Министерство цифрового развития

Федеральное государственное бюджетное образовательное учреждение высшего образования
«Сибирский государственный университет телекоммуникаций и информатики»
(СибГУТИ)

Кафедра прикладной математики и кибернетики

Отчёт

по лабораторной работе № 1 «Первичный анализ и предобработка данных»

Выполнил:

студент группы ИП-213

Дмитриев Антон Александрович

Работу проверил: Преподаватель

Сороковых Дарья Анатольевна

Новосибирск 2025 г.

Введение (задание)
Выбранный набор данных
Этот набор данных содержит записи о транзакциях в кофейне, включая подробную информацию о продажах, типе оплаты, времени покупки и предпочтениях клиентов.

Он специально разработан для визуализации данных, создания информационных панелей и проектов бизнес-аналитики с использованием таких инструментов, как Power BI, Tableau и библиотеки визуализации Python (Matplotlib, Seaborn, Plotly).

Благодаря атрибутам, охватывающим время суток, дни недели, месяцы, виды кофе и выручку, этот набор данных обеспечивает надежную основу для анализа поведения клиентов, моделей продаж и тенденций эффективности бизнеса.

Структура набора данных-

Формат файла: CSV

Столбцы (характеристики):

hour_of_day → Время совершения покупки (0-23)

cash_type → Способ оплаты (наличные / карта)

money → Сумма транзакции (в местной валюте)

coffee_mane → Тип приобретаемого кофе (например, Латте, американо, горячий шоколад)

Time_of_Day → Время покупки по категориям (Утро, день, ночь)

Weekday → День недели (например, Пн, Вт, ...)

Month_name → Месяц покупки (например, январь, февраль, март)

Weekdaysort → Числовое представление для заказа по дням недели (1 = Пн, 7 = Вс)

Monthsort → Числовое представление для заказа по месяцам (1 = январь, 12 = декабрь)

Date → Дата транзакции (ГГГГ-ММ-ДД)

Tine → Точное время совершения транзакции (ЧЧ:ММ:СС)

Задание

1. Выбрать набор данных на платформе Kaggle.
2. Загрузить данные в среду разработки (Jupyter Notebook / Google Colab).
3. Провести все этапы разведочного анализа (EDA), описанные ниже.

Подробная информация о задании:

1. Загрузка и первичный осмотр:
   - Загрузите данные в DataFrame.
   - Выведите первые 5–10 строк.
   - Используйте методы .info(), .describe(), .shape для получения общей информации.
   - Вывод: Опишите общий размер набора данных, типы признаков и наличие пропущенных значений.
2. Анализ пропусков:
   - Посчитайте количество и долю пропусков в каждом столбце.
   - Визуализируйте матрицу пропусков с помощью sns.heatmap().
   - Вывод: определите столбцы с наибольшим процентом пропусков. Предложите стратегию их обработки (удаление, заполнение медианой/модой).
3. Анализ числовых признаков:
   - Для всех числовых столбцов постройте гистограммы и boxplot'ы.
   - Рассчитайте стандартные статистики (среднее, медиана, стандартное отклонение, асимметрия) для ключевых числовых признаков.
   - Вывод: Охарактеризуйте распределения, наличие выбросов.
4. Анализ категориальных признаков:
   - Для категориальных столбцов постройте столбчатые диаграммы (sns.countplot()).
   - Посчитайте количество уникальных категорий в каждом признаке.
   - Вывод: определите категориальные признаки с большим количеством уникальных
   - значений (высокая кардинальность).
5. Анализ взаимосвязей:
   - Постройте матрицу корреляций для числовых признаков и визуализируйте ее тепловой картой (sns.heatmap()).

- Для пар ключевых признаков постройте диаграммы рассеяния (sns.scatterplot()).
- Исследуйте взаимосвязь категориальных и числовых признаков с помощью boxplot'ов (например, sns.boxplot(x='категория', y='число')).
- Вывод: Назовите наиболее коррелирующие пары признаков. Есть ли видимая зависимость между целевой переменной и другими признаками?

6. Базовая предобработка:
   - Приведите названия столбцов к удобному формату (например, нижний регистр).
   - Обработайте пропуски в соответствии с выводами из п.2.
   - Преобразуйте категориальные признаки в числовой формат выбранным методом.
7. Обработка выбросов (*):
   - Выберите один числовой признак с сильными выбросами.
   - Примените к нему один из методов обработки выбросов (например, логарифмирование или "обрезку" на основе IQR).
   - Постройте boxplot до и после обработки и прокомментируйте результат.

Основная часть

1. Загрузка и первичный осмотр.

Загрузка данных и вывод первых 10 строк

```
import pandas as pd

df = pd.read_csv('/content/drive/MyDrive/Coffe_sales.csv')
print("Первые 10 строк:")
df.head(10)
```

```
Первые 10 строк:
   hour_of_day cash_type  money          coffee_name Time_of_Day Weekday
\
0           10      card   38.7                Latte     Morning     Fri
1           12      card   38.7        Hot Chocolate   Afternoon     Fri
2           12      card   38.7        Hot Chocolate   Afternoon     Fri
3           13      card   28.9            Americano   Afternoon     Fri
4           13      card   38.7                Latte   Afternoon     Fri
5           15      card   33.8  Americano with Milk   Afternoon     Fri
6           16      card   38.7        Hot Chocolate   Afternoon     Fri
7           18      card   33.8  Americano with Milk       Night     Fri
8           19      card   38.7                Cocoa       Night     Fri
9           19      card   33.8  Americano with Milk       Night     Fri

  Month_name  Weekdaysort  Monthsort        Date             Time
0        Mar            5          3  2024-03-01  10:15:50.520000
1        Mar            5          3  2024-03-01  12:19:22.539000
2        Mar            5          3  2024-03-01  12:20:18.089000
3        Mar            5          3  2024-03-01  13:46:33.006000
4        Mar            5          3  2024-03-01  13:48:14.626000
5        Mar            5          3  2024-03-01  15:39:47.726000
6        Mar            5          3  2024-03-01  16:19:02.756000
7        Mar            5          3  2024-03-01  18:39:03.580000
8        Mar            5          3  2024-03-01  19:22:01.762000
9        Mar            5          3  2024-03-01  19:23:15.887000
```

Использование методов для получения общей информации.

```
print("Информация о данных:")
df.info()
```

```
Информация о данных:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3547 entries, 0 to 3546
Data columns (total 11 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   hour_of_day  3547 non-null   int64
 1   cash_type    3547 non-null   object
 2   money        3547 non-null   float64
 3   coffee_name  3547 non-null   object
 4   Time_of_Day  3547 non-null   object
 5   Weekday      3547 non-null   object
 6   Month_name   3547 non-null   object
```

```
 7   Weekdaysort  3547 non-null   int64
 8   Monthsort    3547 non-null   int64
 9   Date         3547 non-null   object
 10  Time         3547 non-null   object
dtypes: float64(1), int64(3), object(7)
memory usage: 304.9+ KB
```

```
print("Описательная статистика:")
df.describe()
```

```
Описательная статистика:
       hour_of_day        money  Weekdaysort    Monthsort
count  3547.000000  3547.000000  3547.000000  3547.000000
mean     14.185791    31.645216     3.845785     6.453905
std       4.234010     4.877754     1.971501     3.500754
min       6.000000    18.120000     1.000000     1.000000
25%      10.000000    27.920000     2.000000     3.000000
50%      14.000000    32.820000     4.000000     7.000000
75%      18.000000    35.760000     6.000000    10.000000
max      22.000000    38.700000     7.000000    12.000000
```

```
print("Размерность данных:")
df.shape
```

```
Размерность данных:
(3547, 11)
```

Вывод:

Общий размер набора данных составляет 3547 строк и 11 признаков.

