Модуль 1. Введение.

Содержание модуля:

В этом модуле вы познакомитесь с дисциплиной «Автоматизация машинного обучения», или, как чаще называют эту методологию на практике, MLOps. Информации в модуле будет достаточно для понимания большинства задач дисциплины и методов их решения, даже если вы раньше никогда не слышали о DevOps/MLOps. Поскольку это общее предварительное знакомство, многие сведения изложены обзорно, чтобы сформировать предварительное представление о дисциплине. Более подробно эти сведения изучаются в следующих модулях курса.

В этом модуле слушатели:

* узнают как появилась методология MLOps и почему она так востребована,
* ознакомятся с терминологией MLOps, базовыми понятиями и методами,
* изучат важные аспекты управления проектами машинного обучения: жизненный цикл проекта, состав команды и роли участников проекта,
* познакомятся с наиболее популярными инструментами для решения задач MLOps,
* установят важную взаимосвязь между техническими требованиями к проекту машинного обучения и содержанием задач MLOps.

Изучение любой дисциплины не начинается “с нуля”, поэтому в модуле описывается перечень знаний, необходимых для быстрого и эффективного освоения излагаемого в курсе материала.

Знания, полученные при изучении этого модуля, создадут хорошую основу для понимания информации в следующих модулях. Удачи!

# 

# Модуль 1. Юнит 1: Зачем надо изучать MLOps и что для этого требуется?

*В этом юните:*

Слушатели узнают почему знание методологии MLOps очень востребовано, что необходимо изучать, чтобы стать специалистом MLOps, и какими навыками необходимо обладать, чтобы обучение было наиболее эффективным.

*Содержание юнита:*

Индустрия программного обеспечения, появившаяся в середине XX века, в настоящее время является важной производственной отраслью. Все чаще информационные системы и искусственный интеллект заменяют человеческий труд. Цифровые технологии успешно заменяют человека в таких задачах как обработка документов, управление производством и процессами, консультирование, перевод и даже создание произведений искусства. И дальше эта тенденция будет только усиливаться. “Did you know 2019 (shift happends)” <https://www.youtube.com/watch?v=TwtS6Jy3ll8>. Вот как выглядит, например, картина, “нарисованная” моделью машинного обучения ruDALL-E, нейронной сетью для генерации картинок (https://developers.sber.ru/portal/news/rudall-e-03-11-2021), по запросу “берег моря с парусником”:



<https://rudalle.ru/demo> (https://creativecommons.org/licenses/by/4.0/deed.ru)

Программирование и программная инженерия, разработка программного обеспечения, создание сложных комплексных информационных систем - без этого почти невозможно представить современное предприятие. Задачи и методы в разработке программных систем, постоянно развиваются.

**Основная задача в проектах создания программных систем (как, впрочем, и в любом проекте) это скорейшая реализация проекта с соблюдением имеющихся ограничений на ресурсы.**

Как оказалось, в разработке программного обеспечения есть особенности, которые сильно отличают такие проекты от проектов в промышленности или на производстве. *Например, если увеличить команду разработчиков вдвое, то это практически никогда не приведет к сокращению сроков, а даже часто наоборот - увеличит сроки.* Причина в том, что новую команду надо вводить в курс дела, обучать, настраивать среду для работы. Появляются дополнительные накладные расходы, связанные с ошибками новых участников. Традиционная шутка программистов: “один программист будет делать работу один месяц, а два программиста будут делать ту же работу два месяца”.

Необходимость ускорения вывода продукта на рынок привело к созданию специальных инструментов и методов разработки программного обеспечения:

* интегрированная среда разработки, Integrated Development Environment, IDE,
* стандартные библиотеки, реализующие типовые функции,
* микросервисная архитектура и типовые интерфейсы взаимодействия,
* использование продуктов с открытым программным кодом, которое разрабатывает большое сообщество,
* создание шаблонов (паттернов) для проектирования и разработки систем,
* специальные организационные методы для командной разработки (agile), предполагающие более ранний показ “незрелого” продукта заказчику (а иногда и запуск в промышленную среду) для получения более быстрой обратной связи и ее учет в следующих итерациях процесса разработки.

Эти и многие другие нововведения существенно ускорили вывод продукта разработки на рынок с сохранением приемлемого качества, однако также добавили накладные расходы проекта. Например,

1. скорость выхода продукта сопряжена с появлением так называемого “технического долга” — перечня функций, которые необходимо разработать для нормальной эксплуатации продукта, после его вывода в эксплуатацию,
2. использование готовых библиотек, веб-серверов, систем управления очередями, баз данных существенно ускоряет процесс разработки, однако создает проблему совместимости компонентов между собой,
3. микросервисная архитектура предполагает полную независимость отдельных компонентов общего решения и их разработчиков, вследствие этого “улучшения”, введенные разработчиками одного компонента, могут стать катастрофическими для всей системы целиком, если другие компоненты окажутся несовместимыми с новой версией.

Подробнее об этом можно прочитать в книге Ф.Брукса “Мифический человеко-месяц или Как создаются программные системы”, которая была написана в прошлом веке, но остается актуальной и в настоящий момент.



https://www.ozon.ru/product/mificheskiy-cheloveko-mesyats-ili-kak-sozdayutsya-programmnye-sistemy-83760/?sh=ycWDzgAAAA

Из всего этого следует, что любой улучшающий подход или инструмент необходимо применять аккуратно, постоянно анализируя его влияние на общий процесс. Мониторинг и автоматизация различных этапов разработки как раз и предназначены для того, чтобы избежать описанных выше проблем. Вот методы, которые помогают в борьбе с хаосом в разработке программного обеспечения:

* контроль изменений в программном обеспечении на уровне изменений кода,
* контроль версий (версионирование), управление разными ветками программного обеспечения в разработке,
* автоматизация сборки всей разрабатываемой системы из отдельных модулей, над которыми работают разные команды,
* автоматизация тестирования,
* автоматизация установки разработанной системы на оборудовании для промышленной эксплуатации, включая автоматическую настройку инфраструктуры для развертывания системы,
* непрерывный мониторинг качества на всех этапах.

**Создание и использование этих методов в процессе разработки программного обеспечения и привело к появлению методологии DevOps, являющейся связующим элементом между разработкой (Development, Dev) и эксплуатацией (Operations, Ops).**

Все вышеизложенное относится к любым проектам, в которых создается программный код, и, в том числе, к проектам машинного обучения. Работа с большими данными и машинное обучение (Machine Learning, ML) сравнительно недавно стали популярным инструментом в программных системах для решения производственных задач. Причины, по которым методы, разработанные еще во второй половине XX века, начали внедряться в практику сейчас, следующие:

* увеличение производительности вычислительных систем,
* появление надежных и высокопроизводительных технологий для сбора, передачи, обработки и хранения данных,
* удешевление ресурсов для хранения данных и вычислений,
* повышение сложности информационных систем и их взаимодействия между собой, требующее автоматизации и использования интеллектуальных алгоритмов.

Действующий закон Мура косвенно описывает процесс постоянного уменьшения стоимости вычислительного оборудования и увеличения скорости вычислений. *Например, стоимость расшифровки генома человека (сложная вычислительная задача, позволяющая выявить риск наследственных заболеваний на основе генетики), снизилась на порядки за несколько десятков лет https://vademec.ru/news/2015/10/02/stoimost\_rasshifrovki\_genoma\_snizilas\_do\_1\_000.*

Появившиеся вычислительные возможности привели к взрывному (точнее, экспоненциальному) росту использования различных моделей машинного обучения, в том числе и глубокого обучения (Deep Learning, DL) на основе нейронных сетей. Разработанные алгоритмы машинного обучения и их использование в программных системах дали замечательные практические результаты, которые ранее невозможно было получить: алгоритмы начали распознавать изображения лучше человека, научились играть в шахматы и го, начали управлять производственными процессами и многое другое. Однако у решений на основе машинного обучения и больших данных есть и обратная сторона. Оказывается, что новые технологии и алгоритмы привели к новым проблемам, которые необходимо учитывать и решать при реализации проекта. Это связано с тем, что в машинном обучении есть отличия от обычных проектов разработки программного обеспечения, а именно:

* наборы данных (датасеты) оказывают сильное влияние на качество работы всей системы, при этом имеют свойство существенно меняться и на практике часто содержат ошибки,
* в проекте невозможно создать идеальное решение для продолжительного использования, практически сразу после вывода в продакшн модели машинного обучения начинают постепенно деградировать,
* модели машинного обучения очень чувствительны к окружению, в котором работают, небольшие различия в окружении для обучения и тестирования модели и в продакшн могут привести к кардинальным различиям в результатах работы модели,
* экспериментальный характер системы ведет к необходимости постоянного дообучения модели на основе новых данных или при изменении окружения.

Чтобы понять с какими сложностями может столкнуться специалист MLOps давайте посмотрим на этапы типового проекта машинного обучения, которые схематично приведены на рисунке ниже









Предсказание на промышленных данных

Вывод модели в эксплуатацию



На этой схеме этапы проекта приведены укрупненно, более детально они будут описаны в последующих модулях. Но уже на этой схеме можно сформулировать и описать те дополнительные сложности, с которыми придется столкнуться при реализации проекта ML:

|  |  |  |
| --- | --- | --- |
| Этап | Задачи этапа | Возможные проблемы |
| Подготовка данных | Получение и анализ данных, устранение ошибок.  Передача и хранение данных.  Организация доступа к данным.  Обезличивание данных.  Обогащение данных.  Формирование обучающего и проверочного датасетов. | Много разрозненных датасетов, на которых модель может показывать разные результаты.  Изменение структуры и характеристик данных со временем. |
| Выбор ML алгоритма, решающего задачу | Изучение различных алгоритмов машинного обучения, поиск эффективного. | Много разных алгоритмов, у каждого свои параметры.  Необходимо учитывать ограничения: аппаратные ресурсы, требования к качеству. |
| Обучение модели | Модель ML обучается с использованием набора данных для обучения (train dataset, обучающий датасет). | Результаты обучения модели могут сильно отличаться при незначительном изменении в данных или программном окружении (набор и версии библиотек) |
| Внедрение модели в продуктивную среду | Разработанная модель выводится в промышленную эксплуатацию (как отдельный микросервис или как часть более крупной системы) | Обеспечение совместимости среды, в которой эксплуатируется модель, с моделью.  Взаимодействие модели с другими компонентами. |
| Предсказания на промышленных данных | В среде эксплуатации модель получает новые данные, в том числе и те, которые она ранее не видела. | Новые данные могут отличаться от данных, на которых обучали модель.  Неустойчивая работа модели даже при незначительных изменениях характеристик среды выполнения программы или в данных.  В крайних случаях это может привести к краху модели (проблема повторяемости результатов). |
| Анализ качества | Контроль качества обучения. Принятие решения о переобучении модели. | Методы машинного обучения представляют собой «черный ящик», логику работы которого сложно объяснить (проблема интерпретируемости результатов).  С течением времени данные изменяются, что приводит к ухудшению качества работы модели. |

В совокупности эти проблемы приводят к тому, что даже у самой хорошей идеи, реализованной на уровне гипотезы, возникают сложности в практической эксплуатации. Статистика проектов, использующих большие данные и машинное обучение, говорит о том, что большинство (более 80%) проектов не попадают в промышленную эксплуатацию (production).



Картинка из курса LEAN\_DS в сообществе ODS.

Проблемы весьма серьезные, но если бы это останавливало команды разработчиков проектов машинного обучения, то мы никогда бы не увидели замечательные проекты с использованием технологий распознавания изображений, переводчики текстов, интеллектуальные транспортные системы и беспилотные автомобили. Зная о потенциальных опасностях можно предпринять меры по их недопущению и одним из способов сделать это является правильная организация инфраструктуры и процесса работы над проектом машинного обучения. Собственно, этими задачами и занимается специалист MLOps. Итак, мы пришли к пониманию, что роль MLOps инженера крайне важна для общего успеха проекта.

