ЛЕКЦИЯ 4
Подготовка и обработка данных в системах искусственного интеллекта

Цель лекции

Цель лекции — сформировать у магистрантов системное понимание подготовки и обработки данных как обязательного этапа разработки систем искусственного интеллекта, рассмотреть методы очистки, преобразования, кодирования и масштабирования данных, а также показать на практических примерах, каким образом качество исходных данных влияет на обучение и оценку моделей.

Теоретические материалы:

Современная система искусственного интеллекта обучается не на абстрактных правилах, а на данных, в которых содержится информация об объектах и процессах предметной области. Алгоритм выявляет закономерности именно в тех примерах, которые ему предоставлены. Поэтому качество модели определяется не только выбранным методом машинного обучения, но и качеством данных, способом их представления и корректностью предварительной обработки.

Можно использовать сложную нейронную сеть, но если исходные данные содержат ошибки, дубликаты, пропуски, неверные метки или систематические искажения, модель будет обучаться на этих недостатках. В практике анализа данных часто используют принцип «garbage in — garbage out»: некорректные входные данные приводят к некорректным результатам.

Пример. Разрабатывается система обнаружения сетевых атак. В журнале событий часть записей имеет ошибочную временную метку, некоторые соединения записаны дважды, а для части объектов отсутствует метка normal/attack. Если сразу обучить классификатор, он может принять технические ошибки сбора данных за закономерности сетевой атаки. Поэтому перед обучением необходимо проверить и подготовить набор данных.

Подготовка данных — это совокупность действий, преобразующих исходные данные в форму, пригодную для анализа и обучения модели искусственного интеллекта. Она начинается ещё до запуска алгоритма машинного обучения и может занимать значительную часть работы над проектом.

Типичный процесс можно представить следующей последовательностью:

сбор данных → первичный анализ → очистка → обработка пропусков → обработка выбросов → преобразование категориальных признаков → масштабирование → формирование признаков → разделение выборки → контроль качества подготовленных данных.

Важно, что эта последовательность не всегда строго линейна. После визуализации исследователь может обнаружить ошибку и вернуться к очистке, а после первой модели — понять, что необходимо изменить набор признаков. Подготовка данных является итеративным процессом.

Типы данных в задачах искусственного интеллекта

Перед обработкой необходимо определить, с каким типом данных работает система. От этого зависит выбор методов подготовки.

Структурированные данные обычно представлены таблицей. Каждая строка соответствует объекту, а столбцы — его признакам. Например, в таблице сетевого трафика строка может описывать одно соединение, а столбцы содержать длительность, протокол, число пакетов, объём трафика и метку класса.

Неструктурированные данные не имеют простой табличной организации. К ним относятся тексты, изображения, аудио- и видеозаписи. Перед передачей модели такие объекты также преобразуются в числовое представление.

Полуструктурированные данные занимают промежуточное положение. Например, JSON-документ или журнал событий может содержать поля определённой структуры, но состав полей у разных записей может отличаться.

Пример. Для системы выявления фишинга можно одновременно использовать табличные признаки URL, текст письма и изображение веб-страницы. Для каждого источника потребуется собственная подготовка, после чего признаки могут быть объединены в одной модели.

Объекты, признаки и целевая переменная

В задачах машинного обучения набор данных обычно представляется как совокупность объектов и их признаков. Объект — отдельное наблюдение. Признак — измеряемая или вычисляемая характеристика объекта. Целевая переменная — значение, которое модель должна предсказывать.

Например, при обнаружении сетевых атак объектом является сетевое соединение. Признаками могут быть duration, protocol, packets, bytes, failed_logins. Целевая переменная label принимает значения normal или attack.

```
X = data[["duration", "packets", "bytes", "failed_logins"]]
y = data["label"]
```

В задаче прогнозирования стоимости квартиры признаками могут быть площадь, число комнат, район и год постройки, а целевой переменной — цена. В задаче распознавания спама признаками могут быть характеристики текста письма, а целевой переменной — spam/not spam.

