ИТМО

А.А. Менщиков, Н.С. Кармановский

КИБЕРУСТОЙЧИВОСТЬ СИСТЕМ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

Учебно-методическое пособие

Санкт-Петербург
2024

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

УНИВЕРСИТЕТ ИТМО

А.А. Менщиков, Н.С. Кармановский

КИБЕРУСТОЙЧИВОСТЬ СИСТЕМ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

УЧЕБНО-МЕТОДИЧЕСКОЕ ПОСОБИЕ

РЕКОМЕНДОВАНО К ИСПОЛЬЗОВАНИЮ В УНИВЕРСИТЕТЕ ИТМО
по направлению подготовки 10.04.01 Информационная безопасность
в качестве учебно-методического пособия для реализации основных профессиональных образовательных программ высшего образования магистратуры

Санкт-Петербург
2024

Менщиков А.А., Кармановский Н.С. **Киберустойчивость систем искусственного интеллекта** Учебно-методическое пособие. – СПб: Университет ИТМО, 2024.– 52 с.

Рецензент: Воробьева Алиса Андреевна, кандидат технических наук, доцент факультета безопасности информационных технологий, Университета ИТМО.

Учебное пособие разработано для методической помощи обучающимся по направлению подготовки 10.04.01 – «Информационная безопасность».
В пособии рассмотрены вопросы обеспечения киберустойчивости систем, использующих машинное обучение и искусственный интеллект. Предложены практические задания, тесты и вопросы для самопроверки знаний для изучения подходов к обеспечению и повышению киберустойчивости. Формулируются основные концепции киберустойчивости, принципы внедрения киберустойчивости в системах искусственного интеллекта, а также методы и подходы к обеспечению киберустойчивости современных сложных интеллектуальных систем. Изучаются вопросы оценки угроз существующих интеллектуальных систем и специальные инструменты для настройки базовых механизмов киберустойчивости для сложных систем.
Учебное пособие может быть рекомендовано студентам, осуществляющим подготовку по направлению «Информационная безопасность».

ИТМО (Санкт-Петербург) — национальный исследовательский университет, научно-образовательная корпорация. Альма-матер победителей международных соревнований по программированию. Приоритетные направления: IT и искусственный интеллект, фотоника, робототехника, квантовые коммуникации, трансляционная медицина, Life Sciences, Art&Science, Science Communication.
Лидер федеральной программы «Приоритет-2030», в рамках которой реализовывается программа «Университет открытого кода». С 2022 ИТМО работает в рамках новой модели развития — научно-образовательной корпорации. В ее основе академическая свобода, поддержка начинаний студентов и сотрудников, распределенная система управления, приверженность открытому коду, бизнес-подходы к организации работы. Образование в университете основано на выборе индивидуальной траектории для каждого студента.
ИТМО пять лет подряд — в сотне лучших в области Automation & Control (кибернетика) Шанхайского рейтинга. По версии SuperJob занимает первое место в Петербурге и второе в России по уровню зарплат выпускников в сфере IT. Университет в топе международных рейтингов среди российских вузов. Входит в топ-5 российских университетов по качеству приема на бюджетные места. Рекордсмен по поступлению олимпиадников в Петербурге. С 2019 года ИТМО самостоятельно присуждает ученые степени кандидата и доктора наук.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Киберустойчивость – это способность информационных систем осуществлять свое штатное функционирование в условиях воздействия компьютерных атак в киберпространстве [1]. Сегодня мы уже не говорим о том, что нужно предотвращать атаки. Мы вынуждены учиться строить системы, изначально предполагая, что функционировать таким системам придется в условиях постоянных кибератак, вне зависимости от того, является ли система простым вебсайтом или сложной интеллектуальной системой с использованием искусственного интеллекта. Цель данного пособия – помочь освоить базовые принципы построения киберустойчивых систем и показать принципы их внедрения на практике.

В рамках данного пособия будут рассмотрены основные концепции киберустойчивости, принципы внедрения киберустойчивости в системах искусственного интеллекта, а также методы и подходы к обеспечению киберустойчивости современных сложных интеллектуальных систем. Вы сможете самостоятельно оценивать угрозы существующих интеллектуальных систем, вырабатывать предложения по внедрению практик киберустойчивости, а также использовать специальные инструменты для настройки базовых механизмов киберустойчивости для сложных систем.

Структурно пособие состоит из основных и дополнительных разделов. Разделы содержат как теоретический, так и практический материал. Практический материал предназначен для подготовки к выполнению практических работ по оценке и повышению киберустойчивости информационных систем, использующих машинное обучение и искусственный интеллект. Пособие содержит теоретическую информацию необходимую для получения начальных сведений, а также перечень практических работ, контрольных вопросов и тестов, призванных помочь студентам в усвоении результатов обучения по дисциплине «Киберустойчивость систем искусственного интеллекта». Дополнительно приведен список литературы, который включает литературу, рекомендуемую для дальнейшего самостоятельного освоения.

ГЛАВА 1. КИБЕРУСТОЙЧИВОСТЬ И СИСТЕМЫ ИСКУССТВЕННЫЙ ИНТТЕЛЛЕКТА

Первым делом необходимо определить, что такое киберустойчивость и в чем её отличие от кибербезопасности.

Устойчивость – это способность объекта сохранять своё текущее состояние в условиях влияния на него различных внешних воздействий.

Соответственно, *киберустойчивость* – это способность информационных систем осуществлять свое штатное функционирование в условиях воздействия компьютерных атак в киберпространстве [1, 2].

Данный термин отлично описан в специальной публикации *NIST 800-160*, озаглавленной «*Разработка киберустойчивых систем: подход к безопасной разработке систем*» [3]. В переводе на русский – «*способность предвидеть, выдерживать, восстанавливаться и адаптироваться к неблагоприятным условиям, нагрузкам, атакам или компрометациям систем, которые используют или активируются киберресурсами*».

Сама природа вещей устроена таким образом, что все, что может пойти не так, неизбежно пойдет не так. Это в равной степени относится и к инцидентам, связанным с безопасностью. Нет сомнений, что они произойдут. Необходимо принять как данность невозможность достижения абсолютной защищенности информационных систем. Поэтому все организационные и технические меры применяются лишь для повышения киберустойчивости. Важно понимать, что абсолютно любую программу, информационную систему или сеть можно взломать. Для злоумышленников это лишь вопрос времени и ресурсов.

Соответственно, мы видим переход от иллюзии того, что наши меры помогут стопроцентно защитить продукт от угроз, к тому, чтобы минимизировать вероятность такого события и значительно уменьшить цену восстановления после кибератаки.

Киберустойчивость систем машинного обучения (МО) и искусственного интеллекта (ИИ) имеет свою специфику.

Роли, связанные с машинным обучением и искусственным интеллектом в компаниях

Рассмотрим основные роли, которые существуют в компании, связанной с системами ИИ и МО. Хотя приведенный список ролей не является исчерпывающим, он дает общее представление о вовлеченных людях. Стоит отметить, что эти роли не обязательно принадлежат одному человеку, скорее один человек может – а в небольших организациях часто должен – выполнять несколько ролей [4]. Например, многие специалисты по данным также решают задачи машинного обучения, и иногда DevOps и

информационные технологии (ИТ) в контексте ролей являются синонимами. Эмпирическое правило заключается в том, что чем крупнее организация, тем более специализированными являются отдельные лица.

- Специалисты по обработке и анализу данных занимаются ключевыми задачами, связанными с разработкой моделей машинного обучения, решающих конкретные задачи бизнеса. Их будний день состоит из анализа пользовательских данных, построения моделей, оптимизации метрик.
- Инженеры данных отвечают за инфраструктуру, обеспечивающую хранение и доступность данных. Они создают системы, которые принимают необработанные данные из различных источников и сохраняют их в централизованном хранилище.
- Инженер по машинному обучению является относительно молодой ролью и не всегда отделяется от специалистов по анализу данных (data science). Инженеры по МО занимаются доработкой теоретических концепций, встраивают их в итоговую производственную (Production) систему. Как следует из названия, они понимают теоретическую сторону машинного обучения, но больше вовлечены в парадигмы, встречающиеся в инженерии, такие как непрерывная интеграция и непрерывная доставка. Они занимаются автоматизацией и масштабированием обучения моделей, тестированием моделей перед развертыванием, а также развертыванием и мониторингом моделей. Часто они переписывают или реструктурируют код, написанный специалистами по данным; например, преобразование «Jupyter-блокнотов» в масштабируемые программные сценарии.
- Инженеры DevOps обеспечивают автоматизацию тестирования и доставки нового кода. В проектах машинного обучения DevOps может быть связующим звеном между моделью машинного обучения и приложением конечного пользователя; например, выпуск нового механизма рекомендаций для веб-приложения. Масштабируемость, стабильность и скорость отклика приложения – это все их относительная зона ответственности.
- ИТ – это группа, включающая сотрудников, которые занимаются всей программной инфраструктурой, исключая машинное обучение и искусственный интеллект. Например, команды, которые пишут мобильное приложение, использующее модели машинного обучения, самими моделями они не занимаются, но программируют фронт, бэк, API, настраивают базы данных. Сюда, кстати, относят, в том числе, и специалистов по информационной безопасности, тестировщиков и так далее.

Особенности систем машинного обучения и искусственного интеллекта

В чем же заключается отличие и особенности систем МО и ИИ по сравнению с традиционными системами?

Первое отличие – это данные. Это может показаться очевидным, но добавление данных, обычно больших данных, ставит перед командой разработки совершенно новые задачи.