Числовые признаки: Время транзакции, Сумма транзакции, День недели, Месяц покупки.

Категорийные признаки: Способ оплаты, Название кофе, Время дня, День недели, Название месяца, Дата, Время.

Во всех признаках отсутствуют пропуски.

2. Анализ пропусков

Подсчет количества и доли пропусков

```
missing_count = df.isnull().sum()
missing_percent = (df.isnull().sum() / df.shape[0]) * 100

missing_df_count = pd.DataFrame({
    'Количество пропусков' : missing_count
```

```
})

missing_df_percent = pd.DataFrame({
    'Процент пропусков' : missing_percent
})

missing_df = pd.DataFrame({
    'Количество пропусков' : missing_count,
    'Процент пропусков' : round(missing_percent, 2)
})

print(missing_df)
```

	Количество пропусков	Процент пропусков
hour_of_day	0	0.0
cash_type	0	0.0
money	0	0.0
coffee_name	0	0.0
Time_of_Day	0	0.0
Weekday	0	0.0
Month_name	0	0.0
Weekdaysort	0	0.0
Monthsort	0	0.0
Date	0	0.0
Time	0	0.0

Вывод тепловой карты

```
import matplotlib.pyplot as plt

import seaborn as sns


plt.figure(figsize=(10, 6))

sns.heatmap(missing_df_count, annot = True, cmap="Blues") # Создание тепловой карты

plt.title('Тепловая карта пропущенных значений')

plt.show()
```

Вывод:

Пропуски отсутствуют.

3. Анализ числовых признаков

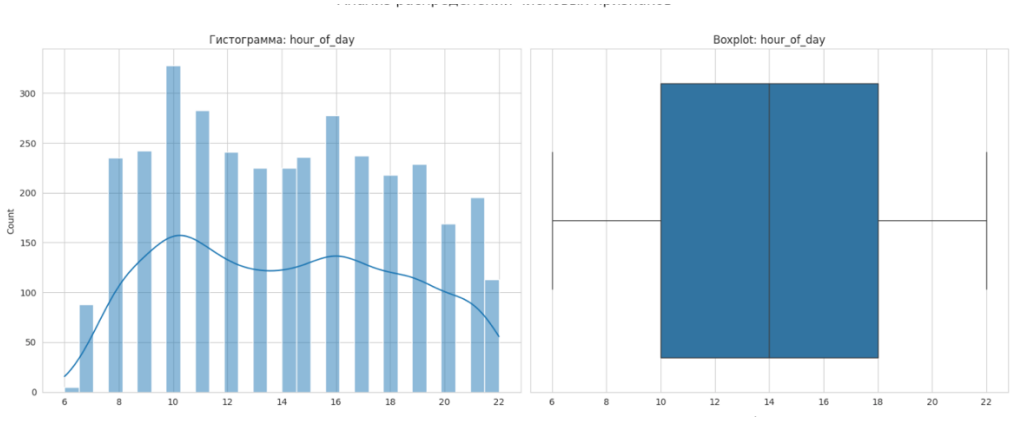
Расчет стандартных признаков

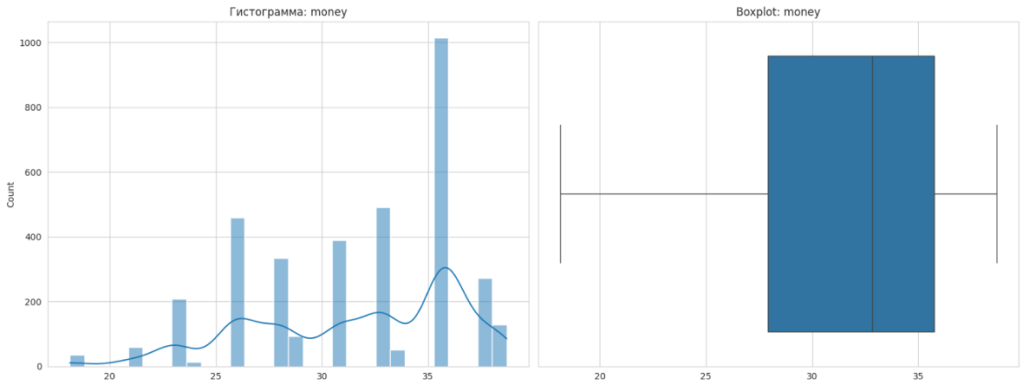
```
from scipy import stats

import numpy as np


# Настройка стиля графиков

sns.set_style("whitegrid")

plt.rcParams['figure.figsize'] = (12, 8)


# Выделяем числовые признаки

numeric_columns =
df.select_dtypes(include=[np.number]).columns.tolist()


# 1. Рассчитаем статистики для всех числовых признаков

numeric_stats = df[numeric_columns].agg(['count','mean', 'median',
'std', 'skew']).round(2)

print("Статистики числовых признаков:")

print(numeric_stats.T)
```

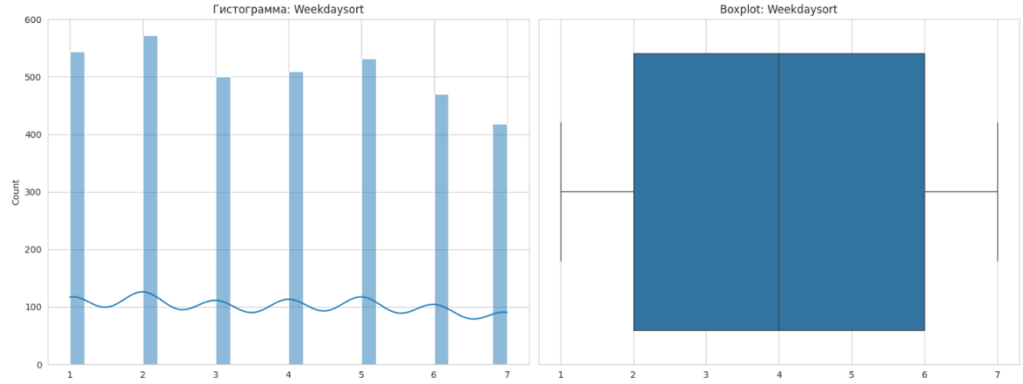
```
Статистики числовых признаков:

              count   mean  median   std  skew
hour_of_day  3547.0  14.19   14.00  4.23  0.12
money        3547.0  31.65   32.82  4.88 -0.54
Weekdaysort  3547.0   3.85    4.00  1.97  0.08
Monthsort    3547.0   6.45    7.00  3.50  0.00
```

