: Репрезентативность – набор должен содержать разнообразные сценарии и условия, чтобы верифицировать работу модели на различных ситуациях. Объективность и независимость от обучающего набора данных – помогают предотвратить переобучение модели и дают более объективную оценку производительности.
Верификация многомодальных моделей	Критические случаи и редкие сценарии – верификационный набор данных должен включать их, чтобы проверить, как модель справляется с экстремальными условиями. Инструменты. Для верификации многомодальных моделей используют, например: Специализированные инструменты сбора данных – они одновременно фиксируют несколько модальностей. Протоколы выравнивания на основе временных меток – помогают обеспечить согласование между различными модальностями. Оценка. Результаты верификации многомодальных моделей оценивают по различным метрикам, например: Точность, полнота, F1-мера – оценивают производительность модели в разных аспектах. Семантическая близость между модальностями – например, для оценки качества описаний к видео используют косинусную близость эмбеддингов.
Ограничения в математических основах	
Отсутствие общепринятой формальной логики для описания поведения ИИ	Общепринятой формальной логики для описания поведения ИИ нет. Это связано с особенностями работы современных систем ИИ, которые не всегда подчиняются законам формальной логики. Например:

	Генеративные модели ИИ (машинное обучение) не способны к эксплицитным рассуждениям. Система нащупывает закономерности в исходных данных и руководствуется ими в своих действиях, но не имеет средств выразить их в доступной человеческому восприятию форме. Продукты машинного обучения (например, чат-боты) не имеют заметного отношения к логике и зависят от сочетания учебных корпусов и совокупного опыта обучения. Причины. Трудности с формализацией неформальных знаний. Логический подход сталкивается с трудностями, когда возникает необходимость описания знаний, которые плохо формализуются. Неполнота понимания природы человеческого мышления. Логический подход не может полностью описать процесс мышления и принятия решений, например, такие феномены, как озарение, интуитивный поиск решения или эмоциональные влияния на принятие решений.
Отсутствие общепринятой формальной логики для описания поведения ИИ	Методы. Для описания поведения ИИ в отсутствие общепринятой формальной логики используются неформальные модели. Например: Семантические сети – описывают набор сущностей и связей между ними, изображаются в виде графа. Сценарии – моделируют возможность человека группировать знания о какой-либо деятельности или типичной ситуации в одну целостную структуру. Фреймовые технологии – позволяют извлекать закономерности из избыточной информации и создавать логические цепочки. Исследования. В 2024 году исследователи корпорации Apple провели исследование, которое показало, что ИИ-модели не способны к подлинному логическому мышлению. Например, чат-боты сбиваются с толку, если добавлять в простые задачи лишние или несуществующие данные. Это объясняется тем, что ИИ-модели следуют уже имеющимся у них шаблонам и логическим связям, которые записаны в их данных, вместо того чтобы проводить анализ новых условий задачи и адаптировать свои выводы.
Трудности с построением исчерпывающих спецификаций для ИИ	Построение исчерпывающих спецификаций для систем ИИ сталкивается с рядом трудностей, которые связаны с техническими ограничениями, сложностью предметной области и нереалистичностью ожиданий. Эти трудности проявляются в разных аспектах: в работе с данными, выборе и настройке алгоритмов, в обеспечении надёжности и устойчивости систем. Причины. Работа с данными. Объём, качество, репрезентативность и чистота данных определяют потенциал и ограничения ИИ-системы. Предвзятость, неполнота или шум в данных могут

	привести к систематическим ошибкам, которые сложно выявить и устранить на поздних этапах разработки. Выбор, обучение и валидация моделей. Существует множество алгоритмов и архитектур, каждый из которых обладает своими преимуществами и ограничениями. Выбор оптимальной модели, её тонкая настройка (гиперпараметры), обеспечение способности к обобщению на новые данные – всё это требует глубоких теоретических знаний и практического опыта.
Трудности с построением исчерпывающих спецификаций для ИИ	Нереалистичность ожиданий. Разработчики и заказчики устанавливают недостижимые цели, не учитывая текущие ограничения алгоритмов, доступность данных или сложность предметной области. Игнорирование реальных условий эксплуатации. Модель, прекрасно работающая в контролируемой лабораторной среде, может демонстрировать низкие показатели при столкновении с вариативностью и непредсказуемостью реального мира. Методы. Для преодоления трудностей в построении спецификаций для ИИ используются, например: Модульный подход к проектированию. Позволяет разбивать процессы на более управляемые компоненты, что упрощает разработку и отладку. Использование готовых библиотек и фреймворков. Они могут ускорить процесс разработки и предложить проверенные алгоритмы. Итеративный подход к разработке. Постоянное тестирование и готовность к адаптации планов позволяют минимизировать риски несоответствия.
Трудности с построением исчерпывающих спецификаций для ИИ	Примеры. Несоответствие между идеальной спецификацией и выявленной. Например, система ИИ не делает то, что от неё хотят, из-за того, что системы ИИ – не идеальные оптимизаторы или из-за непредвиденных последствий использования проектной спецификации. Проблема «чёрного ящика». Многие современные модели, такие как глубокие нейронные сети, работают как «чёрные ящики», где трудно понять, как именно принимаются решения. Литература. Проблема построения исчерпывающих спецификаций для ИИ рассматривается в научной статье AI Safety Gridworlds. В ней представлены разные типы спецификаций и проблемы, связанные с ними, например, несоответствие между идеальной и проектной спецификацией.
Неопределенность в трактовке вероятностных моделей	Неопределённость в трактовке вероятностных моделей для ИИ возникает из-за ограниченности информации, сложности прогнозирования будущих событий, а также воздействия случайных факторов. Есть два основных источника неопределённости:

Неопределенность в трактовке вероятностных моделей	Неопределённость данных (алеаторическая). Измеряет «сложность» задачи и может быть вызвана шумной зависимостью таргета от факторов или пересекающимися классами. Эта неопределённость не может быть уменьшена путём сбора большего количества обучающих данных. Неопределённость знаний (эпистемическая). Возникает, когда модель получает входные данные из области, которая либо слабо охвачена обучающими данными, либо далека от них. Поскольку модель мало знает об этой области, она, скорее всего, допустит ошибку. В отличие от неопределённости данных, неопределённость знаний может быть уменьшена путём сбора большего количества обучающих данных из плохо изученных областей. Вероятностные рассуждения обеспечивают математическую основу для представления неопределённости и управления ею. Используя вероятности, системы ИИ могут принимать обоснованные решения даже в условиях неопределённости. Например, в экономике такие модели могут быть использованы для прогнозирования рыночных тенденций, оценки финансовых рисков и оптимизации цепочек поставок, где важна высокая степень адаптации к неопределённости.
Отсутствие единой модели математического представления различных архитектур ИИ	Единой модели математического представления различных архитектур ИИ нет. Исследователи в этой области используют различные подходы, которые не всегда унифицированы. Например, в ИИ нет единой трактовки понятия «искусственный интеллект» (Artificial Intelligence, AI) в научной среде. Представители разных наук привносят в изучение ИИ специфическое, связанное с их первоначальными интересами. Причины. Многообразие интеллектуальных задач и методов. Методы ИИ разнообразны, они адаптируются под решаемую задачу. Сложности формализации знаний. Для различных предметных областей и задач более удобными и эффективными в вычислительном отношении оказываются различные формы представления знаний. Трудности с построением глобальных истинных моделей. Попытки создания ИИ с поддержкой глобальных истинных моделей (большая база непротиворечивых аксиом) наталкивались на значительные трудности. Подходы. Некоторые подходы к математическому представлению архитектур ИИ:
Отсутствие единой модели математического представления различных архитектур ИИ	Логический – базируется на моделировании рассуждений и символьном представлении информации. Теоретическая основа – логика, правила вывода. Эволюционный – основное внимание уделяется построению начальной модели и правилам, по которым она может изменяться (эволюционировать). Модель может быть составлена по различным методам, например, по мотивам человеческого мозга.