В то время как программное обеспечение, как правило, может относительно легко разрабатываться локально, для внесения изменений в модель, обучающуюся на больших данных, потребуется специальный инструментарий. Некоторые гипотезы просто невозможно проверить без отправки кода на вычислительный кластер. Этот факт существенно усложняет инфраструктуру для работы.

Ещё одна проблема – версионирование. Если в классических приложениях каждая версия кода порождает версию приложения, то в системах МО версию модели порождают уже две сущности – данные и код. Получается уже двумерная сложность. Такая новая переменная резко увеличивает сложность, так как для воспроизведения результатов вам необходимо иметь возможность воспроизвести данные, которые вы использовали.

Что же можно назвать приложением? Определимся, что это будет некая серверная система (неважно, развернутая на голом железе или у облачного провайдера). Для работы с данной системой используются клиентские приложения (например, мобильное приложение или веб-интерфейс в браузере). Отличие такой системы от типового веб-приложения на самом деле незаметно для внешнего пользователя. Он нажимает кнопки и получает результат у себя в браузере. Главное отличие состоит в том, как именно эти данные к нему попадают, какие этапы обработки они проходят, где хранятся, как подготавливаются и так далее. Здесь у систем машинного обучения и искусственного интеллекта существует огромная разница по сравнению с теми же веб-приложениями. Но, с другой стороны, сегодня мы видим признаки использования МО и ИИ даже и в традиционных приложениях, например, в банках или в социальных сетях. Так что граница смывается, а сложность систем все возрастает.

Архитектура систем

Рассмотрим пример самой простой архитектуры (рисунок 1.1), которую можно представить – иными словами, традиционное веб-приложение, которые были так популярны десятки лет назад. В этой архитектуре мы видим клиента, который обращается к сервису при помощи клиентского приложения. В данном случае таким приложением может выступать веб-браузер. Браузер подключается к веб-серверу и общается с

ним по сети при помощи какого-либо протокола. Например, это может быть привычный всем HTTP протокол или же что-то более сложное, например протокол «Websockets» или другой бинарный протокол.

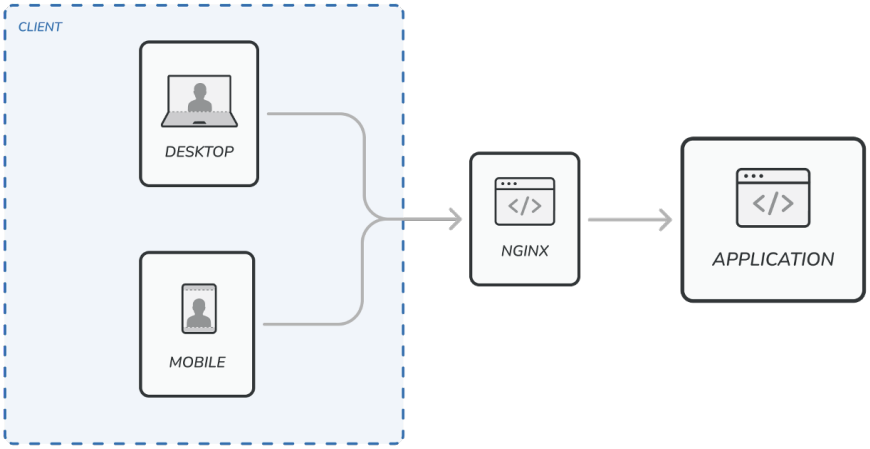

Рисунок 1.1 – Пример простейшей архитектуры

Веб-сервер сейчас уже никто не разрабатывает с нуля, обычно используют какой-то промышленный комбайн: например, Nginx или HAProxy или Traefik. Кто-то еще использует Apache или IIS. Разберем основных действующих лиц [5]:

- Клиент – это пользователь с его клиентским приложением.
- Браузер – приложение, упрощающее взаимодействие пользователей с сервером.
- Мобильное приложение тоже может выступать интерфейсом для пользователя по взаимодействию с веб-приложением.
- Веб-сервер обрабатывает запрос пользователя, формирует ответ и передаёт его обратно пользователю.
- Веб-приложение – это самый главный элемент в системе. Поскольку веб-сервер занимается только обслуживанием канала связи, накидывает SSL шифрование, отдает статику и прочее, саму бизнес-логику должно выполнять приложение, которое пишут разработчики. Обычно это приложение имеет какой-то командный интерфейс, по которому веб-сервер с ним общается. Таким интерфейсом может быть и HTTP протокол. В таком случае веб-приложение просто использует свой встроенный минималистичный HTTP сервер.
- DNS – распределенная система для получения информации о состоянии каждого домена.

Веб-сервер – это та сторона, с которой общается клиентский веб-браузер. Общаются они по специальному заранее обговорённому формату. Этот формат называется протоколом. Существует большое количество различных протоколов. Один из них (является самым известным) – HTTP. Его особенностью является то, что это текстовый протокол, соответственно, человек легко в состоянии этот протокол изучать и анализировать, в отличие

от любых бинарных протоколов. Например, сейчас очень распространённым является протокол Websocket, поверх которого обмениваются различными Protobuf сообщениями, которые представляют из себя сырой набор байтов.

Веб-сервер часто выступает в режиме прокси, когда он получает запрос выполняет различные задачи, например отдачу статики, которую Веб-сервер, в отличие от приложения, умеет делать эффективно с точки зрения потребления памяти и ресурсов.

Веб-сервер производит логирование действий и осуществляет авторизацию, аутентификацию, снимает слой шифрования SSL и затем перенаправляет данные в зависимости от настроек веб-приложений.

Таким образом, один Веб-сервер может обслуживать большое количество разных приложений (рисунок 1.2). Например, бизнес-логику может обслуживать сервис php-fpm, который слушает по пути /main, а чат может обслуживать NodeJS приложение, которое работает на вебсокетах. И все эти сообщения будут перенаправляться через Веб-сервер.

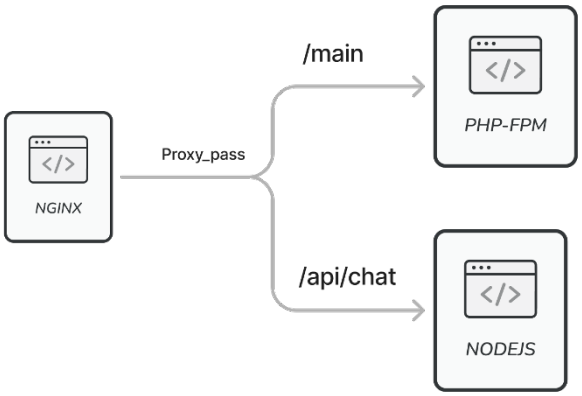

Рисунок 1.2 – Схема обслуживания нескольких приложений Веб-сервером

DNS (система доменных имен) – это технология, позволяющая преобразовывать доменные имена (например, «www.google.com») в IP-адреса (например, «192.168.1.1»). Это важно, потому что IP-адреса нелегко запомнить людям, а доменные имена намного проще.

Когда пользователь вводит доменное имя в свой браузер, компьютер сначала отправляет запрос на DNS-сервер. Он действует как каталог доменных имен и связанных с ними IP-адресов. Если у DNS-сервера нет IP-адреса для запрошенного имени, он будет отправлять запрос на другие серверы до тех пор, пока не будет найден правильный IP-адрес. В зависимости от типа DNS-сервера такой запрос он либо отправит сам, либо скажет вам адрес сервера, чтобы вы сами к нему обратились.

Как только правильный IP-адрес будет найден, компьютер может отправить запрос на соответствующий Веб-сервер для запрошенного веб-сайта.

После того, как данные получены, их необходимо хранить (рисунок 1.3).

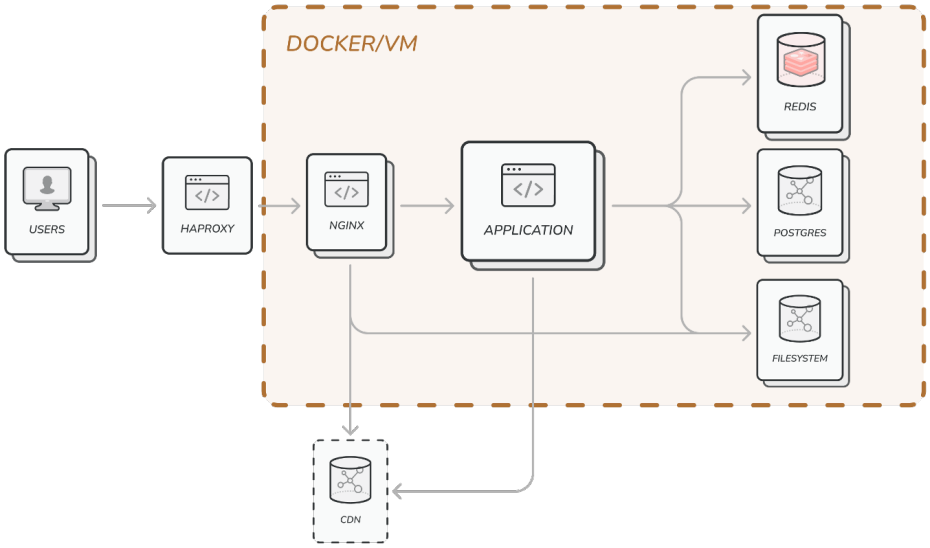

Рисунок 1.3 – Схема обслуживания нескольких приложений веб-сервером

По мере усложнения архитектуры системы Веб-сервер и Веб-приложение начинают взаимодействовать с широким множеством дополнительных систем:

- Базы данных – это упорядоченный набор структурированной информации или данных, которые обычно хранятся в электронном виде в компьютерной системе. База данных обычно управляется системой управления базами данных (реляционные базы данных – MySQL, PostgreSQL, Microsoft SQL Server, Oracle Database и нереляционные базы данных – MongoDB).
- Файловая система служит для хранения файлов на отдельном разделе, откуда их при необходимости может выдавать веб-сервер напрямую, так как он умеет очень эффективно отдавать такую статику.
- Кэш – это промежуточный буфер с быстрым доступом к нему, содержащий информацию, которая может быть запрошена с наибольшей вероятностью. Кэши допустимо эффективно использовать и как самостоятельную базу данных (например, часто для этого используется Redis).
- Балансировщик нагрузки – программа, выполняющая задачи оптимизации использования ресурсов и сокращения времени обслуживания запросов. Балансировщик перенаправляет входящие запросы на один из нескольких серверов. Хорошим примером балансировщика является HAProxy. Впрочем, и на Nginx тоже можно реализовать простые сценарии балансировки нагрузки.
- Контейнеры – это специальные сущности, поддерживаемые ядром операционной системы, которые позволяют создавать нечто похожее на виртуальную машину для изоляции и воспроизводимости сборок.

