Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

Факультет «Информатика и системы управления»

Кафедра «Системы обработки информации и управления»

Отчет по лабораторной работе №1

«Разведочный анализ данных. Исследование и визуализация данных»

по дисциплине «Технологии машинного обучения»

Выполнил:
студент группы ИУ5Ц-83Б
Пронин В.К.
подпись, дата

Проверил:
к.т.н., доц., Гапанюк Ю.Е.
подпись, дата

2026г.

СОДЕРЖАНИЕ ОТЧЕТА

1. Цель лабораторной работы

Изучение различных методов визуализация данных.

2. Задание

2.1. Выбрать набор данных (датасет).

2.2. Для первой лабораторной работы рекомендуется использовать датасет без пропусков в данных.

2.3. Создать ноутбук, который содержит следующие разделы:

2.4. Текстовое описание выбранного Вами набора данных.

2.5. Основные характеристики датасета.

2.6. Визуальное исследование датасета.

2.7. Информация о корреляции признаков.

2.8. Сформировать отчет и разместить его в своем репозитории на github.

3. Текст программы и результаты

Импортируем библиотеки с помощью команды import. Как правило, все команды import размещают в первой ячейке ноутбука, но мы в этом примере будем подключать все библиотеки последовательно, по мере их использования.

[1]:
```
# Загружаем датасет и подключаем библиотеки
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
sns.set(style="ticks")
```

Используем библиотеку pandas для чтения данных из CSV-файла 'SampleSuperstore.csv' и загрузки их в объект DataFrame под названием data.

[2]:
```
data = pd.read_csv('SampleSuperstore.csv')
```

Выводим первые и последние 5 строк.

[3]:
```
# Выводим первые и последние 5 строк
display(data.head())
display(data.tail())
```

	Ship Mode	Segment	Country	City	State	Postal Code	Region	Category	Sub-Category	Sales	Quantity	Discount	Profit
0	Second Class	Consumer	United States	Henderson	Kentucky	42420	South	Furniture	Bookcases	261.9600	2	0.00	41.9136
1	Second Class	Consumer	United States	Henderson	Kentucky	42420	South	Furniture	Chairs	731.9400	3	0.00	219.5820
2	Second Class	Corporate	United States	Los Angeles	California	90036	West	Office Supplies	Labels	14.6200	2	0.00	6.8714
3	Standard Class	Consumer	United States	Fort Lauderdale	Florida	33311	South	Furniture	Tables	957.5775	5	0.45	-383.0310
4	Standard Class	Consumer	United States	Fort Lauderdale	Florida	33311	South	Office Supplies	Storage	22.3680	2	0.20	2.5164

	Ship Mode	Segment	Country	City	State	Postal Code	Region	Category	Sub-Category	Sales	Quantity	Discount	Profit
9989	Second Class	Consumer	United States	Miami	Florida	33180	South	Furniture	Furnishings	25.248	3	0.2	4.1028
9990	Standard Class	Consumer	United States	Costa Mesa	California	92627	West	Furniture	Furnishings	91.960	2	0.0	15.6332
9991	Standard Class	Consumer	United States	Costa Mesa	California	92627	West	Technology	Phones	258.576	2	0.2	19.3932
9992	Standard Class	Consumer	United States	Costa Mesa	California	92627	West	Office Supplies	Paper	29.600	4	0.0	13.3200
9993	Second Class	Consumer	United States	Westminster	California	92683	West	Office Supplies	Appliances	243.160	2	0.0	72.9480

Вводим размер строки и столбца датасета.

[4]:
```
# Размер датасета (строка, столбец)
data.shape
```

[4]:
```
(9994, 13)
```

Подсчитаем количество строк.

[5]:
```
total_count = data.shape[0]
print('Всего строк: {}'.format(total_count))
```

```
Всего строк: 9994
```

Выводим список колонок.

[6]:
```
# Список колонок
data.columns
```

[6]:
```
Index(['Ship Mode', 'Segment', 'Country', 'City', 'State', 'Postal Code',
       'Region', 'Category', 'Sub-Category', 'Sales', 'Quantity', 'Discount',
       'Profit'],
      dtype='object')
```

Выводим список колонок с типами данных.

[7]:
```
# Список колонок с типами данных
data.dtypes
```

[7]:
```
Ship Mode        object
Segment          object
Country          object
City             object
State            object
Postal Code       int64
Region           object
Category         object
Sub-Category     object
Sales           float64
Quantity          int64
Discount        float64
Profit          float64
dtype: object
```

Проверим наличие пустых значений.

[8]:
```
# Проверим наличие пустых значений
# Цикл по колонкам датасета
for col in data.columns:
    # Количество пустых значений - все значения заполнены
    temp_null_count = data[data[col].isnull()].shape[0]
    print('{} - {}'.format(col, temp_null_count))
```

```
Ship Mode - 0
Segment - 0
Country - 0
City - 0
State - 0
Postal Code - 0
Region - 0
Category - 0
Sub-Category - 0
Sales - 0
Quantity - 0
Discount - 0
Profit - 0
```

Выводим основные статистические характеристики набора данных

[9]:
```
# Основные статистические характеристки набора данных
data.describe()
```

[9]:

	Postal Code	Sales	Quantity	Discount	Profit
count	9994.000000	9994.000000	9994.000000	9994.000000	9994.000000
mean	55190.379428	229.858001	3.789574	0.156203	28.656896
std	32063.693350	623.245101	2.225110	0.206452	234.260108
min	1040.000000	0.444000	1.000000	0.000000	-6599.978000
25%	23223.000000	17.280000	2.000000	0.000000	1.728750
50%	56430.500000	54.490000	3.000000	0.200000	8.666500
75%	90008.000000	209.940000	5.000000	0.200000	29.364000
max	99301.000000	22638.480000	14.000000	0.800000	8399.976000

Определим уникальные значения для целевого признака

```
[10]: # Определим уникальные значения для целевого признака
      data['Quantity'].unique()
```

```
[10]: array([ 2,  3,  5,  7,  4,  6,  9,  1,  8, 14, 11, 13, 10, 12],
            dtype=int64)
```

Целевой признак является бинарным и содержит только значения от 2 до 12.

Визуальное исследование датасета.