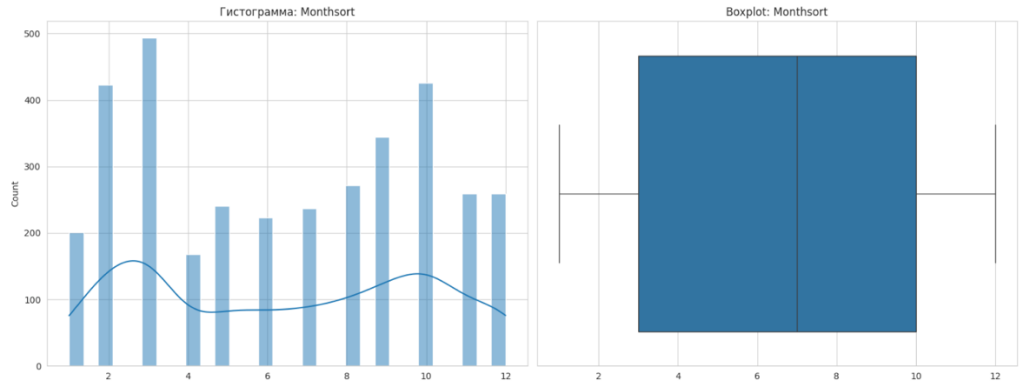
Построение гистограмм и boxplot`ов

```
fig, axes = plt.subplots(len(numeric_columns), 2, figsize=(16,
6*len(numeric_columns)))


for i, column in enumerate(numeric_columns):
    # Гистограмма
    sns.histplot(df[column].dropna(), kde=True, ax=axes[i, 0], bins=30)
    axes[i, 0].set_title(f'Гистограмма: {column}')
    axes[i, 0].set_xlabel('')

    # Boxplot
    sns.boxplot(x=df[column].dropna(), ax=axes[i, 1])
    axes[i, 1].set_title(f'Boxplot: {column}')
    axes[i, 1].set_xlabel('')


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









Вывод:

hour_of_day (час дня)

Распределение: Практически симметричное (скошенность = 0.12)

Центральная тенденция: Среднее (14.19) и медиана (14.00) почти совпадают

Разброс: Умеренный (стандартное отклонение = 4.23 часа)

Диапазон: От 6:00 до 22:00 (16-часовой интервал)

Отсутствие выбросов: Распределение равномерное, без аномальных значений

money (деньги)

Распределение: Умеренная отрицательная асимметрия (скошенность = -0.54)

Центральная тенденция: Медиана (32.82) выше среднего (31.65)

Разброс: Умеренный (стандартное отклонение = 4.88)

Диапазон: От 18.12 до 38.70

Отсутствие выбросов: Все значения находятся в ожидаемом диапазоне

Weekdaysort (дни недели)

Распределение: Практически симметричное (скошенность = 0.08)

Центральная тенденция: Среднее (3.85) и медиана (4.00) почти совпадают

Разброс: Умеренный (стандартное отклонение = 1.97)

Диапазон: От 1 (понедельник) до 7 (воскресенье)

Отсутствие выбросов: Равномерное распределение по дням недели

Monthsort (месяцы)

Распределение: Идеально симметричное (скошенность = 0.00)

Центральная тенденция: Среднее (6.45) близко к медиане (7.00)

Разброс: Значительный (стандартное отклонение = 3.50)

Диапазон: От 1 (январь) до 12 (декабрь)

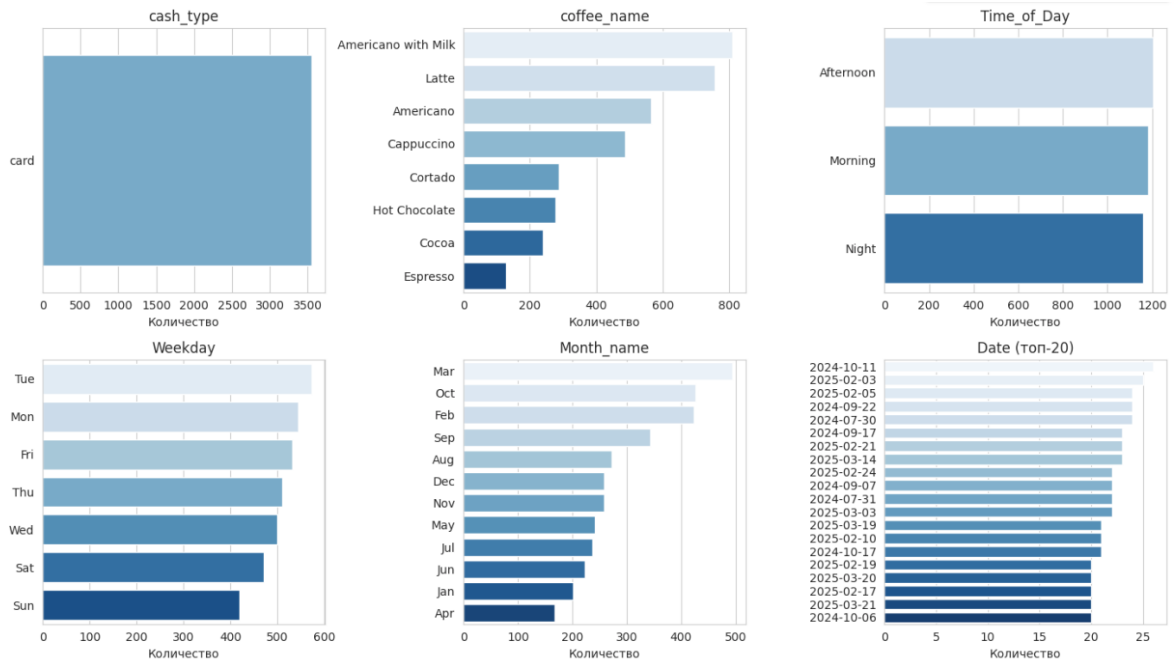
Отсутствие выбросов: Данные равномерно распределены по месяцам

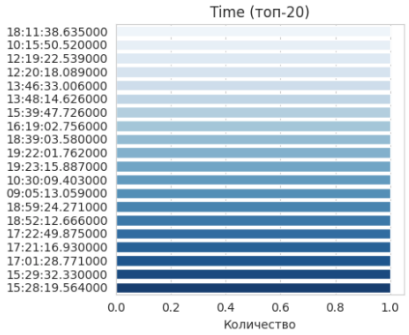
4. Анализ категорийных признаков

Столбчатые диаграммы для категорийных признаков

```
import pandas as pd
```

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

# Настройка стиля графиков
sns.set_style("whitegrid")
plt.figure(figsize=(15, 12))

# Категориальные столбцы
categorical_columns = ['cash_type', 'coffee_name', 'Time_of_Day',
'Weekday', 'Month_name', 'Date', 'Time']

# Построение гистограмм для каждого категориального столбца
for i, column in enumerate(categorical_columns, 1):
    plt.subplot(3, 3, i)

    # Для столбцов с большим количеством уникальных значений покажем топ-
20
    if df[column].nunique() > 20:
        top_20 = df[column].value_counts().head(20)
        sns.barplot(x=top_20.values, y=top_20.index,
                    hue=top_20.index, palette='Blues', legend=False)
        plt.title(f'{column} (топ-20)')
    else:
        # Для столбцов с малым количеством категорий покажем все
        value_counts = df[column].value_counts()
        sns.barplot(x=value_counts.values, y=value_counts.index,
                    hue=value_counts.index, palette='Blues', legend=False)
        plt.title(f'{column}')

    plt.xlabel('Количество')
    plt.ylabel('')

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





Количество уникальных категорий в каждом признаке

```
unique_counts = df.nunique()
print("Количество уникальных значений в каждом столбце:")
print(unique_counts)
```

