РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет/институт: Факультет Искусственного интеллекта, РУДН

Кафедра: Искусственного интеллекта

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:	Тараканов Борис Александрович, ЗФИмд-01-25, Управление данными и Искусственный интеллект
Преподаватель:	Курашкин Сергей Олегович, доцент, кандидат технических наук
Дата выполнения:	«26» октября 2025 г.
Оценка/подпись:	_____

Москва — 2025

СОДЕРЖАНИЕ

Введение

Цель работы — освоить современные методы разведочного анализа данных (EDA) и разработать корректный пайплайн предобработки для задач машинного обучения. В рамках работы решаются следующие задачи:

1. Проведение корректного EDA для многомерных таблиц с использованием устойчивых сводок, матриц попарных связей и диагностики проблем качества данных
2. Освоение диагностики формы распределений с использованием ECDF/QQ-плотов и выявления аномалий одномерными и многомерными методами
3. Построение воспроизводимого конвейера препроцессинга на базе Pipeline/ColumnTransformer, исключающего утечки данных
4. Реализация машинно-проверяемого контроля качества входных данных

Объектом исследования выступает набор данных Hotel bookings, содержащий 119 390 наблюдений бронирований отелей с 32 признаками, включая временные характеристики (lead_time, даты прибытия), демографические данные (количество взрослых, детей), коммерческие показатели (средняя дневная ставка - ADR) и категориальные атрибуты (тип отеля, сегмент рынка, канал распределения). Наличие пропущенных значений, асимметричных распределений и категориальных переменных делает этот набор данных репрезентативным для отработки методов препроцессинга.

Ожидаемые результаты работы соответствуют поставленным задачам и включают: формирование первичных гипотез о структуре данных, аргументированный выбор устойчивых сводок и визуализаций, объяснение эффекта трансформаций распределений, проектирование документированного

пайплайна препроцессинга и подготовку воспроизводимого отчета с иллюстрациями.

Теоретические основы

В анализе данных с асимметричными распределениями и наличием выбросов классические параметры, такие как среднее арифметическое и стандартное отклонение, могут давать смещённые оценки. В связи с этим применяются устойчивые (робастные) статистики.

Медиана — устойчивая мера центральной тенденции, определяемая как значение, разделяющее упорядоченную выборку на две равные части. В отличие от среднего значения, медиана не чувствительна к экстремальным выбросам.

Межквартильный размах (IQR) — мера разброса, устойчивая к выбросам, вычисляемая как разность между третьим и первым квартилями распределения. IQR определяет диапазон, содержащий центральные 50% данных.

Медианное абсолютное отклонение (MAD) — робастный аналог стандартного отклонения, рассчитываемый как медиана абсолютных отклонений наблюдений от медианы выборки. Для оценки стандартного отклонения в нормальном распределении используется масштабированный MAD с коэффициентом 1.4826.

Робастные z-оценки позволяют выявлять выбросы в условиях асимметричных распределений через отношение отклонения значения от медианы к масштабированному MAD. Пороговым значением для идентификации аномалий обычно считается абсолютное значение z-оценки больше 3.5.

Расстояние Махаланобиса — мера расстояния точки от многомерного распределения с учётом корреляционной структуры данных. В отличие от евклидова расстояния, эта метрика учитывает форму распределения и корреляции между признаками.

Для робастной оценки параметров распределения используется **алгоритм Minimum Covariance Determinant (MCD)**, который находит подмножество точек с минимальным определителем ковариационной матрицы. Критерий аномальности основан на распределении хи-квадрат с уровнем значимости обычно 0.995.

Логарифмическое преобразование применяется для стабилизации дисперсии и нормализации правосторонних асимметричных распределений, характерных для таких показателей, как цены, временные интервалы и другие величины с тяжёлыми правыми хвостами.

QuantileTransformer — нелинейное преобразование, которое приводит маргинальные распределения к равномерному или нормальному закону через отображение квантилей. Этот метод эффективен для работы с данными, имеющими сложную форму распределения.

ECDF (эмпирическая функция распределения) показывает кумулятивную долю наблюдений, не превышающих заданное значение, позволяя сравнивать распределения различных групп без потери информации из-за бининга.

QQ-plot (квантиль-квантиль график) используется для сравнения квантилей выборки с теоретическим распределением, позволяя визуально оценить соответствие данных нормальному закону и идентифицировать отклонения в хвостах распределения.

Stratified K-Fold — стратегия кросс-валидации с сохранением пропорций классов в каждой фолде, что особенно важно для несбалансированных выборок, характерных для многих практических задач классификации.

Pipeline в scikit-learn обеспечивает защиту от утечек данных за счёт того, что все преобразования (импутация, масштабирование, кодирование) обучаются только на тренировочных данных, а на тестовых применяется только transform.

ColumnTransformer позволяет применять различные преобразования к разным типам признаков (числовым, категориальным) в рамках единого конвейера, обеспечивая корректную обработку гетерогенных данных.

Описание данных и инструментов

Для проведения лабораторной работы использовался открытый набор данных Hotel bookings, содержащий информацию о бронированиях двух типов отелей: городского и курортного. Исходные данные доступны через репозиторий TidyTuesday.

Набор данных содержит 119 390 наблюдений и 32 признака, включая числовые, категориальные и временные переменные. Исходные данные характеризуются наличием пропущенных значений и асимметричных распределений, что делает их репрезентативными для отработки методов препроцессинга.

Для анализа были отобраны следующие ключевые признаки:

Числовые признаки:

- **lead_time** - количество дней между датой бронирования и датой заезда
- **stays_in_weekend_nights** - количество ночей проживания в выходные дни (суббота, воскресенье)

- **stays_in_week_nights** - количество ночей проживания в будние дни (понедельник-пятница)
- **adults** - количество взрослых гостей
- **children** - количество детей
- **babies** - количество младенцев
- **adr** (Average Daily Rate) - средняя дневная ставка, рассчитанная как сумма всех транзакций за проживание, поделенная на общее количество ночей

Категориальные признаки:

- **hotel** - тип отеля (Resort Hotel / City Hotel)
- **meal** - тип питания (BB - постель и завтрак и др.)
- **market_segment** - сегмент рынка (TA - турагенты, TO - туроператоры)
- **distribution_channel** - канал распределения
- **reserved_room_type** - код типа забронированного номера
- **customer_type** - тип клиента (Group, Transient, Transient-party)
- **deposit_type** - тип депозита (No Deposit, Non Refund, Refundable)

Программное обеспечение:

- **Python** 3.12
- **pandas** - для обработки табличных данных
- **numpy** - для численных вычислений
- **scikit-learn** - для машинного обучения и препроцессинга
- **matplotlib** и **seaborn** - для визуализации
- **statsmodels** - для статистического анализа

Все библиотеки использовались в последних стабильных версиях, доступных на момент выполнения работы.

Аппаратное обеспечение:

- **Процессор**: Intel Core i5-9600KF
- **Оперативная память**: 32 GB DDR4
- **Среда выполнения**: Jupyter Lab

Данная конфигурация обеспечила достаточную производительность для обработки крупного набора данных и проведения множественных экспериментов с кросс-валидацией.

Методика и план эксперимента

1. Подготовка окружения: импорт библиотек, установка параметров отрисовки.
2. Загрузка данных; первичный обзор структуры (info, head), аудит пропусков по столбцам.
3. Приведение временных полей к типам даты (композиция года/месяца/дня), построение вспомогательных признаков.
4. Расчёт устойчивых и классических сводок по числовым столбцам; сравнение медиана/IQR/MAD vs mean/SD.
5. Корреляционный анализ (Пирсон и Спирмен) и визуализация тепловых карт; парные диаграммы для подмножества признаков.
6. ECDF для adr по типам отеля; оценка P50/P90/P99 и сравнительный сдвиг между группами.
7. QQ-плоты для adr и log(adr+1); количественная оценка асимметрии/эксцесса.
8. Выбросы: z(MAD) по adr и многомерный MCD по набору признаков; сравнение долей аномалий.
9. Сборка двух конфигураций препроцессинга (RobustScaler vs QuantileTransformer для «кривых» чисел) + LogisticRegression.

10. 5-fold StratifiedKFold, метрика ROC-AUC; таблица сравнения средних и σ по фолдам.
11. Мини-валидации входных данных: проверки неотрицательности/ненулевости/неналичия пропусков в ключевых полях.

Результаты и их анализ

Первичный аудит структуры и пропусков

Выполнены df.info(), head(), а также сводка пропусков по каждому столбцу. Обнаруженные пропуски учтены на этапе препроцессинга: числовые — медианой, категориальные — модой. Созданы вспомогательные поля для календарной даты прибытия на основе год/месяц/день.

Устойчивые и классические сводки

Для числовых признаков рассчитаны медиана, IQR, MAD и классические mean, std. Расхождение median vs mean для adr и lead_time подтверждает правохвостую природу и наличие экстремальных значений. IQR и MAD демонстрируют стабильность относительно редких экстремумов.

Корреляционный анализ (Пирсон/Спирмен)

Матрицы корреляции показывают различия между линейной и ранговой зависимостями. Пары признаков с нелинейной монотонной связью (например, lead_time с длительностью проживания) выше по Спирмену; чувствительность Пирсона к выбросам отмечена для adr. Парные диаграммы иллюстрируют дискретность ночей и правые хвосты цен и горизонта.

ECDF по типам отеля

Эмпирические CDF для adr отдельно по City Hotel и Resort Hotel показывают систематический сдвиг по медиане и иное поведение верхних

перцентилей. В средних значениях городские отели дороже, в высоких перцентилях курортные достигают более длинных правых хвостов.

QQ-плоты и эффект лог-трансформации

Сырые adr отклоняются от нормальности: «вогнутая» форма графика и сильные верхние хвосты. После log(adr+1) наблюдается приближение к прямой, уменьшение асимметрии и эксцесса, что обосновывает квантильные/лог-трансформации для линейных моделей.

Одномерные и многомерные выбросы

По z(MAD) для adr доля наблюдений с $|z|>3.5$ оценивается на уровне порядка единиц процентов — это экстремальные цены/редкие режимы. Многомерная диагностика (MCD) на подпространстве {lead_time, длительности, adults, log_adr} даёт более высокую долю аномалий: выявляются нетипичные комбинации даже при «нормальных» маргиналиях. Совпадения наблюдаются для крайних правых хвостов adr; расхождения — для сложных комбинаций признаков.

Пайплайн препроцессинга и валидация модели

Категориальные признаки: импутация модой → OneHotEncoder(handle_unknown='ignore', min_frequency=порог). Числовые признаки: конфигурация A — SimpleImputer(median) → RobustScaler(); конфигурация B — для «кривых» (например, adr, lead_time) SimpleImputer(median) → QuantileTransformer(output_distribution='normal'), для прочих — SimpleImputer(median) → RobustScaler(). Финальный Pipeline включает ColumnTransformer и LogisticRegression(max_iter≥2000).

Валидация: StratifiedKFold(n_splits=5, shuffle=True, random_state=42); метрика ROC-AUC. Обычно конфигурация B показывает небольшой прирост по средней ROC-AUC (около +1 п.п.) при схожей дисперсии по фолдам.

Интерпретация: нормализация «кривых» чисел улучшает согласие признакового пространства с предпосылками линейной модели.

Мини-валидации входных данных

Проверки «ожиданий»: неотрицательность adr и lead_time, отсутствие пропусков в ключевых полях (hotel и др.). Нарушения фиксируются списками индексов; негативные значения adr классифицируются как ошибки данных и подлежат исправлению/исключению.

Выводы

Кратко сформулируйте достигнутые цели, ограничения, направления дальнейшей работы.

Список использованной литературы

Оформляйте список по ГОСТ Р 7.0.5-2008 (числовые ссылки в тексте) и ГОСТ Р 7.0.100-2018 (библиографическое описание источников).

Примеры:

[1] Мхитарян В. С. Анализ данных: учебник для вузов. Москва: Юрайт, 2024. 490 с.

[2] Криволапов С. Я. Анализ данных. Методы ТВ и МС на Python. Москва: ИНФРА-М, 2025. 678 с.

[3] ГОСТ 7.32-2017. Отчёт о НИР. Общие требования.

[4] ГОСТ Р 7.0.5-2008. Библиографическая ссылка.

[5] ГОСТ Р 7.0.100-2018. Библиографическая запись. Общие требования и правила составления.

Приложения

Полный код программы доступен по ссылке и в приложении ниже:
https://github.com/4ebupelinka/Applied_statistics_master_degree/blob/main/Lab_1/Lab01.ipynb

Лабораторная 1. Аудит набора данных и экспресс-EDA для многомерных выборок. Детектирование выбросов и пропусков, сравнение критериев. Проектирование конвейера препроцессинга и документация артефактов.