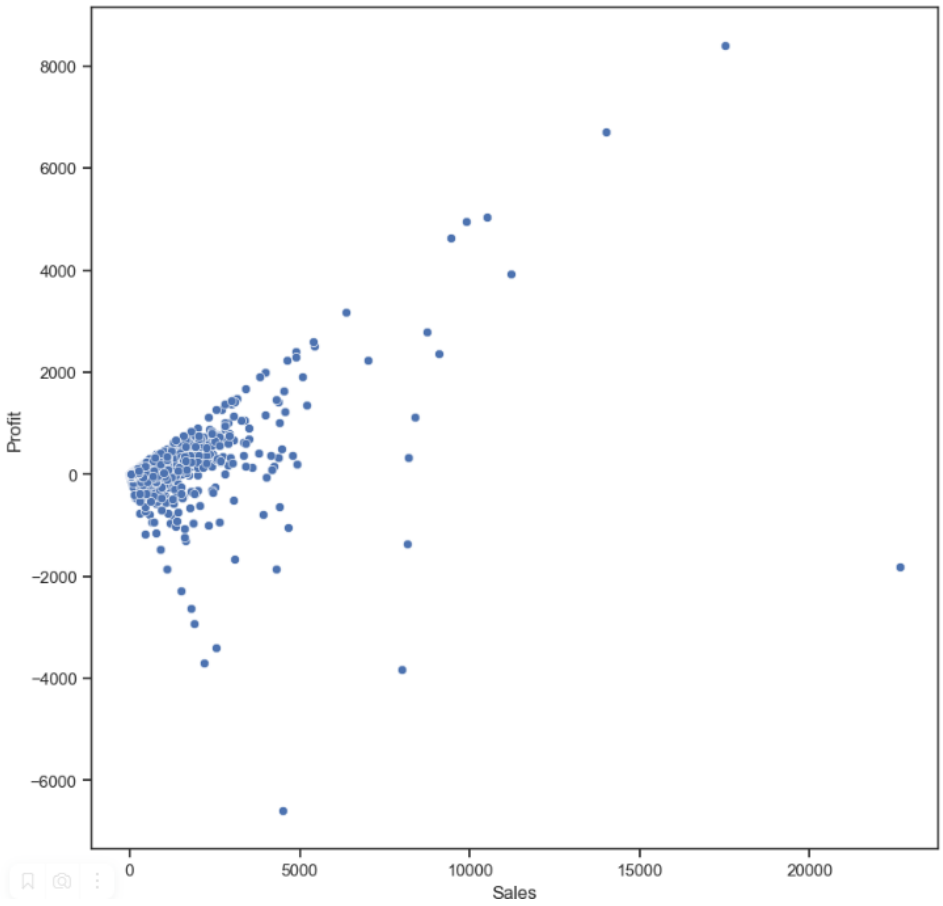
Для визуального исследования могут быть использованы различные виды диаграмм, мы построим только некоторые варианты диаграмм, которые используются достаточно часто.

Диаграмма рассеяния

Позволяет построить распределение двух колонок данных и визуально обнаружить наличие зависимости. Не предполагается, что значения упорядочены.

```
[11]: fig, ax = plt.subplots(figsize=(10,10))
      sns.scatterplot(ax=ax, x='Sales', y='Profit', data=data)
```

```
[11]: <Axes: xlabel='Sales', ylabel='Profit'>
```



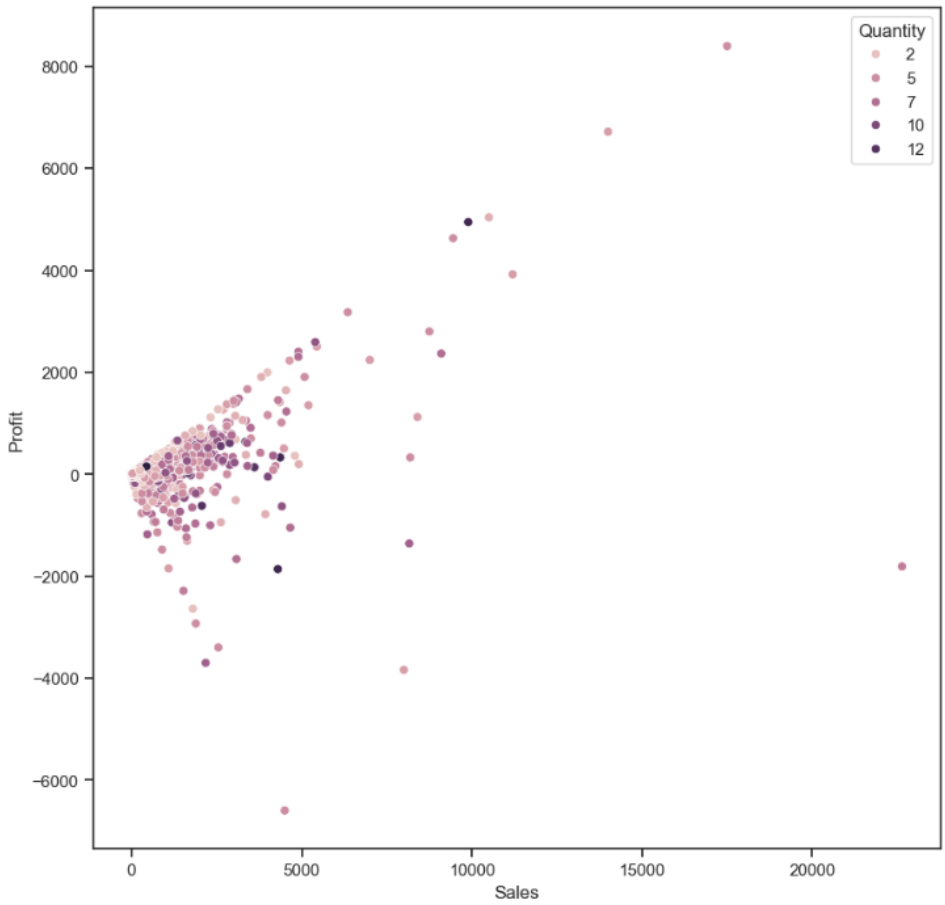
Можно видеть что между полями Sales и Profit присутствует не совсем линейная зависимость.

Посмотрим насколько на эту зависимость влияет целевой признак.

Диаграмма рассеяния, где каждая точка представляет собой товар с определенными значениями продаж, прибыли и количества проданных товаров. Цвет каждой точки показывает количество проданных товаров.

