АО «АЛМАТИНСКИЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет инженерных и информационных технологий

Кафедра компьютерной инженерии

ПРОЕКТИРОВАНИЕ СИСТЕМ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

ЛЕКЦИЯ № 5

Feature Engineering и представление данных

Теоретико-практический конспект

Составитель: Алтыбай А.

Кафедра компьютерной инженерии

Алматы, 2026

Аннотация и назначение лекции

Лекция 5 раскрывает тему «Feature Engineering и представление данных» 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. Назначение признаков
2. Числовые преобразования
3. Категориальные переменные
4. Временные признаки
5. Текстовые представления
6. Визуальные признаки
7. Feature selection
8. Feature leakage и train–serve skew
9. Практические сценарии, контроль качества и проектные задания.

Ожидаемые результаты обучения

По завершении студент способен:

1. объяснить и аргументированно применить принцип «Назначение признаков» к инженерному решению;
2. объяснить и аргументированно применить принцип «Числовые преобразования» к инженерному решению;

3. объяснить и аргументированно применить принцип «Категориальные переменные» к инженерному решению;
4. объяснить и аргументированно применить принцип «Временные признаки» к инженерному решению;
5. объяснить и аргументированно применить принцип «Текстовые представления» к инженерному решению;
6. объяснить и аргументированно применить принцип «Визуальные признаки» 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 классификации входящих писем. Существенные условия: неверные признаки времени, расхождение train и serving. Основная проверяемая характеристика: ошибку классификации по временным срезам. 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. рассмотреть способы представления числовых, категориальных, текстовых и мультимодальных данных; показать роль feature engineering, embeddings и feature store.

Ключевые понятия. feature engineering, scaling, encoding, missing values, feature selection, embedding, representation learning, feature store, online/offline features.

1.1 Признак как интерфейс между данными и моделью

Признак — измеримое представление объекта, используемое моделью. Хороший признак должен быть доступен в момент inference, иметь устойчивое определение и не содержать запрещенной или утекшей информации. Качество признаков часто сильнее влияет на результат табличных моделей, чем замена одного алгоритма другим.

При проектировании полезно отделять исходные поля, производные признаки и представления, изученные моделью.

1.2 Числовые и категориальные признаки

Числовые признаки могут требовать нормализации, логарифмирования, winsorization или явного кодирования пропусков. Категориальные признаки кодируются one-hot, ordinal, frequency/target encoding или embeddings. Target encoding требует особенно аккуратной cross-validation, иначе возникает leakage.

Preprocessing должен быть fit только на обучающей части данных. Например, среднее для стандартизации нельзя вычислять по всей выборке до разбиения.

1.3 Текст, изображения и embeddings

Для текста классические представления включают bag-of-words и TF-IDF; современные системы используют dense embeddings, где семантически близкие объекты имеют близкие векторы. Для изображений признаки часто формируются CNN или vision transformer. Embeddings позволяют унифицировать поиск, кластеризацию и retrieval.

Embedding не является «истиной»: его полезность зависит от модели, домена, размерности и метрики близости.

1.4 Feature selection и устойчивость

Отбор признаков уменьшает сложность, может улучшить интерпретируемость и снизить риск переобучения. Используются фильтровые методы, регуляризация, importance-based подходы и ablation studies. Однако удаление признаков по feature importance одной модели не всегда переносится на другую.

В production важна не только predictive power, но и стабильность признака: насколько надежно он вычисляется, как часто меняется схема, какова цена получения и latency.

1.5 Feature store

Feature store — инфраструктурный слой для хранения и повторного использования определений признаков. Он может предоставлять offline features для обучения и online features для low-latency inference. Ключевая задача — обеспечить одинаковую семантику признаков в обоих режимах.

Feature store особенно полезен, когда несколько моделей используют одинаковые агрегаты, но требует дисциплины версий, ownership и контроля свежести.

1.6 Выводы

▷ Feature engineering — инженерия устойчивых представлений, а не только математическое преобразование.

▷ Preprocessing должен быть встроен в воспроизводимый pipeline.

▷ Embeddings дают компактное векторное представление сложных объектов.

▷ Feature store помогает согласовать offline и online вычисление признаков.

1.7 Контрольные вопросы

- ▷ Почему preprocessing следует обучать только на train?
- ▷ Чем one-hot encoding отличается от embeddings?
- ▷ Какие риски возникают при target encoding?
- ▷ Что такое feature store?
- ▷ Почему стабильность признака может быть важнее небольшого прироста accuracy?

