АО «АЛМАТИНСКИЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет инженерных и информационных технологий

Кафедра компьютерной инженерии

ПРОЕКТИРОВАНИЕ СИСТЕМ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

ЛЕКЦИЯ № 4

Данные в AI-системах: качество, подготовка и управление

Теоретико-практический конспект

Составитель: Алтыбай А.

Кафедра компьютерной инженерии

Алматы, 2026

Аннотация и назначение лекции

Лекция 4 раскрывает тему «Данные в AI-системах: качество, подготовка и управление» в контексте разработки промышленной AI-системы. Материал опирается на тематический план дисциплины и соединяет определения, принципы проектирования, математические оценки, проектные артефакты, практический код и аргументированный анализ рисков. Вопросы построены вокруг решений, которые предстоит принимать инженеру, а не только вокруг названий алгоритмов. [1, 3]

План занятия

1. Источники и происхождение данных
2. Типы дефектов качества
3. Исследовательский анализ
4. Очистка и потери информации
5. Пропуски и недостоверные значения
6. Data leakage
7. Разбиение и группировка
8. Разметка и согласованность экспертов
9. Практические сценарии, контроль качества и проектные задания.

Ожидаемые результаты обучения

По завершении студент способен:

1. объяснить и аргументированно применить принцип «Источники и происхождение данных» к инженерному решению;
2. объяснить и аргументированно применить принцип «Типы дефектов качества» к инженерному решению;

3. объяснить и аргументированно применить принцип «Исследовательский анализ» к инженерному решению;
4. объяснить и аргументированно применить принцип «Очистка и потери информации» к инженерному решению;
5. объяснить и аргументированно применить принцип «Пропуски и недостоверные значения» к инженерному решению;
6. объяснить и аргументированно применить принцип «Data leakage» к инженерному решению;
7. реализовать упрощённый проверяемый прототип и сформулировать критерии его приемки.
8. указать угрозы достоверности результатов и предложить проверки до эксплуатации.

Сквозной кейс. В течение лекции рассматриваем регистратор студенческих обращений. Существенные условия: ошибочную разметку, устаревшую схему полей. Основная проверяемая характеристика: валидность и полноту данных. Числа в расчётных примерах учебные и не являются результатами реальных измерений.

1. Базовые теоретические положения из конспекта курса

Ниже представлены исходные положения учебно-методического конспекта, связанные с данной темой. Следующие разделы раскрывают их с инженерными деталями и проверяемыми примерами. [1, 3] Цель лекции. сформировать понимание данных как управляемого производственного актива и показать требования к качеству, разметке, версиям, governance и воспроизводимому preprocessing.

Ключевые понятия. data pipeline, data quality, schema, label, leakage, sampling bias, lineage, data governance, data versioning, train/validation/test split.

1.1 Данные как часть поведения системы

В традиционном программном обеспечении поведение в основном задается кодом. В ML значительная часть поведения определяется данными. Изменение источника, распределения, способа разметки или пропусков может изменить модель даже при неизменном коде.

Поэтому данные должны иметь владельца, схему, проверки качества, историю происхождения и версию. Фраза «мы обучили модель на таблице data.csv» недостаточна для воспроизводимости.

1.2 Измерения качества данных

Типовые измерения включают полноту, корректность, уникальность, своевременность, согласованность и соответствие схеме. Для ML дополнительно важны покрытие редких случаев, качество labels, соответствие production-популяции и отсутствие leakage.

Проверки качества следует автоматизировать. Например, pipeline может блокировать новую партию, если доля пропусков выросла в несколько раз, появилась неизвестная категория или нарушен диапазон критического признака.

1.3 Data leakage и смещения выборки

Data leakage возникает, когда модель получает информацию, недоступную в момент реального прогноза. Например, признак «дата закрытия заявки» нельзя использовать для предсказания результата в момент создания заявки. Leakage часто дает искусственно высокие offline-метрики.

Sampling bias возникает, когда обучающая выборка не отражает population, на которой будет работать система. Это приводит к систематическим ошибкам даже при хорошем качестве на тесте, собранном тем же способом.

1.4 Разделение данных и временные зависимости

Стандартное случайное train/validation/test split подходит только при условии относительной независимости объектов. Для временных задач необходимо делить по времени, иначе возникает утечка будущей информации. Для данных по пользователям или устройствам иногда требуется group split, чтобы один и тот же объект не попадал одновременно в train и test.