```
Количество уникальных значений в каждом столбце:
hour_of_day      17
cash_type         1
money            13
coffee_name       8
Time_of_Day       3
Weekday           7
Month_name       12
Weekdaysort       7
Monthsort        12
Date            381
Time           3547
dtype: int64
```

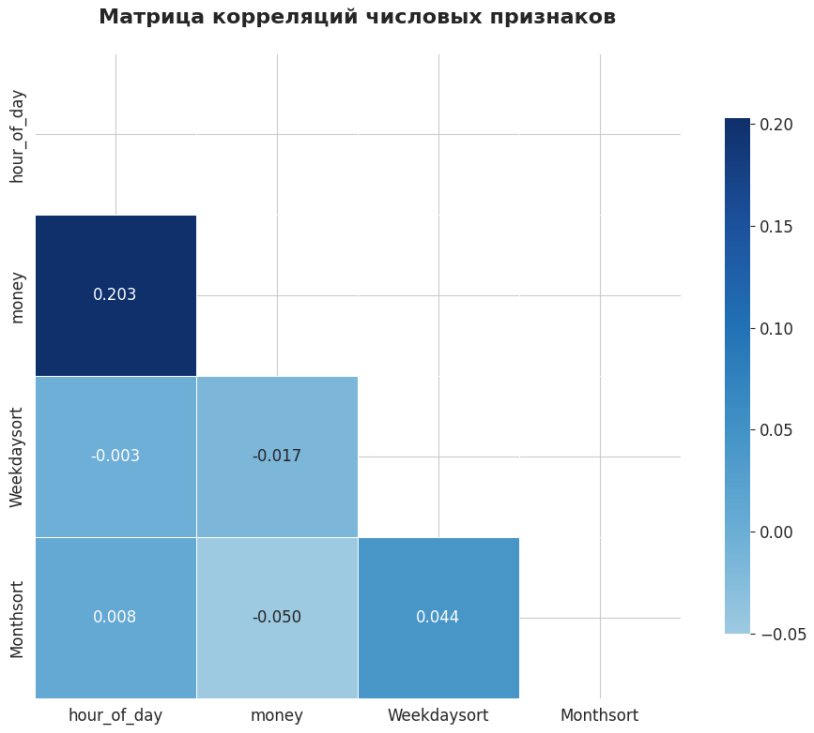
Вывод:
Столбцами с высокой кардинальностью являются Time и Date.

5. Анализ взаимосвязей
Матрица корреляции

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

# Настройка стиля графиков
sns.set_style("whitegrid")
plt.rcParams['figure.figsize'] = (12, 10)
plt.rcParams['font.size'] = 12

# Выделяем числовые признаки
numeric_columns = ['hour_of_day', 'money', 'Weekdaysort', 'Monthsort']
numeric_df = df[numeric_columns]

# Создаем матрицу корреляций
correlation_matrix = numeric_df.corr()
```

```
# Визуализация тепловой карты
plt.figure(figsize=(10, 8))
mask = np.triu(np.ones_like(correlation_matrix, dtype=bool))  # Маска
для верхнего треугольника

sns.heatmap(correlation_matrix,
            mask=mask,
            annot=True,
            fmt='.3f',
            cmap='Blues',
            center=0,
            square=True,
            cbar_kws={'shrink': 0.8},
            linewidths=0.5,
            linecolor='white')

plt.title('Матрица корреляций числовых признаков\n', fontsize=16,
fontweight='bold')
plt.tight_layout()
plt.show()

# Вывод матрицы корреляций в числовом виде
print("Матрица корреляций:")
print(correlation_matrix.round(3))
```

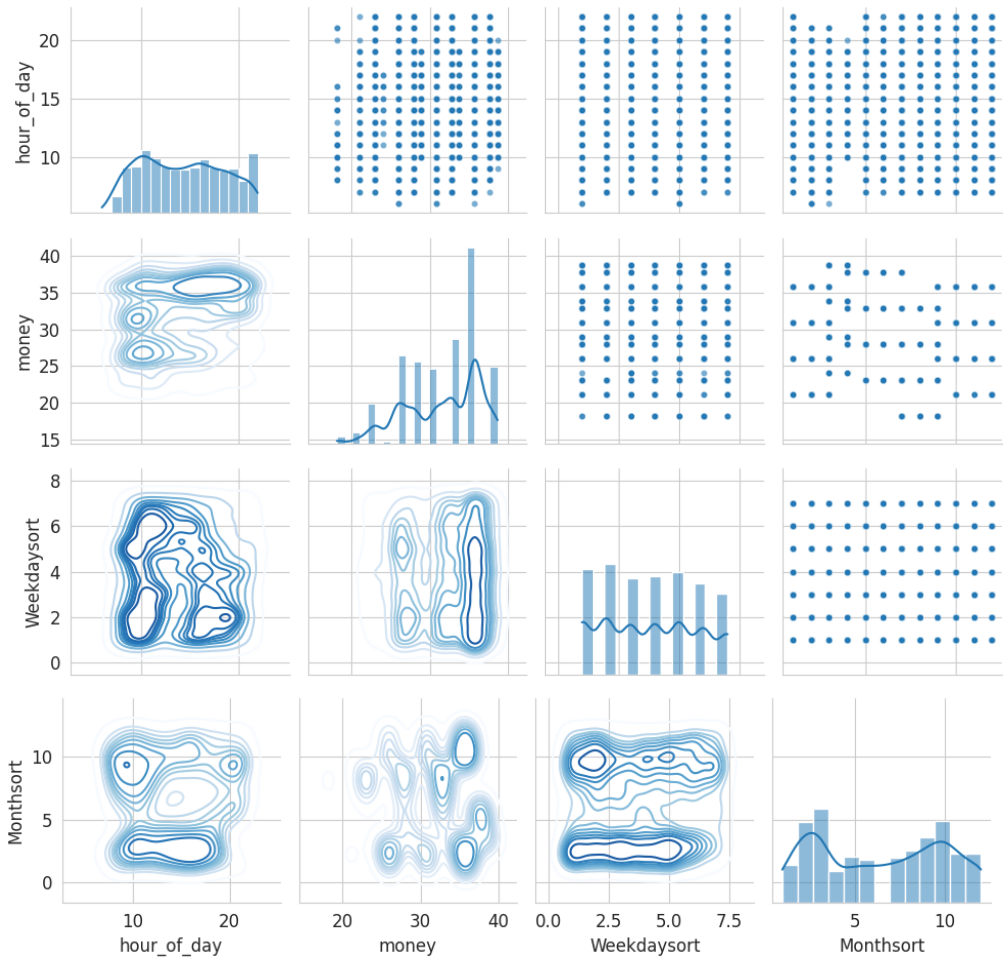
```
Матрица корреляций:
             hour_of_day  money  Weekdaysort  Monthsort
hour_of_day        1.000  0.203       -0.003      0.008
money              0.203  1.000       -0.017     -0.050
Weekdaysort       -0.003 -0.017        1.000      0.044
Monthsort          0.008 -0.050        0.044      1.000
```

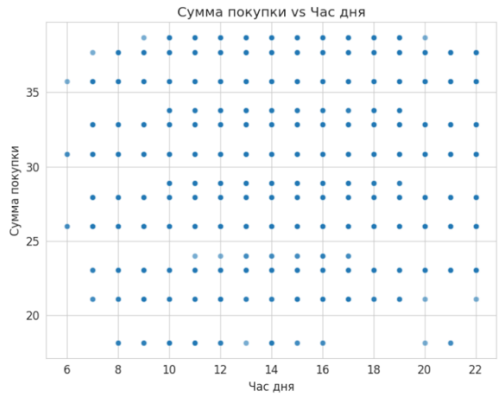


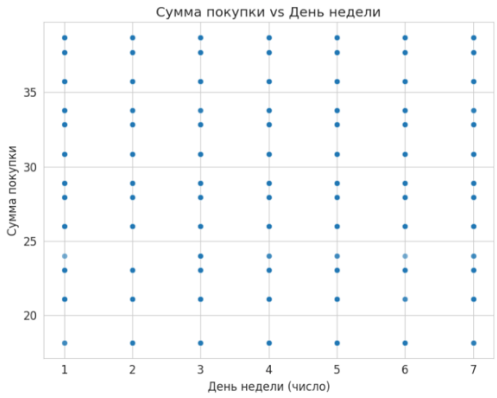
Построение диаграммы рассеяния для пар ключевых признаков