Первичный анализ набора данных

До изменения данных необходимо понять их структуру. Первичный анализ отвечает на вопросы: сколько объектов и признаков имеется, какие типы данных используются, какие значения встречаются, есть ли пропуски, дубликаты, аномальные значения и дисбаланс классов.

```
import pandas as pd

data = pd.read_csv("network_traffic.csv")
print(data.shape)
print(data.head())
print(data.info())
```

Метод shape показывает количество строк и столбцов. head() позволяет визуально проверить первые записи. info() показывает названия столбцов, типы данных и число непустых значений.

Пример. Если столбец packets должен быть числовым, но Pandas определил его как object, необходимо проверить, не содержатся ли в нём значения вида "unknown", пробелы или другие символы. Такая проблема должна быть устранена до обучения модели.

Описательная статистика

Для числовых признаков полезно вычислить описательные статистики: количество значений, среднее, стандартное отклонение, минимум, максимум и квантили. Это позволяет быстро обнаружить подозрительные значения и оценить масштаб признаков.

```
print(data.describe())
```

Пример. Для признака duration большинство соединений длится от 0 до 60 секунд, но максимальное значение равно 999999 секунд. Такое значение требует проверки: это может быть реальное длительное соединение, ошибка измерения или специальное значение, использованное системой сбора данных.

Среднее значение также не всегда достаточно информативно. Если распределение сильно асимметрично, полезно сравнивать среднее с медианой и анализировать квантили.

Проверка и удаление дубликатов

Дубликаты — повторяющиеся записи. Они могут возникать при объединении нескольких файлов, повторной выгрузке журнала или ошибке системы сбора данных. Если один и тот же объект многократно присутствует в обучающей выборке, модель может получить искажённое представление о частоте определённых событий.

```
print(data.duplicated().sum())
data = data.drop_duplicates()
```

Однако удалять все совпадающие строки автоматически нельзя. Иногда одинаковые значения действительно соответствуют разным событиям. Например, два пользователя могут выполнить одинаковые действия в разное время. Поэтому решение об удалении должно учитывать смысл данных.

Пропущенные значения

Пропуск означает отсутствие значения некоторого признака. В Pandas он часто представлен как NaN. Причины пропусков могут быть разными: датчик не передал измерение, пользователь не заполнил поле, запись была повреждена, определённый признак неприменим к объекту.

```
print(data.isnull().sum())
```

Существует несколько стратегий обработки пропусков. Можно удалить строки или столбцы, заполнить числовой признак средним или медианой, категориальный — наиболее частым значением, либо использовать более сложные методы восстановления.

```
data["bytes"] = data["bytes"].fillna(data["bytes"].median())
data["protocol"] =
data["protocol"].fillna(data["protocol"].mode()[0])
```

Пример. В наборе медицинских данных отсутствует возраст у 2% пациентов. Удаление этих строк может быть допустимо при большом объёме выборки, но при небольшом наборе лучше исследовать причину пропуска и рассмотреть заполнение медианой. Если же отсутствует 80% значений некоторого признака, механическое заполнение может создать искусственную информацию.

Пример из кибербезопасности. Значение failed_logins может отсутствовать не из-за ошибки, а потому что для определённого типа соединения этот показатель не регистрируется. В таком случае сам факт отсутствия может нести информацию, и его нельзя трактовать так же, как случайный технический пропуск.

Ошибки и некорректные значения

Очистка данных включает поиск значений, противоречащих смыслу признака. Возраст -5 лет, отрицательное количество пакетов, дата окончания события раньше даты начала или вероятность 1.7 являются логически некорректными значениями.

Проверки можно выполнять с помощью условий.

```
invalid = data[data["packets"] < 0]
print(invalid)
```

Пример. Если признак port содержит значения от 0 до 65535, запись с портом 90000 должна быть проверена. Возможные действия: исправить значение по первичному источнику, пометить его как неизвестное или исключить запись. Простая замена без понимания причины ошибки может исказить данные.

Выбросы и аномальные наблюдения

Выброс — значение, значительно отличающееся от большинства наблюдений. Выброс не обязательно является ошибкой. В задачах информационной безопасности именно редкое необычное событие может быть атакой, которую необходимо обнаружить.

Поэтому выбросы сначала выявляют и анализируют, а не автоматически удаляют. Для числовых данных применяются статистические методы, межквартильный размах, z-оценка и визуализация.