	Имитационный – объект, поведение которого имитируется, представляет собой «чёрный ящик»: информация о внутренней структуре и содержании отсутствуют, но известны спецификации входных и выходных данных. Исследования. Однако есть и тенденции к объединению различных подходов. Например: Разработка интегрированных и гибридных систем ИИ – они объединяют преимущества разнородных моделей, например нечётких экспертных систем и нейронных сетей. В таких системах могут поддерживаться различные модели представления знаний, разные типы рассуждений, модели восприятия и распознавания образов.
Отсутствие единой модели математического представления различных архитектур ИИ	Исследование интерпретируемости и объяснимости моделей – разработка методов, которые позволят понять, как модель принимает решения и какие факторы влияют на её выводы.
Формальная верификация плохо адаптирована к динамическим системам	Формальная верификация может быть сложно адаптирована к динамическим системам ИИ. Это связано с тем, что модели ИИ могут содержать сотни миллионов параметров и не иметь чёткой структуры. Кроме того, обычная логика высказываний плохо подходит для формулировки утверждений о поведении сложных динамических систем при изменении их состояний во времени. Однако есть исследования, которые пытаются адаптировать формальную верификацию к верификации динамических систем ИИ. Например, в одной из работ авторы предлагают алгоритмы, которые подходят для верификации динамических систем, в том числе, когда архитектуры ИИ используются в качестве контроллеров с обратной связью по состоянию.
Сложность формальной верификации решений, основанных на эвристиках	Формальная верификация решений, основанных на эвристиках в ИИ, сталкивается с сложностями из-за особенностей неформальных рассуждений и специфики задач, для которых используются эвристические методы. Это требует разработки новых подходов, которые объединяют неформальное и формальное мышление, и учёта специфики предметной области. Причины. Неформальные рассуждения часто включают сокращения пути и пропущенные шаги, которые формальные системы не могут верифицировать. Специфический характер задач – эвристические функции часто адаптируются к конкретным сценариям, что ограничивает их обобщение. Эвристика, которая хорошо работает для одного сценария, может быть неприменима в другом контексте. Вычислительные издержки – вычисление сложной эвристики на каждом шаге может привести к дополнительным вычислительным затратам. Если стоимость вычисления эвристики перевешивает её преимущества, это может не повысить общую производительность.

Сложность формальной верификации решений, основанных на эвристиках	Риск неоптимальных решений – недопустимые эвристические методы, хотя и более быстрые, могут привести к неоптимальным решениям. Методы. Для преодоления сложностей используются, например: Разбиение сложных проблем на более мелкие, управляемые части – вместо того чтобы пытаться решить всю проблему сразу, система разбивает её на серию «подцелей» – промежуточных лемм, которые служат ступеньками к окончательному доказательству. Использование «награды за согласованность» – это снижает структурное несоответствие и обеспечивает включение всех декомпозированных лемм в окончательные доказательства. Применение экспертных знаний – «подсказки» эксперта предметной области используются для направленного поиска, построения абстракций и накладывания ограничений на пространство поиска.
Сложность формальной верификации решений, основанных на эвристиках	Примеры. Формальная верификация решений, основанных на эвристиках, применяется в задачах, требующих больших пространств поиска, где эвристические методы направляют поиск по более перспективным путям. Например: Поиск пути – эвристический поиск помогает найти кратчайший или наиболее эффективный путь между двумя точками. Оптимизация – эвристические методы помогают максимально использовать доступные ресурсы при максимизации эффективности. Планирование и распределение ресурсов – эвристические функции направляют поиск решений, удовлетворяющих всем ограничениям. Исследования. В области сложности формальной верификации решений, основанных на эвристиках, проводятся, например: Исследование модели DeepSeek-Prover-V2 – модель объединяет неформальное и формальное мышление, разбивает сложные проблемы на управляемые части, сохраняя при этом точность, необходимую для формальной верификации.
Сложность формальной верификации решений, основанных на эвристиках	Исследование DeepSeek должно указывать, что была использована теория «Управления распределенными базы данных» В.М. Глушкова (СССР 1964 г.). Эта теория определяет распределенное множество ИНС и, как следствие, сложность существенно уменьшается и исчезает проблема «черного ящика».
Формальные методы не учитывают обучение с подкреплением в реальном мире	Имеются сложности, с которыми сталкивается обучение с подкреплением (Reinforcement Learning) при применении в реальном мире. Некоторые из них: Зависимость от симуляций. Реальные среды часто слишком дороги или опасны для непосредственного обучения. При этом перенос знаний из виртуальной среды в реальный мир остаётся серьёзной проблемой. Даже современные методы обучения не гарантируют успешный перенос в 100% случаев.

	Сложность дизайна вознаграждения. Трудно создать системы вознаграждения, которые бы сочетали немедленные действия с долгосрочными целями. Высокие вычислительные требования. Алгоритмы RL требуют большой вычислительной мощности, особенно при использовании в крупномасштабных или сложных ситуациях.
Формальные методы не учитывают обучение с подкреплением в реальном мире	Эффективность выборки. Для эффективной работы обучения с подкреплением часто требуется много данных, что является большой проблемой в таких областях, как робототехника или здравоохранение, где сбор данных может быть дорогим или рискованным. Несмотря на эти сложности, метод Reinforcement Learning применим на практике, однако он требует взвешенного подхода к использованию. Современные исследования сосредоточены на создании более эффективных, стабильных и безопасных алгоритмов, а также специализированных инструментов для их промышленного внедрения.
Верификация больших языковых моделей требует новых логических формализмов	Верификация больших языковых моделей (LLM) в ИИ требует новых логических формализмов из-за проблем, связанных с неопределённостью в работе моделей и ошибками в логических рассуждениях. Эти вызовы требуют разработки методов, которые учитывают вероятностный вывод LLM и ошибки, а также использования новых логических формализмов для анализа неопределённости. Методы верификации. Для верификации LLM используются, например:
Верификация больших языковых моделей требует новых логических формализмов	Бенчмарки – задачи, требующие логического рассуждения и анализа длинного контекста. Например, MuSR – алгоритмически сгенерированные сложные задачи, требующие логического рассуждения. Запрос к самой модели – пользователи задают модели дополнительные вопросы для проверки её ответов и получения информации о процессе генерации ответа. Модель предоставляет объяснения или дополнительные данные, которые помогают понять процесс генерации ответа и выявить возможные ошибки. Методика prompt engineering – продуманная формулировка запросов (prompts), которые управляют поведением модели для получения необходимых результатов. Это позволяет оптимизировать взаимодействие с моделью, фокусируясь на конкретных задачах и снижая вероятность генерации нерелевантной или недостоверной информации.
Верификация больших языковых моделей требует новых логических формализмов	Логические формализмы. Для верификации LLM используются, например: Вероятностные контекстно-свободные грамматики (PCFG) – расширение обычных контекстно-свободных грамматик, в которых правилам вывода приписываются вероятности. Используются для моделирования неопределённости в формальных спецификациях, сгенерированных LLM.