К плюсам контейнеров можно отнести легковесность, так как они не тратят больших объемов физической памяти, как виртуальная машина. Также можно выделить портативность, она заключается в том, что контейнеры можно переносить на разные платформы без проблем совместимости. К недостаткам можно отнести несовместимость с некоторыми операционными системами. Также виртуальные машины более безопасны, так как запускается отдельная операционная система, которая имеет свою защиту, в то время как безопасность контейнеров зависит от безопасности хоста, на котором он находится.

- Сеть доставки контента (CDN) – это географически распределенная сеть прокси-серверов и их центров обработки данных. CDN работает путем распределения контента по множеству «пограничных» серверов по всему миру, так что пользователи в итоге загружают активы с «пограничных» серверов, а не с исходного сервера.
- Очереди – большинству веб-приложений необходимо асинхронно выполнять некоторую фоновую работу, которая напрямую не связана с ответом на запрос пользователя. Например, Google необходимо просканировать и проиндексировать весь Интернет, чтобы вернуть результаты поиска. Он не делает этого каждый раз, когда вы ищете. Вместо этого он сканирует Интернет асинхронно, попутно обновляя поисковые индексы [6].

Отдельно стоит выделить хранилище данных, специфичное для систем МО и ИИ – «озеро данных» или «Data Lake». Под термином Data Lake обычно понимают массив неструктурированных данных, загруженных из источников «как есть». Data Lake предназначен для интеграции информации из разных каналов, приведения их к одному виду. Этот подход позволяет превратить сырую информацию в метрики, необходимые бизнесу.

Пока не появилась концепция больших данных (Big Data) – в ходу были системы управления базами данных (СУБД), в которых агрегировали информацию. Там не хранился исходный материал из источника, система приводилась к реляционному виду. Учитывая все увеличивающийся объем данных, многие ведущие компании перешли на Big Data-решения, основанные на Open Source. В том числе это позволило хранить RAW-данные – «сырую» информацию из источников.

Изначально в Data Lake значительный объем занимает слой сырых данных – зеркало источников. Для большого количества алгоритмов необходимо исследование таких данных, потому что при трансформации и агрегации некоторые закономерности в данных теряются. Также этот слой пригодится, если мы сделали какую-то ошибку на уровне представления данных и необходимо их обновить. Далее – слой DDS (detail data storage). Это слой детальный слой данных, в котором они распределены уже по

бизнес-сущностям, которые необходимы конечному потребителю. Когда аналитикам нужно обратиться к детальным данным, им удобно пользоваться этим слоем, так как в нем отражена бизнес-модель. Следующий слой – это общие витрины данных, CDM (common data marts). В этом слое данные уже агрегированы и содержат метрики, которые считаются по определенным алгоритмам. Метрики нужны пользователям, их можно использовать для построения Data Science-моделей, для принятия решений или для BI-систем. Типовой стек технологий здесь – это Apache AirFlow, Apache Spark, а хранятся данные в Hadoop.

Подводя итог главы, можно сделать вывод о том, что архитектура современных приложений очень и очень сложна. Для того, чтобы обеспечивать её безопасность, недостаточно просто формально относиться к требованиям регуляторов и прикрываться бумажками. Необходимо разобраться в каждой составляющей на уровне разработчика или администратора.

ГЛАВА 2. КОНЦЕПЦИЯ MLOPS ДЛЯ ОБЕСПЕЧЕНИЯ КИБЕРУСТОЙЧИВОСТИ СИСТЕМ ИИ

Единого и согласованного определения MLOps сегодня не существует. Многие авторы пытались его дать, но одновременно понятное и системное определение найти очень сложно. MLOps, судя по названию, это такой DevOps для систем МО и ИИ? Или же это методология внедрения МО в существующие DevOps конвейеры? Определимся, что мы будем подразумевать под MLOps и дадим следующее интуитивно понятное определение:

MLOps – это инженерная дисциплина, целью которой является унификация разработки и развертывания систем МО для стандартизации и оптимизации непрерывной доставки высокопроизводительных моделей.

Чтобы понять MLOps, следует сначала разобрать жизненный цикл систем машинного обучения. Некоторые исследователи выделяют следующий жизненный цикл системы МО как продукта. Разные этапы можно разбивать и менять, но суть остается примерно следующей (рисунок 2.1).

Рисунок 2.1 – Этапы жизненного цикла системы с использованием МО и ИИ

В течение жизненного цикла над продуктом может работать несколько разных команд, а соответственно, появляются и различные роли. Их может быть и больше, но в первую очередь выделяют следующие роли [7]:

- Те, кто ставят бизнес-цели, формирует KPI и на основе них формулируют задачи.
- Инженеры, ответственные за подготовку данных.

- Специалисты по анализу данных, которые проектируют саму модель, какая бы она ни была.
- Инженеры, которые занимаются внедрением и мониторингом работающей модели как сервиса.

Соответственно, получается цикл из последовательных постановок задач на разработку, доработку или же обновление сервиса, продукта или новой фичи. Затем происходит подготовка данных, на основе которых будет производиться обучение и верификация. После этого (или даже параллельно) происходит сам процесс проектирования и создания модели. Заканчивается все доставкой в Production.

Если посмотреть на то, как в существующей литературе обычно визуализируют процесс МО, можно увидеть нечто подобное схеме процесса, в котором модель сначала проектируется, затем разрабатывается, потом доставляется в продакшн, фидбек от её использования является триггером на доработку и оптимизацию и так далее.

Однако в реальности многие организации до сих пор переход от разработки модели к её доставке в виде работающего в производственной среде сервиса выполняют вручную, причем этот процесс обычно крайне запутан и нестабилен.

С точки зрения бизнеса важной целью является снижение издержек, окупаемость инвестиций, максимизация прибыли и минимизация технических рисов.

Так что главной целью MLOps является уменьшение технических сложностей, доведения модели от идеи до производства в кратчайшие сроки и с минимальным риском.

Time To Market (TTM) – это время от начала разработки идеи фичи или продукта до её конечной реализации, когда вы продаете её клиенту. То есть Time To Market – это в прямом смысле «время до выхода на рынок».

Разберем, какие риски существуют и как MLOps может помочь их уменьшить:

1. Потеря знаний. Существует термин – *bus factor*. Он описывает риск неожиданного исчезновения ключевого участника проекта (например, из-за того, что он попал под автобус). Если только один инженер знает, как устроен конвейер машинного обучения для конкретного сервиса, то его bus factor равен единице. Если двое – двум. MLOps концепция стремится сделать данные максимально самодокументированными. Конечно, все равно возникают нюансы, которые известны далеко не всем в команде. Но с MLOps фиксацию и передачу знаний организовать гораздо проще. Если у вас до сих пор весь деплой выполняется вручную через сборку разрозненных

частей кода из Jupyter-блокнотов в рабочую систему, то заменить ключевого сотрудника будет гораздо сложнее. В классическом программировании проблему передачи знаний решает документация и система контроля версий. В системах машинного обучения вам потребуется контролировать ещё и версию данных, а также взаимоотношение всех частей системы между собой: данные, код, фичи, модели и так далее.

2. Сбои в Production. Доставка неработающего кода в Production происходит довольно часто, если деплоить руками. Поэтому команды и тратят много времени на создание CI/CD конвейеров, чтобы разработчики могли быть уверены, что их обновление ничего не сломает в рабочей среде. В системах ИИ все так же, только нюанс состоит в том, что тесты работоспособности будут отличаться. Например, нужно проверить, что в данных не появилось аномалий или что точность модели не проседает для различных классов обнаружения.
3. Требования регуляторов и репутация. ИИ – очень молодая область, и нормы здесь только проектируются. Уже можно видеть дискуссии по поводу авторского права на генеративные результаты, обученные на чужих произведениях. Модели машинного обучения вынуждены блокировать запросы политического, порнографического или нецензурного содержания.
4. Безопасность. Введение новых сущностей влияет на защищенность итоговой системы. Атаки на сами модели машинного обучения – это предмет отдельного курса, но все, что касается инфраструктуры, мы еще будем разбирать.

Представим типовую картину использования системы ИИ в компании, которая предоставляет услуги различным клиентам или своим подразделениям. Во-первых, модели редко работают в течение длительного времени и, как правило, требуют регулярного дообучения. Во-вторых, модели часто переобучаются для каждого клиента на основе его конкретных данных или географического положения и т. д. Как только мы приходим к тому, что нам нужно накатить новую версию программного обеспечения на множество версий модели на множестве версий данных для разных клиентов, мы сразу получаем экспоненциальный рост числа деплоев. Чтобы получить действительно масштабируемую бизнес-модель на основе МО и ИИ, обычно требуется сделать шаг назад и подумать о своем продукте с точки зрения конвейера, а не отдельного экземпляра обученной модели. Это делают сегодня самые успешные компании.

