АО «АЛМАТИНСКИЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет инженерных и информационных технологий

Кафедра компьютерной инженерии

ПРОЕКТИРОВАНИЕ СИСТЕМ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

ЛЕКЦИЯ № 6

Проектирование моделей машинного обучения

Теоретико-практический конспект

Составитель: Алтыбай А.

Кафедра компьютерной инженерии

Алматы, 2026

Аннотация и назначение лекции

Лекция 6 раскрывает тему «Проектирование моделей машинного обучения» 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, 2]

План занятия

1. Baseline и проверяемая гипотеза
2. Выбор семейства алгоритмов
3. Обучающая цель
4. Train/validation/test
5. Дисбаланс классов
6. Воспроизводимость экспериментов
7. Гиперпараметры и бюджет
8. Математическая калибровка
9. Практические сценарии, контроль качества и проектные задания.

Ожидаемые результаты обучения

По завершении студент способен:

1. объяснить и аргументированно применить принцип «Baseline и проверяемая гипотеза» к инженерному решению;
2. объяснить и аргументированно применить принцип «Выбор семейства алгоритмов» к инженерному решению;

3. объяснить и аргументированно применить принцип «Обучающая цель» к инженерному решению;
4. объяснить и аргументированно применить принцип «Train/validation/test» к инженерному решению;
5. объяснить и аргументированно применить принцип «Дисбаланс классов» к инженерному решению;
6. объяснить и аргументированно применить принцип «Воспроизводимость экспериментов» 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 сервис предсказания срочности. Существенные условия: плохой baseline, утечку данных. Основная проверяемая характеристика: macro-F1 и время инференса. 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, 2] Цель лекции. научить выбирать модель как компонент системы, строить baseline, организовывать сравнение алгоритмов и управлять гиперпараметрами с учетом качества, latency и стоимости.

Ключевые понятия. baseline, model selection, bias-variance, hyperparameters, cross-validation, calibration, latency, memory footprint, cost-aware modeling.

1.1 Модель как инженерный компонент

Выбор алгоритма начинается с требований. Для табличной задачи градиентный бустинг может быть сильнее сложной нейронной сети и проще в эксплуатации. Для текста или изображений pretrained foundation model может дать быстрый старт, но повысить стоимость и зависимость от инфраструктуры.

В production важны размер модели, время загрузки, throughput, требования к CPU/GPU, возможность объяснения и стабильность при обновлениях.

1.2 Baseline и лестница сложности

Рекомендуется строить последовательность: простое правило → линейная модель → сильная классическая модель → более сложная модель. Каждый шаг должен давать измеримое улучшение, оправдывающее стоимость. Это предотвращает ситуацию, когда команда долго оптимизирует сложную архитектуру без доказанной бизнес-ценности.

Baseline также служит контрольной точкой при мониторинге: новая версия не должна быть хуже простого надежного решения по критическим метрикам.

1.3 Bias, variance и переобучение

Недостаточная сложность проявляется как высокий bias: модель систематически не описывает закономерности. Слишком высокая чувствительность к обучающей выборке проявляется как variance и приводит к переобучению. Диагностика выполняется через train/validation curves, cross-validation и error analysis.

Регуляризация, больше данных, упрощение модели и корректная валидация помогают управлять переобучением.

1.4 Гиперпараметры и поиск

Гиперпараметры задаются до обучения: глубина дерева, learning rate, число слоев, коэффициент регуляризации. Grid search удобен для небольшого пространства, random search часто эффективнее при большом числе параметров, Bayesian optimization и современные оптимизаторы trials полезны для дорогих экспериментов.

Поиск должен выполняться на validation/cross-validation; тестовая выборка не используется для итеративной настройки.

1.5 Пример воспроизводимого baseline

Ниже показан принцип объединения preprocessing и модели в один Pipeline. Это уменьшает риск различий между обучением и применением.

from sklearn.pipeline import Pipeline from sklearn.preprocessing import StandardScaler from sklearn.linear_model import LogisticRegression

model = Pipeline([(”scale”, StandardScaler()), (”clf”, LogisticRegression(max_iter=1000))]) model.fit(X_train, y_train) score = model.score(X_valid, y_valid)

1.6 Выводы

- Модель выбирается по совокупности качества и эксплуатационных ограничений.
- Baseline обязателен как точка сравнения.
- Гиперпараметры настраиваются без использования final test.
- Pipeline помогает сделать preprocessing и inference согласованными.

1.7 Контрольные вопросы

- Какие системные факторы влияют на выбор модели?

- Что означает bias-variance trade-off?
- Почему test нельзя использовать при подборе гиперпараметров?
- Когда простая модель предпочтительнее сложной?
- Зачем объединять preprocessing и модель в Pipeline?