Пример. В обычном сетевом трафике объём передачи составляет 1–20 МБ, а одна запись содержит 4 ГБ. Если это ошибка счётчика, значение следует исправить или исключить. Если это реальная эксфильтрация данных, удаление такого объекта уничтожит важнейший пример атаки.

Визуализация как инструмент контроля качества данных

Графики позволяют увидеть то, что трудно заметить в таблице. Гистограмма показывает распределение признака, boxplot помогает обнаружить экстремальные значения, диаграмма рассеяния — зависимость между признаками, столбчатая диаграмма — распределение категорий и классов.

```
import matplotlib.pyplot as plt

data["bytes"].hist(bins=30)
plt.xlabel("Объём данных")
plt.ylabel("Количество соединений")
plt.show()
```

Пример. Если гистограмма показывает два отчётливо разделённых диапазона значений, это может означать наличие двух типов объектов или ошибку единиц измерения: часть данных записана в байтах, а часть — в килобайтах.

Категориальные признаки

Многие алгоритмы машинного обучения работают с числами, поэтому категориальные значения необходимо преобразовать. Категориальный признак принимает значения из конечного набора категорий: protocol = TCP, UDP, ICMP; city = Almaty, Astana, Shymkent; device_type = mobile, desktop, server.

Один из простых подходов — заменить категории числами. Однако числа могут создать ложное ощущение порядка. Если TCP=0, UDP=1, ICMP=2, модель не должна считать, что ICMP «в два раза больше» UDP.

Для номинальных категорий часто используется one-hot encoding, при котором каждая категория становится отдельным бинарным столбцом.

```
data = pd.get_dummies(data, columns=["protocol"], dtype=int)
```

После преобразования исходный столбец protocol может быть заменён признаками protocol_TCP, protocol_UDP и protocol_ICMP со значениями 0 или 1.

Пример. Для уровня риска low, medium, high порядок имеет смысл. Здесь возможно порядковое кодирование. Для названий операционных систем Windows, Linux, macOS естественного порядка нет, поэтому one-hot encoding обычно логичнее.

Масштабирование признаков

Признаки могут измеряться в совершенно разных диапазонах. Например, failed_logins принимает значения от 0 до 10, packets — от 1 до 5000, bytes — от 100 до нескольких миллионов. Для алгоритмов, использующих расстояния или градиентную оптимизацию, различие масштабов может существенно влиять на результат.

Нормализация Min-Max преобразует значения к заданному диапазону, часто от 0 до 1:

$x_new = (x - x_min) / (x_max - x_min)$

Стандартизация преобразует признак с учётом среднего значения и стандартного отклонения:

$z = (x - \mu) / \sigma$

Пример. Пусть возраст пользователя равен 35, а годовой объём транзакций — 5 000 000. Без масштабирования алгоритм k-ближайших соседей будет значительно сильнее

реагировать на разницу в миллионах, чем на разницу в возрасте, даже если возраст важен для задачи.

При этом масштабирование не требуется для всех моделей одинаково. Например, деревья решений обычно менее чувствительны к масштабу признаков, чем kNN или многие методы на основе градиентной оптимизации.

Преобразование распределений

Некоторые признаки имеют сильно асимметричное распределение. Например, большинство пользователей выполняет десятки операций в день, но небольшая группа — десятки тысяч. В таких случаях может применяться логарифмическое преобразование, уменьшающее влияние очень больших значений.

```
import numpy as np
data["bytes_log"] = np.log1p(data["bytes"])
```

Функция log1p вычисляет log(1+x), поэтому корректно работает и для нулевых значений. Такое преобразование не следует применять автоматически: необходимо учитывать смысл признака и требования выбранной модели.

Создание новых признаков — Feature Engineering

Подготовка данных включает не только исправление исходных значений, но и создание информативных признаков. Feature Engineering — процесс формирования новых характеристик на основе исходных данных и знаний предметной области.

Пример 1. Для сетевого соединения можно создать средний размер пакета: bytes_per_packet = bytes / packets. Этот показатель иногда информативнее, чем два исходных признака по отдельности.

```
data["bytes_per_packet"] = data["bytes"] / data["packets"].replace(0,
1)
```

Пример 2. Из временной метки можно получить час суток, день недели или признак «рабочее/нерабочее время». Для системы обнаружения мошенничества транзакция в 03:00 может иметь другой контекст, чем такая же операция в рабочее время.