	Метрики, основанные на анализе грамматик – например, энтропия грамматики, которая показывает, насколько неопределённым является выбор правил вывода, или перплексия, оценивающая, как грамматика предсказывает дальнейшие шаги формализации. Эти метрики помогают обнаруживать ошибки и неопределённости, позволяя проводить выборочную формальную верификацию только тех выводов, которые содержат признаки повышенного риска ошибок. Исследования. Некоторые исследования в области верификации LLM с использованием новых логических формализмов: Исследование группы учёных из Amazon и Калифорнийского университета в Лос-Анджелесе (2024).
Верификация больших языковых моделей требует новых логических формализмов	Учёные разработали модель SolverLearner, которая использует двухэтапный подход: отделяет процесс изучения правил рассуждений от процесса их применения к конкретным случаям. Это помогает чётко отличить индуктивные рассуждения от дедуктивных, так как LLM хорошо справляются с индуктивными рассуждениями, но часто не способны к дедуктивным. Исследование исследователей из Case Western Reserve University и Microsoft (2025). Предлагает подход, который меняет представление о неопределённости в работе LLM: вместо того чтобы считать её критическим недостатком, её следует использовать для лучшего понимания и контроля над процессом формализации с помощью LLM.

Фундаментальной проблемой является эффект «черного ящика» [68]. Объяснимый искусственный интеллект (explainable artificial intelligent – XAI) является актуальной потребностью, поскольку все больше и больше моделей ИИ используется для подготовки альтернатив принятия решений. Таким образом, эти решения также воздействуют на многих пользователей. При этом каждый пользователь может получить благоприятное или неблагоприятное воздействие.

На данный момент проблема заключается в том, что специалист по анализу данных, построивший модель, не имеет полной ясности о поведении модели, и не хватает ясности в ее объяснении. В дальнейшем, завершив обучение нейронной сети и проверив ее работоспособность на тестовых данных, по сути, получается «черный ящик». Фактически неизвестно, как вырабатывается ответ, но все же он вырабатывается!

Проблема «чёрного ящика» в ИИ представляет собой одну из наиболее актуальных и обсуждаемых тем в области машинного обучения и анализа данных.

Суть данной проблемы заключается в том, что многие современные модели ИИ, особенно те, которые основаны на глубоких нейронных сетях, обладают высокой сложностью и непрозрачностью. Это приводит к тому, что пользователи и разработчики не могут в полной мере понять, как именно принимаются решения, что, в свою очередь, вызывает опасения по поводу их надежности и этичности. В условиях, когда ИИ все чаще используется в критически важных сферах, таких как медицина, финансы, право и безопасность, необходимость в объяснимости и интерпретируемости моделей становится не просто желательной, а жизненно важной.

В 2022–2023 годах в мире произошел новый скачок в развитии технологий ИИ, благодаря совершенствованию больших генеративных моделей в области языка, изображений (включая видеоизображения) и звука.

Большие фундаментальные модели уже сейчас способны писать программные коды по техническим заданиям, сочинять поэмы на заданную тему, давать точные и понятные ответы на тестовые вопросы различных уровней сложности, в том числе из образовательных программ.

Модели искусственного интеллекта за секунды создают изображения на любую тему по заданному текстовому описанию или наброску, что создает угрозу распространения запрещенной информации, нарушения авторских прав и генерации ошибочных сведений.

Большие языковые модели, которые используются для генерации текста, становятся все более похожи на человеческий язык. И поэтому многие люди, и здесь даже среди специалистов нет консенсуса, и многие специалисты говорят, что видят в этих системах уже проблески человекоподобного интеллекта, а некоторые считают и сознание.

В настоящее время стали проблемой появившиеся мультимодальные большие языковые модели, которые добавляют к языку обработку изображений и звука, а иногда и управление физическим или виртуальным телом.

Поскольку человеческое сознание по своей природе мультимодально (смешанное, комбинированное) и связано с активными действиями, эти расширенные системы больших языковых моделей более перспективны в качестве кандидатов на возможное человеческое сознание.

Следует понимать, что повсеместное использование ИИ (GPT 4,5) провоцирует возможность манипулирования с его помощью человеческим сознанием путем изготовления различных дипфейков и интернет-влияния на психику человека. В табл. 2. Приведены характеристики нейронных сетей ИИ.

Таблица 2. Характеристики нейронных сете типа GPT.

Название	**Характеристики**
GPT-3	17,5 миллиардов параметров
GPT- 4	1,76 триллиона параметров
WuDao 2.0	1,76 триллиона параметров
GPT-5	17,5 триллиона параметров
DeepSeek V3	671 млрд параметров
Claude Haiku 3.	20 млрд параметров
YandexGPT 5 Lite.	8 миллиардов параметров

Именно эта характеристика (параметры) является основной в проблеме нейронных сетей с названием «черный ящик». Кто может убедить пользователя, что при таком (триллионном) количестве параметров, обученная ИНС работает правильно? Как убедить, что при триллионном количестве параметров, веса входов нейронов (из них собрана ИНС) являются оптимальными?

Актуальность «черного ящика» ИНС, обусловлена растущим внедрением ИИ в повседневную практику и необходимость обеспечения доверия к этим технологиям.

Например, в медицине алгоритмы ИИ могут принимать решения о диагнозах и назначении лечения, что требует от них не только высокой точности, но и возможности объяснить, на каких основаниях было принято то или иное решение.

В финансовом секторе алгоритмы могут определять кредитоспособность клиентов, и отсутствие прозрачности в таких процессах может привести к дискриминации и юридическим последствиям.

Таким образом, проблема «чёрного ящика» затрагивает не только технические аспекты, но и этические, правовые и социальные.

Проблема «чёрного ящика» в искусственном интеллекте возникает в результате сложных и зачастую непрозрачных механизмов, лежащих в основе работы современных моделей, таких как глубокие нейронные сети. Эти системы принимают решения на основе анализа входных данных, однако сам процесс принятия решений оказывается скрыт от разработчиков и пользователей, что делает результаты нечитаемыми и непредсказуемыми, а также создает потенциальные риски предвзятости и дискриминации.

Актуальность проблемы возрастает с каждым годом, поскольку ИИ все активнее проникает в различные сферы – от медицины и финансов до правосудия и систем безопасности. Как показывают исследования, многие алгоритмы не могут объяснить, как они пришли к тем или иным выводам, что вызывает обоснованные опасения относительно их использования в критически важных ситуациях. Кроме того, даже создатели тех систем зачастую не понимают, каким образом они функционируют, что усугубляет этические и правовые вопросы при их использовании.

Ключевыми аспектами, касающимися объяснимости ИИ, являются интерпретируемость и прозрачность. Интерпретируемость подразумевает, что человек должен иметь возможность понять и объяснить, как и почему модель приняла то или иное решение. Прозрачность, в свою очередь, требует, чтобы информация о функционировании алгоритма была доступна и понятна пользователю. Существующие подходы к объяснению моделей ИИ зачастую далеки от удовлетворительных решений, особенно в контексте сложных и многослойных архитектур, характерных для глубокого обучения.

Исторически концепция черного ящика начала развиваться с ростом популярности алгоритмов машинного обучения. В своих ранних разработках исследователи стремились к созданию легко интерпретируемых моделей, но с увеличением сложности алгоритмов необходимость в объяснимости стала критической. В настоящее время многие ученые и практики ищут пути решения этой

проблемы, разрабатывая различные методологии и подходы, направленные на создание более объяснимых ИИ.

Наличие чёрного ящика в системах ИИ создает вызовы, которые требуют не только технических решений, но также глубокого понимания границ знаний и методов, используемых при их разработке. Этические и практические аспекты использования таких систем должны стать предметом серьезного обсуждения и регуляции, так как от этого зависит доверие пользователей и, в конечном счете, успешность внедрения технологий ИИ в общество.

