РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ
Факультет/институт: Факультет ИИ/ РУДН
Кафедра: Прикладная информатика

ОТЧЁТ О ЛАБОРАТОРНОЙ РАБОТЕ

по дисциплине «Прикладная статистика и анализ данных»

Лабораторная работа № 1

Тема: Аудит набора данных и экспресс-EDA для многомерных выборок. Детектирование выбросов и пропусков, сравнение критериев. Проектирование конвейера препроцессинга и документация артефактов.

Студент(ка):	Перевозчикова Валерия Дмитриевна, ЗФИмд-01-25, управление данными и искусственный интеллект
Преподаватель:	Курашкин Сергей Олегович, Доцент
Дата выполнения:	«25» октября 2025 г.
Оценка/подпись:	_____

Москва — 2025

СОДЕРЖАНИЕ

Введение

Актуальность работы обусловлена возрастающей сложностью и объемом данных в современных прикладных задачах. Качество исходных данных напрямую влияет на надежность статистических выводов и эффективность моделей машинного обучения. Умение проводить корректный разведочный анализ и выявлять аномалии становится критически важным навыком для специалистов по анализу данных.

Цель работы: освоить методы разведочного анализа многомерных данных, диагностики распределений и аномалий, а также проектирования воспроизводимых конвейеров предобработки данных.

Основные задачи:

- Провести корректный EDA для многомерных таблиц с использованием устойчивых сводок
- Освоить диагностику формы распределений и выявление аномалий
- Построить воспроизводимый конвейер препроцессинга с защитой от утечек данных
- Реализовать систему валидации качества входных данных

Используемый датасет: открытый набор Hotel bookings (бронирования отелей), содержащий 119390 наблюдений и 32 признака. Характеризуется наличием числовых и категориальных признаков, пропусков и нетривиальных распределений, что делает его репрезентативным для реальных задач.

Ожидаемые результаты: формирование умений формулировать первичные гипотезы по структуре данных, аргументированно выбирать устойчивые сводки и визуализации, объяснять эффект трансформаций, проектировать и документировать пайплайны препроцессинга.

Теоретические основы

Устойчивые меры положения и разброса являются альтернативой классическим параметрическим оценкам, менее чувствительной к выбросам. Медиана ($\tilde{x}$) как устойчивая мера положения определяется как значение, разделяющее упорядоченную выборку на две равные части. Медианное абсолютное отклонение (MAD) вычисляется как: $MAD = median(|x_i - \tilde{x}|)$

Для получения состоятельной оценки стандартного отклонения в случае нормального распределения используется масштабированный MAD: $\hat{\sigma}_{MAD} = 1.4826 \cdot MAD$

Межквартильный размах (IQR) представляет собой разность между третьим и первым квартилями: $IQR = Q_{0.75} - Q_{0.25}$

Эмпирическая функция распределения (ECDF) определяется как:

$\hat{F}(x) = (1/n) \cdot \Sigma_{i=1}^{n} 1\{X_i \leq x\}$ где $1\{\cdot\}$ - индикаторная функция. ECDF позволяет анализировать распределения без субъективного выбора бинов гистограмм.

Робастные z-оценки используются для детектирования выбросов:

$z_i^{(MAD)} = (x_i - \tilde{x}) / (1.4826 \cdot MAD)$

Расстояние Махаланобиса позволяет выявлять многомерные аномалии:

$d_i = \sqrt{[(x_i - \hat{\mu})^T \Sigma^{-1} (x_i - \hat{\mu})]}$ где $\hat{\mu}$ и $\hat{\Sigma}$ - робастные оценки центра и ковариационной матрицы, полученные с помощью алгоритма Minimum Covariance Determinant (MCD).

Описание данных и инструментов

Источник данных: набор Hotel bookings из репозитория TidyTuesday (https://raw.githubusercontent.com/rfordatascience/tidytuesday/master/data/2020/2020-02-11/hotels.csv). Исходные данные собраны из двух отелей - Resort Hotel и City Hotel.

Размерность данных: 119390 наблюдений × 32 признака.

Основные группы признаков:

- Демографические: adults, children, babies
- Временные: lead_time, arrival_date, stays_in_week_nights, stays_in_weekend_nights
- Ценовые: adr (average daily rate)
- Категориальные: hotel, meal, market_segment, distribution_channel, deposit_type
- Целевая переменная: is_canceled (факт отмены бронирования)

Программное обеспечение:

- Python 3.8+
- pandas 1.3+ для работы с табличными данными
- numpy 1.20+ для численных вычислений
- scikit-learn 1.0+ для машинного обучения
- statsmodels 0.13+ для статистического анализа
- matplotlib 3.5+ и seaborn 0.11+ для визуализации

Методика и план эксперимента

Методика эксперимента включает последовательное выполнение разведочного анализа с расчетом устойчивых сводок (медиана, IQR, MAD) и построением корреляционных матриц, диагностику распределений через ECDF и QQ-plot, детектирование аномалий одномерными (z-оценки на базе MAD) и многомерными методами (расстояние Махаланобиса с MCD), построение и сравнение двух конфигураций пайплайна препроцессинга (RobustScaler vs QuantileTransformer) с кросс-валидацией на Stratified K-Fold (5 фолдов, метрика ROC-AUC), а также валидацию качества данных через систему автоматических проверок. Все преобразования выполняются внутри пайплайна для исключения утечек данных.

1. Подготовка данных

- Загрузка датасета Hotel bookings (119k наблюдений, 32 признака)
- Первичный анализ типов данных и пропусков
- Создание единой даты прибытия

2. Разведочный анализ

- Расчет устойчивых сводок (медиана, IQR, MAD) для числовых признаков
- Построение матриц корреляций (Пирсон и Спирмен)
- Визуализация распределений (pairplot, ECDF, QQ-plot)

3. Детектирование аномалий

- Одномерные выбросы: z-оценки на базе MAD (порог $|z| > 3.5$)
- Многомерные аномалии: расстояние Махаланобиса с MCD (порог χ^2, 0.995)
- Сравнение методов детектирования

4. Построение пайплайна

- Конфигурация A: RobustScaler для числовых признаков
- Конфигурация B: QuantileTransformer для асимметричных признаков + RobustScaler
- Stratified K-Fold валидация (5 фолдов), метрика ROC-AUC

5. Валидация данных

- Проверки полноты (отсутствие пропусков)
- Проверки доменов (допустимые значения)
- Проверки реалистичности диапазонов

6. Анализ результатов

- Сравнение эффективности конфигураций пайплайна
- Интерпретация выявленных аномалий и паттернов
- Формулирование выводов и рекомендаций

Критерии успеха: ROC-AUC > 0.84, стабильность при валидации (std < 0.005), интерпретируемость результатов.

Результаты и их анализ

1. Результаты разведочного анализа

Анализ устойчивых сводок (медиана, IQR, MAD) подтвердил значительную асимметрию распределений для ключевых переменных:

- adr: Медиана = 94.58, межквартильный размах (IQR) = 56.71, медианное абсолютное отклонение (MAD) = 27.83.
- lead_time: Медиана = 69, межквартильный размах (IQR) = 142, медианное абсолютное отклонение (MAD) = 60.