[12]:
```
fig, ax = plt.subplots(figsize=(10,10))
sns.scatterplot(ax=ax, x='Sales', y='Profit', data=data, hue='Quantity')
```

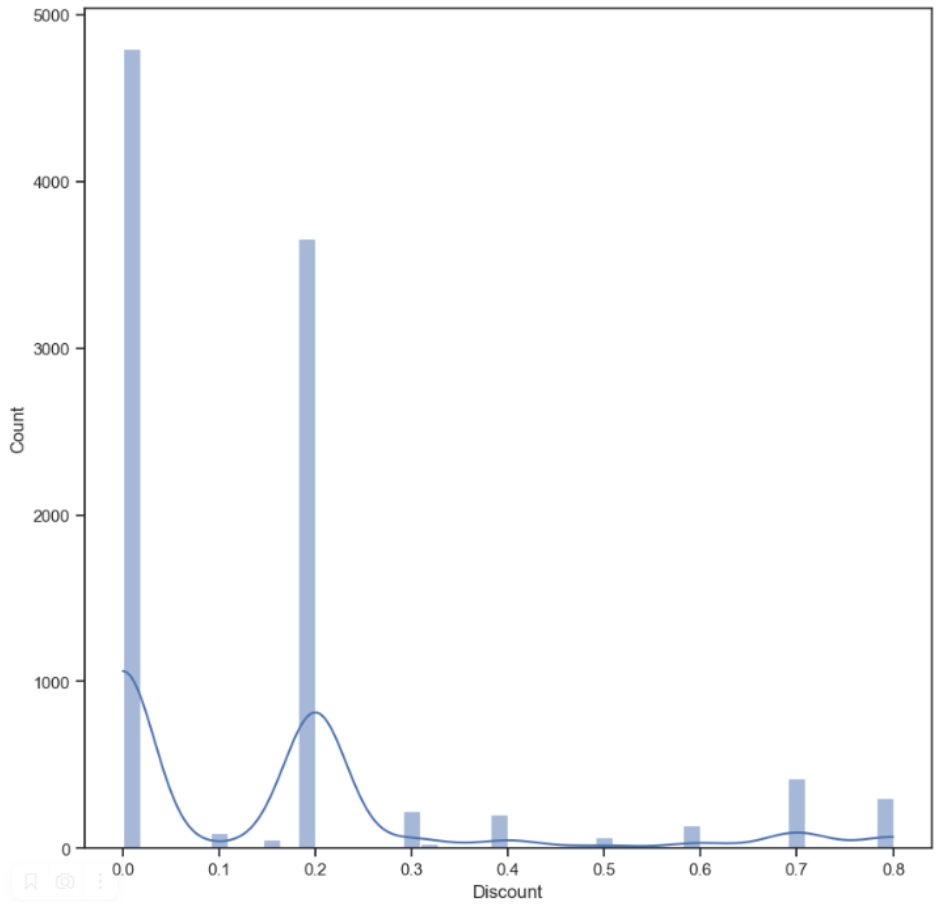
[12]:
```
<Axes: xlabel='Sales', ylabel='Profit'>
```



Гистограмма

Позволяет оценить плотность вероятности распределения данных. Оцениваем количество вероятности распределения данных

[13]:
```
# Оцениваем количество вероятности распределения данных
fig, ax = plt.subplots(figsize=(10, 10))
sns.histplot(data['Discount'], kde=True)
plt.show()
```

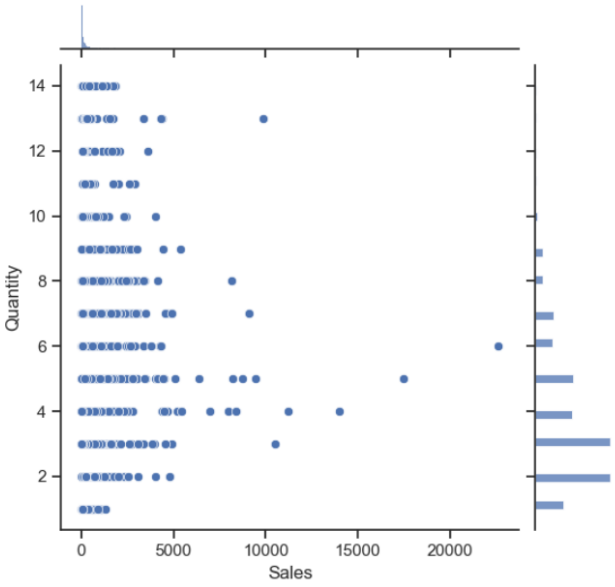


Jointplot

Комбинация гистограмм и диаграмм рассеивания.

[14]:
```
sns.jointplot(x='Sales', y='Quantity', data=data)
```

[14]:
```
<seaborn.axisgrid.JointGrid at 0x23940816970>
```

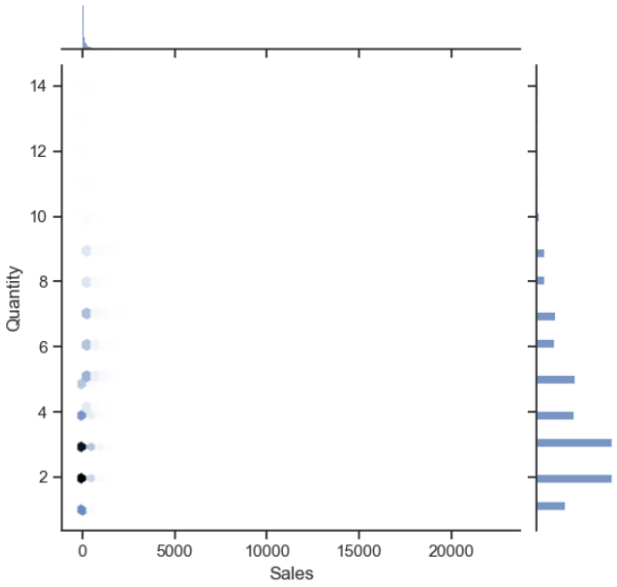


Строим совместный график для двух переменных 'Sales' и 'Quantity'. Ось

x представляет значения 'Sales', ось y - значения 'Quantity'. В данном случае, kind="hex" указывает использовать шестиугольные ячейки (hexbin) для представления данных, что полезно для визуализации плотности точек.

```
[15]: sns.jointplot(x='Sales', y='Quantity', data=data, kind="hex")
```

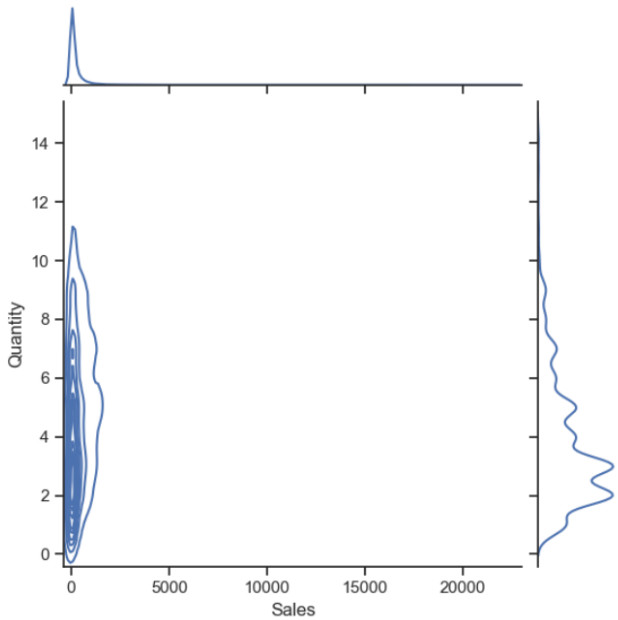
```
[15]: <seaborn.axisgrid.JointGrid at 0x2326e783370>
```