Как вы увидите в последующих модулях курса, от MLOps инженера требуется большая универсальность в знаниях и навыках, но перечень необходимых знаний может сильно зависеть от проекта и команды. В ходе работы MLOps инженеру могут понадобиться навыки:

* настройки и администрирования аппаратного обеспечения,
* программирования python (и некоторых других языков), понимание принципов работы окружения этих языков (компиляторов, интерпретаторов, служебных библиотек, установщиков и пр.)
* администрирования операционных систем linux, написания bash скриптов,
* создания средств виртуализации: виртуальные машины, контейнеры docker/kubernetes,
* настройка баз данных и написание SQL запросов,
* понимание механизмов работы и форматов API, http, JSON,
* знание архитектуры и протоколов сетей передачи данных,
* понимание методов машинного обучения и работы с большими данными,

и многое другое…

Кроме технологических знаний (так называемые “hard-skills” навыки) от MLOps инженера требуются и “soft-skills” навыки, например,

* понимание agile методов организации работы команд разработки (scrum, kanban),
* умение коммуницировать и доносить свою позицию до других участников команды,
* активность в поиске и устранении “слабых мест” в проекте,
* желание изучать новые технологии, в том числе относящиеся к смежным подразделениям.

В крупнейшем русскоязычном сообществе Open Data Science (https://ods.ai) есть отдельная ветка LeanDS (https://ods.ai/projects/leands), в которой изучаются вопросы применения agile-методов для проектов машинного обучения.

Для быстрого освоения материала и выполнения практических заданий очень важно иметь эти требуемые навыки, поэтому слушателям, которые не чувствуют себя уверенно в данных темах, рекомендуется изучить их подробнее самостоятельно.

*Итак, в данном юните вы узнали, что:*

* методологии DevOps/MLOps призваны ускорить процесс разработки и повысить качество проектов,
* на разных этапах проекта машинного обучения возникают различные проблемы, в решении которых помогает MLOps, например
  + различные результаты работы системы в тестовом и промышленно окружении
  + большое количество модулей и библиотек и их несовместимость между собой
  + необходимость выполнять итерации жизненного цикла проекта машинного обучения чаще, чем в обычных проектах разработки
  + и многие другие
* от инженера MLOps требуется большой кругозор и знание проекта, причем не существует исчерпывающего универсального списка, перечень знаний и навыков существенно зависит от команды и проекта, что мы увидим в дальнейшем.

Теперь вы знаете почему появились методологии DevOps и ее адаптация для проектов машинного обучения MLOps. Конечно, для решения задач в MLOps уже разработаны различные эффективные инструменты, библиотеки, скрипты, организационные подходы. Их перечень очень большой, большинство имеют статус свободно распространяемого программного обеспечения, выбор того или иного инструмента зависит от множества факторов (тип проекта, количество человек в команде, ограничения по ресурсам, опыт MLOps инженера). По сути, сложность инфраструктуры большого проекта машинного обучения сопоставима со сложностью комплексной технологической системы, например, автомобиля. Итак, самое время заглянуть “под капот” MLOps.

*Проверим:*

1. Что содержит аббревиатура MLOps
   1. **Machine Learning + Operations**
   2. Machine Learning + Opinions
   3. Maximum Learning + Operations
   4. Machine Lean + Operations
2. Отметьте причины, по которым в проектах разработки программного обеспечения добавились дополнительные риски, касающиеся машинного обучения
   1. **чувствительность модели машинного обучения к любому шуму в данных**
   2. малое количество статей и готовых решений
   3. **невозможность долговременного сохранения окружения, в котором эксплуатируется модель машинного обучения**
   4. использование сложных математических алгоритмов
3. Расположите этапы проекта машинного обучения в правильном хронологическом порядке
   1. сбор данных
   2. преобразование данных
   3. тренировка модели машинного обучения
   4. вывод в эксплуатацию
   5. предсказание на промышленных данных
   6. оценка качества работы
4. Поставьте в соответствие навыки, которыми должен обладать MLOps, и причины важности обладания этими навыками

|  |  |
| --- | --- |
| Навык | Причина |
| Программирование на C++ | Основной код разрабатываемой системы написан на этом языке |
| Администрирование linux | Большинство инструментов, которыми пользуется команда, реализованы этими средствами |
| SQL запросы | Для организации сбора данных |
| bash, python | Для разработки скриптов автоматизации процедур |
| Машинное обучение, метрики | Для своевременного отслеживания ухудшения качества работы системы |

# Модуль 1. Юнит 2: Из чего состоит MLOps?

*В этом юните:*

Слушатели узнают о задачах и инструментах MLOps более подробно.

*Содержание юнита:*

Идеология DevOps, на основе которой появился MLOps, предполагает непрерывность всех этапов процесса разработки от планирования системы до ее эксплуатации. И тут возникает самое интересное, современные так называемые гибкие (agile) подходы к управлению проектами разработки программного обеспечения предполагают не только то, что после эксплуатации сразу может следовать новое планирование, но и то, что таких повторений (итераций) может быть много, чем больше, тем лучше! Получается такой магический символ бесконечности (перевернутая “восьмерка” в математике является символом “бесконечности”, что символизирует постоянное повторение итераций цикла проекта), символизирующий постоянное улучшение процесса и повторение шагов: планирование (plan) - разработка кода (code) - сборка системы (build) - тестирование (test) - выпуск релиза (release) - развертывание (deploy) - эксплуатация (operate) - мониторинг (monitor)

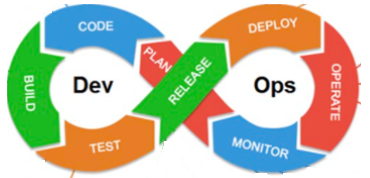


Схема жизненного цикла agile-проекта разработки программного обеспечения

При этом принимается тот факт, что на первых итерациях качество разработанного решения может быть явно недостаточным, однако **польза такого подхода состоит в том, что команда выводит решение как можно раньше и получает обратную связь от заказчиков, других подразделений, которую можно проанализировать и учесть в следующей итерации.** Продолжительность такой итерации и их число сильно зависят от конкретного проекта и отрасли, возможности испытывать предварительные версии на системе или заказчике, наличия ресурсов. Однако ясно, что раз эти действия являются повторяющимися и от скорости их выполнения зависит успех всего проекта, то это совершенно необходимо автоматизировать.

*Рассмотрим это на примере: молодая команда разработчиков web-сайтов испытывает большие сложности с получением точного технического задания от заказчика, маркетолога производственного предприятия, который хочет, чтобы “было все красиво”, но совершенно не готов обсуждать “верстку CSS”, используемые стили и скорость работы отдельных функций сайта. Отчаявшись, ребята сделали прототип сайта как посчитали более правильным, максимально учитывая те крохи информации, которую они получили от маркетолога. На быстро последовавшем показе маркетолог сказал: “Ну, наконец то до вас дошло что я хотел, только надо еще вот это и вот это”, и в последующих итерациях уже указанные конкретные проблемы были устранены. Для лучшего понимания этой ситуации наберите в youtube “семь перпендикулярных красных линий”.*

И, как вы уже наверное догадываетесь, для проекта машинного обучения эта ситуация становится еще более сложной. Проект машинного обучения содержит стандартные этапы, которые представлены на следующей диаграмме



Этапы проекта машинного обучения

На каждом из этапов существуют отдельные задачи, результатом каждого этапа является так называемый **артефакт**, который передается для обработки на следующий уровень.Рассмотрим подробнее эти этапы, попутно рассуждая какие артефакты возникают при выполнении этих этапов и какие задачи при этом возникают для специалиста MLOps.

1. Сбор данных

Сбор данных может осуществляться из различных разнородных источников с использованием различных средств передачи информации, например, служебных технологических протоколов (OPC, modbus), стандартных сетевых протоколов и средств (SNMP, syslog), интерфейсов баз данных (SQL, JSON). Для получения данных может потребоваться авторизация. Устройств для опроса может быть огромное количество.

Задачи MLOps:

* автоматизировать сбор информации, организовать опросы устройству по списку и расписанию, с учетом их интерфейсов взаимодействия,
* организовать мониторинг доступности источников данных и каналов передачи информации, формирование сообщений об ошибке в случае проблем с получением данных,
* создать систему для надежного хранения полученных данных, включая, при необходимости, резервирование и авторизованный доступ к данным.

Артефакты этапа:

* информация, собранная из источников данных и сохраненная в структурированном виде
* данные системы мониторинга

1. Предобработка данных

В данных почти всегда содержатся ошибки. Это может быть связано с ошибками передачи информации, выходом из строя источника данных, аномальным поведением окружающей среды или ошибкой, связанной с человеческим фактором. Ошибки в данных могут дорого стоить на следующих этапах проекта, например при эксплуатации модели (иногда даже применяется термин “отравление модели”), а использование плохих датасетов на этапе тренировки модели вообще может сделать невозможным получение качественно работающей системы. Поэтому обязательным этапом работы с данными перед передачей их в работу модели является обработка данных: очистка от аномальных значений, нормализация, удаление шумов, заполнение пропусков. Это рутинные типовые операции, которые целесообразно автоматизировать. Кроме того, существует предобработка, позволяющая существенно улучшить данные для их использования в проекте. Например, обогащение имеющихся данными с использованием информации из других источников. На этом этапе уже есть место эксперименту, поскольку только экспериментально можно уточнить какой набор данных даст наилучший результат. В здесь возникает необходимость работы с большим количеством различных датасетов, а, следовательно, и контроля их версионности.

Задачи MLOps:

* автоматизация рутинных процедур обработки сырых данных: выгрузка, преобразование типов, устранение ошибок или пропусков, обработка выбросов
* создание и использование системы версионирования различных конвейеров предобработки данных
* интеграция решений MLOps этого этапа со следующими этапами (корректная передача артефактов этого этапа на следующие уровни).

Артефакты этапа:

* обработанные наборы данных (датасеты) для использования на последующих этапах
* сохраненные конвейеры (пайплайны) обработки данных

1. Разведывательный анализ данных

На этом этапе в данных ищут скрытые закономерности, строят предварительные экспериментальные модели машинного обучения, проводят вычислительные эксперименты и формируют гипотезы для последующей проверки. Этот этап связан с большим количеством преобразований данных и испытаний различных моделей с разным набором параметров. Поэтому, кроме контроля версий датасетов на этом этапе необходимо контролировать версии кода для проверки гипотез (чаще всего это jupyter-ноутбуки).

Задачи MLOps:

* организация рабочего окружения для исследователей данных: автоматическое создание среды выполнения эксперимента с нужными версиями библиотек и настроек среды, выделение вычислительных мощностей для эксперимента, сохранение и версионирование результатов эксперимента
* автоматизация вычислительных экспериментов
* автоматизация подбора параметров моделей машинного обучения и сохранения наилучших результатов
* мониторинг экспериментов и обучения моделей машинного обучения

Артефакты этапа:

* программный код (например, jupyter-ноутбуки), являющийся результатом выполнения эксперимента
* сохранённые модели (например, гиперпараметры, веса нейронной сети), полученные в результате предварительных экспериментов и являющиеся ориентировочными (так называемыми бэйзлайнами) для улучшения в последующих экспериментов
* данные мониторинга процесса обучения

1. Выделение важной информации

На этом этапе выделяют (или формируют новые) признаки для организации обучения модели машинного обучения. Возникает необходимость в хранилище признаков (так называемый feature store).