Тестовая выборка должна оставаться максимально независимой от решений, принимаемых в процессе экспериментов.

1.5 Data governance и lineage

Data governance определяет правила доступа, хранения, качества, ответственности и соответствия требованиям. Data lineage отвечает на вопросы: откуда пришло поле, какие преобразования были применены, какой dataset использовался для конкретной версии модели.

Для чувствительных данных дополнительно важны минимизация собираемых полей, контроль доступа, срок хранения, маскирование и аудит использования.

1.6 Выводы

▷ Качество данных — такая же инженерная характеристика, как качество кода.

▷ Leakage и смещение выборки могут полностью обесценить offline-метрики.

▷ Разделение данных должно учитывать временную и групповую структуру.

▷ Версионирование и lineage необходимы для воспроизводимости модели.

1.7 Контрольные вопросы

- Какие измерения качества данных важны для ML?
- Приведите пример data leakage.
- Когда случайное разделение train/test некорректно?
- Что такое data lineage?
- Почему data governance важен для промышленного AI?

2. Систематическое изложение темы и инженерные решения

2.1 Источники и происхождение данных

Данные поступают через API, выгрузки, события и ручную разметку; для каждого источника нужна договоренность о схеме, периодичности и ответственности. Provenance важен для отладки ошибок. [2]

Проверяемый пример. Зафиксируйте происхождение 1000 клиентских обращений и версию правил их разметки.

Вопрос для обсуждения. Как проверить выполнение такого требования: Зафиксируйте происхождение 1000 клиентских обращений и версию правил их разметки.

2.2 Типы дефектов качества

Completeness, validity, consistency, uniqueness, timeliness и accuracy отвечают за разные проблемы. Числовая корректность не доказывает семантическую достоверность. [3]

Проверяемый пример. Найдите в CSV пропуски, неверные даты, повторы идентификаторов и конфликтующие категории.

Вопрос для обсуждения. Как проверить выполнение такого требования: Найдите в CSV пропуски, неверные даты, повторы идентификаторов и конфликтующие категории.

2.3 Исследовательский анализ

EDA изучает распределения, дисбаланс меток, зависимости и временные изменения. Графики должны отвечать на конкретный вопрос об обучении модели. [4]

Проверяемый пример. Постройте распределение длин сообщений для проверки необходимости truncation.

Вопрос для обсуждения. Как проверить выполнение такого требования: Постройте распределение длин сообщений для проверки необходимости truncation.

2.4 Очистка и потери информации

Удаление выбросов и дубликатов может уменьшить шум, но также исключить важный редкий класс. Каждое правило очистки оценивают на репрезентативность. [5]

Проверяемый пример. Сравните долю срочных обращений до и после фильтрации очень длинных текстов.

Вопрос для обсуждения. Как проверить выполнение такого требования: Сравните долю срочных обращений до и после фильтрации очень длинных текстов.

2.5 Пропуски и недостоверные значения

Пропуск может иметь причинную связь с целевой переменной, поэтому простая иммутация не всегда безвредна. Статистики иммутации обучаются только на train. [6]

Проверяемый пример. Для табличного набора рассчитайте median на train и примените его к validation без пересчета.

Вопрос для обсуждения. Как проверить выполнение такого требования: Для табличного набора рассчитайте median на train и примените его к validation без пересчета.

2.6 Data leakage

Утечка возникает при использовании будущей информации, целевого признака либо общих объектов между train и test. Random split ненадёжен для временных событий. [1]

Проверяемый пример. Проверьте, не содержит ли признак «время закрытия обращения» информацию из будущего.

Вопрос для обсуждения. Как проверить выполнение такого требования: Проверьте, не содержит ли признак «время закрытия обращения» информацию из будущего.

2.7 Разбиение и группировка

Тестовые данные должны моделировать ожидаемые в эксплуатации объекты. При зависимостях применяют group split, при временной структуре — chronological split. [2]

Проверяемый пример. Не допускайте, чтобы сообщения одного клиента оказались одновременно в train и test при проверке обобщения.

Вопрос для обсуждения. Как проверить выполнение такого требования: Не допускайте, чтобы сообщения одного клиента оказались одновременно в train и test при проверке обобщения.

2.8 Разметка и согласованность экспертов

Labeling guidelines с примерами трудных случаев снижают неоднозначность. Разногласия экспертов следует оценивать, а не скрывать голосованием. [3]

Проверяемый пример. Создайте инструкцию для классов «срочный», «обычный», «спам» и спорных случаев.