2. Систематическое изложение темы и инженерные решения

2.1 Baseline и проверяемая гипотеза

Построение ML-модели начинается с метрики простого алгоритма: most frequent, линейного классификатора или эвристики. Важен честный общий тест. [2]

> **Проверяемый пример.** Сравните DummyClassifier, LogisticRegression и RandomForest на неизменном split.

Вопрос для обсуждения. Как проверить выполнение такого требования: Сравните DummyClassifier, LogisticRegression и RandomForest на неизменном split.

2.2 Выбор семейства алгоритмов

Линейные модели быстры и объяснимы; ансамбли деревьев подходят для табличных признаков; DL полезно для неструктурированных данных. Подбор отражает данные и требования. [3]

> **Проверяемый пример.** Для 50 тысяч строк с категориальными признаками аргументируйте выбор градиентного бустинга против CNN.

Вопрос для обсуждения. Как проверить выполнение такого требования: Для 50 тысяч строк с категориальными признаками аргументируйте выбор градиентного бустинга против CNN.

2.3 Обучающая цель

Функция потерь задает то, что минимизирует алгоритм, а продуктовая метрика — то, что важно бизнесу. Эти величины могут различаться. [4]

Проверяемый пример. Сопоставьте log loss в оптимизаторе и F1 при выборе порога решения.

Вопрос для обсуждения. Как проверить выполнение такого требования: Сопоставьте log loss в оптимизаторе и F1 при выборе порога решения.

2.4 Train/validation/test

Train обновляет параметры, validation выбирает модель, test оценивает окончательную процедуру. Повторный выбор по test делает оценку оптимистичной. [5]

Проверяемый пример. Зафиксируйте test до поиска лучших hyperparameters.

Вопрос для обсуждения. Как проверить выполнение такого требования: Зафиксируйте test до поиска лучших hyperparameters.

2.5 Дисбаланс классов

Accuracy может скрывать провал по редким классам; class weights, изменение порога и sampling помогают разными способами. Повторная выборка осуществляется внутри train-fold. [6]

Проверяемый пример. Выведите, почему accuracy=0.99 у модели «всё отрицательно» при 1% позитивных объектов.

Вопрос для обсуждения. Как проверить выполнение такого требования: Выведите, почему accuracy=0.99 у модели «всё отрицательно» при 1% позитивных объектов.

2.6 Воспроизводимость экспериментов

Фиксируют версии данных, код, seeds, характеристики среды и артефакты. Один seed не дает абсолютной детерминированности на любой аппаратуре. [1]

Проверяемый пример. Сравните несколько запусков и запишите среднее и разброс F1.

Вопрос для обсуждения. Как проверить выполнение такого требования: Сравните несколько запусков и запишите среднее и разброс F1.

2.7 Гиперпараметры и бюджет

Определение пространства поиска до эксперимента снижает вероятность случайного подбора по test. Время обучения и энергоресурсы входят в стоимость. [2]

Проверяемый пример. Поставьте бюджет 20 trials и примените одинаковые правила ранней остановки.

Вопрос для обсуждения. Как проверить выполнение такого требования: Поставьте бюджет 20 trials и примените одинаковые правила ранней остановки.

2.8 Математическая калибровка

Предсказанные вероятности должны соответствовать наблюдаемым частотам, когда от них зависит решение. Калибровку проводят на отдельной выборке. [3]

Проверяемый пример. Для группы с прогнозом 0.8 проверьте, какая доля событий реально произошла.

Вопрос для обсуждения. Как проверить выполнение такого требования:

Для группы с прогнозом 0.8 проверьте, какая доля событий реально произошла.

2.9 Интерпретация ошибок

Confusion matrix и анализ срезов позволяют найти слабые места, которые средняя метрика скрывает. Анализ должен завершаться гипотезой изменения данных или модели. [4]

Проверяемый пример. Исследуйте ошибки для разных языков текстов обращений.

Вопрос для обсуждения. Как проверить выполнение такого требования: Исследуйте ошибки для разных языков текстов обращений.

2.10 Скорость и размер модели

Сложность модели влияет на память, inference latency и возможность обновлений. Сравнение quality-only неполно. [5]

Проверяемый пример. Измерьте p95 и размер сериализованной модели для двух моделей с близким F1.

Вопрос для обсуждения. Как проверить выполнение такого требования: Измерьте p95 и размер сериализованной модели для двух моделей с близким F1.

2.11 Честное сравнение

Используйте одинаковые границы данных и feature pipeline, иначе улучшение объясняется не моделью, а утечкой или изменением условий. [6]

Проверяемый пример. Сравните две модели по одинаковым разбиениям, с диапазоном колебаний метрик.