Задачи MLOps:

* автоматизация процесса подбора эффективных признаков (анализ существующих и подбор новых)
* сохранение и версионирование набора признаков

Артефакты этапа:

* сохраненный набор признаков и значения покаателей качества работы системы, связанные с этими наборами

1. Построение и обучение модели машинного обучения

На этапе тренировки модель машинного обучения обучается с использованием тренировочного набора данных (train dataset), а после обучения модель проверяется на новых данных (test dataset). Предполагается, что у инженера машинного обучения (ML инженер) есть набор данных, описывающий необходимый процесс, и из этого набора выделяются тренировочный (train) и тестовый (test) датасеты. Важно, чтобы эти датасеты не пересекались, соотношение при разделении обычно 70% данных для тренировочной выборки и 30% данных для тестовой. Такой подход в машинном обучении называется кросс-валидация. Еще одним подходом является разделение имеющихся данных на три датасета: тренировочный (train), тестовый (test) и валидационный (validation). При этом тестовый и валидационный наборы данных отдельно используются для выбора модели и подбора их гиперпараметров. Обычно соотношение в объеме данных в этом случае составляет 60% для тренировочной выборки, 20% для тестовой и 20% для валидационной выборок. Для разных типов задач и разных моделей используются различные подходы для формирования датасетов, их подбор также может быть частью эксперимента, поэтому данный процесс необходимо автоматизировать и уметь запоминать различные варианты датасетов, на которых обучалась и проверялась модель, то есть необходимо версионирование.

Часто алгоритмы машинного обучения представляют собой «черный ящик», логику работы которого сложно объяснить. С практический точки зрения этого бывает достаточно, однако такая непредсказуемость может вести к неустойчивой работе при отклонениях в окружении работы модели. Поэтому необходимо контролировать версии как самих моделей, так и наборов гиперпараметров и окружения (библиотеки, переменные), в которых работают модели. Важно на этом этапе учитывать, что успешные модели на следующих этапах перейдут в промышленное окружение, поэтому надо создавать обучающее окружение как можно более похожее на промышленную среду. Это касается использования библиотек, ограничения на производительность процессора и расход оперативной памяти, требований к производительности модели.

Задачи MLOps:

* автоматизация вычислительных экспериментов и обучения модели машинного обучения
* автоматизация подбора обучающего и тренировочного набора данных, их версионирование и привязку к обученной модели и окружению, в котором обучалась модель
* автоматизация развертывания среды обучения и испытания модели

Артефакты этапа:

* сохранённые модели (например, гиперпараметры, веса нейронной сети), полученные в результате предварительных экспериментов и являющиеся ориентировочными (так называемыми бэйзлайнами) для улучшения в последующих экспериментов
* параметры работы модели (требования к памяти, CPU, планируемая скорость работы при инференсе)

1. Развертывание модели машинного обучения

При развертывании важно как можно ближе сохранить окружение для модели машинного обучения. Это достигается за счет использования технологий виртуализации (виртуальные машины, контейнеры), контейнеров (docker, kubernetes) и автоматической настройки инфраструктуры (Ansible, Teraform).

Задачи MLOps:

* автоматически создать окружение для работы модели, включая согласованные версии операционной системы, системных утилит и библиотек, переменных окружения,
* выполнить сборку и загрузку проекта в рабочее окружение,
* произвести автоматические настройки оборудования и программного обеспечения для запуска модели машинного обучения,
* обеспечить необходимый мониторинг работы системы.

Артефакты этапа:

* работающая модель машинного обучения в промышленной среде

1. Эксплуатация

На этом этапе построенная модель выводится в промышленное окружение и начинает работать на реальных данных. На этом этапе часто возникают проблемы, связанные с качеством данных на реальных объектах, поэтому в эксплуатации важны функции мониторинга работы системы в целом и качества работы модели машинного обучения. Наиболее часто для целей мониторинга применяются средства визуализации и контроля.

Задачи MLOps:

* автоматизированный мониторинг работы системы, реагирование на инциденты,
* обеспечение автоматизированного контроля данных, на которых работает модель машинного обучения.

Артефакты этапа:

* результаты эксплуатации модели машинного обучения в промышленной среде: данные мониторинга, сообщения об ошибках, информация об использовании ресурсов (оперативная память, CPU), данные о скорости выполнения операций

1. Контроль и оценка качества

Постоянный автоматизированный контроль качества и, при необходимости, переобучение модели являются обязательными в эксплуатации любой модели машинного обучения. Практически невозможно создать универсальную модель для долгого срока эксплуатации, так как промышленные данные и окружение имеют свойство изменяться, что приводит к ухудшению качества работы модели.

Задачи MLOps:

* постоянный автоматизированный анализ качества работы модели машинного обучения,
* анализ инцидентов, обнаружение причин,
* рекомендации по дальнейшей работе с моделью машинного обучения (дообучение на новых данных, создание другой модели).

Артефакты этапа:

* данные о соответствии результатов работы модели машинного обучения запланированным метрикам качества
* наборы данных (датасеты) для дообучения или переобучения модели машинного обучения.

Итак, мы теперь знаем какие задачи MLOps возникают на каждом из этапов проекта машинного обучения. Рассмотрим с практической точки зрения самые важные из них.

Задача 1: обеспечение повторяемости результатов проекта.

Данные в обучающем датасете и в реальных данных в эксплуатации модели могут отличаться. Кроме того, могут отличаться среды, в которых работают модели. Например, исследователи могут строить модели в операционной системе Linux Debian, а на объекте может быть установлена система Kali Linux. Большая практическая проблема, касающаяся всех участников проекта ML, это обеспечение повторяемости результатов работы модели в среде разработки и производственной среде.Поиск решения проблемы, связанной с использованием разных версий библиотек, может занимать большое количество времени. Для этого необходимо контролировать множество параметров:

* наборы данных (количество признаков, качество данных, однородность данных)
* настройки окружения (характеристики аппаратного обеспечения, скорость передачи данных)
* гиперпараметры модели
* используемые служебные библиотеки

Задача 2: автоматизация.

Представьте, что ML инженер наконец-то обучил свою модель на мощных графический процессорах, сохранил веса этой модели и готов теперь применить полученную модель для анализа производственных данных. Не тут-то было! Его супер-модель является лишь частью большого проекта и для запуска в промышленную эксплуатацию надо собрать всю систему. И протестировать. И обеспечить наличие всех необходимых библиотек, настроек. Конечно это все можно делать руками и глазами, но можете себе представить насколько это эффективно, когда приходится одни и те же действия повторять бесчисленное количество раз. Тут недалеко и от ошибки, связанной с “человеческим фактором”. И на помощь приходит MLOps, решающий задачу автоматизации. С использованием специальных утилит автоматизации (git, Jenkins и т.п.) или даже обычных скриптов (bash, python) можно автоматизировать рутинные операции по сборке всего пакета программного обеспечения целиком.

Задача 3: создание правильного рабочего окружения.

Развертывание рабочего окружения может стать головной болью для инженера MLOps, поскольку включает в себя множество узлов со специальными настройками. Поднимать сервера с нужной операционной системой, устанавливать библиотеки и делать настройки среды, выбирать правильные и совместимые версии для используемого программного обеспечения (базы данных, технологии контейнеризации, компиляторы и интерпретаторы) - выполнение этих задач может растянуться надолго, а проект требует быстрого вывода в продакшн. Поэтому на данном этапе активно используют инструменты для автоматизации рутинных повторяющихся действий:

* docker, docker-compose или kubernetes для виртуализации
* chef, ansible, terrafrom для создания необходимых конфигураций

*В этом юните вы узнали:*

* из каких этапов состоит процесс разработки программного обеспечения
* во чем отличия проекта машинного обучения от обычных проектов разработки программного обеспечения
* что является результатом работы на каждом этапе
* какие задачи возникают на различных этапах и как эти задачи решает MLOps:
  + автоматизация на всех этапах
  + контроль версий датасетов, моделей, окружений

Теперь изучим этот вопрос в привязке к жизненному циклу.

*Проверим:*

1. Что символизирует символ “бесконечность” на рисунке, описывающем процесс DevOps
   1. то, что проблем в DevOps/MLOps бесконечно много
   2. **то, что итерации в процессе DevOps/MLOps должны быстро и много повторяться**
   3. то, что в DevOps/MLOps используется бесконечно много разных инструментов
   4. то, что расходы на DevOps/MLOps неограничены
2. Что такое артефакт этапа?
   1. документ с перечнем ошибок
   2. список задач этапа
   3. **результат выполнения этапа, который передается на следующий этап**
   4. недокументированные возможности работы системы
3. Что можно автоматизировать в проекте машинного обучения?
   1. подготовку технического задания
   2. **сбор данных**
   3. **обучение модели машинного обучения**
   4. **вывод системы в продакшн**
4. Какие проблемы может решить MLOps?
   1. **человеческие ошибки при повторении большого количества одинаковых рутинных операций**
   2. высокая стоимость вычислительных ресурсов
   3. **ошибки, связанные с несовместимостью различных версий библиотек, использованных в решении**
   4. некорректное техническое задание

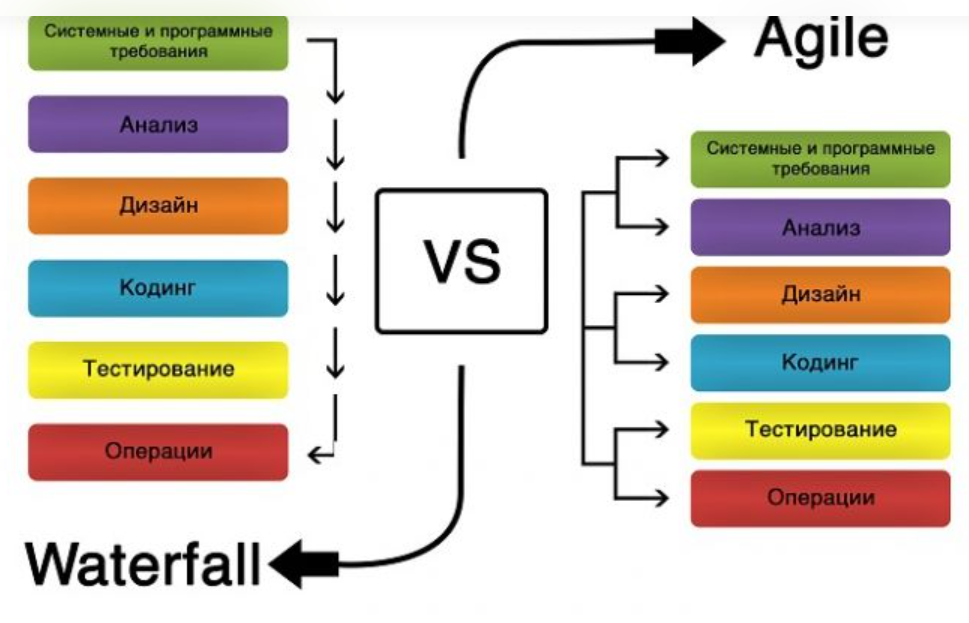
# Модуль 1. Юнит 3: Управление проектом и MLOps.

*Об этом юните:*

Существуют различные методологии управления проектом, но для повышения эффективности проекта машинного обучения ключевыми факторами являются эффективная автоматизация и обеспечение непрерывности процесса. В этом юните мы узнаем о том как MLOps помогает сделать проект эффективнее.

*Содержание юнита:*

Любой проект обладает своим жизненным циклом. **Жизненные циклы различаются для разных типов проектов, как по перечню этапов и задач, так и по их продолжительности.** В проектах типа “водяной мельницы” каждый этап может продолжаться очень долго, необходимым условием для перехода на следующий этап является выполнение всех задач предыдущего этапа.



Сравнение методов управления проектами, https://worksection.com/blog/waterfall-vs-agile.html.

В противоположность этому в гибких (agile) подходах акцент сделан на быстрое создание прототипа и большое количество повторений всех циклов проекта. Проекты машинного обучения практически всегда управляются с использованием agile-методологии, и вызвано это не столько желанием руководителя проекта поскорее показать первую версию проекта заказчику, сколько тем, что главным свойством объектов в проекте (датасеты, модели, конвейеры, окружение) является изменчивость. Вчерашние данные могут существенно отличаться от сегодняшних, разработчики программного обеспечения могут внести изменения в библиотеку, а заказчик может поменять настройки окружения. Поэтому команда проекта машинного обучения должна быть всегда готова к изменениям, постоянные и частые итерации проекта должны быть девизом участников команды.