DevOps – важная практика разработки и эксплуатации крупномасштабных программных систем. Эта практика обеспечивает такие преимущества, как сокращение циклов разработки, увеличение скорости

развертывания и повышения надежности релиза [8]. Для достижения этих преимуществ вводят две концепции разработки программных систем:

- Непрерывная интеграция (CI);
- Непрерывная доставка (CD).

Система машинного обучения (включая ИИ) – это программная система, поэтому аналогичные методы применяются, чтобы гарантировать надежную сборку и эксплуатацию систем машинного обучения в любом масштабе.

Системы ИИ и МО схожи с традиционными программными системами в области непрерывной интеграции, модульного тестирования, интеграционного тестирования и непрерывной доставки. Однако существуют несколько заметных отличий:

- Непрерывная интеграция (CI) – это уже не только тестирование и проверка кода и компонентов, но также тестирование и проверка данных, схем данных и моделей.
- Непрерывная доставка (CD) больше не относится к одному программному пакету или сервису, а представляет собой систему (или конвейер обучения), которая должна автоматически развертывать другой сервис (например, сервис прогнозирования на основе моделей).
- Непрерывное обучение (CT) – это новое свойство, уникальное для систем ML, которое связано с автоматическим переобучением и обслуживанием моделей [9].

Системы МО и ИИ также имеют свои отличия и в глобальных рабочих процессах:

- Навыки разработчиков. В команду обычно входят специалисты по данным или аналитики, которые не являются профессиональными программистами, способными создавать сервисы и разбираться в сложной архитектуре.
- Шаблоны разработки. В разработке систем, связанных с ML, высока доля экспериментов. Разработчики постоянно тестируют алгоритмы, методы и подбирают подходящие параметры для получения требуемого результата. В обычной разработке данный фактор экспериментов существенно ниже.
- Шаблоны тестирования. Поскольку в дополнение к привычным модульным и интеграционным тестам потребуется проверка данных и оценка качества обученной модели, процесс становится существенно более сложным.
- Развертывание. Аналогично предыдущему пункту, теперь необходимо проектировать и автоматизировать развертывание данных, моделей и использовать новое специфическое оборудование.

- Production. Модели машинного обучения могут работать плохо не только из-за неоптимального кода, но и из-за постоянно изменяющихся данных. Необходимо постоянно следить за качеством работы модели, чтобы оперативно производить переобучение, когда значения отклоняются от ожидаемых [8, 9].

В MLOps существуют различные модели, определяющие уровни зрелости технологии. Они помогают компаниям понять, на каком этапе развития они находятся сейчас и какими средствами они могут перейти на следующий уровень. Также использование общепринятых методик определения зрелости позволяет определить свое место среди конкурентов. Рассмотрим три основные модели:

- Модель Google – самая ранняя, но при этом весьма подробно описанная;
- Модель Microsoft Azure;
- Модель GigaOm – одна из самых подробно описанных и во многом понятных моделей.

Модель зрелости GigaOm

В модели от GigaOm каждый уровень зрелости расписывается через пять категорий: стратегия, архитектура, моделирование, процессы и управление.

Руководствуясь этой моделью на ранних этапах создания ML-системы, можно заранее обдумать важные аспекты и уменьшить шансы на неудачу.

На нулевом уровне организациям еще предстоит найти место для ML, не говоря уже о MLOps. У них нет планов нанимать специалистов по обработке и анализу данных, и они плохо понимают, с какими сложностями им придется столкнуться. Процессы ручные, практически без возможности измерить успех.

Организации на первом уровне, как правило, хороши в традиционном сборе данных и аналитике и могут иметь определенный уровень приверженности облачным подходам; например, они могут уже получать доступ к данным из облака. Тем не менее, им еще предстоит заняться МО и ИИ стратегически.

Организации на втором уровне активно стремятся использовать преимущества машинного обучения и добились определенного прогресса, но теперь им необходимо координировать свои усилия с целью масштабирования.

На третьем уровне организации внедряют машинное обучение эффективным образом, видят полученные преимущества и выросли до нескольких специалистов по данным. Разработчики искусственного

интеллекта тесно сотрудничают с разработчиками программного обеспечения в рамках рабочего процесса, согласованного между процессами MLOps и DevOps.

Для организаций на четвертом уровне бизнес принципиально отличается от того, что был до перехода MLOps, благодаря комплексному использованию машинного обучения. Управление стало центральным элементом стратегии ML, учитывая как возможность злонамеренного использования МО в жизненном цикле MLOps, так и необходимость обеспечения объяснимых и прозрачных результатов.

Модель зрелости Microsoft Azure

Модель Microsoft Azure во многом напоминает предыдущую.

- Уровень 0 – отсутствуют процессы MLOps, всё деплоится вручную;
- Уровень 1 – DevOps для кода приложения существует, но без специфики для ML;
- Уровень 2 – Автоматизированы процессы обучения;
- Уровень 3 – Автоматизированы процессы деплоя и доставки;
- Уровень 4 – Все операции автоматизированы.

На нулевом уровне вам трудно управлять жизненным циклом ML, команды разделены, релизить тяжело, а большинство систем – «черные ящики». Команда собирает релизы вручную, тестирует и обучает тоже вручную и не собирает никаких метрик, кроме проверки точности перед релизом.

На первом уровне уже есть DevOps, но релиз блокируется командой по работе с данными, процессы которой не автоматизированы. Код может храниться в системе контроля версий, но не понятно как он связан с данными и какие наборы использовались в конкретном эксперименте. Однако само приложение уже собирается автоматически, и присутствуют интеграционные и юнит тесты.

На втором уровне мы видим, что тренировочная среда уже полностью управляема, работу модели легко воспроизвести, но релиз все также происходит вручную. Тем не менее уже можно управлять версионированием моделей и централизованно хранить метрики.

На третьем уровне релизы автоматизированы, что дает полную документированность и воспроизводимость начиная от обучения модели до её работы в Production. Уже можно проводить A/B тесты.

На финальном, четвертом уровне автоматизированы метрики, триггеры и масштабированы процессы сборки и доставки продукта.

Модель зрелости Google

Практика MLOps означает, что вы выступаете за автоматизацию и мониторинг на всех этапах создания системы машинного обучения, включая интеграцию, тестирование, выпуск, развертывание и управление инфраструктурой.

Специалисты по обработке и анализу данных, конечно, могут внедрить и обучить модель машинного обучения с некоторой эффективностью на автономном наборе данных. Однако реальная проблема заключается не в построении модели машинного обучения, а в создании интегрированной системы машинного обучения и непрерывном использовании ее в производственной среде.

Лишь небольшая часть реальной системы машинного обучения состоит из непосредственно программирования моделей. Требуемые окружающие элементы не менее обширны и сложны.

Во многих командах есть специалисты по данным и специалисты по машинному обучению, которые могут создавать современные модели, но их процесс создания и развертывания моделей – полностью ручной. Это считается базовым уровнем зрелости или уровнем 0, характеризуемым следующими особенностями:

- Во-первых, это ручной, управляемый сценариями и интерактивный процесс: каждый шаг выполняется вручную, включая анализ данных, подготовку данных, обучение модели и проверку. Каждый шаг требует ручного выполнения и перехода от текущего шага к последующему. Все эти шаги, возможно, заскриптованы каким-то временным, экспериментальным кодом.
- Во-вторых, это разрыв между машинным обучением и средой исполнения. Процесс разделяет специалистов, которые создают модель, и инженеров, которые обслуживают модель как сервис. ML-специалисты передают обученную модель в качестве артефакта команде инженеров для развертывания в инфраструктуре. Обычно модель загружают в репозиторий или вовсе копируют в файловое хранилище.
- Следующая особенность – это редкие итерации деплоя. Процесс предполагает, что ваша команда управляет несколькими моделями, которые не меняются часто – либо произошло существенное обновление реализации модели, либо произошло переобучение модели с новыми данными. Новая версия модели развертывается всего пару раз в год.
- Отсутствие CI. Поскольку предполагается всего несколько изменений реализации, CI игнорируется. Обычно тестирование кода

производится просто в Jupyter-блокнотах или скриптами на компьютере ML-специалиста.

- А также отсутствие активного мониторинга производительности.

Целью уровня 1 является непрерывное обучение модели путем автоматизации конвейера машинного обучения. Такой процесс позволяет обеспечить непрерывную доставку модели в продакшн для работы в качестве сервиса. Чтобы автоматизировать процесс использования новых данных для переобучения моделей в рабочей среде, вам необходимо внедрить автоматизированные этапы проверки данных и моделей в конвейер, а также триггеры конвейера и управление метаданными.

Разберем основные характеристики процесса данного уровня зрелости:

- Быстрое экспериментирование. Этапы эксперимента машинного обучения хорошо организованы. Переход между этапами автоматизирован, что приводит к легкому проведению итерации экспериментов.
- Непрерывное обучение. Модель автоматически обучается в Production среде с использованием свежих данных при помощи событий-триггеров конвейера в реальном времени. Отправили команду или нажали кнопку, и процесс пошел.
- Унификация среды. Реализация конвейера, используемая в среде разработки или эксперимента, используется как в stage окружении, так и в Production. Если код отработал в stage, высока вероятность, что он соберется в Production окружении с той же эффективностью.
- Модульность компонентов. Для создания конвейеров машинного обучения компоненты должны быть пригодными для повторного использования, компоновки и, возможно, общими для конвейеров машинного обучения. Поэтому, хотя код все еще может храниться в Jupyter-блокнотах, исходный код компонентов должен быть модульным. Кроме того, в идеале компоненты должны быть помещены в контейнеры для изоляции среды выполнения и воспроизводимости сборок.
- Непрерывная доставка моделей. Работающий конвейер машинного обучения постоянно собирает сервис, готовый для исполнения бизнес-задачи для новых моделей, которые обучаются на новых данных. Этап развертывания полностью автоматизирован.

