Санкт-Петербургский государственный университет
Кафедра системного программирования
Группа 24.М41-мм

Обзор принципов, подходов и инструментов MLSecOps, используемых при реализации ML-пайплайнов

Хокимзода Муборакшои Иноятулло

Отчёт по учебной практике

Научный руководитель:
доцент кафедры СП, к.ф.-м.н.
Луцив Д.В.

Консультант:
старший преподователь каф. СП,
Андриенко В.А.

Санкт-Петербург 2024

Содержание

Введение

Сегодня машинное обучение широко применяется в различных сферах хозяйсвенной деятельности от финансов до здравоохранения. С развитием технологий искуссвенного интеллекта расширяется также область угроз безопасности при его использовании. Пайплайны обработки данных, то есть последоввательности шагов по подготовке информации для последующего обучения моделей, подвержны различным атакам, что может привести к недостоверности получаемых предсказаний.

Существует около десяти ключевых этапов системы MLSecOps для крупных языковых моделей(LLM), направленных на обеспечение защиты, целостности и качества машинных моделей в течение всего их жизненного цикла. В предлагаемую в рамках данной работы архитектуру MLSecOps будут интегрированы механизмы мониторинга, управления рисками и реагирования на инциденты для снижения вероятности подобных проблем в любой фазе работы системы.

Целью иследования является обзор архитектурной схемы для системы анализа безопасности в машинном обучении. Объектом изучения являются все этапы жизненного цикла моделей от сбора данных до их развертывания. Существующие решения недостаточно учитывают угрозы, связанные с большими объемами данных и автоматизацией процессов, что значительно увеличивает риск утечек информации, манипуляций с моделями и других видов атак. В предлагаемую в рамках данной работы архитектуру MLSecOps будут интегрированы механизмы мониторинга, управления рисками и реагирования на инциденты для снижения вероятности подобных проблем в любой фазе работы системы.

Постановка задачи

Обзор существующих принципов, подходов и инструментов MLSecOps, обеспечивающей интеграцию передовых практик безопасности и надежности в весь процесс разработки, развертывания и эксплуатации пайплайнов машинного обучения. Для достижения поставленной цели определяются следующие подзадачи:

- Исследование концепций и принципов MLSecOps
- Анализ существующих архитектурных решений и инструментов для обеспечения безопасности в ML
- Разработка комплексной архитектурной схемы MLSecOps включающей все этапы жизненного цикла моделей (от сбора данных до развертывания)
- Визуализация разработанной архитектуры для удобства понимания и внедрения
- Формирование подробных рекомендаций по практическому внедрению системы MLSecOps, включая методы управления рисками, мониторинг безопасности на всех этапах и реагирования на инциденты.

Обзор

MLSecOps является комплексным подходом к безопасности машинного обучения, охватывающим все процессы выбора, обучения и запуска ML-моделей. Вместо того чтобы рассматривать безопасность как отдельную функцию, как это часто делается в традиционных процессах, подходы MLSecOps интегрируются непосредственно в фундаментальную основу рабочего процесса нашего проекта. Основными критериями являются: безопасность данных, защита моделей, защищенный деплоймент и непрерывное развертывние.

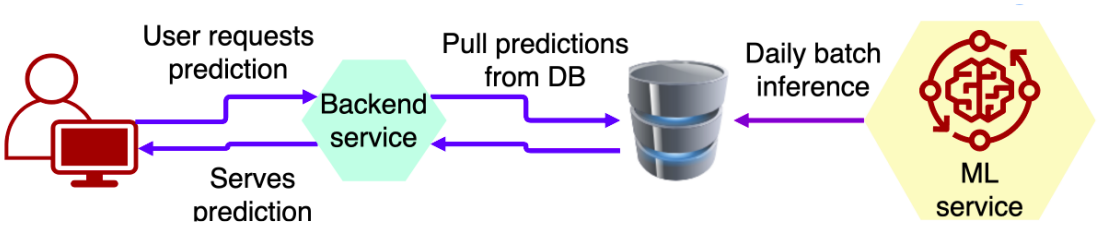
Безопасность данных включает защиту информации на этапах сбора, хранения и обработки. Это достигается за счёт шифрования, анонимизации и строгого контроля доступа. Защита моделей направлена на предотвращение атак, (включая adversarial attacks) и утечек информации о моделях. Безопасный деплоймент предполагает надежное развертывание моделей в продакшн-среде с учетом управления версиями и конфигурациями. Наконец, мониторинг и управление позволяют постоянно отслеживать состояние моделей и инфраструктуры, выявлять аномалии и оперативно реагировать на потенциальные угрозы.

Деплой ML-модели в продакшн

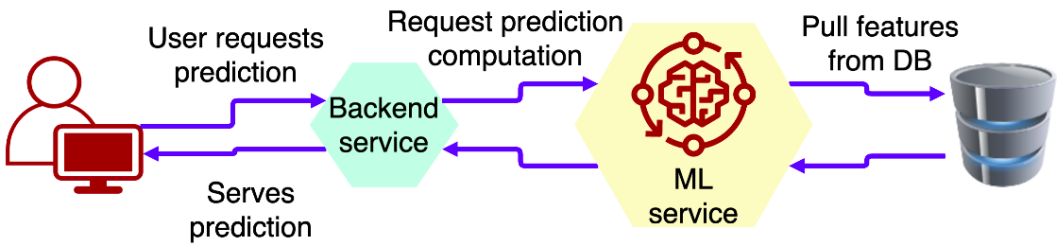
В процессе развертывания моделей машинного обучения существует несколько подходов, каждый из которых имеет свои особенности и подходит для различных сценариев использования.

Одним из таких подходов является пакетное развертывание (Batch Deployment). В этом случае предсказания модели вычисляются с определенной периодичностью, например, ежедневно. Результаты сохраняются в базе данных, что обеспечивает их доступность при необходимости. Однако данный метод имеет недостаток: он не позволяет использовать самые

свежие данные, и прогнозы могут быстро устаревать, что снижает их актуальность.