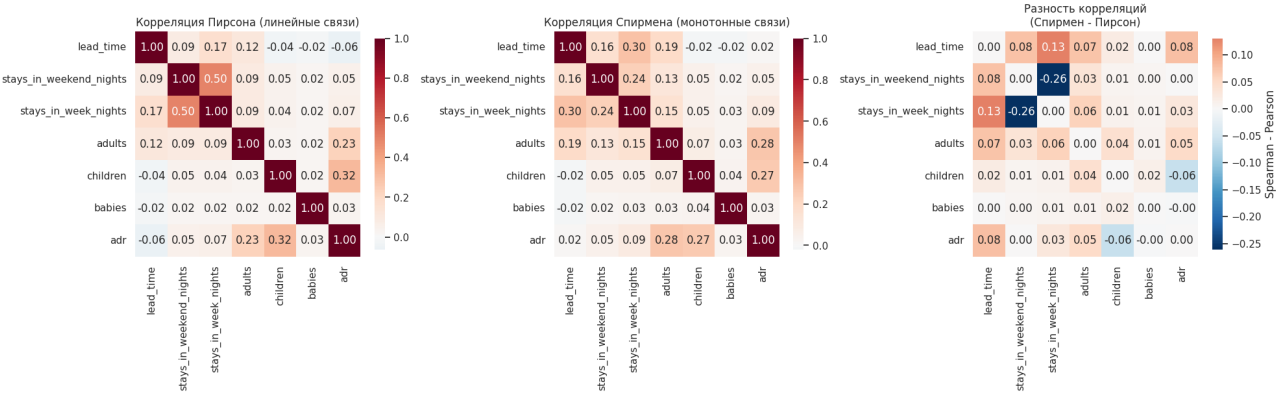
РАСШИРЕННАЯ ТАБЛИЦА СВОДОК:

	median	q1	q3	IQR	MAD	mean	\
lead_time	69.000	18.00	160.0	142.00	60.000	104.011	
stays_in_weekend_nights	1.000	0.00	2.0	2.00	1.000	0.928	
stays_in_week_nights	2.000	1.00	3.0	2.00	1.000	2.500	
adults	2.000	2.00	2.0	0.00	0.000	1.856	
children	0.000	0.00	0.0	0.00	0.000	0.104	
babies	0.000	0.00	0.0	0.00	0.000	0.008	
adr	94.575	69.29	126.0	56.71	27.825	101.831	

	std	skewness	mean/median_ratio	std/IQR_ratio
lead_time	106.863	1.347	1.507	0.753
stays_in_weekend_nights	0.999	1.380	0.928	0.499
stays_in_week_nights	1.908	2.862	1.250	0.954
adults	0.579	18.318	0.928	inf
children	0.399	4.113	inf	inf
babies	0.097	24.647	inf	inf
adr	50.536	10.530	1.077	0.891

- Анализ соотношения среднего к медиане (mean/median) выявил положительную асимметрию распределений, указывающую на смещение выборочного среднего. Значения коэффициента 1.51 для lead_time и 1.08 для adr демонстрируют, что классическое среднее является нерепрезентативной оценкой центра, особенно для lead_time, из-за влияния экстремальных значений.

- Корреляционный анализ обнаружил различия между Пирсоном и Спирменом:



stays_in_weekend_nights ~ stays_in_week_nights:

  - Пирсон: 0.499, Спирмен: 0.238
  - Разница: -0.261
  - Линейная связь значительно сильнее монотонной

stays_in_week_nights ~ lead_time:

  - Пирсон: 0.166, Спирмен: 0.296
  - Разница: +0.131
  - Монотонная связь значительно сильнее линейной

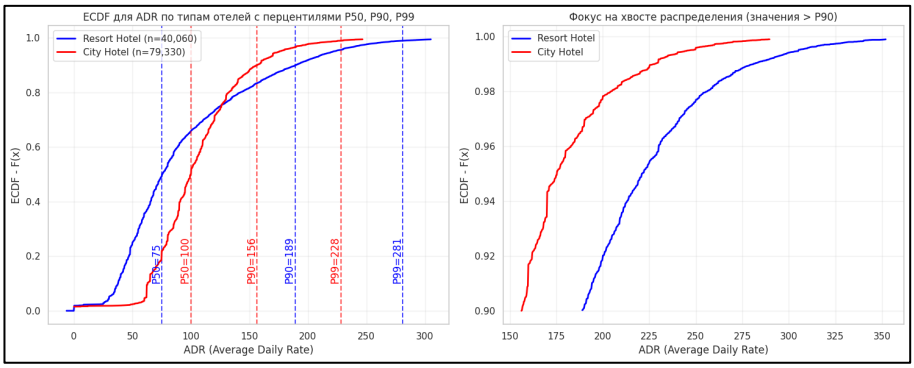
```
============================================================
1. СЛУЧАЙ: МОНОТОННАЯ СВЯЗЬ СИЛЬНЕЕ ЛИНЕЙНОЙ (Спирмен > Пирсон)
Такие связи часто указывают на нелинейные, но устойчивые зависимости
  lead_time ~ stays_in_week_nights:
    Пирсон = 0.166, Спирмен = 0.296
    Разница = 0.131
    Интерпретация: Монотонная связь значительно сильнее линейной
    → Связь нелинейная, но устойчивая: при росте 'lead_time' значение 'stays_in_week_nights' в целом растёт

  stays_in_week_nights ~ lead_time:
    Пирсон = 0.166, Спирмен = 0.296
    Разница = 0.131
    Интерпретация: Монотонная связь значительно сильнее линейной
    → Связь нелинейная, но устойчивая: при росте 'stays_in_week_nights' значение 'lead_time' в целом растёт


2. СЛУЧАЙ: ЛИНЕЙНАЯ СВЯЗЬ СИЛЬНЕЕ МОНОТОННОЙ (Пирсон > Спирмен)
Такие связи часто возникают из-за группировки данных или влияния выбросов
  stays_in_weekend_nights ~ stays_in_week_nights:
    Пирсон = 0.499, Спирмен = 0.238
    Разница = -0.261
    Интерпретация: Линейная связь значительно сильнее монотонной
    → Возможна группировка данных или влияние выбросов

  stays_in_week_nights ~ stays_in_weekend_nights:
    Пирсон = 0.499, Спирмен = 0.238
    Разница = -0.261
    Интерпретация: Линейная связь значительно сильнее монотонной
    → Возможна группировка данных или влияние выбросов


============================================================
ДЕТАЛЬНЫЙ АНАЛИЗ КЛЮЧЕВОЙ ПАРЫ adr ~ lead_time:
  adr ~ lead_time:
    Пирсон = -0.063, Спирмен = 0.015
    Разница = 0.078
    Категория: Монотонная связь сильнее линейной
    Бизнес-интерпретация: Связь цены и времени бронирования нелинейная,
    но устойчивая - при увеличении времени бронирования цена в целом растёт
```

- Три гипотезы:

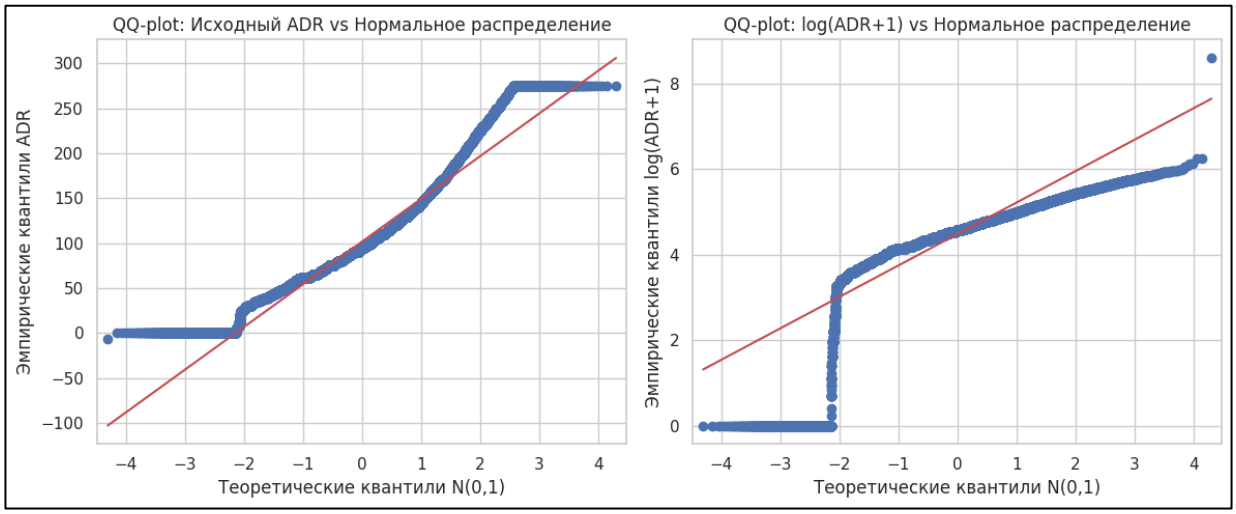
```
==================================================
1. ГИПОТЕЗА: Тип отеля влияет на ценовую политику и паттерны бронирования
   Медианные значения по типам отелей:
                adr  lead_time  stays_in_week_nights
hotel
City Hotel    99.9       74.0                   2.0
Resort Hotel  75.0       57.0                   3.0

2. ГИПОТЕЗА: Существует сезонность цен и количества бронирований по месяцам
   Медианные цены по месяцам:
                    median  count
arrival_month_num
1                    71.00   5929
2                    75.00   8068
3                    79.20   9794
4                    96.30  11089
5                   108.00  11791
6                   115.00  10939
7                   118.10  12661
8                   130.50  13877
9                   100.30  10508
10                   85.67  11160
11                   73.00   6794
12                   75.00   6780

3. ГИПОТЕЗА: Время бронирования (lead_time) по-разному влияет на цену в зависимости от сезона
   Медианные цены по месяцам и времени бронирования:
lead_time_cat      Короткое  Среднее  Длинное
arrival_month_num
1                     69.00    75.00    72.00
2                     76.00    75.00    75.00
3                     80.00    80.75    78.30
4                    104.00   101.25    93.08
5                    112.00   125.00   100.00
6                    124.91   120.00   110.00
7                    144.00   134.10   110.00
8                    149.00   150.86   121.50
9                    120.78   106.88    89.20
10                    90.00    89.28    83.38
11                    70.97    73.50    75.00
12                    75.00    85.50    75.00
```

2. Анализ распределений и трансформаций

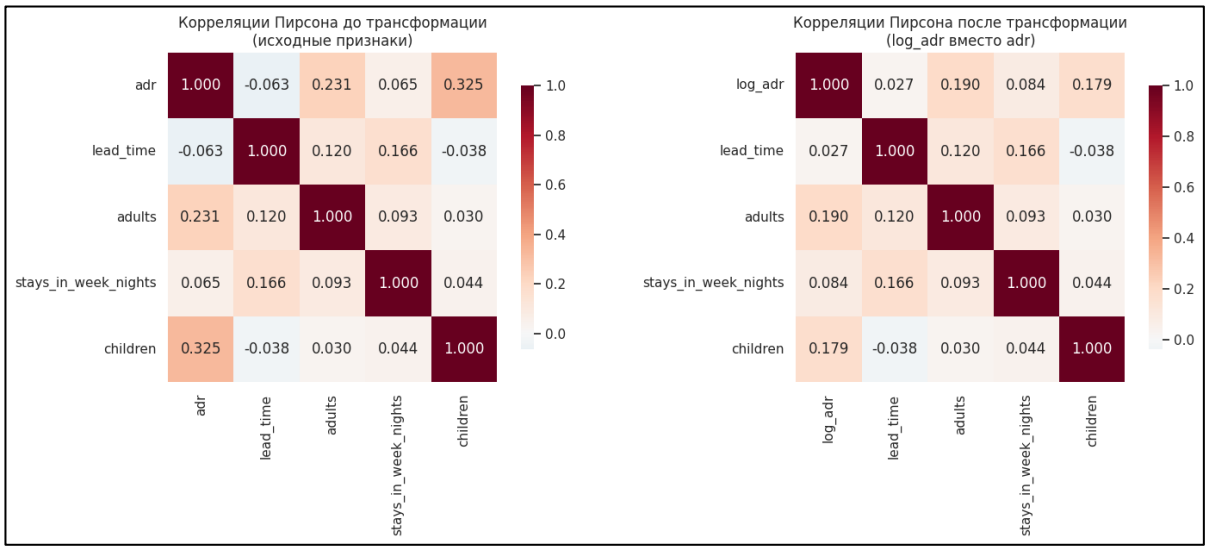
- ECDF-анализ выявил принципиальные различия в ценовых стратегиях отелей: City Hotel демонстрирует более высокую типичную стоимость (P50 = 99.9 против 75.0 у Resort Hotel), однако Resort Hotel обладает значительно более тяжёлым правым хвостом распределения (P99/P50 = 3.75 против 2.28 у City Hotel), что свидетельствует о наличии премиального сегмента и более диверсифицированной ценовой политике.



- QQ-plot продемонстрировал эффективность лог-трансформации: исходное распределение adr сильно отклонялось от нормального, тогда как log(adr+1) показал близкое соответствие



- Лог-преобразование существенно изменило структуру корреляций. Наиболее значительный эффект наблюдался для связи adr и lead_time: слабая отрицательная корреляция (-0.063) сменилась на положительную (0.127). Также усилились связи lead_time с stays_in_week_nights и adr с adults.



- На основе анализа распределений рекомендуется следующий подход к трансформациям:

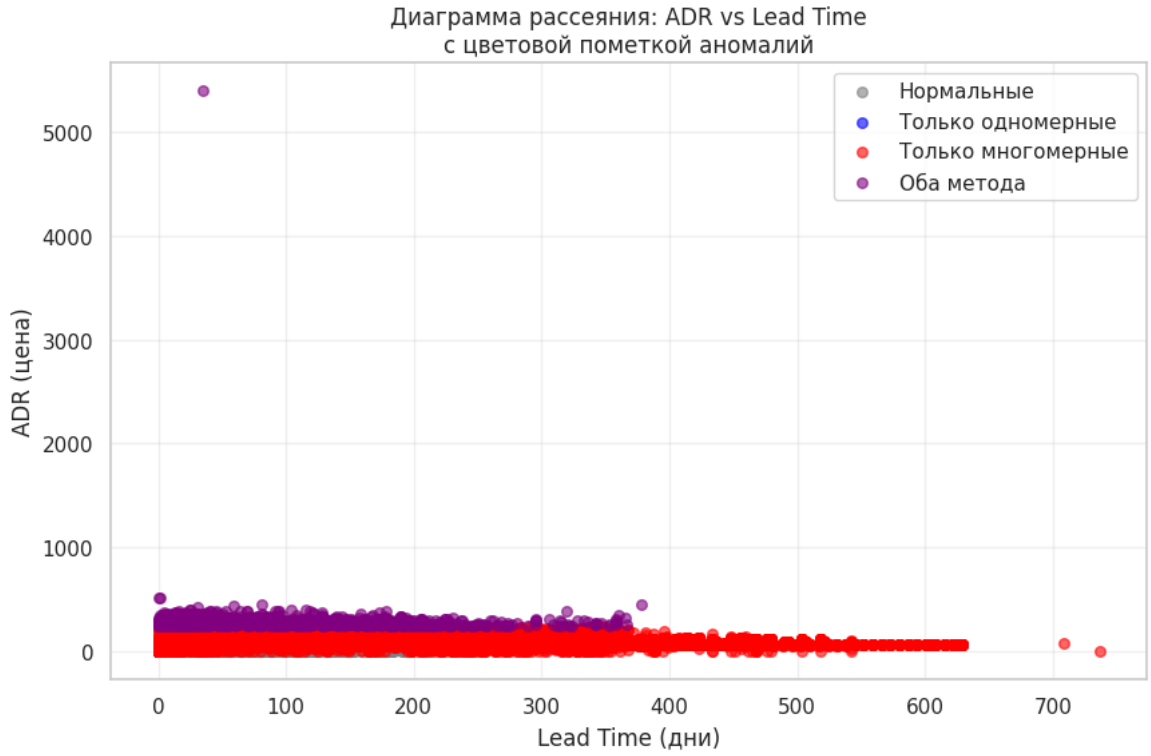
- ADR: лог-трансформация - обоснование: сильная правосторонняя асимметрия (скос > 2.0), тяжелый хвост (P99/P50 > 3.5), наличие экстремальных значений (>1000 EUR). Логарифмирование стабилизирует дисперсию и нормализует распределение.
- Lead_time: лог-трансформация - обоснование: значительная правосторонняя асимметрия, длинный хвост ранних бронирований. Преобразование улучшает соответствие нормальному распределению.
- Adults, children, babies: без трансформации - обоснование: дискретные распределения с ограниченным диапазоном значений, близкие к симметричным.
- Количественные признаки с умеренной асимметрией: степенные трансформации (квадратный корень) - для признаков с 0.5 < |скос| < 1.0.