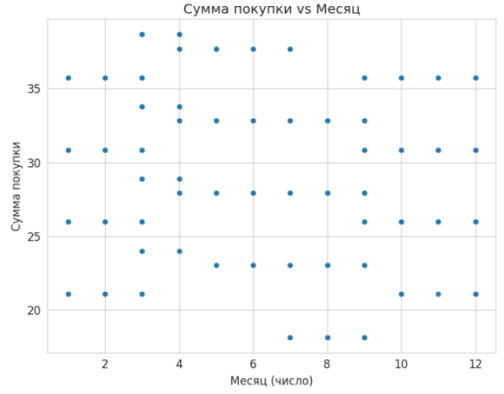
```
# Строим диаграммы рассеяния для пар признаков
g = sns.PairGrid(numeric_df)
g.map_upper(sns.scatterplot, alpha=0.6, s=20)  # Верхний треугольник -
scatterplot
g.map_diag(sns.histplot, kde=True)             # Диагональ -
гистограммы
g.map_lower(sns.kdeplot, cmap='Blues')         # Нижний треугольник -
kdeplot

plt.suptitle('Парные отношения числовых признаков', y=1.02,
fontsize=16)
plt.tight_layout()
plt.show()

# Альтернативно: отдельные scatterplot для ключевых пар
fig, axes = plt.subplots(2, 2, figsize=(15, 12))

# Money vs Hour_of_day
sns.scatterplot(data=df, x='hour_of_day', y='money', ax=axes[0, 0],
alpha=0.6)
axes[0, 0].set_title('Сумма покупки vs Час дня')
axes[0, 0].set_xlabel('Час дня')
axes[0, 0].set_ylabel('Сумма покупки')

# Money vs Weekdaysort
sns.scatterplot(data=df, x='Weekdaysort', y='money', ax=axes[0, 1],
alpha=0.6)
axes[0, 1].set_title('Сумма покупки vs День недели')
axes[0, 1].set_xlabel('День недели (число)')
axes[0, 1].set_ylabel('Сумма покупки')

# Money vs Monthsort
sns.scatterplot(data=df, x='Monthsort', y='money', ax=axes[1, 0],
alpha=0.6)
axes[1, 0].set_title('Сумма покупки vs Месяц')
axes[1, 0].set_xlabel('Месяц (число)')
axes[1, 0].set_ylabel('Сумма покупки')

# Hour_of_day vs Weekdaysort
sns.scatterplot(data=df, x='Weekdaysort', y='hour_of_day', ax=axes[1,
1], alpha=0.6)
axes[1, 1].set_title('Час дня vs День недели')
axes[1, 1].set_xlabel('День недели (число)')
axes[1, 1].set_ylabel('Час дня')

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

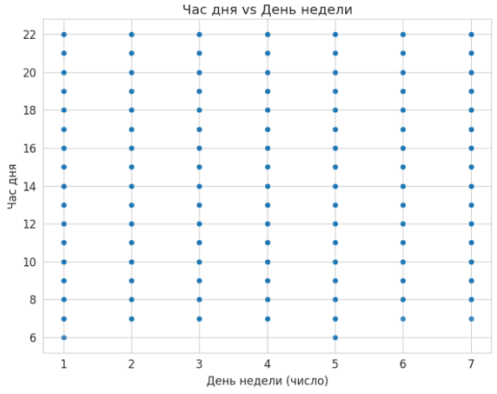
Парные отношения числовых признаков







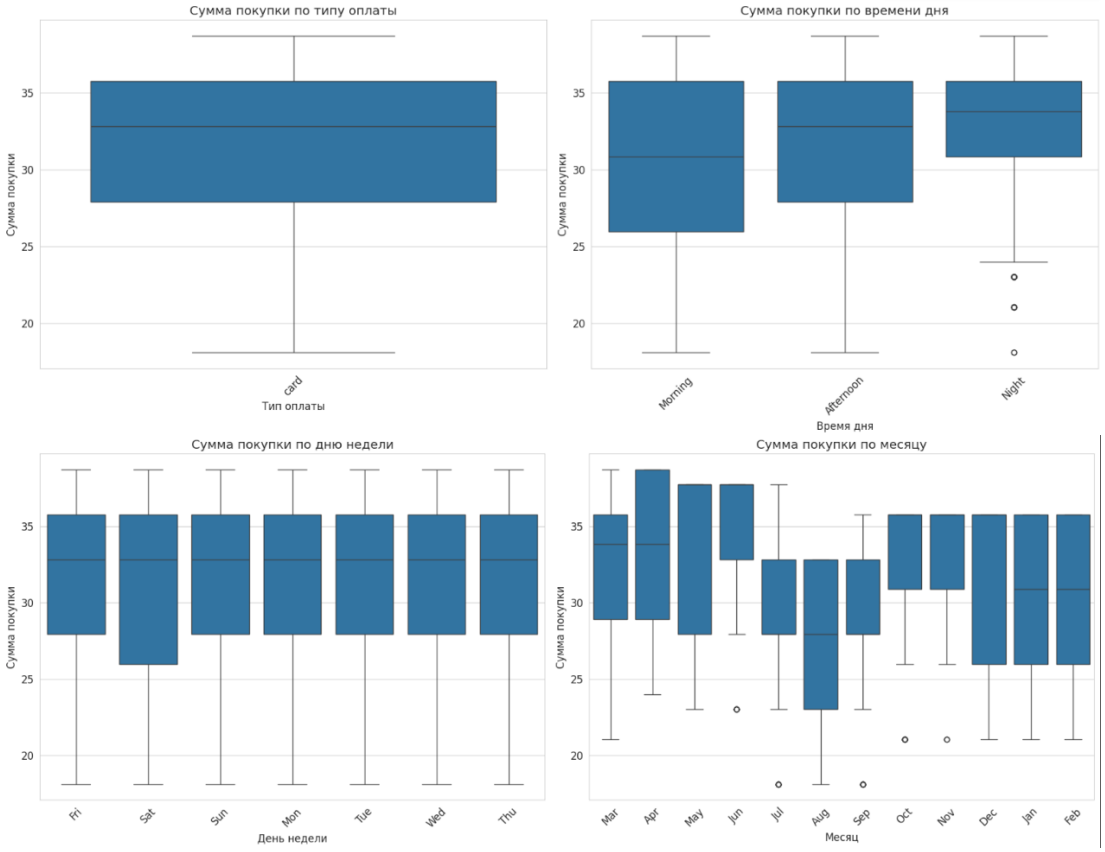


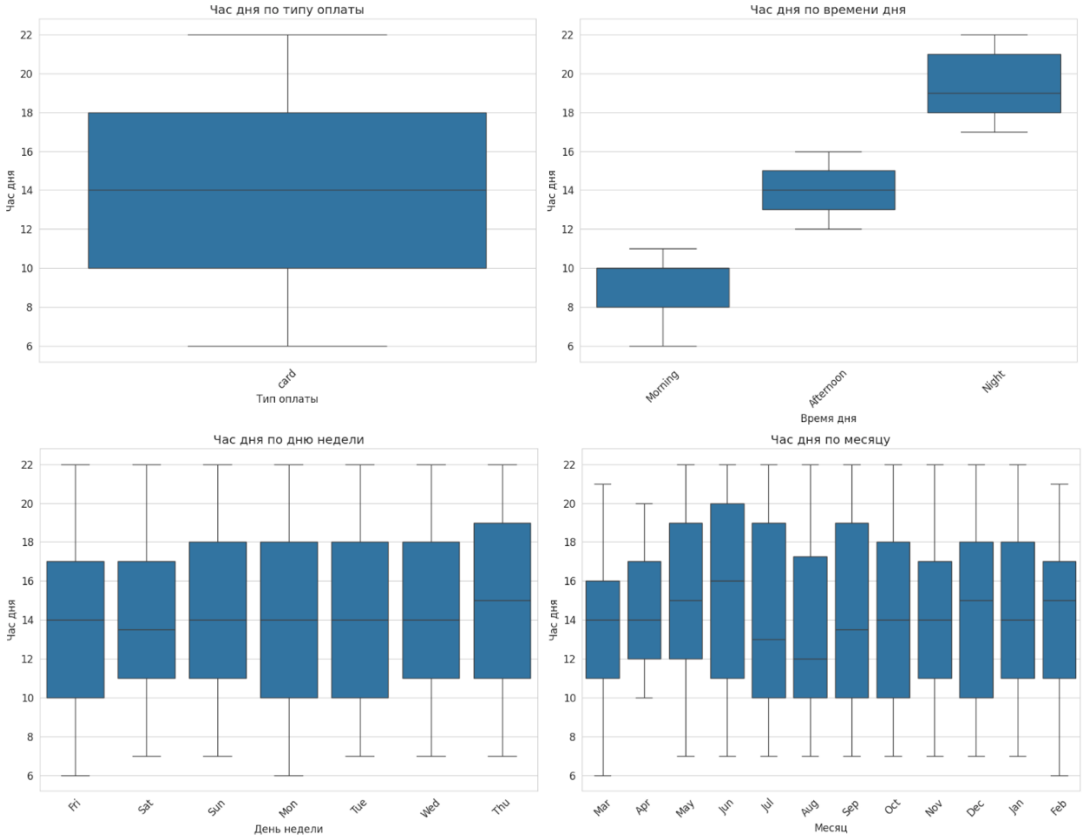


Взаимосвязь категориальных и числовых признаков с помощью boxplot`ов