Пример 3. Для юридического текста можно вычислить длину документа, число упоминаний определённых терминов или использовать векторное представление текста. Для изображения признаками могут быть значения пикселей или признаки, автоматически извлечённые нейронной сетью.

Отбор признаков

Большое количество признаков не всегда улучшает модель. Некоторые столбцы могут быть неинформативными, дублировать друг друга или содержать сведения, недоступные в момент реального прогнозирования. Отбор признаков уменьшает избыточность и может повысить интерпретируемость модели.

Пример. В наборе данных есть user_id. Уникальный идентификатор полезен для связи записей в базе, но часто не является содержательным признаком поведения пользователя. Если оставить его без обоснования, модель может запоминать конкретные идентификаторы вместо выявления закономерностей.

Другой пример — два признака bytes и kilobytes, содержащие одну и ту же информацию в разных единицах. Их одновременное использование создаёт избыточность.

Утечка данных

Одной из наиболее опасных ошибок является data leakage — ситуация, когда при обучении модель получает информацию, которая в реальной эксплуатации недоступна в момент прогнозирования. Это приводит к искусственно высокой оценке качества.

Пример. Строится модель прогнозирования, будет ли студент отчислен. Если среди признаков оставить поле «приказ об отчислении», модель почти идеально решит задачу, но такой прогноз не имеет смысла: приказ появляется уже после наступления события.

Пример из обнаружения атак. Если в признаки включить поле, сформированное системой реагирования после подтверждения инцидента, модель фактически получает готовый ответ.

Утечка возможна и при предобработке. Например, если стандартизацию выполнить по всему набору данных до разделения на train и test, информация о распределении тестовой выборки косвенно попадёт в обучение. Правильнее сначала разделить данные, затем вычислять параметры преобразования только по обучающей части и применять их к тестовой.

Разделение данных на обучающую, валидационную и тестовую выборки

Для объективной оценки модели данные разделяются. Обучающая выборка используется для настройки параметров модели. Валидационная — для выбора модели и гиперпараметров. Тестовая — для итоговой независимой оценки.

В простом учебном эксперименте часто используются две части: train и test. Например, 80% объектов — для обучения, 20% — для тестирования. Конкретное соотношение зависит от объёма данных и задачи.

Для классификации важно сохранять пропорции классов, особенно при дисбалансе. В scikit-learn для этого применяется параметр stratify.

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)
```

Параметр random_state позволяет воспроизводить одинаковое разбиение при повторном запуске эксперимента.

Дисбаланс классов

В реальных задачах классы часто представлены неравномерно. ***Например,*** 99% банковских операций нормальные и только 1% мошеннические; большинство сетевых событий безопасны и лишь малая часть относится к атакам.

Если модель всегда отвечает «нормально», при распределении 99/1 она получит accuracy около 99%, хотя не обнаружит ни одной атаки. Поэтому необходимо анализировать распределение классов и использовать подходящие метрики: precision, recall, F1-score, ROC-AUC или PR-AUC в зависимости от задачи.

```
print(data["label"].value_counts())
print(data["label"].value_counts(normalize=True))
```

Для борьбы с дисбалансом могут применяться веса классов, изменение стратегии выборки, oversampling или undersampling. Однако такие операции должны выполняться только на обучающей выборке, чтобы не искажать тестовую оценку.

Пример полного цикла подготовки табличных данных

Рассмотрим упрощённый пример подготовки набора сетевого трафика. Пусть имеются признаки duration, protocol, packets, bytes и label.

```
import pandas as pd

data = pd.read_csv("traffic.csv")

# 1. Удаляем точные дубликаты
data = data.drop_duplicates()

# 2. Заполняем пропуски
data["bytes"] = data["bytes"].fillna(data["bytes"].median())
data["protocol"] =
data["protocol"].fillna(data["protocol"].mode()[0])
```

```
# 3. Кодируем категориальный признак
data = pd.get_dummies(data, columns=["protocol"], dtype=int)

