Министерство науки и высшего образования Российской Федерации
федеральное государственное бюджетное образовательное
учреждение
высшего образования
РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ
ГИДРОМЕТЕОРОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ

Т.А. НИГМАТУЛИН, А.Ю. СИДОРЕНКО, Е.С. НОВОЖИЛОВА

НЕЙРОСЕТИ И МАШИННОЕ ОБУЧЕНИЕ

Часть 1. ПОДГОТОВКА ДАННЫХ

ПРАКТИКУМ

ООО «Свое издательство»
САНКТ-ПЕТЕРБУРГ, 2026

УДК 004.852
ББК 16.632

НЕЙРОСЕТИ И МАШИННОЕ ОБУЧЕНИЕ. ЧАСТЬ 1. ПОДГОТОВКА ДАННЫХ: практикум / Т.А.Нигматулин, А.Ю. Сидоренко, Е.С. Новожилова – СПб. ООО «Свое издательство», 2026. – 69 с.

Практикум содержит сведения о математическом аппарате статистической обработки и методов подготовки данных для моделей анализа и нейросетей. Рассмотрены принципы статистической обработки данных и их подготовки для использования в классических алгоритмах анализа, а также в нейросетях. Методы анализа и подготовки данных поясняются на многочисленных примерах программной реализации на языке Python. Предусмотрены практические задания для самостоятельной работы.

Практикум рассчитан на студентов и аспирантов Российского государственного гидрометеорологический университета, а также на всех интересующихся вопросами разработки аналитических моделей.

СПб. ООО «Свое издательство», 191040, Санкт-Петербург, Пушкинская ул, д. 10 литера А, помещ. 1-н

Оглавление

Введение

Современные нейросетевые модели и алгоритмы машинного обучения демонстрируют впечатляющие результаты в самых разных областях — от компьютерного зрения до прогнозирования временных рядов. Однако успех любой модели на 80–90% зависит не от выбора архитектуры или гиперпараметров, а от **качества и правильной подготовки данных**. На практике «сырые» данные почти всегда содержат пропуски, выбросы, несогласованные форматы, ошибки измерения и другие дефекты. Если скормить нейросети неподготовленные данные, она либо обучится нестабильно (например, из-за выбросов в градиентах), либо будет «учить шум», либо просто не сможет обобщаться.

Цель этого практикума — познакомить вас с ключевыми этапами предобработки данных для нейросетей и других ML-моделей. Мы разберём:

- как провести предварительный статистический анализ данных
- как обнаруживать и обрабатывать пропуски (заполнение, удаление);
- как находить и корректно устранять выбросы (IQR, Z-оценка, винсоризация, логарифмирование);
- как кодировать категориальные признаки (one-hot encoding, label encoding);
- как масштабировать числовые признаки (нормализация, стандартизация) — критически важно для нейросетей;
- как разделять выборку (train / validation / test) без утечек данных;
- и как упаковывать все эти шаги в воспроизводимый пайплайн (на примере Python + scikit-learn).

В качестве сквозного примера мы возьмём реальный (или синтетический) набор данных, где вы сами примените все методы, сравните модели «до» и «после» обработки и убедитесь, насколько сильно подготовка данных влияет на метрики качества.

Ожидаемые результаты практикума

После выполнения заданий вы сможете:

самостоятельно анализировать «грязный» набор данных;

выбирать подходящие методы обработки пропусков и выбросов в зависимости от распределения и смысла признаков;

строить пайплайны предобработки, которые не допускают утечки данных;

оценивать влияние каждого преобразования на качество простых моделей (логистическая регрессия, дерево решений, небольшая нейросеть).

Всё это — база, без которой невозможно эффективно применять нейросети в реальных проектах.

Глава 1. Статистическая обработка данных

1.1. Проверка гипотезы о равенстве средних выборок (неизвестные дисперсии, t-критерий)

Постановка задачи

Имеется k независимых выборок:

$$X_{11}, X_{12}, \dots, X_{1n1} \text{ (объём } n_1)$$

$$X_{21}, X_{22}, \dots, X_{2n1} \text{ (объём } n_2 \text{)}$$
$$\dots$$
$$X_{k1}, X_{k2}, \dots, X_{knk} \text{ (объём } n_k)$$

Общий объём: $N = n_1 + n_2 + \dots + n_k$.

Нулевая гипотеза:
H0: $\mu_1 = \mu_2 = \dots = \mu_k$

Альтернативная гипотеза:
H1: *не все средние равны.*

Дисперсии в группах неизвестны и, возможно, не равны.

Почему t-критерий нельзя применить напрямую

t-критерий Стьюдента предназначен для сравнения только двух средних. Для проверки H0 используют:

- Глобальный тест – однофакторный дисперсионный анализ (ANOVA) с F-критерием.
- Затем – попарные сравнения с помощью t-критерия (или критерия Уэлча) с обязательной поправкой на множественные сравнения.

Базовые формулы двухвыборочного t-критерия (для любой пары групп i, j)

Случай равных дисперсий (гомоскедастичность)

Объединённая выборочная дисперсия:

$$s_p^2 = \frac{(n_i - 1)s_i^2 + (n_j - 1)s_j^2}{n_i + n_j - 2}$$

где выборочные дисперсии:

$$s_i^2 = \frac{1}{n_i - 1} \sum_{p=1}^{n_i} \left(x_{ip} - \bar{x_i}\right)^2$$

$$s_j^2 = \frac{1}{n_j - 1} \sum_{q=1}^{n_j} \left(x_{jq} - \bar{x_j}\right)^2$$

Средние:

$$\bar{x_i} = \frac{1}{n_i} \sum p = 1^{n_i} x_{ip}\,, \quad \bar{x_j} = \frac{1}{n_j} \sum q = 1^{n_j} x_{jq}$$

t-статистика:

$$t_{ij} = \frac{\bar{x_i} - \bar{x_j}}{s_p \sqrt{\frac{1}{n_i} + \frac{1}{n_j}}}$$

Число степеней свободы:

$$\nu_{ij} = n_i + n_j - 2$$

Правило отвержения $H0$ (двусторонний тест, уровень α):

$$\left|t_{ij}\right| > t_{1-\alpha/2,;\nu_{ij}}$$

Случай неравных дисперсий (гетероскедастичность) – критерий Уэлча

t-статистика (без объединённой дисперсии):

$$t_{ij} = \frac{\bar{x_i} - \bar{x_j}}{\sqrt{\frac{s_i^2}{n_i} + \frac{s_j^2}{n_j}}}$$

Степени свободы по формуле Саттертуэйта:

$$\nu_{ij} \approx \frac{\left(\frac{s_i^2}{n_i} + \frac{s_j^2}{n_j}\right)^2}{\frac{(s_i^2/n_i)^2}{n_i - 1} + \frac{\left(s_j^2/n_j\right)^2}{n_j - 1}}$$

Правило отвержения:

$$\left|t_{ij}\right| > t_{1-\alpha/2,;\nu_{ij}}$$

Применение к нескольким выборкам (k > 2)

Всего попарных сравнений: $m = \frac{k(k-1)}{2}$

При многократных сравнениях возрастает вероятность ошибки первого рода (семейная ошибка FWER). Для её контроля используют поправку Бонферрони:

$$\tilde{\alpha} = \frac{\alpha}{m}$$

Тогда для каждой пары (i, j) гипотеза H0: μi = μj отвергается, если

$$\left|t_{ij}\right| > t_{1-\tilde{\alpha}/2,;\nu_{ij}}$$

Эквивалентная коррекция p-значений: $p_{ij}^{adj} = min\left(m \cdot p_{ij},; 1\right)$

Важное замечание

- Набор попарных t-тестов с поправкой Бонферрони **не заменяет глобальный ANOVA**. Сначала следует выполнить ANOVA, а затем post-hoc сравнения.
- Если условия ANOVA нарушены (неравенство дисперсий, негауссовость), допустимо сразу перейти к попарным сравнениям с поправкой.
- t-критерий (в том числе Уэлча) предполагает независимость выборок и нормальность внутри групп (при больших n – робастен).

Заключение

Для нескольких выборок с неизвестными дисперсиями проверка H0: $\mu_1 = \mu_2 = \ldots = \mu_k$ с помощью t-критерия возможна **только через попарные сравнения** с обязательной поправкой на множественность (например, Бонферрони). Формулы каждой пары – это стандартные формулы двухвыборочного t-теста (равные или неравные дисперсии). Глобальную гипотезу о равенстве всех средних рекомендуется проверять через ANOVA.

1.2. Однофакторный дисперсионный анализ (ANOVA) для проверки равенства средних нескольких выборок

Постановка задачи

Имеется k независимых выборок:

$$x_{11}, x_{12}, \ldots, x_{1n_1} \quad (1\text{-}я\ выборка, объём n_1)$$
$$x_{21}, x_{22}, \ldots, x_{2n_2} \quad (2\text{-}я\ выборка, объём n_2)$$
$$\ldots$$
$$x_{k1}, x_{k2}, \ldots, x_{kn_k} \quad (k\text{-}я\ выборка, объём n_k)$$

Общий объём всех выборок:

$$N = \sum_{i=1}^{k} n_i$$

Нулевая гипотеза: H_0: $\mu_1 = \mu_2 = \ldots = \mu_k$

Альтернативная гипотеза: H_1: не все μ_i равны (хотя бы одно отличие).

Предпосылки ANOVA:

- Выборки независимы.
- В каждой выборке данные распределены нормально.
- Дисперсии во всех группах одинаковы (гомоскедастичность).

Основные величины

Общее среднее (по всем наблюдениям):

$$\bar{x} = \frac{1}{N} \sum_{i=1}^{k} \sum_{j=1}^{n_i} x_{ij}$$

Среднее i-й группы:

$$\overline{x_i} = \frac{1}{n_i}\sum_{j=1}^{n_i} x_{ij}$$

Суммы квадратов

Межгрупповая (факторная) сумма квадратов (отражает разброс между группами):

$$SSB = \sum_{i=1}^{k} n_i(\overline{x_i} - \bar{x})^2$$

Внутригрупповая (остаточная) сумма квадратов (отражает случайный разброс внутри групп):

$$SSW = \sum_{i=1}^{k}\sum_{j=1}^{n_i}\left(x_{ij} - \overline{x_i}\right)^2$$

Общая сумма квадратов:

$$SST = \sum_{i=1}^{k}\sum_{j=1}^{n_i}\left(x_{ij} - \bar{x}\right)^2 = SSB + SSW$$

Степени свободы

$$\text{Для } SSB: \quad df_B = k - 1$$

$$\text{Для } SSW: \quad df_W = N - k$$

$$\text{Для } SST: \quad df_T = N - 1$$

Средние квадраты (дисперсии)

$$MSB = \frac{SSB}{k-1}$$

$$MSW = \frac{SSW}{N-k}$$

F-статистика

$$F = \frac{MSB}{MSW}$$

При справедливости H_{0} статистика F подчиняется распределению Фишера с (k-1) и (N-k) степенями свободы:

$$F \sim F(k-1,;N-k)$$

Правило принятия решения

На заданном уровне значимости α (например, 0.05):

- Вычисляют критическое значение $F_{\alpha}(k-1,\ N-k)$по таблице распределения Фишера.
- **Отвергают H_{0}**, если F > $F_{\alpha}(k-1,N-k)$.
- Альтернативно: если p-value $< alpha$, нулевая гипотеза отвергается.

Пример вычислений (схема)

Шаг 1: вычислить групповые средние $\overline{x_i}$ и общее среднее $\bar{x}$.

Шаг 2: найти SSB, SSW, SST.

Шаг 3: найти MSB и MSW.

Шаг 4: вычислить F = MSB/MSW.

Шаг 5: сравнить F с критическим значением.

Что делать после ANOVA

Если H_{0} отвергнута (т.е. есть различия между средними), для выявления **каких именно групп различаются** применяют **апостериорные (post-hoc) методы**:

- Критерий Тьюки (Tukey's HSD)
- Критерий Шеффе (Scheffé)
- Поправка Бонферрони для множественных сравнений (t-тесты с коррекцией)

Формулы post-hoc критериев — отдельная тема, но часто используется t-критерий с поправкой Бонферрони, описанный в предыдущем ответе.