Если на нулевом уровне вы каждый раз развертываете обученную модель в Production, то для уровня 1 вы развертываете конвейер, который автоматически и периодически запускается для сборки и доставки вашей модели.

Можно видеть также новые компоненты:

- Валидация данных, требующаяся перед обучением модели, чтобы решить, следует ли переобучить модель или сразу остановить выполнение конвейера. Это решение принимается автоматически, если было определено, что в данных присутствуют аномалии.
- Валидация модели. Этот шаг выполняется после успешного обучения модели на новых данных. Он оценивает и проверяет модель, прежде чем она будет запущена в Production. Обычно тут считаются метрики, сравнивается loss с предыдущей итерацией и проверяется соответствие внешним интерфейсам, например, корректность структуры выдаваемых результатов.
- Хранилище фичей. Необязательный компонент, который представляет из себя централизованный репозиторий с описанием и структурой фичей. Это помогает создавать новые наборы фичей и переиспользовать существующие, исключая повторение. Настраивать правила валидации и рассчитывать историю метрик по ним.
- Подсистема управления метаданными. Содержит информацию о каждом выполнении конвейера. Позволяет изучить происхождение и изменение данных, артефактов, а также откатывать версии. Каждый раз, когда вы запускаете конвейер, в хранилище метаданных машинного обучения записываются:
    - Конвейер, компонент и версия
    - Время начала, окончания и длительности всех этапов
    - Аргументы
    - Артефакты и метрики на каждом шаге обработки
- Триггеры. Бывают по запросу (например, нажатие кнопки), по таймеру или расписанию, по появлению новых данных в репозитории и даже по падению метрик на Production (например, если произошел так называемый Concept Drift – статистические показатели модели начали изменяться существенно с течением времени ввиду изменения поведения пользователей).

В целом такая система уже весьма неплохо себя показывает, если предположить, что новые реализации конвейера развертываются нечасто и вы управляете только несколькими конвейерами, а также, если вы вручную развертываете новые реализации конвейера. Если же у вас много разнородных команд и проектов с конвейерами МО и вам нужно быстро развертывать новые реализации компонентов, то вам нужна полноценная среда для автоматизации сборки, тестирования и развертывания уже самих конвейеров машинного обучения [10].

Рассмотрим реализацию конвейера машинного обучения с использованием CI/CD или второй уровень зрелости. Архитектура похожа на предыдущую, но включает в себя некоторые новые компоненты: этап сборки и контроля кода самих компонентов и формируемых пакетов.

Получается, что тем самым стирается граница между МО и администрированием. Теперь это один единый MLOps процесс. Рассмотрим теперь его упрощенный конвейер обработки.

Он состоит из следующих этапов:

- Разработка и экспериментирование. Можно итеративно испытывать новые алгоритмы машинного обучения. Результатом этого этапа является исходный код шагов конвейера машинного обучения, который затем помещается в исходный репозиторий.
- Конвейерная непрерывная интеграция. Вы публикуете код и автоматически запускаете различные тесты. Результатом этого этапа являются компоненты конвейера (пакеты, исполняемые файлы и артефакты), которые будут впоследствии развернуты.
- Конвейерная непрерывная доставка. Вы развертываете артефакты, созданные на этапе CI, в целевой среде. Результатом этого этапа является развернутый конвейер с новой реализацией модели.
- Непрерывное обучение. Конвейер автоматически запускается в рабочей среде по расписанию или по иному триггеру. Результатом этого этапа является обученная модель, которая помещается в реестр моделей.
- Непрерывная доставка модели. Используем обученную модель для выполнения бизнес-функции, например, в качестве сервиса прогнозирования. Результатом этого этапа является развернутая служба.
- Мониторинг. Собираем статистику о производительности модели на основе данных в реальном времени. Результатом этого этапа является триггер для запуска конвейера или запуска нового цикла эксперимента.

Этап анализа данных и моделей по-прежнему является ручным процессом.

Подводя итог, можно сказать, что внедрение машинного обучения в производственной среде означает не только развертывание модели, но и развертывание конвейера машинного обучения, который может автоматизировать переобучение и сборку новых моделей. Настройка системы CI/CD позволяет автоматически тестировать и поднимать новые реализации конвейера. Можно сделать вывод, что иногда машинное обучение не является основным продуктом компании, а лишь вспомогательным инструментом и компания может проходить постепенный путь от экспериментов с МО и до осознания, что теперь настало время, когда MLOps стал действительно необходим.

ГЛАВА 3. ЭТИКА БЕЗОПАСНОГО ИСПОЛЬЗОВАНИЯ ИИ

Тема безопасности ИИ совсем еще молода. Стандартов много, но все они структурированы очень хаотично, оперируют одними и теми же иногда весьма абстрактными практиками и не всегда привязаны к конкретным приземленным рекомендациям. В целом это понятно, поскольку вся инфраструктура изменяется настолько быстро, что очередной hardening гайд по Kubernetes перестает быть актуален в течение полугода, в то время как базовые принципы вроде минимизации привилегий и запутывания злоумышленников никогда не выйдут из моды.

Рассмотрим несколько интересных попыток и лучших заявленных практик в области киберустойчивости ИИ.

Сотрудники института этики ИИ и машинного обучения, например, решили составить аналог OWASP Top 10 для машинного обучения. Также они выделяют некоторые принципы ответственного машинного обучения. Цель данных принципов – предоставить инженерам конкретные рекомендации по ответственной разработке систем машинного обучения и искусственного интеллекта [11].

1. Первый принцип – верификация человеком

Следует оценивать влияние неверных прогнозов и, когда это разумно, разрабатывать системы с процессами проверки с участием человека. Внедряя автоматизацию с помощью систем машинного обучения, легко забыть о влиянии неверных прогнозов. Технологи должны понимать последствия неверных прогнозов, особенно при автоматизации критически важных процессов, которые могут оказать существенное влияние на жизнь людей (например, здравоохранение, транспорт и т. д.).

2. Второй принцип – выявление предвзятости.

Необходимо развивать процессы, которые позволяют документировать и отслеживать предвзятость при принятии решений.

Специалисты должны сосредоточиться на создании процессов и методов для выявления и документирования неотъемлемого смещения данных, функций и результатов вывода, а затем и последствий этого смещения.

3. Объяснимость путем обоснования

Необходимо внедрять процессы для постоянного улучшения прозрачности и объяснимости моделей машинного обучения там, где это целесообразно. Из-за шумихи вокруг глубокого обучения часто загружают большие объемы данных в сложные конвейеры машинного обучения, надеясь, что что-то сработает, не понимая, как конвейеры работают внутри.

Можно использовать различные инструменты и подходы, чтобы сделать системы машинного обучения более объяснимыми, например, добавляя знания предметной области через сами фичи вместо того, чтобы просто позволять глубоким/сложным моделям делать их выводы.

Даже несмотря на то, что в некоторых ситуациях точность может снизиться, выигрыш в прозрачности и объяснимости может быть значительным.

4. Воспроизводимость

Инфраструктура должна обеспечивать разумный уровень воспроизводимости операций систем машинного обучения и ИИ.

Часто Production системы не имеют возможности диагностировать или эффективно реагировать, когда с моделью происходит что-то плохое, не говоря уже о воспроизведении тех же результатов.

Важно уметь выполнять стандартные процедуры, такие как возврат модели к предыдущей версии или воспроизведение входных данных для отладки определенных функций, что усложняет инфраструктуру.

5. Стратегия вытеснения

Следует разработать процессы бизнес-изменений таким образом, чтобы смягчить влияние на автоматизацию работников.

При развертывании систем, автоматизирующих процессы среднего и крупного масштаба, почти всегда возникает влияние на уровне организации или отрасли, которое затрагивает многих людей. Если этот процесс выполнять бесконтрольно, то может произойти ситуация потери знаний или потери контроля в критически важных точках, в которых могут происходить сбои.

6. Практичность выбора метрик

Необходимо, чтобы функции измерения точности и стоимости соответствовали приложениям, специфичным для предметной области.

Часто недостаточно просто использовать простую метрику точности или показатели по умолчанию, поскольку то, что может быть «правильным» для компьютера, может быть «неправильным» для человека (и наоборот).

7. Конфиденциальность

При разработке крупномасштабных систем, которые обучаются на основе данных, часто существует большое количество заинтересованных сторон, которые могут быть затронуты прямо или косвенно. Укрепление доверия между соответствующими заинтересованными сторонами достигается не только путем информирования о том, какие данные хранятся,

но и с помощью процессов, связанных с данными, а также понимания того, почему защита данных важна.

8. Осведомленность о рисках

Автономные системы принятия решений открывают двери для новых потенциальных нарушений безопасности.

Важно отметить, что значительная доля нарушений безопасности происходит из-за человеческой ошибки, а не из-за реальных взломов (например, кто-то случайно отправляет набор данных по электронной почте или теряет свой ноутбук/телефон).

Необходимо подготовиться к таким рискам безопасности, обучать персонал и настраивать безопасные процессы в организации.

УКАЗАНИЯ К ВЫПОЛНЕНИЮ И ПРОВЕРКЕ ПРАКТИЧЕСКИХ РАБОТ

Практические работы выполняются обучающимися индивидуально или в группе в течение курса.