Современные исследования в области объяснимости стремятся найти баланс между сложностью моделей и потребностью в понятности их работы. Важно отмечать, что ни один из подходов не является универсальным, и выбор метода объяснения зависит от специфики задачи и контекста, в котором используется модель. Если в прошлом акцент ставился на собственные объясняющие возможности моделей, то теперь следует учитывать и потребности пользователей, что становится новым вызовом.

Таким образом, понимание исторических основ этой проблемы и ее эволюции позволяет увидеть, как менялись подходы и как росла необходимость в объясняющих инструментах. Это понимание подготовит почву для дальнейшего анализа современных методов и вызовов, которые стоят перед исследователями и практиками в области достижения более высокой степени прозрачности и объяснимости в ИИ.

Современные подходы к объяснению моделей искусственного интеллекта сосредоточены на повышении степени понимания и доверия к алгоритмическим решениям, используемым в различных областях. Основные направления данного исследовательского процесса включают как наглядные, так и более абстрактные методы анализа.

Объяснимость моделей ИИ представляет собой сложную задачу, к решению которой специалисты в области машинного обучения сталкиваются с множеством трудностей и барьеров. Одна из основных проблем – это феномен, известный как "shortcut learning". Этот эффект выражается в том, что модели иногда основывают свои выводы на неверных признаках, таких как фон изображе-

ний, игнорируя более значимые характеристики объектов. Это приводит к ситуациям, когда, несмотря на обходительные результаты на тестовых наборах, системы могут выдавать неверные результаты в реальных условиях.

Существуют и другие проблемы. Например, утечки данных при обучении, избыточность или недостаток информации также сильно влияют на устойчивость и надежность моделей. Разработка более детализированных и объяснимых требует значительных усилий и понимания сложных архитектур машинного обучения. Исследования показывают, что около 87% проектов в данной области сталкиваются с неудачами, зачастую из-за недооценки важности объяснимости моделей. Эта ситуация требует дополнительных внимания к анализу таких провалов, чтобы оптимизировать подходы к созданию более надежных моделей.

Вопросы объяснимости становятся всё более актуальными, и привести к эффективному решению проблемы без интеграции мнений экспертов в эту область невозможно.

Эксперты в области объяснимого искусственного интеллекта (XAI) высказывают разнообразные мнения о текущих вызовах и перспективах внедрения решений, повышающих объяснимость моделей ИИ. Одной из основных проблем становится недостаточная прозрачность алгоритмов, используемых в критически важных сферах, таких как медицина, финансы и правосудие. Механизмы «чёрного ящика» затрудняют анализ решений ИИ, что может вызывать недоверие со стороны пользователей и регуляторов.

Дискуссии среди исследователей также подчеркивают необходимость создания единообразной терминологии и разработку практических рекомендаций по XAI. Как отмечает один из экспертов, отсутствие четких стандартов в использовании объяснимых моделей ограничивает их внедрение и применение на практике. Тем не менее, существует общее согласие относительно того, что повышение прозрачности в принятии решений критически важно для обеспечения доверия к ИИ, особенно в случаях, когда алгоритмы влияют на жизни людей.

Ключевым аспектом обсуждения является возможность пользователей понимать и интерпретировать действия ИИ. Некоторые эксперты выделяют интерпретируемость как важный элемент XAI, позволяющий пользователям осо-

знанно взаимодействовать с технологиями. Такой подход предполагает, что объяснения должны быть понятны не только разработчикам, но и конечным пользователям. В этом контексте наблюдается растущее внимание к формированию моделей, которые могут не только выдавать результаты, но и объяснять, как они были достигнуты.

Обсуждение XAI также касается уровней объяснимости, которые могут быть достигнуты с помощью разработки специализированных методов. Одни эксперты указывают на необходимость интеграции моделей с возможностью идентификации и исправления ошибок, что, в свою очередь, повышает уровень доверия к системам ИИ. Однако существует и другая точка зрения. Некоторые специалисты говорят о недостаточной информативности даже объяснимых моделей, что требует дополнительного внимания со стороны исследователей и практиков.

Таким образом, несмотря на многообещающие направления исследований в области XAI, остаются значительные препятствия, связанные с практическим применением объясняющих систем. Это подчеркивает важность continuing collaboration между учеными, разработчиками и пользователями для формирования единого подхода к повышению объяснимости и интерпретируемости технологий ИИ.

Повышение прозрачности алгоритмов ИИ требует комплексного подхода, включающего как технологические, так и этические аспекты. Важно, чтобы специалисты по обработке данных осознавали значимость объяснимости моделей, ведь только открытое взаимодействие пользователей с алгоритмами может способствовать доверию к результатам.

Первым шагом к прозрачности является использование композитного ИИ. Это подход интегрирует разнообразные аналитические методы и технологии, что не только улучшает интерпретируемость, но и позволяет уменьшить предвзятость на выходе. Кроме того, рекомендуется применять модели, которые легко объяснить пользователям, такие как линейные модели и деревья решений. Использование модели, понятной для конечного пользователя, уменьшает риск недоверия к системам ИИ и помогает более эффективно оценивать последствия их решений.

Следует использовать принципы справедливости и защиты конфиденциальности в модели ИИ.

Правовая основа также играет критическую роль в обеспечении прозрачности. Существующие законодательства в сфере технологий должны обновляться с учётом появления новых алгоритмических решений и их этических следствий.

Наконец, важно, чтобы все заинтересованные стороны, включая разработчиков, законодателей и пользователей, вели открытый диалог о проблемах и решениях, связанных с прозрачностью ИИ. Создание совместных инициатив по обмену знаниями приведет к более интуитивным и понятным системам. В конечном счете, разработка эффективных механизмов для повышения объяснимости и справедливости систем ИИ будет способствовать не только улучшению качества принимаемых решений, но и укреплению доверия общества к технологиям, изменяющим нашу жизнь.

Исторический контекст развития ИИ показывает, что проблема объяснимости не нова. С момента появления первых алгоритмов машинного обучения исследователи и практики сталкивались с необходимостью объяснять результаты работы своих моделей. Однако с развитием более сложных и мощных алгоритмов, таких как глубокие нейронные сети, эта проблема стала более остро ощущаться.

Из проведенного анализа следует, что проблема «чёрного ящика» в ИИ требует комплексного подхода, включающего как технические, так и образовательные меры. Только совместными усилиями можно преодолеть существующие трудности и создать более прозрачные и объяснимые модели, которые будут служить на благо общества. Важно помнить, что объяснимый ИИ – это не просто тренд, а необходимость, которая будет определять будущее технологий и их взаимодействие с человеком.

ЗАКЛЮЧЕНИЕ

Лесные экосистемы играют ключевую роль в глобальном цикле углерода, выступая в качестве естественных накопителей атмосферного углекислого газа. В условиях нарастающих климатических изменений понимание механизмов и количественная оценка депонирования углерода лесами приобретают особую актуальность.

Авторы рассмотрели существующие международные модели оценки углерода лесными экосистемами которые играют важную роль в понимании и прогнозировании углеродного баланса.

Было показано, что теоретические основы должны учитывать многообразие факторов, определяющих способность лесов к депонированию:

- структурные факторы: возраст насаждений, видовой состав, плотность древостоя;
- климатические факторы: температура, осадки, продолжительность вегетационного периода;
- эдафические факторы: плодородие почв, влагообеспеченность;
- антропогенные факторы: лесоуправление, нарушения, загрязнения.