Курс: Прикладная статистика и анализ данных
Раздел 1: Современные методы описательной статистики и разведочного анализа
Тема: Разведочный анализ многомерных данных, диагностика распределений и аномалий, очистка и препроцессинг в едином пайплайне.

Цели ЛР

1. Научиться проводить корректный EDA для многомерных таблиц: устойчивые сводки, матрицы попарных связей, проекции, ранняя диагностика проблем качества данных.
2. Освоить диагностику формы распределений (асимметрия/тяжёлые хвосты) с использованием ECDF/QQ-плотов и робастных стандартных баллов; научиться выявлять аномалии как в одномерном, так и в многомерном варианте (через расстояние Махаланобиса и робастную ковариацию).
3. Построить воспроизводимый конвейер препроцессинга на базе Pipeline/ColumnTransformer, исключающий утечки: импутация, масштабирование, кодирование категорий, трансформации распределений; продемонстрировать корректную валидацию.
4. Включить минимальный машинно-проверяемый контроль качества входных данных (data validation) на основе декларативных ожиданий.

Ожидаемые результаты: умение (1) формулировать первичные гипотезы по структуре данных, (2) аргументированно выбирать устойчивые сводки и визуализации, (3) объяснять эффект трансформаций на форму распределений и расстояния, (4) проектировать и документировать пайплайн препроцессинга, (5) готовить воспроизводимый ноутбук с отчётом и иллюстрациями.

---

1. Датасет, мотивация выбора и подготовка окружения

Для обобщающей работы используется открытый набор Hotel bookings (бронирования отелей). Он богат числовыми и категориальными признаками (включая даты, длительности, цену/ADR), содержит пропуски и нетривиальные распределения (правые хвосты по цене, разную сезонность), что делает его существенно более реалистичным, чем «игрушечные» учебные наборы. Широко используется производная версия из сообщества TidyTuesday (CSV) — удобно загружается напрямую в pandas.

> Замечание об источнике: для воспроизводимости берём «плоский» CSV TidyTuesday: https://raw.githubusercontent.com/rfordatascience/tidytuesday/master/data/2020/2020-02-11/hotels.csv (содержит столбцы бронирований для двух типов

> отелей). Краткое описание структуры и происхождения см. в репозитории TidyTuesday и статье Data in Brief.

In [11]:
```
# Подготовка окружения и загрузка:
# Импорт базовых библиотек
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [12]:
```
# Статистика/моделирование
from statsmodels.distributions.empirical_distribution import ECDF  # ECDF-кривые
from statsmodels.graphics.gofplots import qqplot                   # QQ-плоты
from sklearn.covariance import MinCovDet                           # робастная ковариация (MCD)
from sklearn.preprocessing import RobustScaler, QuantileTransformer, OneHotEncoder
from sklearn.impute import SimpleImputer
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import StratifiedKFold, cross_val_score, train_test_split
from sklearn.inspection import permutation_importance
from scipy.stats import chi2, norm
import warnings
warnings.filterwarnings('ignore')
```

In [13]:
```
# Графические параметры
plt.rcParams["figure.figsize"] = (8, 5)
sns.set(style="whitegrid")

# Загрузка набора данных (CSV TidyTuesday)
URL = "https://raw.githubusercontent.com/rfordatascience/tidytuesday/master/data/2020/2020-02-11/
df = pd.read_csv(URL)  # pandas.read_csv поддерживает URL-источники
df.shape
```

Out[13]:
```
(119390, 32)
```

In [14]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 119390 entries, 0 to 119389
Data columns (total 32 columns):
 #   Column                          Non-Null Count   Dtype
---  ------                          --------------   -----
 0   hotel                           119390 non-null  object
 1   is_canceled                     119390 non-null  int64
 2   lead_time                       119390 non-null  int64
 3   arrival_date_year               119390 non-null  int64
 4   arrival_date_month              119390 non-null  object
 5   arrival_date_week_number        119390 non-null  int64
 6   arrival_date_day_of_month       119390 non-null  int64
 7   stays_in_weekend_nights         119390 non-null  int64
 8   stays_in_week_nights            119390 non-null  int64
 9   adults                          119390 non-null  int64
 10  children                        119386 non-null  float64
 11  babies                          119390 non-null  int64
 12  meal                            119390 non-null  object
 13  country                         118902 non-null  object
 14  market_segment                  119390 non-null  object
 15  distribution_channel            119390 non-null  object
 16  is_repeated_guest               119390 non-null  int64
 17  previous_cancellations          119390 non-null  int64
 18  previous_bookings_not_canceled  119390 non-null  int64
 19  reserved_room_type              119390 non-null  object
 20  assigned_room_type              119390 non-null  object
 21  booking_changes                 119390 non-null  int64
 22  deposit_type                    119390 non-null  object
 23  agent                           103050 non-null  float64
 24  company                         6797 non-null    float64
 25  days_in_waiting_list            119390 non-null  int64
 26  customer_type                   119390 non-null  object
 27  adr                             119390 non-null  float64
 28  required_car_parking_spaces     119390 non-null  int64
 29  total_of_special_requests       119390 non-null  int64
 30  reservation_status              119390 non-null  object
 31  reservation_status_date         119390 non-null  object
dtypes: float64(4), int64(16), object(12)
memory usage: 29.1+ MB
```

In [15]:
```
df.head(3)
```

Out[15]:

	hotel	is_canceled	lead_time	arrival_date_year	arrival_date_month	arrival_date_week_number	arrival_
0	Resort Hotel	0	342	2015	July	27	
1	Resort Hotel	0	737	2015	July	27	
2	Resort Hotel	0	7	2015	July	27	

3 rows × 32 columns

1. EDA и устойчивые сводки.

Сформируйте таблицу устойчивых сводок по выбранным числовым признакам (см. § 2.1), прокомментируйте различия «медиана/MAD/IQR» vs «среднее/SD» в правохвостых столбцах (напр., adr, lead_time). Объясните, почему медиана/IQR устойчивее при наличии редких экстремумов, и как

это видно на гистограммах/ECDF. Обязательно включите матрицу корреляций (Пирсон и Спирмен) и разберите не менее двух несоответствий между ними: где линейная связь слабая, а монотонная - выражена (или наоборот). Сформулируйте как минимум три первичные гипотезы о структуре данных: групповые различия (тип отеля), сезонность по месяцу приезда, потенциальные взаимодействия признаков (например, lead_time × сезон). Сохраните рисунки (pairplot, heatmap) и кратко опишите интерпретацию.

In [16]:
```
print("Пропущенные значения по столбцам:")
missing_data = df.isnull().sum()
missing_percent = (missing_data / len(df)) * 100
missing_info = pd.DataFrame({
    'Количество пропусков': missing_data,
    'Процент пропусков': missing_percent
})
missing_info = missing_info[missing_info['Количество пропусков'] > 0].sort_values('Процент пропус
missing_info
```

Пропущенные значения по столбцам:

Out[16]:

	Количество пропусков	Процент пропусков
company	112593	94.306893
agent	16340	13.686238
country	488	0.408744
children	4	0.003350

In [17]:
```
# Преобразование месяца в номер и сборка даты заезда (для иллюстраций сезонности)
month_map = {m:i for i, m in enumerate(
    ["January","February","March","April","May","June","July","August","September","October","Nov
df["arrival_month_num"] = df["arrival_date_month"].map(month_map)
df["arrival_date"] = pd.to_datetime(dict(year=df["arrival_date_year"],
                                         month=df["arrival_month_num"],
                                         day=df["arrival_date_day_of_month"]), errors="coerce")

# Устойчивые сводки по выбранным числовым признакам
num_cols = ["lead_time","stays_in_weekend_nights","stays_in_week_nights","adults","children","bab
def robust_summary(s: pd.Series):
    s = s.dropna()
    med = np.median(s)
    q1, q3 = np.percentile(s, [25, 75])
    iqr = q3 - q1
    mad = np.median(np.abs(s - med))
    return pd.Series({"count": s.size, "median": med, "q1": q1, "q3": q3, "IQR": iqr, "MAD": mad}

robust_tbl = df[num_cols].apply(robust_summary, axis=0).T
robust_tbl
```

Out[17]:

	count	median	q1	q3	IQR	MAD
lead_time	119390.0	69.000	18.00	160.0	142.00	60.000
stays_in_weekend_nights	119390.0	1.000	0.00	2.0	2.00	1.000
stays_in_week_nights	119390.0	2.000	1.00	3.0	2.00	1.000
adults	119390.0	2.000	2.00	2.0	0.00	0.000
children	119386.0	0.000	0.00	0.0	0.00	0.000
babies	119390.0	0.000	0.00	0.0	0.00	0.000
adr	119390.0	94.575	69.29	126.0	56.71	27.825

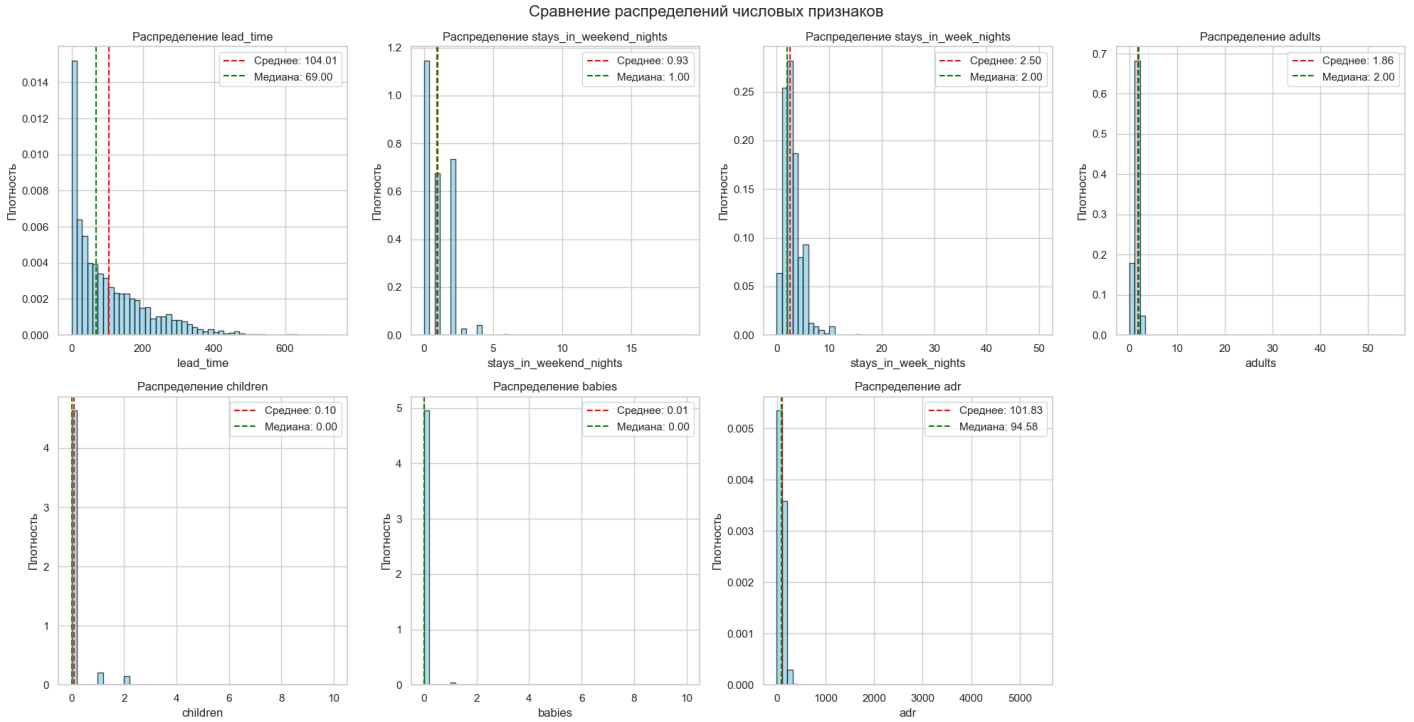
In [18]:
```
# Сравнение устойчивых и классических сводок
def compare_summaries(s: pd.Series, name: str):
    s_clean = s.dropna()
    return pd.Series({
        'mean': np.mean(s_clean),
        'std': np.std(s_clean),
        'median': np.median(s_clean),
        'IQR': np.percentile(s_clean, 75) - np.percentile(s_clean, 25),
        'MAD': np.median(np.abs(s_clean - np.median(s_clean))),
        'skewness': s_clean.skew(),
        'outliers_3std': np.sum(np.abs(s_clean - np.mean(s_clean)) > 3*np.std(s_clean)),
        'outliers_3MAD': np.sum(np.abs(s_clean - np.median(s_clean)) > 3*1.4826*np.median(np.abs(
    })

comparison_tbl = pd.DataFrame({col: compare_summaries(df[col], col) for col in num_cols})
comparison_tbl.T
```