Выполнение практической работы состоит в выполнении заданий, оформлении отчета и опциональной защиты в форме устного доклада. Задания необходимо выполнить с учетом указаний преподавателя. Оставшиеся невыполненными пункты задания практического занятия студент обязан доделать самостоятельно. Выполненные задания заранее представляются на проверку и оценивание преподавателю (1-ый этап работы).

На занятии студент готовит отчет о проделанной работе, фиксируя процесс выполнения работы. Представление отчета и защита работ проходят в конце курса (2-й этап). После проверки отчета преподаватель проводит устный опрос студентов для контроля усвоения ими основных теоретических и практических знаний по теме занятия (студенты должны знать смысл полученных ими результатов и ответы на контрольные вопросы).

В случае если оформление отчета и доклад обучающегося во время защиты соответствуют указанным требованиям, обучающийся получает максимальную оценку.

Комплект заданий для практических работ

№ п/п	Номер раздела дисциплины	Наименование практической работы	Цель работы
1	1	Особенности инфраструктуры систем искусственного интеллекта	Цель работы – изучить процедуры развертывания современных систем искусственного интеллекта (MLOps)
2	2	Принципы построения киберустойчивых систем искусственного интеллекта	Цель работы – изучить алгоритмы построения защищенных и киберустойчивых систем искусственного интеллекта (MLSecOps).
3	3	Инструментальные средства обеспечения киберустойчивости систем искусственного интеллекта	Цель работы – изучить инструментальные средства обеспечения защищенности и киберустойчивости систем искусственного интеллекта

Шаблон отчета по практической работе

Отчет по практической работе № ____

«____»
(название практической работы)

1. Цель и задачи работы: ____
2. Методика проведения исследования: ____
3. Анализ погрешностей: ____
4. Результаты: ____
5. Выводы: ____

Шкала оценивания и критерии оценки

К практической работе предъявляются следующие требования:

1. *к выполнению заданий* – в работе должны быть:
   a. представлены в логической последовательности основные этапы исследования или решения,
   b. указаны используемые теоретические положения и методы,
   c. получены точные численные результаты и построены требуемые графические изображения;
2. *к оформлению отчета* – отчет должен быть представлен в печатном или электронном виде в форматах doc, docx и содержать:
   a. титульный лист (название работы, ФИО исполнителей, номера групп, ФИО проверяющего);
   b. условия всех заданий;
   c. решение (исследование), его теоретическое обоснование, численные результаты;
   d. графики или рисунки, иллюстрирующие решение каждой задачи (выполненные в графическом или математическом редакторе);
   e. выводы.
3. *к докладу* – для доклада отводится от 7 до 10 минут. Во время доклада оценивается качество устного изложения материала и ответы на вопросы по теме работы. Доклад должен содержать:
   a. постановку задачи;
   b. изложение основных этапов исследования или решения;

c. ссылки на теоретический материал, используемый при исследовании и решении;

d. результаты исследования или решения и их оценку;

e. выводы.

Основаниями для снижения оценки являются:

- небрежное, непоследовательное выполнение, неполнота и нерациональность решения (исследования),
- необоснованное и некорректное применение методов решения,
- некорректность результатов, некорректная обработка результатов измерений,
- низкое качество графического материала (неверный выбор масштаба чертежей, отсутствие указания единиц измерения на графиках),
- неполнота отчета,
- низкое качество оформления отчета,
- низкая содержательность и низкое качество устного изложения материала
- некорректность и неполнота ответа на дополнительные вопросы,
- отсутствие необходимых разделов.

Оценка / Уровень	**Критерий**
«5» (отлично) / Высокий	выполнены все задания, обучающийся четко и без ошибок ответил на все контрольные вопросы
«4» (хорошо) / Средний	выполнены все задания; обучающийся ответил на все контрольные вопросы с замечаниями
«3» (удовлетворительно) / Низкий	выполнены все задания с замечаниями; обучающийся ответил на все контрольные вопросы с замечаниями
«2» (неудовлетворительно) / Неудовлетворительный	обучающийся не выполнил или выполнил неправильно задания; обучающийся ответил на контрольные вопросы с ошибками или не ответил на контрольные вопросы

ПРАКТИЧЕСКАЯ РАБОТА №1. ОСОБЕННОСТИ ИНФРАСТРУКТУРЫ СИСТЕМ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

Форма проведения

решение проблемной задачи.

Цель работы

изучение процедуры развертывания современных систем искусственного интеллекта (MLOps).

Задачи

- проанализировать существующие MLOps платформы;
- сформировать критерии сравнения MLOps и произвести оценку имеющихся платформ;
- изучить и установить MLOps платформу;
- освоить выполнение базовых операций в рамках платформы.

Описание

Работа выполняется в группах по 2–3 человека. Включает в себя выполнение поискового исследования, сравнения и экспериментальное изучение.

Ход выполнения

1. Изучите существующие на рынке MLOps платформы и выберите те, которые являются open-source либо имеют бесплатный тестовый период для свободного использования.
2. Изучите описания данных платформ.
3. Выберите подмножество из 10 платформ, ориентируясь на их особенности, преимущества и ограничения.
4. Сформулируйте критерии для сравнения платформ, включающие в себя стоимостные и технические критерии оценки.
5. На основе данных критериев выберите 1 из наилучших по вашему мнению платформ.
6. Установите платформу и исследуйте доступный функционал (рекомендуется использование документации на сайте платформы).
7. Используя платформу, создайте и запустите конвейер МО, реализующий следующие элементы: ресурс, загружающий перечень новостей через API; этап классификации данных;

управление расписаниями запуска; сохранение данных и артефактов моделей.

Варианты заданий

Необходимо самостоятельно произвести выбор доступной MLOps платформы. Преподаватель может привести перечень платформ (например):

- dagster
- ClearML
- cnvrg.io
- MLFlow
- Metaflow
- Kubeflow
- Flyte
- Seldon
- Iguazio

Преподаватель контролирует возможность выбора одинакового варианта заданий различными группами.

ПРАКТИЧЕСКАЯ РАБОТА №2. ПРИНЦИПЫ ПОСТРОЕНИЯ КИБЕРУСТОЙЧИВЫХ СИСТЕМ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

Форма проведения

решение проблемной задачи.

Цель работы

изучить алгоритмы построения защищенных и киберустойчивых систем искусственного интеллекта (MLSecOps).

Задачи

- разработка типового приложения с использованием ML;
- внедрение в приложение поддержки Cloud Native и принципов 12-факторного приложения;
- использование CI/CD;
- настройка средств безопасности DevSecOps.

Описание

Работа выполняется в группах по 2–3 человека. Включает в себя программирование приложения, использование сервиса GitHub и эксплуатация программных средств обеспечения безопасности DevSecOps.

Ход выполнения

1. Установить Linux/MacOS/WSL
2. Установить и настроить следующее ПО: git, IDE (рекомендуется vscode), Docker, Docker compose
3. Установить Jupyter Notebook
4. Зарегистрироваться на GitHub
5. Разработать MVP приложения «Счетчик посещений»
    - Ручки /visit/itmo.ru и /show/itmo.ru
    - Хранение счетчиков в Redis
6. Реализация фактора №1 – Кодовая база (Codebase)
7. Создать репозиторий на GitHub, содержащий исходный код приложения
    - Файл Python
    - Инструкцию в файле README.md
8. Доработать приложение «Счетчик посещений»
    - Фактор №2 – dependencies
    - Фактор №3 – config

- Фактор №4 – Backing services
- Добавить .gitignore
- Добавить Dockerfile
- Добавить .dockerignore

9. Освоить линтер
    - Выбрать свой линтер
    - Сформировать явным образом конфигурацию правил
    - Добавить в README.md
10. Доработать приложение «Счетчик посещений»
    - Фактор №5 – Build, release, run
    - Фактор №10 – Dev/prod parity
    - Фактор №11 – Logs
11. Добавить GitHub Action
    - Проверка Python синтаксиса с помощью linter’a
    - Сборка Dockerfile (проверка корректности)
    - Настройка Dependabot и dependabot.yml

Варианты заданий

Допускается выполнение идентичных заданий, но требуется реализация минимального приложения с нуля.

При необходимости возможно для каждого исполнителя сформулировать отдельное техническое задание на реализацию приложения и выбрать отдельный набор факторов для реализации.

ПРАКТИЧЕСКАЯ РАБОТА №3. ИНСТРУМЕНТАЛЬНЫЕ СРЕДСТВА ОБЕСПЕЧЕНИЯ КИБЕРУСТОЙЧИВОСТИ СИСТЕМ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

Форма проведения

решение проблемной задачи.

Цель работы

изучение инструментальных систем обеспечения защищенности и киберустойчивости систем искусственного интеллекта (MLSecOps).

Задачи

- проанализировать существующие MLSecOps инструментов;
- сформировать критерии категоризации инструментов и произвести группировку имеющихся инструментов;
- изучить и установить MLSecOps инструмент;
- освоить выполнение базовых операций в рамках данного инструмента.

Описание

Работа выполняется в группах по 2–3 человека. Включает в себя выполнение поискового исследования, категоризацию и экспериментальное изучение.

Ход выполнения

1. Изучите существующие MLSecOps инструменты из перечня (https://github.com/RiccardoBiosas/awesome-MLSecOps) и выберите те, которые являются open-source либо имеют бесплатный тестовый период для свободного использования.
2. Изучите описания данных инструментов.
3. Сформулируйте критерии для категоризации (не менее 5) инструментов и соотнесите каждый из инструментов с соответствующей категорией.
4. Выберите один инструмент в соответствии с выданным вариантом.
5. Установите инструмент и исследуйте доступный функционал (рекомендуется использование документации).
6. Используя данный инструмент, создайте и запустите демонстрационную версию возможностей.