В методологии управления проектами большое внимание уделяется соблюдению параметров проекта, которыми являются: время, ресурсы, ожидаемые функции (качество). Редкому руководителю проекта удается сдать проект в оговоренное время, не превысив имеющиеся ресурсы, при этом реализовать все, что от проекта ожидалось. Практически всегда один из параметров страдает.



Параметры проекта

Ограничения на ресурсы проекта могут быть следующие:

* размер команды,
* используемый технологический стек (в т.ч. возможность использования открытого ПО),
* возможность изменений технического задания при работе над проектом.

Искусство проектного управления состоит в том, чтобы провести проект по всем этапам с соблюдением всех существующих ограничений. Наиболее популярными agile-методами являются Scrum и Kanban. Эти методы позволили существенно ускорить выход продукта на рынок, повысить качество. Основная идея этих методов состоит в том, что команда работает над проектом короткими итерациями (спринтами), в ходе которых осуществляется сборка всего проекта целиком и показ заказчику, после которого уточняется содержание задач в “техническом долге” и формируется план работы на следующий спринт. Такой подход требует повторения одинаковых рутинных операций, которые эффективно объединить в группы и разработать для каждой группы свои правила и методы, в том числе автоматизацию процессов.

В любом проекте разработки программного обеспечения контролируемая автоматизация связывается с обеспечением так называемой **непрерывности** процесса. **Под непрерывностью понимается управляемый и контролируемый автоматически выполняемый набор стандартных операций, которые ведут к выполнению какой-либо крупной задачи в проекте.** Обычно в качестве таких операций в проекте машинного обучения рассматриваются:

* непрерывная сборка (Continuous Integration, CI),
  + проверка кода,
  + выполнение задач,
  + валидация кода,
  + unit тестирование,
  + объединение кода,
* непрерывное развертывание и доставка (Continuous Delivery, CD),
  + сборка кода в единую систему,
  + тестирование,
  + выпуск релиза,
* непрерывное обучение (Continuous Training, CT).
  + мониторинг работы модели,
  + измерения,
  + дополнительное обучение, тюнинг,
  + эксплуатация.

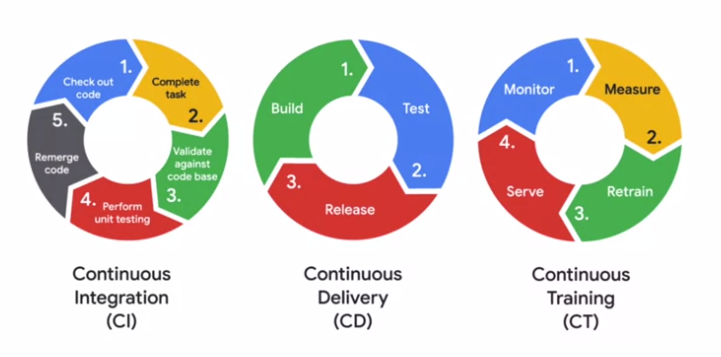


Рисунок “Автоматизация для разных операций проекта машинного обучения”

Практически все задачи на всех этапах в проекте машинного обучения можно выполнять вручную или автоматически. Понятно, что автоматизация позволяет быстрее и точнее решать задачи, без автоматизации немыслима реализация большого проекта в котором большое количество участников, много отдельных модулей и систем, большой объем кода, тестов.

Важным параметром проекта является “технический долг”: накопленные задачи и замечания, появляющиеся после показов заказчику и эксплуатации проекта. Технический долг присутствует практически в любом проекте, управляемом по agile, поскольку идеология agile-методов предполагает как можно более быстрый вывод прототипа проекта в эксплуатацию и получение обратной, как правило негативной, связи от заказчика. Техническим долгом необходимо эффективно управлять, определяя наиболее важные для устранения замечания, устраняя причины появления замечаний, улучшая качество кода и архитектуры.

*Пример: уже известная нам из предыдущих юнитов команда амбициозных web-разработчиков решила реализовать на свой страх и риск версию сайта без уточнения всех деталей у постановщика. Автоматически после первой демонстрации у команды образовался “технический долг”: функции и элементы сайта, которые необходимо доделать.*

Технический долг в проектах машинного обучения имеет обыкновение увеличиваться быстрее, чем в классических проектах разработки программного обеспечения, так как его могут создавать:

* многофункциональная команда, большое количество ролей и специализаций,
* экспериментальный характер процесса (на начальном этапе неясно будет ли та или иная гипотеза эффективной),
* сложность тестирования (отсутствие обучающих данных),
* сложность развертывания (множество разных библиотек),
* разрушение модели в эксплуатации (со временем данные изменяются, необходимо переучивать модель).

Наиболее системно эти факты изложены в известной статье Google “Hidden technical debts in machine learning systems”.



Если коротко сформулировать основную идею Google, то она выглядит так: “Разработка и развертывание систем машинного обучения происходит относительно быстро и дешево, но поддерживать их в течение долгого времени дорого и сложно”.

Идеология MLOps появилась для того, чтобы ускорять и улучшать качество в проекте машинного обучения, используя автоматизацию процессов, контроль версий компонентов решения, тестирование, сборку и вывод решения в производственную среду. Методы и прикладные инструменты MLOps, которые мы будем рассматривать в этом учебном курсе, как раз направлены на решение проблемы разрастания “технического долга”. Жизненный цикл проектов машинного обучения состоит из типовых этапов, изученных нами в предыдущем юните. На разных этапах используются разные методики и инструменты MLOps для борьбы с “техническим долгом”.

На **этапах сбора и предобработки данных** накапливается большое количество сырых и обработанных данных, которые необходимо сохранять, управлять изменениями в них, обеспечивать доступ к нужным данным. На этом этапе применяются следующие методики и инструменты MLOps:

* контроль и версионирование данных (dvc),
* автоматизация обработки данных, конвейеры данных (python, sklearn, Apache)
* скрипты автоматизации и загрузки данных (python, Apache).

**Разведывательный анализ данных и выделение важной информации**, в том числе важнейших признаков и взаимосвязей в данных, предполагает активное исследование, эксперименты с данными, создание новых признаков. На этом этапе необходимо контролировать версионность экспериментальных данных, создаваемых конвейеров и пробных моделей машинного обучения.

Методики и инструменты MLOps:

* автоматизация процесса подбора эффективных признаков, анализ существующих и подбор новых, хранилище признаков (streamSQL)
* сохранение и версионирование набора моделей (git),
* сохранение и версионирование датасетов и пайплайнов (dvc)

На **этапе построения и обучения модели машинного обучения** проводится много исследований и проверок гипотез, поэтому часто возникают ситуации, когда модель начинает показывать результаты хуже, поэтому требуется вернуться к предыдущей версии.

Методики и инструменты MLOps:

* создание виртуальной среды выполнения модели (venv, virtualenv, poetry, docker)
* версионирование и контроль моделей машинного обучения (git)
* контроль процесса обучения модели (tensorboard)

При **развертывании модели машинного обучения** важно быстро создавать правильное окружение для работы модели, поэтому на этом этапе используются различные технологии виртуализации. Перед развертыванием как правила проводится сборка всего проекта целиком, с тестированием отдельных компонентов и всего решения.

Методики и инструменты MLOps:

* автоматическая сборка решения (Jenkins),
* тестирование (pytest),
* создание контейнеров микросервисов для организации работы модели (docker, docker-compose, kubernetes),
* настройка среды выполнения программы (Ansible)

При **эксплуатации** важны качественный мониторинг всех параметров работы системы. Важной частью процесса эксплуатации является этап **контроля и оценки качества работы** модели.

Методики и инструменты MLOps:

* мониторинг, реагирование на инциденты, визуализация (grafana, Kibana),
* обеспечение автоматизированного контроля данных, на которых работает модель машинного обучения.

Отдельно необходимо отметить важность визуализации на каждом из этапов проекта. Хорошая визуализация полезна на любом этапе жизненного цикла проекта.

* При анализе данных с помощью содержательной диаграммы можно легче найти характерные признаки в данных, например, наличие выбросов или статистические свойства.
* При обучении модели требуется постоянно анализировать графики качества работы модели, чтобы избежать переобучения.
* В эксплуатации от средств мониторинга зависит скорость реагирования на внештатные ситуации.
* Незаменимыми являются хорошие, содержательные, легко интерпретируемые графики при презентации проделанной работы руководству.

*В этом юните мы узнали, что*

* MLOps тесно связан с каждым этапом жизненного цикла модели машинного обучения
  + сбор, обработка и анализ данных
  + проверка гипотез, эксперименты и построение моделей
  + сборка и вывод в эксплуатацию проекта
  + эксплуатация и анализ качества
* для каждого этапа разработано множество инструментов, в том числе с открытым исходным кодом (open-source)
* визуализация очень важна на каждом из этапов

Теперь рассмотрим вопрос с точки зрения состава команды проекта, ролей и их функций.

*Проверим:*

1. что такое “технический долг”
   1. задолженность исполнителя перед заказчиком
   2. задолженность заказчика перед исполнителем
   3. **перечень функций, которые необходимо реализовать, чтобы получить приемлемое для заказчика качество продукта**
   4. список ошибок, найденных при тестировании
2. что такое “версионирование”
   1. **контроль версий**
   2. правила нумерации датасетов
   3. правила нумерации всех объектов
   4. формат номера
3. каких задач MLOps нет в обычном проекте разработки
   1. **версионирование датасетов**
   2. **версионирование пайплайнов**
   3. управление изменениями в коде
   4. настройка виртуального окружения
4. поставьте в соответствие этапам задачи, которые решаются с помощью визуализации

|  |  |
| --- | --- |
| анализ данных | обнаружение ошибок в данных, выявление характерных статистических признаков |
| построение моделей | контроль переобучения |
| эксплуатация | контроль качества работы решения, обнаружение отклонений в работе |
| анализ результатов проекта | создание наглядных интерпретируемых графиков для презентации результатов заказчику |

# Модуль 1. Юнит 4: Команда ML проекта (роли)

*В этом юните:*

Мы поговорим о том какие существуют роли участников в проектах машинного обучения. Для каждой роли существует свой набор задач и мы узнаем как с этим связаны задачи MLOps.

*Содержание юнита:*

При реализации проектов машинного обучения часто используются стандартные подходы из классического проектного управления разработкой программного обеспечения, однако в проектах ML есть свои особенности

* бизнесовая составляющая: качество работы модели машинного обучения измеряется различными математическими метриками и большим искусством является превратить бизнес-требования и бизнес-метрики в задачи для команды проекта и технологические метрики,
* научная составляющая: важной частью проекта ML является работа с данными и исследовательский этап,
* инфраструктурная составляющая: при испытаниях и использовании модели на промышленных данных важно учитывать большое количество факторов, например, окружение модели, производительность, ожидаемую скорость работы, интеграцию с другими системами и т.п.

Особенности проектов ML находят отражение в определении необходимых ролей участников проекта и требований к ним. Этих ролей может быть много, в больших компаниях за каждой ролью может быть закреплен целый отдел, а, напротив, в небольших компаниях-стартапах один человек может выполнять задачи сразу нескольких ролей.