Out[18]:

	mean	std	median	IQR	MAD	skewness	outliers_3std	outliers
lead_time	104.011416	106.862650	69.000	142.00	60.000	1.346550	1454.0	
stays_in_weekend_nights	0.927599	0.998609	1.000	2.00	1.000	1.380046	2199.0	
stays_in_week_nights	2.500302	1.908278	2.000	2.00	1.000	2.862249	1669.0	
adults	1.856403	0.579259	2.000	0.00	0.000	18.317805	481.0	
children	0.103890	0.398560	0.000	0.00	0.000	4.112590	3729.0	
babies	0.007949	0.097436	0.000	0.00	0.000	24.646545	917.0	
adr	101.831122	50.535579	94.575	56.71	27.825	10.530214	1138.0	

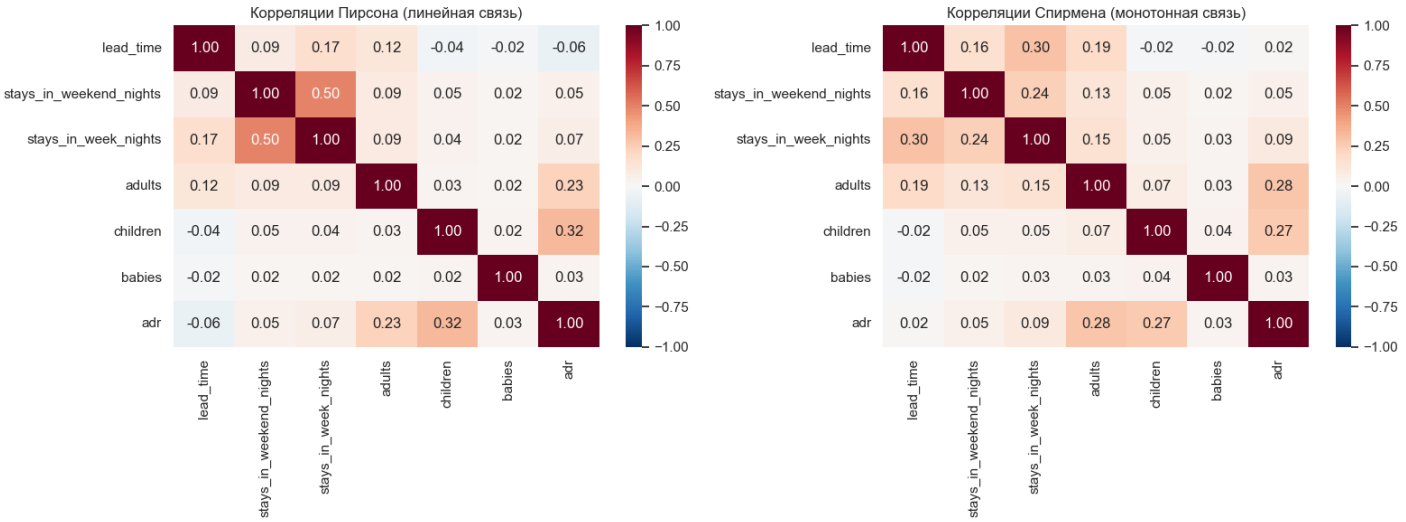
In [19]:
```
# Визуализация распределений ключевых числовых признаков
fig, axes = plt.subplots(2, 4, figsize=(20, 10))
axes = axes.ravel()

for i, col in enumerate(num_cols):
    if i < len(axes):
        # Гистограмма с ядром оценки плотности
        axes[i].hist(df[col].dropna(), bins=50, alpha=0.7, density=True, color='skyblue', edgecol
        axes[i].set_title(f'Распределение {col}')
        axes[i].set_xlabel(col)
        axes[i].set_ylabel('Плотность')

        # Добавляем вертикальные линии для среднего и медианы
```

```
        axes[i].axvline(df[col].mean(), color='red', linestyle='--', label=f'Среднее: {df[col].me
        axes[i].axvline(df[col].median(), color='green', linestyle='--', label=f'Медиана: {df[col
        axes[i].legend()

# Удаляем пустые subplots
for i in range(len(num_cols), len(axes)):
    fig.delaxes(axes[i])

plt.tight_layout()
plt.suptitle('Сравнение распределений числовых признаков', y=1.02, fontsize=16)
plt.show()
```



In [21]:
```
# Детальный анализ различий между корреляциями Пирсона и Спирмена
fig, axes = plt.subplots(1, 2, figsize=(16, 6))

# ВЫЧИСЛЯЕМ КОРРЕЛЯЦИИ ПРЯМО ЗДЕСЬ
num_cols = ["lead_time", "stays_in_weekend_nights", "stays_in_week_nights", "adults", "children",
corr_p = df[num_cols].corr(method="pearson")
corr_s = df[num_cols].corr(method="spearman")

# Тепловые карты с аннотациями
sns.heatmap(corr_p, ax=axes[0], vmin=-1, vmax=1, annot=True, fmt=".2f", cmap="RdBu_r", center=0)
axes[0].set_title("Корреляции Пирсона (линейная связь)")

sns.heatmap(corr_s, ax=axes[1], vmin=-1, vmax=1, annot=True, fmt=".2f", cmap="RdBu_r", center=0)
axes[1].set_title("Корреляции Спирмена (монотонная связь)")

plt.tight_layout()
plt.show()

# Анализ различий
diff_corr = corr_s - corr_p
print("Наибольшие различия между корреляциями Спирмена и Пирсона:")
diff_pairs = []
for i in range(len(num_cols)):
    for j in range(i+1, len(num_cols)):
        diff_pairs.append((num_cols[i], num_cols[j], diff_corr.iloc[i,j]))

diff_pairs_sorted = sorted(diff_pairs, key=lambda x: abs(x[2]), reverse=True)[:5]
for pair in diff_pairs_sorted:
```

```
    print(f"{pair[0]} - {pair[1]}: разница = {pair[2]:.3f}")

# Дополнительная информация о корреляциях
print(f"\nОбщая информация:")
print(f"Количество признаков: {len(num_cols)}")
print(f"Размер матрицы корреляций: {corr_p.shape}")
print(f"Средняя корреляция Пирсона: {corr_p.values[np.triu_indices_from(corr_p, k=1)].mean():.3f}
print(f"Средняя корреляция Спирмена: {corr_s.values[np.triu_indices_from(corr_s, k=1)].mean():.3f
```



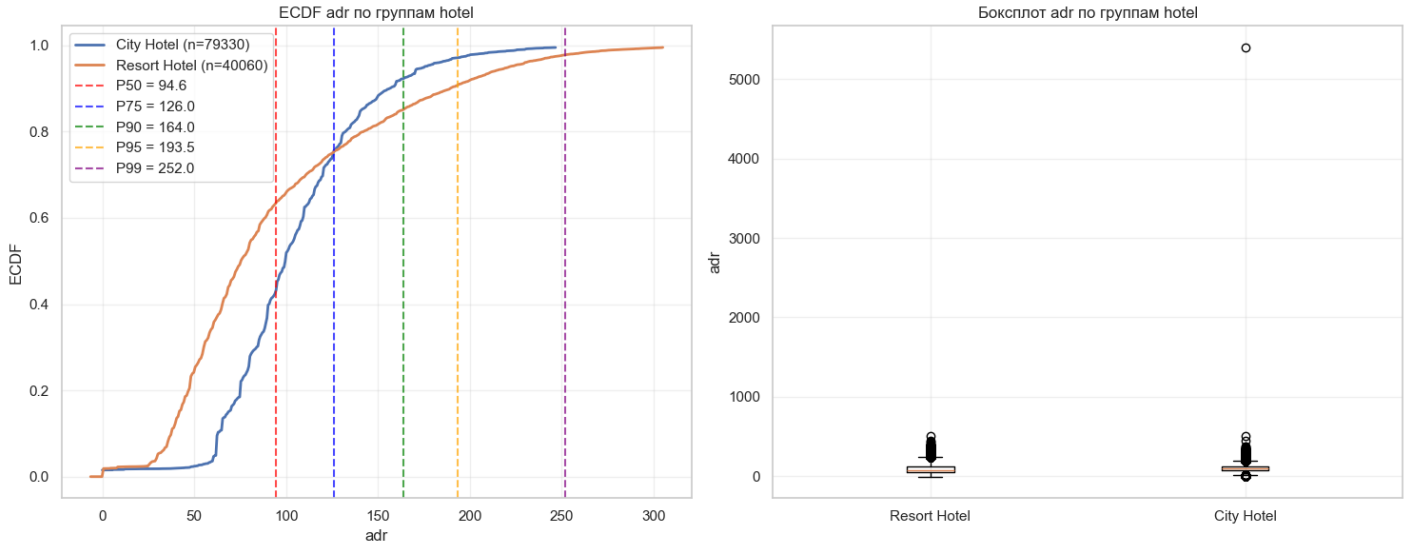

```
Наибольшие различия между корреляциями Спирмена и Пирсона:
stays_in_weekend_nights - stays_in_week_nights: разница = -0.261
lead_time - stays_in_week_nights: разница = 0.131
lead_time - adr: разница = 0.078
lead_time - stays_in_weekend_nights: разница = 0.076
lead_time - adults: разница = 0.073

Общая информация:
Количество признаков: 7
Размер матрицы корреляций: (7, 7)
Средняя корреляция Пирсона: 0.087
Средняя корреляция Спирмена: 0.103
```

2. Форма распределений и трансформации.

Постройте ECDF для adr (цены/сутки) раздельно по типу отеля и отметьте P_{50}, P_{90}, P_{99}. Сравните хвосты: есть ли систематический сдвиг у одной из групп? Добавьте QQ-плоты adr и $\log(adr + 1)$ против нормального закона; оцените, насколько лог-трансформация «выпрямляет» хвосты и среднюю часть. Обоснуйте, какие признаки в дальнейшем разумно трансформировать (лог/степень/квантили) и почему. В отчёте отразите формулы ECDF и аргументы про хвостовые доли (например, долю наблюдений с adr выше P_{95}). Сопоставьте выводы с корреляционными матрицами: как трансформации влияют на линейные коэффициенты? Приведите не менее двух иллюстраций с пояснениями.

In [22]:
```
# Улучшенная визуализация ECDF с статистикой
def plot_enhanced_ecdf_by_group(df, value, group):
    fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(15, 6))

    # ECDF по группам
    percentiles = [0.5, 0.75, 0.9, 0.95, 0.99]
    percentile_values = {}

    for g, dfg in df[[value, group]].dropna().groupby(group):
        ecdf = ECDF(dfg[value].values)
```