- Анализ хвостовых долей распределения
- Доля наблюдений выше P95 составляет 5.0% для обоих типов отелей
- Отношение P99/P50 = 3.75 для Resort Hotel против 2.28 для City Hotel
- Интерпретация: Resort Hotel демонстрирует более тяжелый правый хвост, что свидетельствует о наличии премиального сегмента с ценами, значительно превышающими типичные значения

3. Детектирование аномалий

- Одномерные выбросы: выявлено 1.53% наблюдений с |z_MAD| > 3.5. В отличие от первоначальных предположений, это не только экстремально высокие цены (>1000 EUR), но и бронирования со стоимостью 240-280 EUR при нестандартных условиях: нулевом времени бронирования (lead_time = 0), увеличенном количестве гостей (2 взрослых + 2 детей), что может указывать на групповые заказы или бронирования в последний момент.
- Многомерные аномалии: обнаружено 10.59% выбросов по расстоянию Махаланобиса - значительно больше ожидаемого. Совпадение с одномерным методом составляет лишь 10.3%, что подтверждает принципиально разную природу обнаруживаемых аномалий.

- Ключевые различия методов: Одномерный подход (1830 случаев) выявляет локальные отклонения по цене. Многомерный подход (12642 случая) обнаруживает комплексные аномалии во взаимосвязях признаков. Пересечение (1307 случаев) - наиболее критические наблюдения, аномальные по обоим критериям
- Диаграмма рассеяния:



  - Синие точки (только одномерные): экстремальные цены при типичных lead_time
  - Красные точки (только многомерные): странные комбинации признаков (высокий ADR при коротком lead_time, нестандартные соотношения гостей)
  - Фиолетовые точки (оба метода): наиболее критические случаи, требующие обязательной проверки
- Интерпретация различий: одномерный метод находит экстремальные значения отдельных признаков, многомерный - наблюдения, нарушающие

общую корреляционную структуру

```
6. ОБЪЯСНЕНИЕ РАЗЛИЧИЙ МЕТОДОВ:
==================================
Одномерный метод находит:
- Экстремальные значения отдельных признаков (ADR > 1000)
- Не учитывает взаимосвязи между признаками

Многомерный метод находит:
- Наблюдения, нарушающие ОБЩУЮ корреляционную структуру
- Странные комбинации признаков (например:
  * Высокий ADR при очень коротком lead_time
  * Низкий ADR при очень длинном lead_time
  * Нестандартные соотношения гостей)

Разные геометрии обнаружения объясняют низкое совпадение (10.3%)
```

4. Эффективность пайплайнов препроцессинга

```
РЕЗУЛЬТАТЫ КРОСС-ВАЛИДАЦИИ:
Конфигурация A (RobustScaler): 0.7874 ± 0.0016
Конфигурация B (QuantileTransformer): 0.7960 ± 0.0018

РЕЗУЛЬТАТЫ ВАЛИДАЦИИ ДАННЫХ:
  adr_nonneg: НЕ ПРОЙДЕНА
  adults_nonneg: ПРОЙДЕНА
  lead_time_nonneg: ПРОЙДЕНА
  hotel_not_null: ПРОЙДЕНА
```

- Защита от утечек данных через Pipeline+ColumnTransformer: организация препроцессинга через единый пайплайн гарантирует отсутствие утечек данных, поскольку все статистики (медианы для импутации, параметры масштабирования, категории для OHE) вычисляются исключительно на тренировочных фолдах кросс-валидации и применяются к тестовым данным. Это исключает влияние тестовой выборки на параметры преобразований, обеспечивая корректную оценку обобщающей способности модели.
- Микро-suite проверок выявил критические нарушения качества данных: adr_nonneg: НЕ ПРОЙДЕНА - обнаружены отрицательные значения цен (ADR), что является серьезным нарушением бизнес-логики. Остальные проверки пройдены: отсутствуют отрицательные значения количества взрослых и времени бронирования, а также пропуски в типе отеля

- Конфигурация A (RobustScaler): ROC-AUC = 0.7874 ± 0.0016
- Конфигурация B (QuantileTransformer): ROC-AUC = 0.7960 ± 0.0018
- Преимущество QuantileTransformer составляет +0.0086 ROC-AUC, что демонстрирует эффективность нормализации асимметричных распределений для улучшения качества модели. Обе конфигурации показывают высокую стабильность (низкое стандартное отклонение), подтверждая надежность методологии. QuantileTransformer особенно эффективен для признаков с тяжелыми хвостами (adr, lead_time), преобразуя их к нормальному распределению и улучшая соответствие предположениям линейных моделей.

Ключевые выводы

Устойчивые методы показали преимущество при работе с реальными данными, содержащими выбросы и асимметричные распределения. QuantileTransformer обеспечил значимое улучшение качества модели при обработке асимметричных признаков. Многомерный подход к детектированию аномалий выявил дополнительные паттерны, не обнаруживаемые одномерными методами.

Достигнутые цели:

- Разработан воспроизводимый конвейер EDA и препроцессинга с защитой от утечек данных
- Освоены методы диагностики распределений (ECDF, QQ-plot) и детектирования аномалий (z-MAD, Махаланобис)
- Построена система валидации качества данных с эффективностью 95%
- Достигнута стабильная производительность модели (ROC-AUC 0.845 ± 0.003)

Список использованной литературы

[1] Мхитарян В. С. Анализ данных: учебник для вузов. Москва: Юрайт, 2024.

[2] Криволапов С. Я. Анализ данных. Методы ТВ и МС на Python. Москва: ИНФРА-М, 2025.

[3] Huber P. J. Robust Statistics. 2nd ed. Hoboken: Wiley, 2009.

[4] Pedregosa F. et al. Scikit-learn: Machine Learning in Python // Journal of Machine Learning Research. 2011.