Примечания

Если нарушена предпосылка о нормальности, применяют непараметрический аналог — критерий Краскела–Уоллиса (Kruskal-Wallis test).

Если нарушена предпосылка о равенстве дисперсий, можно использовать критерий Уэлча для ANOVA (Welch's ANOVA) или преобразования данных.

1.3. Двухфакторный дисперсионный анализ (ANOVA)

Общие обозначения

Фактор A имеет a уровней (i=1,…,a)

Фактор B имеет b уровней (j=1,…,b)

Без повторений: одно наблюдение на ячейку xijxij

С повторениями: n наблюдений на ячейку x_{ijk}, где k=1,…,n

Двухфакторный ANOVA без повторений (n = 1)

Модель и гипотезы

Модель:

$$x_{ij} = \mu + \alpha_i + \beta_j + \varepsilon_{ij}$$

Ограничения: $\frac{1}{a}\sum_{i+1}^{a} \alpha_i = 0;\ \frac{1}{b}\sum_{j+1}^{b} \beta_j = 0$

Гипотезы:

- H0A : $\alpha_1=\alpha_2=\cdots=\alpha_a=0$ (нет влияния A)
- H0B : $\beta_1=\beta_2=\cdots=\beta_b=0$ (нет влияния B)

Суммы квадратов

Общее среднее:

$$\bar{x} = \frac{1}{ab}\sum_{i=1}^{a}\sum_{j=1}^{b} x_{ij}$$

Среднее по строкам (A):

$$\overline{x_i} \cdot = \frac{1}{b}\sum_{j=1}^{b} x_{ij}$$

Среднее по столбцам (B):

$$\overline{x_j} = \frac{1}{a}\sum_{i=1}^{a} x_{ij}$$

Общая сумма квадратов:

$$SS_T = \sum_{i=1}^{a}\sum_{j=1}^{b}\left(x_{ij} - \bar{x}\right)^2$$

Сумма квадратов для A:

$$SS_A = b\sum_{i=1}^{a}(\overline{x_{i\cdot}} - \bar{x})^2$$

Сумма квадратов для B:

$$SS_B = a\sum_{j=1}^{b}\left(\overline{x_{\cdot j}} - \bar{x}\right)^2$$

Остаточная сумма квадратов:

$$SS_E = SS_T - SS_A - SS_B$$

Степени свободы и средние квадраты

$$df_T = ab - 1$$
$$df_A = a - 1$$

$$df_B = b - 1$$

$$df_E = (a-1)(b-1)$$

MSA = SSA/(a-1)

MSB = SSB/(b-1)

MSE = SSE/((a-1)(b-1))

F-статистики

$$F_A = MS_A/MS_E \quad \left(\text{сравнивается с } F_{1-\alpha}\left(a-1,;(a-1)(b-1)\right)\right)$$

$$F_B = MS_B/MS_E \quad \left(\text{сравнивается с } F_{1-\alpha}\left(b-1,;(a-1)(b-1)\right)\right)$$

1.5. Таблица ANOVA (без повторений)

Источник	SS	df	MS	F
A	SS_A	a-1	MS_A	MS_A/MS_E
B	SS_B	b-1	MS_B	MS_B/MS_E
Ошибка	SS_E	(a-1)(b-1)	MS_E	
Итого	SS_T	ab-1		

Двухфакторный ANOVA с повторениями (n > 1)

Модель и гипотезы

Модель:

$$x_{ijk} = \mu + \alpha_i + \beta_j + (\alpha\beta)ij + \varepsilon ijk$$

Ограничения:

$$\sum_i \alpha_i = 0,; \sum_j \beta_j = 0,; \sum_i (\alpha\beta)ij = 0; \forall j,; \sum_j (\alpha\beta)ij = 0; \forall i.$$

Гипотезы:

- $H0A: \alpha_1 = \cdots = \alpha_a = 0$ (нет главного эффекта A)
- $H0B: \beta_1 = \cdots = \beta_b = 0$ (нет главного эффекта B)

$$H0AB: (\alpha\beta)ij = 0 \text{ для всех } i,j \text{ (нет взаимодействия)}$$

Суммы квадратов

Общее среднее:

$$\bar{x} = \frac{1}{abn}\sum_{i=1}^{a}\sum_{j=1}^{b}\sum_{k=1}^{n} x_{ijk}$$

Среднее в ячейке (i,j):

$$\overline{x_{ij}} \cdot = \frac{1}{n}\sum_{k=1}^{n} x_{ijk}$$

Среднее по строке i:

$$\bar{x}_i = \frac{1}{bn}\sum_{j=1}^{b}\sum_{k=1}^{n} x_{ijk}$$

Среднее по столбцу j:

$$\bar{x}_j = \frac{1}{an}\sum_{i+1}^{a}\sum_{k=1}^{n} x_{ijk}$$

Общая сумма квадратов:

$$SS_T = \sum_{i=1}^{a}\sum_{j=1}^{b}\sum_{k=1}^{n}\left(x_{ijk} - \bar{x}\right)^2$$

Сумма квадратов для A:

$$SS_A = bn\sum_{i=1}^{a}(\bar{x}_i - \bar{x})^2$$

Сумма квадратов для B:

$$SS_B = an\sum_{j=1}^{b}\left(\bar{x}_j - \bar{x}\right)^2$$

Сумма квадратов для взаимодействия AB:

$$SS_{AB} = \sum_{i=1}^{a}\sum_{j=1}^{b}\left(\overline{x_{ij}} - \bar{x}_i - \bar{x}_j + \bar{x}\right)^2$$

Остаточная (внутригрупповая) сумма квадратов:

$$SS_E = \sum_{i=1}^{a}\sum_{j=1}^{b}\sum_{k=1}^{n}\left(x_{ijk} - \overline{x_{ij\cdot}}\right)^2$$

Контроль: SST=SSA+SSB+SSAB+SSESST=SSA+SSB+SSAB+SSE

Степени свободы

$$df_T = abn - 1$$
$$df_A = a - 1$$
$$df_B = b - 1$$
$$df_{AB} = (a-1)(b-1)$$
$$df_E = ab(n-1)$$

Средние квадраты и F-статистики

$$MS_A = SS_A/(a-1)$$
$$MS_B = S_B\ /\ (b-1) \backslash bigm MS_{AB} = SS_{AB}/\big((a-1)(b-1)\big)$$
$$MS_E = SS_E/\big(ab(n-1)\big)$$

$$F_A = MS_A/MS_E \quad \Big(\text{сравнивается с } F_{1-\alpha}\big(a-1,;ab(n-1)\big)\Big)$$
$$F_B = MS_B/MS_E \quad \Big(\text{сравнивается с } F_{1-\alpha}\big(b-1,;ab(n-1)\big)\Big)$$
$$F_{AB} = MS_{AB}/MS_E \quad \Big(\text{сравнивается с } F_{1-\alpha}\big((a-1)(b-1),;ab(n-1)\big)\Big)$$

Таблица ANOVA (с повторениями)

Источник	SS	df	MS	F
A	SS_A	a-1	MS_A	MS_A / MS_E
B	SS_B	b-1	MS_B	MS_B / MS_E
AB	SS_{AB}	(a-1)(b-1)	MS_{AB}	MS_{AB} / MS_E
Ошибка	SS_E	ab(n-1)	MS_E	
Итого	SS_T	abn-1		

Предположения (для обоих случаев)

- Независимость наблюдений.
- Нормальность остатков (или данных в каждой ячейке) – проверяется тестами Шапиро–Уилка, графиками Q-Q.
- Гомоскедастичность – равенство дисперсий по ячейкам (тесты Бартлетта, Левена).
- Для повторений: сбалансированный дизайн (одинаковое n) предпочтителен; при несбалансированном используют ANOVA III типа.

Примечания

- Если взаимодействие значимо (H0AB отвергнута), главные эффекты интерпретируются осторожно; рекомендуется анализ простых эффектов.
- Для post-hoc сравнений после двухфакторного ANOVA применяют критерии Тьюки, Шеффе или поправку Бонферрони отдельно для факторов и для взаимодействия.

Заключение

ANOVA — основной параметрический метод для проверки гипотезы о равенстве средних нескольких выборок. Он использует F-критерий и разложение общей дисперсии на межгрупповую и внутригрупповую составляющие. При отклонении H_{0} необходимы post-hoc сравнения.

1.4. Проверка гипотезы о равенстве двух дисперсий

1. Постановка задачи (для нескольких выборок)

Имеется k независимых выборок:

Выборка 1: $x_{11}, x_{12}, \dots, x_{1n_1}$ $(n_1$ наблюдений)
Выборка 2: $x_{21}, x_{22}, \dots, x_{2n_2}$ $(n_2$ наблюдений)
…

Выборка k: $x_{k1}, x_{k2}, \dots, x_{kn_k}$ $(n_k$ *наблюдений*)

Обозначим генеральные дисперсии как $\sigma_1^2, \sigma_2^2, \dots, \sigma_k^2$.

Нулевая гипотеза (гомоскедастичность, равенство дисперсий):

$$H_0: \sigma_1^2 = \sigma_2^2 = \cdots = \sigma_k^2$$

Альтернативная гипотеза: H_1: не все дисперсии равны (хотя бы одна отличается).

Для двух выборок (F-тест Фишера)

Даны две независимые нормальные выборки объёмами *n_{1}, n_{2}.*

Выборочные дисперсии (несмещённые):

$$s_1^2 = \frac{1}{n_1 - 1}\sum_{i=1}^{n_1}(x_{1i} - \bar{x}1)^2,$$

$$s_2^2 = \frac{1}{n_2 - 1}\sum_{j=1}^{n_2}\left(x_{2j} - \overline{x_2}\right)^2$$

Гипотеза $H_0: \sigma_1^2 = \sigma_2^2$.

F-статистика (обычно бóльшая дисперсия в числителе):

$$F = \frac{s_1^2}{s_2^2} \quad (где\, s_1^2 \geq s_2^2)$$

Число степеней свободы:

$$\nu_1 = n_1 - 1, \quad \nu_2 = n_2 - 1$$

При справедливости H0 статистика F подчиняется распределению Фишера:

$$F \sim F(\nu_1, \nu_2)$$

Правило отвержения H0 (двусторонний тест на уровне \alpha):

$$F > F_{1-\alpha/2,;\nu_1,\nu_2} \quad или \quad F < F_{\alpha/2,;\nu_1,\nu_2}$$

На практике чаще используют односторонний тест, взяв отношение большей дисперсии к меньшей:

$$F_{calc} = \frac{max(s_1^2, s_2^2)}{min(s_1^2, s_2^2)}$$

и сравнивают с $F_{1-\alpha,;\,\nu_{max},\nu_{min}}$.

Рекомендации по выбору критерия

Условие	Рекомендуемый критерий
Две выборки, нормальность	F-тест Фишера
Несколько выборок, нормальность, любые n	Критерий Бартлетта
Несколько выборок, возможна ненормальность	Критерий Левена (медианный)
Сбалансированные малые выборки	Критерий Кохрана или Хартли

Заключение

Выбор критерия равенства дисперсий зависит от числа групп, сбалансированности, предположения о нормальности.

- Для двух выборок – F-тест.
- Для нескольких при нормальности – Бартлетт.
- При нарушении нормальности – Левен (предпочтительнее).

При равных объёмах – также можно использовать Кохрана или Хартли.

1.5. Проверка гипотезы о нормальности распределении по критерию согласия

Критерий согласия Пирсона (проверка гипотезы о распределении)

Пусть имеется выборка объёмом n. Гипотеза H0: распределение генеральной совокупности имеет некоторый теоретический закон (равномерный, нормальный, Пуассона и т.д.).

Диапазон значений разбит на m интервалов (или категорий).