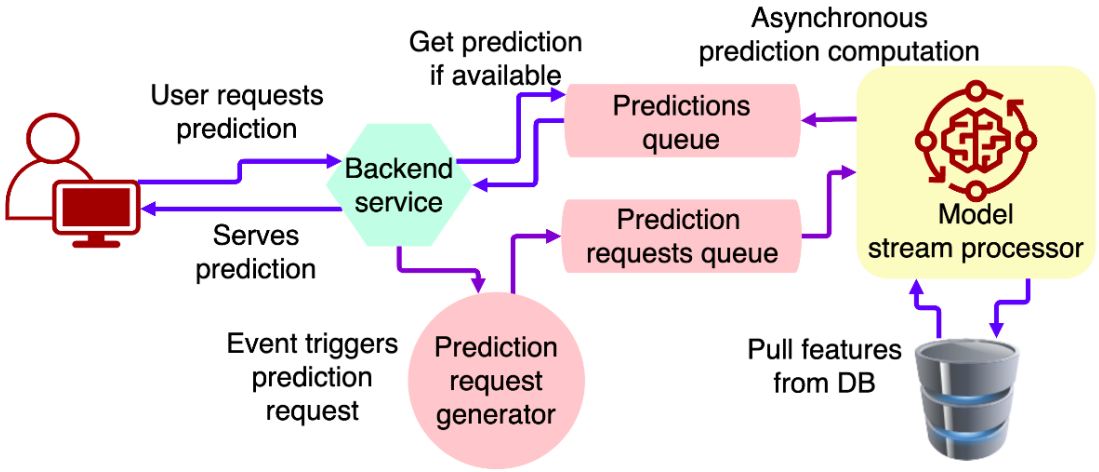


Развертывание в реальном времени (Real-time Deployment) - процесс является синхронным: пользователь запрашивает прогноз, запрос передается на бэкэнд-сервис через HTTP API, который затем обращается к ML-службе для получения результата. Такой способ отлично подходит для получения персонализированных предсказаний, учитывающих актуальную контекстную информацию (например, время суток или последние поисковые запросы пользователя). Однако основным ограничением этого подхода является необходимость ожидания возврата прогноза, что может создавать узкие места. Для эффективной обработки множества одновременных запросов требуется использование многопоточных процессов и вертикальное масштабирование путем добавления дополнительных серверов.

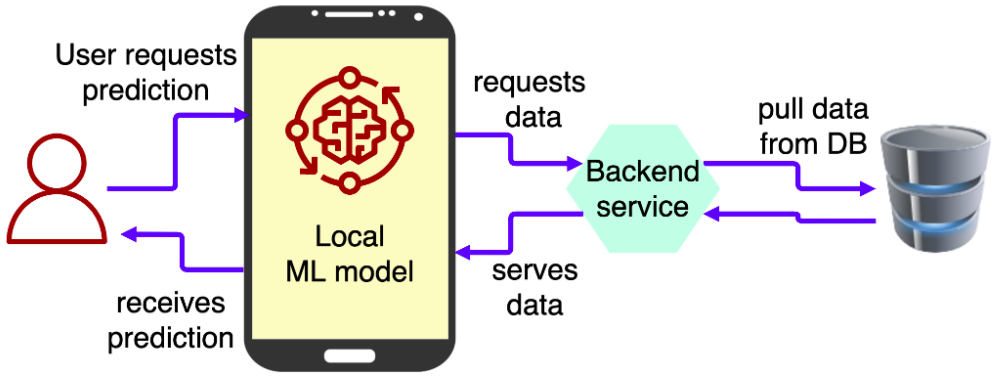


Потоковое развертывание (Streaming Deployment) — позволяет использовать более асинхронный процесс: действие пользователя, например загрузка страницы социальной сети, инициирует расчет предасказаний - в

частности, начинается ранжирование ленты новостей еще до начала прокрутки. К моменту взаимодействия с содержимым (например, началу прокручивания), лента уже готова к отображению пользователю. Запросы помещаются в брокер сообщений типа Kafka, где модель машинного обучения обрабатывает их по мере готовности очереди. Этот подход существенно освобождает серверную часть и обеспечивает эффективное использование вычислительных ресурсов благодаря оптимизации обработки задач из очередей. Результаты прогнозов также могут храниться в этой же системе очередей, позволяя сервису использовать их по мере необходимости.



Развертывание на Edge-устройствах (Edge Deployment) предполагает установку моделей машинного обучения непосредственно на клиентских устройствах - таких как смартфоны или устройства Интернета вещей (IoT). Данный подход обеспечивает минимальную задержку в получении результатов предсказаний благодаря локальной обработке данных без необходимости постоянного онлайн-соединения. Однако ключевым требованием является создание компактных моделей, способных эффективно функционировать на ограниченном по рессурсам устройствах.



Архитектуры внедрения ML-пайплайнов

Запуск пайплайна — это первый шаг на пути к реализации надежного и безопасного процесса машинного обучения. Он происходит автоматически, в ответ на заранее заданное событие, такое как слияние кода, коммит, таймер или задание в Jenkins. Такой подход позволяет обеспечить непрерывную интеграцию и постоянное улучшение модели, что, в свою очередь, способствует внедрению CI/CD — концепции, направленной на защиту данных и моделей от кибератак. После запуска пайплайна происходит загрузка артефактов, таких как датасеты, обучающие данные и другие важные материалы, которые необходимы для обучения и представления модели. Эти артефакты сохраняются в надежном месте, например, на S3 Buckets, Nexus или через FTP, что гарантирует их доступность и сохранность в будущем.

Использование подобных подходов в рамках ML-пайплайнов позволяет существенно снизить риски, связанные с безопасностью данных и моделей. Это, в свою очередь, повышает доверие к результатам машинного обучения и позволяет соответствовать строгим нормативным требованиям. В результате создаются системы, способные обеспечить непрерывность бизнес-процессов и защитить от различных атак.Существует несколько основных обзор подходов и инструментов, которые могут быть использова-

ны для реализации MLSecOps. Одним из популярных решений является Tensorflow Extended (TFX) — платформа для оркестрации ML-пайплайнов, позволяющая связывать интеграцию различных инструментов безопасности. В рамках TFX определены основные компоненты для сбора, обработки данных, обучения моделей и их развертывания, что обеспечивает поддержку всего жизненного цикла ML-проекта.

На следующем этапе применяются различные инструменты безопасности и контроля качества, такие как Gitleak, Sonarqube, Trivy, Owasp, Dependency-Check, NB Defense, OpenPubKey, а также Compliance-Checker и собственные скрипты для проверки соответствия наборов данных установленным политикам и стандартам. Эти инструменты сканируют и тестируют код и его зависимости на предмет уязвимостей и проблем с качеством, а также проводят проверку обучающих данных на безопасность. Далее применяется Quality Gate, который проверяет соответствие артефактов стандартам безопасности и качества, установленным политиками и правилами. Только в случае соблюдения всех требований артефакты переходят к следующему этапу.

Еще одним важным инструментом является Kubeflow — оркестрационная платформа для машинного обучения на базе Kubernetes. Kubeflow предоставляет мощные средства управления пайплайнами, моделями и их развертываниями, что позволяет автоматизировать процессы и интегрировать механизмы безопасности на всех этапах. Hashicorp Vault используется для управления секретами и шифрования данных, обеспечивая безопасное хранение и доступ к конфиденциальной информации.

На этапе обучения модели используются технические методы, такие как раннее прекращение обучения (early stopping) и кросс-валидация (KFold), что позволяет обеспечить точность и надежность обучаемой модели. Затем модель оценивается по заранее определенным метрикам и пороговым значениям, чтобы убедиться, что она соответствует установленным

критериям производительности.