2. Систематическое изложение темы и инженерные решения

2.1 Назначение признаков

Признаки преобразуют сырые наблюдения в представление, доступное выбранному алгоритму. Полезность определяется не только корреляцией, но также доступностью при предсказании и стабильностью во времени. [2]

Проверяемый пример. Для клиентского обращения рассмотрите длину текста, канал, время регистрации и историю клиента.

Вопрос для обсуждения. Как проверить выполнение такого требования: Для клиентского обращения рассмотрите длину текста, канал, время регистрации и историю клиента.

2.2 Числовые преобразования

Стандартизация и нормализация меняют геометрию оптимизации, но не делают плохой признак содержательно полезным. Statistics fitted on train only. [3]

Проверяемый пример. Сравните StandardScaler и RobustScaler при экстремальных суммах заказов.

Вопрос для обсуждения. Как проверить выполнение такого требования: Сравните StandardScaler и RobustScaler при экстремальных суммах заказов.

2.3 Категориальные переменные

One-hot, ordinal, target encoding и learned embeddings учитывают структуру категорий по-разному. Target encoding особенно чувствителен к утечке. [4]

Проверяемый пример. Объясните, почему ID клиента нельзя безоговорочно трактовать как числовую величину.

Вопрос для обсуждения. Как проверить выполнение такого требования: Объясните, почему ID клиента нельзя безоговорочно трактовать как числовую величину.

2.4 Временные признаки

Циклические компоненты часа и дня недели выражаются синусом и косинусом; окна и лаги зависят от порядка событий. Признак будущего недоступен при онлайн-скоринге. [5]

Проверяемый пример. Закодируйте время 23:00 и 00:00 так, чтобы они были близки в геометрии признаков.

Вопрос для обсуждения. Как проверить выполнение такого требования: Закодируйте время 23:00 и 00:00 так, чтобы они были близки в геометрии признаков.

2.5 Текстовые представления

Bag-of-words и TF-IDF фиксируют частотную информацию, embeddings кодируют плотные векторы и семантическую близость. Представление зависит от задачи и стоимости инференса. [6]

Проверяемый пример. Сравните TF-IDF+логистическую регрессию с эмбеддингами для тематической классификации обращений.

Вопрос для обсуждения. Как проверить выполнение такого требования: Сравните TF-IDF+логистическую регрессию с эмбеддингами для тематической классификации обращений.

2.6 Визуальные признаки

Изображение можно описать ручными дескрипторами или иерархией признаков CNN. Масштабирование и цветовая нормировка должны совпадать при обучении и serving. [1]

Проверяемый пример. Проверьте, что порядок каналов RGB не меняется между обучением и API.

Вопрос для обсуждения. Как проверить выполнение такого требования: Проверьте, что порядок каналов RGB не меняется между обучением и API.

2.7 Feature selection

Фильтровые, wrapper и embedded подходы снижают избыточность, шум и вычисления. Влияние на качество измеряют на независимой validation-выборке. [2]

Проверяемый пример. Оцените пользу удаления двух почти полностью коррелирующих признаков.

Вопрос для обсуждения. Как проверить выполнение такого требования:

Оцените пользу удаления двух почти полностью коррелирующих признаков.

2.8 Feature leakage и train–serve skew

Точная копия preprocessing pipeline предотвращает расхождение между offline и online. Неявное дублирование расчета признака опасно. [3]

Проверяемый пример. Найдите ошибку, когда train использует UTC, а API — локальное время без зоны.

Вопрос для обсуждения. Как проверить выполнение такого требования: Найдите ошибку, когда train использует UTC, а API — локальное время без зоны.

2.9 Feature store

Feature store хранит определения, происхождение и иногда online/offline представления признаков. Point-in-time correctness защищает от использования будущих значений. [4]

Проверяемый пример. Опишите, как получить сумму покупок до момента t, а не по полной сегодняшней таблице.

Вопрос для обсуждения. Как проверить выполнение такого требования: Опишите, как получить сумму покупок до момента t, а не по полной сегодняшней таблице.

2.10 Размерность и проклятие размерности

Рост числа признаков усложняет оптимизацию и поиск соседей, увеличивая потребность в данных. Снижение размерности оценивают через сохранение качества задачи. [5]

Проверяемый пример. Сравните PCA на dense-признаках и TruncatedSVD на разреженном TF-IDF.

Вопрос для обсуждения. Как проверить выполнение такого требования: Сравните PCA на dense-признаках и TruncatedSVD на разреженном TF-IDF.