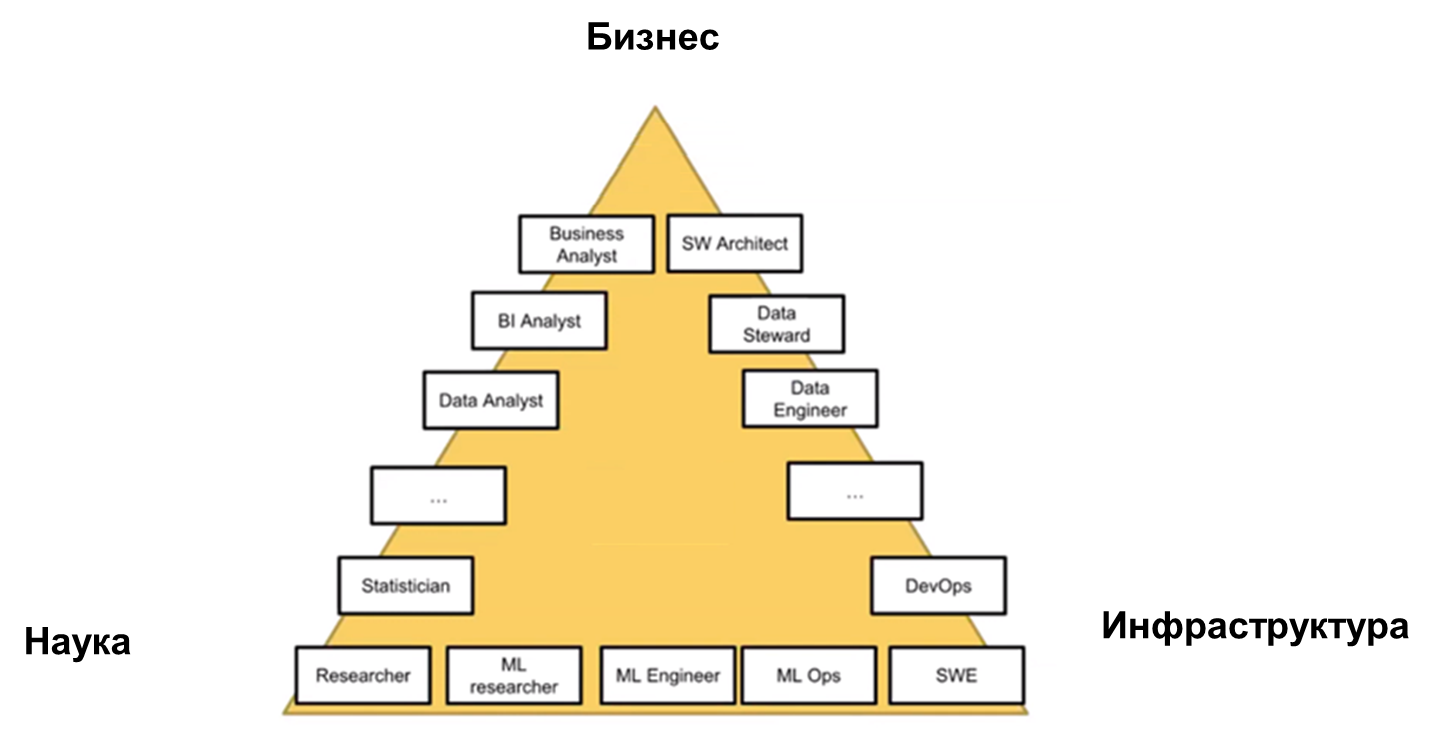


Рисунок “Различные роли участников проекта машинного обучения”.

Давайте разберемся с тем что делают различные участники проекта, чтобы понять какие задачи MLOps от них могут появляться:

* Бизнес
  + Эксперт, разбирающийся в предметной области (финансы, медицина, государственное управление, образование и т.п.), обычно это представитель заказчика, однако для эффективной реализации проекта такой человек должен быть и в команде проекта. В сферу ответственности эксперта входят постановка задачи, уточнение бизнес-требований, консультирование по технологиям, итоговый вывод о соответствии созданного решения поставленной задаче и анализ эффективности (в том числе технико-экономический анализ).
  + Руководитель проекта: обеспечение выполнения проекта в срок с требуемым качеством в рамках установленного бюджета
  + Руководитель продукта: связь с рынком, CustDev
  + Бизнес-аналитик: постановка задачи разработчиком на основе полученных задач от заказчика
  + Заказчик (спонсор) проекта, может быть как внешний, так и внутренний: выделение ресурсов для проекта, приоритезация, политическая и организационная поддержка
* Работа с данными и моделями
  + Дата инженер, архитектор данных: сбор, передача, хранение данных, формирование всего конвейера работы с данными, понимание источников получения данных
  + Дата аналитик: разведочный анализ данных, поиск признаков, формирование гипотез, извлечение смысла из данных
  + Исследователь данных: интеллектуальный анализ данных, формулировка гипотез
  + ML инженер: организация работы инфраструктуры, контроль версий библиотек
  + ML исследователь: проверка гипотез, реализация прототипов, анализ SOTA
* Реализация проекта
  + Архитектор, разработчик: создание архитектуры решения, написание кода
  + Тестировщик: тестирование решения
  + DevOps/MLOps/DataOps: автоматизация задач, развертывание систем
  + Дизайнеры: создание графических интерфейсов, дашбордов
  + Технические писатели: написание документации
* Эксплуатация
  + специалисты технической поддержки: обеспечивают работоспособность решения, реагируют на инциденты.

Практически невозможно найти так называемого специалиста DataScience (на эту тему подробнее можно посмотреть выступление В.Бабушкина “Почему вы никогда не найдете датасайнтиста” <https://www.youtube.com/watch?v=Cs3ae65tmKA>). Главная причина состоит в том, что задач в проекте очень много и их перечень сильно зависит от типа проекта и команды. Как правило это означает, что в команде проекта машинного обучения много ролей и все участники пользуются различными инструментами для решения конкретных задач.

Чтобы понять какие задачи возникают в MLOps давайте еще раз рассмотрим всю последовательность от постановки задачи до вывода модели в производственную среду и ее эксплуатации.

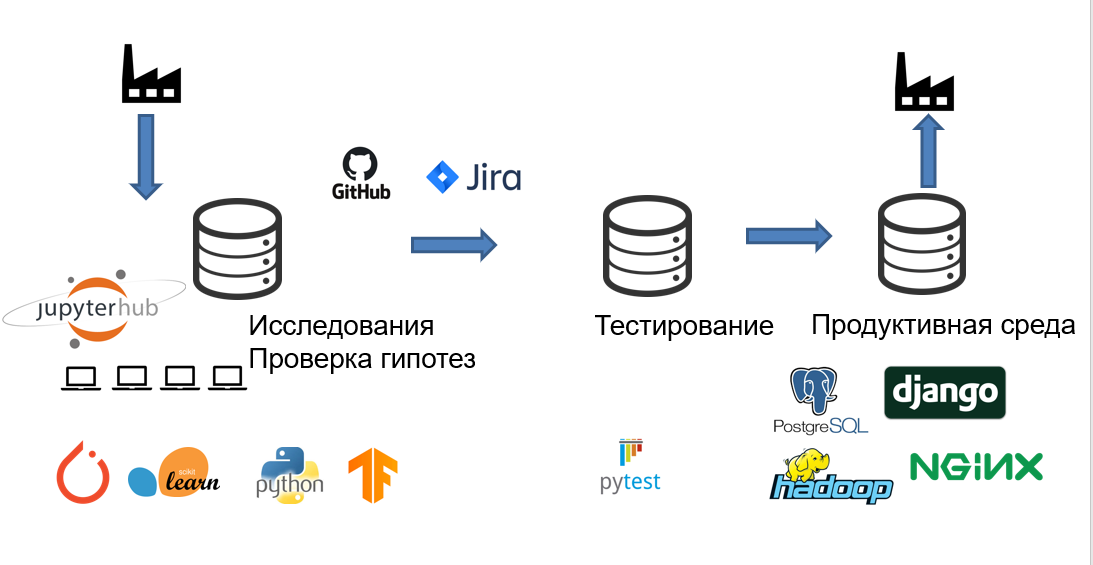
**

Рисунок “Последовательное выполнение задач проекта машинного обучения и некоторые инструменты для их решения”

Схема работы над проектом:

1. бизнес-аналитик совместно с заказчиком и экспертом описывает задачу, определяет требования к качеству, ограничения задачи

Задачи MLOps:

* участие в формировании и обсуждении технического задания с целью определить требования к системе, ограничения
* выбрать инструменты с учетом имеющихся ограничений на ресурсы
* определить целевую конфигурацию аппаратного обеспечения для
  + сбора и хранения данных и производных датасетов, признаков
  + проверки гипотез и хранения моделей
  + сборки и тестирования решения
  + развертывания решения у заказчика

На этом этапе никаких специальных инструментом MLOps специалист не использует, занимается выработкой требований и определением стека технологий для будущих задач.

1. дата аналитик и дата инженер исследуют источники получения данных, проводят их систематизацию, определяют способ получения и преобразования данных

Задачи MLOps:

* организовать сбор данных с использованием определенных в техническом задании технологий
* обеспечить надежное хранение данных,
* реализовать политики безопасного доступа к данным с учетом их специфики (коммерческая тайна, персональные данные и т.п.)
* организовать рабочее окружение для обработки данных

Инструменты MLOps:

* сетевые протоколы и утилиты для передачи информации: socket, API, ftp, syslog
* программное обеспечение для сбора и преобразования данных: Apache Flume, Nifi
* управление версиями датасетов и пайплайнов-обработчиков: dvc

1. с полученными данными начинает работать ML исследователь и ML инженер, которые должны построить модели

Задачи MLOps:

* организовать доступ к данным
* организовать рабочее окружение для ML инженера
  + виртуальная среда
  + интерактивное окружение для работы
  + рабочие библиотеки

Инструменты MLOps:

* библиотеки машинного обучения: python, numpy, pandas, matplotlib, sklearn, TensorFlow, PyTorch и др.
* интерактивные python-ноутбуки Jupyter, JupyterHub, binder для работы исследователей
* создание виртуальных окружений: venv, virtualenv, conda, poetry

1. Для моделей ML формируются технологические метрики оценки качества, увязанные с бизнес-метриками, необходим их постоянный контроль

Задачи MLOps:

* мониторинг метрик
* мониторинг процесса обучения модели
* сохранение наилучших параметров модели для переиспользования

Инструменты MLOps:

* визуализация мониторинга: grafana, Kibana
* визуализация контроля обучения: tensorboard

1. В соответствии с полученными метриками и построенными моделями инженер-тестировщик проводит тестирование модели

Задачи MLOps:

* автоматизация тестирования модели

Инструменты MLOps:

* автоматизация тестирования: pytest
* визуализация контроля обучения: tensorboard

1. На основе созданного и протестированного прототипа модели инженер-программист разрабатывает продакшн-вариант, либо интегрирует уже существующий модуль в общее решение, с учетом имеющихся ограничений на ресурсы, сформулированные заказчиком

Задачи MLOps:

* сборка решения
* общее тестирование решения

Инструменты MLOps:

* автоматизация сборки: Jenkins, git
* тестирование: pytest
* виртуализация: docker, docker-compose, kubernetes.
* различные решения для организации работы системы целиком
  + базы данных
  + веб-сервер
  + бэкенд
  + фронтэнд
  + очереди сообщений
  + API
  + и т.п.

1. Инженеры выводят созданную модель в продуктивное использование и обеспечивают контроль работоспособности модели в продуктивной среде.

Задачи MLOps:

* организация рабочего окружения для работы решения
* вывод в продакшн
* автоматизированный мониторинг работы системы
  + качество данных
  + качество модели
  + использование ресурсов

Инструменты MLOps:

* автоматизация сборки: Jenkins, git
* мониторинг: grafana, Kibana.
* виртуализация: docker, docker-compose, kubernetes.

*Теперь вы знаете, что:*

* для вывода проекта машинного обучения в промышленную эксплуатацию необходима большая команда разных специалистов, у каждого свои задачи,
* роль MLOps наладить технологическую одновременную работу всех специалистов, автоматизировать рутинные операции для ускорения получения результата и повышения качества.
* на каждом этапе существуют различные инструменты, которыми должен уметь пользоваться специалист MLOps
* также во многих проектах важно знать и уметь пользоваться инструментами, которыми пользуются другие участники команды

Теперь пришло время более подробно изучить инструментарий.