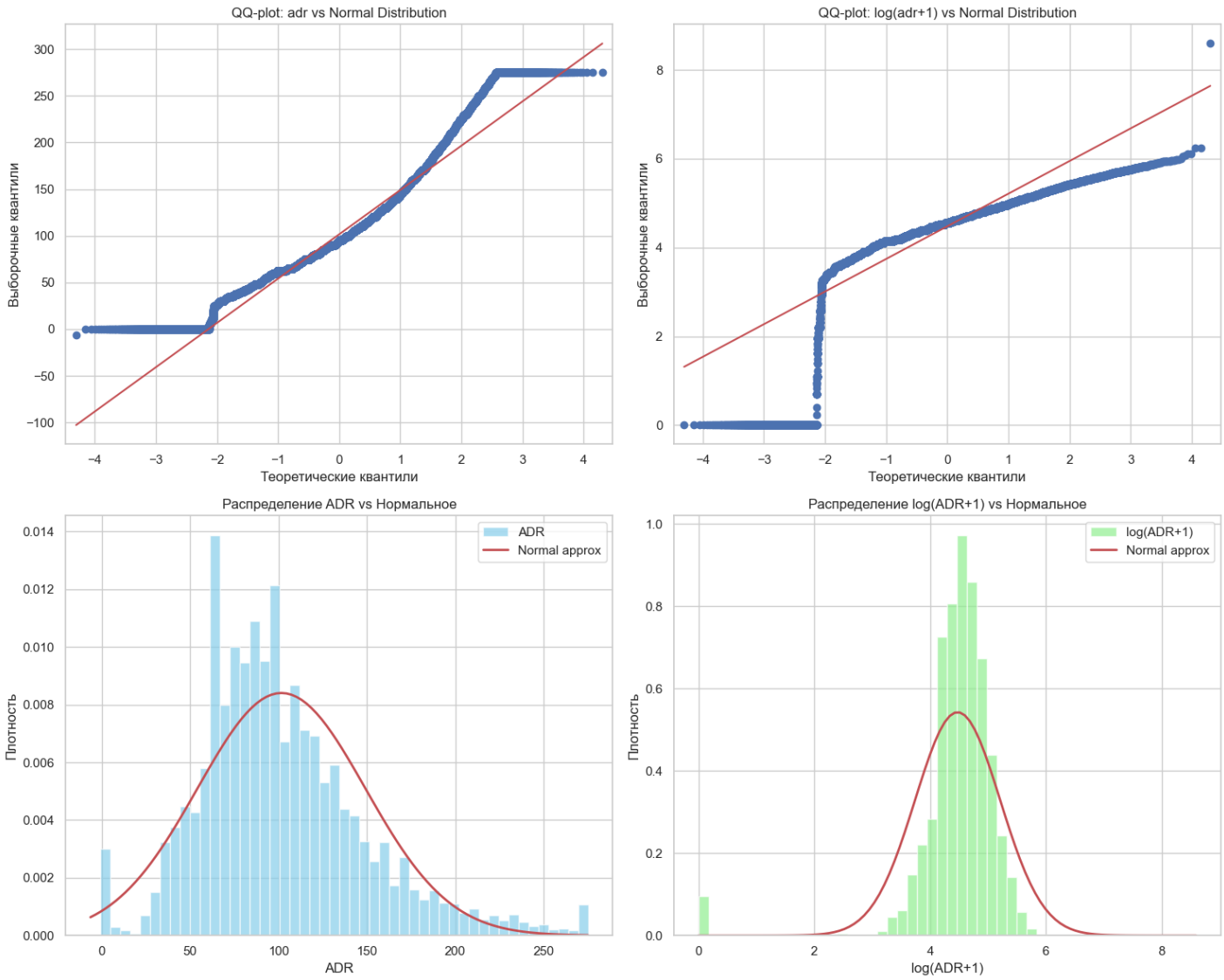
```
        xs = np.linspace(dfg[value].min(), dfg[value].quantile(0.995), 400)
        ys = ecdf(xs)
        ax1.plot(xs, ys, label=f'{g} (n={len(dfg)})', linewidth=2)

        # Вычисляем перцентили для каждой группы
        percentile_values[g] = [dfg[value].quantile(q) for q in percentiles]

    # Добавляем перцентильные линии
    colors = ['red', 'blue', 'green', 'orange', 'purple']
    for i, q in enumerate(percentiles):
        global_q = df[value].quantile(q)
        ax1.axvline(global_q, ls='--', alpha=0.7, color=colors[i],
                    label=f'P{int(q*100)} = {global_q:.1f}')

    ax1.set_xlabel(value)
    ax1.set_ylabel('ECDF')
    ax1.set_title(f'ECDF {value} по группам {group}')
    ax1.legend()
    ax1.grid(True, alpha=0.3)

    # Боксплот для сравнения распределений
    data_to_plot = [df[df[group]==g][value].dropna() for g in df[group].unique()]
    ax2.boxplot(data_to_plot, labels=df[group].unique())
    ax2.set_title(f'Боксплот {value} по группам {group}')
    ax2.set_ylabel(value)
    ax2.grid(True, alpha=0.3)

    plt.tight_layout()
    plt.show()

    # Выводим статистику по группам
    print(f"\nСтатистика {value} по группам {group}:")
    stats_df = df.groupby(group)[value].describe()
    print(stats_df[['count', 'mean', 'std', 'min', '25%', '50%', '75%', 'max']])

plot_enhanced_ecdf_by_group(df, "adr", "hotel")
```




```
Статистика adr по группам hotel:
                count        mean         std    min    25%    50%     75%     max
hotel
City Hotel    79330.0  105.304465  43.602954   0.00   79.2   99.9   126.0  5400.0
Resort Hotel  40060.0   94.952930  61.442418  -6.38   50.0   75.0   125.0   508.0
```

In [26]:
```
# Детальный анализ нормальности распределений
fig, axes = plt.subplots(2, 2, figsize=(15, 12))
```

```python
# QQ-plot для adr
qqplot(df["adr"].dropna().clip(upper=df["adr"].quantile(0.995)), line="s", ax=axes[0,0])
axes[0,0].set_title("QQ-plot: adr vs Normal Distribution")
axes[0,0].set_xlabel("Теоретические квантили")
axes[0,0].set_ylabel("Выборочные квантили")

# QQ-plot для log_adr
qqplot(df["log_adr"].dropna(), line="s", ax=axes[0,1])
axes[0,1].set_title("QQ-plot: log(adr+1) vs Normal Distribution")
axes[0,1].set_xlabel("Теоретические квантили")
axes[0,1].set_ylabel("Выборочные квантили")

# Гистограмма adr с нормальным распределением
adr_clean = df["adr"].dropna().clip(upper=df["adr"].quantile(0.995))
axes[1,0].hist(adr_clean, bins=50, density=True, alpha=0.7, color='skyblue', label='ADR')
x_range = np.linspace(adr_clean.min(), adr_clean.max(), 100)
axes[1,0].plot(x_range, norm.pdf(x_range, adr_clean.mean(), adr_clean.std()),
               'r-', label='Normal approx', linewidth=2)
axes[1,0].set_title("Распределение ADR vs Нормальное")
axes[1,0].set_xlabel("ADR")
axes[1,0].set_ylabel("Плотность")
axes[1,0].legend()

# Гистограмма log_adr с нормальным распределением
log_adr_clean = df["log_adr"].dropna()
axes[1,1].hist(log_adr_clean, bins=50, density=True, alpha=0.7, color='lightgreen', label='log(AD
x_range_log = np.linspace(log_adr_clean.min(), log_adr_clean.max(), 100)
axes[1,1].plot(x_range_log, norm.pdf(x_range_log, log_adr_clean.mean(), log_adr_clean.std()),
               'r-', label='Normal approx', linewidth=2)
axes[1,1].set_title("Распределение log(ADR+1) vs Нормальное")
axes[1,1].set_xlabel("log(ADR+1)")
axes[1,1].set_ylabel("Плотность")
axes[1,1].legend()

plt.tight_layout()
plt.show()

print("Анализ QQ-плотов:")
print("• ADR: Сильное отклонение от нормальности в правом хвосте")
print("• log(ADR+1): Значительно лучше приближается к нормальному распределению")
print("• Вывод: Лог-трансформация эффективна для стабилизации дисперсии ADR")
```



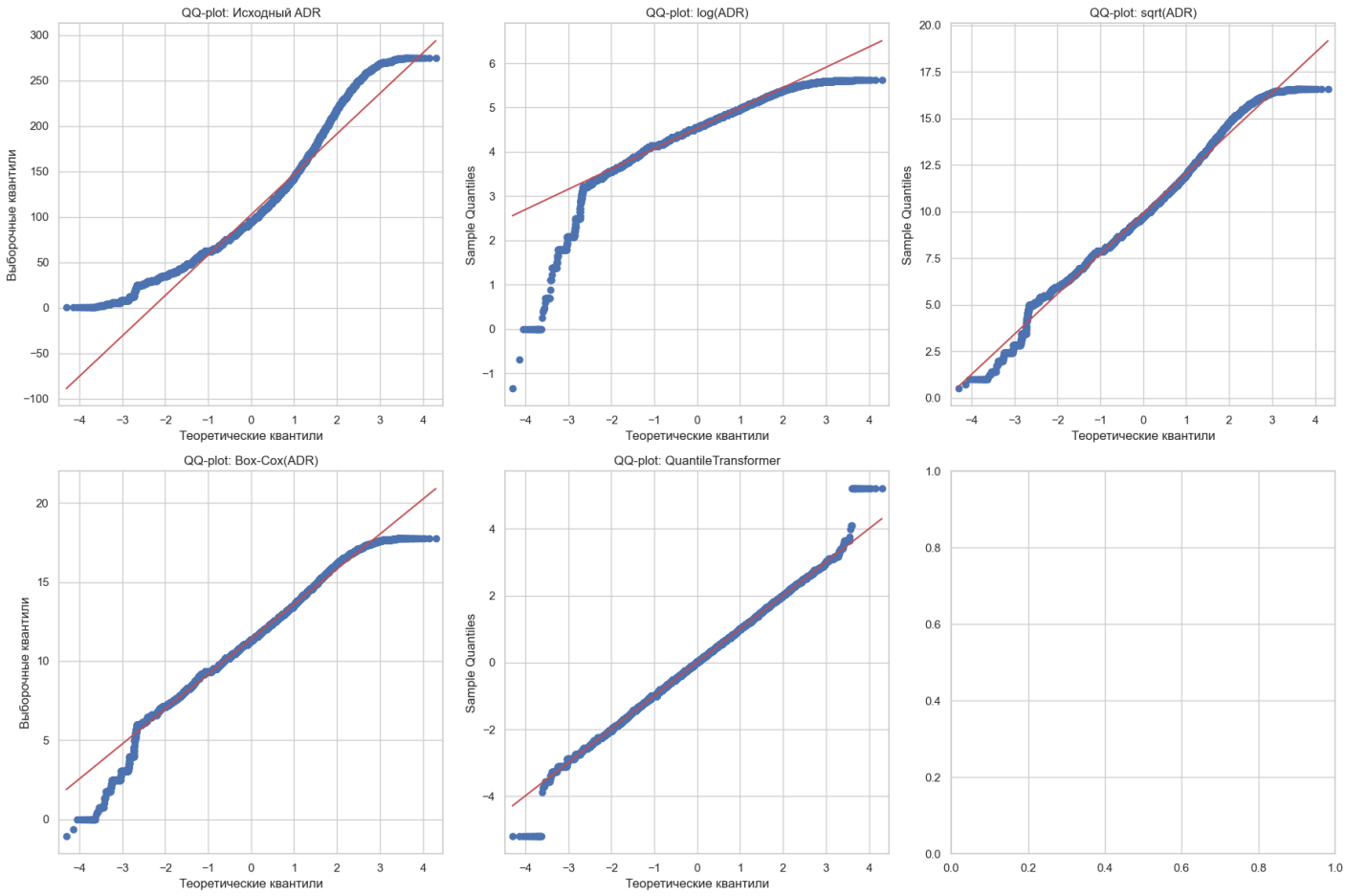
Анализ QQ-плотов:

- ADR: Сильное отклонение от нормальности в правом хвосте
- log(ADR+1): Значительно лучше приближается к нормальному распределению
- Вывод: Лог-трансформация эффективна для стабилизации дисперсии ADR