2.11 Embedding space

Косинусное сходство оценивает угол между векторами и зависит от нормализации. Близость в пространстве векторов не гарантирует корректность ответа приложения. [6]

Проверяемый пример. Постройте поиск топ-5 похожих обращений и вручную оцените релевантность.

Вопрос для обсуждения. Как проверить выполнение такого требования: Постройте поиск топ-5 похожих обращений и вручную оцените релевантность.

2.12 Критерии надежного pipeline

Обучение трансформеров только на train, серийные тесты схемы и единый сериализованный pipeline делают признаки воспроизводимыми. [1]

Проверяемый пример. Сохраните sklearn Pipeline и проверьте одинаковый результат до/после сериализации.

Вопрос для обсуждения. Как проверить выполнение такого требования: Сохраните sklearn Pipeline и проверьте одинаковый результат до/после сериализации.

3. Математические инструменты и количественные критерии

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

$$z_j = \frac{x_j - \mu_j}{\sigma_j} \tag{1}$$

$$\mathrm{sim}(u, v) = \frac{u^\top v}{\|u\|_2 \|v\|_2} \tag{2}$$

Оценки μ_j и σ_j получают на train. Преобразование применяют к будущим объектам без повторного обучения статистик.

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, 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 numpy as np
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression

# Define columns before feature fitting.
prep = ColumnTransformer([
    ("numeric", StandardScaler(), ["age", "spend"]),
    ("category", OneHotEncoder(handle_unknown="ignore"), ["channel"])
])
model = Pipeline([("features", prep),
                  ("classifier", LogisticRegression(max_iter=300))])
print(model.get_params()["features"])
```

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