Вопрос для обсуждения. Как проверить выполнение такого требования: Создайте инструкцию для классов «срочный», «обычный», «спам» и спорных случаев.

2.9 Схемы и контракты

Автоматические проверки данных ловят типы, диапазоны, уникальность и допустимые категории до запуска обучения. Validation gate должен останавливать проблемную сборку. [4]

Проверяемый пример. Добавьте проверку, что доля пустых текстов не превышает порог, а timestamp не в будущем.

Вопрос для обсуждения. Как проверить выполнение такого требования: Добавьте проверку, что доля пустых текстов не превышает порог, а timestamp не в будущем.

2.10 Версии и воспроизводимость

Data version включает сами данные или контрольные суммы, код преобразования и условия выборки. Сохранение только имени CSV недостаточно. [5]

Проверяемый пример. Для эксперимента запишите hash снапшота, commit preprocessing и дату получения меток.

Вопрос для обсуждения. Как проверить выполнение такого требования:

Для эксперимента запишите hash снапшота, commit preprocessing и дату получения меток.

2.11 Governance и доступ

Данные содержат требования хранения, минимизации, цели использования и удаления. Контроль доступа и аудит действий важны на всех стадиях. [6]

> **Проверяемый пример.** Разделите роли сотрудника с доступом к обезличенным признакам и администратора исходных персональных данных.

Вопрос для обсуждения. Как проверить выполнение такого требования: Разделите роли сотрудника с доступом к обезличенным признакам и администратора исходных персональных данных.

2.12 Данные и эксплуатационный цикл

Продакшен-пайплайн обязан обнаруживать дрейф схемы, смену источников и необычные пропорции категорий. Повторная подготовка признаков должна быть детерминированной. [1]

> **Проверяемый пример.** Сконструируйте alert при внезапном росте пропусков с 2% до 30%.

Вопрос для обсуждения. Как проверить выполнение такого требования: Сконструируйте alert при внезапном росте пропусков с 2% до 30%.

3. Математические инструменты и количественные критерии

Для перехода от неформального решения к проверяемой постановке введём следующие величины.

$$\text{MissingRate}(j) = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}[x_{ij} \text{ отсутствует}] \tag{1}$$

$$\hat{p}_c = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}[y_i = c] \tag{2}$$

Изменения доли пропусков и классов надо проверять отдельно для train, validation и production. Средние значения не заменяют проверку отдельных источников.

При выборе числового критерия нужно отличать обучающую функцию потерь, продуктовую ценность и эксплуатационные ограничения. Первую можно оптимизировать дифференцированием, вторая определяется процессом принятия решения, а третьи проверяются нагрузочными испытаниями, политиками безопасности и мониторингом.

3.1 Пошаговый расчёт на учебных данных

Рассмотрим две конфигурации одной системы. Вариант A имеет условную оценку качества $q_A = 0.86$ и задержку $t_A = 90$ мс; вариант B — $q_B = 0.89$ и $t_B = 280$ мс. Предположим, что требование заказчика задаёт $q \geq 0.85$ и $p95 \leq 150$ мс. Тогда A удовлетворяет обоим условиям, тогда как B нарушает второе. Следовательно, выбор B только потому, что его offline-метрика выше, не является обоснованным. Числа иллюстративны; реальный вывод требует измерений при одинаковой нагрузке и одинаковом наборе данных.

$$\text{accept}(A) = \mathbf{1}\{0.86 \geq 0.85 \wedge 90 \leq 150\} = 1,$$
$$\text{accept}(B) = \mathbf{1}\{0.89 \geq 0.85 \wedge 280 \leq 150\} = 0.$$

3.2 Как интерпретировать число, а не только вычислить его

Сначала следует проверить единицы измерения, состав выборки и смысл агрегирования. Например, p95 на спокойном стенде нельзя переносить на рабочую нагрузку с иным числом одновременных запросов. Затем исследуют чувствительность к изменениям входных данных и неопределённость оценок. Наконец, фиксируют последствия порога: какую ошибку он предотвращает, а какую может увеличить. Этот подход делает количественный критерий частью инженерной аргументации, а не украшением отчёта. [1, 3]

4. Практическое программирование на Python

Пример предназначен для демонстрации ключевого инженерного принципа. Он не претендует на обработку реального production-трафика. Запускать код следует в изолированном окружении и фиксировать версии библиотек, если они используются. Вывод зависит от указанных входных данных; для стохастических алгоритмов он может меняться.