Следующий этап включает тестирование модели с фокусом на уязвимости, соответствующие Owasp Top 10 для приложений на базе LLM, такие как инъекции запросов (prompt injection), отравление данных (data poisoning) и утечка конфиденциальной информации. Это позволяет выявить и устранить потенциальные уязвимости, обеспечивая безопасность модели. После успешного тестирования модель проходит через еще один контроль качества, где вновь проверяется соответствие всем стандартам перед финальными шагами. Это гарантирует, что модель соответствует всем требованиям перед её одобрением.

На этапе одобрения модели и токенизатора создаются артефакты, такие как файлы Model.H5 и tokenizer.Pickle, которые подписываются для подтверждения их целостности и происхождения. Затем подписанные артефакты сохраняются в безопасных местах, таких как бакеты S3, Nexus или через FTP, обеспечивая их надежное хранение.

Для мониторинга и визуализации метрик часто используют Prometheus и Grafana. Первый позволяет собирать и хранить метрики производительности и состояния системы, а Grafana обеспечивает возможность визуализации сохраненных данных. Elk Stack (Elasticsearch, Logstash, Kibana) отлично подходит для централизованного логирования и анализа событий. С помощью этих инструментов можно эффективнее обнаруживать и реагировать на внештатные ситуации.

Все вышеперечисленные инструменты позволяют интегрировать механизмы безопасности на разных этапах ML-пайплайна. Это гарантирует полноценную безопасность как данных, так и моделей. Выбор конкретных инструментов зависит от требований проекта, инфраструктуры и предпочтений команды.

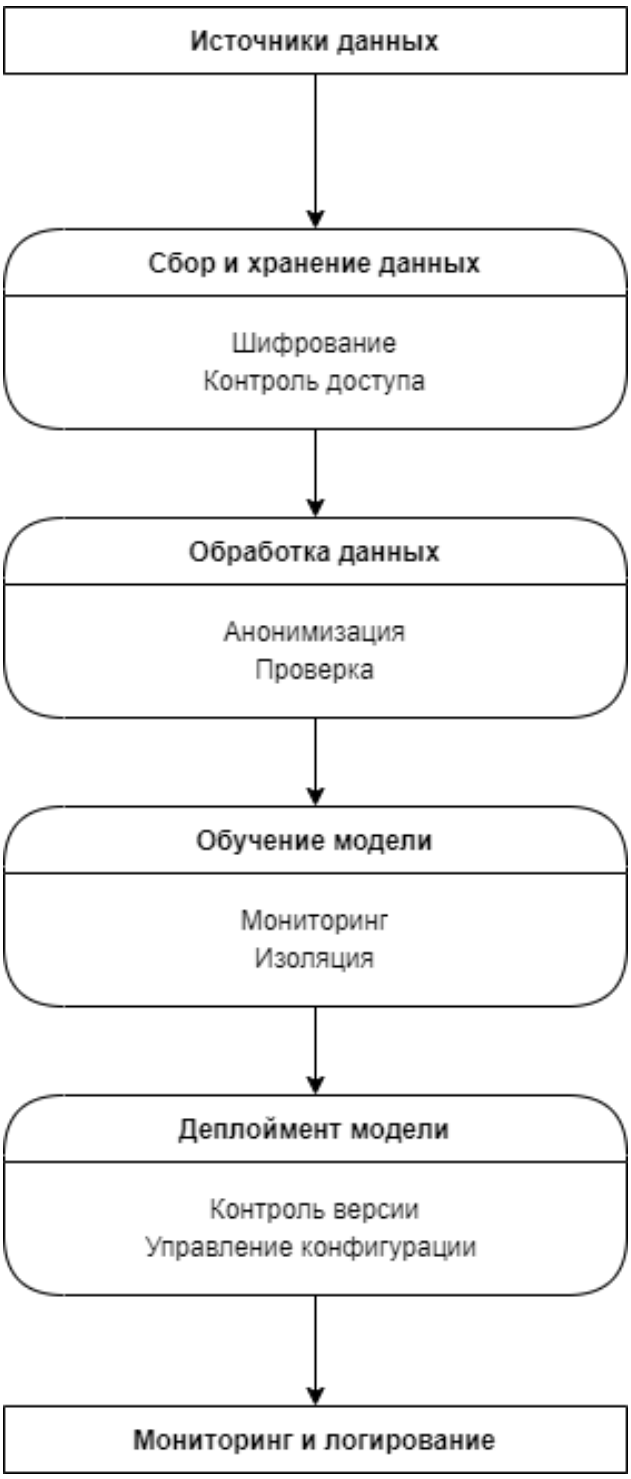
На основе проведенного анализа существующих подходов и инструментов была разработана архитектурная схема MLSecOps для ML-пайплайна,

который будет использоваться для предсказания о состоянии сердца пациентов. Ее составными элементами будут следующие компоненты, каждый из которых соответствующим образом интегрирован с механизмами обеспечения безопасности и надежности:

- Сбор и хранение данных: в этом компоненте будут выступать различные Cloud Storages, обеспечивающие надежное хранение сырых данных с включенным шифрованием по умолчанию. Регулирование доступа к хранилищу осуществляется системой идентификации и присвоения прав.

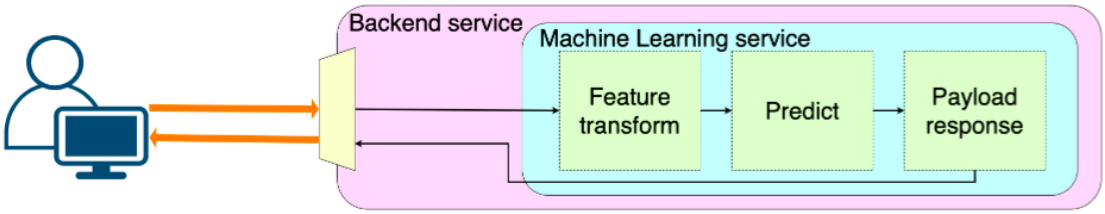
- Обработка данных ETL: эту задачу будет выполнять Apache Airflow. На данном этапе данные проходят анонимизацию и проверку целостности, что защищает персональные данные и предотвращает потенциальный дата-лик. Apache Airflow помогает автоматизировать процесс, повышая эффективность и надежность пайплайна.

- Обучение модели: обучение модели осуществляется в Kubeflow, который обеспечивает оркестрацию процесса обучения в деизолированных средах[3].