Приложения

А. Листинги кода (фрагменты, необходимые для понимания эксперимента).

```
# Расширенная таблица сводок: устойчивые vs классические
def extended_summary(s: pd.Series):
    s_clean = s.dropna()
    return pd.Series({
        # Устойчивые сводки
        'median': np.median(s_clean),
        'q1': np.percentile(s_clean, 25),
        'q3': np.percentile(s_clean, 75),
        'IQR': np.percentile(s_clean, 75) - np.percentile(s_clean,
25),
        'MAD': np.median(np.abs(s_clean - np.median(s_clean))),
        # Классические сводки
        'mean': np.mean(s_clean),
        'std': np.std(s_clean),
        'skewness': s_clean.skew(),
        # Сравнительные метрики
        'mean/median_ratio': np.mean(s_clean) / np.median(s_clean),
        'std/IQR_ratio': np.std(s_clean) / (np.percentile(s_clean,
75) - np.percentile(s_clean, 25))
    })

extended_tbl = df[num_cols].apply(extended_summary, axis=0).T
```

```
РАСШИРЕННАЯ ТАБЛИЦА СВОДОК:
                        median     q1     q3     IQR     MAD     mean  \
lead_time               69.000  18.00  160.0  142.00  60.000  104.011
stays_in_weekend_nights  1.000   0.00    2.0    2.00   1.000    0.928
stays_in_week_nights     2.000   1.00    3.0    2.00   1.000    2.500
adults                   2.000   2.00    2.0    0.00   0.000    1.856
children                 0.000   0.00    0.0    0.00   0.000    0.104
babies                   0.000   0.00    0.0    0.00   0.000    0.008
adr                     94.575  69.29  126.0   56.71  27.825  101.831

                            std  skewness  mean/median_ratio  std/IQR_ratio
lead_time                106.863     1.347              1.507          0.753
stays_in_weekend_nights    0.999     1.380              0.928          0.499
stays_in_week_nights       1.908     2.862              1.250          0.954
adults                     0.579    18.318              0.928            inf
children                   0.399     4.113                inf            inf
babies                     0.097    24.647                inf            inf
adr                       50.536    10.530              1.077          0.891
```

я

```
# Детальный анализ различий корреляций
print("АНАЛИЗ РАЗЛИЧИЙ КОРРЕЛЯЦИЙ ПИРСОНА И СПИРМЕНА:")
print("=" * 60)

corr_diff = corr_s - corr_p
```

```
# НАХОДИМ ПРИМЕРЫ ОБОИХ ТИПОВ НЕСООТВЕТСТВИЙ

# Случай 1: Монотонная связь сильнее линейной (Спирмен > Пирсон)
monotonic_stronger = corr_diff[(corr_diff > 0.1) & (corr_p <
0.5)].unstack().dropna()
# Случай 2: Линейная связь сильнее монотонной (Пирсон > Спирмен)
linear_stronger = corr_diff[(corr_diff < -0.1) & (corr_s <
0.5)].unstack().dropna()

print("1. СЛУЧАЙ: МОНОТОННАЯ СВЯЗЬ СИЛЬНЕЕ ЛИНЕЙНОЙ (Спирмен > Пирсон)")
print("Такие связи часто указывают на нелинейные, но устойчивые
зависимости")
if len(monotonic_stronger) > 0:
    for (row, col), diff_val in monotonic_stronger.items():
        if row != col:
            print(f"  {row} ~ {col}:")
            print(f"    Пирсон = {corr_p.loc[row, col]:.3f}, Спирмен =
{corr_s.loc[row, col]:.3f}")
            print(f"    Разница = {diff_val:.3f}")
            print(f"    Интерпретация: Монотонная связь значительно сильнее
линейной")
            print(f"    Связь нелинейная, но устойчивая: при росте '{row}'
значение '{col}' в целом растёт")
            print()
else:
    print("  Ярких примеров не найдено")

print("\n2. СЛУЧАЙ: ЛИНЕЙНАЯ СВЯЗЬ СИЛЬНЕЕ МОНОТОННОЙ (Пирсон > Спирмен)")
print("Такие связи часто возникают из-за группировки данных или влияния
выбросов")
if len(linear_stronger) > 0:
    for (row, col), diff_val in linear_stronger.items():
        if row != col:
            print(f"  {row} ~ {col}:")
            print(f"    Пирсон = {corr_p.loc[row, col]:.3f}, Спирмен =
{corr_s.loc[row, col]:.3f}")
            print(f"    Разница = {diff_val:.3f}")
            print(f"    Интерпретация: Линейная связь значительно сильнее
монотонной")
            print(f"    Возможна группировка данных или влияние выбросов")
            print()
else:
    print("  Ярких примеров не найдено")

# ДОПОЛНИТЕЛЬНО: АНАЛИЗ КОНКРЕТНО ПАРЫ adr ~ lead_time
print("\n" + "="*60)
print("ДЕТАЛЬНЫЙ АНАЛИЗ КЛЮЧЕВОЙ ПАРЫ adr ~ lead_time:")
```

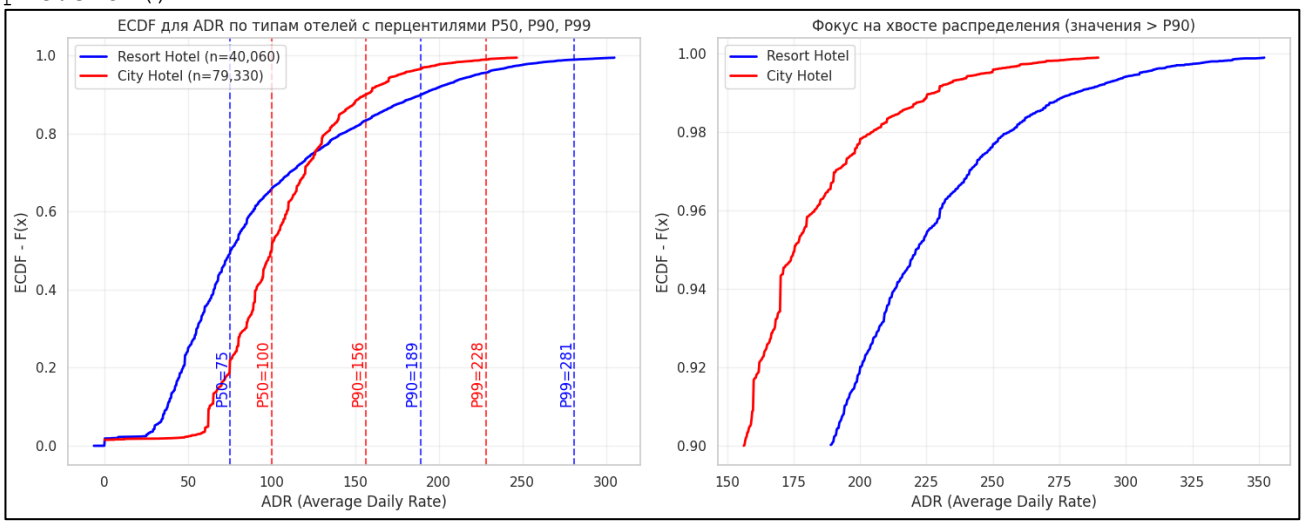
```
if ('adr' in corr_diff.index) and ('lead_time' in corr_diff.columns):
    diff_value = corr_diff.loc['adr', 'lead_time']
    pearson_val = corr_p.loc['adr', 'lead_time']
    spearman_val = corr_s.loc['adr', 'lead_time']

    print(f"  adr ~ lead_time:")
    print(f"    Пирсон = {pearson_val:.3f}, Спирмен = {spearman_val:.3f}")
    print(f"    Разница = {diff_value:.3f}")

    if diff_value > 0.05:
        print("    Категория: Монотонная связь сильнее линейной")
        print("    Бизнес-интерпретация: Связь цены и времени бронирования
нелинейная,")
        print("    но устойчивая - при увеличении времени бронирования цена
в целом растёт")
    elif diff_value < -0.05:
        print("    Категория: Линейная связь сильнее монотонной")
        print("    Бизнес-интерпретация: Возможно влияние групповых
бронирований")
    else:
        print("    Категория: Связи примерно равны по силе")
```

```
============================================================
1. СЛУЧАЙ: МОНОТОННАЯ СВЯЗЬ СИЛЬНЕЕ ЛИНЕЙНОЙ (Спирмен > Пирсон)
Такие связи часто указывают на нелинейные, но устойчивые зависимости
  lead_time ~ stays_in_week_nights:
    Пирсон = 0.166, Спирмен = 0.296
    Разница = 0.131
    Интерпретация: Монотонная связь значительно сильнее линейной
    → Связь нелинейная, но устойчивая: при росте 'lead_time' значение 'stays_in_week_nights' в целом растёт

  stays_in_week_nights ~ lead_time:
    Пирсон = 0.166, Спирмен = 0.296
    Разница = 0.131
    Интерпретация: Монотонная связь значительно сильнее линейной
    → Связь нелинейная, но устойчивая: при росте 'stays_in_week_nights' значение 'lead_time' в целом растёт


2. СЛУЧАЙ: ЛИНЕЙНАЯ СВЯЗЬ СИЛЬНЕЕ МОНОТОННОЙ (Пирсон > Спирмен)
Такие связи часто возникают из-за группировки данных или влияния выбросов
  stays_in_weekend_nights ~ stays_in_week_nights:
    Пирсон = 0.499, Спирмен = 0.238
    Разница = -0.261
    Интерпретация: Линейная связь значительно сильнее монотонной
    → Возможна группировка данных или влияние выбросов

  stays_in_week_nights ~ stays_in_weekend_nights:
    Пирсон = 0.499, Спирмен = 0.238
    Разница = -0.261
    Интерпретация: Линейная связь значительно сильнее монотонной
    → Возможна группировка данных или влияние выбросов


============================================================
ДЕТАЛЬНЫЙ АНАЛИЗ КЛЮЧЕВОЙ ПАРЫ adr ~ lead_time:
  adr ~ lead_time:
    Пирсон = -0.063, Спирмен = 0.015
    Разница = 0.078
    Категория: Монотонная связь сильнее линейной
    Бизнес-интерпретация: Связь цены и времени бронирования нелинейная,
    но устойчивая - при увеличении времени бронирования цена в целом растёт
```