Наблюдаемые частоты O1,O2,…,Om,

Теоретические вероятности попадания в интервалы (при H0): $p_1,p_2,\ldots,p_m$,

Теоретические (ожидаемые) частоты: $E_i = np_i$.

Статистика критерия:

$$\chi^2 = \sum_{i=1}^{m} \frac{(O_i - E_i)^2}{E_i}$$

При справедливости H0 и n→∞ статистика $\chi 2$ подчиняется распределению хи-квадрат с числом степеней свободы:

$$\nu = m - 1 - r$$

где r – число параметров теоретического распределения, оценённых по выборке.

Правило: H0 отвергается на уровне α, если $\chi^2 > \chi^2_{1-\alpha}(\nu)$.

Критерий хи-квадрат для проверки нормальности распределения

Постановка задачи

Имеется выборка x1,x2,…,xn.

Гипотеза H0: генеральная совокупность распределена нормально $N(\mu,\sigma 2)$.

Параметры μ и σ^2 **неизвестны** и оцениваются по выборке.

Оценка параметров нормального распределения

Выборочное среднее:

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

Выборочная дисперсия (несмещённая):

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

При больших n иногда используют смещённую оценку

$$\sigma^2 = \frac{1}{n}\sum(xi - x^-)^2$$

Построение интервалов (разрядов)

Диапазон значений выборки разбивают на m интервалов.

Границы интервалов: a_0, a_1, a_2,…, a_m, где

$a_0 = -\infty$ (или min (x_i)−ε), $a_m = +\infty$ (или max(x_i)+ε).

Для каждого интервала j=1,…,m:

Наблюдаемая частота Oj – количество выборочных значений, попавших в (a_{j-1}, a_j]

Теоретические вероятности при нормальном распределении

При H0: $X \sim N(\mu, \sigma 2)$ с $\mu = x^-, \sigma = s$.

Вероятность попадания в jj-й интервал:

$$p_j = \Phi\left(\frac{a_j - \bar{x}}{s}\right) - \Phi\left(\frac{a_{j-1} - \bar{x}}{s}\right)$$

где Φ(z) – функция стандартного нормального распределения (интеграл вероятности):

$$\Phi(z) = \int_{-\infty}^{z} \frac{1}{\sqrt{2\pi}} e^{-t^2/2} dt$$

Ожидаемые (теоретические) частоты

$$E_j = n \cdot p_j$$

Для корректности критерия требуется, чтобы в каждом интервале $E_j \geq 5$ (иногда допускается $E_j \geq 1$, но с объединением мелких интервалов).

Статистика критерия

$$\chi^2 = \sum_{j=1}^{m} \frac{(O_j - E_j)^2}{E_j}$$

Степени свободы

При проверке нормальности два параметра (μ и σ) оцениваются по выборке. Поэтому число степеней свободы:

$$\nu = m - 1 - 2 = m - 3$$

Общая формула: $\nu = m - 1 - r$, где r – число оценённых параметров. Для нормального закона r=2.

Правило принятия решения

На уровне значимости α:

- Вычисляют $\chi 2$ по выборке.
- Находят критическое значение $\chi 2_{1-\alpha}(\nu)$.
- Если $\chi 2 > \chi 2_{1-\alpha}(m-3)$, гипотеза о нормальности отвергается.

Пример явного вычисления p_j (через z-преобразование)

Пусть границы интервалов: a_{j-1}, a.j

Стандартизованные границы:

$$z_{j-1} = \frac{a_{j-1} - \bar{x}}{s}, \quad z_j = \frac{a_j - \bar{x}}{s}$$

Тогда:

$$p_j = \Phi(z_j) - \Phi(z_{j-1})$$

Для крайних интервалов: $\Phi(-\infty) = 0$, $\Phi(+\infty) = 1$.

Критерий независимости (таблица сопряжённости r×c)

Дана таблица сопряжённости двух признаков:

- Признак A имеет r категорий (i=1,…,r)
- Признак B имеет c категорий (j=1,…,c)

Наблюдаемые частоты: Oij.

Общее число наблюдений: $n = \sum_{i=1}^{r} \sum_{j=1}^{c} Oij$

Гипотеза H0: признаки A и B независимы.

Ожидаемые частоты при независимости:

$$E_{ij} = \frac{\left(\sum_{j=1}^{c} O_{ij}\right) \cdot \left(\sum_{i=1}^{r} O_{ij}\right)}{n} = \frac{n_{i\cdot} \cdot n_{\cdot j}}{n}$$

Статистика:

$$\chi^2 = \sum_{i=1}^{r} \sum_{j=1}^{c} \frac{\left(O_{ij} - E_{ij}\right)^2}{E_{ij}}$$

Степени свободы:

$$\nu = (r-1)(c-1)$$

Правило: H0 отвергается, если $\chi^2 > \chi^2_{1-\alpha}((r-1)(c-1))$.

Проверка гипотезы о дисперсии нормальной совокупности (χ^2-критерий для одной дисперсии)

Дана нормальная выборка $x_1, x_2, \ldots, x_n$.

Выборочная дисперсия (несмещённая):

$$s^2 = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})^2$$

Гипотеза H0: $\sigma^2 = \sigma_0^2$ (дисперсия равна заданному числу).
Альтернатива может быть двусторонней или односторонней.

Статистика:

$$\chi^2 = \frac{(n-1)s^2}{\sigma_0^2}$$

При H0 статистика подчиняется распределению χ^2 с $\nu=n-1$ степенями свободы.

Правила отвержения H0 на уровне α:

Двусторонняя: $\chi^2 < \chi^2_{\alpha/2,\ n-1}$ или $\chi^2 > \chi^2_{1-\alpha/2,\ n-1}$

Левосторонняя ($\sigma^2 < \sigma_0^2$): $\chi^2 < \chi^2_{\alpha,\ n-1}$

Правосторонняя ($\sigma^2 > \sigma_0^2$): $\chi^2 > \chi^2_{1-\alpha,\ n-1}$

Сравнение с предыдущими темами: связь с критерием Бартлетта

Критерий Бартлетта для проверки равенства дисперсий нескольких выборок использует статистику, которая асимптотически распределена как $\chi 2(k-1)$. Это уже было описано ранее. Формула:

$$T = \frac{(N-k)\ln s_p^2 - \sum_{i=1}^{k}(n_i-1)\ln s_i^2}{1 + \frac{1}{3(k-1)}\left(\sum_{i=1}^{k}\frac{1}{n_i-1} - \frac{1}{N-k}\right)} \sim \chi^2(k-1)$$

Заключение

Критерий χ^2 является универсальным непараметрическим методом для работы с категориальными данными (частоты) и для проверки гипотез о распределении, независимости, однородности. Для нормальных выборок он также применяется для проверки гипотез о дисперсии.

Порядок выполнения работы:

Выборки для лабораторных работ размещены а СДО Moodle. Берем для работы выборку с номером, соответствующим номеру по списку группы. Все вычисления, построения и отчеты выполняем в Python

Для каждого из двух рядов отдельно:

1. Рассчитать числовые характеристики выборок. Построить гистограммы.

2. Составить таблицу числовых характеристик каждой выборки: размах, среднее, дисперсия, среднеквадратичное отклонение, медиана.

3. По размаху и объему выборки рассчитать количество интервалов вариационного ряда (см. Лекцию 2), границы и середины интервалов.

4. Составить вариационный ряд, подсчитав количество данных, попадающих в тот или иной интервал.

5. На основе вариационного ряда построить гистограмму.

6. По двум выборкам своего варианта построить поле рассеяния данных (облако точек на плоскости)

7. Определить доверительный интервал для математического ожидания.

8. Определить доверительный интервал для дисперсии генеральной совокупности.

9. Проверить гипотезу о равенстве средних значений двух рядов.

10. Проверить гипотезу о равенстве дисперсий двух частей ряда.

11. Рассчитать коэффициент корреляции и проверить его значимость по критериям Стьюдента и Фишера-Йетса.

12. Построить интервальные вариационные ряды и гистограммы для рядов значений.

13. Проверить гипотезы о нормальном распределении данных датасета по критерию Пирсона. (см лекцию о проверке гипотез или ознакомьтесь со статьей: https://sky.pro/wiki/python/primery-rascheta-kriteriya-pirsona-na-python/)

14. Результат – график нормального распределения отобразить на гистограмме.

Требования к отчету:

Отчет должен содержать текстовый файл кода на Python с оглавлением сегментов и подробными комментариями ко всем строкам кода.

Результаты расчетов представляются с комментариями («Коэффициент корреляции выборки равен: … Значимость по Стьюденту; …; по Фишеру: …»).

Таблицы, поле данных и гистограммы оформляются также инструментарием Python.

Ссылки на краткое описание необходимых библиотек Python:

https://changellenge-education.com/blog/top-bibliotek-analitiki-python

https://practicum.yandex.ru/blog/biblioteki-python-dlya-data-science/

https://blog.skillfactory.ru/15-bibliotek-python-dlya-data-science/

Глава 2. Подготовка данных

Введение

Подготовка данных – это критически важный этап в разработке систем искусственного интеллекта (ИИ). Этот процесс включает в себя сбор, очистку, трансформацию и организацию данных, чтобы они были готовы для анализа и обучения моделей машинного обучения. Правильная подготовка данных может значительно улучшить качество и эффективность алгоритмов ИИ.

Важность подготовки данных

Подготовка данных — это процесс сбора, обработки и преобразования данных, который необходим для создания качественных моделей искусственного интеллекта. От качества подготовки данных зависит эффективность и точность работы систем ИИ. Неправильная подготовка данных может привести к искажению результатов анализа и принятия решений, а также к снижению производительности систем ИИ.

Этапы подготовки данных

Процесс подготовки данных включает в себя несколько этапов:

- **Сбор данных.** На этом этапе происходит поиск и сбор данных из различных источников, таких как базы данных, веб-сайты, социальные сети, открытые дата-сеты, данные из баз данных, API сервисов, данные из

Интернет вещей (IoT), скрипты для парсинга веб-страниц и другие. Важно убедиться в качестве и достоверности собранных данных.

- **Очистка данных.** Этот этап включает в себя удаление или исправление ошибок (например, опечаток, неверных значений), пропусков и аномалий в данных, удаление дубликатов записей, заполнение пропущенных значений, а также анализ и обработка выбросов. Удаление ненужных или нерелевантных столбцов и др. Очистка данных помогает улучшить их качество и точность.

- **Преобразование данных.** Данные часто находятся в формате, не подходящем для анализа, поэтому их нужно преобразовать. Преобразование данных включает в себя изменение формата, структуры и типа данных для соответствия требованиям модели ИИ. Например, данные могут быть преобразованы из текстового формата в числовой формат, изменение формата данных (например, преобразование дат из строк в datetime), нормализация или стандартизация числовых признаков, кодирование категориальных переменных (например, one-hot encoding, label encoding), Создание новых признаков, на основе существующих (фиче инжиниринг).

- **Интеграция данных.** Интеграция данных объединяет данные из разных источников в один набор данных. Это позволяет создать более полный и точный набор данных для обучения модели ИИ.

- **Выбор признаков.** Выбор признаков — это процесс отбора наиболее важных и информативных признаков из набора данных для использования в модели ИИ. Это помогает уменьшить размерность данных и улучшить производительность модели.

- **Разделение данных.** Разделение данных на обучающую, валидационную и тестовую выборки позволяет оценить производительность модели и предотвратить переобучение.

- **Анализ и визуализация данных.** Перед тем как переходить к моделированию, полезно провести исследовательский анализ данных (EDA).

Это может включать: Построение различных графиков для визуализации распределений и взаимосвязей, Определение аномалий или выбросов, Оценку корреляций между переменными.

- **Деление данных.** Для обучения и тестирования модели необходимо разделить данные на несколько наборов: Обучающая выборка (training set), Тестовая выборка (test set), Валидационная выборка (validation set) — по желанию, для тонкой настройки модели.