*Проверим:*

1. Распределите инструменты, которые применяются участниками команды проекта машинного обучения с учетом их роли
   1. Data Engineer - SQL
   2. ML Researcher - Jupyter Notebook
   3. топ-менеджер, спонсор проекта - dashboard
   4. MLOps Engineer - Jenkins
2. Установите соответствие между инструментом и его назначением

|  |  |
| --- | --- |
| pytest | автоматизация тестирования |
| sklearn | библиотека машинного обучения |
| dvc | контроль версий датасетов |
| python | интерпретатор языка программирования |

# Модуль 1. Юнит 5: Обзор инструментов ML/MLOps

*В этом юните:*

Сделан обзор наиболее популярных инструментов, используемых в проектах машинного обучения и MLOps. Перечень инструментов может меняться. Знать плюсы и минусы, особенности применения, важно для эффективного решения задач MLOps.

*Содержание:*

На практике существует множество инструментов, используемых для автоматизации задач машинного обучения, о некоторых из них мы уже узнали в предыдущем юните. В этом юните мы изучим эти инструменты, что позволит наконец переходить к практическим задачам.

Состояние индустрии машинного обучения достигло такого уровня, что практически для каждой задачи сейчас существуют надежные и высокопроизводительные библиотеки и фреймворки, хорошо решающие эту задачу. Это объясняется активностью сообщества, большим количеством open-source проектов, доступностью технологий и вовлечением большого количества исследователей. Стандартные фреймворки (например, TensorFlow, PyTorch, sklearn и многие другие) уже содержат готовые, протестированные, хорошо документированные функции для решения практически всех возможных задач машинного обучения. **Поэтому сейчас чаще всего разработка модели машинного обучения связана с использованием и адаптацией готовых решений, создание модели «с нуля» зачастую является «изобретением велосипеда» и неэффективно с точки зрения ресурсов.** У такого подхода есть и минусы – поскольку большинство инструментов находятся в постоянной разработке, да еще и разработчиков огромное количество, то постоянно появляются новые версии библиотек, при этом часто возникает проблема совместимости разных версий библиотек между собой, так как большой проект включает в себя множество разных библиотек. Это может привести к тому, что при незначительном изменении версии одной из используемых библиотек модель начинает вести себя по-другому. Такого рода ошибки особенно сложно выявлять, **поэтому управление составом и версиями используемых компонентов, датасетов и конфигурациями модели позволяет существенно сократить время на создание, тестирование, оптимизацию, поиск ошибок в моделях машинного обучения.**

На разных этапах жизненного цикла проекта используется различное программное обеспечение, его выбор и конфигурация определяются условиями проекта, имеющимися ограничениями и иногда личным вкусом участников проекта. Однако перед использованием того или иного инструмента важно задавать следующие вопросы:

* Какова стоимость используемого решения и оправдано ли соотношение цены и качества? Можно ли использовать бесплатное решение?
* Какие временные затраты связаны с внедрением и эксплуатацией?
* Есть ли встроенные механизмы контроля качества?
* Насколько просто будет обучить разработчиков и пользователей, есть ли необходимая документация?
* Все ли ожидаемые функции выполняются?
* Какие потребуются ресурсы: вычислительная мощность, оперативная память?

Список вопросов может быть шире для различных проектов.

Вот далеко не полный перечень некоторых инструментов, используемых для автоматизации проектов разработки программного обеспечения, с привязкой этих инструментам к этапам жизненного цикла проекта.

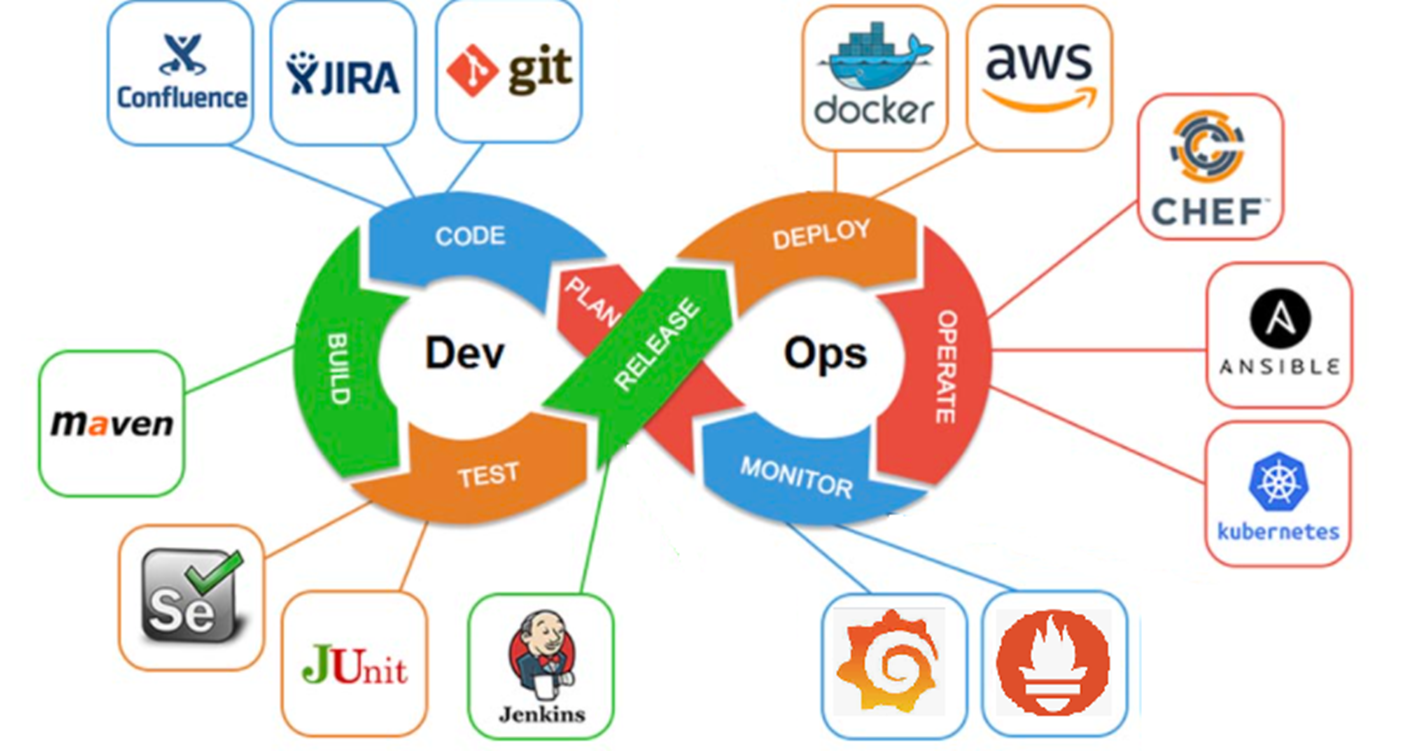


Рисунок “Этапы жизненного цикла проекта и некоторые инструменты автоматизации”

Далее рассмотрим эти инструменты более подробно и обсудим какие навыки они требуют от MLOps инженера.

1. Инструменты исследователей данных, ML исследователей и инженеров

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Источники | Описание | Что требуется от MLOps |
| tensorflow, tensorboard | tensorflow.org | Открытый фреймворк машинного обучения, созданных Google. Позволяет быстро создавать и обучать сложные модели (например, нейронные сети). Эффективен для проверки гипотез. | Установка, наладка, мониторинг, контроль версий и окружения, администрирование работы с GPU. |
| pytorch | pytorch.org | Открытый фреймворк машинного обучения, эффективен для продакшн | Установка, наладка, мониторинг, контроль версий и окружения, администрирование работы с GPU. |
| sklearn | sklearn.org | Множество различных алгоритмов машинного обучения. Удобен для проверки гипотез. | Установка, наладка, мониторинг, контроль версий и окружения. |
| jupyter, jupyterhub, binder | jupyter.org | Интерактивная среда разработки. | Установка, настройка,  контроль ресурсов (память, CPU, HDD),  поддержка. |
| python | python.org | Язык разработки, интерпретатор. | Установка, настройка, поддержка. |
| venv, virtualenv, poetry, conda | pip  python.org  anaconda.com | Виртуальные окружения. Позволяют создавать индивидуальное окружение (версии библиотек, переменные окружения) для отдельного проекта. | Установка, настройка, поддержка. |

1. Управление версионированием кода, датасетов, моделей, конвейеров

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Источники | Описание | Что требуется от MLOps |
| github | github.com | Система хранения и версионирования кода программного обеспечения | создание репозиториев,  установка и настройка, организация работы, интеграция с другими системами (например, Jenkins) |
| dvc | dvc.org | Система хранения и версионирования датасетов | настройка, организация работы, интеграция с другими системами |
| Apache Airflow | airflow.apache.org | Автоматизация конвейеров машинного обучения | настройка, организация работы |

1. Автоматизация тестирования и сборки

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Источники | Описание | Что требуется от MLOps |
| jenkins | jenkins.org | Автоматизация всех этапов от создания до эксплуатации | настройка, организация работы, написание скриптов автоматизации |
| pytest | pytest.org | Автоматизация тестирования | настройка, организация работы |
| pydoc | pydoc.org | Автоматизация подготовки документации | настройка, организация работы |

1. Контейнеризация

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Источники | Описание | Что требуется от MLOps |
| docker | docker.org | Инструмент для создания виртуальных сред - контейнеров для выполнения программ | установка и настройка, организация работы, контроль ресурсов |
| docker-compose | docker.org | Управление различными контейнерами | установка и настройка, организация работы, контроль ресурсов |
| kubernetes | kubernetes.org | Управление различными контейнерами | установка и настройка, организация работы, контроль ресурсов |

1. Мониторинг

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Источники | Описание | Что требуется от MLOps |
| grafana | grafana.org | Построение дашбордов | настройка, организация работы, интеграция с базами данных |
| kibana | elastic.org | Визуализация | настройка, организация работы |
| tensorboard | tensorflow.org | Визуализация и контроль работы модели машинного обучения | настройка, организация работы, интеграция со средой обучения модели |

С этими инструментами более подробно вы познакомитесь в следующих модулях курса.

*В изученном юните:*

* рассмотрели наиболее популярные инструменты для проектов машинного обучения и MLOps:
  + версионирование: git, dvc,
  + инструменты исследователей данных и моделей машинного обучения: python, sklearn, tensorflow
  + автоматизация: jenkins, pytest, pydoc
  + контейнеризация: docker, docker-compose, kubernetes,
  + мониторинг: grafana, kibana
* узнали на что необходимо обращать внимание при выборе того или иного инструмента: требования к ресурсам, навыки команды, оборудование.

Теперь когда мы вооружены знаниями об инструментах мы можем смело вступать во взаимодействие с внешней средой - заказчиком, создающим бизнес-требования, оказывающие существенное влияние на все аспекты процесса и, в том числе, на MLOps задачи.

*Проверим:*

1. Отметьте инструменты, которые используются для контроля версий различных артефактов
   1. **git**
   2. **dvc**
   3. grafana
   4. linux
2. Кто является разработчиком TensorFlow, осуществляющим поддержку этого решения на открытом исходном коде
   1. Илон Маск
   2. Facebook
   3. **Google**
   4. открытое сообщество
3. Какие инструменты используются для мониторинга
   1. **grafana**
   2. tensorflow
   3. pytorch
   4. **kibana**
4. Какие инструменты используются для организации виртуальной среды
   1. **venv**
   2. **docker**
   3. **kubernetes**
   4. http
5. Установите соответствие между участниками команды и используемыми ими инструментами

|  |  |
| --- | --- |
| jupyter ноутбук | исследователь |
| jenkins | DevOps инженер |
| grafana | эксплуатация |
| pytest | тестировщик |

# Модуль 1. Юнит 6: Учет бизнес-требований и их влияние на MLOps

*В этом юните вы узнаете:*

* почему для реального практического проекта важно учитывать бизнес-требования,
* как правильно должны быть сформулированы бизнес-требования, чтобы это не стало головной болью участников проекта машинного обучения и, в том числе, MLOps,
* что может входить в постановку задачи,
* как постановка задачи влияет на MLOps.