7. Представьте отчет, содержащий необходимые шаги для выполнения данных действий и скриншоты выполнения каждого действия.

Варианты заданий

Рекомендуется давать студентам на выбор инструменты со страницы https://github.com/RiccardoBiosas/awesome-MLSecOps.

Допустимо использовать и другие инструменты на усмотрение преподавателя. Преподаватель контролирует возможность выбора одинакового варианта заданий различными группами.

ПЕРЕЧЕНЬ ВОПРОСОВ И РЕКОМЕНДАЦИИ ДЛЯ ПРОВЕДЕНИЯ ПРОМЕЖУТОЧНОЙ АТТЕСТАЦИИ

Рекомендации по методике оценивания

Промежуточная аттестация по дисциплине осуществляется с помощью следующих оценочных средств: собеседование по билетам.

- формат проведения – устный зачет в формате ответов на вопросы билета;
- время, отводимое на подготовку ответа – 1 час;
- требования к ответу – полное раскрытие темы вопросов, наличие и разбор практического примера по теме каждого вопроса;
- порядок формирования билета — два случайных вопроса из перечня вопросов.

Шкала оценивания и критерии оценки

Критерии оценки	Минимальное количество баллов	Максимальное количество баллов
1. Уровень усвоения материала, предусмотренного программой	3	5
2. Умение выполнять задания, предусмотренные программой	3	5
3. Уровень раскрытия причинно-следственных связей	1,2	2
4. Уровень раскрытия междисциплинарных связей	1,2	2
5. Качество ответа (его общая композиция, логичность, общая эрудиция)	1,2	2
6. Изложение: полнота, аргументированность	1,8	3
7. Деловые и волевые качества: ответственное отношение к работе, стремление к достижению высоких результатов.	0,6	1
Итого:	**12**	**20**

Оценка		Минимальное количество баллов	Максимальное количество баллов
«5» (отлично)	Зачтено	18	20
«4» (хорошо)		15	17
«3» (удовлетворительно)		12	14
«2» (неудовлетворительно)	Не зачтено	0	11

Знания, умения и навыки обучающихся при промежуточной аттестации определяются оценками «зачтено (отлично)», «зачтено (хорошо)», «зачтено (удовлетворительно)», «не зачтено (неудовлетворительно)».

«Зачтено (отлично)» – обучающийся глубоко и прочно усвоил весь программный материал, исчерпывающе, последовательно, грамотно и логически стройно его излагает, не затрудняется с ответом при видоизменении задания, свободно справляется с задачами и практическими заданиями, правильно обосновывает принятые решения, умеет самостоятельно обобщать и излагать материал, не допуская ошибок.

«Зачтено (хорошо)» – обучающийся твердо знает программный материал, грамотно и по существу излагает его, не допускает существенных неточностей в ответе на вопрос, может правильно применять теоретические положения и владеет необходимыми умениями и навыками при выполнении практических заданий.

«Зачтено (удовлетворительно)» – обучающийся усвоил только основной материал, но не знает отдельных деталей, допускает неточности, недостаточно правильные формулировки, нарушает последовательность в изложении программного материала и испытывает затруднения в выполнении практических заданий.

«Не зачтено (неудовлетворительно)» – обучающийся не знает значительной части программного материала, допускает существенные ошибки, с большими затруднениями выполняет практические задания, задачи.

Перечень вопросов

1. Жизненный цикл деплоймента систем искусственного интеллекта
2. Управление ресурсами в системах искусственного интеллекта
3. Понятие и основные принципы обеспечения киберустойчивости
4. Инструменты логирования и анализа событий систем искусственного интеллекта
5. Архитектура безопасности интеллектуальных систем
6. Тестирование безопасности инфраструктуры систем искусственного интеллекта
7. Основные термины и определения кибербезопасности
8. Киберустойчивость. Цели, задачи и техники для построения киберустойчивых систем
9. Уязвимости и угрозы, оценка киберустойчивости и киберучения
10. Роли и особенности в системах МО и ИИ
11. Архитектура современных интеллектуальных систем

12. Особенности архитектуры разработки и МО
13. Cloud native
14. Методология 12 factor apps
15. Упущенные факторы из методологии 12 factor app
16. DevOps
17. DevSecOps
18. Введение в концепцию MLOps
19. Уровни зрелости MLOps
20. Составные этапы MLOps
21. MLOps компоненты
22. Подходы к защите MLOps
23. Защита данных в MLOps
24. Принципы киберустойчивости на этапе проектирования и разработки
25. Принципы киберустойчивости на этапе внедрения
26. Принципы киберустойчивости на этапе эксплуатации и консервации
27. Применение Kubeflow для оркестрации систем ИИ
28. Потоковая обработка данных в системах ИИ
29. Hardening систем ИИ

Пример билета

1. Вопрос №1: «Понятие и основные принципы обеспечения киберустойчивости»;
2. Вопрос №2: «Методология 12 factor apps».

ВОПРОСЫ ДЛЯ САМОКОНТРОЛЯ

В данном разделе представлены дополнительные вопросы тестового формата, которые могут использоваться для самоконтроля освоения материалов курса. Приведены варианты ответов. Вопросы рекомендуется загрузить в систему управления обучением.

Введение в кибербезопасность и киберустойчивость

1. Подход к проверке безопасности системы, при котором проверяющему неизвестно ее внутреннее устройство
   a. blackbox
   b. redbox
   c. greybox
   d. greenbox
2. Угрозой какой составляющей информационной безопасности является DDoS-атака?
   a. конфиденциальность
   b. целостность
   c. доступность
   d. актуальность
3. Что такое уязвимость нулевого дня?
   a. уязвимость, о которой еще никому не известно
   b. уязвимость, связанная с неверной обработкой даты и времени
   c. уязвимость, для которой не существует исправления
   d. уязвимость, для которой выпущен патч или обновление безопасности
4. Какой идентификатор присваивают уязвимости в каком-либо конкретном продукте или системе?
   a. cwe
   b. cve
   c. mitre
   d. cvss
5. Что помимо самой методики оценки киберустойчивости содержится в Cyber Resilence Rewiew?
   a. Руководства по внедрению необходимых практик

b. Перечень top-10 уязвимостей

c. Правильные ответы на опросный лист

Архитектура и жизненный цикл интеллектуальных систем

1. В чем же заключается основное отличие систем искусственного интеллекта по сравнению с традиционными системами?
   a. Производительность ИИ систем выше
   b. Требование управлять и версионировать данные
   c. Повышенные требования к безопасности
   d. Появление роли DevOps инженера
2. Зачем нужен балансировщик нагрузки?
   a. Для горизонтального масштабирования
   b. Для вертикального масштабирования
   c. Для обеспечения безопасности и конфиденциальности передачи данных
   d. Для повышения целостности
3. Как называется технология, которая работает путем распределения контента по множеству «пограничных» серверов по всему миру, так что пользователи в итоге загружают активы с «пограничных» серверов, а не с исходного сервера?
   a. DNS
   b. TLS
   c. CDN
   d. SDN
4. Что такое ETL система?
   a. ETL (Extract, Transform, Load) - Извлечение, преобразование, загрузка
   b. ETL (Extract, Transfer, Load) - Извлечение, передача, загрузка
   c. ETL (Extract, Translate, Load) - Извлечение, перевод, загрузка
   d. ETL (Extract, Transcribe, Load) - Извлечение, транскрипция, загрузка
5. Что из нижеприведенного является минусом микросервисной архитектуры?
   a. Сложно комбинировать разные технологии разработки
   b. Сложно часто выпускать новые релизы

c. Сложно отслеживать данные, проходящие через различные границы

d. Сложно горизонтально масштабировать

Cloud Native

1. Какие сервисы могут выступать в качестве примеров модели обслуживания "Домашних животных" (pets model)?
   a. Одиночные серверы
   b. Массивы веб-серверов
   c. Поисковые кластеры
   d. Хранилища данных с несколькими мастер-нодами
2. Сколько кодовых баз и развертываний может существовать для каждого приложения?
   a. Любое количество
   b. Только 1 кодовая база и 1 развертывание
   c. Только 1 развертывание и любое количество кодовых баз
   d. Только 1 кодовая база и любое количество развертываний
3. Что необходимо сделать приложению, чтобы не зависеть от неявно существующих зависимостей?
   a. Подключить репозиторий с перечнем библиотек
   b. Автоматизировать загрузку зависимостей с помощью bash скрипта
   c. Внедрить манифест декларации зависимостей
   d. Скопировать виртуальную машину с настроенным окружением
4. Что не входит в конфигурацию приложения?
   a. Идентификаторы подключения к ресурсам БД
   b. Регистрационные данные для сторонних API
   c. Значения зависимые от среды развертывания
   d. Все вышеперечисленное входит
5. Как называется принцип, который направлен на устранение различий во времени, различий персонала и различий инструментов?
   a. Сторонние службы
   b. Паритет разработки и исполнения
   c. Утилизируемость

d. Сборка, релиз, выполнение

DevOps и DevSecOps

1. Какие проверки использует Kubernetes в рамках фактора «Наблюдаемость»?
   a. Rediness и Stableness
   b. Liveness и Rediness
   c. Secureness и Liveness
   d. Stableness и Secureness
2. Как называются методы, которые автоматизируют процесс выпуска программного обеспечения, начиная от сборки и вплоть до развертывания?
   a. Continuous integration и Continuous teaching
   b. Continuous integration и Continuous delivery
   c. Continuous delivery и Continuous deploy
   d. Continuous interaction и Continuous deploy
3. Какая методология является логическим продолжением подхода «инфраструктура как код»?
   a. CI/CD
   b. Helm
   c. GitOps
   d. CI/CD/CT
4. Зачем нужны SDL или SDLC?
   a. Ускорение сборки и загрузки
   b. Улучшение customer experience
   c. Участие безопасности на каждом этапе разработки
   d. Формирование требований security compliance
5. Как называется процесс, который автоматизирует обеспечение видимости использования программного обеспечения с открытым исходным кодом с целью управления рисками, обеспечения безопасности и соответствия лицензиям?
   a. SAST
   b. CE
   c. IAST