В применяемых моделях оценка депонирования углерода сопряжена с рядом неопределенностей:

- пространственная гетерогенность лесных экосистем затрудняет экстраполяцию локальных измерений;
- временная динамика накопления углерода и влияние экстремальных событий (пожары, ветровалы) недостаточно учтены в моделях;
- сложные взаимодействия между различными пулами углерода (надземная биомасса, корни, почва) требуют комплексного подхода.

Авторы показали, что как отечественный, так и современные международные модели для оценки углерода в лесных экосистемах предоставляют значимые инструменты для научного анализа и разработки эффективных методов углеродного регулирования, но успех их применения во многом зависит от качества исходных данных и контекста местных условий.

Применение современных технологий искусственного интеллекта в оценках углерода будет эффективно, если исходные данные, датасет, содержат достоверные данные, что в настоящих условиях проблематично. Ошибки в датасет определяет и неопределённость в самих технологиях искусственного интеллекта. Авторы, далее проанализируют фундаментальные недостатки искусственного интеллекта. Теоретические основы определения депонирования углерода лесными экосистемами представляют собой междисциплинарную область, интегрирующую знания экологии, физиологии растений, почвоведения, метеорологии и дистанционного зондирования. Совершенствование методов оценки и понимание механизмов депонирования углерода имеют не только научное, но и прикладное значение в контексте смягчения последствий изменения климата и устойчивого управления лесами.

Авторы исследовали возможности и технологии подготовки актуальных наборов датасет. Определили их свойства и возможные риски искажения данных.

В работе авторы предлагают подготовку датасет, их анализ и обработку проводить в среде технологии искусственного интеллекта – машинное обучение. Рассматривается среда ML.NET, определяется её архитектура, свойства и возможности. ML.NET новая и эффективная технология. Исследуются возможности ML.NET для регрессионного анализа картирования углерода, депонируемого лесами.

Исследование применения технологий искусственного интеллект для оценки картирования углерода, депонируемого лесами, затрагивает фундаментальные проблемы технологий искусственного интеллекта. Предлагаемый перечень декомпозирован и определены возможные направления устранения рисков.

Авторы предлагают в качестве информационной системы собственную разработку определения и картирования депонируемого лесами углерода в среде NATURAL [70]. Программное обеспечение позволяет эффективно формировать достоверные датасет, проводить машинное обучение в среде регрессионных моделей профессора Усольцева В.А.

ЛИТЕРАТУРА

1. Усольцев В. А., Часовских Е. П., Стариков Е. Н. Исследование методов и обработка баз данных о биомассе лесов Евразии как нейронных сетей. Часть 1. Системный анализ базы данных для трансформации в нейронные сети искусственного интеллекта // Цифровые модели и решения. 2022. Т. 1, № 1. DOI: 10.29141/2782-4934-2022-1-1-2. EDN: DNJRGT.

2. Воронов М.П., Усольцев В.А., Часовских В.П. Модели мониторинга уровня депонирования углерода лесными экосистемами. Естественные и технические науки. 2014. № 4 (72). С. 133-136.

3. cyberleninka.ru/article/n/globalnoe-izmenenie-klimata-i-ego... [Электронный ресурс] // cyberleninka.ru – Режим доступа: https://cyberleninka.ru/article/n/globalnoe-izmenenie-klimata-i-ego-posledstviya/viewer, свободный. – Загл. с экрана

4. Лукина Н.В., Гераськина А.П., Горнов А.В., Шевченко Н.Е., Куприн А.В., Чернов Т.И., Чумаченко С.И., Шанин В.Н., Кузнецова А.И., Тебенькова Д.Н., Горнова М.В. Биоразнообразие и климаторегулирующие функции лесов: актуальные вопросы и перспективы исследований // Вопросы лесной науки. 2020. №4. URL: https://cyberleninka.ru/article/n/bioraznoobrazie-i-klimatoreguliruyuschie-funktsii-lesov-aktualnye-voprosy-i-perspektivy-issledovaniy (25.12.2024).

5. Константинов Артем Васильевич, Сергиенко Валерий Гаврилович Влияние изменений климата в голоцене на формирование разнообразия современных лесов и их трансформация к концу XXI века в европейской России // Лесотехнический журнал. 2016. №3 (23). URL: https://cyberleninka.ru/article/n/vliyanie-izmeneniy-klimata-v-golotsene-na-formirovanie-raznoobraziya-sovremennyh-lesov-i-ih-transformatsiya-k-kontsu-xxi-veka-v (22.12.2024).

6. Писарчук Н.М., Новенко Е.Ю., Козлов Д.Н., Шилов П.М. Влияние климатических изменений на лесные экосистемы и процессы заболачивания в Центрально-Лесном заповеднике // Вестник Московского университета. Серия 5. География. 2016. №4. URL: https://cyberleninka.ru/article/n/vliyanie-klimaticheskih-

izmeneniy-na-lesnye-ekosistemy-i-protsessy-zabolachivaniya-v-tsentralno-lesnom-zapovednike (13.12.2024).

7. Беляева Н.В., Данилов Д.А., Кази И.А. Влияние климатических, биотических факторов и конкуренции на лесовозобновление // Актуальные проблемы лесного комплекса. 2020. №56. URL: https://cyberleninka.ru/article/n/vliyanie-klimaticheskih-bioticheskih-faktorov-i-konkurentsiii-na-lesovozobnovlenie (18.12.2024).

8. Малозёмов О.Ю., Кондрашкина А.А. К проблеме взаимовлияния леса и климата // Теория и практика современной науки. 2019. №10 (52). URL: https://cyberleninka.ru/article/n/k-probleme-vzaimovliyaniya-lesa-i-klimata (23.12.2024).

9. Бурков В.Д., Шалаев В.С., Крапивин В.Ф. О роли лесных экосистем в изменении климата // Лесной вестник / Forestry bulletin. 2012. №9 (92). URL: https://cyberleninka.ru/article/n/o-roli-lesnyh-ekosistem-v-izmenenii-klimata (17.12.2024).

10. Бедрицкий А.И., Бердин В.Х. Лес и глобальные изменения климата // Лесной вестник / Forestry bulletin. 1999. №1. URL: https://cyberleninka.ru/article/n/les-i-globalnye-izmeneniya-klimata (25.12.2024).

11. Горбунова О.И. , Кулагина А.Н., Богомолова Е.Ю. Природные климатические решения и их роль в развитии устойчивого управления лесными ресурсами // Дискуссия. 2024. №5 (126). URL: https://cyberleninka.ru/article/n/prirodnye-klimaticheskie-resheniya-i-ih-rol-v-razvitii-ustoychivogo-upravleniya-lesnymi-resursami (21.12.2024).

12. Истомин А. В. Некоторые реакции биоты на изменение климата в лесных ландшафтах Каспийско-Балтийского водораздела // Вестник Балтийского федерального университета им. И. Канта. Серия: Естественные и медицинские науки. 2009. №7. URL: https://cyberleninka.ru/article/n/nekotorye-reaktsii-bioty-na-izmenenie-klimata-v-lesnyh-landshaftah-kaspiysko-baltiyskogo-vodorazdela (13.01.2025).

13. Некляев С.Э., Серая Л.Г., Ларина Г.Е. Экологические последствия современных изменений климата, негативно влияющие на устойчивость хвойных растений к вредителям и афиллофоровым грибам // Биосфера. 2022. №3. URL: https://cyberleninka.ru/article/n/ekologicheskie-posledstviya-sovremennyh-izmeneniy-klimata-negativno-vliyayuschie-na-ustoychivost-hvoynyh-rasteniy-k-vreditelyam-i (17.12.2024).