4.1 Основной воспроизводимый пример

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression

frame = pd.DataFrame({"duration": [1., 2., None, 4., 5., 6.],
                      "amount": [5, 9, 8, 14, 20, 19],
                      "target": [0, 0, 0, 1, 1, 1]})
X, y = frame[["duration", "amount"]], frame["target"]
X_tr, X_te, y_tr, y_te = train_test_split(
    X, y, random_state=42, stratify=y)
model = Pipeline([("imputer", SimpleImputer(strategy="median")),
                  ("scaler", StandardScaler()),
                  ("clf", LogisticRegression())])
model.fit(X_tr, y_tr)
print(model.predict(X_te))
```

Проверка результатов. Сначала убедитесь, что код завершился без исключений; проверьте формы массивов, схему результата и отсутствие недопустимых значений. Если используются внешние данные, включите в отчёт способ их получения и идентификатор версии. Измените один входной параметр и объясните, почему наблюдаемый результат изменился или остался прежним.

4.2 Компактная проверка контракта или метрики

```python
from hashlib import sha256
raw = b"one,12,positive\ntwo,31,negative\n"
print("snapshot digest:", sha256(raw).hexdigest()[:12])
```

При самостоятельном повторении предложите минимум один негативный тест: некорректное значение, пустой вход, несовместимую версию схемы либо нарушение SLO. Переход от успешного демонстрационного запуска к надежной системе требует именно таких проверок.

5. Разбор трёх инженерных ситуаций

5.1 Ситуация 1: Источники и происхождение данных

Заказчик использует регистратор студенческих обращений. Новое требование заставляет рассмотреть принцип «Источники и происхождение данных». Основная неопределённость состоит в том, как одновременно обеспечить валидность и полноту данных, не допустить устаревшую схему полей и сохранить возможность изменения модели.

Постановка. Данные поступают через API, выгрузки, события и ручную разметку; для каждого источника нужна договоренность о схеме, периодичности и ответственности. Provenance важен для отладки ошибок.

Действие инженера. Зафиксируйте происхождение 1000 клиентских обращений и версию правил их разметки.

Альтернативы. Сравните три подхода: (A) простое правило без ML, (B) модульное ML-решение с отдельной проверкой входных данных и (C) интегрированный сервис со сложной моделью. Для каждого укажите потребность в данных, ожидаемые инженерные риски, возможность тестирования и минимальную инфраструктуру. Таблица ниже задаёт форму отчёта, а не готовый рейтинг.

Вариант	Данные и зависимости	Проверка	Основной риск
Правила	Фиксированные условия	Набор тест-кейсов	Ограниченная гибкость
ML-модуль	Версионированные признаки	Offline и API tests	Drift и skew
Сложный сервис	Несколько компонентов	End-to-end и нагрузка	Каскадный отказ

Проверка решения. Подготовьте две позитивные и две негативные пробы; задайте метрику успеха, базовый сценарий и критерий остановки внедрения. Обсудите, какие допущения останутся непроверенными даже после удачного теста.

5.2 Ситуация 2: Data leakage

Заказчик использует регистратор студенческих обращений. Новое требование заставляет рассмотреть принцип «Data leakage». Основная неопределённость состоит в том, как одновременно обеспечить валидность и полноту данных, не допустить устаревшую схему полей и сохранить возможность изменения модели.

Постановка. Утечка возникает при использовании будущей информации, целевого признака либо общих объектов между train и test. Random split ненадёжен для временных событий.

Действие инженера. Проверьте, не содержит ли признак «время закрытия обращения» информацию из будущего.

Альтернативы. Сравните три подхода: (A) простое правило без ML, (B) модульное ML-решение с отдельной проверкой входных данных и (C) интегрированный сервис со сложной моделью. Для каждого укажите потребность в данных, ожидаемые инженерные риски, возможность тестирования и минимальную инфраструктуру. Таблица ниже задаёт форму отчёта, а не готовый рейтинг.

Вариант	Данные и зависимости	Проверка	Основной риск
Правила	Фиксированные условия	Набор тест-кейсов	Ограниченная гибкость
ML-модуль	Версионированные признаки	Offline и API tests	Drift и skew
Сложный сервис	Несколько компонентов	End-to-end и нагрузка	Каскадный отказ

Проверка решения. Подготовьте две позитивные и две негативные пробы; задайте метрику успеха, базовый сценарий и критерий остановки внедрения. Обсудите, какие допущения останутся непроверенными даже после удачного теста.

5.3 Ситуация 3: Governance и доступ