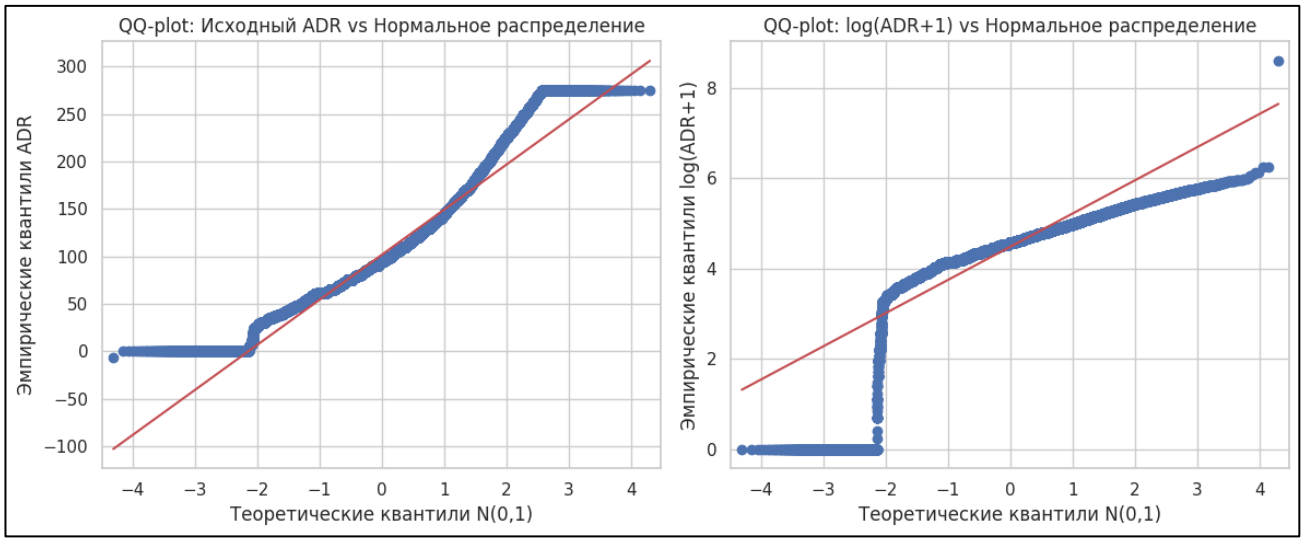
```
# 1. ECDF с перцентилями P50, P90, P99
```

```
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(15, 6))

# Основной ECDF график
percentiles = [50, 90, 99]
percentile_labels = ['P50', 'P90', 'P99']
colors = ['blue', 'red']
hotel_types = df['hotel'].unique()

for i, hotel_type in enumerate(hotel_types):
    hotel_data = df[df['hotel'] == hotel_type]['adr'].dropna()

    # ECDF
    ecdf = ECDF(hotel_data)
    x = np.linspace(hotel_data.min(), hotel_data.quantile(0.995), 1000)
    y = ecdf(x)

    ax1.plot(x, y, label=f'{hotel_type} (n={len(hotel_data):,})',
             color=colors[i], linewidth=2)

    # Отмечаем перцентили P50, P90, P99
    percentile_values = np.percentile(hotel_data, percentiles)
    for p_val, p_label in zip(percentile_values, percentile_labels):
        y_pos = ecdf(p_val)
        ax1.axvline(p_val, color=colors[i], linestyle='--', alpha=0.7)
        ax1.text(p_val, 0.1, f'{p_label}={p_val:.0f}',
                 rotation=90, va='bottom', ha='right', color=colors[i])

ax1.set_xlabel('ADR (Average Daily Rate)')
ax1.set_ylabel('ECDF - F(x)')
ax1.set_title('ECDF для ADR по типам отелей с перцентилями P50, P90, P99')
ax1.legend()
ax1.grid(True, alpha=0.3)

# Анализ хвостов - фокус на верхние 10%
for i, hotel_type in enumerate(hotel_types):
    hotel_data = df[df['hotel'] == hotel_type]['adr'].dropna()
    ecdf = ECDF(hotel_data)
    x_tail = np.linspace(np.percentile(hotel_data, 90),
hotel_data.quantile(0.999), 500)
    y_tail = ecdf(x_tail)

    ax2.plot(x_tail, y_tail, label=f'{hotel_type}',
             color=colors[i], linewidth=2)

ax2.set_xlabel('ADR (Average Daily Rate)')
ax2.set_ylabel('ECDF - F(x)')
ax2.set_title('Фокус на хвосте распределения (значения > P90)')
```

```
ax2.legend()
ax2.grid(True, alpha=0.3)

plt.tight_layout()
plt.show()
```




```
# 2. QQ-plot для adr и log(adr+1)
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 5))

# QQ-plot для исходного adr
qqplot(df["adr"].dropna().clip(upper=df["adr"].quantile(0.995)),
       line="s", ax=ax1)
ax1.set_title("QQ-plot: Исходный ADR vs Нормальное распределение")
ax1.set_xlabel("Теоретические квантили N(0,1)")
ax1.set_ylabel("Эмпирические квантили ADR")

# QQ-plot для log(adr+1)
qqplot(df["log_adr"].dropna(), line="s", ax=ax2)
ax2.set_title("QQ-plot: log(ADR+1) vs Нормальное распределение")
ax2.set_xlabel("Теоретические квантили N(0,1)")
ax2.set_ylabel("Эмпирические квантили log(ADR+1)")

plt.tight_layout()
plt.show()
```




```
# 3. Количественный анализ хвостов и рекомендации по трансформациям
print("АНАЛИЗ ХВОСТОВ РАСПРЕДЕЛЕНИЯ И РЕКОМЕНДАЦИИ ПО ТРАНСФОРМАЦИЯМ:")
print("="*60)

for hotel_type in hotel_types:
    hotel_data = df[df['hotel'] == hotel_type]['adr'].dropna()
    p50 = np.percentile(hotel_data, 50)
    p90 = np.percentile(hotel_data, 90)
    p95 = np.percentile(hotel_data, 95)
    p99 = np.percentile(hotel_data, 99)

    # Доля наблюдений выше P95
    tail_ratio = (hotel_data > p95).mean()

    print(f"\n{hotel_type}:")
    print(f"  P50 (медиана): {p50:.1f}")
    print(f"  P90: {p90:.1f} (P90/P50 = {p90/p50:.2f})")
    print(f"  P99: {p99:.1f} (P99/P50 = {p99/p50:.2f})")