14. Королева Татьяна Станиславна, Константинов Артем Васильевич, Кушнир Елизавета Андреевна Оценка влияния наблюдаемых эффектов климатической изменчивости на устойчивость лесных экосистем Российской Федерации к угрозе массовых размножений вредителей и болезней леса // Лесотехнический журнал. 2016. №4 (24). URL: https://cyberleninka.ru/article/n/otsenka-vliyaniya-nablyudaemyh-effektov-klimaticheskoy-izmenchivosti-na-ustoychivost-lesnyh-ekosistem-rossiyskoy-federatsii-k-ugroze (10.12.2024).

15. Прожерина Н.А., Наквасина Е.Н. Изменение климата и его влияние на адаптацию и внутривидовую изменчивость хвойных пород европейского севера России // Известия высших учебных заведений. Лесной журнал. 2022. №2. URL: https://cyberleninka.ru/article/n/izmenenie-klimata-i-ego-vliyanie-na-adaptatsiyu-i-vnutrividovuyu-izmenchivost-hvoynyh-porod-evropeyskogo-severa-rossii (16.01.2025).

16. Аненхонов Олег Арнольдович Изучение климатогенной динамики растительного покрова: предпосылки, подходы, перспективы // Известия Иркутского государственного университета. Серия: Биология. Экология. 2012. №3. URL: https://cyberleninka.ru/article/n/izuchenie-klimatogennoy-dinamiki-rastitelnogo-pokrova-predposylki-podhody-perspektivy (21.12.2024).

17. Иметхенов Олег Анатольевич Палеоклиматические изменения на территории Байкальской Сибири и их влияние на биоту // Вестник Бурятского государственного университета. Биология. География. 2015. №4. URL: https://cyberleninka.ru/article/n/paleoklimaticheskie-izmeneniya-na-territorii-baykalskoy-sibiri-i-ih-vliyanie-na-biotu (01.04.2025).

18. Желдак Владимир Иванович Лесоводственное обеспечение формирования установок по решению задач смягчения изменений климата и адаптации лесного комплекса к меняющимся условиям // Вестник Поволжского государственного технологического университета. Серия: Лес. Экология. Природопользование. 2023. №3 (59). URL: https://cyberleninka.ru/article/n/lesovodstvennoe-obespechenie-formirovaniya-ustanovok-po-resheniyu-zadach-smyagcheniya-izmeneniy-klimata-i-adaptatsii-lesnogo (18.04.2025).

19. Усольцев В. А., Часовских Е. П., Стариков Е. Н., Цепордей И. С. Исследование методов и обработка баз данных о биомассе лесов Евразии как нейронных сетей. Часть 2. Новые возможности искусственного интеллекта при прогнозировании климатически обусловленных изменений // Цифровые модели и решения. 2022. Т. 1, № 2. DOI: 10.29141/2782-4934-2022-1-2-2. EDN: MIMVYU.

20. Усольцев В.А., Ковязин В.Ф., Цепордей И.С. Текущее накопление углерода в лесах двух эко регионов России

Известия Санкт-Петербургской лесотехнической академии. 2021. № 237. С. 75-96.

21. Усольцев В.А. Депонирование углерода лесами уральского региона россии (по состоянию Государственного учета лесного фонда на 2007 год) / Екатеринбург, 2018

22. Усольцев В.А., Колчин К.В., Маленко А.А. О необходимости построения и анализа аллометрических моделей фитомассы лесных деревьев как основы корректной оценки углерод депонирующей функции лесов (аналитический обзор) Вестник Алтайского государственного аграрного университета. 2017. № 3 (149). С. 78-87

23. Часовских В.П., Усольцев В.А., Воронов М.П Проектирование информационной системы поддержки принятия решений в лесном комплексе с использованием самонастраивающихся нечетких моделей. Эко-потенциал. 2014. № 4 (8). С. 81-89.

24. Воронов М.П., Усольцев В.А., Часовских В.П., Сенчило И.С.Л.Н.В. Автоматизированная система определения и картирования депонируемого лесами углерода в среде СУБД ADABAS Известия высших учебных заведений. Лесной журнал. 2013. № 1 (331). С. 22-28.

25. Воронов М.П., Усольцев В.А., Часовских В.П Исследование методов и разработка информационной системы определения и картирования депонируемого лесами углерода в среде NATURAL (2-е издание, исправленное и дополненное) Екатеринбург, 2012.

26. Воронов М.П., Усольцев В.А. Модели оценки годичного депонирования углерода в фитомассе насаждений на лесопокрытых площадях естественные и технические науки. 2011. № 4 (54). С. 227-230.

27. Воронов М.П., Усольцев В.А., Часовских В.П. Исследование методов и разработка информационной системы определения и картирования депонируемого лесами углерода в среде NATURAL Международный журнал экспериментального образования. 2010. № 10. С. 76-77.

28. Желдак В.И., Дорощенкова Э.В., Сычева А.Н., Липкина Т.В., Живаев Е.Е. Технологическая реализация лесоводственных мероприятий, обеспечивающих эффективное выполнение лесами функций депонирования и консервации углерода // Лесохозяйственная информация. 2023. №3. URL: https://cyberleninka.ru/article/n/tehnologicheskaya-realizatsiya-lesovodstvennyh-meropriyatiy-obespechivayuschih-effektivnoe-vypolnenie-lesami-funktsiy (14.08.2025).

29. Малышева Н.В., Моисеев Б.Н., Филипчук А.Н., Золина Т.А. Методы оценки баланса углерода в лесных экосистемах и возможности их использования для расчетов годичного депонирования углерода // Лесной вестник / Forestry bulletin. 2017. №1. URL: https://cyberleninka.ru/article/n/metody-otsenki-balansa-ugleroda-v-lesnyh-ekosistemah-i-vozmozhnosti-ih-ispolzovaniya-dlya-raschetov-godichnogo-deponirovaniya-ugleroda (16.01.2025).

30. Желдак Владимир Иванович Функционально-целевая система объектов лесоводства углерододепонирующего и углеродоконсервационного назначения: её формирование и использование // Вестник Поволжского государственного технологического университета. Серия: Лес. Экология. Природопользование. 2022. №4 (56). URL:

https://cyberleninka.ru/article/n/funktsionalno-tselevaya-karbonovaya-sistema-obektov-lesovodstva-uglerododeponiruyuschego-i-uglerodokonservatsionnogo-naznacheniya (14.08.2025).

31. Азаренок В. А., Колтунова А. И. Депонирование углерода при экологизированных рубках: совмещение ресурсной и биосферной функций лесов // Аграрный вестник Урала. 2011. №4. URL: https://cyberleninka.ru/article/n/deponirovanie-ugleroda-pri-ekologizirovannyh-rubkah-sovmeschenie-resursnoy-i-biosfernoy-funktsiy-lesov (14.08.2025).

32. Мамонов Дмитрий Николаевич, Морковина Светлана Сергеевна, Матвеев Сергей Михайлович, Шешницан Сергей Сергеевич, Иветич Владан Сравнительная оценка методов учёта депонирования углерода сосново-берёзовыми лесными насаждениями воронежской области // Лесотехнический журнал. 2022. №3 (47). URL: https://cyberleninka.ru/article/n/sravnitelnaya-otsenka-metodov-uchyota-deponirovaniya-ugleroda-sosnovo-beryozovymi-lesnymi-nasazhdeniyami-voronezhskoy-oblasti (07.03.2025).