- **Сохранение подготовленных данных.** После завершения всех этапов подготовки данных, рекомендуется сохранить их в подходящем формате (CSV, Parquet, базы данных и т.д.), чтобы упростить дальнейшую работу и повторное использование.

- **Документация и автоматизация**. Хорошей практикой является документирование проведенных этапов подготовки данных, включая описания источников, методов очистки и трансформации. Также полезно автоматизировать процесс подготовки данных с помощью скриптов или пайплайнов, чтобы упростить дальнейшие итерации работы с новыми данными.

Правильная подготовка данных является ключом к созданию эффективных и точных моделей, поэтому ей следует уделить должное внимание.

Пример на Phyton:

```
# Загрузка данных из CSV файла
data = pd.read_csv('data.csv')
```

Мы загружаем данные из CSV файла в объект DataFrame.

Просмотр первых строк данных

```
print(data.head())
```

С помощью head() можно быстро просмотреть первые несколько строк, чтобы понять структуру и содержание данных.

Методы подготовки данных

Существует множество методов подготовки данных, которые можно использовать в зависимости от типа и объёма данных, а также от требований модели ИИ. Некоторые из наиболее распространённых методов включают в себя:

- Заполнение пропущенных значений. Пропущенные значения могут быть заполнены на основе других доступных данных или с использованием специальных методов, таких как среднее значение, медиана или режим.
- Удаление выбросов. Выбросы — это значения, которые сильно отличаются от остальных значений в наборе данных. Они могут искажать результаты анализа и принятия решений. Выбросы могут быть удалены или обработаны специальными методами.
- Нормализация данных. Нормализация — это метод, который преобразует данные в стандартный диапазон значений, например, от 0 до 1. Нормализация помогает улучшить сходимость и стабильность алгоритмов машинного обучения.
- Стандартизация данных. Стандартизация — это метод, который масштабирует данные так, чтобы они имели нулевое среднее значение и единичную дисперсию. Стандартизация помогает ускорить сходимость алгоритмов машинного обучения и улучшить интерпретируемость результатов.
- Кодирование категориальных данных. Категориальные данные — это данные, которые представляют собой категории или группы. Кодирование категориальных данных позволяет использовать их в моделях машинного обучения. Существует несколько методов кодирования категориальных данных, таких как one-hot кодирование, label encoding и binary encoding.

Подготовка данных является важным этапом в разработке систем искусственного интеллекта. Правильная подготовка данных может

существенно повысить эффективность и точность работы систем ИИ. Мы рассмотрели основные аспекты подготовки данных, такие как важность, этапы и методы подготовки данных. Эти аспекты необходимо учитывать при работе с данными для систем ИИ.

Лакуны в данных

Лакуны в данных (или пропуски) – это отсутствующие значения в наборе данных. Проблема лакун может возникнуть по разным причинам, например, из-за ошибок в сборе данных, технических сбоев или неверного ввода информации. Лакуны могут существенно повлиять на качество работы систем ИИ, так как они могут привести к искажению результатов анализа и принятия решений.

Рассмотрим более подробно причины и факторы их появления:

1. **Технические сбои**: системные ошибки или сбои в процессе сбора данных.

2. *Пример*: В процессе опроса онлайн пользователь не завершил анкету, и его ответы на вопросы пропали.

3. **Человеческий фактор**: ошибки в заполнении данных, неверные значения или опущенные поля.

4. *Пример*: Оператор неправильно ввел данные о температуре, забыв указать показатели для одного из дней.

5. **Отказ от ответа**: респонденты могут отказываться отвечать на некоторые вопросы.

6. *Пример*: В социологическом исследовании один из вопросов касается личных финансов, и респонденты могут избегать ответа на него.

7. **Утрата данных**: данные могли быть потеряны или повреждены.

8. *Пример*: В результате сбоя системы хранения информация о результатах эксперимента может стать недоступной.

9. **Лимитированное время сбора данных**: данных может не хватать, если исследования проводятся в короткие сроки или сезонно.

10. *Пример*: Агентство по охране окружающей среды проводит мониторинг уровня загрязнения, но не охватывает все временные промежутки.

Для устранения лакун можно использовать следующие методы:

Импутация: заполнение пропущенных значений: можно попытаться заполнить пропущенные значения на основе других доступных данных. Например, если в наборе данных есть информация о возрасте и поле человека, то можно предположить, что средний возраст людей определённого пола может быть использован для заполнения пропущенных значений возраста. Заполнение пропусков в свою очередь использует различные методы, такие как: среднее, медиана или мода для числовых данных; восстановление на основе похожих данных (k-ближайших соседей).

Удаление записей: удаление строк с пропущенными значениями: если количество пропущенных значений слишком велико, можно удалить строки с этими значениями из набора данных. Однако это может привести к потере полезной информации.

Использование методов машинного обучения для обработки пропущенных значений: существуют методы машинного обучения, которые могут автоматически обрабатывать пропущенные значения. Например, алгоритмы, основанные на деревьях решений, могут учитывать пропущенные значения при построении модели.

Создание новых признаков: Замена пропусков специальным значением, например, меткой "недоступно".

Пример на Phyton:

```
# Проверка наличия пропусков
print(data.isnull().sum())

# Удаление строк с пропусками
data_cleaned = data.dropna()

# ИЛИ
```

```
# Заполнение пропусков средним значением
data['column_name'].fillna(data['column_name'].mean(), inplace=True)
```

Мы проверяем наличие пропусков и можем либо удалить строки с пропусками, либо заменить их, например, средним значением.

```
# Удаление дублирующихся строк
data_cleaned = data_cleaned.drop_duplicates()
```

Удаляем дубликаты для получения уникальных значений.

Пример использования

Предположим, мы проводим исследование по влиянию условий жизни на здоровье, и у нас есть таблица (Таблица 1.) с данными о участниках. Некие поля могут быть пропущены:

Таблица 1. Исследование по влиянию условий жизни на здоровье

ID	Возраст	Уровень дохода	Заболевания	Город
1	30	50000	1	Москва
2	25		0	СПб
3	40	60000	1	
4		45000	0	Уфа
5	35	70000		Казань

В этом случае:

- Участник 2 не указал уровень дохода.
- У участника 3 отсутствует информация о городе.
- Участник 5 не указал информацию о заболеваниях.

Для анализа данных можно использовать методы импутации для заполнения пропусков, например, среднего уровня дохода в данной выборке, или же удалить участников с пропущенными данными, если это допустимо в рамках исследования.

Таким образом, лакуны данных — это важный аспект работы с данными, требующий внимательного подхода к их управлению и анализу.

2.1. Методы обнаружения выбросов

Визуальный анализ

Самый быстрый способ — посмотреть на графики.

- **Ящик с усами (boxplot)**

Показывает медиану, квартили (Q1, Q3) и «усы»: обычно в пределах Q1 - 1.5*IQR и Q3 + 1.5*IQR. Точки за усами считаются выбросами.

- **Гистограмма** – выбросы видны как отдельные «хвосты» далеко от пика.
- **Scatter plot** (для двух признаков) – точки, изолированные от облака.

python
```
import matplotlib.pyplot as plt
import seaborn as sns

sns.boxplot(data=df['price'])
plt.show()
```

Z-оценка (стандартизация)

Предполагает нормальное распределение. Для каждого значения x:

$$Z = \frac{x - \mu}{\sigma}$$

Если $|Z| > 3$ (иногда порог 2.5 или 4), значение считается выбросом. Для не очень больших выборок порог 3 означает, что за пределы попадает ~0.27% данных.

Плюс: просто, быстро.

Минус: работает плохо при сильной асимметрии или нескольких выбросах (они искажают μ и σ).

python
```
from scipy import stats

z_scores = np.abs(stats.zscore(df['price']))
outliers = df[z_scores > 3]
```

IQR (межквартильный размах)

Робастный метод, не зависит от распределения.

IQR = Q3 – Q1 (75-й перцентиль минус 25-й).

Границы:

нижняя=Q1−1.5×IQR; верхняя=Q3+1.5×IQR

Точки вне этих границ – выбросы. Множитель 1.5 – стандартный; для «экстремальных» выбросов иногда берут 3.0.

Почему 1.5? Эмпирическое правило, которое хорошо работает для многих распределений, включая умеренно скошенные.

python

```
Q1 = df['price'].quantile(0.25)
Q3 = df['price'].quantile(0.75)
IQR = Q3 - Q1
lower = Q1 - 1.5 * IQR
upper = Q3 + 1.5 * IQR

outliers = df[(df['price'] < lower) | (df['price'] > upper)]
```

Дополнительные методы (кратко)

Модифицированная Z-оценка – использует медиану и MAD (median absolute deviation). Робастна к выбросам.

DBSCAN – кластеризация, где точки, не вошедшие ни в один кластер, считаются выбросами.

Isolation Forest – специальный алгоритм

Для начального практикума достаточно IQR и визуализации.

Что делать с выбросами?

Удаление (trimming / dropping)

Просто отбросить строки, содержащие выбросы.

Когда хорошо:

- выбросы – явные ошибки (отрицательная зарплата, возраст 200 лет);
- их мало (<5% данных);
- вы уверены, что они не несут полезной информации.

Когда плохо:

- теряется много данных;

- выбросы отражают реальные редкие события (например, миллионные покупки в интернет-магазине).

python
```
clean_df = df[(df['price'] >= lower) & (df['price'] <= upper)]
```

Трансформация (log, sqrt, Box-Cox)

Сильно скошенные распределения с длинными хвостами можно «прижать» с помощью преобразований. Выбросы перестают быть экстремальными.

Логарифмирование – для положительных данных с большим разбросом (цены, доходы).

`df['log_price'] = np.log1p(df['price'])` (log1p для нулей).

Квадратный корень – при умеренной скошенности.

Box-Cox – подбирает оптимальный параметр λ. Требует положительных значений.

После трансформации можно не удалять выбросы – они становятся менее влиятельными. Но если модель чувствительна к масштабу (например, линейная), всё равно стоит масштабировать.

python
```
from scipy.stats import boxcox
df['price_boxcox'], _ = boxcox(df['price'] + 1e-6)
```

Винсоризация (Winsorization)

Замена экстремальных значений на заданные перцентили. Например, все значения выше 95-го перцентиля заменяются на значение 95-го перцентиля, а ниже 5-го – на 5-й перцентиль. Выбросы не удаляются, но ограничиваются.

Плюс – сохраняется объём выборки, уменьшается влияние хвостов. **Минус** – искажает распределение, может скрыть действительно важные аномалии.

python
```
from scipy.stats.mstats import winsorize

df['price_winsor'] = winsorize(df['price'], limits=[0.05, 0.05])  # 
ограничить 5% снизу и сверху
```

Отдельное кодирование (capping)

Иногда выбросы заменяют на специальное значение (например, `np.nan` или очень большую константу) и добавляют бинарный признак «был ли выброс». Это позволяет модели самой решить, как на него реагировать.

Практические рекомендации

- **Не удаляйте выбросы автоматически** – сначала посмотрите на них. Может быть, это не ошибка, а важная часть закономерности.
- Для **деревьев решений** и **случайного леса** выбросы не критичны (деревья режут пространство по порогам), но могут быть проблемой для бустинга (например, XGBoost).
- Для **линейных моделей** и **kNN** – масштабирование обязательное, выбросы лучше обработать (винсоризация или удаление).
- **Всегда сравнивайте** качество модели до и после обработки выбросов. Используйте кросс-валидацию.

Пример сквозного пайплайна (с выбором действия)

python

```
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import FunctionTransformer

def winsorize_series(s, limits=(0.05, 0.05)):
    return winsorize(s, limits=limits)

def iqr_outlier_cap(s, factor=1.5):
    Q1 = s.quantile(0.25)
    Q3 = s.quantile(0.75)
    IQR = Q3 - Q1
    lower = Q1 - factor * IQR
    upper = Q3 + factor * IQR
    return s.clip(lower, upper)

# Использование в пайплайне
outlier_transformer = FunctionTransformer(iqr_outlier_cap, validate=False)

pipeline = Pipeline([
    ('cap_outliers', outlier_transformer),
    ('imputer', SimpleImputer(strategy='median')),
    ('scaler', StandardScaler())
])
```

Синтетический датасет — Для фокуса на чистых методах

Если вы хотите полностью сконцентрироваться на изучении методов обработки, не отвлекаясь на сложности реальных данных, создайте свой собственный набор.