Совместные графики с ядерной оценкой плотности полезны для визуализации распределения и взаимосвязи между двумя переменными. График KDE создает контурные линии, показывающие уровни плотности в различных областях графика. Это позволяет лучше оценить форму распределения и сильные стороны взаимосвязи между 'Sales' и 'Quantity'.

```
[16]: sns.jointplot(x='Sales', y='Quantity', data=data, kind="kde")
```

```
[16]: <seaborn.axisgrid.JointGrid at 0x23945b5b940>
```



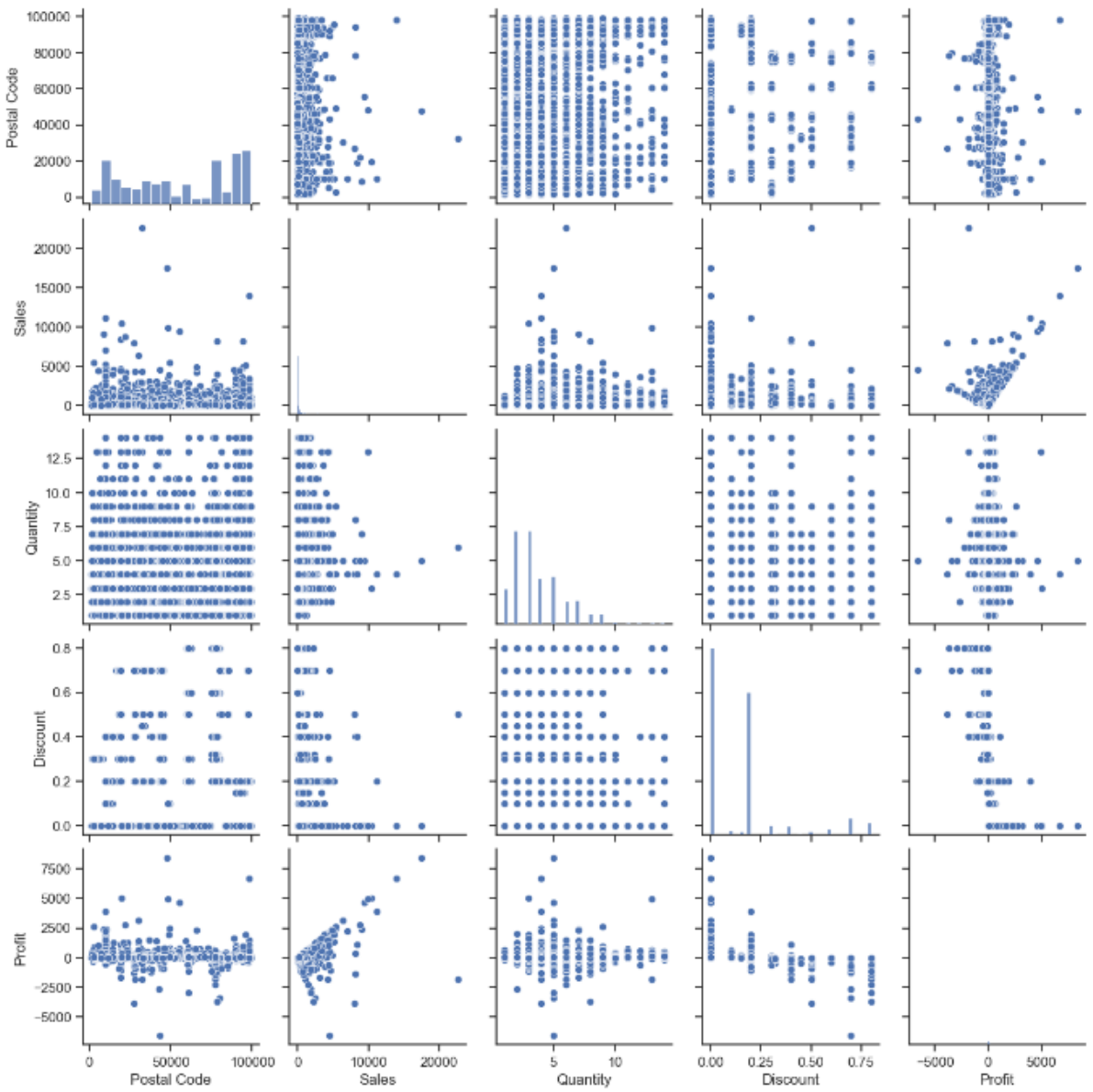
"Парные диаграммы"

Комбинация гистограмм и диаграмм рассеивания для всего набора данных.

Выводится матрица графиков. На пересечении строки и столбца, которые соответствуют двум показателям, строится диаграмма рассеивания. В главной диагонали матрицы строятся гистограммы распределения соответствующих показателей.

```
[17]: sns.pairplot(data)
```

