Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное 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 рубежному контролю №1

«Технологии разведочного анализа и обработки данных»

по дисциплине «Технологии машинного обучения»

Вариант №27

Выполнил:
студент группы ИУ5Ц-84Б
Папин А.В.
подпись, дата

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

2024 г.

СОДЕРЖАНИЕ ОТЧЕТА

1. Примечания:

Если в Вашем наборе данных отсутствуют данные, необходимые для решения задачи, создайте их искусственно. Например, если отсутствуют категориальные признаки, создайте категориальный признак на основе числового. Если отсутствуют пропуски, замените на пропуски часть значений в одном или нескольких признаках.

Также Вы можете дополнительно использовать датасеты, содержащие необходимые данные, например использовать дополнительный датасет, содержащий пропуски.

2. Дополнительные требования по группам:

1. Для студентов группы ИУ5-64Б, ИУ5Ц-84Б - для произвольной колонки данных построить график "Скрипичная диаграмма (violin plot)".

Задача №4

Для заданного набора данных постройте основные графики, входящие в этап разведочного анализа данных. В случае наличия пропусков в данных удалите строки или колонки, содержащие пропуски. Какие графики Вы построили и почему? Какие выводы о наборе данных Вы можете сделать на основании построенных графиков?

Наборы данных: https://www.kaggle.com/carlolepelaars/toy-dataset

3. Листинг

3.1. Подключение библиотеки и получение датасета

```
# Подключаем все необходимые библиотеки
import pandas as pd
import seaborn as sb
import matplotlib.pyplot as plt
```

```
# Получаем датасет
try:
    df = pd.read_csv('toy_dataset.csv', delimiter=',')
    print('Загружен датасет')
except Exception as ex:
    print('Отсутствует датасет. Проверьте путь файла')
    print('Error:', ex)
```

```
Загружен датасет
```

3.2. Изучение данных

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150000 entries, 0 to 149999
Data columns (total 6 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   Number   150000 non-null  int64  
 1   City     150000 non-null  object 
 2   Gender   150000 non-null  object 
 3   Age      150000 non-null  int64  
 4   Income   150000 non-null  float64
 5   Illness  150000 non-null  object 
dtypes: float64(1), int64(2), object(3)
memory usage: 6.9+ MB
```

```
# Привести названия всех колонок к нижнему регистру
df.columns = df.columns.str.lower()
```

```
df.head()
```

	number	city	gender	age	income	illness
0	1	Dallas	Male	41	40367.0	No
1	2	Dallas	Male	54	45084.0	No
2	3	Dallas	Male	42	52483.0	No
3	4	Dallas	Male	40	40941.0	No
4	5	Dallas	Male	46	50289.0	No

```
df.tail()
```

	number	city	gender	age	income	illness
149995	149996	Austin	Male	48	93669.0	No
149996	149997	Austin	Male	25	96748.0	No
149997	149998	Austin	Male	26	111885.0	No
149998	149999	Austin	Male	25	111878.0	No
149999	150000	Austin	Female	37	87251.0	No

Рассмотрим описательную статистику

```
df.describe()
```

	number	age	income
count	150000.000000	150000.000000	150000.000000
mean	75000.500000	44.950200	91252.798273
std	43301.414527	11.572486	24989.500948
min	1.000000	25.000000	-654.000000
25%	37500.750000	35.000000	80867.750000
50%	75000.500000	45.000000	93655.000000
75%	112500.250000	55.000000	104519.000000
max	150000.000000	65.000000	177157.000000

Анализ описательной статистики

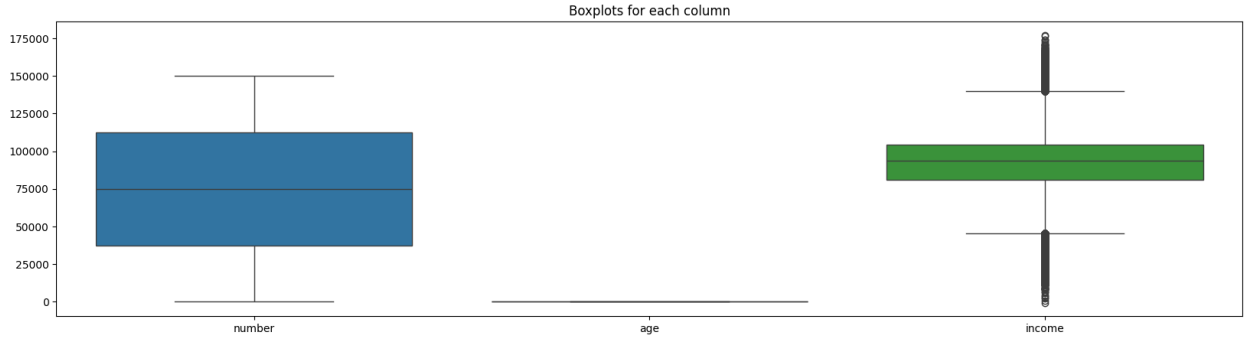
number: Индексы распределены равномерно от 1 до 150000.

age: Средний возраст примерно 45 лет, с небольшим стандартным отклонением. Минимальный возраст 25 лет, максимальный - 65 лет.

income: Средний годовой доход около $91253, но с отрицательным минимальным значением, что может быть артефактом или ошибкой в данных.

3.3. Ящик с усами (Анализ выбросов)