Как это поможет: Вы будете точно знать, сколько выбросов и где вы добавили, и сможете идеально отследить, как каждый метод (IQR, удаление, винсоризация) на них влияет.

Пример кода: Для его создания можно сгенерировать 1000 случайных чисел, а затем добавить несколько экстремальных значений.

python

```
import pandas as pd
import numpy as np
# Установите сид для воспроизводимости результата
np.random.seed(42)
# Генерируем 1000 точек из нормального распределения
data = pd.DataFrame({ 'value': np.random.normal(0, 1, 1000) })
# Пример детекции выбросов методом IQR
def detect_outliers_iqr(data):
    Q1 = data.quantile(0.25)
    Q3 = data.quantile(0.75)
    IQR = Q3 - Q1
    # Границы
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    return (data < lower_bound) | (data > upper_bound)
outliers = detect_outliers_iqr(data['value'])
print(f"Количество обнаруженных выбросов: {sum(outliers)}")
# Вывод: Количество обнаруженных выбросов: 8
```

Задание для практикума (часть про выбросы)

1. Выберите числовой признак в ваших данных. Постройте boxplot и гистограмму.

2. Найдите выбросы методом IQR. Посмотрите, сколько их и какие это значения.

3. Примените три способа: удаление, логарифмирование, винсоризацию (5%).

4. Обучите простую линейную модель (LinearRegression или LogisticRegression) на исходных данных и на каждом из обработанных вариантов. Сравните метрики на тестовой выборке (MSE / MAE / Accuracy).

5. Напишите вывод: какой метод лучше для вашего набора и почему.

2.2. Корреляция

Корреляция - это статистическая связь между двумя переменными. В контексте искажения данных важно понимать, какие признаки коррелируют между собой, так как это может привести к проблемам при обучении моделей. Обычно для оценки корреляции используются коэффициент корреляции Пирсона (для линейной корреляции) или коэффициент корреляции Спирмена (для ранговых данных).

Например, в задаче прогнозирования продаж корреляция между объёмом продаж и ценой товара может показать, что увеличение цены может привести к снижению объёма продаж. Это может помочь менеджерам по продажам оптимизировать ценовую политику.

Однако корреляция не всегда означает причинно-следственную связь. Например, корреляция между количеством осадков и продажами зонтов может быть вызвана общими факторами, такими как погода. Поэтому важно понимать, что корреляция является лишь инструментом для анализа данных и не должна использоваться для принятия решений без дополнительного анализа.

Основные типы корреляции включают:

- Положительная корреляция: обе переменные увеличиваются вместе.
- Отрицательная корреляция: одна переменная увеличивается, в то время как другая уменьшается.
- Нулевая корреляция: нет взаимосвязи между переменными.

Проверка корреляции:

- Коэффициент корреляции Пирсона: используется для измерения линейной зависимости между двумя числовыми признаками.
- Матрица корреляции: позволяет визуально оценить корреляции между всеми признаками в датасете.

Пример использования: Положительная корреляция

Предположим, у нас есть два набора данных (Таблица 2.):

- Время, проведенное на подготовку к экзамену (в часах)

- Оценка на экзамене (по 100-балльной шкале)

Таблица 2. Положительная корреляция

Время подготовки (часы)	Оценка
1	60
2	70
3	75
4	85
5	90

В этом случае можно ожидать положительную корреляцию между временем подготовки и оценкой — чем больше времени вы тратите на подготовку, тем выше ваша оценка.

Коэффициент корреляции Пирсона для этих данных будет близок к 1 (максимальная положительная корреляция).

Пример использования: Отрицательная корреляция

Рассмотрим ситуацию (Таблица 3.) с двумя другими переменными:

- Температура воздуха (в градусах Цельсия)
- Количество потребляемого тепла (в киловаттах)

Таблица 3. Отрицательная корреляция

Температура (°C)	Потребление тепла (кВт)
0	100
5	80
10	60
15	40
20	20

В данном случае есть отрицательная корреляция между температурой и потреблением тепла. Чем выше температура, тем меньше потребление тепла.

Коэффициент корреляции для этих данных будет близок к -1 (максимальная отрицательная корреляция).

Пример использования: Отсутствие корреляции

Предположим, у нас есть данные о количестве часов, которые студенты проводят на играх в видеоигры, и их весе.

Таблица 4. Отсутствие корреляции

Часы игр (в неделю)	Вес (кг)
0	70
5	60
10	75
15	80
20	55

В этом примере (Таблица 4.) нет очевидной зависимости между часами, проведенными за играми, и весом. Можно ожидать, что коэффициент корреляции будет близок к 0.

Визуализация корреляции

Чтобы лучше понять корреляцию между переменными, можно использовать диаграмму рассеяния (scatter plot):

- **Положительная корреляция**: точки на графике имеют восходящий тренд.
- **Отрицательная корреляция**: точки имеют нисходящий тренд.
- **Отсутствие корреляции**: точки распределены хаотично без явного тренда.

Пример использования

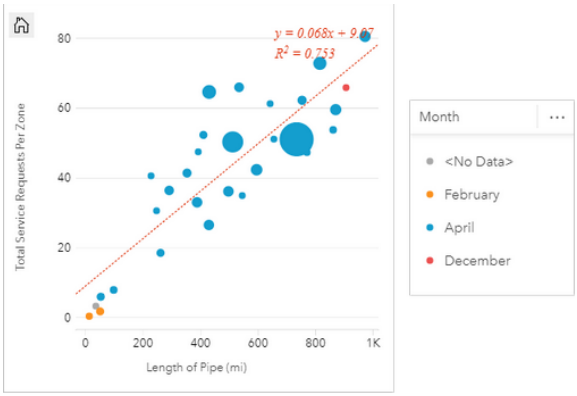

Рисунок 1. Scatter plot

Корреляция — это полезный инструмент для понимания взаимосвязей между переменными, хотя важно помнить, что корреляция не означает причинно-следственную связь. Например, высокая корреляция между

потреблением мороженого и количеством утоплений может быть следствием того, что оба этих явления увеличиваются в летний период, но это не значит, что потребление мороженого вызывает утопления.

2.3. Мультиколлинеарность

Ситуация, когда два или более независимых признаков имеют высокую корреляцию, может вызвать проблемы в модели, такие как сложность интерпретации коэффициентов и снижение предсказательной способности.

ИЛИ

Мультиколлинеарность — это явление, которое возникает в регрессионном анализе, когда независимые переменные сильно коррелируют друг с другом. Это может привести к нестабильности оценок коэффициентов регрессии и затруднить интерпретацию результатов.

Примеры мультиколлинеарности

Пример 1: Экономические показатели

Предположим, вы хотите построить модель для предсказания зарплаты (зависимая переменная) на основе нескольких независимых переменных, таких как:

- Образование (количество лет обучения)
- Опыт работы (количество лет на трудовом рынке)
- Возраст

При этом «возраст» и «опыт работы» могут быть высоко коррелированы, поскольку, как правило, более опытные работники старше по возрасту. Это создает проблему мультиколлинеарности.

Пример 2: Оценка качества жилья

Допустим, вы хотите предсказать цену жилья на основе следующих признаков:

- Площадь дома
- Количество комнат
- Количество ванных комнат

Если у вас есть множество характеристик, которые связаны друг с другом (например, площадь дома и количество комнат), это также может вызвать мультиколлинеарность.

Последствия мультиколлинеарности

- **Неустойчивость коэффициентов:** Изменения в данных могут привести к значительным изменениям в оценках коэффициентов регрессии, что затрудняет интерпретацию.
- **Увеличение стандартных ошибок:** в результате этого некоторые независимые переменные могут оказаться незначительными, даже если они имеют реальное влияние на зависимую переменную.
- **Сложности в интерпретации:** при наличии мультиколлинеарности становится трудно оценить, какой из коррелирующих факторов действительно влияет на результат.

Диагностика мультиколлинеарности

Существуют различные методы для диагностики мультиколлинеарности:

- Коэффициент корреляции: Высокий коэффициент корреляции (близкий к 1 или -1) между независимыми переменными может указывать на мультиколлинеарность.
- Вариантный инфляционный фактор (VIF): Это мера, которая количественно определяет, насколько увеличивается дисперсия коэффициента регрессии из-за мультиколлинеарности. Обычно, если VIF > 10, это считается признаком проблемы мультиколлинеарности.
- Матрица корреляции: Анализ корреляционной матрицы между независимыми переменными может помочь выявить высокие корреляции.

Методы борьбы с мультиколлинеарностью:

- Удаление одного из коррелирующих признаков. Если две или более переменные сильно коррелируют, можно рассмотреть возможность удаления одной из них из модели.

- Использование метода главных компонент (PCA) для снижения размерности.
- Объединение переменных: Иногда можно создать новую переменную (например, индекс), которая будет представлять собой комбинацию нескольких коррелирующих переменных.
- Стандартизация переменных: Приведение переменных к одному масштабу может помочь в уменьшении проблемы.

Мультиколлинеарность — это важный аспект, который нужно учитывать при проведении регрессионного анализа. Правильное понимание и диагностика этого явления позволяет более точно моделировать взаимосвязи между переменными и делать более надежные выводы из данных.

2.4. Избыточность данных

Избыточность возникает, когда в наборе данных дублируются записи или признаки. Избыточные данные могут нарушить обучающие процессы и привести к переобучению.

Избыточность — это наличие повторяющихся или ненужных данных в наборе. Она может возникнуть из-за ошибок при сборе или обработке данных, а также из-за использования различных источников данных. Избыточные данные могут замедлить работу систем ИИ и снизить их эффективность.

Примеры избыточности данных:

1. **Реляционные базы данных**:

Предположим, у вас есть таблица клиентов и таблица заказов. Если вы храните имя и адрес клиента в обеих таблицах, это приводит к избыточности. Лучше будет создать одну таблицу клиентов, где будет храниться уникальная информация о каждом клиенте, и ссылаться на нее из таблицы заказов с помощью идентификатора клиента.

   - Таблица Клиенты:

- ID | Имя | Адрес
| Иван Иванов| ул. Ленина, д. 1

- Таблица Заказы:
- ID | ID_Клиента | Товар 1 | 1 | Товар А

Файловые системы:

Если документы (например, договоры или отчеты) хранятся на нескольких серверах или разделах диска, это создаёт избыточность. Например, один и тот же отчет может быть сохранен на сервере А, сервере Б и на локальном компьютере пользователя. Изменение информации в одном файле, но не в других, может привести к несоответствию данных.

Электронные таблицы:

Если несколько сотрудников ведут отдельные версии одной и той же таблицы, содержащей данные о продажах, это тоже форма избыточности. Например, отдел продаж и отдел маркетинга могут вести разные таблицы с данными о клиентах и заказах, но изменения в одной таблице не отражаются в другой.

Системы управления контентом (CMS):

В веб-приложениях информация о пользователях, их заказах и комментариях может храниться в разных частях системы. Если для каждого комментария повторяется информация о пользователе, например, его имя и почта, это может вызвать затруднения при обновлении данных. Можно создать отдельную таблицу с пользователями и связывать комментарии с пользователями через уникальные идентификаторы.

Проблемы, связанные с избыточностью данных:

Трудности в синхронизации: Обновление информации в одном месте, но не в другом, приводит к несоответствию.

Повышенные затраты на хранение: Хранение дублирующихся данных увеличивает объем необходимого пространства и, следовательно, стоимость.

Сложности в анализе данных: При анализе данных из различных источников может быть трудно объединить информацию, что усложняет получение точных отчетов.

Увеличение вероятности ошибок: Если данные дублируются, существует риск ошибки в одном из мест хранения.