```
import numpy as np
h = np.array([23, 0, 1, 12])
```

```
a = 2 * np.pi * h / 24
print(np.c_[np.sin(a), np.cos(a)].round(2))
```

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: Назначение признаков

Заказчик использует сервис классификации входящих писем. Новое требование заставляет рассмотреть принцип «Назначение признаков». Основная неопределённость состоит в том, как одновременно обеспечить ошибку классификации по временным срезам, не допустить расхождение train и serving и сохранить возможность изменения модели.

Постановка. Признаки преобразуют сырые наблюдения в представление, доступное выбранному алгоритму. Полезность определяется не только корреляцией, но также доступностью при предсказании и стабильностью во времени.

Действие инженера. Для клиентского обращения рассмотрите длину текста, канал, время регистрации и историю клиента.

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: Визуальные признаки

Заказчик использует сервис классификации входящих писем. Новое требование заставляет рассмотреть принцип «Визуальные признаки». Основная неопределённость состоит в том, как одновременно обеспечить ошибку классификации по временным срезам, не допустить расхождение train и serving и сохранить возможность изменения модели.

Постановка. Изображение можно описать ручными дескрипторами или иерархией признаков CNN. Масштабирование и цветовая нормировка должны совпадать при обучении и serving.

Действие инженера. Проверьте, что порядок каналов RGB не меняется между обучением и API.

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: Embedding space

Заказчик использует сервис классификации входящих писем. Новое требование заставляет рассмотреть принцип «Embedding space». Основная неопределённость состоит в том, как одновременно обеспечить ошибку классификации по временным срезам, не допустить расхождение train и serving и сохранить возможность изменения модели.

Постановка. Косинусное сходство оценивает угол между векторами и зависит от нормализации. Близость в пространстве векторов не гарантирует корректность ответа приложения.

Действие инженера. Постройте поиск топ-5 похожих обращений и вручную оцените релевантность.

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. **Назначение признаков.** Для клиентского обращения рассмотрите длину текста, канал, время регистрации и историю клиента. Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.
2. **Числовые преобразования.** Сравните StandardScaler и RobustScaler при экстремальных суммах заказов. Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.
3. **Категориальные переменные.** Объясните, почему ID клиента нельзя безоговорочно трактовать как числовую величину. Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.
4. **Временные признаки.** Закодируйте время 23:00 и 00:00 так, чтобы они были близки в геометрии признаков. Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.
5. **Текстовые представления.** Сравните TF-IDF+логистическую регрессию с эмбеддингами для тематической классификации обращений. Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.
6. **Визуальные признаки.** Проверьте, что порядок каналов RGB не меняется между обучением и API. Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.
7. **Feature selection.** Оцените пользу удаления двух почти полностью коррелирующих признаков. Сформулируйте проверяемое требование, про-

демонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.

8. **Feature leakage и train–serve skew.** Найдите ошибку, когда train использует UTC, а API — локальное время без зоны. Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.

9. **Feature store.** Опишите, как получить сумму покупок до момента t, а не по полной сегодняшней таблице. Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.

10. **Размерность и проклятие размерности.** Сравните PCA на dense-признаках и TruncatedSVD на разреженном TF-IDF. Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.

11. **Embedding space.** Постройте поиск топ-5 похожих обращений и вручную оцените релевантность. Сформулируйте проверяемое требование, продемонстрируйте альтернативу и приложите протокол хотя бы одного эксперимента.

12. **Критерии надежного pipeline.** Сохраните sklearn Pipeline и проверьте одинаковый результат до/после сериализации. 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 «Назначение признаков»?
2. Как проверить на практике утверждение: «Для клиентского обращения рассмотрите длину текста, канал, время регистрации и историю клиента.»?
3. В чём назначение и ограничения принципа «Числовые преобразования»?

4. Как проверить на практике утверждение: «Сравните StandardScaler и RobustScaler при экстремальных суммах заказов.»?
5. В чём назначение и ограничения принципа «Категориальные переменные»?
6. Как проверить на практике утверждение: «Объясните, почему ID клиента нельзя безоговорочно трактовать как числовую величину.»?
7. В чём назначение и ограничения принципа «Временные признаки»?
8. Как проверить на практике утверждение: «Закодируйте время 23:00 и 00:00 так, чтобы они были близки в геометрии признаков.»?
9. В чём назначение и ограничения принципа «Текстовые представления»?
10. Как проверить на практике утверждение: «Сравните TF-IDF+логистическую регрессию с эмбеддингами для тематической классификации обращений.»?
11. В чём назначение и ограничения принципа «Визуальные признаки»?
12. Как проверить на практике утверждение: «Проверьте, что порядок каналов RGB не меняется между обучением и API.»?
13. В чём назначение и ограничения принципа «Feature selection»?
14. Как проверить на практике утверждение: «Оцените пользу удаления двух почти полностью коррелирующих признаков.»?
15. В чём назначение и ограничения принципа «Feature leakage и train–serve skew»?
16. Как проверить на практике утверждение: «Найдите ошибку, когда train использует UTC, а API — локальное время без зоны.»?
17. В чём назначение и ограничения принципа «Feature store»?
18. Как проверить на практике утверждение: «Опишите, как получить сумму покупок до момента t, а не по полной сегодняшней таблице.»?
19. В чём назначение и ограничения принципа «Размерность и проклятие размерности»?

20. Как проверить на практике утверждение: «Сравните PCA на dense-признаках и TruncatedSVD на разреженном TF-IDF.»?
21. В чём назначение и ограничения принципа «Embedding space»?
22. Как проверить на практике утверждение: «Постройте поиск топ-5 похожих обращений и вручную оцените релевантность.»?
23. В чём назначение и ограничения принципа «Критерии надежного pipeline»?
24. Как проверить на практике утверждение: «Сохраните sklearn Pipeline и проверьте одинаковый результат до/после сериализации.»?
25. Почему высокое offline-качество не гарантирует надежность промышленной AI-системы?
26. Какие артефакты нужны для воспроизводимости и анализа инцидента?

8. Глоссарий ключевых понятий

Назначение признаков
Признаки преобразуют сырые наблюдения в представление, доступное выбранному алгоритму.

Числовые преобразования
Стандартизация и нормализация меняют геометрию оптимизации, но не делают плохой признак содержательно полезным.

Категориальные переменные
One-hot, ordinal, target encoding и learned embeddings учитывают структуру категорий по-разному.

Временные признаки
Циклические компоненты часа и дня недели выражаются синусом и косинусом; окна и лаги зависят от порядка событий.

Текстовые представления
Bag-of-words и TF-IDF фиксируют частотную информацию, embeddings кодируют плотные векторы и семантическую близость.

Визуальные признаки
Изображение можно описать ручными дескрипторами или иерархией признаков CNN.

Feature selection
Фильтровые, wrapper и embedded подходы снижают избыточность, шум и вычисления.

Feature leakage и train–serve skew
Точная копия preprocessing pipeline предотвращает расхождение между offline и online.

Feature store
Feature store хранит определения, происхождение и иногда online/offline представления признаков.

Размерность и проклятие размерности
Рост числа признаков усложняет оптимизацию и поиск соседей, увеличивая потребность в данных.

Embedding space

Косинусное сходство оценивает угол между векторами и зависит от нормализации.

Критерии надежного pipeline

Обучение трансформеров только на train, серийные тесты схемы и единый сериализованный pipeline делают признаки воспроизводимыми.

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

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

[4] Feast. Feature Store Documentation. https://docs.feast.dev/.

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

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