Вопрос для обсуждения. Как проверить выполнение такого требования: Сравните две модели по одинаковым разбиениям, с диапазоном колебаний метрик.

2.12 Доставка модели

Артефакты модели, входная схема, преобразователи и зависимые версии должны сохраняться как единый контракт. [1]

> **Проверяемый пример.** Покажите, почему pickle-модель без preprocessing не готова к внедрению.

Вопрос для обсуждения. Как проверить выполнение такого требования: Покажите, почему pickle-модель без preprocessing не готова к внедрению.

3. Математические инструменты и количественные критерии

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

$$\hat{\theta} = \arg\min_{\theta} \left\{ \frac{1}{n} \sum_{i=1}^{n} \ell(f_\theta(x_i), y_i) + \lambda \Omega(\theta) \right\} \tag{1}$$

$$\widehat{\mathrm{Risk}}_{\mathrm{test}} = \frac{1}{m} \sum_{i=1}^{m} \ell(f_{\hat{\theta}}(x_i), y_i) \tag{2}$$

Выбор архитектуры и λ должен происходить без использования отложенного test, иначе оценка обобщения смещается.

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.

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

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, 2]

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

```
from sklearn.datasets import load_breast_cancer
from sklearn.dummy import DummyClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score

X, y = load_breast_cancer(return_X_y=True)
X_tr, X_te, y_tr, y_te = train_test_split(
    X, y, stratify=y, test_size=0.2, random_state=17)
for name, algorithm in [("baseline", DummyClassifier(strategy="most_frequent")),
                        ("logreg", LogisticRegression(max_iter=3000))]:
    algorithm.fit(X_tr, y_tr)
    pred = algorithm.predict(X_te)
    print(name, "F1:", round(f1_score(y_te, pred), 3))
```

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

```
from sklearn.metrics import confusion_matrix
y_true, y_pred = [0, 1, 1, 0], [0, 0, 1, 0]
```

```
print(confusion_matrix(y_true, y_pred))
```

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: Baseline и проверяемая гипотеза

Заказчик использует сервис предсказания срочности. Новое требование заставляет рассмотреть принцип «Baseline и проверяемая гипотеза». Основная неопределённость состоит в том, как одновременно обеспечить macro-F1 и время инференса, не допустить утечку данных и сохранить возможность изменения модели.

Постановка. Построение ML-модели начинается с метрики простого алгоритма: most frequent, линейного классификатора или эвристики. Важен честный общий тест.

Действие инженера. Сравните DummyClassifier, LogisticRegression и RandomForest на неизменном split.

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: Воспроизводимость экспериментов

Заказчик использует сервис предсказания срочности. Новое требование заставляет рассмотреть принцип «Воспроизводимость экспериментов». Основная неопределённость состоит в том, как одновременно обеспечить macro-F1 и время инференса, не допустить утечку данных и сохранить возможность изменения модели.

Постановка. Фиксируют версии данных, код, seeds, характеристики среды и артефакты. Один seed не дает абсолютной детерминированности на любой аппаратуре.

Действие инженера. Сравните несколько запусков и запишите среднее и разброс F1.

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: Честное сравнение

Заказчик использует сервис предсказания срочности. Новое требование заставляет рассмотреть принцип «Честное сравнение». Основная неопределённость состоит в том, как одновременно обеспечить macro-F1 и время инференса, не допустить утечку данных и сохранить возможность изменения модели.

Постановка. Используйте одинаковые границы данных и feature pipeline, иначе улучшение объясняется не моделью, а утечкой или изменением условий.

Действие инженера. Сравните две модели по одинаковым разбиениям, с диапазоном колебаний метрик.

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. **Baseline и проверяемая гипотеза.** Сравните DummyClassifier, LogisticRegression и RandomForest на неизменном split. Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.

2. **Выбор семейства алгоритмов.** Для 50 тысяч строк с категориальными признаками аргументируйте выбор градиентного бустинга против CNN. Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.

3. **Обучающая цель.** Сопоставьте log loss в оптимизаторе и F1 при выборе порога решения. Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.

4. **Train/validation/test.** Зафиксируйте test до поиска лучших hyperparameters. Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.

5. **Дисбаланс классов.** Выведите, почему accuracy=0.99 у модели «всё отрицательно» при 1% позитивных объектов. Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.

6. **Воспроизводимость экспериментов.** Сравните несколько запусков и запишите среднее и разброс F1. Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.

7. **Гиперпараметры и бюджет.** Поставьте бюджет 20 trials и примените одинаковые правила ранней остановки. Сформулируйте проверяемое

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

8. **Математическая калибровка.** Для группы с прогнозом 0.8 проверьте, какая доля событий реально произошла. Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.

9. **Интерпретация ошибок.** Исследуйте ошибки для разных языков текстов обращений. Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.