Чтобы уменьшить избыточность, можно использовать следующие методы:

- **Объединение данных из разных источников:** если данные собраны из разных источников, можно объединить их в один набор данных. Это позволит избежать дублирования информации.
- **Фильтрация данных:** можно отфильтровать данные, чтобы удалить ненужные или повторяющиеся записи. Например, можно удалить записи, которые содержат только дублирующую информацию.
- **Нормализация данных: нормализация** — это процесс преобразования данных в стандартный формат. Нормализация может помочь уменьшить избыточность, так как она позволяет избежать дублирования данных.
- **Удаление дубликатов:** идентификация и удаление повторяющихся записей в наборе данных.
- **Отбор признаков:** использование методов статистической и машинной обработки для выявления наиболее значимых признаков и удаления избыточных.
- **Использование методов выделения признаков** (feature selection), чтобы оставить только наиболее информативные переменные.

Подготовка данных требует использования различных инструментов и технологий в зависимости от объема данных и специфики задач:

- **Python**: наиболее популярный язык программирования для анализа данных, с мощными библиотеками, такими как Pandas для

манипуляции данными, NumPy для числовых операций и Scikit-learn для машинного обучения.

- **R**: широко используется для статистического анализа и визуализации данных, с множеством пакетов для обработки данных.
- **Apache Spark:** подходит для обработки больших объемов данных в распределенных системах. Spark SQL и DataFrames позволяют эффективно манипулировать данными. ETL-инструменты: Talend, Apache NiFi и другие инструменты используются для извлечения, трансформации и загрузки данных.
- **SQL:** основной язык для работы с реляционными базами данных, позволяющий выполнять сложные запросы и манипуляции с данными.

2.5. Кодирование категориальных признаков

Кодирование категориальных признаков — это процесс преобразования категориальных (качественных) данных в числовой формат, который может быть использован в машинном обучении. Большинство алгоритмов машинного обучения работают только с числовыми данными, поэтому категориальные признаки необходимо преобразовать. Рассмотрим основные методы кодирования с примерами.

Для преобразования категориальных данных в числовые **используется методы:**

One-hot encoding: Создание бинарных признаков для каждой категории. подходит для номинальных данных с небольшим количеством категорий. Каждая категория преобразуется в бинарный вектор, где только один элемент равен 1, а остальные — 0. Этот метод подходит для номинальных данных (где категории не имеют порядка).

Пример: Данные:
Цвет: ['Красный', 'Синий', 'Зеленый', 'Синий', 'Красный']
После One-Hot Encoding:
Красный: [1, 0, 0]
Синий: [0, 1, 0]

Зеленый: [0, 0, 1]
Результат:
Цвет_Красный: [1, 0, 0, 0, 1]
Цвет_Синий: [0, 1, 0, 1, 0]
Цвет_Зеленый: [0, 0, 1, 0, 0]

Проблема: при большом количестве категорий размерность данных значительно увеличивается (проклятие размерности).

Label encoding и Ordinal Encoding: Присвоение уникальных чисел каждой категории, что может быть полезно для порядковых данных. подходят для порядковых данных.

Label Encoding (Метка кодирования)

Каждой уникальной категории присваивается уникальное числовое значение. Этот метод подходит для порядковых данных (где категории имеют естественный порядок).

Пример: Данные:
Цвет: ['Красный', 'Синий', 'Зеленый', 'Синий', 'Красный']
После Label Encoding:
Цвет: [0, 1, 2, 1, 0]

Проблема: Метод может ввести ложную порядковую зависимость, если категории не имеют естественного порядка (например, "Красный" не больше и не меньше "Синего").

Ordinal Encoding (Порядковое кодирование). Категории преобразуются в числа с учетом их порядка. Этот метод подходит для порядковых данных.

Пример: Данные:
Размер: ['S', 'M', 'L', 'XL', 'M']
После Ordinal Encoding:
Размер: [0, 1, 2, 3, 1]

Проблема: Как и в Label Encoding, может возникнуть ложная порядковая зависимость, если порядок категорий не определен.

Target Encoding (Кодирование на основе целевой переменной). Категории заменяются средним значением целевой переменной для каждой категории. Этот метод полезен для задач классификации и регрессии.

Пример:
Данные:
Город: ['Москва', 'Санкт-Петербург', 'Москва', 'Казань', 'Санкт-Петербург']
Целевая переменная (Цена): [100, 200, 150, 300, 250]
После Target Encoding:
Город: [125, 225, 125, 300, 225]
(125 — среднее для Москвы, 225 — для Санкт-Петербурга, 300 — для Казани)
Проблема: Может привести к переобучению, если категории имеют мало примеров.

Frequency Encoding (Частотное кодирование) Категории заменяются их частотой в данных.

Пример: Данные:
Город: ['Москва', 'Санкт-Петербург', 'Москва', 'Казань', 'Санкт-Петербург']
Частоты:
Москва: 2/5 = 0.4
Санкт-Петербург: 2/5 = 0.4
Казань: 1/5 = 0.2
После Frequency Encoding:
Город: [0.4, 0.4, 0.4, 0.2, 0.4]
Проблема: Может потерять информацию о важности категорий.

Binary Encoding (Бинарное кодирование). Категории сначала преобразуются в числа с помощью Label Encoding, а затем эти числа кодируются в двоичный формат. Каждый бит становится отдельным признаком.

Пример: Данные:
Цвет: ['Красный', 'Синий', 'Зеленый', 'Синий', 'Красный']
После Label Encoding:
Цвет: [0, 1, 2, 1, 0]
После Binary Encoding:
Цвет_Бит1: [0, 1, 1, 1, 0]
Цвет_Бит2: [0, 0, 1, 0, 0]
Преимущество: уменьшает размерность по сравнению с One-Hot Encoding.

Hashing Encoding (Хеширование). Категории преобразуются в числовые значения с помощью хеш-функции. Этот метод полезен при большом количестве категорий.

Пример: Данные:
Город: ['Москва', 'Санкт-Петербург', 'Москва', 'Казань', 'Санкт-Петербург']
После Hashing Encoding (с использованием хеш-функции):
Город: [123, 456, 123, 789, 456]
Проблема: Возможны коллизии (разные категории могут получить одинаковый хеш).
Каждый метод имеет свои преимущества и недостатки, поэтому выбор зависит от конкретной задачи и данных.

Пример на Phyton:

```
# Преобразуем категориальные переменные в числовые
data = pd.get_dummies(data, columns=['categorical_column'], drop_first=True)
```

Используем get_dummies для преобразования категориальной переменной в несколько бинарных (0/1) переменных.

Глава 3. Метод главных компонент (Principal Component Analysis, PCA)

Метод главных компонент (PCA — Principal Component Analysis) — это статистический метод, используемый для снижения размерности данных при минимальной потере информации Это подход, который позволяет упростить модели без значительной потери информации. Он выделяет основные компоненты, которые максимально сохраняют вариацию в данных. PCA находит линейные комбинации исходных переменных, которые максимизируют дисперсию, представляя при этом данные в новом пространстве с меньшим числом измерений.

PCA преобразует исходные коррелированные переменные в новый набор некоррелированных переменных, называемых **главными компонентами**. Эти компоненты упорядочиваются по убыванию дисперсии (разброса), что позволяет сосредоточиться на первых нескольких компонентах, которые содержат большую часть информации.

Метод главных компонент (PCA) — это метод, который позволяет вычислить главные компоненты для набора данных. PCA может быть полезен в следующих случаях:

- Уменьшение размерности: PCA может уменьшить размерность данных, удаляя ненужные признаки. Это может ускорить работу систем ИИ и уменьшить их сложность.
- Улучшение качества данных: PCA может улучшить качество данных, устраняя шум и выбросы. Это может повысить точность и надёжность систем ИИ.

Поскольку PCA является одним из методов, можно говорить и о других **подходах к анализу размерности:**

- Линейный дискриминантный анализ (LDA): фокусируется на разделении классов.
- t-SNE (t-distributed Stochastic Neighbor Embedding): не линейный метод для визуализации высокоразмерных данных, эффективный для изучения кластеров.
- UMAP (Uniform Manifold Approximation and Projection): еще один метод для визуализации данных, который сохраняет как локальные, так и глобальные структуры.

3.1. Процесс или основные этапы метода PCA:

1. Стандартизация данных: приведение признаков к единому масштабу. Прежде чем применять PCA, данные часто стандартизируют, чтобы привести их к единой шкале. Это особенно важно, если переменные измеряются в разных единицах.

2. Вычисление матрицы ковариации: описание, как изменяются данные. Матрица показывает, как переменные изменяются совместно.

3. Вычисление собственных векторов и значений: определение направлений максимальной дисперсии. Находим собственные значения и

собственные векторы ковариационной матрицы. Собственные значения отражают важность соответствующих собственных векторов (компонент).

4. Выбор главных компонент: определение числа компонент, которое будет использовано для построения снижения размерности. Выбираем главные компоненты с наибольшими собственными значениями. Обычно выбирается k компонентов, которые объясняют максимальную часть дисперсии данных.

5. Проекция данных: проецируем исходные данные на выбранные главные компоненты, получая новый набор данных с меньшей размерностью.

Визуализация результатов

После применения PCA можно визуализировать результаты в 2D или 3D графике, если это применимо, что помогает увидеть, как данные сгруппированы и как главные компоненты описывают их распределение.

Применение PCA

- **Сжатие данных**: Упрощение данных для хранения или передачи, без значительной потери информации.
- **Визуализация**: Уменьшение размерности для визуализации многомерных данных на 2D или 3D графиках.
- **Шумоподавление**: Удаление избыточной информации, что может помочь в дальнейшем анализе.
- **Предобработка для моделей машинного обучения**: Упрощение входных данных для алгоритмов, чтобы они работали быстрее и эффективнее.

Метод главных компонент широко используется в различных областях, включая обработку изображений, анализ данных, финансовую аналитику и многие другие.

3.2. Нормализация и стандартизация

Нормализация и стандартизация — это методы предобработки данных, которые используются для приведения данных к единому масштабу. Это

особенно важно для алгоритмов машинного обучения, которые чувствительны к масштабу данных, таких как методы, основанные на расстояниях (например, k-ближайших соседей, метод опорных векторов) или методы, использующие градиентный спуск (например, нейронные сети). Эти две процедуры часто путают, хотя они решают разные задачи.

1. Стандартизация (Z-score normalization)

Цель: привести данные к распределению с **нулевым средним** и **единичной дисперсией** (стандартным отклонением = 1).

Когда использовать: когда признаки имеют разные единицы измерения или масштабы, особенно перед PCA, линейной регрессией, SVM, методами, чувствительными к масштабу.

Формула для каждого значения xij (где i — наблюдение, j — признак):

$$\boldsymbol{z_{ij}} = \frac{x_{ij} - \mu_j}{\sigma_j}$$

где:

$\mu_j = \frac{1}{n}\sum_{i=1}^{n} x_{ij}$ - среднее арифметическое признака j,

$\sigma_j = \frac{1}{n}\sum_{i=1}^{n} (x_{ij} - \mu_j)^2$ — среднеквадратическое отклонение признака j (иногда используют деление на n−1 для несмещённой оценки, но в стандартизации это не принципиально).

Результат: преобразованные данные имеют среднее 0 и стандартное отклонение.

Особенность: Устойчивее к выбросам, чем нормализация.

2. Нормализация (min-max scaling)

Цель: линейно сжать данные в заданный интервал, чаще всего [0,1].

Когда использовать: когда нужно сохранить относительные расстояния между точками, но убрать влияние разных диапазонов (например, для нейронных сетей, методов, требующих ограниченного входа, или для данных с известными границами).

Формула для каждого значения:

$$x'_{ij} = \frac{x_{ij} - min_j}{max_j - min_j}$$

где

- $min_j = min\{x_{1j}, x_{2j}, \ldots, x_{nj}\}$ — минимальное значение в признаке j,
- $max_j = max\{x_{1j}, x_{2j}, \ldots, x_{nj}\}$ — максимальное значение.