Источники данных
Сбор и хранение данных
Шифрование
Контроль доступа
Обработка данных
Анонимизация
Проверка
Обучение модели
Мониторинг
Изоляция
Деплоймент модели
Контроль версии
Управление конфигурации
Мониторинг и логирование

Особенности внедрения в Real-time сервисы

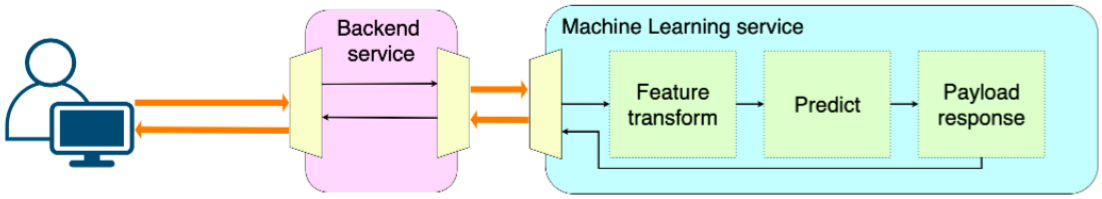
Внедрение моделей машинного обучения в real-time сервисы требует тщательного выбора архитектуры, обеспечивающей эффективность, масштабируемость и надёжность системы. Одним из подходов является монолитная архитектура, при которой кодовая база модели непосредственно интегрируется с кодовой базой бэкэнд-сервиса. Такой подход предполагает тесное взаимодействие между специалистами по машинному обучению и разработчиками бэкенда. Однако он имеет ряд недостатков: процесс непрерывной интеграции и доставки (CI/CD) затрудняется из-за необходимости проведения юнит-тестов для сервисов машинного обучения, а также увеличение размера модели и требования к вычислительным ресурсам создают дополнительную нагрузку на серверы бэкенда. Монолитный подход целесообразно использовать лишь в тех случаях, когда инференс (использование) модели не требует значительных вычислительных ресурсов и минимально влияет на общую производительность системы.



Альтернативой монолитной архитектуре является подход, при котором модель машинного обучения развертывается на отдельном сервере (микросервисной архитектуры). Такой метод позволяет:

- Отделить процессы разработки и развёртывания модели от основной бизнес-логики сервиса, что упрощает работу как ML-инженеров, так и разработчиков бэкенда.

- Гибко управлять ресурсами и масштабировать модель с использованием балансировщиков нагрузки при увеличении объема запросов.
- Таким образом, создание систем мониторинга и логирования становится более простым благодаря функциональной независимости модели от основного сервиса.
- Обеспечивает лучшую поддерживаемость и надёжность системы, позволяя моделям быть сложными без негативного влияния на остальную инфраструктуру.



Для более сложных и масштабируемых систем часто применяется микросервисная архитектура:

- Каждая часть ML-пайплайна реализуется в виде отдельного сервиса:
  - Предобработка данных
  - Тренировочный процесс моделей

В такой подходе требуются высокий уровень зрелости и эффективное управление жизненным циклом моделей (MLOps), поскольку:

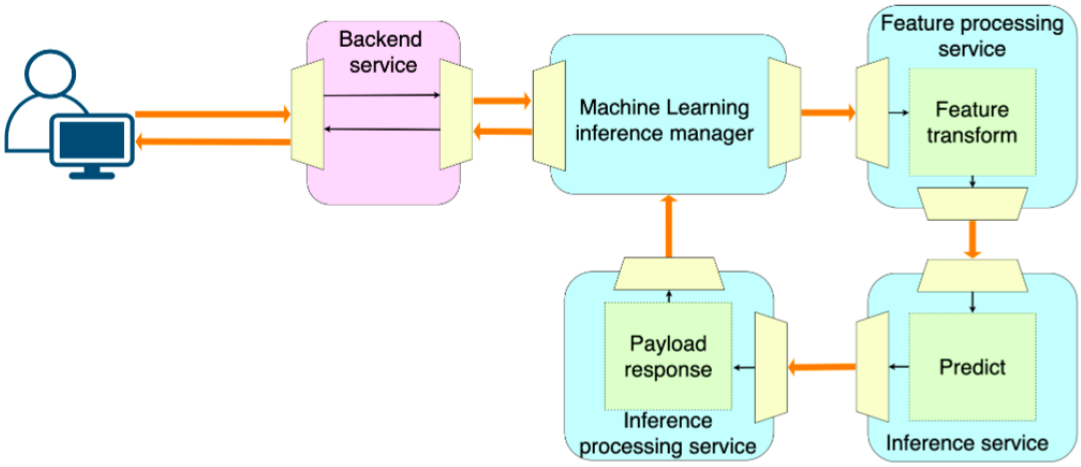
- Необходимо управлять множеством взаимосвязанных сервисов.

Микросервисная архитектура особенно полезна, когда:

- Различные компоненты обработки данных используются в нескольких моделях:
- Пример: ранжирование рекламы для Facebook и Instagram с общим аукционным процессом.

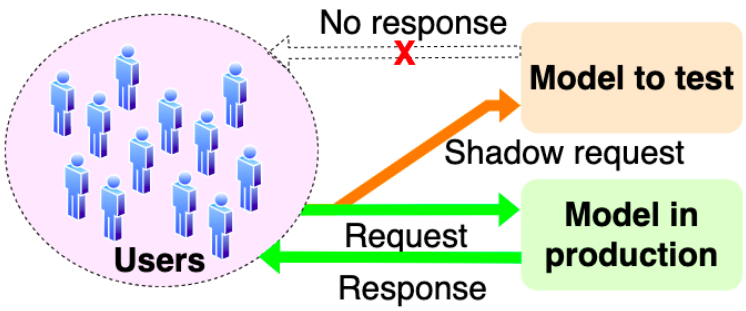
Для управления сложностью микросервисов критически важны следующие инструменты и практики:

1. Использование оркестраторов, таких как Kubernetes:
   - Автоматизация развертывания (deployment)
   - Масштабирование сервисов в зависимости от нагрузки
2. Организация CI/CD процессов для каждого микросервиса.
3. Централизованное управление конфигурациями и секретами.
4. Мониторинг, логирование и централизация логгирования (например, ELK Stack).
5. Разработка API-интерфейсов между сервисами с использованием стандартов обмена данными.
6. Применение принципов DevOps для обеспечения непрерывной интеграции и доставки.
7. Регулярные рефакторинг и оптимизация архитектуры в соответствии с изменениями требований к системам машинного обучения.