*Содержание юнита:*

Выполнение любого проекта связано с достижением целей, сформулированных постановщиком задачи, как правило это заказчик. Эти требования связаны с достижением бизнес-целей. Бизнес-целью может быть, например, повышение конверсии посещений сайта в заключенные договоры, увеличение трафика в торговой точке, снижение издержек на транспортировку товара. Есть бизнес-цели и в проектах машинного обучения. То, кто уже имеет опыт работы в реальных промышленных проектах машинного обучения, уже знают, что реальный проект отличается от учебных и тренировочных (например, от соревнований на площадке kaggle.com). Самое главное отличие состоит в том, что в соревновательном проекте зачастую главное это показать наилучший результат работы модели, “побить бейзлайн”, а в реальном проекте необходимо очень хорошо понимать и выполнять бизнес-требования заказчика проекта, при этом возможна ситуация, когда построенная модель показывает не лучшее качество, однако укладывается в имеющиеся ограничения (оперативная память, скорость работы, CPU). С определением бизнес-требования вы ознакомились в рамках курса “Управление проектами”. В этом же юните мы посмотрим на влияние бизнес-требований на задачи MLOps, поскольку понимание бизнес-требований позволяет эффективно планировать инфраструктуру, что в последующем экономит силы и время и является залогом общего успеха проекта.

Одним из важных условий создания и запуска в эксплуатацию эффективной модели машинного обучения является наличие грамотно составленного технического задания. **Техническое задание это важнейший проектный документ, фиксирующий ожидания от проекта и возможные ограничения.** Обычные пункты такого технического задания включают:

* Сроки проекта.
* Требования к качеству, бизнес-метрики качества модели. На основе понятных и прозрачных бизнес-метрик аналитики команды проекта осуществляют их привязку к технологическим метрикам, понятным разработчикам и исследователям.
* Технологические ограничения, связанные с выводом модели в продакшн:
  + ограничения по памяти,
  + ресурсам процессора,
  + скорости работы модели,
  + требования к устойчивости модели к шуму в данных,
  + уровень надежности.

На практике внедрение новейших технологий, фреймворков, наилучших передовых алгоритмов (SOTA, State-of-the-Art) не является самоцелью. Главное в проекте, это решение практической производственной задачи, которое приносит ощутимую пользу, при этом укладывается в заданные ограничения по ресурсам. Если руководитель проекта не может сказать какой бизнес кейс решает проектная команда, причем в цифрах, то такой проект скорее всего будет бесполезен.

Для проектов машинного обучения сложно посчитать точно эффект, так как машинное моделирование не дает никогда 100% точности. Поэтому чаще оценивают – какой эффект для бизнеса принесет улучшение качества работы модели (например, на 0.1%, 1%, 10%). Часто бывает, что потраченные усилия на дообучение модели не дают желаемого коммерческого эффекта, поэтому можно было это дообучение не проводить.

**Пример: если стоимость создания модели оценивается в миллионы рублей (с учетом аппаратного и программного обеспечения, стоимости оплаты работы команды, электроэнергии и остальных затрат), а задача могла бы быть выполнена с использованием ручного труда небольшого числа специалистов низкой квалификации, то создание такой модели должно быть признано неэффективным.**

Модели машинного обучения эффективно применять для задач, на которые у человека потребовалось бы большое количество времени. Важно при этом понимать:

* какую задачу (проблему) мы хотим решить?
* какой ожидаем получить результат?
* какой уже имеется результат и можем ли мы его превзойти?
* как понять, что полученный результат лучше существующих решений, например, ручной обработки?
* какие дополнительные накладные расходы готов нести заказчик

*Пример неэффективной постановки задачи: у нас есть большое количество разных данных, давайте попробуем что-то в них найти или применить алгоритмы “X” и “Y”, чтобы система стала работать лучше.*

*Пример эффективной постановки задачи: мы хотим повысить объем продаж на сайте на 10 процентов за счет использования новой рекомендательной системы для рекомендаций товаров посетителям.*

Без четкой постановки задачи применение технологий машинного обучения будет скорее всего неуспешным. Важно понимать, что для разных участников проекта изложенные выше вопросы будут преобразованы в различные цели, связь между которыми не очевидна даже для самих участников проекта. Например, инженер, решающий задачу автоматизации процесса развертывания, реализует развертывание по расписанию с выполнением необходимых автотестов и выводом решения в продакшн. Такой специалист как правило не знает, что бизнес цель и развертывания этой модели это сократить процент оттока посетителей онлайн-сервиса на 10%. В командах, работающих по agile, обычно приветствуется обмен информацией, в том числе все участники проекта имеют возможность получить обратную связь от рынка и оценить важность бизнес-составляющей проекта. Однако в больших проектах такое не всегда возможно, поэтому важна роль экспертов и бизнес-аналитиков, обеспечивающих связь бизнес-метрик (например, увеличение конверсии посетителей на 10%, повышение среднего чека на 5%, уменьшение оттока покупателей на 1% и т.п.) и технологических метрик (precision, recall, F1 в классификации, MAE, MSE в регрессии и т.п.).

Важным документом, фиксирующим условия выполнения проекта, необходимые требования и ресурсы, является техническое задание (ТЗ). Это типовой документ, часто имеющий юридическую силу (если является приложением к договору). От качественного составления и согласования технических требований в техническом задании очень сильно зависит успех проекта. Наиболее важными в техническом задании проекта машинного обучения являются следующие пункты, от точности описания которых зависит объем ресурсов, сроки, требуемые навыки команды:

* бизнес цели проекта в количественных показателях,
* скорость обучения модели и скорость работы системы в производственных условиях,
* точность модели,
* сроки реализации,
* ограничения в аппаратных ресурсах (производительность процессора, оперативная память, размер жесткого диска, количество серверов, скорость передачи данных),
* необходимость резервирования,
* перечень системам, с которыми необходимо будет интегрировать модель ML,
* бизнес-процесс, включающий в себя использованием ML (включая обработку FP, FN)
* план внедрения системы, как правило внедрение должно идти постепенно, с анализом качества работы

На практике специалиста MLOps часто привлекают к обсуждению технического задания, поскольку эти сильно влияет на выбор инструментов и подходов к решению задач MLOps.

*В этом юните вы узнали:*

* на что необходимо обращать внимание при постановке задачи
  + конкретные и измеримые показатели, которых нужно достичь
  + ограничения ресурсов, которые нельзя нарушать
  + организационные требования: сроки, частота релизов
* что оказывает сильное влияние на задачи MLOps
  + какую задачу (проблему) мы хотим решить?
  + какой ожидаем получить результат?
  + как понять, что полученный результат лучше существующих решений, например, ручной обработки?
  + какие есть ресурсы?

Теперь вы знаете, что MLOps очень востребован в современных проектах машинного обучения в связан с множеством различных задач и технологий. Из примеров, изложенных в данном модуле, становится ясно, что специалистам MLOps необходимо очень много знать для эффективного выполнения своих задач. Пришло время убедиться в этом на конкретных примерах.

*Проверим:*

1. Отметьте примеры правильных бизнес-метрик
   1. **уменьшить отток клиентов на 1%**
   2. сделать сайт покруче
   3. увеличить выручку
   4. **увеличить процент посетителей сайта, совершивших покупку, на 5%**
2. Отметьте примеры правильных технологических метрик
   1. быстрая скорость обучения модели
   2. **F1 не ниже 0.9**
   3. **скорость инференса (работы модели на реальных данных) не более 0.5 секунды**
   4. использование GPU
3. Установите соответствие между бизнес задачей и технологической метрикой

|  |  |
| --- | --- |
| Определить лояльных пользователей | accuracy, precision, recall, F1 |
| Спрогнозировать уровень нагрузки сети | MSE, MAE |
| Разделить магазины на группы (кластеры), обладающие сходством элементов | метрика “силуэт” |

1. Поставьте в соответствие ограничение в техническом задании и объект, на который оказывает влияние это ограничение

|  |  |
| --- | --- |
| скорость инференса | модель машинного обучения |
| объем жесткого диска | количество и тип признаков для обучения |
| частота релизов | инструменты автоматизации |
| возможность появления шума в данных | инструменты мониторинга и резервирования |

# 

# Модуль 1. Юнит 7: Обзор требований к специалистам MLOps

*Об этом юните:*

После изучения предыдущих юнитов данного модуля вы уже знаете, что к специалистам MLOps предъявляются очень высокие требования. Наилучший способ убедится в этом - рассмотреть реальные вакансии компаний на позиции, связанные с MLOps.

*Содержание юнита:*

Каждый этап жизненного цикла проекта и каждый участник проекта создают определенные задачи для MLOps, при этом состав задач MLOps может сильно зависеть от конкретного предприятия, отрасли или проекта. Поэтому от специалистов MLOps требуется широкий кругозор. В качестве примера проанализированы требования к специалистам MLOps на основе открытых вакансий от промышленных компаний. Но даже в таких неопределенных условиях можно применять типовые сценарии и инструменты, которые будут изучаться в данном курсе.В данном юните на примере конкретных описаний требований к вакансиям MLOps разбираются требования.

Вакансия 1

Описание компании:

Небольшая, географически распределенная ИТ-компания реализует проект в сфере работы с медиа контентом.

Описание должности:

MLOps инженер, от 320 000 руб. до вычета налогов, требуемый опыт работы: 3–6 лет, полная занятость, удаленная работа

Задачи:

* участие в жизненном цикле проектов Data Science на всех этапах, включая дизайн и разработку сервисов, планирование ресурсов и обслуживание,
* участие в устранении и расследовании причин сбоев, постмортемы,
* улучшение процессов связанных с обслуживанием систем (отказоустойчивость, масштабирование, работа в режиме высокой нагрузки и т.п.),
* автоматизация процессов сборки, тестирования и доставки приложений;
* автоматизация построения пайплайнов обработки данных, поддержка инфраструктуры,
* взаимодействие с командами разработки и тестирования в части подготовки, запуска и поддержки сервисов.

Требования:

* опыт в области машинного обучения более трех лет,
* опыт использования инструментов для построения инфраструктуры работы с данными, платформ для управления жизненным циклом ML, оркестрацией и процессами (MLflow, Kubeflow, Airflow, DVC или аналоги),
* знакомство с процессами машинного обучения,
* опыт работы с JupyterHub/Lab, Spark, Kafka, Presto, Hive, Flink, MySQL,
* умение проектировать архитектуру приложения и хранилища,
* опыт работы с torch serving, tensorflow serving, triton serving,
* опыт конвертации моделей в ONNX, TensorRT,
* уверенное владение Linux,
* знание языков программирования (Python, Go желательно),
* понимание принципов построения отказоустойчивых решений,
* опыт работы с HighLoad системами,
* понимание работы и опыт организации CI/CD,
* аналитический склад ума,
* аккуратность, ответственность,
* желание разбираться в сложных задачах.

Как видите, кроме опыта работы с необходимым инструментарием MLOps от специалиста требуется широкий круг знаний и навыков: программирование (python, Go), проектирование, администрирование систем. Кроме того, подчеркивается важность так называемых soft-skills (аналитический склад ума, ответственность, готовность к сложным задачам), что обосновано важностью роли специалиста в команде - MLOps инженер в этом проекте активно участвует в развитии продукта и много взаимодействует с другими участниками. Это характерно для проектов небольших компаний-стартапов, в которых участники команды обычно выполняют множество обязанностей.

Вакансия 2

Описание компании:

Команда внутри большой компании (страховой бизнес) создает современную платформу для задач Data Science. В команду требуется MLOps (DevOps) инженер.