Заказчик использует регистратор студенческих обращений. Новое требование заставляет рассмотреть принцип «Governance и доступ». Основная неопределённость состоит в том, как одновременно обеспечить валидность и полноту данных, не допустить устаревшую схему полей и сохранить возможность изменения модели.

Постановка. Данные содержат требования хранения, минимизации, цели использования и удаления. Контроль доступа и аудит действий важны на всех стадиях.

Действие инженера. Разделите роли сотрудника с доступом к обезличенным признакам и администратора исходных персональных данных.

Альтернативы. Сравните три подхода: (A) простое правило без ML, (B) модульное ML-решение с отдельной проверкой входных данных и (C) интегрированный сервис со сложной моделью. Для каждого укажите потребность в данных, ожидаемые инженерные риски, возможность тестирования и минимальную инфраструктуру. Таблица ниже задаёт форму отчёта, а не готовый рейтинг.

Вариант	Данные и зависимости	Проверка	Основной риск
Правила	Фиксированные условия	Набор тест-кейсов	Ограниченная гибкость
ML-модуль	Версионированные признаки	Offline и API tests	Drift и skew
Сложный сервис	Несколько компонентов	End-to-end и нагрузка	Каскадный отказ

Проверка решения. Подготовьте две позитивные и две негативные пробы; задайте метрику успеха, базовый сценарий и критерий остановки внедрения. Обсудите, какие допущения останутся непроверенными даже после удачного теста.

6. Самостоятельная работа: индивидуальные проектные задания

Для каждого задания студент создаёт отдельный небольшой артефакт: схему, набор тестов, код либо таблицу сравнения, прикладывает наблюдаемый результат и объясняет ограничения.

1. **Источники и происхождение данных.** Зафиксируйте происхождение 1000 клиентских обращений и версию правил их разметки. Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.

2. **Типы дефектов качества.** Найдите в CSV пропуски, неверные даты, повторы идентификаторов и конфликтующие категории. Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.

3. **Исследовательский анализ.** Постройте распределение длин сообщений для проверки необходимости truncation. Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.

4. **Очистка и потери информации.** Сравните долю срочных обращений до и после фильтрации очень длинных текстов. Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.

5. **Пропуски и недостоверные значения.** Для табличного набора рассчитайте median на train и примените его к validation без пересчета. Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.

6. **Data leakage.** Проверьте, не содержит ли признак «время закрытия обращения» информацию из будущего. Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.

7. **Разбиение и группировка.** Не допускайте, чтобы сообщения одного клиента оказались одновременно в train и test при проверке обобщения.

Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.

8. **Разметка и согласованность экспертов.** Создайте инструкцию для классов «срочный», «обычный», «спам» и спорных случаев. Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.

9. **Схемы и контракты.** Добавьте проверку, что доля пустых текстов не превышает порог, а timestamp не в будущем. Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.

10. **Версии и воспроизводимость.** Для эксперимента запишите hash снапшота, commit preprocessing и дату получения меток. Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.

11. **Governance и доступ.** Разделите роли сотрудника с доступом к обезличенным признакам и администратора исходных персональных данных. Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.

12. **Данные и эксплуатационный цикл.** Сконструируйте alert при внезапном росте пропусков с 2% до 30%. Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.

7. Вопросы для текущего и итогового контроля

1. В чём назначение и ограничения принципа «Источники и происхождение данных»?

2. Как проверить на практике утверждение: «Зафиксируйте происхождение 1000 клиентских обращений и версию правил их разметки.»?

3. В чём назначение и ограничения принципа «Типы дефектов качества»?

4. Как проверить на практике утверждение: «Найдите в CSV пропуски, неверные даты, повторы идентификаторов и конфликтующие категории.»?
5. В чём назначение и ограничения принципа «Исследовательский анализ»?
6. Как проверить на практике утверждение: «Постройте распределение длин сообщений для проверки необходимости truncation.»?
7. В чём назначение и ограничения принципа «Очистка и потери информации»?
8. Как проверить на практике утверждение: «Сравните долю срочных обращений до и после фильтрации очень длинных текстов.»?
9. В чём назначение и ограничения принципа «Пропуски и недостоверные значения»?