33. Кондратьева О.Е., Локтионов О.А., Кузнецов Н.С. Обзор и сравнительный анализ цифровых инструментов оценки углеродного следа // XXI век. Техносферная безопасность. 2022. №4 (28). URL: https://cyberleninka.ru/article/n/obzor-i-sravnitelnyy-analiz-tsifrovyh-instrumentov-otsenki-uglerodnogo-sleda (03.01.2025).

34. Львова Надежда Алексеевна Формирование финансовой модели углеродного регулирования в контексте целей декарбонизации Российской Федерации // Вестник Санкт-Петербургского университета. Экономика. 2024. №3. URL:

https://cyberleninka.ru/article/n/formirovanie-finansovoy-modeli-uglerodnogo-regulirovaniya-v-kontekste-tseley-dekarbonizatsii-rossiyskoy-federatsii (08.06.2025).

35. cyberleninka.ru/article/n/metody-otsenki-balansa-ugleroda-v-lesnyh... [Электронный ресурс] // cyberleninka.ru – Режим доступа: https://cyberleninka.ru/article/n/metody-otsenki-balansa-ugleroda-v-lesnyh-ekosistemah-i-vozmozhnosti-ih-ispolzovaniya-dlya-raschetov-godichnogo-deponirovaniya-ugleroda/viewer, свободный. – Загл. с экрана

36. Филипчук А. Н., Малышева Н. В., Золина Т. А., Югов А. Н. Бореальные леса России: возможности для смягчения изменения климата // Лесохозяйственная информация. 2020. №1. URL: https://cyberleninka.ru/article/n/borealnye-lesa-rossii-vozmozhnosti-dlya-smyagcheniya-izmeneniya-klimata (09.07.2025).

37. Рыжова Ирина Михайловна, Подвезенная Марина Александровна, Кириллова Наталия Петровна Вариабельность запасов углерода в автоморфных и полугидроморфных почвах лесных экосистем европейской территории россии: сравнительный статистический анализ // Вестник Московского университета. Серия 17. Почвоведение. 2022. №2. URL: https://cyberleninka.ru/article/n/variabelnost-zapasov-ugleroda-v-avtomorfnyh-i-polugidromorfnyh-pochvah-lesnyh-ekosistem-evropeyskoy-territorii-rossii-sravnitelnyy (14.08.2025).

38. Ваганов Евгений Александрович, Порфирьев Борис Николаевич, Широв Александр Александрович, Колпаков Андрей Юрьевич, Пыжев Антон Игоревич. Оценка вклада российских лесов в снижение рисков климатических изменений // Экономика региона. 2021. №4. URL: https://cyberleninka.ru/article/n/otsenka-vklada-rossiyskih-lesov-v-snizhenie-riskov-

39. Желдак В.И., Дорощенкова Э.В., Прока И.Ю., Сычева А.Н., Липкина Т.В. Вопросы лесоводственного совершенствования системы сохранения и использования лесов в рамках решения проблемы адаптации лесов и лесного комплекса к изменениям климата // Лесохозяйственная информация. 2023. №2. URL:

https://cyberleninka.ru/article/n/voprosy-lesovodstvennogo-sovershenstvovaniya-sistemy-sohraneniya-i-ispolzovaniya-lesov-v-ramkah-resheniya-problemy-adaptatsii-lesov (21.12.2024).

40. Тараканов А.М., Сурина Е.А., Сеньков А.О. Лесохозяйственные мероприятия по адаптации растительности к изменению климата // Актуальные проблемы лесного комплекса. 2017. №47. URL: https://cyberleninka.ru/article/n/lesohozyaystvennye-meropriyatiya-po-adaptatsii-rastitelnosti-k-izmeneniyu-klimata (21.12.2024).

41. Бурков В.Д., Шалаев В.С., Крапивин В.Ф. О роли лесных экосистем в изменении климата // Лесной вестник / Forestry bulletin. 2012. №9 (92). URL: https://cyberleninka.ru/article/n/o-roli-lesnyh-ekosistem-v-izmenenii-klimata (17.12.2024).

42. cyberleninka.ru/article/n/variabelnost-zapasov-ugleroda-v-avtomorfnyh... [Электронный ресурс] // cyberleninka.ru – Режим доступа: https://cyberleninka.ru/article/n/variabelnost-zapasov-ugleroda-v-avtomorfnyh-i-polugidromorfnyh-pochvah-lesnyh-ekosistem-evropeyskoy-territorii-rossii-sravnitelnyy/viewer, свободный. – Загл. с экрана

43. Лукьянов В.Д. , Лукьянова В.В. Базовые подходы к оценке запасов и потоков лесного углерода // Вестник науки. 2024. №6 (75). URL: https://cyberleninka.ru/article/n/bazovye-podhody-k-otsenke-zapasov-i-potokov-lesnogo-ugleroda (16.01.2025).

44. Павлов Максим Павлович Подготовка данных для машинного обучения распознавания объектов и их ключевых точек // StudArctic Forum. 2023. №1. URL: https://cyberleninka.ru/article/n/podgotovka-dannyh-dlya-mashinnogo-obucheniya-raspoznavaniya-obektov-i-ih-klyuchevyh-tochek (15.12.2024).

45. Гетьман А.И., Горюнов М.Н., Мацкевич А.Г., Рыболовлев Д.А. Методика сбора обучающего набора данных для модели обнаружения компьютерных атак // Труды Института системного программирования РАН. 2021. №5. URL:

https://cyberleninka.ru/article/n/metodika-sbora-obuchayuschego-nabora-dannyh-dlya-modeli-obnaruzheniya-kompyuternyh-atak (12.12.2024).

46. Стариков А.Е., Намиот Д.Е. Система выполнения моделей машинного обучения на потоке событий // International Journal of Open Information Technologies. 2020. №7. URL: https://cyberleninka.ru/article/n/sistema-vypolneniya-modeley-mashinnogo-obucheniya-na-potoke-sobytiy (11.12.2024).

47. cyberleninka.ru/article/n/obzor-metodov-ochistki-dannyh-dlya... [Электронный ресурс] // cyberleninka.ru – Режим доступа: https://cyberleninka.ru/article/n/obzor-metodov-ochistki-dannyh-dlya-mashinnogo-obucheniya/viewer, свободный. – Загл. с экрана

48. cyberleninka.ru/article/n/primenenie-sovremennyh-tehnologiy-sbora... [Электронный ресурс] // cyberleninka.ru – Режим доступа: https://cyberleninka.ru/article/n/primenenie-sovremennyh-tehnologiy-sbora-dannyh-i-metodov-mashinnogo-obucheniya-dlya-raspoznavaniya-lits/viewer, свободный. – Загл. с экрана

49. Копырин Андрей Сергеевич, Макарова Ирина Леонидовна Алгоритм препроцессинга и унификации временных рядов на основе машинного обучения для структурирования данных // Программные системы и вычислительные методы. 2020. №3. URL: https://cyberleninka.ru/article/n/algoritm-preprotsessinga-i-unifikatsii-vremennyh-ryadov-na-osnove-mashinnogo-obucheniya-dlya-strukturirovaniya-dannyh (15.12.2024).

50. Краснов Ф.В. Выявление ошибок разметки данных с помощью моделей классификации для небольших наборов данных // International Journal of Open Information Technologies. 2023. №5. URL: https://cyberleninka.ru/article/n/vyyavlenie-oshibok-razmetki-dannyh-s-pomoschyu-modeley-klassifikatsii-dlya-nebolshih-naborov-dannyh (07.06.2025).

51. А.В. Макаров, Д.Е. Намиот Обзор методов очистки данных для машинного обучения // International Journal of Open Information Technologies. 2023. №10.