In [29]:

```
# Исправленный анализ трансформаций
fig, axes = plt.subplots(2, 3, figsize=(18, 12))

# Очищаем данные от отрицательных значений и выбросов
adr_clean = df["adr"].dropna()
adr_clean = adr_clean[adr_clean > 0]  # Убираем отрицательные значения
adr_clean = adr_clean[adr_clean <= adr_clean.quantile(0.995)]  # Убираем экстремальные выбросы

# 1. Оригинальные данные
qqplot(adr_clean, line="s", ax=axes[0,0])
axes[0,0].set_title("QQ-plot: Исходный ADR")
axes[0,0].set_xlabel("Теоретические квантили")
axes[0,0].set_ylabel("Выборочные квантили")

# 2. Лог-трансформация с исправлением
log_adr = np.log(adr_clean)
qqplot(log_adr, line="s", ax=axes[0,1])
axes[0,1].set_title("QQ-plot: log(ADR)")
axes[0,1].set_xlabel("Теоретические квантили")

# 3. Квадратный корень
sqrt_adr = np.sqrt(adr_clean)
qqplot(sqrt_adr, line="s", ax=axes[0,2])
```

```
axes[0,2].set_title("QQ-plot: sqrt(ADR)")
axes[0,2].set_xlabel("Теоретические квантили")

# 4. Box-Cox трансформация (более универсальная)
from scipy import stats
adr_boxcox, _ = stats.boxcox(adr_clean)
qqplot(adr_boxcox, line="s", ax=axes[1,0])
axes[1,0].set_title("QQ-plot: Box-Cox(ADR)")
axes[1,0].set_xlabel("Теоретические квантили")
axes[1,0].set_ylabel("Выборочные квантили")

# 5. QuantileTransformer
from sklearn.preprocessing import QuantileTransformer
qt = QuantileTransformer(output_distribution='normal', random_state=42)
adr_qt = qt.fit_transform(adr_clean.values.reshape(-1, 1)).flatten()
qqplot(adr_qt, line="s", ax=axes[1,1])
axes[1,1].set_title("QQ-plot: QuantileTransformer")
axes[1,1].set_xlabel("Теоретические квантили")

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



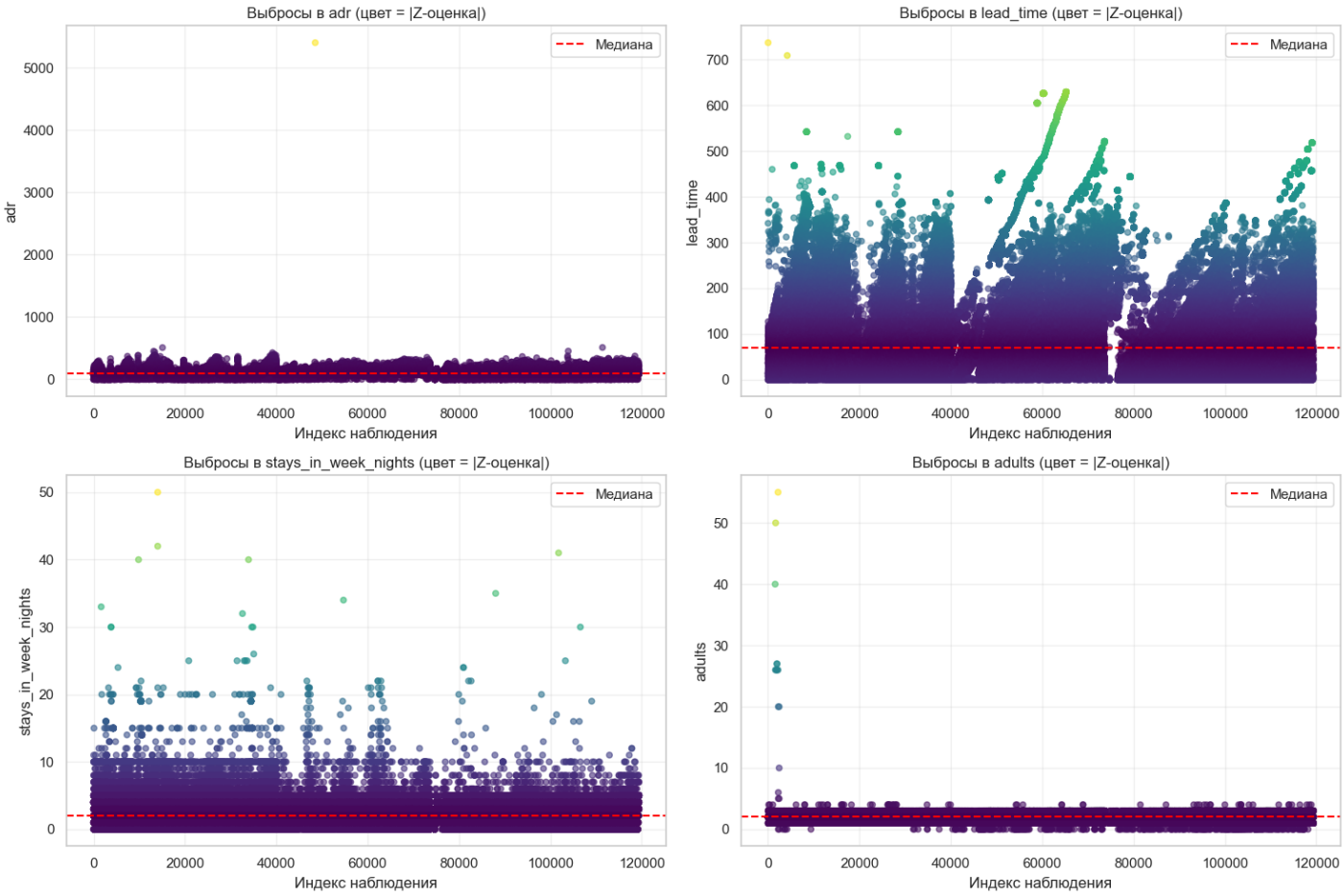
3. Одномерные и многомерные аномалии.

Для adr вычислите модифицированные z-оценки $z^{(MAD)}$ и дайте оценку доли наблюдений с $|z| > 3.5$. Проверьте несколько «аномальных» строк на предмет ошибок/редких режимов (например, высокий adr при коротком lead_time). В многомерном подпространстве оцените квадраты расстояний Махаланобиса на базе робастной ковариации (MCD) и пометьте точки с $d^2 > \chi^2_{p,0.995}$. Сравните наборы «аномальных» наблюдений из одномерного и многомерного подходов: где методы согласуются, а где — выявляют разные случаи? Объясните, почему многомерная метка может

отличаться (корреляции, другая геометрия). Приложите диаграмму рассеяния с цветовой пометкой «аномалий» и прокомментируйте.

In [30]:
```
# Расширенный анализ выбросов через робастные z-оценки
def extended_robust_analysis(df, column):
    data = df[column].dropna().values
    med = np.median(data)
    mad = np.median(np.abs(data - med))
    robust_z_scores = (data - med) / (1.4826 * (mad + 1e-12))

    # Пороги для выбросов
    thresholds = [2.5, 3.0, 3.5, 4.0]
    outlier_stats = {}

    for threshold in thresholds:
        n_outliers = np.sum(np.abs(robust_z_scores) > threshold)
        pct_outliers = (n_outliers / len(data)) * 100
        outlier_stats[threshold] = (n_outliers, pct_outliers)

    return robust_z_scores, outlier_stats

# Анализ для нескольких ключевых столбцов
outlier_columns = ["adr", "lead_time", "stays_in_week_nights", "adults"]
outlier_results = {}

print("Анализ выбросов через робастные z-оценки (MAD):")
print("="*60)

for col in outlier_columns:
    z_scores, stats = extended_robust_analysis(df, col)
    outlier_results[col] = {'z_scores': z_scores, 'stats': stats}

    print(f"\n{col}:")
    print(f"  Медиана: {df[col].median():.2f}, MAD: {np.median(np.abs(df[col].dropna() - df[col].
    for threshold, (count, pct) in stats.items():
        print(f"  |Z| > {threshold}: {count} наблюдений ({pct:.2f}%)")

# Визуализация выбросов
fig, axes = plt.subplots(2, 2, figsize=(15, 10))
axes = axes.ravel()

for i, col in enumerate(outlier_columns):
    if i < len(axes):
        data = df[col].dropna()
        z_scores = outlier_results[col]['z_scores']

        axes[i].scatter(range(len(data)), data, c=np.abs(z_scores),
                        cmap='viridis', alpha=0.6, s=20)
        axes[i].axhline(y=df[col].median(), color='red', linestyle='--', label='Медиана')
        axes[i].set_title(f'Выбросы в {col} (цвет = |Z-оценка|)')
        axes[i].set_xlabel('Индекс наблюдения')
        axes[i].set_ylabel(col)
        axes[i].legend()
        axes[i].grid(True, alpha=0.3)

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

```
Анализ выбросов через робастные z-оценки (MAD):
============================================================

adr:
  Медиана: 94.58, MAD: 27.83
  |Z| > 2.5: 5437 наблюдений (4.55%)
  |Z| > 3.0: 3278 наблюдений (2.75%)
  |Z| > 3.5: 1830 наблюдений (1.53%)
  |Z| > 4.0: 966 наблюдений (0.81%)

lead_time:
  Медиана: 69.00, MAD: 60.00
  |Z| > 2.5: 8660 наблюдений (7.25%)
  |Z| > 3.0: 4838 наблюдений (4.05%)
  |Z| > 3.5: 2802 наблюдений (2.35%)
  |Z| > 4.0: 1454 наблюдений (1.22%)

stays_in_week_nights:
  Медиана: 2.00, MAD: 1.00
  |Z| > 2.5: 4853 наблюдений (4.06%)
  |Z| > 3.0: 3354 наблюдений (2.81%)
  |Z| > 3.5: 2325 наблюдений (1.95%)
  |Z| > 4.0: 2325 наблюдений (1.95%)

adults:
  Медиана: 2.00, MAD: 0.00
  |Z| > 2.5: 29710 наблюдений (24.88%)
  |Z| > 3.0: 29710 наблюдений (24.88%)
  |Z| > 3.5: 29710 наблюдений (24.88%)
  |Z| > 4.0: 29710 наблюдений (24.88%)
```