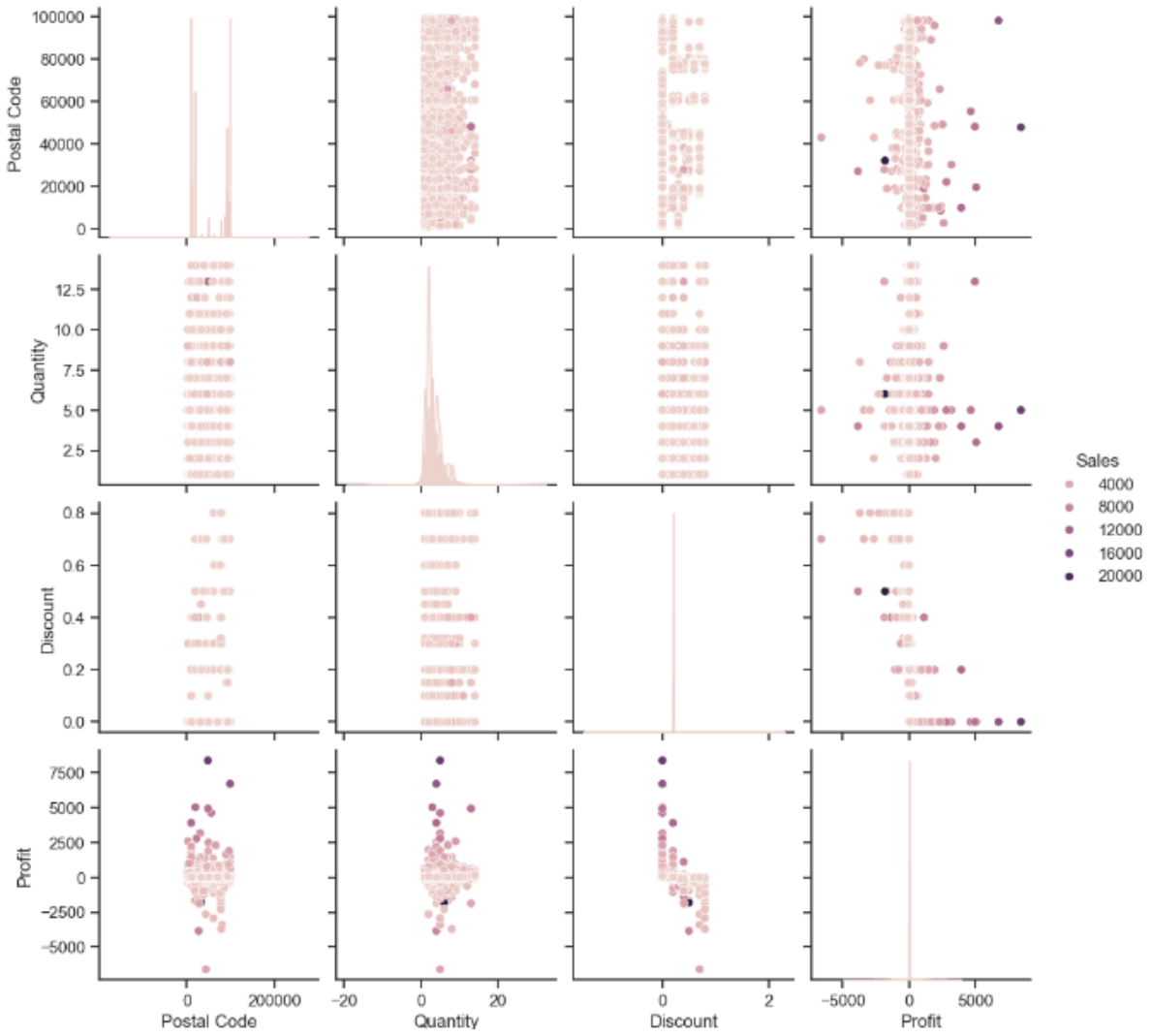
```
[17]: <seaborn.axisgrid.PairGrid at 0x23945fe6070>
```



С помощью параметра "hue" возможна группировка по значениям какого-либо признака.

[18]:
```
# С помощью параметра "hue" возможна группировка по значениям какого-либо признака.
sns.pairplot(data, hue="Sales")
```

[18]:
```
<seaborn.axisgrid.PairGrid at 0x2394da19550>
```

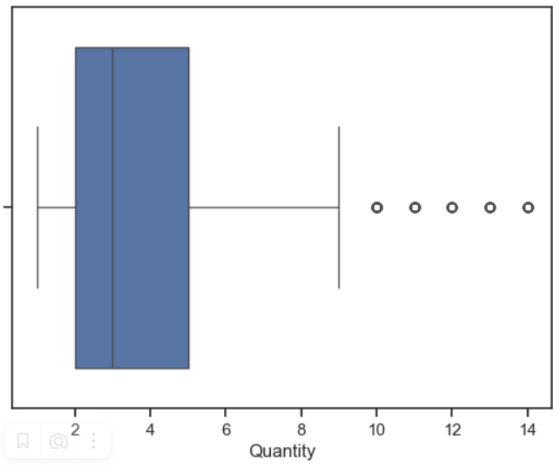


Ящик с усами

Отображает одномерное распределение вероятности.