d. SCA

MLOps

1. Что означает термин Time To Market (TTM)?
   a. Время от окончания работ по техническому заданию (ТЗ) до выхода продукта на рынок
   b. Время от появления идеи до появления первого рабочего экземпляра (MVP)
   c. Время от начала разработки идеи фичи или продукта до её конечной реализации, когда вы продаете её клиенту
   d. Время от появления первого рабочего экземпляра (MVP) до получения прибыли от реализации
2. В вашей команде 3 человека: менеджер, инженер технической поддержки и DevOps. Чему равен bus factor вашего GitOps оператора?
   a. 1
   b. 2
   c. 3
   d. 4
3. Что такое CT?
   a. Это свойство, связанное с тестированием и проверкой кода и компонентов
   b. Это свойство, связанное с автоматическим переобучением и обслуживанием моделей
   c. Это свойство, связанное с тестированием и проверкой данных, схем данных и моделей
   d. Это свойство, связанное с системой автоматического развертывания сервисов
4. На каком уровне зрелости организациям еще предстоит найти место для МО, не говоря уже о MLOps.?
   a. На нулевом
   b. На первом
   c. На втором
   d. На третьем
5. Какой уровень в модели Microsoft Azure соответствует тому, что все операции автоматизированы?

a. 2
b. 3
c. 4
d. 5

Безопасность MLOps

1. Какую задачу не решают платформы данных?
   a. Контроль изменений
   b. Версионирование данных
   c. Автоматическая разметка
   d. Переобучение моделей
2. Какая из систем вероятнее всего допускает использование пакетного вывода результатов?
   a. Система отслеживания мошеннических операций в ДБО
   b. Система подсчета всех новых лидов, поступающих в отдел продаж
   c. Система учета пробок на дорогах
   d. Система оперативного реагирования на события информационной безопасности
3. Как называется принцип, согласно которому можно разделить данные по нескольким категориям: общедоступные, внутренние, конфиденциальные?
   a. Цель хранения
   b. Окружение
   c. Чувствительность
   d. Классификация
4. Какой из векторов атак не относится к атакам на данные?
   a. Отравление данных
   b. Компрометация инфраструктуры приложения
   c. Утечка от инсайдеров
   d. Все относятся
5. Какой из следующих процессов отличается наивысшими требованиями к SLA?
   a. Система реального времени
   b. Система обновления сводных дашбордов

c. Система резервного копирования

d. Все одинаковые

Принципы киберустойчивости систем ИИ

1. Какой из приёмов не поможет при составлении системных требований устойчивости предлагаемой системы к присущим ИИ/МО угрозам?

   a. Внедрение Red teaming

   b. Обучение сотрудников

   c. Понижение bus factor

   d. Поощрение позитивной культуры безопасности

2. Какой принцип наиболее точно характеризуется уменьшением раскрытия подробной информации о системе и защиты наборов данных и библиотек?

   a. Проектирование от безопасности

   b. Подключение разработчиков

   c. Минимизация знаний противника

   d. Защита цепочек поставок

3. Как можно описать суть принципа «подключения разработчиков»?

   a. Вы осознаете риски, когда вы покупаете или используете модели, которые не имеют прозрачного метода разработки

   b. Вы понимаете важность анализа влияния на безопасность всех проектных решений на протяжении всего процесса разработки

   c. Вы разрабатываете процессы для выявления и/или исправления ошибок в случае непреднамеренного сбоя или по вине злоумышленника

   d. Вы обеспечиваете, чтобы сотрудники были знакомы с уязвимостями и режимами сбоев

4. Атаки на какую сущность включают в себя «атаки уклонения», «атаки извлечения» и «инверсионные атаки»?

   a. Данные

   b. Модель

   c. Инфраструктура

   d. Цепочки поставок

5. Какой процесс характеризует следующее утверждение: «Оно может проявляться во многих формах, когда на поведение вашей модели влияют данные, собранные во время работы (например, добавление новой точки данных в алгоритм k ближайших соседей во время его развертывания)»?
   a. Непрерывная интеграция
   b. Непрерывное развертывание
   c. Непрерывная обработка
   d. Непрерывное обучение

Специальные инструменты киберустойчивости

1. Для чего используется утилита kind?
   a. Для развертывания CI/CD
   b. Для поднятия локального k8s
   c. Для интеграции Kubeflow с облаком k8s
   d. Для тестирования защищенности k8s
2. Какие операторы не поддерживает Kubeflow?
   a. Pytorch
   b. Tensorflow
   c. PaddlePaddle
   d. Все вышеперечисленные поддерживаются
3. Как называется фреймворк и движок для statefull вычислений над неограниченными и ограниченными потоками данных?
   a. ElasticSearch
   b. Apache Cassandra
   c. Apache Flink
   d. Apache Kafka
4. Какой из следующих принципов не относится к концепции нулевого доверия (zero trust)?
   a. Аутентификация и авторизация при каждой попытке доступа
   b. Микросегментация
   c. Резервное копирование
   d. Непрерывный мониторинг

5. Типичный post-mortem начинается с регистрации объективных доказательств. Какое доказательство из перечня является наименее оперативно значимым?
   a. Что вызвало инцидент
   b. Какой урон он нанес
   c. Время его обнаружения и нейтрализации
   d. Как заинтересовать руководство исправить уязвимость

СПИСОК ЛИТЕРАТУРЫ

1. Кибербезопасность, киберустойчивость, киберучения [Электронный ресурс] / Режим доступа: https://www.securityvision.ru/blog/razbor-terminologii-kiberbezopasnost-kiberustoychivost-kiberucheniya-chto-eto/ (дата обращения: 02.03.2024)
2. Чио, К. Машинное обучение и безопасность : руководство / К. Чио, Д. Фримэн ; перевод с английского А. В. Снастина. — Москва : ДМК Пресс, 2020. — 388 с. — ISBN 978-5-97060-713-8. — Текст : электронный // Лань : электронно-библиотечная система. — Режим доступа: https://e.lanbook.com/book/131707 (дата обращения: 02.03.2024)
3. Engineering Trustworthy Secure Systems [Электронный ресурс] / Режим доступа: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-160v1r1.pdf (дата обращения: 02.03.2024)
4. Херинг, М. DevOps для современного предприятия : учебное пособие / М. Херинг ; перевод с английского М. А. Райтмана.. — Москва : ДМК Пресс, 2020. — 232 с. — ISBN 978-5-97060-836-4. — Текст : электронный // Лань : электронно-библиотечная система. — Режим доступа: https://e.lanbook.com/book/140580 (дата обращения: 02.03.2024)
5. Treveil M. Introducing MLOps: How to Scale Machine Learning in the Enterprise. — O'REILLY, 2020. — Режим доступа: https://www.amazon.co.uk/dp/1492083291
6. Ишутин, А. А. основные понятия и принципы веб-архитектуры / А. А. Ишутин, А. С. Пантюхов // Программная инженерия: современные тенденции развития и применения (пи-2020) : сборник материалов IV Всероссийской научно-практической конференции, посвященной 30-летию создания кафедры программной инженерии, Курск, 12–13 марта 2020 года. – Курск: Юго-Западный государственный университет, 2020. – С. 276-280.
7. Грувер, Г. Запуск и масштабирование DevOps на предприятии / Г. Грувер. — Москва : ДМК Пресс, 2018. — 80 с. — ISBN 978-5-97060-704-6. — Текст : электронный // Лань : электронно-библиотечная система. — Режим доступа: https://e.lanbook.com/book/116130 (дата обращения: 02.03.2024)
8. MLOps: Continuous delivery and automation pipelines in machine learning [Электронный ресурс] / Режим доступа: https://cloud.google.com/architecture/mlops-continuous-delivery-and-automation-pipelines-in-machine-learning (дата обращения: 02.03.2024)
9. Гридчин, В. С. Методология управления машинным обучением - MLOps / В. С. Гридчин, Н. А. Чаплыгин, В. А. Балаев //

Фундаментальные и прикладные научные исследования: актуальные вопросы, достижения и инновации : сборник статей L Международной научно-практической конференции, Пенза, 15 ноября 2021 года. – Пенза: Наука и Просвещение (ИП Гуляев Г.Ю.), 2021. – С. 43-45

10. Чесалин, А. Н. Основы искусственного интеллекта с приложениями в информационной безопасности. Практикум : учебное пособие / А. Н. Чесалин. — Москва : РТУ МИРЭА, 2020. — 75 с. — Текст : электронный // Лань : электронно-библиотечная система. — Режим доступа: https://e.lanbook.com/book/163838 (дата обращения: 02.03.2024)
11. The MLSecOps Top 10 [Электронный ресурс] / Режим доступа: https://ethical.institute/security.html (дата обращения: 02.03.2024)

Менщиков Александр Алексеевич

Кармановский Николай Сергеевич

Киберустойчивость систем искусственного интеллекта

Учебно-методическое пособие

В авторской редакции
Редакционно-издательский отдел Университета ИТМО
Зав. РИО Н.Ф. Гусарова
Подписано к печати
Заказ №
Тираж
Отпечатано на ризографе

Редакционно-издательский отдел
Университета ИТМО
197101, Санкт-Петербург, Кронверский пр., 49, литер А