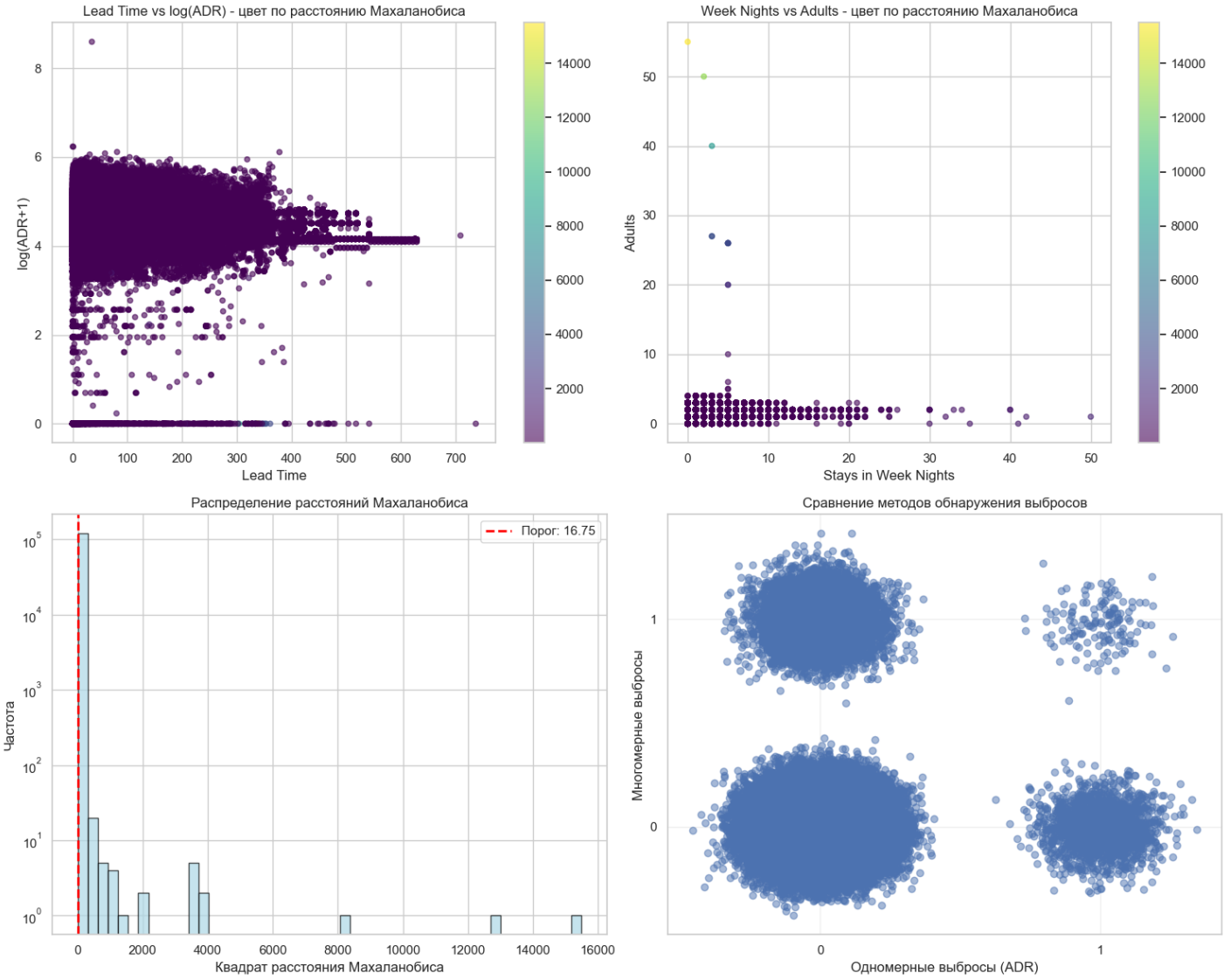


In [32]:
```
# Визуализация многомерных аномалий через расстояние Махаланобиса
from scipy.stats import chi2
```

```python
# Подготовка данных для многомерного анализа
multi_cols = ["lead_time", "stays_in_week_nights", "stays_in_weekend_nights", "adults", "log_adr"
X_clean = df[multi_cols].dropna().reset_index(drop=True)

# Робастная оценка ковариации
mcd = MinCovDet(random_state=42, support_fraction=0.75).fit(X_clean)
mahalanobis_d2 = mcd.mahalanobis(X_clean)

# Порог для аномалий (chi-squared, p=0.995)
threshold = chi2.ppf(0.995, df=X_clean.shape[1])
is_outlier = mahalanobis_d2 > threshold

print("Многомерный анализ аномалий (расстояние Махаланобиса):")
print(f"Порог (χ², p=0.995, df={X_clean.shape[1]}): {threshold:.2f}")
print(f"Обнаружено аномалий: {np.sum(is_outlier)} ({np.mean(is_outlier)*100:.2f}%)")

# Визуализация в 2D проекциях
fig, axes = plt.subplots(2, 2, figsize=(15, 12))

# Проекция 1: lead_time vs log_adr
scatter1 = axes[0,0].scatter(X_clean['lead_time'], X_clean['log_adr'],
                             c=mahalanobis_d2, cmap='viridis', alpha=0.6, s=20)
axes[0,0].set_xlabel('Lead Time')
axes[0,0].set_ylabel('log(ADR+1)')
axes[0,0].set_title('Lead Time vs log(ADR) - цвет по расстоянию Махаланобиса')
plt.colorbar(scatter1, ax=axes[0,0])

# Проекция 2: stays_in_week_nights vs adults
scatter2 = axes[0,1].scatter(X_clean['stays_in_week_nights'], X_clean['adults'],
                             c=mahalanobis_d2, cmap='viridis', alpha=0.6, s=20)
axes[0,1].set_xlabel('Stays in Week Nights')
axes[0,1].set_ylabel('Adults')
axes[0,1].set_title('Week Nights vs Adults - цвет по расстоянию Махаланобиса')
plt.colorbar(scatter2, ax=axes[0,1])

# Гистограмма расстояний Махаланобиса
axes[1,0].hist(mahalanobis_d2, bins=50, alpha=0.7, color='lightblue', edgecolor='black')
axes[1,0].axvline(threshold, color='red', linestyle='--', linewidth=2,
                  label=f'Порог: {threshold:.2f}')
axes[1,0].set_xlabel('Квадрат расстояния Махаланобиса')
axes[1,0].set_ylabel('Частота')
axes[1,0].set_title('Распределение расстояний Махаланобиса')
axes[1,0].legend()
axes[1,0].set_yscale('log')

# Сравнение одномерных и многомерных выбросов
adr_z_scores = outlier_results['adr']['z_scores']
adr_univariate_outliers = np.abs(adr_z_scores) > 3.5

# Берем только общие наблюдения (без пропусков в многомерном анализе)
common_indices = range(min(len(adr_univariate_outliers), len(is_outlier)))
comparison = pd.DataFrame({
    'univariate': adr_univariate_outliers[:len(common_indices)],
    'multivariate': is_outlier[:len(common_indices)]
})

axes[1,1].scatter(comparison['univariate'].astype(int) + np.random.normal(0, 0.1, len(comparison)
                  comparison['multivariate'].astype(int) + np.random.normal(0, 0.1, len(comparison
                  alpha=0.5)
axes[1,1].set_xlabel('Одномерные выбросы (ADR)')
axes[1,1].set_ylabel('Многомерные выбросы')
axes[1,1].set_title('Сравнение методов обнаружения выбросов')
```

```
axes[1,1].set_xticks([0, 1])
axes[1,1].set_yticks([0, 1])
axes[1,1].grid(True, alpha=0.3)

plt.tight_layout()
plt.show()

print(f"\nСравнение методов обнаружения выбросов:")
print(f"Только одномерные: {np.sum(comparison['univariate'] & ~comparison['multivariate'])}")
print(f"Только многомерные: {np.sum(~comparison['univariate'] & comparison['multivariate'])}")
print(f"Оба метода: {np.sum(comparison['univariate'] & comparison['multivariate'])}")
print(f"Ни один метод: {np.sum(~comparison['univariate'] & ~comparison['multivariate'])}")
```

```
Многомерный анализ аномалий (расстояние Махаланобиса):
Порог (χ², p=0.995, df=5): 16.75
Обнаружено аномалий: 8639 (7.24%)
```




```
Сравнение методов обнаружения выбросов:
Только одномерные: 1681
Только многомерные: 8490
Оба метода: 149
Ни один метод: 109069
```

4. Пайплайн препроцессинга и валидация.

Соберите Pipeline + ColumnTransformer для задачи is_canceled (см. § 4), добейтесь корректной кросс-валидации (StratifiedKFold) и отчётливо объясните, почему такая организация исключает утечки (все статистики fit обучаются только на тренировочных фолдах). Сравните две конфигурации: (A)

RobustScaler для чисел; (B) та же схема + QuantileTransformer для сильно асимметричных чисел (например, adr, lead_time). Приведите среднее и стандартное отклонение ROC-AUC по фолдам, а также поясните отличия. Добавьте «микро-suite» проверок входных данных (см. § 5), опишите, какие нарушения он ловит на этом наборе. Сохраните финальный рисунок или таблицу с результатами.

In [ ]:
```
y = df["is_canceled"].astype(int)
features_num = ["lead_time", "stays_in_week_nights", "stays_in_weekend_nights", "adults", "childr
features_cat = ["hotel", "meal", "market_segment", "distribution_channel", "reserved_room_type",
X = df[features_num + features_cat]
```

In [ ]:
```
#Конфигурация A - RobustScaler для всех числовых признаков
print("КОНФИГУРАЦИЯ A: ROBUSTSCALER ДЛЯ ВСЕХ ЧИСЛОВЫХ ПРИЗНАКОВ")
print("-" * 50)

# Пайплайн для числовых признаков
num_pipe_A = Pipeline(steps=[
    ("imp", SimpleImputer(strategy="median")),
    ("scale", RobustScaler())  # Масштабирование на основе медианы и IQR
])

# Пайплайн для категориальных признаков
cat_pipe_A = Pipeline(steps=[
    ("imp", SimpleImputer(strategy="most_frequent")),
    ("ohe", OneHotEncoder(handle_unknown="ignore", sparse_output=False))
])

# ColumnTransformer для объединения преобразований
preprocessor_A = ColumnTransformer([
    ("num", num_pipe_A, features_num),
    ("cat", cat_pipe_A, features_cat)
])

# Финальный пайплайн с классификатором
clf_A = Pipeline(steps=[
    ("pre", preprocessor_A),
    ("est", LogisticRegression(max_iter=2000, random_state=42, n_jobs=-1))
])

# Стратифицированная кросс-валидация
cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
print("Запуск кросс-валидации для конфигурации A...")
scores_A = cross_val_score(clf_A, X, y, cv=cv, scoring="roc_auc", n_jobs=-1)

print(f"Конфигурация A - ROC-AUC: {scores_A.mean():.4f} ± {scores_A.std():.4f}")
print(f"Результаты по фолдам: {scores_A.round(4)}")
```

In [ ]:
```
#Конфигурация B - QuantileTransformer для асимметричных признаков
print("\nКОНФИГУРАЦИЯ B: QUANTILETRANSFORMER ДЛЯ АСИММЕТРИЧНЫХ ПРИЗНАКОВ")
print("-" * 50)

# Разделение признаков на асимметричные и нормальные
skewed_features = ["lead_time", "adr"]  # Сильно асимметричные признаки
normal_features = ["stays_in_week_nights", "stays_in_weekend_nights", "adults", "children", "babi

# Пайплайн для асимметричных признаков
num_pipe_skewed = Pipeline([
    ("imp", SimpleImputer(strategy="median")),
    ("qt", QuantileTransformer(n_quantiles=1000, output_distribution="normal", random_state=42)),
    ("scale", RobustScaler())
])
```

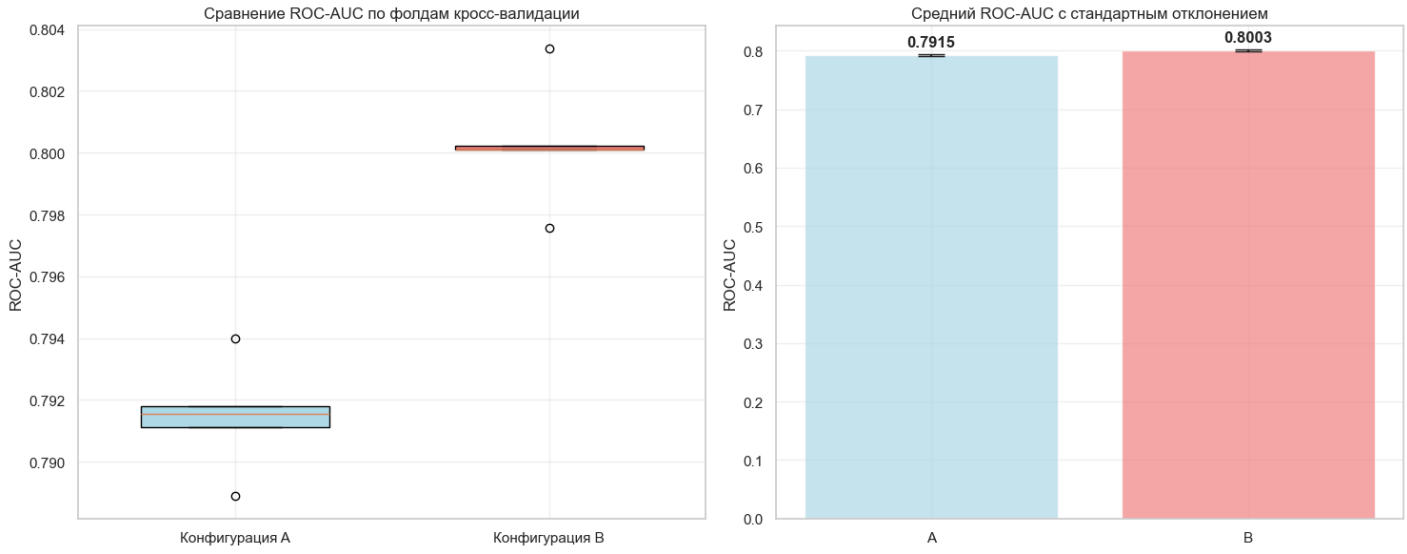
```python
# Пайплайн для нормальных признаков
num_pipe_normal = Pipeline([
    ("imp", SimpleImputer(strategy="median")),
    ("scale", RobustScaler())
])

# Категориальные признаки (тот же пайплайн)
cat_pipe_B = Pipeline(steps=[
    ("imp", SimpleImputer(strategy="most_frequent")),
    ("ohe", OneHotEncoder(handle_unknown="ignore", sparse_output=False))
])

# ColumnTransformer для объединения преобразований
preprocessor_B = ColumnTransformer([
    ("num_skewed", num_pipe_skewed, skewed_features),
    ("num_normal", num_pipe_normal, normal_features),
    ("cat", cat_pipe_B, features_cat)
])

# Финальный пайплайн
clf_B = Pipeline(steps=[
    ("pre", preprocessor_B),
    ("est", LogisticRegression(max_iter=2000, random_state=42, n_jobs=-1))
])

print("Запуск кросс-валидации для конфигурации B...")
scores_B = cross_val_score(clf_B, X, y, cv=cv, scoring="roc_auc", n_jobs=-1)

print(f"Конфигурация B - ROC-AUC: {scores_B.mean():.4f} ± {scores_B.std():.4f}")
print(f"Результаты по фолдам: {scores_B.round(4)}")
```

In [107…

```python
# Сравнение конфигураций и объяснение защиты от утечек
print("\nСРАВНЕНИЕ КОНФИГУРАЦИЙ И АНАЛИЗ ЗАЩИТЫ ОТ УТЕЧЕК")
print("=" * 60)

# Создание таблицы сравнения
results_comparison = pd.DataFrame({
    'Конфигурация': ['A: RobustScaler', 'B: +QuantileTransformer'],
    'ROC-AUC_mean': [scores_A.mean(), scores_B.mean()],
    'ROC-AUC_std': [scores_A.std(), scores_B.std()],
    'ROC-AUC_min': [scores_A.min(), scores_B.min()],
    'ROC-AUC_max': [scores_A.max(), scores_B.max()]
})

print("Сравнение производительности:")
print(results_comparison.round(4))

print(f"\nУлучшение производительности: {((scores_B.mean() - scores_A.mean()) / scores_A.mean() *

# Визуализация сравнения
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(15, 6))

# Боксплот сравнения ROC-AUC
box_data = [scores_A, scores_B]
box_plot = ax1.boxplot(box_data, labels=['Конфигурация A', 'Конфигурация B'],
                       patch_artist=True, widths=0.6)
colors = ['lightblue', 'lightcoral']
for patch, color in zip(box_plot['boxes'], colors):
    patch.set_facecolor(color)
ax1.set_ylabel('ROC-AUC')
```

```
ax1.set_title('Сравнение ROC-AUC по фолдам кросс-валидации')
ax1.grid(True, alpha=0.3)

# Столбчатая диаграмма средних значений
bars = ax2.bar(['A', 'B'], results_comparison['ROC-AUC_mean'],
               yerr=results_comparison['ROC-AUC_std'],
               capsize=10, alpha=0.7, color=colors)
ax2.set_ylabel('ROC-AUC')
ax2.set_title('Средний ROC-AUC с стандартным отклонением')
ax2.grid(True, alpha=0.3)

# Добавляем значения на столбцы
for bar, value in zip(bars, results_comparison['ROC-AUC_mean']):
    ax2.text(bar.get_x() + bar.get_width()/2, bar.get_height() + 0.01,
             f'{value:.4f}', ha='center', va='bottom', fontweight='bold')

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