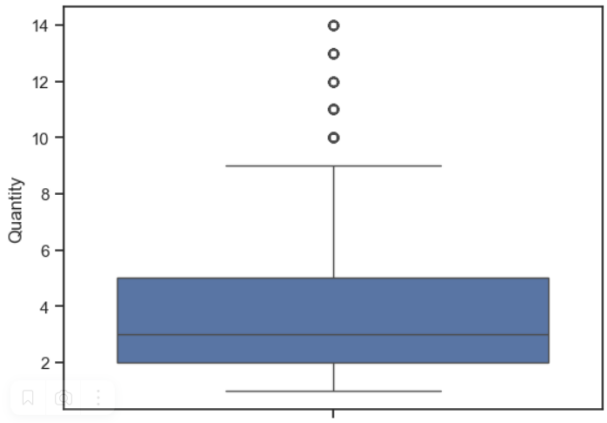
[19]:
```
sns.boxplot(x=data['Quantity'])
```

[19]:
```
<Axes: xlabel='Quantity'>
```



```
[20]: # По вертикали
sns.boxplot(y=data['Quantity'])
```

```
[20]: <Axes: ylabel='Quantity'>
```



Violin plot

Похоже на предыдущую диаграмму, но по краям отображаются распределения плотности.

```
[21]: sns.violinplot(x=data['Quantity'])
```

```
[21]: <Axes: xlabel='Quantity'>
```

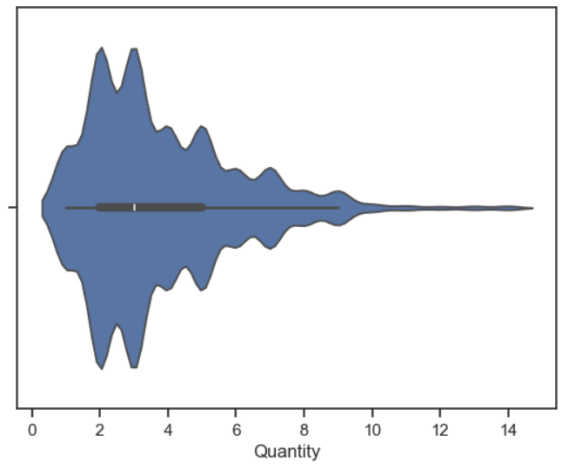
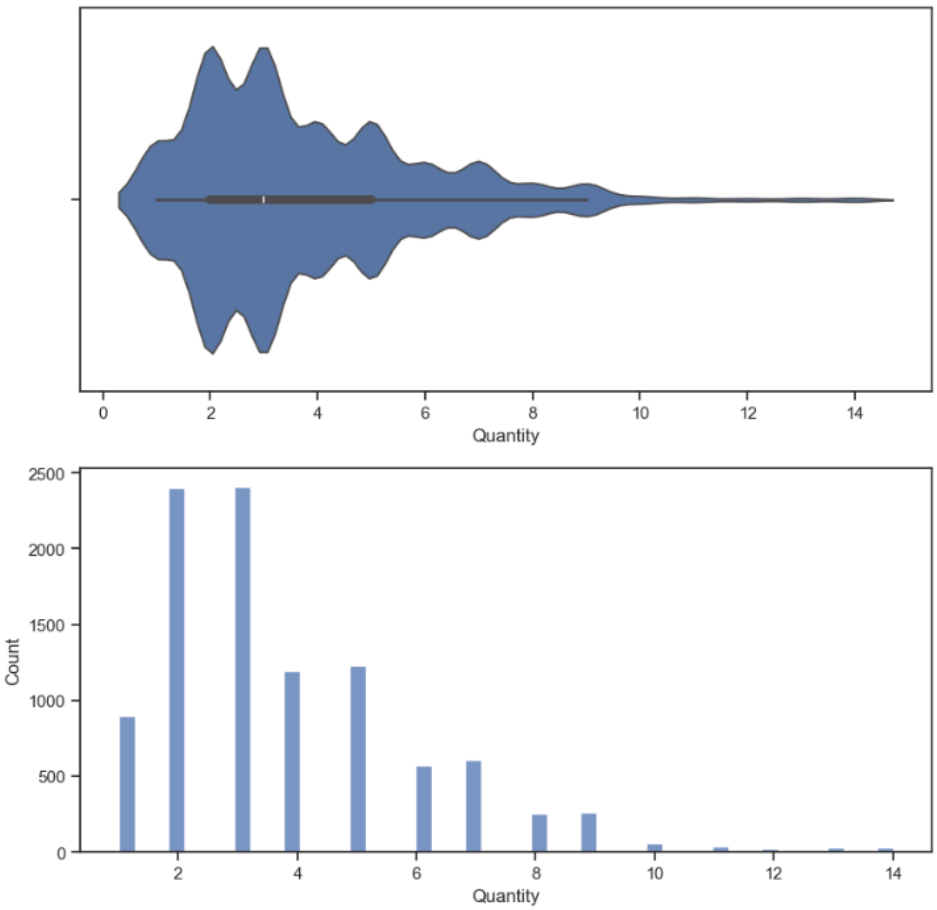