# 4. Формируем X и y
X = data.drop(columns=["label"])
y = data["label"]
```

Далее выполняется разделение данных, а масштабирование обучается только на X_train.

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Обратите внимание на различие fit_transform и transform. На обучающих данных вычисляются параметры стандартизации и сразу выполняется преобразование. Для тестовых данных используются уже найденные параметры. Это предотвращает утечку информации из test в train.

Пример подготовки текстовых данных

Для текстов этапы отличаются от обработки обычной таблицы, но общая логика сохраняется: проверка качества → очистка → преобразование в числовое представление → разделение данных.

Например, в задаче классификации обращений пользователей можно удалить технические дубликаты, проверить пустые документы, унифицировать кодировку и сформировать метки классов. Затем текст преобразуется в признаки — например, TF-IDF-векторы или эмбеддинги.

Важно не удалять информацию механически. В задаче выявления фишинга символы, URL, регистр, необычные последовательности и специальные знаки могут быть полезными признаками. То, что является «шумом» в одной задаче, может быть сигналом в другой.

Пример подготовки изображений

Изображение для модели представляется числовым массивом. Типичная подготовка включает приведение изображений к единому размеру, масштабирование значений пикселей, проверку повреждённых файлов и формирование меток классов.

Например, изображения размером 224×224 пикселя могут быть преобразованы к значениям в диапазоне [0,1]. Для увеличения разнообразия обучающей выборки иногда применяется аугментация: поворот, отражение, небольшое изменение масштаба. Но преобразование должно сохранять смысл объекта. Для распознавания цифр переворот изображения может изменить класс и быть недопустимым.

Безопасность, приватность и качество данных

При подготовке данных для систем ИИ необходимо учитывать не только техническое качество, но и требования безопасности. Набор данных может содержать персональные данные, идентификаторы, адреса, медицинские сведения, финансовую или служебную информацию.

Перед передачей данных в учебную или исследовательскую среду необходимо определить, какие поля действительно нужны модели. Избыточные идентификаторы следует исключать или обезличивать в соответствии с требованиями проекта и законодательства. Доступ к исходным данным должен быть ограничен, а действия по подготовке — документированы.

Для кибербезопасности важен и другой аспект: обучающие данные могут быть намеренно искажены. Если злоумышленник получает возможность добавлять специально подготовленные примеры, возникает риск poisoning attack — отравления обучающей выборки. Поэтому происхождение данных, контроль целостности и проверка аномалий являются частью безопасного ML-процесса.

Pipeline предобработки

В реальном проекте полезно объединять последовательные преобразования в единый pipeline. Это уменьшает риск случайно применить разные операции к обучающим и новым данным и повышает воспроизводимость эксперимента.

Например, pipeline может выполнять заполнение пропусков, one-hot encoding категориальных признаков, стандартизацию числовых признаков и затем передавать результат модели. При поступлении нового объекта выполняется та же последовательность операций, что использовалась при обучении.

Особенно важно сохранять не только обученную модель, но и параметры предобработки: список признаков, словари категорий, scaler, правила заполнения пропусков. Иначе модель в эксплуатации может получать данные в иной форме, чем во время обучения.

Типичные ошибки при подготовке данных

Первая ошибка — удалять все пропуски и выбросы автоматически. Решение должно зависеть от природы данных и задачи.

Вторая ошибка — кодировать категории произвольными числами и тем самым создавать ложный порядок.

Третья ошибка — выполнять масштабирование до разделения train/test и тем самым допускать утечку данных.

Четвёртая ошибка — использовать accuracy как единственную метрику при сильном дисбалансе классов.

Пятая ошибка — оставлять идентификаторы и признаки, которые содержат готовый ответ.

Шестая ошибка — по-разному обрабатывать данные при обучении и эксплуатации модели.

Седьмая ошибка — не сохранять информацию о выполненных преобразованиях, из-за чего эксперимент невозможно воспроизвести.

Практический мини-кейс: подготовка данных для обнаружения подозрительного входа

Предположим, система анализирует попытки входа в корпоративную сеть. Для каждой попытки имеются user_id, hour, country, device, failed_attempts, ip_reputation и label. Требуется подготовить данные для модели, определяющей normal/suspicious.