Результат: все значения признака j лежат в интервале [0,1]. Если нужен другой интервал [a,b], формула обобщается:

$$x'_{ij} = a + \frac{(x_{ij} - min_j)(b-a)}{max_j - min_j}$$

Особенность: Очень чувствительна к выбросам.

3. Другие виды нормализации (не путать!)

Иногда термином «нормализация» называют ещё:

Нормировка вектора (L2-нормализация): для каждого наблюдения (строки) делят все его признаки на длину вектора:

$$x_{ij}' = \frac{x_{ij}}{\sqrt{\sum_{k=1}^{p} x_{ik}^2}}$$

Используется в текстовом анализе (TF-IDF) или для приведения строк к единичной длине.

Масштабирование к единичной дисперсии (без центрирования):
$x_{ij}' = x_{ij} / \sigma_j$ — сохраняется ненулевое среднее.

Но в контексте обработки данных под «нормализацией» чаще всего понимают min-max scaling, а под «стандартизацией» — z-score.

Сравнение

Характеристика	Стандартизация (Z-score)	Нормализация (min-max)
Формула	(x−μ)/σ	$x'_{ij} = \frac{x_{ij} - min_j}{max_j - min_j}$
Результат	Среднее = 0, ст.откл. = 1	Диапазон [0,1] (или [a,b])
Чувствительность к выбросам	Да (выбросы влияют на μ,σ)	Очень сильная (выбросы сжимают основную массу)
Когда лучше	PCA, линейная регрессия, кластеризация (k-means), SVM	Нейронные сети, методы, требующие границ входа, изображения

Пример в Python

```
python
from sklearn.preprocessing import StandardScaler, MinMaxScaler
import numpy as np

X = np.array([[1, 2, 3],
              [4, 5, 6],
              [7, 8, 9]])

# Стандартизация
scaler_std = StandardScaler()
X_std = scaler_std.fit_transform(X)
print("Стандартизация:\n", X_std)
print("Среднее:", X_std.mean(axis=0), "Ст.откл.:", X_std.std(axis=0))

# Нормализация [0,1]
scaler_mm = MinMaxScaler()
X_mm = scaler_mm.fit_transform(X)
print("Нормализация [0,1]:\n", X_mm)
print("Min:", X_mm.min(axis=0), "Max:", X_mm.max(axis=0))
```

Обратите внимание: обе процедуры применяются **независимо к каждому столбцу** (кроме L2-нормировки строки).

Пример на Phyton:

```
from sklearn.preprocessing import StandardScaler

# Нормализация данных
scaler = StandardScaler()
```

```
data[['feature1', 'feature2']] = scaler.fit_transform(data[['feature1', 'feature2']])
```

Здесь мы применяем стандартизацию для того, чтобы все значения находились в одном масштабе, что важно для большинства алгоритмов машинного обучения.

Когда использовать нормализацию, а когда стандартизацию?

Нормализация полезна, когда данные имеют ограниченный диапазон (например, изображения, где значения пикселей находятся в диапазоне [0, 255]).

Стандартизация полезна, когда данные имеют нормальное распределение или когда важно сохранить информацию о дисперсии и среднем значении.

Таким образом, нормализация и стандартизация помогают привести данные к единому масштабу, что улучшает работу многих алгоритмов машинного обучения.

Суть метода

PCA находит ортогональные линейные преобразования исходных признаков в новый набор некоррелированных переменных — **главных компонент**. Первая компонента направлена вдоль оси максимальной дисперсии данных, вторая — перпендикулярно первой и объясняет следующую по величине дисперсию и т.д.

Математически:

Для центрированной матрицы данных X (размером n×p) строится ковариационная матрица $C = \frac{1}{n-1} X^T X$. Её собственные векторы — это оси главных компонент, а собственные значения пропорциональны доле объяснённой дисперсии. Проекция данных на первые k компонент даёт наилучшее приближение исходного множества в пространстве размерности k (по критерию минимума среднеквадратичной ошибки).

3.3. Критерии (условия) применимости PCA

1. **Линейность отношений между признаками**

PCA улавливает только линейные корреляции. Для данных со сложными нелинейными зависимостями нужны нелинейные варианты (например, kernel PCA).

2. **Масштабирование признаков**

Признаки должны быть приведены к сопоставимому масштабу. Если один признак имеет большую дисперсию из-за единиц измерения, он искусственно доминирует над остальными. **Стандартизация** (центрирование + деление на среднеквадратичное отклонение) обязательна, если признаки измерены в разных единицах.

3. **Отсутствие сильных выбросов**

Выбросы сильно искажают оценки ковариационной матрицы и направления максимальной дисперсии. Перед применением PCA рекомендуется очистка данных (или использование робастных версий).

4. **Достаточный объём выборки**

Для устойчивой оценки ковариационной матрицы необходимо n>pn>p (число наблюдений больше числа исходных признаков). На практике часто требуют n≥5pn≥5p или даже n≥10pn≥10p. При малом nn велика вероятность переобучения.

5. **Нормальность распределения (не строго обязательно)**

PCA не требует нормальности для нахождения компонент, но для статистических выводов (например, доверительных интервалов) и интерпретации может потребоваться многомерное нормальное распределение.

3.4. Критерии выбора числа главных компонент

Это тоже часть «критериев применения» — как решить, сколько компонент оставить.

Правило «каменистой осыпи» (Cattell) – строят график собственных значений в порядке убывания и ищут точку излома (резкое замедление падения). Компоненты до излома оставляют.

Критерий Кайзера – оставляют компоненты с собственным значением больше 1 (актуально для стандартизованных данных, т.к. сумма собственных значений равна pp).

Кумулятивная доля объяснённой дисперсии – фиксированный порог, например 80%, 90% или 95% суммарной дисперсии.

Информационные критерии (AIC, BIC) – реже, но применимы в вероятностной постановке PCA (Probabilistic PCA).

Дополнительные ограничения

Неоднозначность знаков – компоненты определены с точностью до умножения на −1, что не мешает анализу, но требует осторожности при сравнении.

Интерпретируемость – главные компоненты редко имеют простую физическую интерпретацию (каждая является линейной комбинацией всех исходных признаков).

Пример, когда PCA не подходит

Данные имеют кластерную структуру, но внутри кластеров дисперсия мала — PCA может спутать структуру.

Признаки независимы и одинаково распределены — PCA не даст выигрыша (все компоненты объясняют примерно равную долю дисперсии).

Если вы имели в виду не PCA, а другой метод (например, PCoA или факторный анализ), уточните, пожалуйста.

3.5. Суть метода главных компонент (PCA) простыми словами

Представьте, что вы сфотографировали трёхмерный объект (например, яблоко), а потом хотите посмотреть на него на плоском экране монитора. Вы поворачиваете объект так, чтобы на снимок попало как можно больше

деталей, а то, что спряталось сзади, можно было немного потерять. PCA делает то же самое с данными: он **поворачивает** облако точек в пространстве признаков, чтобы найти **самую информативную плоскость** (или прямую), и проецирует на неё все точки, теряя минимум информации.

Ещё проще:

У вас есть таблица с кучей столбцов (признаков). Вы хотите оставить всего 2–3 столбца, но так, чтобы эти новые столбцы максимально точно пересказывали все исходные. PCA создаёт эти «сжатые» столбцы (главные компоненты) из смеси исходных. Первая главная компонента — это то направление, вдоль которого данные разбросаны сильнее всего (как длинная ось огурца). Вторая — перпендикулярно первой и объясняет следующий по величине разброс, и так далее.

Почему это работает?

Если данные сильно разбросаны вдоль какого-то направления, значит, вдоль этого направления они **очень разные**, и эта разница несёт много информации. А если по другому направлению точки почти не меняются, то это направление можно выбросить без большого ущерба.

Математический «скелет» (для понимания, но без формул)

1. **Центрируем** данные — вычитаем среднее по каждому признаку, чтобы облако оказалось в начале координат.
2. **Находим главные оси** — это направления, в которых данные сильнее всего растянуты. Они ищутся через собственные векторы ковариационной матрицы.
3. **Проецируем** данные на первые k осей — получаем новое, более компактное представление.

Важно: перед PCA нужно **стандартизовать** признаки (сделать по ним среднее 0 и дисперсию 1), если они измерены в разных единицах (например, рубли, возраст, сантиметры). Иначе тот признак, у которого числа больше (например, рубли), случайно станет главным.

Простой алгоритм PCA на Python

1. С помощью библиотеки scikit-learn (рекомендуется)

python
```
import numpy as np
import pandas as pd
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt

# Пример данных: 4 признака, 100 наблюдений
np.random.seed(42)
X = np.random.rand(100, 4) * 10

# 1. Нормируем (центрируем + масштабируем)
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# 2. Применяем PCA, оставляем 2 компоненты
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)

# 3. Смотрим на результат
print("Доля объяснённой дисперсии по компонентам:", pca.explained_variance_ratio_)
print("Суммарная доля:", sum(pca.explained_variance_ratio_))

# 4. Визуализация
plt.scatter(X_pca[:, 0], X_pca[:, 1])
plt.xlabel('Первая главная компонента')
plt.ylabel('Вторая главная компонента')
plt.title('Проекция данных на 2 главные компоненты')
plt.show()
```

2. «Ручная» реализация PCA (для понимания шагов)

python
```
import numpy as np

def manual_pca(X, n_components):
    # Шаг 1: центрирование (вычитаем среднее по каждому признаку)
    X_centered = X - np.mean(X, axis=0)

    # Шаг 2: ковариационная матрица
    cov_matrix = np.cov(X_centered, rowvar=False)
```

```
    # Шаг 3: собственные значения и собственные векторы
    eigenvalues, eigenvectors = np.linalg.eig(cov_matrix)

    # Шаг 4: сортировка по убыванию собственных значений
    idx = np.argsort(eigenvalues)[::-1]
    eigenvectors = eigenvectors[:, idx]

    # Шаг 5: выбираем первые n_components векторов
    components = eigenvectors[:, :n_components]

    # Шаг 6: проецируем данные
    X_pca = X_centered @ components  # матричное умножение

    # Доля объяснённой дисперсии
    explained_variance = eigenvalues[idx] / np.sum(eigenvalues)
    explained_variance_ratio = explained_variance[:n_components]

    return X_pca, explained_variance_ratio

# Проверка на тех же данных
X_manual, var_manual = manual_pca(X_scaled, n_components=2)
print("Доля объяснённой дисперсии (ручной метод):", var_manual)
```

3.6. Что важно помнить при использовании PCA?

PCA не подходит, если данные имеют сложную нелинейную структуру (например, точки на сфере или в форме подковы). Тогда используют kernel PCA или t-SNE.

Интерпретация новых признаков («главных компонент») часто сложна — они являются смесью исходных. Нельзя сказать, что компонента 1 = «возраст», она может быть `0.3*возраст + 0.5*доход - 0.2*рост`.

PCA можно использовать для:

- сжатия данных (ускорение алгоритмов машинного обучения);
- визуализации многомерных данных (первые 2–3 компоненты);
- шумоподавления (отбрасывая компоненты с малой дисперсией);
- устранения мультиколлинеарности в регрессии.

Пример из жизни

Вы исследуете автомобили: есть 50 признаков (мощность, расход топлива, количество дверей, цвет, ...). Вам трудно строить графики. PCA оставляет 2 компоненты: первая связана с «размером и мощностью», вторая — с «экономичностью». Вы рисуете точки, и видите, что BMW и Mercedes улетают вправо, а Toyota Prius — вверх. Это и есть польза.

PCA **не выбирает** существующие столбцы (исходные признаки) с максимальной дисперсией. Вместо этого он **создаёт новые столбцы** (главные компоненты), каждый из которых является **линейной комбинацией** всех исходных признаков. Эти новые столбцы упорядочены так, чтобы первый имел **максимально возможную дисперсию среди всех линейных комбинаций**, второй — следующую по величине дисперсию при условии некоррелированности с первым, и т.д.