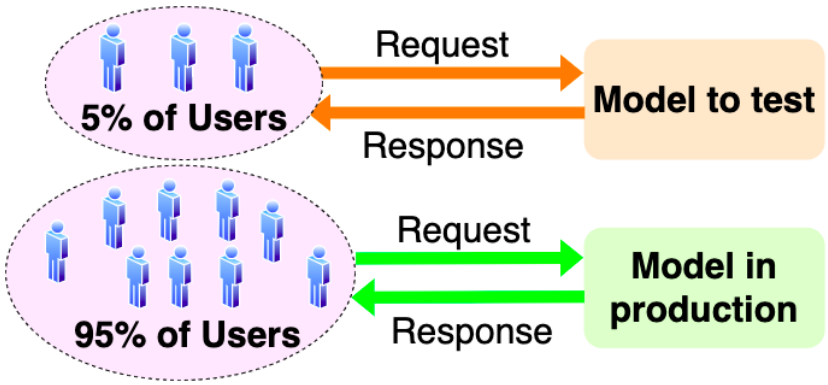


Тестирование модели

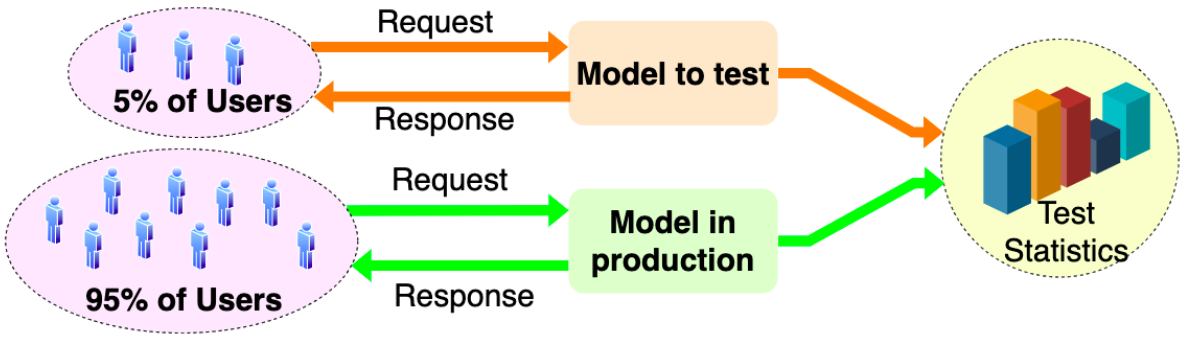
Один из способов проверки новой модели — использование shadow-режима. В этом режиме запросы пользователей одновременно отправляются как на действующую модель в продакшене, так и на новую тестовую модель. Однако пользователи видят только ответы действующей модели. Такой подход позволяет проверить распределение предсказаний новой модели и сопоставить его с данными, полученными на этапе обучения. Shadow-режим также помогает протестировать сервисную часть до выпуска модели в эксплуатацию; однако он не даёт полноценной оценки влияния новой модели на пользователей, так как ответы текущей модели всё ещё доминируют.



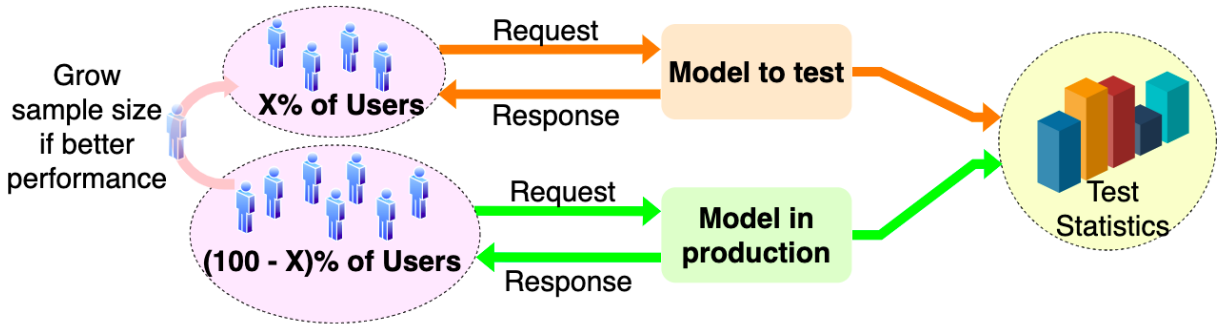
Следующим шагом может быть canary deployment, представляющий собой end-to-end тестирование всей системы с ограничением объема пользователей, взаимодействующих с новой моделью. Обычно предсказания от новой модели предоставляются только небольшой группе пользователей или используются внутренними сотрудниками компании; это можно сравнить с интеграционным тестированием в продакшене. Однако, как и при shadow-режиме, canary deployment не даёт чёткой оценки качества модели по критерию эффективности.



Для проверки качества модели и оценки её влияния на бизнес-метрики применяется A/B-тестирование. В этом случае небольшой части пользователей предоставляются предсказания новой модели, в то время как поведение этой группы сравнивается с действиями пользователей, продолжающих взаимодействовать с действующей моделью. Основное внимание при A/B-тестировании уделяется таким метрикам, которые сложно измерить в рамках офлайн-экспериментов, как выручка или уровень удовлетворённости пользователей. Для проверки гипотез используется строгая математическая статистика, что обеспечивает надёжность и точность результатов.

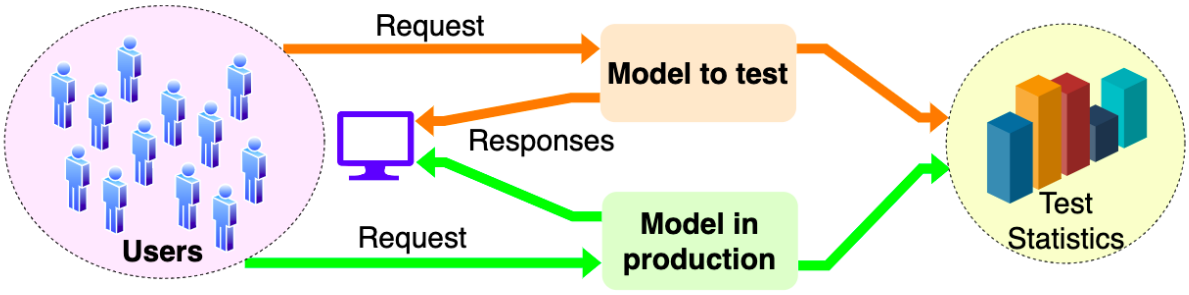


Альтернативным подходом является использование многоруких бандитов, где размер группы пользователей, взаимодействующих с каждой моделью, динамически адаптируется в зависимости от их производительности. Это даёт возможность исследовать разные варианты и направлять больше ресурсов на те модели, которые демонстрируют наилучшие результаты. Такой метод особенно актуален при работе с динамичными системами, где качество моделей может изменяться со временем. Более того, многорукий бандит позволяет одновременно тестировать не только две, но десятки или даже сотни моделей, что значительно повышает эффективность использования данных.



Ещё одним методом является Interleaving, предполагающий объединение моделей: при этом пользователям одновременно демонстрируются

результаты работы нескольких моделей. Данный подход позволяет оценить предпочтения и реакцию пользователей, например, сочетая рекомендации двух систем или списки поисковых результатов. Однако этот метод имеет ограничения: он сосредоточен преимущественно на выявлении пользовательских предпочтений и может игнорировать другие существенные бизнес-показатели, такие как конверсия или долгосрочная прибыль.



Выбор подхода к тестированию зависит от целей и задач эксперимента, а также от доступных ресурсов. Каждый из методов имеет свои сильные и слабые стороны, которые необходимо учитывать при разработке стратегии тестирования модели в продакшене[4].