10. Как проверить на практике утверждение: «Для табличного набора рассчитайте median на train и примените его к validation без пересчета.»?
11. В чём назначение и ограничения принципа «Data leakage»?
12. Как проверить на практике утверждение: «Проверьте, не содержит ли признак «время закрытия обращения» информацию из будущего.»?
13. В чём назначение и ограничения принципа «Разбиение и группировка»?
14. Как проверить на практике утверждение: «Не допускайте, чтобы сообщения одного клиента оказались одновременно в train и test при проверке обобщения.»?
15. В чём назначение и ограничения принципа «Разметка и согласованность экспертов»?
16. Как проверить на практике утверждение: «Создайте инструкцию для классов «срочный», «обычный», «спам» и спорных случаев.»?
17. В чём назначение и ограничения принципа «Схемы и контракты»?
18. Как проверить на практике утверждение: «Добавьте проверку, что доля пустых текстов не превышает порог, а timestamp не в будущем.»?

19. В чём назначение и ограничения принципа «Версии и воспроизводимость»?
20. Как проверить на практике утверждение: «Для эксперимента запишите hash снапшота, commit preprocessing и дату получения меток.»?
21. В чём назначение и ограничения принципа «Governance и доступ»?
22. Как проверить на практике утверждение: «Разделите роли сотрудника с доступом к обезличенным признакам и администратора исходных персональных данных.»?
23. В чём назначение и ограничения принципа «Данные и эксплуатационный цикл»?
24. Как проверить на практике утверждение: «Сконструируйте alert при внезапном росте пропусков с 2% до 30%.»?
25. Почему высокое offline-качество не гарантирует надежность промышленной AI-системы?
26. Какие артефакты нужны для воспроизводимости и анализа инцидента?

8. Глоссарий ключевых понятий

Источники и происхождение данных
Данные поступают через API, выгрузки, события и ручную разметку; для каждого источника нужна договоренность о схеме, периодичности и ответственности.

Типы дефектов качества
Completeness, validity, consistency, uniqueness, timeliness и accuracy отвечают за разные проблемы.

Исследовательский анализ
EDA изучает распределения, дисбаланс меток, зависимости и временные изменения.

Очистка и потери информации
Удаление выбросов и дубликатов может уменьшить шум, но также исключить важный редкий класс.

Пропуски и недостоверные значения
Пропуск может иметь причинную связь с целевой переменной, поэтому простая иммутация не всегда безвредна.

Data leakage
Утечка возникает при использовании будущей информации, целевого признака либо общих объектов между train и test.

Разбиение и группировка
Тестовые данные должны моделировать ожидаемые в эксплуатации объекты.

Разметка и согласованность экспертов
Labeling guidelines с примерами трудных случаев снижают неоднозначность.

Схемы и контракты
Автоматические проверки данных ловят типы, диапазоны, уникальность и допустимые категории до запуска обучения.

Версии и воспроизводимость
Data version включает сами данные или контрольные суммы, код преобра-

зования и условия выборки.

Governance и доступ
Данные содержат требования хранения, минимизации, цели использования и удаления.

Данные и эксплуатационный цикл
Продакшен-пайплайн обязан обнаруживать дрейф схемы, смену источников и необычные пропорции категорий.

9. Итоги и связь со следующей темой

Представленные методы имеют смысл как части целостного управляемого жизненного цикла. Ключевой результат инженерного проектирования состоит не в максимальном числе компонентов или сложности модели, а в убедительной связи требований, проектных решений, измерений и механизмов сопровождения. Дальнейшее изучение курса развивает эти связи на следующем этапе жизненного цикла.

Список литературы

[1] Huyen C. Designing Machine Learning Systems. O’Reilly Media, 2022. https://www.oreilly.com/library/view/designing-machine-learning/9781098107956/.

[2] Kleppmann M. Designing Data-Intensive Applications. O’Reilly Media, 2017. https://dataintensive.net/.

[3] Géron A. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow. 3rd ed., O’Reilly Media, 2022. https://www.oreilly.com/library/view/hands-on-machine-learning/9781098125967/.

[4] Scikit-learn Developers. User Guide: Model Selection, Pipelines and Metrics. https://scikit-learn.org/stable/user_guide.html.

[5] NIST. AI Risk Management Framework (AI RMF 1.0). NIST AI 100-1, 2023. https://doi.org/10.6028/NIST.AI.100-1.

[6] Google Developers. Rules of Machine Learning: Best Practices for ML Engineering. https://developers.google.com/machine-learning/guides/rules-of-ml.