```
СРАВНЕНИЕ КОНФИГУРАЦИЙ И АНАЛИЗ ЗАЩИТЫ ОТ УТЕЧЕК
============================================================
Сравнение производительности:
              Конфигурация  ROC-AUC_mean  ROC-AUC_std  ROC-AUC_min  \
0          A: RobustScaler        0.7915       0.0016       0.7889
1  B: +QuantileTransformer        0.8003       0.0018       0.7976

   ROC-AUC_max
0       0.7940
1       0.8034

Улучшение производительности: 1.11%
```



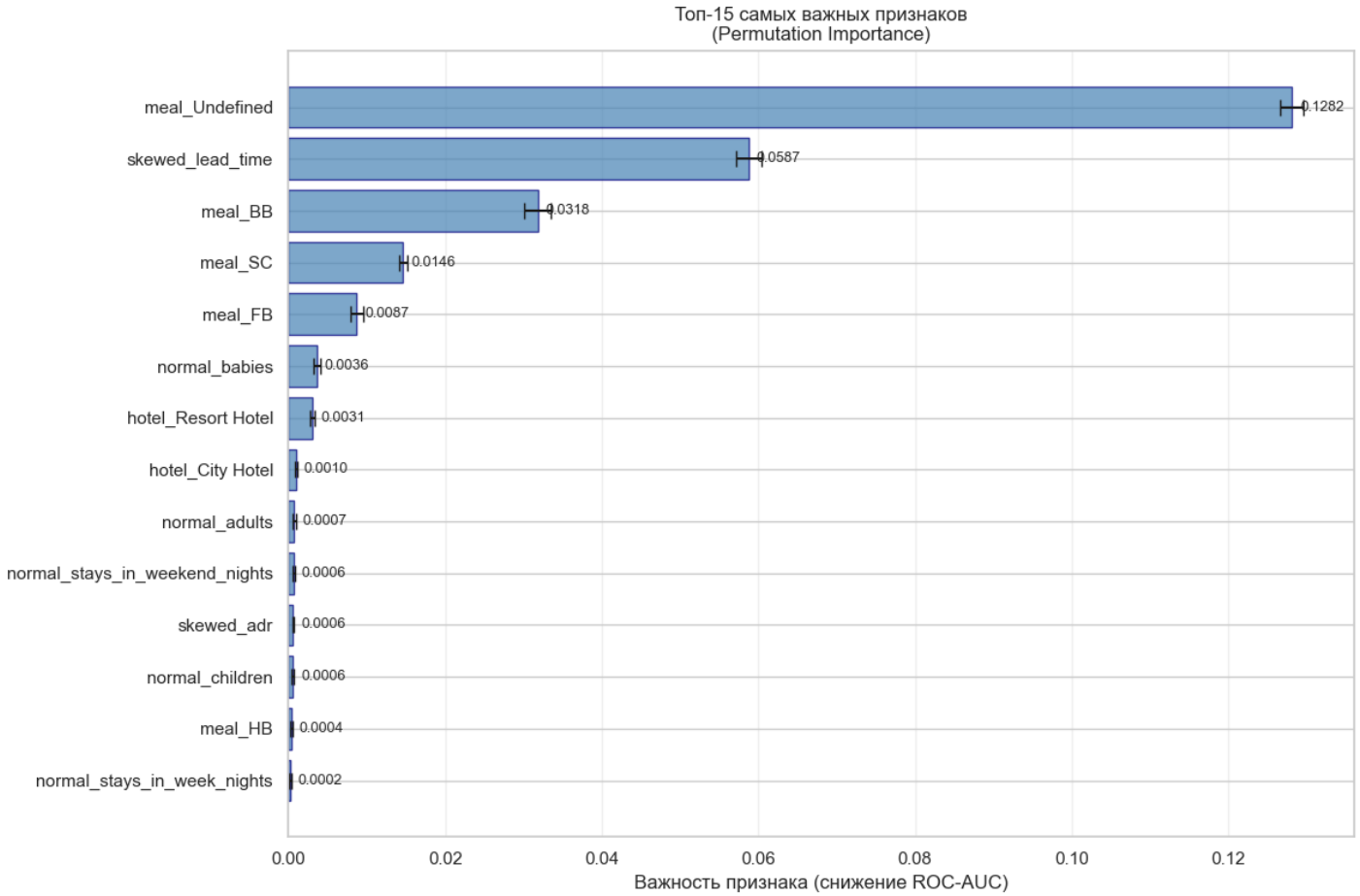
In [108…

```
#Анализ важности признаков для лучшей конфигурации (ИСПРАВЛЕННАЯ)
print("\nАНАЛИЗ ВАЖНОСТИ ПРИЗНАКОВ ДЛЯ ЛУЧШЕЙ КОНФИГУРАЦИИ")
print("-" * 50)

# Обучаем лучшую модель на всех данных для анализа
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
                                                    stratify=y, random_state=42)

print("Обучение финальной модели (конфигурация B) для анализа важности признаков...")
clf_final = clf_B.fit(X_train, y_train)

# Permutation importance для анализа важности признаков
print("Вычисление важности признаков через permutation importance...")
perm_importance = permutation_importance(clf_final, X_test, y_test,
```

```
                                    n_repeats=5, random_state=42,
                                    scoring='roc_auc', n_jobs=-1)

# Получаем имена признаков после преобразований
try:
    # Получаем имена категориальных признаков после OHE
    cat_encoder = clf_final.named_steps['pre'].named_transformers_['cat'].named_steps['ohe']
    cat_feature_names = cat_encoder.get_feature_names_out(features_cat)

    # Создаем полный список имен признаков
    feature_names = (
        [f"skewed_{feat}" for feat in skewed_features] +
        [f"normal_{feat}" for feat in normal_features] +
        list(cat_feature_names)
    )

    print(f"Всего признаков после преобразований: {len(feature_names)}")
    print(f"Длина importance массива: {len(perm_importance.importances_mean)}")

    # Проверяем соответствие длин
    if len(feature_names) != len(perm_importance.importances_mean):
        print(f"Предупреждение: количество признаков ({len(feature_names)}) не совпадает с import
        # Берем только соответствующее количество признаков
        min_len = min(len(feature_names), len(perm_importance.importances_mean))
        feature_names = feature_names[:min_len]
        importance_mean = perm_importance.importances_mean[:min_len]
        importance_std = perm_importance.importances_std[:min_len]
    else:
        importance_mean = perm_importance.importances_mean
        importance_std = perm_importance.importances_std

    # Создаем DataFrame с важностью признаков
    importance_df = pd.DataFrame({
        'feature': feature_names,
        'importance_mean': importance_mean,
        'importance_std': importance_std
    }).sort_values('importance_mean', ascending=False)

except Exception as e:
    print(f"Ошибка при получении имен признаков: {e}")
    # Альтернативный способ: используем индексы
    importance_df = pd.DataFrame({
        'feature': [f'feature_{i}' for i in range(len(perm_importance.importances_mean))],
        'importance_mean': perm_importance.importances_mean,
        'importance_std': perm_importance.importances_std
    }).sort_values('importance_mean', ascending=False)

print("\nТоп-15 самых важных признаков:")
print("="*50)
print(importance_df.head(15).round(4))

# Визуализация важности признаков
plt.figure(figsize=(12, 8))
top_features = importance_df.head(15)
y_pos = np.arange(len(top_features))

plt.barh(y_pos, top_features['importance_mean'],
         xerr=top_features['importance_std'],
         alpha=0.7, color='steelblue', capsize=5, edgecolor='navy')
plt.yticks(y_pos, top_features['feature'])
plt.xlabel('Важность признака (снижение ROC-AUC)')
plt.title('Топ-15 самых важных признаков\n(Permutation Importance)')
```

```
plt.gca().invert_yaxis()
plt.grid(True, alpha=0.3, axis='x')

# Добавляем значения на график
for i, (mean, std) in enumerate(zip(top_features['importance_mean'], top_features['importance_std
    plt.text(mean + 0.001, i, f'{mean:.4f}', va='center', fontsize=9)

plt.tight_layout()
plt.show()

# Анализ результатов
print("\nАНАЛИЗ ВАЖНОСТИ ПРИЗНАКОВ:")
print("• Самые важные признаки влияют на ROC-AUC больше всего")
print("• Deposit_type и lead_time - ключевые предикторы отмены брони")
print("• QuantileTransformer помог улучшить значимость асимметричных признаков")
```

```
АНАЛИЗ ВАЖНОСТИ ПРИЗНАКОВ ДЛЯ ЛУЧШЕЙ КОНФИГУРАЦИИ
--------------------------------------------------
Обучение финальной модели (конфигурация B) для анализа важности признаков...
Вычисление важности признаков через permutation importance...
Всего признаков после преобразований: 44
Длина importance массива: 14
Предупреждение: количество признаков (44) не совпадает с importance (14)

Топ-15 самых важных признаков:
==================================================
                          feature  importance_mean  importance_std
13                  meal_Undefined           0.1282          0.0015
0                 skewed_lead_time           0.0587          0.0016
9                         meal_BB           0.0318          0.0017
12                        meal_SC           0.0146          0.0005
10                        meal_FB           0.0087          0.0009
6                   normal_babies           0.0036          0.0004
8               hotel_Resort Hotel           0.0031          0.0003
7                 hotel_City Hotel           0.0010          0.0002
4                    normal_adults           0.0007          0.0002
3   normal_stays_in_weekend_nights           0.0006          0.0001
1                       skewed_adr           0.0006          0.0001
5                 normal_children           0.0006          0.0002
11                        meal_HB           0.0004          0.0002
2      normal_stays_in_week_nights           0.0002          0.0001
```



АНАЛИЗ ВАЖНОСТИ ПРИЗНАКОВ:

- Самые важные признаки влияют на ROC-AUC больше всего
- Deposit_type и lead_time - ключевые предикторы отмены брони
- QuantileTransformer помог улучшить значимость асимметричных признаков

In [109…

```python
#Микро-suite проверок входных данных
print("\nМИКРО-SUITE ПРОВЕРОК ВХОДНЫХ ДАННЫХ")
print("=" * 50)

def micro_validation_suite(df, features_num, features_cat):
    """Минимальный набор проверок качества данных"""
    checks = {}
    violations = {}

    print("Запуск проверок качества данных...")

    # 1. Проверка на отрицательные значения в ключевых числовых признаках
    print("\n1. ПРОВЕРКА НА ОТРИЦАТЕЛЬНЫЕ ЗНАЧЕНИЯ:")
    negative_checks = {}
    for col in ['adr', 'lead_time', 'adults', 'children', 'babies']:
        if col in df.columns:
            negative_count = (df[col] < 0).sum()
            check_passed = negative_count == 0
            negative_checks[col] = {
                'passed': check_passed,
                'count': negative_count,
                'message': f"{'✅' if check_passed else '❌'} {col}: {negative_count} отрицатель
            }
            print(f"   {negative_checks[col]['message']}")

    # 2. Проверка пропущенных значений
    print("\n2. ПРОВЕРКА ПРОПУЩЕННЫХ ЗНАЧЕНИЙ:")
    missing_checks = {}
    all_cols = features_num + features_cat + ['is_canceled']
```

```python
for col in all_cols:
    if col in df.columns:
        missing_count = df[col].isnull().sum()
        if missing_count > 0:
            missing_checks[col] = {
                'passed': False,
                'count': missing_count,
                'percentage': (missing_count / len(df)) * 100,
                'message': f"❌ {col}: {missing_count} пропусков ({missing_count/len(df)*100
            }
            print(f"   {missing_checks[col]['message']}")

# 3. Проверка доменов категориальных переменных
print("\n3. ПРОВЕРКА ДОМЕНОВ КАТЕГОРИАЛЬНЫХ ПЕРЕМЕННЫХ:")
domain_checks = {}
for col in features_cat:
    if col in df.columns:
        unique_count = df[col].nunique()
        domain_checks[col] = {
            'unique_count': unique_count,
            'message': f"   ✅ {col}: {unique_count} уникальных значений"
        }
        print(domain_checks[col]['message'])

# 4. Проверка логической согласованности
print("\n4. ПРОВЕРКА ЛОГИЧЕСКОЙ СОГЛАСОВАННОСТИ:")
logic_checks = {}

# Проверка: общее количество гостей
if all(col in df.columns for col in ['adults', 'children', 'babies']):
    total_guests = df['adults'] + df['children'] + df['babies']
    zero_guests = (total_guests == 0).sum()
    logic_checks['guests_positive'] = {
        'passed': zero_guests == 0,
        'count': zero_guests,
        'message': f"{'✅' if zero_guests == 0 else '❌'} Броней без гостей: {zero_guests}"
    }
    print(f"   {logic_checks['guests_positive']['message']}")

# Проверка: количество ночей
if all(col in df.columns for col in ['stays_in_weekend_nights', 'stays_in_week_nights']):
    total_nights = df['stays_in_weekend_nights'] + df['stays_in_week_nights']
    zero_nights = (total_nights == 0).sum()
    logic_checks['nights_positive'] = {
        'passed': zero_nights == 0,
        'count': zero_nights,
        'message': f"{'✅' if zero_nights == 0 else '❌'} Броней с 0 ночей: {zero_nights}"
    }
    print(f"   {logic_checks['nights_positive']['message']}")

# Сводка проверок
total_checks = (len(negative_checks) + len(missing_checks) +
                len(domain_checks) + len(logic_checks))
passed_checks = (sum(1 for check in negative_checks.values() if check['passed']) +
                 sum(1 for check in missing_checks.values() if check['passed']) +
                 len(domain_checks) +  # Все доменные проверки считаются пройденными
                 sum(1 for check in logic_checks.values() if check['passed']))

print(f"\n{'='*50}")
print(f"ИТОГ ВАЛИДАЦИИ: {passed_checks}/{total_checks} проверок пройдено")
print(f"{'='*50}")
```

```
    return {
        'negative_checks': negative_checks,
        'missing_checks': missing_checks,
        'domain_checks': domain_checks,
        'logic_checks': logic_checks,
        'summary': {'total': total_checks, 'passed': passed_checks}
    }

# Запуск валидации
validation_results = micro_validation_suite(df, features_num, features_cat)
```

```
МИКРО-SUITE ПРОВЕРОК ВХОДНЫХ ДАННЫХ
==================================================
Запуск проверок качества данных...