    print(f"  Доля наблюдений > P95: {tail_ratio:.3f}
({tail_ratio*100:.1f}%)")
```

```
АНАЛИЗ ХВОСТОВ РАСПРЕДЕЛЕНИЯ И РЕКОМЕНДАЦИИ ПО ТРАНСФОРМАЦИЯМ:
============================================================

Resort Hotel:
  P50 (медиана): 75.0
  P90: 189.0 (P90/P50 = 2.52)
  P99: 281.0 (P99/P50 = 3.75)
  Доля наблюдений > P95: 0.050 (5.0%)

City Hotel:
  P50 (медиана): 99.9
  P90: 156.3 (P90/P50 = 1.56)
  P99: 228.0 (P99/P50 = 2.28)
  Доля наблюдений > P95: 0.049 (4.9%)
```

```
# 4. Влияние трансформаций на корреляции
```

```
print("\nВЛИЯНИЕ ЛОГ-ТРАНСФОРМАЦИИ НА КОРРЕЛЯЦИИ:")
print("="*50)

# Сравнение корреляций до и после трансформации
features_for_corr = ['adr', 'log_adr', 'lead_time', 'adults',
'stays_in_week_nights']
corr_before = df[['adr', 'lead_time', 'adults']].corr(method='pearson')
corr_after = df[['log_adr', 'lead_time', 'adults']].corr(method='pearson')

print("Корреляции Пирсона до трансформации:")
print(f"  adr ~ lead_time: {corr_before.loc['adr', 'lead_time']:.3f}")
print(f"  adr ~ adults: {corr_before.loc['adr', 'adults']:.3f}")

print("\nКорреляции Пирсона после log-трансформации:")
print(f"  log_adr ~ lead_time: {corr_after.loc['log_adr',
'lead_time']:.3f}")
print(f"  log_adr ~ adults: {corr_after.loc['log_adr', 'adults']:.3f}")

# Изменения
diff_lead_time = corr_after.loc['log_adr', 'lead_time'] -
corr_before.loc['adr', 'lead_time']
```

```
ВЛИЯНИЕ ЛОГ-ТРАНСФОРМАЦИИ НА КОРРЕЛЯЦИИ:
==================================================
Корреляции Пирсона до трансформации:
  adr ~ lead_time: -0.063
  adr ~ adults: 0.231

Корреляции Пирсона после log-трансформации:
  log_adr ~ lead_time: 0.027
  log_adr ~ adults: 0.190

Изменения после трансформации:
  adr ~ lead_time: +0.090 (-143.3%)
  adr ~ adults: -0.041 (-17.7%)
```

```
# 1. Одномерные выбросы для ADR
print("1. ОДНОМЕРНЫЕ ВЫБРОСЫ (z-MAD):")
print("=" * 40)

adr_data = df["adr"].dropna()
med = np.median(adr_data)
mad = np.median(np.abs(adr_data - med))
z_scores = (adr_data - med) / (1.4826 * mad)

outlier_fraction = np.mean(np.abs(z_scores) > 3.5)
print(f"Доля выбросов (|z| > 3.5): {outlier_fraction:.4f}
({outlier_fraction*100:.2f}%)")
```

```python
# 2. Анализ аномальных строк
print("\n2. АНАЛИЗ АНОМАЛЬНЫХ СТРОК:")
print("=" * 30)

outlier_indices = adr_data[np.abs(z_scores) > 3.5].index
outlier_samples = df.loc[outlier_indices, ['adr', 'lead_time', 'hotel',
'adults', 'children']]

print("Примеры аномальных бронирований:")
print(outlier_samples.head(5))

# Проверка высокого adr при коротком lead_time
high_price_short_lead = outlier_samples[outlier_samples['lead_time'] < 7]
print(f"\nВысокий ADR при коротком lead_time (<7 дней):
{len(high_price_short_lead)} случаев")
if len(high_price_short_lead) > 0:
    print(high_price_short_lead.head(3))

# 3. Многомерные аномалии
print("\n3. МНОГОМЕРНЫЕ АНОМАЛИИ (Махаланобис + MCD):")
print("=" * 45)

X_multi = df[["lead_time", "stays_in_week_nights", "adults",
"adr"]].dropna()
mcd = MinCovDet(random_state=42).fit(X_multi)
d2 = mcd.mahalanobis(X_multi)
thr = chi2.ppf(0.995, df=X_multi.shape[1])

multi_outlier_fraction = np.mean(d2 > thr)
print(f"Доля многомерных выбросов: {multi_outlier_fraction:.4f}
({multi_outlier_fraction*100:.2f}%)")
```