Задачи:

* развитие среды исполнения моделей машинного обучения,
* обучение сотрудников отдела практикам и владению инструментами MLops,
* настройка CI/CD для проектов DS,
* помощь в настройке систем сбора признаков, исполнения моделей и мониторинга.

Требования:

* опыт настройки CI/CD в ML,
* уверенное знание Linux,
* знание и опыт применения контейнеризации (Docker, Kubernetes, Openshift),
* опыт настройки и использования Apache Airflow.

Это уже совсем другой проект, сильно отличающийся от первой вакансии. От специалиста MLOps ожидается уже меньшее количество навыков, так как команда работает внутри большой компании, в которой уже многие процессы налажены и информационные системы внедрены. Поэтому не требуется навыком проектирования систем и хранилищ.

Вакансия 3

Описание компании:

Команда проекта создает модели машинного обучения для беспилотного транспорта (self-driving).

Описание требований:

* MLOps Engineer (AI Driving Data), требуемый опыт работы: 3–6 лет, полная занятость, полный день,
* улучшение качества ML-решений,
* выбор и внедрение инструментов MLOps, поддержка инфраструктуры для обучения моделей, создание и поддержка ML-пайплайнов,
* организация обработки данных от беспилотников, участие в создании хранилища для этих данных,
* стек: MLFlow, Kubeflow, DVC, ETL, CI/CD/CT,
* опыт разработки на Python от трех лет,
* знание алгоритмов и методов машинного обучения,
* опыт построения инфраструктуры работы с данными,
* опыт организации CI/CD,
* опыт разработки на С++,
* знание лучших практик DevOps.

Перечень задач:

* внедрить и развивать платформу полного жизненного цикла ML-моделей,
* собирать и запускать ML-пайплайны,
* поддерживать модели, наборы данных и параметры в системах CI/CD/CT,
* поддерживать модели в проде, решать сложные проблемы.

Вакансия 4

Описание команды и проекта:

Команда внутри большой компании создает набор продуктов и сервисов, предназначенных для решения задач кибербезопасности: противодействие внутреннему и внешнему мошенничеству, защита инфраструктуры организации (банковский сектор), анализ рисков и угроз, аудит кибербезопасности и безопасной разработки.

Ядро платформы - кластер Hadoop (Cloudera), хранит более 10Пб данных, которые обрабатываются на более чем 5000 процессорных ядер и 50Тб оперативной памяти в поточном и пакетном режимах. Состав разрабатываемой системы:

* среда исследования и моделирования - продукт, позволяющий дата-сайнтистам проводить исследования на промышленных данных. Продукт представляет из себя кластер Hadoop (Cloudera), поднимающийся on-demand, а так же набор сервисов для работы с данными на кластере - Spark, Hive, Apache Kafka, Apache Ni-fi, Apache Flink, Clickhouse, Jupyterhub, Apache Airflow,
* Model Execution Framework (MEF) - фреймворк, в основе которого лежит Opehshift, позволяющий типовым образом деплоить обученные AI-модели на тестовые и промышленные среды.

Обязанности:

* развитие среды исследования и моделирования,
* деплой моделей в MEF,
* автоматизация процесса сопровождения,
* устранение инцидентов,
* обеспечение надежности сервисов,
* ведение документации,
* мониторинг,
* наставничество.

Требования:

* релевантный опыт работы в качестве DevOps инженера,
* понимание как работает Hadoop и сервисы его экосистемы,
* уверенное владение Linux (желательно RHEL),
* умение писать роли и плейбуки Ansible,
* умение создавать пайплайны в Jenkins, либо аналогичных системах,
* Git на уровне создать ветку, выставить PR, пройти code review и “вмержить” в мастер,
* желательно знакомство с Zabbix,
* опыт работы с Kubernetes/Openshift,
* владение Python для дебага моделей.

Из описания видно, что компания уже обладает развитой мощной экосистемой для работы с большими данными, имеет отлаженные процессы и настроенные инструменты, а также уже созданные продукт, который необходимо развивать. Поэтому основной акцент на задачах, близких к эксплуатации и развитию системы, а также обучению (наставничество).

Вакансия 5

Описание компании и проекта:

We at Samsung AI Center are looking for a MLOps engineer for several computer vision projects.

We are doing R&D projects in generative computer vision and 3D computer vision, which require a lot of data processing, storing and experiments and we want to improve our main research and production pipelines. Samsung AI Center Moscow has been founded in 2018. Project topics include:

* Generation of photorealistic human avatars
* Point clouds-based 3D image and video synthesis
* Techniques for image manipulation using generative networks
* Training neural networks using Bayesian methods
* Creation of multi-agent systems
* Single-view and multi-view 3D reconstruction
* Localization and navigation using 3D vision

Responsibilities:

* Benchmarking our hardware and software setups on clusters
* Collecting and serving different docker images and repository
* Creating new tools to improve our experiments tracking and deployment pipelines
* Continuously support our engineers in best code practices and automation of all development pipelines
* Introduce CI/CD tools in our working process
* Probably take part in NAS and mobile networks papers
* Maintain our local clusters, monitor jobs and support software updates
* Write and run scripts in python and bash to process raw data, collect statistics, clean, store dataset and show real time previews

Requirements:

* Professional experience with
* Maintaining linux-based servers and clusters, including CUDA
* Python, Pytorch, Tensorflow, Bash scripts
* Any frontend development framework (e.g. Angular, React, Vue.js)
* Docker, Kubernetes, KubeFlow
* Nvidia NGC docker repo, Nvidia Triton Inference server
* Basic data exchange protocols, e.g. GRPC, JSON, Mosquitto Buffers
* Experience with data storage formats, such as SQL, LMDB, HDFS and parallel/distributed files systems, e.g. GPFS
* As a plus:
  + Got ML and DL algorithms knowledge
  + Experience with writing/reimplementing ML/DL papers
  + Contributed to open-source github projects
  + Have a track of contributions to IT products and services
  + Experience with TFLite and Mediapipe
  + Professional C++/CUDA, OpenCL or OpenGL experience
  + Experience with MLFlow/ Neptune.ai/ Weights and Biases and other experiments tracking software
  + Experience with Fairscale, Torch DDP, Pytorch Lightning, Tensorflow Extended, GPipe

Это самая “богатая” по требованиям вакансия из приведенных в данном юните, от одной из ведущих исследовательских лабораторий в сфере искусственного интеллекта Samsung AI Center. Компания использует передовой стек технологий, в том числе такие инструменты, которые не входят в данный курс.

Итак, из рассмотренных вакансий ясно, что требования к специалисту MLOps могут быть самые различные, это очень зависит от компании и проекта. Но зачастую эти требования гораздо шире, чем просто автоматизация процесса развертывания модели, поэтому специалисту MLOps для эффективного выполнения своих обязанностей необходимо вникать в предметную область (финансы, медицина, телекоммуникации, страховое дело, …), а также понимать цели и задачи смежных подразделений.

*Из этого юнита вы узнали, что*

* требования, предъявляемые к специалистам MLOps, обеспечивающим автоматизацию в проектам машинного обучения, очень высокие, что мы увидели на примере рассмотренных вакансий,
* часто от специалиста MLOps требуется погружение в смежные задачи (команды, заказчиков) для лучшего понимания проекта и более эффективного решения задач
  + понимание программного кода,
  + умение работать с инструментами, используемыми командой, понимание правил их настройки,
  + умение вникнуть в техническое задание и сформулировать требования к инфраструктуре на его основе.
* сложно составить “универсальный” перечень знаний, который подойдет для любого проекта, однако есть “необходимый” уровень знаний, востребованных в каждом проекте
  + знание основных инструментов автоматизации (python, bash, Jenkins или аналоги)
  + программирование скриптов
  + умение создавать виртуальные среды
  + настройка оборудования и программного обеспечения.

# Юнит №8

*В этом модуле были изучены следующие вопросы:*

* Зачем изучать MLOps?
  + использование этой идеологии позволяет сократить время и издержки, повысить качество проекта машинного обучения
* Какие знания нужны на старте?
  + разработка программного обеспечения (подходы, архитектура, технологии)
  + операционные системы, чаще всего linux
  + машинное обучение
  + скрипты автоматизации (python, bash)
* Из чего состоит MLOps?
  + автоматизация процессов
  + версионирование объектов проекта
    - датасеты
    - пайплайны обработки данных
    - модели машинного обучения
    - виртуальные среды
* Как архитектура и жизненный цикл проекта машинного обучения влияют на задачи MLOps?
  + на каждом этапе жизненного цикла проекта машинного обучения возникают свои специфические задачи MLOps
  + архитектура решения и имеющиеся ограничения по ресурсам оказывают влияние на задачи и инструментарий MLOps
* Какие инструменты используются в MLOps?
  + версионирование: git, dvc
  + виртуализация: venv, virtualenv, poetry
  + контейнеризация: docker, docker-compose, kubernetes
* Как постановка задачи и бизнес-требования влияют на MLOps?
  + бизнес-требования трансформируются в технические требования к инфраструктуре и технические метрики проекта.
* Какие требования предъявляют компании к специалистам MLOps (на примере вакансий hh.ru)?

# Дополнительные материалы

LEAN\_DS <https://ods.ai/tracks/lean-ds-df2020>

<https://www.atlassian.com/software/jira>

<https://www.atlassian.com/ru/software/confluence>

https://cloud.google.com/architecture/mlops-continuous-delivery-and-automation-pipelines-in-machine-learning#devops\_versus\_mlops

# Юнит №9 Практическое задание к модулю

Цель задания:

В этом задании вы решите простую и во многом учебную задачу по созданию автоматического конвейера проекта машинного обучения. Подобный подход с применением простых скриптов автоматизации для “склейки” отдельных частей конвейера используется в небольших проектах. Чаще же для автоматизации используется специализированное программное обеспечение, например, Jenkins.

Содержание задания:

Необходимо из “подручных средств” создать простейший конвейер для автоматизации работы с моделью машинного обучения. Отдельные этапы конвейера машинного обучения описываются в разных python–скриптах, которые потом соединяются (иногда используют термин “склеиваются”) с помощью bash-скрипта.

Этапы:

1. Создайте python-скрипт (data\_creation.py), который создает различные наборы данных, описывающие некий процесс (например, изменение дневной температуры). Таких наборов должно быть несколько, в некоторые данные можно включить аномалии или шумы. Часть наборов данных должны быть сохранены в папке “train”, другая часть в папке “test”.
2. создайте python-скрипт (model\_preprocessing.py), который выполняет предобработку данных, например, с помощью sklearn.preprocessing.StandardScaler.
3. создайте python-скрипт (model\_preparation.py), который создает и обучает модель машинного обучения на построенных данных из папки “train”.
4. создайте python-скрипт (model\_testing.py), проверяющий модель машинного обучения на построенных данных из папки “test”.
5. Напишите bash-скрипт (pipeline.sh), последовательно запускающий все python-скрипты.

Подготовленные скрипты необходимо опубликовать в git репозитории, ссылку на который необходимо предоставить как результат выполнения задания.

**Творческое задание на вебинар:**

Предположим, что заказчик поставил перед командой задачи: разработать прогнозную модель для зарплат специалистов MLOps на два года вперед. Заказчик хочет Определять уровень зарплат в среднем по рынку, чтобы нанимать в течении двух лет 10 специалистов и не переплатить более 500.000 т.р. Решение заказчик хочет эксплуатировать на своих серверах (поскольку не хочет раскрывать персональные данных своих сотрудников). Однако сервера у заказчика ненадежные, технические описания он не предоставляет.

1. Предложите технические метрики машинного обучения, которые могли бы соответствовать этой постановке задачи и инструменты для ее решения.
2. С какими сложностями может столкнуться специалист MLOps при реализации задач в такой постановке
3. Какой перечень мероприятий можно предложить для минимизации рисков.