График скрипичного графика в верхнем под графике и гистограммы в нижнем под графике для переменной 'Quantity'. Это позволяет визуально оценить распределение и характеристики переменной 'Quantity'.

```
[22]: fig, ax = plt.subplots(2, 1, figsize=(10,10))
      sns.violinplot(ax=ax[0], x=data['Quantity'])
      sns.histplot(data['Quantity'], ax=ax[1])
```

```
[22]: <Axes: xlabel='Quantity', ylabel='Count'>
```



Информация о корреляции признаков

Проверка корреляции признаков позволяет решить две задачи:

1. Понять какие признаки (колонки датасета) наиболее сильно коррелируют с целевым признаком. Именно эти признаки будут наиболее информативными для моделей машинного обучения. Признаки, которые слабо коррелируют с целевым признаком, можно попробовать исключить из построения модели, иногда это повышает качество модели. Нужно отметить, что некоторые алгоритмы машинного обучения автоматически определяют ценность того или иного признака для построения модели.
2. Понять какие нецелевые признаки линейно зависимы между собой. Линейно зависимые признаки, как правило, очень плохо влияют на качество моделей. Поэтому если несколько признаков линейно зависимы, то для построения модели из них выбирают какой-то один признак.

Получим информацию и преобразуем типы.

[23]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9994 entries, 0 to 9993
Data columns (total 13 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   Ship Mode     9994 non-null   object 
 1   Segment       9994 non-null   object 
 2   Country       9994 non-null   object 
 3   City          9994 non-null   object 
 4   State         9994 non-null   object 
 5   Postal Code   9994 non-null   int64  
 6   Region        9994 non-null   object 
 7   Category      9994 non-null   object 
 8   Sub-Category  9994 non-null   object 
 9   Sales         9994 non-null   float64
 10  Quantity      9994 non-null   int64  
 11  Discount      9994 non-null   float64
 12  Profit        9994 non-null   float64
dtypes: float64(3), int64(2), object(8)
memory usage: 1015.1+ KB
```

[24]:
```
# Преобразуем типы
data_buff = data[['Postal Code', 'Sales', 'Quantity', 'Discount', 'Profit']].copy()
```

Используем для вычисления корреляционной матрицы для числовых столбцов в DataFrame data_buff. Каждый элемент этой матрицы представляет собой коэффициент корреляции между соответствующими парами столбцов.

[25]:
```
data_buff.corr()
```

[25]:

	Postal Code	Sales	Quantity	Discount	Profit
Postal Code	1.000000	-0.023854	0.012761	0.058443	-0.029961
Sales	-0.023854	1.000000	0.200795	-0.028190	0.479064
Quantity	0.012761	0.200795	1.000000	0.008623	0.066253
Discount	0.058443	-0.028190	0.008623	1.000000	-0.219487
Profit	-0.029961	0.479064	0.066253	-0.219487	1.000000

На основе корреляционной матрицы можно сделать следующие выводы:

1. Postal Code и другие переменные:

- Postal Code не имеет сильной линейной корреляции с Sales, Quantity, Discount и Profit. Коэффициент корреляции близок к нулю в каждом случае.

2. Sales и другие переменные:

- Sales имеет положительную корреляцию с Profit (коэффициент 0.479), что может указывать на то, что увеличение продаж связано с увеличением прибыли. Это может быть ожидаемым результатом.
- Sales также имеет слабую положительную корреляцию с Quantity (коэффициент 0.200), что может означать, что более высокие объемы продаж могут соответствовать большему количеству проданных товаров.

3. Quantity и другие переменные:

- Quantity имеет слабую положительную корреляцию с Sales (коэффициент 0.200), что подтверждает наблюдение о том, что более высокие объемы продаж могут соответствовать большему количеству проданных товаров.

4. Discount и другие переменные:

- Discount имеет небольшую отрицательную корреляцию с Profit (коэффициент -0.219), что может свидетельствовать о том, что предоставление скидок может быть связано с уменьшением прибыли.

5. Profit и другие переменные:

- Profit имеет положительную корреляцию с Sales (коэффициент 0.479), что было отмечено выше.

- Profit также имеет слабую положительную корреляцию с Quantity (коэффициент 0.066), что может указывать на то, что более высокая прибыль может быть связана с большим количеством проданных товаров.

Общий вывод: Некоторые переменные, такие как Sales и Quantity, могут иметь положительную корреляцию с прибылью, в то время как другие, такие как Discount, могут иметь отрицательное влияние на прибыль. Эти корреляции могут помочь в формулировании гипотез и дальнейшем анализе данных. Однако, корреляция не всегда означает причинно-следственную связь, и дополнительные исследования могут потребоваться для полного понимания взаимосвязей.

По умолчанию при построении матрицы используется коэффициент корреляции Пирсона. Возможно также построить корреляционную матрицу на основе коэффициентов корреляции Кендалла и Спирмена. На практике три метода редко дают значимые различия.

[26]:
```
data_buff.corr(method='pearson')
```

[26]:

	Postal Code	Sales	Quantity	Discount	Profit
Postal Code	1.000000	-0.023854	0.012761	0.058443	-0.029961
Sales	-0.023854	1.000000	0.200795	-0.028190	0.479064
Quantity	0.012761	0.200795	1.000000	0.008623	0.066253
Discount	0.058443	-0.028190	0.008623	1.000000	-0.219487
Profit	-0.029961	0.479064	0.066253	-0.219487	1.000000

[27]:
```
data_buff.corr(method='kendall')
```

[27]:

	Postal Code	Sales	Quantity	Discount	Profit
Postal Code	1.000000	-0.001465	0.010015	0.037496	-0.003111
Sales	-0.001465	1.000000	0.238978	-0.041826	0.452118
Quantity	0.010015	0.238978	1.000000	-0.000698	0.171843
Discount	0.037496	-0.041826	-0.000698	1.000000	-0.428205
Profit	-0.003111	0.452118	0.171843	-0.428205	1.000000

[28]:
```
data_buff.corr(method='spearman')
```

[28]:

	Postal Code	Sales	Quantity	Discount	Profit
Postal Code	1.000000	-0.002059	0.013952	0.052793	-0.005451
Sales	-0.002059	1.000000	0.327426	-0.056969	0.518407
Quantity	0.013952	0.327426	1.000000	-0.000878	0.234491
Discount	0.052793	-0.056969	-0.000878	1.000000	-0.543350
Profit	-0.005451	0.518407	0.234491	-0.543350	1.000000

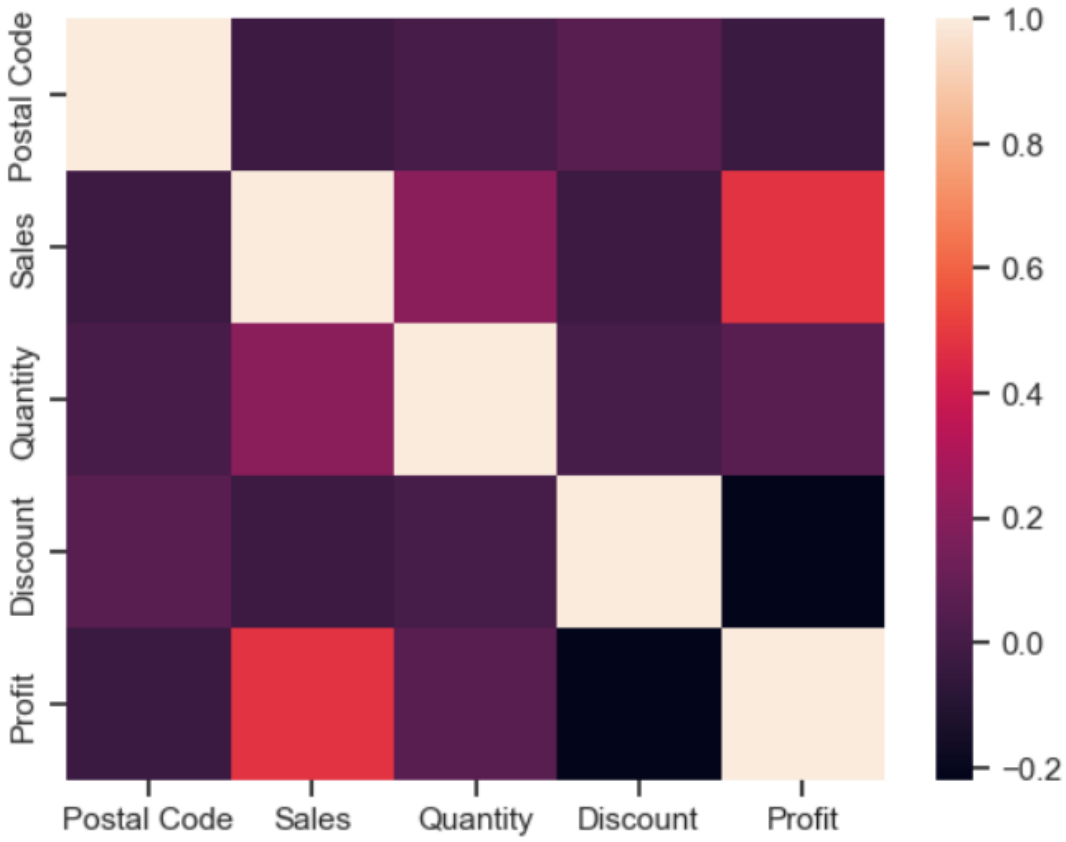
В случае большого количества признаков анализ числовой корреляционной матрицы становится неудобен.

Для визуализации корреляционной матрицы будем использовать "тепловую карту" heatmap которая показывает степень корреляции различными цветами.

Используем метод heatmap библиотеки seaborn.