Безопасность в MLSecOps

Конвейер MLSecOps для больших языковых моделей (LLM) включает около десяти этапов, каждый из которых предназначен для обеспечения безопасности, целостности и качества моделей машинного обучения на протяжении всего их жизненного цикла. Вот анализ каждого этапа и его цели:

На начальном этапе пайплайн инициализируется в ответ на определённые события: запросы на слияние (merge requests), коммиты кода, запланированные временные интервалы (таймеры) или выполнение заданий в Jenkins. Автоматизация запуска пайплайна при возникновении этих событий обеспечивает непрерывную интеграцию и развертывание, что позволяет поддерживать актуальность и качество проекта без необходимости ручного вмешательства.

Загрузка артефактов включает загрузку всех необходимых компонентов, обеспечивающих возможность обучения и оценки моделей: наборы данных, предварительно обученные модели, ноутбуки Jupyter и требования к библиотекам. Загрузка этих артефактов гарантирует, что все ресурсы готовы для подготовки данных и последующего обучения модели. После загрузки артефакты сохраняются в различных хранилищах — таких как бакеты AWS S3, репозитории Nexus или через FTP. Важно поддерживать список доверенных внешних библиотек и компонентов, используемых в процессе обучения; хранить их на собственных системах и регулярно проводить оценку их уязвимостей. Прямое скачивание этих компонентов из интернета не рекомендуется из-за возможных рисков, связанных с безопасностью и надежностью.

Проводится тщательное тестирование безопасности всех артефактов перед началом процесса обучения модели. Для этого применяются различные инструменты: Gitleak, SonarQube, Trivy, OWASP Dependency-Check, NB Defense, compliance-checker и собственные скрипты/инструменты (на-

пример OpenPubKey). Эти средства позволяют:

- детально анализировать исходный код;
- выявлять потенциальные риски безопасности;
- оценивать качество кода;
- подтверждать целостность и происхождение артефактов.

Кроме того, данные инструменты обеспечивают соответствие артефактов установленным политикам и стандартам. Инструменты типа compliance-checker или собственные скрипты используются для проверки обучающих наборов данных:

- гарантируя отсутствие конфиденциальной информации;
- уделяя особое внимание категориям персональных данных, конфиденциальных ключей и другой чувствительной информации.

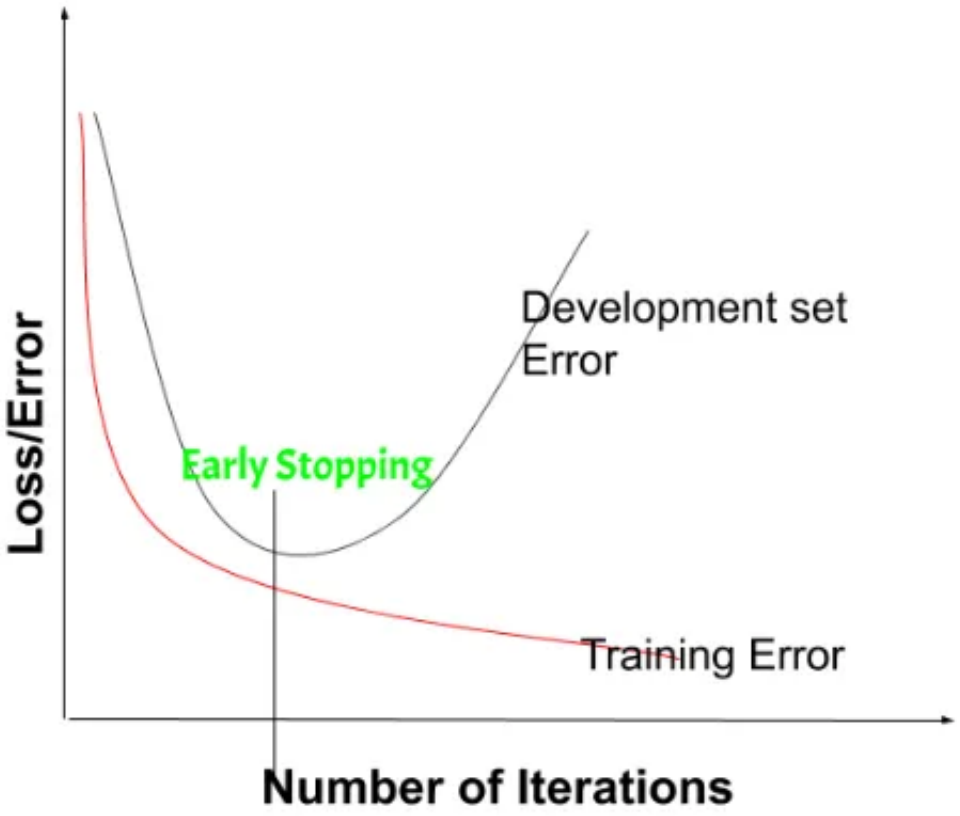
Это позволяет обеспечить полное соответствие требованиям безопасности, предотвращая возможные утечки или нарушения.Ниже приведены основные категории информации, требующие тщательного изучения:

Категория	Характерные примеры	Рекомендованная мера
Персонально-идентифицируемая информация (PII)	- Имена - Адреса - Номера телефонов - Электронные адреса - Номера социального страхования - Номера кредитных карт - Номера паспортов	Удалить или анонимизировать Использовать псевдонимы или редактирование
Конфиденциальные персональные данные	- Медицинская информация (медицинские записи, рецепты и т.д.) - Финансовая информация (банковские счета, транзакции и т.д.) - Биометрические данные (отпечатки пальцев, данные распознавания лиц и т.д.) - Учетные данные (имена пользователей, пароли, секретные вопросы)	Удалить Шифровать конфиденциальные данные перед обработкой
Конфиденциальная бизнес-информация	- Коммерческая тайна - Собственные алгоритмы или код - Внутренняя коммуникация - Клиентские списки - Неопубликованные финансовые данные	Удалить При необходимости использовать методы суммаризации
Юридическая ограниченная информация	- Данные, защищённые GDRP и HIPAA - Материалы, защищённые авторским правом без надлежащего разрешения - Информация под NDA	Удалить Обеспечить соответствие юридическим требованиям
Неподобающий или оскорбительный контент	- Речи ненависти - Эксплицитный контент - Домогательства или злоупотребления	Отфильтровать Использовать инструменты модерации контента
Ненадёжные или низкокачественные данные	- Спам или реклама - Нерелевантный контент - Дубликаты - Неправильно отформатированные или повреждённые	Очистить данные Удалить дубликаты и верифицировать данные
Специальные категории персональных данных (по GDPR)	- Расовое или этнические происхождение - Политические взгляды - Религиозные или философские убеждения - Членство в профсоюзе - Генетические данные	Удалить или анонимизировать Использовать псевдонимы или редактирование