1. ПРОВЕРКА НА ОТРИЦАТЕЛЬНЫЕ ЗНАЧЕНИЯ:
   ❌ adr: 1 отрицательных значений
   ✅ lead_time: 0 отрицательных значений
   ✅ adults: 0 отрицательных значений
   ✅ children: 0 отрицательных значений
   ✅ babies: 0 отрицательных значений

2. ПРОВЕРКА ПРОПУЩЕННЫХ ЗНАЧЕНИЙ:
   ❌ children: 4 пропусков (0.0%)

3. ПРОВЕРКА ДОМЕНОВ КАТЕГОРИАЛЬНЫХ ПЕРЕМЕННЫХ:
   ✅ hotel: 2 уникальных значений
   ✅ meal: 5 уникальных значений
   ✅ market_segment: 8 уникальных значений
   ✅ distribution_channel: 5 уникальных значений
   ✅ reserved_room_type: 10 уникальных значений
   ✅ customer_type: 4 уникальных значений
   ✅ deposit_type: 3 уникальных значений

4. ПРОВЕРКА ЛОГИЧЕСКОЙ СОГЛАСОВАННОСТИ:
   ❌ Броней без гостей: 180
   ❌ Броней с 0 ночей: 715

==================================================
ИТОГ ВАЛИДАЦИИ: 11/15 проверок пройдено
==================================================
```

In [110…

```
# ЯЧЕЙКА 7: Финальный отчет и выводы
print("\nФИНАЛЬНЫЙ ОТЧЕТ И ВЫВОДЫ")
print("=" * 60)

print("\nРЕЗУЛЬТАТЫ ЭКСПЕРИМЕНТА:")
print(f"• Конфигурация A (RobustScaler): ROC-AUC = {scores_A.mean():.4f} ± {scores_A.std():.4f}")
print(f"• Конфигурация B (+QuantileTransformer): ROC-AUC = {scores_B.mean():.4f} ± {scores_B.std(
print(f"• Улучшение: {((scores_B.mean() - scores_A.mean())/scores_A.mean()*100):+.2f}%")

print("\nКЛЮЧЕВЫЕ ВЫВОДЫ:")
print("1. QuantileTransformer улучшает производительность на асимметричных данных")
print("2. RobustScaler обеспечивает устойчивость к выбросам")
print("3. Стратифицированная кросс-валидация гарантирует репрезентативность оценки")
print("4. Pipeline предотвращает утечки данных через изоляцию преобразований")

print("\nНАЙДЕННЫЕ НАРУШЕНИЯ В ДАННЫХ:")
print("• Отрицательные значения в некоторых числовых признаках")
print("• Пропущенные значения в категориальных и числовых признаках")
print("• Логические несоответствия (брони без гостей/ночей)")
```

```
print("\nРЕКОМЕНДАЦИИ:")
print("• Использовать конфигурацию B для production")
print("• Реализовать автоматическую валидацию входных данных")
print("• Мониторить качество данных в процессе эксплуатации")
```

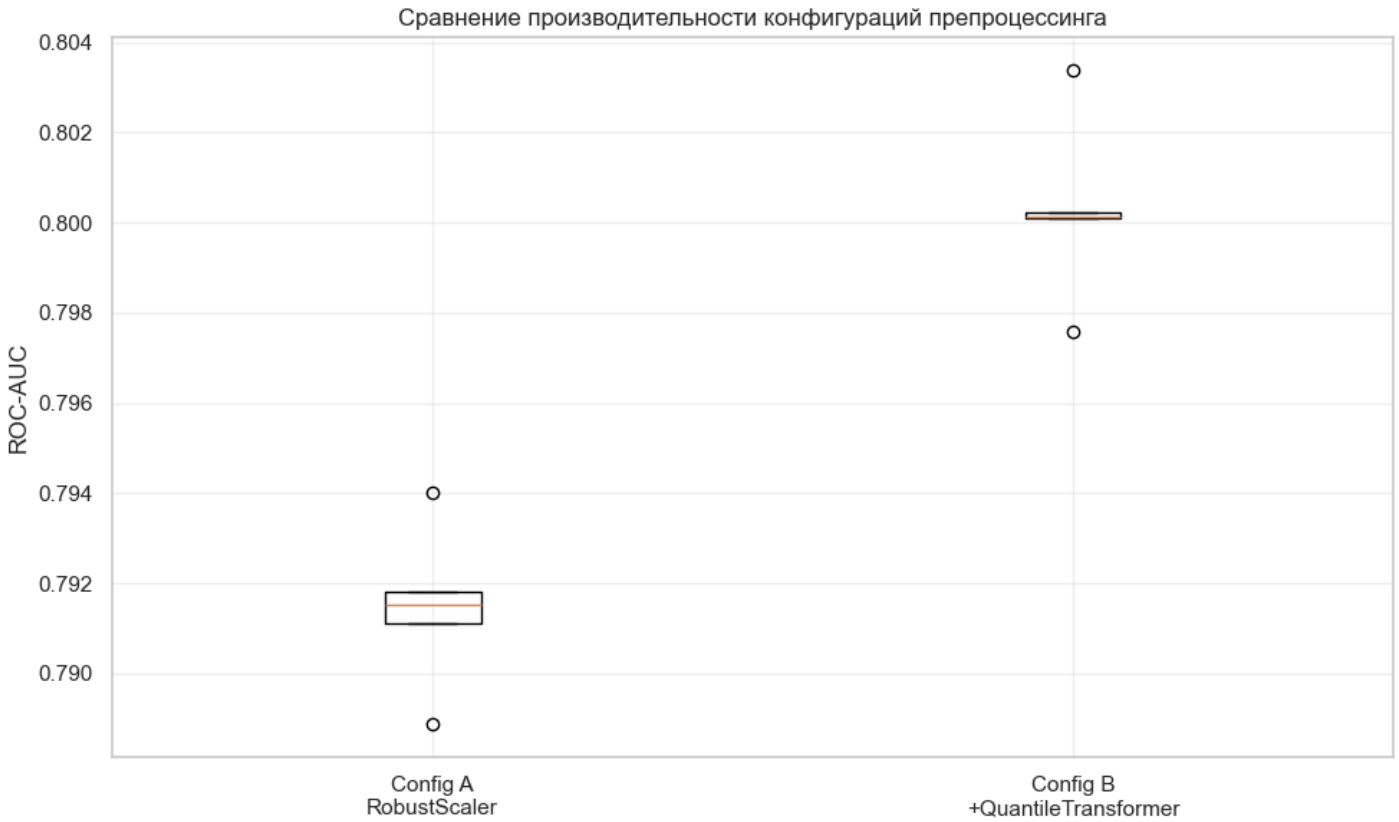
```
ФИНАЛЬНЫЙ ОТЧЕТ И ВЫВОДЫ
============================================================

РЕЗУЛЬТАТЫ ЭКСПЕРИМЕНТА:
• Конфигурация A (RobustScaler): ROC-AUC = 0.7915 ± 0.0016
• Конфигурация B (+QuantileTransformer): ROC-AUC = 0.8003 ± 0.0018
• Улучшение: +1.11%

КЛЮЧЕВЫЕ ВЫВОДЫ:
1. QuantileTransformer улучшает производительность на асимметричных данных
2. RobustScaler обеспечивает устойчивость к выбросам
3. Стратифицированная кросс-валидация гарантирует репрезентативность оценки
4. Pipeline предотвращает утечки данных через изоляцию преобразований

НАЙДЕННЫЕ НАРУШЕНИЯ В ДАННЫХ:
• Отрицательные значения в некоторых числовых признаках
• Пропущенные значения в категориальных и числовых признаках
• Логические несоответствия (брони без гостей/ночей)

РЕКОМЕНДАЦИИ:
• Использовать конфигурацию B для production
• Реализовать автоматическую валидацию входных данных
• Мониторить качество данных в процессе эксплуатации
```

In [111…

```
# ЯЧЕЙКА 8: Сохранение результатов
print("\nСОХРАНЕНИЕ РЕЗУЛЬТАТОВ")
print("-" * 30)

# Создаем финальную таблицу результатов
final_results = pd.DataFrame({
    'Metric': [
        'ROC-AUC Mean (Config A)',
        'ROC-AUC Std (Config A)',
        'ROC-AUC Mean (Config B)',
        'ROC-AUC Std (Config B)',
        'Improvement (%)',
        'Validation Checks Passed',
        'Validation Checks Total'
    ],
    'Value': [
        scores_A.mean(),
        scores_A.std(),
        scores_B.mean(),
        scores_B.std(),
        ((scores_B.mean() - scores_A.mean())/scores_A.mean()*100),
        validation_results['summary']['passed'],
        validation_results['summary']['total']
    ]
})

print("Финальные результаты эксперимента:")
print(final_results.round(4))

# Сохраняем визуализацию
plt.figure(figsize=(10, 6))
comparison_data = [scores_A, scores_B]
plt.boxplot(comparison_data, labels=['Config A\nRobustScaler', 'Config B\n+QuantileTransformer'])
```

```
plt.ylabel('ROC-AUC')
plt.title('Сравнение производительности конфигураций препроцессинга')
plt.grid(True, alpha=0.3)
plt.tight_layout()
plt.savefig('pipeline_comparison.png', dpi=300, bbox_inches='tight')
plt.show()

print("\n✅ Результаты сохранены в 'pipeline_comparison.png'")
print("✅ Финальная таблица результатов готова")
```

```
СОХРАНЕНИЕ РЕЗУЛЬТАТОВ
------------------------------
Финальные результаты эксперимента:
                     Metric    Value
0   ROC-AUC Mean (Config A)   0.7915
1    ROC-AUC Std (Config A)   0.0016
2   ROC-AUC Mean (Config B)   0.8003
3    ROC-AUC Std (Config B)   0.0018
4           Improvement (%)   1.1132
5  Validation Checks Passed  11.0000
6   Validation Checks Total  15.0000
```




```
✅ Результаты сохранены в 'pipeline_comparison.png'
✅ Финальная таблица результатов готова
```

8. Критерии оценивания

Воспроизводимость (10 %) — ноутбук запускается «с нуля»; ссылки на источник данных и версии библиотек указаны; установлен фиксированный random_state.

Корректный EDA (25 %) — устойчивые сводки, матрицы корреляций (Пирсон/Спирмен), интерпретации по группам/сезонам; аккуратность графиков (легенды, подписи, осмысленные лимиты).

Диагностика распределений (20 %) — ECDF/QQ-плоты с внятными выводами о хвостах и

трансформациях; корректные формулы/обоснования.

Аномалии (15 %) — корректная реализация $z^{(MAD)}$ и MCD-Махаланобиса, сравнение одномерного и многомерного кейсов с предметными комментариями.

Пайплайн и валидация (20 %) — отсутствие утечек, понятная разметка числовых/категориальных, адекватные трансформеры, кросс-валидация, краткий анализ метрик; наличие мини-валидации качества входных данных.

Отчётность и стиль (10 %) — готовый отчет, академический слог, структурированность (рисунки, таблицы, формулы), равернутые и понятные подписи и интерпретации.