10. **Скорость и размер модели.** Измерьте p95 и размер сериализованной модели для двух моделей с близким F1. Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.

11. **Честное сравнение.** Сравните две модели по одинаковым разбиениям, с диапазоном колебаний метрик. Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.

12. **Доставка модели.** Покажите, почему pickle-модель без preprocessing не готова к внедрению. 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 «Baseline и проверяемая гипотеза»?

2. Как проверить на практике утверждение: «Сравните DummyClassifier, LogisticRegression и RandomForest на неизменном split.»?

3. В чём назначение и ограничения принципа «Выбор семейства алгоритмов»?

4. Как проверить на практике утверждение: «Для 50 тысяч строк с категориальными признаками аргументируйте выбор градиентного бустинга против CNN.»?
5. В чём назначение и ограничения принципа «Обучающая цель»?
6. Как проверить на практике утверждение: «Сопоставьте log loss в оптимизаторе и F1 при выборе порога решения.»?
7. В чём назначение и ограничения принципа «Train/validation/test»?
8. Как проверить на практике утверждение: «Зафиксируйте test до поиска лучших hyperparameters.»?
9. В чём назначение и ограничения принципа «Дисбаланс классов»?
10. Как проверить на практике утверждение: «Выведите, почему accuracy=0.99 у модели «всё отрицательно» при 1% позитивных объектов.»?
11. В чём назначение и ограничения принципа «Воспроизводимость экспериментов»?
12. Как проверить на практике утверждение: «Сравните несколько запусков и запишите среднее и разброс F1.»?
13. В чём назначение и ограничения принципа «Гиперпараметры и бюджет»?
14. Как проверить на практике утверждение: «Поставьте бюджет 20 trials и примените одинаковые правила ранней остановки.»?
15. В чём назначение и ограничения принципа «Математическая калибровка»?
16. Как проверить на практике утверждение: «Для группы с прогнозом 0.8 проверьте, какая доля событий реально произошла.»?
17. В чём назначение и ограничения принципа «Интерпретация ошибок»?
18. Как проверить на практике утверждение: «Исследуйте ошибки для разных языков текстов обращений.»?

19. В чём назначение и ограничения принципа «Скорость и размер модели»?
20. Как проверить на практике утверждение: «Измерьте p95 и размер сериализованной модели для двух моделей с близким F1.»?
21. В чём назначение и ограничения принципа «Честное сравнение»?
22. Как проверить на практике утверждение: «Сравните две модели по одинаковым разбиениям, с диапазоном колебаний метрик.»?
23. В чём назначение и ограничения принципа «Доставка модели»?
24. Как проверить на практике утверждение: «Покажите, почему pickle-модель без preprocessing не готова к внедрению.»?
25. Почему высокое offline-качество не гарантирует надежность промышленной AI-системы?
26. Какие артефакты нужны для воспроизводимости и анализа инцидента?

8. Глоссарий ключевых понятий

Baseline и проверяемая гипотеза

Построение ML-модели начинается с метрики простого алгоритма: most frequent, линейного классификатора или эвристики.

Выбор семейства алгоритмов

Линейные модели быстры и объяснимы; ансамбли деревьев подходят для табличных признаков; DL полезно для неструктурированных данных.

Обучающая цель

Функция потерь задает то, что минимизирует алгоритм, а продуктовая метрика — то, что важно бизнесу.

Train/validation/test

Train обновляет параметры, validation выбирает модель, test оценивает окончательную процедуру.

Дисбаланс классов

Accuracy может скрывать провал по редким классам; class weights, изменение порога и sampling помогают разными способами.

Воспроизводимость экспериментов

Фиксируют версии данных, код, seeds, характеристики среды и артефакты.

Гиперпараметры и бюджет

Определение пространства поиска до эксперимента снижает вероятность случайного подбора по test.

Математическая калибровка

Предсказанные вероятности должны соответствовать наблюдаемым частотам, когда от них зависит решение.

Интерпретация ошибок

Confusion matrix и анализ срезов позволяют найти слабые места, которые средняя метрика скрывает.

Скорость и размер модели

Сложность модели влияет на память, inference latency и возможность обновлений.

Честное сравнение

Используйте одинаковые границы данных и feature pipeline, иначе улучшение объясняется не моделью, а утечкой или изменением условий.

Доставка модели

Артефакты модели, входная схема, преобразователи и зависимые версии должны сохраняться как единый контракт.

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] 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/.

[2] Murphy K. P. Probabilistic Machine Learning: An Introduction. MIT Press, 2022. https://probml.github.io/pml-book/.

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

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

[5] Goodfellow I., Bengio Y., Courville A. Deep Learning. MIT Press, 2016. https://www.deeplearningbook.org/.

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