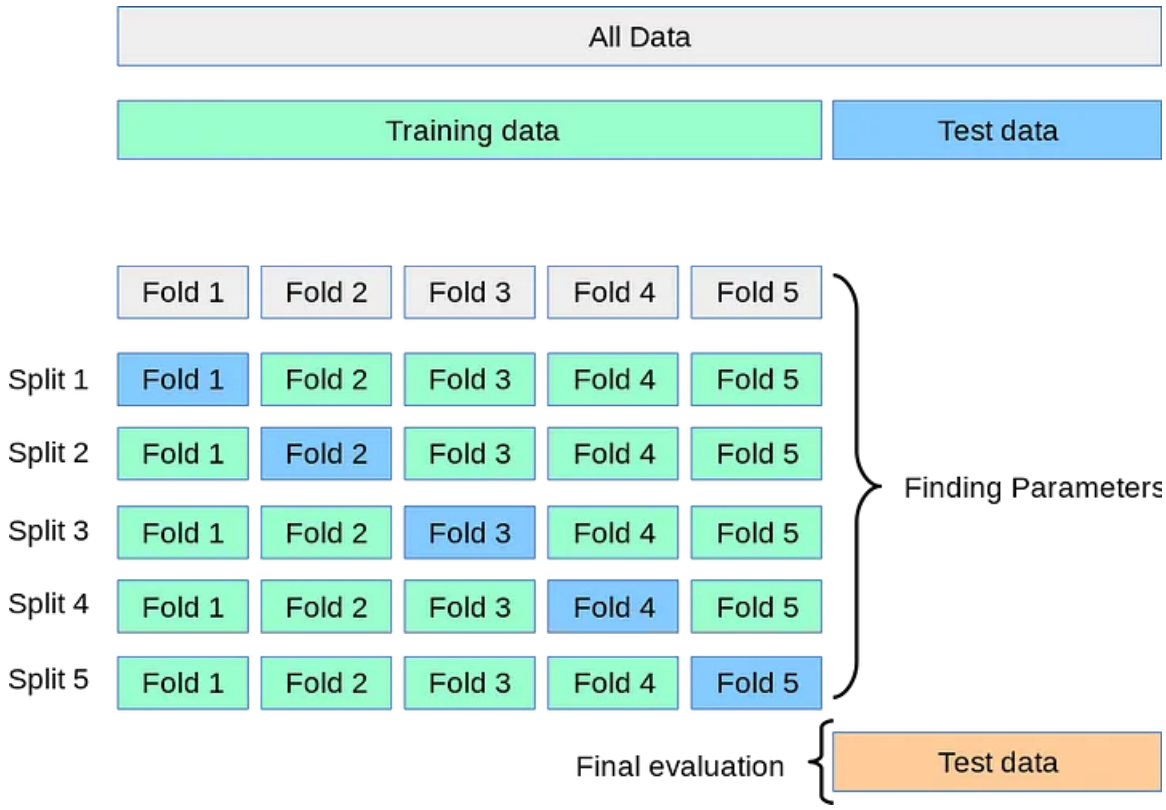
Применяется контроль качестве, который включает проверку политик и правил. Это необходимо для гарантии того, что только код и артефакты, соответствующие установленным требованиям, могут переходить к последующим стадиям пайплайна. Широко известно, что данные, используемые в обучении моделей машинного обучения, являются критически важными. Поэтому обеспечение отсутствия в них конфиденциальной или персонально идентифицируемой информации (PII) имеет первостепенное значение. Таким образом, процесс контроля качества повторяет принципы DevSecOps и дополняется проверкой соответствия обучающих данных перед их использованием в процессе моделирования. Прохождение через контроль качества означает соответствие как требованиям безопасности, так и стандартам соответствия.

Quality Gate - PASS = Security AND Compliance

Проводится обучение модели с использованием технических методов: ранней остановки (early stopping), что помогает избежать переобучения;кросс-валидации с применением KFold для более точной оценки производительности на различных подмножествах данных. Применение таких методов существенно важно для обеспечения устойчивости и высокой точности модели. Эти техники являются частью множества стратегий обучения, направленных на достижение оптимальных результатов в машинном обучении и повышение надежности моделей.



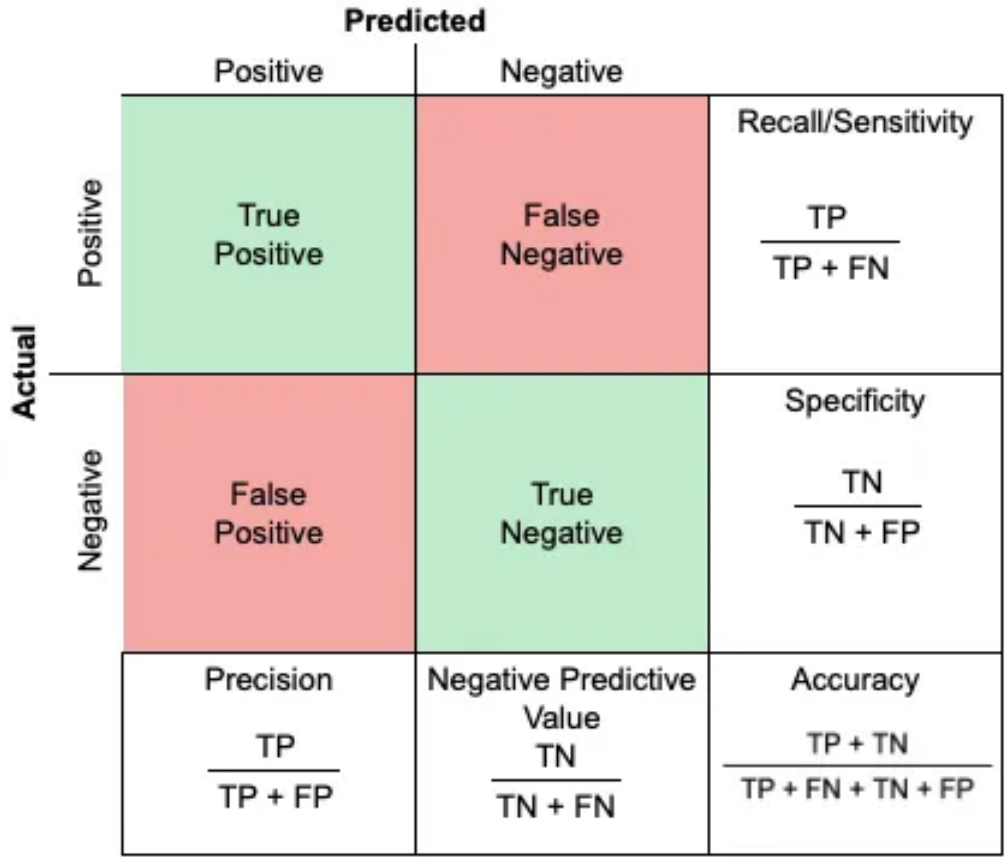
Регуляризация путем ранней остановки



Крос-валидация(KFold); Оценка производительности;

Из документации scikit-learn 1.5.1

Для оценки моделей будут применяться различные метрики и методы подтверждения соответствия необходимым критериям производительности. Анализируя полученные результаты, определятся соответствующие пороговые значения для вновь обученной модели. Этот процесс позволяет объективно измерять эффективность модели и гарантировать достижение требуемого уровня качества и надежности. Использование разнообразных метрик обеспечивает всестороннюю оценку модели, что способствует её дальнейшему совершенствованию и адаптации к специфическим требованиям проекта.