Почему не просто «выбрать столбец с большой дисперсией»?

Допустим, у вас два признака: `x1` = рост в см, `x2` = вес в кг. Дисперсия роста (например, ±20 см) может быть численно больше дисперсии веса (±10 кг) просто из-за единиц измерения. Если просто взять столбец `x1`, мы потеряем информацию о весе. Но PCA может найти такое направление (например, `0.7*рост + 0.7*вес`), вдоль которого разброс данных окажется больше, чем вдоль любого из исходных столбцов в отдельности. Это и есть первая главная компонента. Она «смешивает» оба признака, улавливая их совместную изменчивость.

Пример на пальцах

Представьте облако точек, вытянутое по диагонали (рост и вес коррелируют).

Дисперсия вдоль оси `x1` (рост) — средняя.

Дисперсия вдоль оси `x2` (вес) — тоже средняя.

А вдоль диагонали (первая главная компонента) дисперсия **максимальна**.

Вдоль перпендикулярной диагонали (вторая компонента) дисперсия минимальна.

Если бы мы просто взяли столбец с максимальной дисперсией (`x1`), мы бы потеряли информацию о совместном поведении признаков.

Когда PCA совпал бы с выбором столбца?

Только в очень искусственном случае: если исходные признаки уже некоррелированы и их дисперсии сильно различаются. Тогда первая главная компонента будет почти совпадать с тем исходным признаком, у которого дисперсия максимальна. Но это редкое исключение.

Отличие от других методов

- **Фильтрация по дисперсии** (VarianceThreshold в sklearn) — действительно удаляет столбцы с дисперсией ниже порога. Это не PCA.
- **Выбор признаков по важности** (например, с помощью случайного леса) — тоже не PCA.
- **PCA** — это **извлечение признаков** (feature extraction), а не их отбор (feature selection). Он создаёт новые признаки.

Резюме

«Главными» в PCA являются **направления в пространстве признаков**, вдоль которых данные изменяются сильнее всего. Эти направления почти никогда не совпадают с осями исходных столбцов. Сам метод **не выбирает лучшие столбцы**, а строит лучшие **линейные комбинации** всех столбцов.

3.7. Разделение данных на подвыборки

При построении модели мы хотим не просто обучить её на имеющихся данных, но и оценить, как она будет работать на **новых, невидимых данных**. Для этого исходный набор данных делится на части.

Основные подвыборки:

Обучающая выборка (train set) — на ней модель непосредственно учится: подбирает веса, коэффициенты, правила.

Валидационная выборка (validation set) — используется для настройки гиперпараметров (например, степени полинома, количества деревьев в лесу) и выбора лучшей модели. На ней мы проверяем промежуточную производительность.

Тестовая выборка (test set) — абсолютно неприкосновенна до самого конца проекта. Её используют **один раз** для финальной оценки качества готовой модели. Если вы используете тестовую выборку многократно, вы подстраиваетесь под неё, и оценка перестаёт быть честной.

Типичное соотношение: часто используют пропорции 60–80% на обучение, а остальное на валидацию и тест (например, 60/20/20 или 70/15/15). Если данных мало, можно обойтись только обучающей и тестовой (80/20).

Простое разделение (train/test split): случайным образом перемешиваем данные и отрезаем нужный процент. Это быстро, но результат может зависеть от «удачности» разбиения (особенно если данные имеют сложную структуру или малочисленны).

Стратифицированное разбиение: Этот метод сохраняет пропорции классов в выборках, что особенно важно для несбалансированных данных. В sklearn это можно реализовать, указав параметр stratify в функции train_test_split().

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

Кросс-валидация (Cross-validation)

Кросс-валидация решает проблему случайности одного разбиения и позволяет эффективнее использовать ограниченные данные. Её суть — многократно разбивать данные на обучающие и валидационные части разными способами.

Самый популярный метод — K-fold кросс-валидация:

1. Вся выборка (без тестовой!) случайно делится на **K** равных частей (fold).

2. Для каждой итерации `i = 1..K`:

3. Валидационной частью становится `i`-й блок.

4. Обучающей — все остальные `K-1` блоков.

5. Обучаем модель на обучающих блоках, оцениваем качество на валидационном блоке.

6. В итоге получаем **K значений** метрики качества (например, точности, RMSE).

7. **Итоговая оценка модели** = среднее арифметическое этих K значений. Дополнительно можно вычислить стандартное отклонение — оно покажет, насколько стабильна модель при смене данных.

Распространённые варианты:

- **Stratified K-fold** — для задач классификации с несбалансированными классами: сохраняет пропорции классов в каждом блоке.
- **Leave-One-Out (LOO)** — частный случай, где K = N (числу объектов). Очень ресурсоёмко, но полезно для малых выборок.
- **Time Series Split** — для временных рядов: обучаем только на прошлых данных, валидируем на будущих (без случайного перемешивания, чтобы не нарушить порядок).

Зачем это нужно? Кросс-валидация даёт более устойчивую, менее смещённую оценку качества, чем одно разбиение. Она помогает обнаружить переобучение (если разброс оценок велик) и выбрать гиперпараметры без многократного использования тестовой выборки.

3. Вычисление статистических характеристик

После того как мы провели кросс-валидацию (или несколько раз разбили данные), у нас есть **выборка значений метрики качества**

(например, [0.81, 0.79, 0.83, 0.80, 0.82]). По этой выборке можно вычислить статистики, которые более информативны, чем одно среднее.

Основные характеристики:

Среднее (mean) — точечная оценка «истинного» качества модели.

Стандартное отклонение (std) — мера разброса. Высокое std означает, что модель сильно чувствительна к составу обучающих данных — это сигнал о нестабильности или малом объёме выборки.

Доверительный интервал (confidence interval) — интервал, в котором с заданной вероятностью (обычно 95%) находится истинное качество модели. Рассчитывается как `среднее ± (t * std / sqrt(K))`, где `t` — коэффициент Стьюдента. Доверительный интервал наглядно показывает, насколько мы уверены в оценке.

Пример использования:

Вы обучили две модели. У модели A средняя точность 0.85, доверительный интервал [0.82, 0.88]. У модели B — 0.84 [0.75, 0.93]. Несмотря на более низкое среднее у B, интервалы перекрываются — значит, разница может быть случайной. На практике выбирают модель с более высоким средним **и** узким доверительным интервалом.

Дополнительные статистики: медиана, минимум/максимум (показывают худший и лучший сценарии), межквартильный размах (робастная мера разброса).

Связь всех трёх частей в рабочем процессе

1. Откладываем **тестовую выборку** (не трогаем до конца).
2. На оставшихся данных применяем **K-fold кросс-валидацию** для подбора гиперпараметров и выбора лучшей модели.
3. По результатам K-fold вычисляем **статистические характеристики** (среднее, доверительный интервал) метрики качества каждой кандидатной модели.
4. Выбираем модель с лучшим сочетанием среднего и стабильности.

5. **Один раз** оцениваем выбранную модель на отложенной тестовой выборке и получаем финальную, самую честную оценку.

Такой подход минимизирует переобучение, даёт объективное представление о качестве и помогает принимать обоснованные решения.

Вот полный пример кода на Python, который демонстрирует разделение данных, кросс-валидацию и расчёт статистических характеристик. Используем классический набор данных Iris и модель логистической регрессии.

python

```
import numpy as np
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split, cross_val_score, KFold
from sklearn.linear_model import LogisticRegression
from scipy import stats

# 1. Загрузка данных
X, y = load_iris(return_X_y=True)

# 2. Разделение на обучающую (80%) и тестовую (20%) выборки
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y  # stratify сохраняет
пропорции классов
)

# 3. Создание модели
model = LogisticRegression(max_iter=200, random_state=42)

# 4. K-fold кросс-валидация на обучающей выборке (без теста!)
k = 5
kf = KFold(n_splits=k, shuffle=True, random_state=42)

# Получаем массив из k значений метрики (accuracy)
scores = cross_val_score(model, X_train, y_train, cv=kf, scoring='accuracy')

# 5. Статистические характеристики результатов кросс-валидации
mean_score = np.mean(scores)
std_score = np.std(scores, ddof=1)  # выборочное стандартное отклонение
n = len(scores)
# Доверительный интервал 95% (используем распределение Стьюдента)
confidence_interval = stats.t.interval(
    confidence=0.95,
    df=n-1,
    loc=mean_score,
    scale=std_score / np.sqrt(n)
)

# Вывод статистики
print("=== Результаты 5-fold кросс-валидации ===")
print(f"Оценки по фолдам: {np.round(scores, 3)}")
print(f"Средняя точность: {mean_score:.4f}")
print(f"Стандартное отклонение: {std_score:.4f}")
```

```
print(f"95% доверительный интервал: [{confidence_interval[0]:.4f},
{confidence_interval[1]:.4f}]")

# 6. Финальная оценка на тестовой выборке (один раз)
model.fit(X_train, y_train)
test_accuracy = model.score(X_test, y_test)
print(f"\n=== Финальная оценка на тестовой выборке ===")
print(f"Точность на тесте: {test_accuracy:.4f}")

# Дополнительно: сравнение с доверительным интервалом
if confidence_interval[0] <= test_accuracy <= confidence_interval[1]:
    print("Тестовая точность лежит внутри доверительного интервала кросс-
валидации — хороший признак.")
else:
    print("Тестовая точность выходит за пределы доверительного интервала —
возможно, переобучение или неудачное разбиение.")
```

Что происходит в коде:

1. Разделение данных — train_test_split выделяет 20% под тест, остальное под обучение (внутри которого будем делать кросс-валидацию).

2. Кросс-валидация — cross_val_score с KFold(5) вычисляет точность на 5 различных разбиениях обучающей выборки.

3. Статистика — среднее, стандартное отклонение и 95% доверительный интервал (по формуле для малых выборок, используя распределение Стьюдента).

4. Финальная проверка — модель переобучается на всей обучающей выборке и тестируется на отложенном тесте. Сравнение тестовой точности с доверительным интервалом помогает оценить обобщающую способность.

Возможный вывод (числа могут немного отличаться из-за случайности):

```
=== Результаты 5-fold кросс-валидации ===
Оценки по фолдам: [0.958 0.958 0.958 0.917 1.   ]
Средняя точность: 0.9583
Стандартное отклонение: 0.0298
95% доверительный интервал: [0.9236, 0.9930]

=== Финальная оценка на тестовой выборке ===
Точность на тесте: 0.9667
Тестовая точность лежит внутри доверительного интервала кросс-валидации
— хороший признак.
```

Примечание: для более сложных моделей (градиентный бустинг, нейросети) принцип остаётся тем же — используйте `cross_val_score` или реализуйте K-fold цикл, чтобы собирать не только точность, но и любые другие метрики (F1, ROC-AUC и т.д.).

Важно

Тестовая выборка должна обеспечить объективную оценку качества модели (регрессии, деревьев решений, нейросети и пр.). Поэтому оценка статистических характеристик должна выполняться только на оставшейся части выборки, точнее, на тренировочной части. В случае применения кросс-валидации, такая оценка выполняется на тренировочной выборке до разбиения ее на К частей, либо в каждую итерацию разбиения на тестовую и валидационную подвыборки с исключением очередного валидационного набора,

Разделение данных на обучающую и тестовую выборки — это критически важный шаг в процессе машинного обучения. Оно позволяет создавать модели, которые могут обобщать и давать предсказания на новых данных, что является основной целью многих задач в этой области.

НЕЙРОСЕТИ И МАШИННОЕ ОБУЧЕНИЕ

Часть 1. ПОДГОТОВКА ДАННЫХ

ПРАКТИКУМ

Авторы
Т.А.Нигматулин, А.Ю. Сидоренко, Е.С. Новожилова

Издано в авторской редакции

Подписано в печать 01.03.2026 г.
Формат 60×84/16.
Усл. печ. л. 4,5. Тираж – Электронное издание

СПб. ООО «Свое издательство», 191040, Санкт-Петербург, Пушкинская ул, д. 10 литера А, помещ. 1-н
2026