```
[29]: sns.heatmap(data_buff.corr())
```

```
[29]: <Axes: >
```



Вывод значений в ячейках

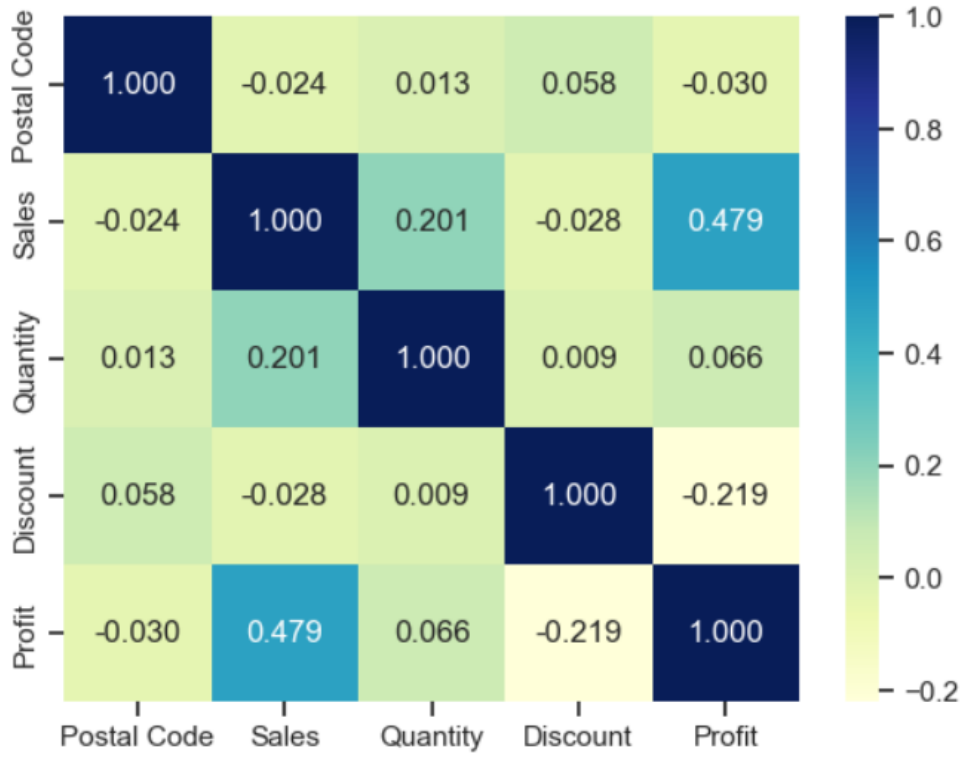
[30]:
```
# Вывод значений в ячейках
sns.heatmap(data_buff.corr(), annot=True, fmt='.3f')
```

[30]: <Axes: >

Изменение цветовой гаммы

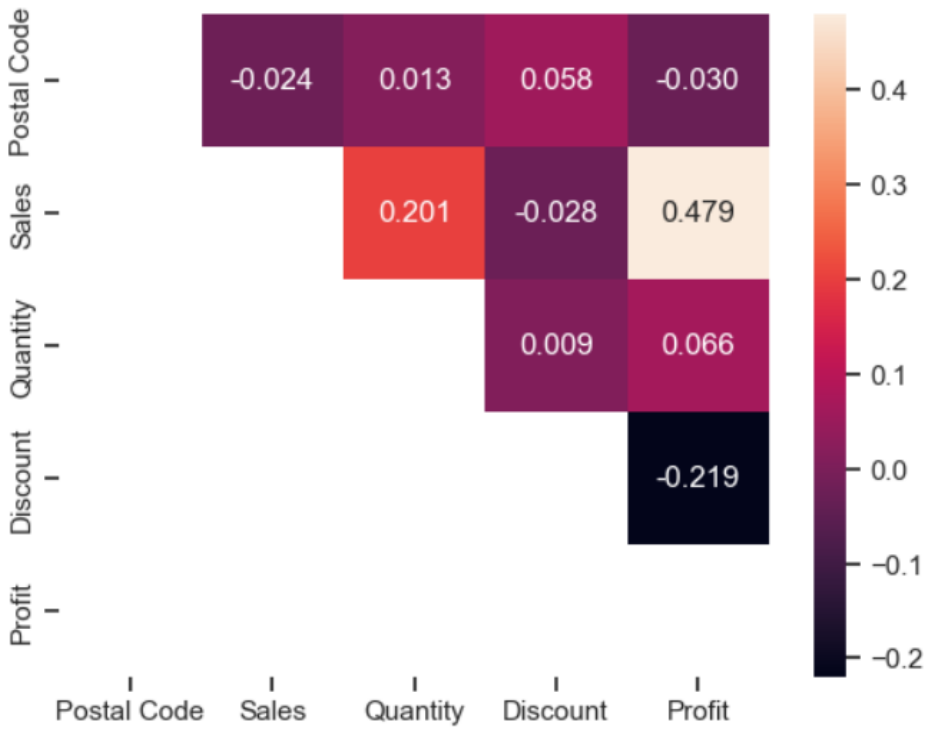
[31]:
```
# Изменение цветовой гаммы
sns.heatmap(data_buff.corr(), cmap='YlGnBu', annot=True, fmt='.3f')
```

[31]: <Axes: >



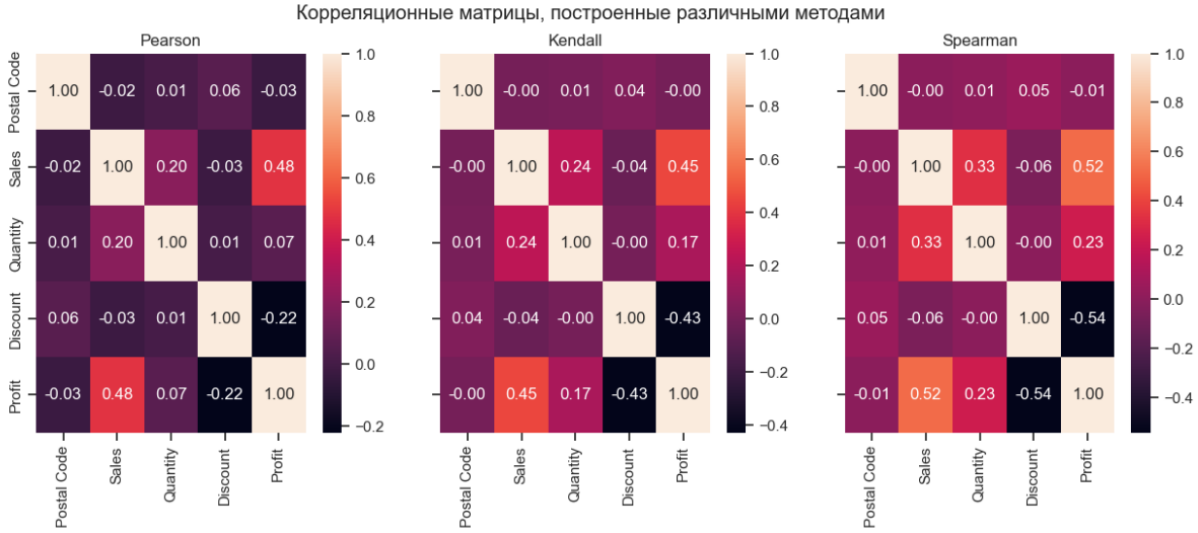
Треугольный вариант матрицы

[32]:
```
# Треугольный вариант матрицы
numeric_data = data.select_dtypes(include='number')
mask = np.zeros_like(numeric_data.corr(), dtype=bool)
# Оставим верхнюю часть матрицы
mask[np.tril_indices_from(mask)] = True
sns.heatmap(numeric_data.corr(), mask=mask, annot=True, fmt='.3f')
plt.show()
```



Корреляционные матрицы, построенные различными методами

[33]:
```
fig, ax = plt.subplots(1, 3, sharex='col', sharey='row', figsize=(15,5))
sns.heatmap(data_buff.corr(method='pearson'), ax=ax[0], annot=True, fmt='.2f')
sns.heatmap(data_buff.corr(method='kendall'), ax=ax[1], annot=True, fmt='.2f')
sns.heatmap(data_buff.corr(method='spearman'), ax=ax[2], annot=True, fmt='.2f')
fig.suptitle('Корреляционные матрицы, построенные различными методами')
ax[0].title.set_text('Pearson')
ax[1].title.set_text('Kendall')
ax[2].title.set_text('Spearman')
```



Необходимо отметить, что тепловая карта не очень хорошо подходит для определения корреляции нецелевых признаков между собой.

В примере тепловая карта помогает определить значимую корреляцию между признаками Sales и Quantity, следовательно только один из этих признаков можно включать в модель.

Но в реальной модели могут быть сотни признаков и коррелирующие признаки могут образовывать группы, состоящие более чем из двух признаков. Увидеть такие группы с помощью тепловой карты сложно.

4. Вывод

В ходе выполнения лабораторной работы изучили различные методы визуализация данных.