```
# Создаем фигуру для boxplot'ов
fig, axes = plt.subplots(2, 2, figsize=(18, 14))

# 1. Сумма покупки по типу оплаты
sns.boxplot(data=df, x='cash_type', y='money', ax=axes[0, 0])
axes[0, 0].set_title('Сумма покупки по типу оплаты')
axes[0, 0].set_xlabel('Тип оплаты')
axes[0, 0].set_ylabel('Сумма покупки')
axes[0, 0].tick_params(axis='x', rotation=45)

# 2. Сумма покупки по времени дня
sns.boxplot(data=df, x='Time_of_Day', y='money', ax=axes[0, 1])
axes[0, 1].set_title('Сумма покупки по времени дня')
axes[0, 1].set_xlabel('Время дня')
axes[0, 1].set_ylabel('Сумма покупки')
axes[0, 1].tick_params(axis='x', rotation=45)

# 3. Сумма покупки по дню недели
sns.boxplot(data=df, x='Weekday', y='money', ax=axes[1, 0])
axes[1, 0].set_title('Сумма покупки по дню недели')
axes[1, 0].set_xlabel('День недели')
axes[1, 0].set_ylabel('Сумма покупки')
axes[1, 0].tick_params(axis='x', rotation=45)

# 4. Сумма покупки по месяцу
sns.boxplot(data=df, x='Month_name', y='money', ax=axes[1, 1])
axes[1, 1].set_title('Сумма покупки по месяцу')
axes[1, 1].set_xlabel('Месяц')
axes[1, 1].set_ylabel('Сумма покупки')
axes[1, 1].tick_params(axis='x', rotation=45)

plt.tight_layout()
plt.show()

# Дополнительные boxplot'ы для других отношений
fig, axes = plt.subplots(2, 2, figsize=(18, 14))

# 1. Час дня по типу оплаты
sns.boxplot(data=df, x='cash_type', y='hour_of_day', ax=axes[0, 0])
axes[0, 0].set_title('Час дня по типу оплаты')
axes[0, 0].set_xlabel('Тип оплаты')
axes[0, 0].set_ylabel('Час дня')
axes[0, 0].tick_params(axis='x', rotation=45)

# 2. Час дня по времени дня
sns.boxplot(data=df, x='Time_of_Day', y='hour_of_day', ax=axes[0, 1])
axes[0, 1].set_title('Час дня по времени дня')
axes[0, 1].set_xlabel('Время дня')
axes[0, 1].set_ylabel('Час дня')
```

```
axes[0, 1].tick_params(axis='x', rotation=45)

# 3. Час дня по дню недели
sns.boxplot(data=df, x='Weekday', y='hour_of_day', ax=axes[1, 0])
axes[1, 0].set_title('Час дня по дню недели')
axes[1, 0].set_xlabel('День недели')
axes[1, 0].set_ylabel('Час дня')
axes[1, 0].tick_params(axis='x', rotation=45)

# 4. Час дня по месяцу
sns.boxplot(data=df, x='Month_name', y='hour_of_day', ax=axes[1, 1])
axes[1, 1].set_title('Час дня по месяцу')
axes[1, 1].set_xlabel('Месяц')
axes[1, 1].set_ylabel('Час дня')
axes[1, 1].tick_params(axis='x', rotation=45)

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





Вывод:
На основе матрицы корреляций можно сделать следующие выводы:

Анализ корреляций между признаками Сильных корреляций (|r| > 0.7) не обнаружено между ни одной парой признаков. Все коэффициенты корреляции находятся в диапазоне слабых и очень слабых связей.
Наиболее заметные корреляции (хотя и слабые):

hour_of_day и money (r = 0.203) - слабая положительная корреляция

Это означает, что с увеличением часа дня сумма покупки увеличивается

Monthsort и money (r = -0.050) - очень слабая отрицательная корреляция

Незначительное уменьшение суммы покупки с увеличением номера месяца

Weekdaysort и Monthsort (r = 0.044) - очень слабая положительная корреляция

Минимальная связь между днем недели и месяцем

Зависимость целевой переменной (money) от других признаков
Числовые признаки:
hour_of_day: Обнаружена слабая положительная зависимость (r = 0.203). Это означает, что в более поздние часы дня суммы покупок увеличиваются.

Weekdaysort: практически отсутствует корреляция (r = -0.017), что свидетельствует о том, что день недели почти не влияет на сумму покупки.

Monthsort: очень слабая отрицательная корреляция (r = -0.050), что может указывать на незначительное снижение сумм покупок в течение года.

6. Базовая предобработка

Приведение названия столбцов к удобному формату

```
# Создаем копию DataFrame для предобработки
df_processed = df.copy()

# Приводим названия столбцов к нижнему регистру
df_processed.columns = df_processed.columns.str.lower()
print("Новые названия столбцов:")
print(df_processed.columns.tolist())


Новые названия столбцов:

['hour_of_day', 'cash_type', 'money', 'coffee_name', 'time_of_day',
'weekday', 'month_name', 'weekdaysort', 'monthsort', 'date', 'time']



# Выделяем категориальные признаки

categorical_columns =
df_processed.select_dtypes(include=['object']).columns.tolist()

print("Категориальные признаки для преобразования:")

print(categorical_columns)



# Анализируем уникальные значения в каждом категориальном признаке

for col in categorical_columns:

    unique_count = df_processed[col].nunique()

    print(f"{col}: {unique_count} уникальных значений")

    if unique_count <= 10:  # Показываем значения только если их немного

        print(f"   Значения: {df_processed[col].unique()}")
```

```
Категориальные признаки для преобразования:

['cash_type', 'coffee_name', 'time_of_day', 'weekday', 'month_name',
'date', 'time']

cash_type: 1 уникальных значений

   Значения: ['card']

coffee_name: 8 уникальных значений

   Значения: ['Latte' 'Hot Chocolate' 'Americano' 'Americano with Milk'
'Cocoa'

 'Cortado' 'Espresso' 'Cappuccino']

time_of_day: 3 уникальных значений

   Значения: ['Morning' 'Afternoon' 'Night']

weekday: 7 уникальных значений

   Значения: ['Fri' 'Sat' 'Sun' 'Mon' 'Tue' 'Wed' 'Thu']

month_name: 12 уникальных значений

date: 381 уникальных значений

time: 3547 уникальных значений
```