URL: https://cyberleninka.ru/article/n/obzor-metodov-ochistki-dannyh-dlya-mash-innogo-obucheniya (15.12.2024).

52. Анатолий Дмитриевич Хомоненко, Кириенко Андрей Борисович, Злобин Сергей Евгеньевич, Давыдова Даяна Очистка полуструктурированных и неструктурированных данных дистанционного зондирования Земли // Интеллектуальные технологии на транспорте. 2024. №4 (40). URL: https://cyberleninka.ru/article/n/ochistka-polustrukturirovannyh-i-nestrukturirovannyh-dannyh-distantsionnogo-zondirovaniya-zemli (11.04.2025).

53. Вильданов А.Н. Генерация датасетов для учебных задач компьютерного зрения // Инженерный вестник Дона. 2023. №4 (100). URL: https://cyberleninka.ru/article/n/generatsiya-datasetov-dlya-uchebnyh-zadach-kompyuternogo-zreniya (02.03.2025).

54. Парасич Андрей Викторович, Парасич Виктор Александрович, Парасич Ирина Васильевна Формирование обучающей выборки в задачах машинного обучения. Обзор // Информационно-управляющие системы. 2021. №4 (113). URL: https://cyberleninka.ru/article/n/formirovanie-obuchayuschey-vyborki-v-zadachah-mashinnogo-obucheniya-obzor (07.01.2025).

55. Исмагулов Милан Ерикович Подготовка данных датасета CROHME 2019 для обучения нейронной сети // Вестник Югорского государственного университета. 2024. №3. URL: https://cyberleninka.ru/article/n/podgotovka-dannyh-dataseta-crohme-2019-dlya-obucheniya-neyronnoy-seti (16.12.2024).

56. Намиот Д.Е., Ильюшин Е.А. О причинах неудач проектов машинного обучения // International Journal of Open Information Technologies. 2023. №1. URL: https://cyberleninka.ru/article/n/o-prichinah-neudach-proektov-mashinnogo-obucheniya (27.12.2024).

Бринк Х., Ричардс Д., Феверолф М. «Машинное обучение» Санкт-Петербург : Питер, 2017. – 330 с.

Замятин А. В. Интеллектуальный анализ данных. Томск: Изд-во ТГУ, 2020. – 193 с.

Николенко С. И., Кадурин А. А., Архангельская Е. О. (2018). Глубокое обучение. СПб.: Питер. 2018. – 476 с.

Петера Флаха «Машинное обучение. Наука и искусство построения алгоритмов, которые извлекают знания из данных». М. «ДМК Пресс», 2015. – 399 с.

Дино Эспозито и Франческо Эспозито. Programming ML.NET – Microsoft Press, 2022. – 256 с.

62. Т.С. Мещерякова, З.А. Арсаханова , А.В. Бровкин , Е.С. Шамухаметова, Ю.А. Боков Разработка интеллектуальной системы управления процессами добычи и обогащения угля на основе технологий искусственного интеллекта и промышленного интернета вещей // Уголь. 2024. №12. URL: https://cyberleninka.ru/article/n/razrabotka-intellektualnoy-sistemy-upravleniya-protsessami-dobychi-i-obogascheniya-uglya-na-osnove-tehnologiy-iskusstvennogo (25.12.2024).

63. Как искусственный интеллект помогает российской энергетике... [Электронный ресурс] // www.eprussia.ru – Режим доступа: https://www.eprussia.ru/market-and-analytics/7912376.htm, свободный. – Загл. с экрана

64. Интеллектуальное управление и автоматизированное... [Электронный ресурс] // mining-media.ru – Режим доступа: https://mining-media.ru/ru/article/newtech/19515-intellektualnoe-upravlenie-i-avtomatizirovannoe-regulirovanie-v-ugolnoj-promyshlennosti-transformatsiya-tekhnologicheskikh-sistem-v-epokhu-tsifrovizatsii, свободный. – Загл. с экрана

65. Усольцев В. А., Часовских Е. П., Цепордей И. С. Иисследование методов и обработка баз данных о биомассе лесов евразии как нейронных сетей. часть 2. новые возможности искусственного интеллекта при прогнозировании климатически обусловленных изменений // Цифровые модели и решения. 2022. №2. URL: https://cyberleninka.ru/article/n/issledovanie-metodov-i-obrabotka-baz-dannyh-o-biomasse-lesov-evrazii-kak-neyronnyh-setey-chast-2-novye-vozmozhnosti-iskusstvennogo (14.08.2025).

66. Указ Президента Российской Федерации от 10.10.2019 № 490 «О развитии искусственного интеллекта в Российской Федерации». Собрание законодательства РФ, 14.10.2019, № 41, ст. 5700

67. О внесении изменений в Указ Президента Российской Федерации от 10 октября 2019 г. № 490 "О развитии искусственного интеллекта в Российской Федерации" и в Национальную стратегию, утвержденную этим Указом: указ Президента РФ от 15 февраля 2024 г. № 124// Собрание законодательства РФ, 19.02.2024, № 4, ст. 1102.

68. Часовских, В. П. Естественный и искусственный интеллект как инструмент преобразования данных [Электронный ресурс]: монография / В. П. Часовских, В. А. Усольцев, Е. В. Кох. – Электрон. текст. дан. (2,9 Мб). – Киров: Изд-во МЦИТО, 2025. – 1 электрон. опт. диск (CD-R). – Систем. требования: PC, Intel 1 ГГц, 512 Мб RAM, 2,9 Мб свобод. диск. пространства; CD-привод; ОС Windows XP и выше, ПО для чтения pdf-файлов. – Загл. с экрана.

69. Igor Anureev. What Prevents the Use of Formal Methods for Al Verification. [Видео]. 24 марта 2025 г. Видеохостинг RUTUBE, Россия. [Электронный ресурс]. URL: https://rutube.ru/video/ad09740477e5dd2b1df496fa54441746/?ysclid=mdnei2x5sr29817 3362 (15 июня 2025).

70. Воронов М.П., Усольцев В.А., Часовских В.П. Исследование методов и разработка информационной системы определения и картирования депонируемого лесами углерода в среде NATURAL. Екатеринбург, УГЛТУ, (2-е издание, исправленное и дополненное), 2012. – 193 с.

СВЕДЕНИЯ ОБ АВТОРАХ

Часовских Виктор Петрович – профессор кафедры шахматного искусства и компьютерной математики Уральского государственного экономического университета, доктор технических наук, профессор
e-mail: u2007u@ya.ru

Усольцев Владимир Андреевич – профессор Уральского государственного лесотехнического университета, доктор сельскохозяйственных наук, профессор
e-mail: usoltsev50@mail.ru

Кох Елена Викторовна – доцент кафедры шахматного искусства и компьютерной математики Уральского государственного экономического университета, кандидат сельскохозяйственных наук, доцент
e-mail: elenakox@mail.ru

Акчурина Галия Абдулазисовна – преподаватель кафедры шахматного искусства и компьютерной математики Уральского государственного экономического университета, кандидат сельскохозяйственных наук, доцент
e-mail: gaa20111@yandex.ru

Оформление и верстка *Ю. Болдырева*

Дата подписания к использованию: 19.08.2025
Объем издания: 2,0 Мб. Комплектация: 1 электрон. опт. диск (CD-R)
Тираж 17 экз.

Издательство АНО ДПО «Межрегиональный центр
инновационных технологий в образовании»
610047, г. Киров, ул. Свердлова, 32а, пом. 1003
Тел.: 8-800-222-30-98
https://mcito.ru/publishing; e-mail: book@mcito.ru

ISBN 978-5-907974-79-1