Показатели оценки модели ML

Проводится комплексное финальное тестирование модели, включающее как стандартные, так и технические проверки. Особое внимание уделяется наиболее распространённым угрозам: инъекциям запросов (prompt injection), отравлению данных (data poisoning) и утечкам конфиденциальной информации. Для выявления этих и других потенциальных рисков применяются специализированные инструменты, такие как modelscan, Vigil и Garak. Проведение всесторонних тестов безопасности необходимо для обнаружения и ликвидации уязвимостей, что обеспечивает соответствие модели установленным стандартам безопасности и защищённости. Это гарантирует не только её функциональность и эффективность, но также безопасное использование в реальных условиях с минимизацией рисков безопасности.

Контроль качества направлен на тщательную проверку конечного продукта согласно принятым стандартам и требованиям перед его выпуском в эксплуатацию. Особое внимание уделяется соблюдению политик и правил, чтобы обеспечить полное соответствие всех необходимых критериев качества. Этот контроль является ключевым этапом в процессах MLSecOps, гарантируя, что только те модели, успешно прошедшие предыдущие проверки и тестирования, допускаются к использованию в производственной среде. Финальный контроль качества служит неотъемлемой гарантией того, что модель обладает надлежащим уровнем надежности, безопасности и эффективности, соответствующей высоким стандартам для успешной реализации проекта.

Quality Gate - PASS = Security

Процесс подписания обученной модели – важный шаг для обеспечения её целостности и подтверждения. Подписание с помощью инструментов, таких как OpenPubKey, гарантирует неизменность модели после создания и подтверждает разработку доверенным источником. Эта процедура обеспечивает надёжную защиту от несанкционированных изменений и подделок, что является критичным для сохранения доверия к использованию модели на продуктиве в любых средах.

Для сохранения подписанной модели в надежных системах хранения артефактов используются такие решения, как AWS S3 бакеты, репозитории Nexus или FTP-серверы. Сохранение подписанных моделей в этих хранилищах обеспечивает их доступность для последующего использования и защищает от потери данных. Важно, чтобы к этим хранилищам был строго контролируемый доступ, что позволяет поддерживать высокий уровень безопасности и защищённости моделей на протяжении всего их жизненного цикла.

Управление ключами (Key Management) также играет роль в защите чувствительных данных на всех этапах пайплайна MLSecOps. Для этого используются такие инструменты как HashiCorp Vault, AWS Secret Manager и Amazon Web Services Key Management Service (AWS KMS). Эти инструменты обеспечивают безопасное хранение и управление секретными, публичными и приватными ключами, необходимыми для различных операций: шифрования данных и подписи моделей. Эффективное управление ключами предотвращает несанкционированный доступ к данным и обеспечивает высокий уровень безопасности всей инфраструктуры.

Наконец, мониторинг является неотъемлемой частью пайплайна, обеспечивая постоянный контроль и управление уязвимостями на всех этапах разработки и развертывания моделей. Для этого используются различные инструменты: Telegram и Slack для уведомлений; Defect Dojo – для управления уязвимостями; ELK Stack (Elasticsearch, Logstash, Kibana) - для сбора и анализа логов; PagerDuty – для управления инцидентами; а также Prometheus и Grafana – для мониторинга метрик и визуализации данных. Эти инструменты позволяют эффективно отслеживать состояние пайплайна, быстро реагировать на возникающие проблемы и поддерживать высокий уровень безопасности и надёжности моделей.

Для повышения безопасности больших языковых моделей (LLM) с использованием конвейера MLSecOps данный структурированный подход гарантирует безопасность каждого этапа жизненного цикла модели машинного обучения: от разработки и тестирования до обучения и мониторинга. Благодаря интеграции передовых методов обеспечения безопасности на каждом этапе, потенциальные уязвимости выявляются и устраняются на ранней стадии, что обеспечивает работу LLM в безопасной и отказоустойчивой среде. Этот подход защищает не только сами модели, но и данные, а также инфраструктуру, на которой они базируются, обеспечивая комплексную защиту передовых приложений машинного обучения.

Заключение

В данной работе были рассмотрены существующие подходы, принципы и инструменты MLSecOps, включающие инструменты контроля качества, механизмы управления конфиденциальной информацией и безопасностью, а также системы мониторинга и управления инцидентами. Основной вывод работы заключается в том, что внедрение подхода MLSecOps на каждом этапе жизненного цикла модели позволяет минимизировать уязвимости, снизить риски утечек данных и атак, а также обеспечить стабильную и безопасную эксплуатацию моделей в реальных условиях.

Таким образом, описанные принципы, подходы, а также инструментарий MLSecOps можно рассматривать как комплексное решение, способное повысить доверие к системам машинного обучения и обеспечить их надежность и отказоустойчивость в современных приложениях.

Список литературы

[1] Н. Гифт, К. Берман, А. Деза, Г. Георгу Python и DevOps: Ключ к автоматизации Linux. Перевел с английского И. Пальти. ООО "ПИТЕР". Россия, Санкт-Петербург 2022

[2] Ю.В. Литвинов Курс лекций "Архитектура и проектирование программного обеспечения". СПбГУ
https://github.com/yurii-litvinov/courses/tree/master/software-design-extended

[3] Github Repository: A curated list of awesome open-source tools, resources, and tutorials for MLSecOps.
https://github.com/noobpk/MLSecOps-DevSecOps-Awesome

[4] ML System Design: основные способы деплоя и тестирования моделей машинного обучения в продакшене.
https://habr.com/ru/articles/739316/

[5] Архитектура реальной системы машинного обучения. https://habr.com/ru/companies/vk/articles/673782/

[6] Нишант Шакла. Машинное обучение и TensorFlow. ООО "ПИТЕР". Россия, Санкт-Петербург 2019

[7] Луис Серрано. Грокаем машинное обучение. ООО "ПИТЕР". Россия, Санкт-Петербург 2024

[8] Джульен Вехен. Безопасный DevOps. Эффективная эксплуатация систем. ООО "ПИТЕР". Россия, Санкт-Петербург 2020

[9] Эрик Чоу. Python для сетевых инженеров. Автоматизация сети, программирование и DevOps. ООО "ПИТЕР". Россия, Санкт-Петербург 2024.