Преобразование категориальных признаков в числовой формат

```
from sklearn.preprocessing import LabelEncoder
import pandas as pd

# Создаем копию DataFrame для преобразований
df_corrected = df_processed.copy()

# 1. Преобразование cash_type (Label Encoding)
le_cash = LabelEncoder()
df_corrected['cash_type'] =
le_cash.fit_transform(df_corrected['cash_type'])
print("Преобразован cash_type. Соответствие значений:")
for i, class_ in enumerate(le_cash.classes_):
    print(f"  {class_} → {i}")

# 2. Преобразование coffee_name (частотное кодирование)
coffee_freq = df_corrected['coffee_name'].value_counts().to_dict()
df_corrected['coffee_name_freq'] =
df_corrected['coffee_name'].map(coffee_freq)
df_corrected = df_corrected.drop('coffee_name', axis=1)
print("Применено частотное кодирование для coffee_name")

# 3. Преобразование time_of_day (правильное порядковое кодирование)
# Определяем правильный порядок времени суток на основе английских
названий
time_order = {'Morning': 0, 'Afternoon': 1, 'Night': 2}
df_corrected['time_of_day'] = df_corrected['time_of_day'].map(time_order)
```

```
print("Преобразован time_of_day с сохранением порядка")

# 4. Преобразование weekday (правильное порядковое кодирование)
# Определяем порядок дней недели на основе английских названий
weekday_order = {
    'Mon': 0, 'Tue': 1, 'Wed': 2, 'Thu': 3,
    'Fri': 4, 'Sat': 5, 'Sun': 6
}
df_corrected['weekday'] = df_corrected['weekday'].map(weekday_order)
print("Преобразован weekday с сохранением порядка")

# 5. Преобразование month_name (правильное порядковое кодирование)
# Определяем порядок месяцев на основе английских названий
month_order = {
    'Jan': 1, 'Feb': 2, 'Mar': 3, 'Apr': 4,
    'May': 5, 'Jun': 6, 'Jul': 7, 'Aug': 8,
    'Sep': 9, 'Oct': 10, 'Nov': 11, 'Dec': 12
}
df_corrected['month_name'] = df_corrected['month_name'].map(month_order)
print("Преобразован month_name с сохранением порядка")

# 6. Преобразование date и time в datetime features
df_corrected['date'] = pd.to_datetime(df_corrected['date'])
df_corrected['time'] = pd.to_datetime(df_corrected['time'])

# Извлекаем дополнительные временные features
df_corrected['year'] = df_corrected['date'].dt.year
df_corrected['month'] = df_corrected['date'].dt.month
df_corrected['day'] = df_corrected['date'].dt.day
df_corrected['day_of_week'] = df_corrected['date'].dt.dayofweek
df_corrected['hour'] = df_corrected['time'].dt.hour
df_corrected['minute'] = df_corrected['time'].dt.minute

# Удаляем исходные столбцы date и time
df_corrected = df_corrected.drop(['date', 'time'], axis=1)
print("Извлечены features из date и time")

# 7. Проверяем, нужно ли удалить дублирующие столбцы
# Если weekdaysort и monthsort дублируют weekday и month_name, удаляем их
if 'weekdaysort' in df_corrected.columns and 'weekday' in
df_corrected.columns:
    # Сравниваем значения
    if (df_corrected['weekdaysort'] == df_corrected['weekday']).all():
        df_corrected = df_corrected.drop('weekdaysort', axis=1)
        print("Удален дублирующий столбец weekdaysort")

if 'monthsort' in df_corrected.columns and 'month_name' in
df_corrected.columns:
    # Сравниваем значения
    if (df_corrected['monthsort'] == df_corrected['month_name']).all():
        df_corrected = df_corrected.drop('monthsort', axis=1)
```

```
        print("Удален дублирующий столбец monthsort")

# Проверяем результат
print("\nТипы данных после преобразования:")
print(df_corrected.dtypes)

print("\nПервые 5 строк после преобразования:")
print(df_corrected.head())

print("\nРазмерность данных после преобразования:")
print(df_corrected.shape)
```

```
Преобразован cash_type. Соответствие значений:
  card → 0
Применено частотное кодирование для coffee_name
Преобразован time_of_day с сохранением порядка
Преобразован weekday с сохранением порядка
Преобразован month_name с сохранением порядка
Извлечены features из date и time
Удален дублирующий столбец monthsort

Типы данных после преобразования:
hour_of_day           int64
cash_type             int64
money               float64
time_of_day           int64
weekday               int64
month_name            int64
weekdaysort           int64
coffee_name_freq      int64
year                  int32
month                 int32
day                   int32
day_of_week           int32
hour                  int32
minute                int32
dtype: object

Первые 5 строк после преобразования:
   hour_of_day  cash_type  money  time_of_day  weekday  month_name  \
0           10          0   38.7            0        4           3
1           12          0   38.7            1        4           3
2           12          0   38.7            1        4           3
3           13          0   28.9            1        4           3
4           13          0   38.7            1        4           3

   weekdaysort  coffee_name_freq  year  month  day  day_of_week  hour  minute
0            5               757  2024      3    1            4    10      15
1            5               276  2024      3    1            4    12      19
2            5               276  2024      3    1            4    12      20
3            5               564  2024      3    1            4    13      46
```

```
4            5               757  2024      3    1            4    13      48

Размерность данных после преобразования:
(3547, 14)
```

Вывод первых 10 записей после преобразования

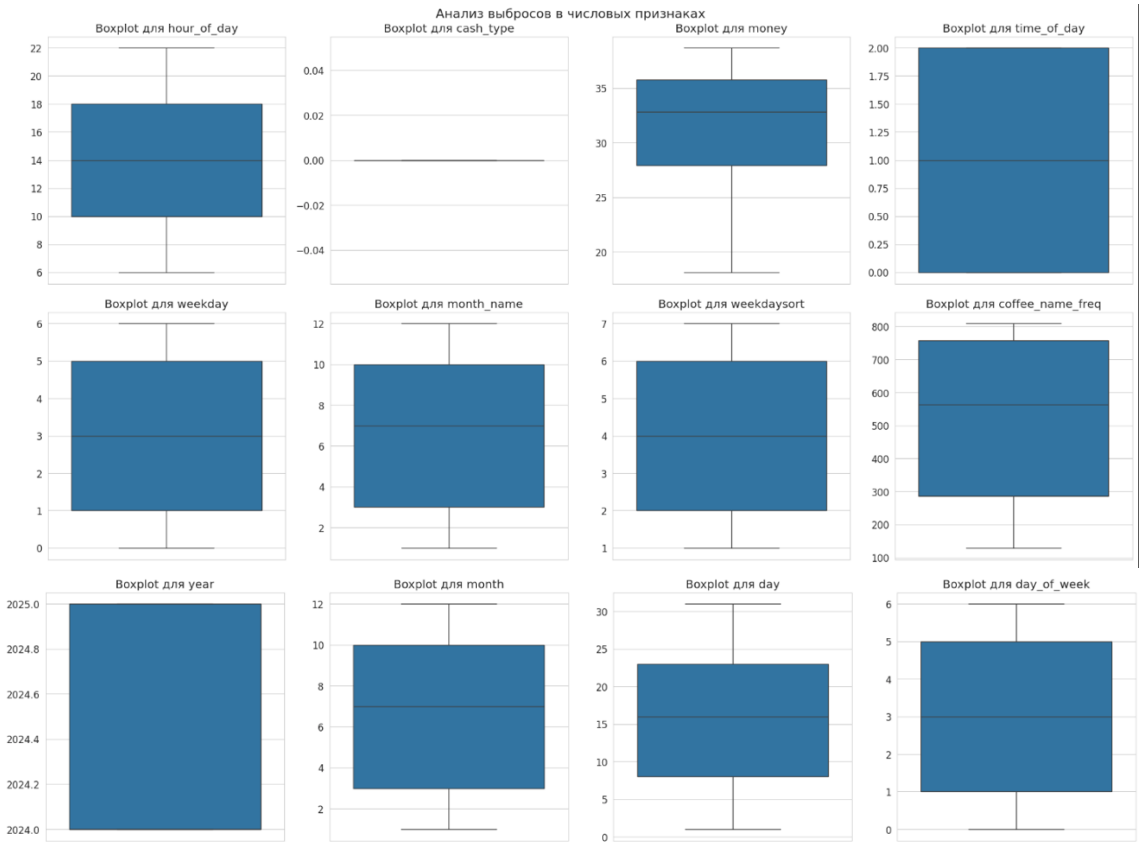
```
   hour_of_day  cash_type  money  time_of_day  weekday  month_name  \
0           10          0   38.7            0        4           3
1           12          0   38.7            1        4           3
2           12          0   38.7            1        4           3
3           13          0   28.9            1        4           3
4           13          0   38.7            1        4           3
5           15          0   33.8            1        4           3
6           16          0   38.7            1        4           3
7           18          0   33.8            2        4           3
8           19          0   38.7            2        4           3
9           19          0   33.8            2        4           3

   weekdaysort  coffee_name_freq  year  month  day  day_of_week  hour  minute
0            5               757  2024      3    1            4    10      15
1            5               276  2024      3    1            4    12      19
2            5               276  2024      3    1            4    12      20
3            5               564  2024      3    1            4    13      46
4            5               757  2024      3    1            4    13      48
5            5               809  2024      3    1            4    15      39
6            5               276  2024      3    1            4    16      19
7            5               809  2024      3    1            4    18      39
8            5               239  2024      3    1            4    19      22
9            5               809  2024      3    1            4    19      23
```