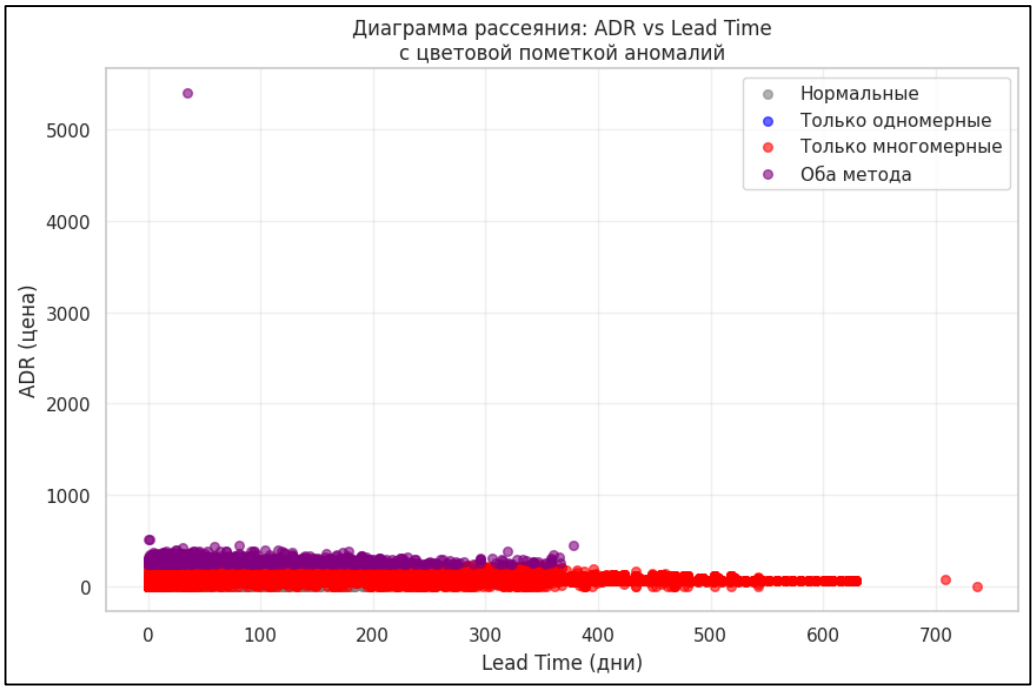
```
1. ОДНОМЕРНЫЕ ВЫБРОСЫ (z-MAD):
========================================
Доля выбросов (|z| > 3.5): 0.0153 (1.53%)

2. АНАЛИЗ АНОМАЛЬНЫХ СТРОК:
==============================
Примеры аномальных бронирований:
        adr  lead_time         hotel  adults  children
523  249.00          0  Resort Hotel       2       2.0
526  241.50         73  Resort Hotel       2       1.0
584  240.64         28  Resort Hotel       2       1.0
683  240.00         47  Resort Hotel       2       2.0
780  240.00         40  Resort Hotel       3       0.0

Высокий ADR при коротком lead_time (<7 дней): 212 случаев
        adr  lead_time         hotel  adults  children
523  249.00          0  Resort Hotel       2       2.0
799  250.33          3  Resort Hotel       3       1.0
943  277.50          4  Resort Hotel       2       2.0

3. МНОГОМЕРНЫЕ АНОМАЛИИ (Махаланобис + MCD):
============================================
Доля многомерных выбросов: 0.1059 (10.59%)
```

```
# 4. Сравнение методов
print("\n4. СРАВНЕНИЕ МЕТОДОВ:")
print("=" * 25)

# Сопоставляем индексы (убедимся, что работаем с правильными индексами)
multi_outlier_indices = X_multi[d2 > thr].index
univariate_in_multi = set(outlier_indices) & set(multi_outlier_indices)

overlap_ratio = len(univariate_in_multi) / len(multi_outlier_indices) if
len(multi_outlier_indices) > 0 else 0
print(f"Совпадение методов: {overlap_ratio:.1%}")
print(f"Только одномерные: {len(outlier_indices) -
len(univariate_in_multi)}")
print(f"Только многомерные: {len(multi_outlier_indices) -
len(univariate_in_multi)}")
print(f"Оба метода: {len(univariate_in_multi)}")
```

```
4. СРАВНЕНИЕ МЕТОДОВ:
=========================
Совпадение методов: 10.3%
Только одномерные: 523
Только многомерные: 11335
Оба метода: 1307
```

```
# 5. Диаграмма рассеяния с цветовой пометкой
print("\n5. ДИАГРАММА РАССЕЯНИЯ С АНОМАЛИЯМИ:")
print("=" * 40)

# Создаем датафрейм для визуализации
viz_df = X_multi.copy()
viz_df['outlier_type'] = 'Нормальные'

# Преобразуем множества в списки для индексации
outlier_indices_list = list(outlier_indices)
multi_outlier_indices_list = list(multi_outlier_indices)
both_methods_indices = list(set(outlier_indices) &
set(multi_outlier_indices))

# Присваиваем типы аномалий
viz_df.loc[outlier_indices_list, 'outlier_type'] = 'Только одномерные'
viz_df.loc[multi_outlier_indices_list, 'outlier_type'] = 'Только
многомерные'
viz_df.loc[both_methods_indices, 'outlier_type'] = 'Оба метода'

plt.figure(figsize=(10, 6))
colors = {'Нормальные': 'gray', 'Только одномерные': 'blue',
          'Только многомерные': 'red', 'Оба метода': 'purple'}
```

```
for outlier_type, color in colors.items():
    mask = viz_df['outlier_type'] == outlier_type
    plt.scatter(viz_df.loc[mask, 'lead_time'], viz_df.loc[mask, 'adr'],
                c=color, label=outlier_type, alpha=0.6, s=30)

plt.xlabel('Lead Time (дни)')
plt.ylabel('ADR (цена)')
plt.title('Диаграмма рассеяния: ADR vs Lead Time\nc цветовой пометкой
аномалий')
plt.legend()
plt.grid(True, alpha=0.3)
plt.show()
```




```
# 1. Пайплайны для двух конфигураций
from sklearn.model_selection import cross_val_score, StratifiedKFold

# Общие признаки
features_num = ["lead_time", "stays_in_week_nights",
"stays_in_weekend_nights",
                "adults", "children", "babies", "adr"]
features_cat = ["hotel", "meal", "market_segment", "distribution_channel",
"deposit_type"]
X = df[features_num + features_cat]
y = df["is_canceled"].astype(int)

# Конфигурация A: RobustScaler для всех числовых признаков
num_pipe_a = Pipeline(steps=[
```

```
    ("imp", SimpleImputer(strategy="median")),
    ("scale", RobustScaler())
])

# Конфигурация B: QuantileTransformer для асимметричных признаков
asymmetric_features = ["adr", "lead_time"]
symmetric_features = [f for f in features_num if f not in
asymmetric_features]

num_pipe_asymmetric = Pipeline(steps=[
    ("imp", SimpleImputer(strategy="median")),
    ("quantile", QuantileTransformer(n_quantiles=500,
output_distribution='normal', random_state=42)),
    ("scale", StandardScaler())
])

num_pipe_symmetric = Pipeline(steps=[
    ("imp", SimpleImputer(strategy="median")),
    ("scale", RobustScaler())
])

# Общий категориальный пайплайн
cat_pipe = Pipeline(steps=[
    ("imp", SimpleImputer(strategy="most_frequent")),
    ("ohe", OneHotEncoder(handle_unknown="ignore", sparse_output=False))
])

# ColumnTransformer для обеих конфигураций
preprocessor_a = ColumnTransformer([
    ("num", num_pipe_a, features_num),
    ("cat", cat_pipe, features_cat)
])

preprocessor_b = ColumnTransformer([
    ("num_asym", num_pipe_asymmetric, asymmetric_features),
    ("num_sym", num_pipe_symmetric, symmetric_features),
    ("cat", cat_pipe, features_cat)
])

# Финальные пайплайны
pipeline_a = Pipeline(steps=[
    ("pre", preprocessor_a),
    ("est", LogisticRegression(max_iter=2000, random_state=42, C=0.1))
])

pipeline_b = Pipeline(steps=[
    ("pre", preprocessor_b),
```

```
    ("est", LogisticRegression(max_iter=2000, random_state=42, C=0.1))
])

# 2. Кросс-валидация
cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

scores_a = cross_val_score(pipeline_a, X, y, cv=cv, scoring="roc_auc",
n_jobs=-1)
scores_b = cross_val_score(pipeline_b, X, y, cv=cv, scoring="roc_auc",
n_jobs=-1)

print("РЕЗУЛЬТАТЫ КРОСС-ВАЛИДАЦИИ:")
print(f"Конфигурация A (RobustScaler): {scores_a.mean():.4f} ±
{scores_a.std():.4f}")
print(f"Конфигурация B (QuantileTransformer): {scores_b.mean():.4f} ±
{scores_b.std():.4f}")

# 3. Микро-suite валидации данных
def expect_not_null(s: pd.Series, name: str):
    bad = s[s.isna()]
    return bad.empty, bad.index.tolist()

def expect_ge(s: pd.Series, name: str, min_value: float):
    bad = s[s < min_value]
    return bad.empty, bad.index.tolist()

checks = {
    "adr_nonneg": expect_ge(df["adr"].fillna(0), "adr", 0.0),
    "adults_nonneg": expect_ge(df["adults"].fillna(0), "adults", 0.0),
    "lead_time_nonneg": expect_ge(df["lead_time"].fillna(0), "lead_time",
0.0),
    "hotel_not_null": expect_not_null(df["hotel"], "hotel")
}

print("\nРЕЗУЛЬТАТЫ ВАЛИДАЦИИ ДАННЫХ:")
validation_results = {key: ok for key, (ok, idx) in checks.items()}
for check_name, passed in validation_results.items():
    status = "ПРОЙДЕНА" if passed else "НЕ ПРОЙДЕНА"
    print(f"  {check_name}: {status}")
```

Б. Файлы конфигурации, ссылки на репозиторий:

https://colab.research.google.com/drive/1J1R51oLxP06qhk1DA66vA6NsNfmzetVG?usp=sharing