Шаг 1. Проверяем дубликаты и пропуски. Если ip_reputation отсутствует, выясняем, означает ли это отсутствие оценки или ошибку загрузки.

Шаг 2. user_id рассматриваем отдельно: уникальный идентификатор может привести к запоминанию пользователей. Вместо него можно сформировать поведенческие признаки, например число входов за последние сутки, если такие данные доступны.

Шаг 3. country и device являются категориальными признаками и требуют кодирования. hour можно оставить числом либо преобразовать в признаки времени суток.

Шаг 4. Анализируем классы. Если suspicious составляет 2%, одной accuracy недостаточно; особенно важен recall подозрительных входов и число ложных тревог.

Шаг 5. Разделяем train/test до обучения преобразований. Если данные имеют временную природу, случайное перемешивание не всегда корректно: для проверки будущего поведения может потребоваться разделение по времени.

Этот пример показывает, что подготовка данных — это не набор механических команд Pandas. Каждый шаг связан с постановкой задачи, смыслом признаков и будущими условиями эксплуатации модели.

Заключение

Подготовка и обработка данных являются фундаментальным этапом разработки систем искусственного интеллекта. До обучения модели необходимо исследовать структуру набора данных, выявить пропуски, дубликаты, ошибки и аномалии, корректно обработать категориальные признаки, привести числовые признаки к подходящему масштабу и сформировать информативное пространство признаков.

Особое значение имеет правильная организация эксперимента: разделение обучающей и тестовой выборок, предотвращение утечки данных, учёт дисбаланса классов и применение одинаковой процедуры предобработки при обучении и эксплуатации.

Качественная предобработка не гарантирует идеальную модель, но некорректная подготовка способна сделать бессмысленным даже самый современный алгоритм. Поэтому специалист по искусственному интеллекту должен уметь не только выбирать модель, но и критически оценивать данные, на которых она строится.

Контрольные вопросы

1. Что понимается под подготовкой данных в системах искусственного интеллекта?
2. Почему качество исходных данных влияет на качество модели?
3. Чем отличаются структурированные, полуструктурированные и неструктурированные данные?
4. Что такое объект, признак и целевая переменная?
5. Какие действия включает первичный анализ набора данных?
6. Для чего используется описательная статистика?
7. Почему дубликаты нельзя всегда удалять автоматически?
8. Какие причины возникновения пропущенных значений возможны?
9. Какими способами можно обрабатывать пропуски?
10. Чем выброс отличается от ошибочного значения?
11. Почему аномальные значения особенно осторожно следует обрабатывать в задачах кибербезопасности?
12. Для чего выполняется кодирование категориальных признаков?
13. В чём заключается one-hot encoding?
14. Чем нормализация отличается от стандартизации?
15. Для каких алгоритмов масштаб признаков особенно важен?
16. Что такое Feature Engineering? Приведите пример.
17. Для чего выполняется отбор признаков?
18. Что такое утечка данных и почему она опасна?
19. Почему параметры масштабирования следует определять только по обучающей выборке?
20. Чем отличаются обучающая, валидационная и тестовая выборки?
21. Что такое дисбаланс классов и почему accuracy может вводить в заблуждение?
22. Какие особенности имеет подготовка текстовых данных?
23. Какие операции могут использоваться при подготовке изображений?
24. Какие требования безопасности необходимо учитывать при работе с данными?
25. Почему pipeline предобработки повышает воспроизводимость AI-проекта?

Список источников:

1. Бессмертный И.А., Авдюшина А.Е., Кугаевских А.В., Королева Ю.А., Рущенко Н.Г. Системы искусственного интеллекта. Методическое пособие. Санкт-Петербург:

Университет ИТМО, 2024. — основной источник по практической подготовке данных и классическим методам машинного обучения.

2. Рабочая программа дисциплины «Искусственный интеллект и машинное обучение». Москва, 2024. — использована структура этапов ML: сбор, исследование и подготовка данных, обучение и оценка модели; перечень инструментов NumPy, Pandas, Matplotlib и scikit-learn.

3. Сидоркина И.Г. Системы искусственного интеллекта. Учебное пособие. Москва: КНОРУС, 2018. — дополнительный источник по системам искусственного интеллекта и представлению данных.

4. Калинин М.О., Крундышев В.М. Основы искусственного интеллекта. Безопасность искусственного интеллекта. Учебное пособие. Санкт-Петербург, 2024. — дополнительный источник по вопросам безопасности ИИ.