7. Обработка выбросов

Подсчет количества выбросов после преобразования столбцов

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats

# Настройка стиля графиков
sns.set_style("whitegrid")
plt.rcParams['figure.figsize'] = (12, 8)

# Выделяем числовые признаки
numeric_columns =
df_corrected.select_dtypes(include=[np.number]).columns.tolist()
print("Числовые признаки для анализа выбросов:")
print(numeric_columns)
```

```
# Построение boxplot для всех числовых признаков
fig, axes = plt.subplots(3, 4, figsize=(20, 15))
axes = axes.ravel()

for i, column in enumerate(numeric_columns):
    if i < len(axes):  # Убедимся, что не выходим за границы
        sns.boxplot(y=df_corrected[column], ax=axes[i])
        axes[i].set_title(f'Boxplot для {column}')
        axes[i].set_ylabel('')

# Удаляем лишние subplots
for j in range(len(numeric_columns), len(axes)):
    fig.delaxes(axes[j])

plt.suptitle('Анализ выбросов в числовых признаках', fontsize=16, y=0.98)
plt.tight_layout()
plt.show()
```




```
# Расчет показателей асимметрии и эксцесса для выявления признаков с выбросами
outlier_analysis = pd.DataFrame({
    'column': numeric_columns,
    'skewness': [df_corrected[col].skew() for col in numeric_columns],
    'kurtosis': [df_corrected[col].kurtosis() for col in numeric_columns],
    'iqr': [df_corrected[col].quantile(0.75) - df_corrected[col].quantile(0.25) for col in numeric_columns],
```

```
    'outlier_count': [np.sum((df_corrected[col] <
(df_corrected[col].quantile(0.25) - 1.5 *
(df_corrected[col].quantile(0.75) - df_corrected[col].quantile(0.25)))) |
                      (df_corrected[col] >
(df_corrected[col].quantile(0.75) + 1.5 *
(df_corrected[col].quantile(0.75) - df_corrected[col].quantile(0.25)))))
                     for col in numeric_columns]
})

# Сортировка по количеству выбросов
outlier_analysis = outlier_analysis.sort_values('outlier_count',
ascending=False)

print("Анализ выбросов по признакам:")
print(outlier_analysis)

# Выбираем признак с наибольшим количеством выбросов
selected_column = outlier_analysis.iloc[0]['column']
print(f"\nВыбран признак с наибольшим количеством выбросов:
{selected_column}")
```

```
Анализ выбросов по признакам:
              column  skewness  kurtosis     iqr  outlier_count
0        hour_of_day  0.121513 -1.126781    8.00              0
1          cash_type  0.000000  0.000000    0.00              0
2              money -0.544507 -0.671208    7.84              0
3        time_of_day  0.010312 -1.485742    2.00              0
4            weekday  0.082132 -1.224040    4.00              0
5         month_name  0.004314 -1.383442    7.00              0
6        weekdaysort  0.082132 -1.224040    4.00              0
7   coffee_name_freq -0.425259 -1.236409  470.00              0
8               year  1.060418 -0.876008    1.00              0
9              month  0.004314 -1.383442    7.00              0
10               day  0.034922 -1.143229   15.00              0
11       day_of_week  0.082132 -1.224040    4.00              0
12              hour  0.121666 -1.126521    8.00              0
13            minute  0.065050 -1.197986   30.00              0

Выбран признак с наибольшим количеством выбросов: hour_of_day
```

Вывод:
Выбросы отсутствуют.

Заключение

Общие выводы по проведенному анализу:

a) Основными проблемами с данными были некорректный формат данных в базах и наличие лишних столбцов, которые можно было бы объединить в один (например, в датасете Uber данные связанные с отменой или непреднамеренным завершением заказа со стороны водителя или пасажира).
b) Примененные предобработки:
   1. Label Encoding для cash_type

   Что сделано: Преобразование категориального признака в числовой с помощью LabelEncoder()
   Почему применено:

   - cash_type - бинарный или категориальный признак с небольшим количеством уникальных значений
   - Label Encoding сохраняет простоту и не увеличивает размерность данных
   - Подходит для алгоритмов, которые могут работать с порядковыми числами

   2. Frequency Encoding для coffee_name

   Что сделано: Замена названий кофе на частоту их встречаемости в данных
   Почему применено:

   - coffee_name имеет много уникальных значений (высокая кардинальность)
   - One-Hot Encoding создал бы слишком много новых признаков
   - Frequency Encoding сохраняет информацию о популярности напитков
   - Уменьшает размерность данных

   3. Порядковое кодирование для временных признаков

Что сделано: Ручное mapping значений
для time_of_day, weekday, month_name
Почему применено:

- Сохранение семантики порядка: Утро → День → Вечер имеет естественный порядок
- Понятные числовые значения: 0=Утро, 1=День, 2=Вечер
- Сохранение временной последовательности: Понедельник=0, Вторник=1, etc.
- Избежание ложных метрических отношений: Числа отражают порядок, а не математические операции

4. Извлечение features из даты и времени

Что сделано: Разложение date и time на отдельные компоненты
Почему применено:

- Год (year): Для анализа трендов по годам
- Месяц (month): Для выявления сезонности
- День (day): Для анализа внутримесячных паттернов
- День недели (day_of_week): Альтернативное представление дня недели
- Час (hour) и минута (minute): Для детального анализа времени

5. Удаление дублирующих столбцов

Что сделано: Удаление weekdaysort и monthsort при дублировании
Почему применено:

- Избежание мультиколлинеарности: Дублирующие признаки не несут новой информации
- Уменьшение размерности: Упрощение dataset без потери информации
- Улучшение интерпретируемости: Удаление избыточных признаков

c) Проведенная работа позволит позволила нам научиться базово обрабатывать данные и позволит грамотно подбирать и использовать данные для построения моделей.

Ссылка на выбранный датасет на Kaggle:
https://www.kaggle.com/datasets/navjotkaushal/coffee-sales-dataset
Ссылка на google colab:
https://colab.research.google.com/drive/10PKquTpjeJshiz5qE34n-lRj0PT9l1xx?usp=sharing

Код программы