```
plt.figure(figsize=(20, 5))
sb.boxplot(data=df)
plt.title("Boxplots for each column")
plt.show()
```

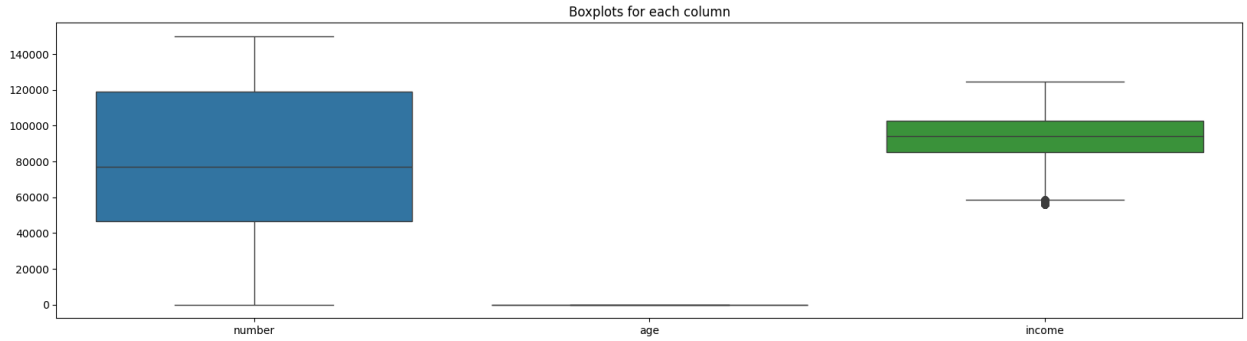


Виден огромный выброс у колонки income, устраняем.

```
# Скорпируем
df_clean = df.copy()
```

```
# Очистим
df_clean = df_clean[df_clean['income'] > (df_clean.describe()['income']['25%'] - df_clean.describe()['income']['std'])]
df_clean = df_clean[df_clean['income'] < (df_clean.describe()['income']['75%'] + df_clean.describe()['income']['std'])]
```

```
plt.figure(figsize=(20, 5))
sb.boxplot(data=df_clean)
plt.title("Boxplots for each column")
plt.show()
```



Получилось более менее нормально, с минимальным выбросом.

3.4. Пропущенные значения

```
df_clean.isna().sum()
```

```
number     0
city       0
gender     0
age        0
income     0
illness    0
dtype: int64
```

Отсутствует пропущенные значения. Давайте искусственно создадим их.

```
# Заменяем часть значений в одном или нескольких признаках на пропуски
# Пусть, давайте, например, заменим 10% значений в столбце 'income' на пропуски
percentage_of_missing_values = 0.1
num_of_values_to_replace = int(len(df_clean) * percentage_of_missing_values)
indices_to_replace = np.random.choice(df_clean.index, num_of_values_to_replace, replace=False)
df_clean.loc[indices_to_replace, 'income'] = np.nan
```

```
# Снова выводим информацию о пропусках
df_clean.isna().sum()
```

```
number         0
city           0
gender         0
age            0
income     12080
illness        0
dtype: int64
```

3.5. Дублирующие значения

```
# Кол-во дублирующие значения
df_clean.duplicated().sum()
```

```
0
```

Отсутствует дублирующие значения

3.6. Удаление неинформативного признака

Существует колонка - number, которая по сути является как индексация датафрейма, поэтому устраняем их.

```
df_clean = df_clean.drop('number', axis=1)
```

```
df_clean.head()
```

	city	gender	age	income	illness
8	Dallas	Male	51	68667.0	No
12	Dallas	Male	46	62749.0	No
26	Dallas	Male	58	57322.0	No
27	Dallas	Male	44	61704.0	No
33	Dallas	Male	27	56645.0	No

3.7. Обработка категориальных признаков

Можно закодировать названия города, но это потребуется для машинного обучения, поэтому колонку illness преобразуем в булевый тип.

```
df_clean['illness'] = df_clean['illness'].replace({'Yes': True, 'No': False})
```

```
/tmp/ipykernel_47979/3484491388.py:1: FutureWarning: Downcasting behavior in `replace` is deprecated and will be removed in a future versio
n. To retain the old behavior, explicitly call `result.infer_objects(copy=False)`. To opt-in to the future behavior, set `pd.set_option('fu
ture.no_silent_downcasting', True)`
  df_clean['illness'] = df_clean['illness'].replace({'Yes': True, 'No': False})
```

```
df_clean['gender'] = df_clean['gender'].replace({'Male': True, 'Female': False})
```

```
/tmp/ipykernel_47979/1457730574.py:1: FutureWarning: Downcasting behavior in `replace` is deprecated and will be removed in a future versio
n. To retain the old behavior, explicitly call `result.infer_objects(copy=False)`. To opt-in to the future behavior, set `pd.set_option('fu
ture.no_silent_downcasting', True)`
  df_clean['gender'] = df_clean['gender'].replace({'Male': True, 'Female': False})
```

3.8. Создание нового признака

3.8.1. Группировка по возрастам

Создадим функцию, которая группирует данных по возрастам и создания новой колонки.

```
df_clean['age'].value_counts().sort_index()
```

```
age
25    1502
26    3186
27    3061
28    3092
29    3099
30    2965
31    3017
32    3022
33    3016
34    2981
35    2925
36    3053
37    3049
38    3057
39    2975
40    3009
41    3101
```

```
def create_age_group(df):
    # Задаем группы возрастов и метки
    age_bins = [25, 35, 45, 55, 65]
    age_labels = ['25-34', '35-44', '45-54', '55-65']

    # Создаем новую колонку 'Age_Group' на основе групп возрастов
    df['age_group'] = pd.cut(df['age'], bins=age_bins, labels=age_labels, right=False)

    return df
```

```
# Создадим
df_clean = create_age_group(df_clean)
```

```
df_clean['age_group'].unique()
```

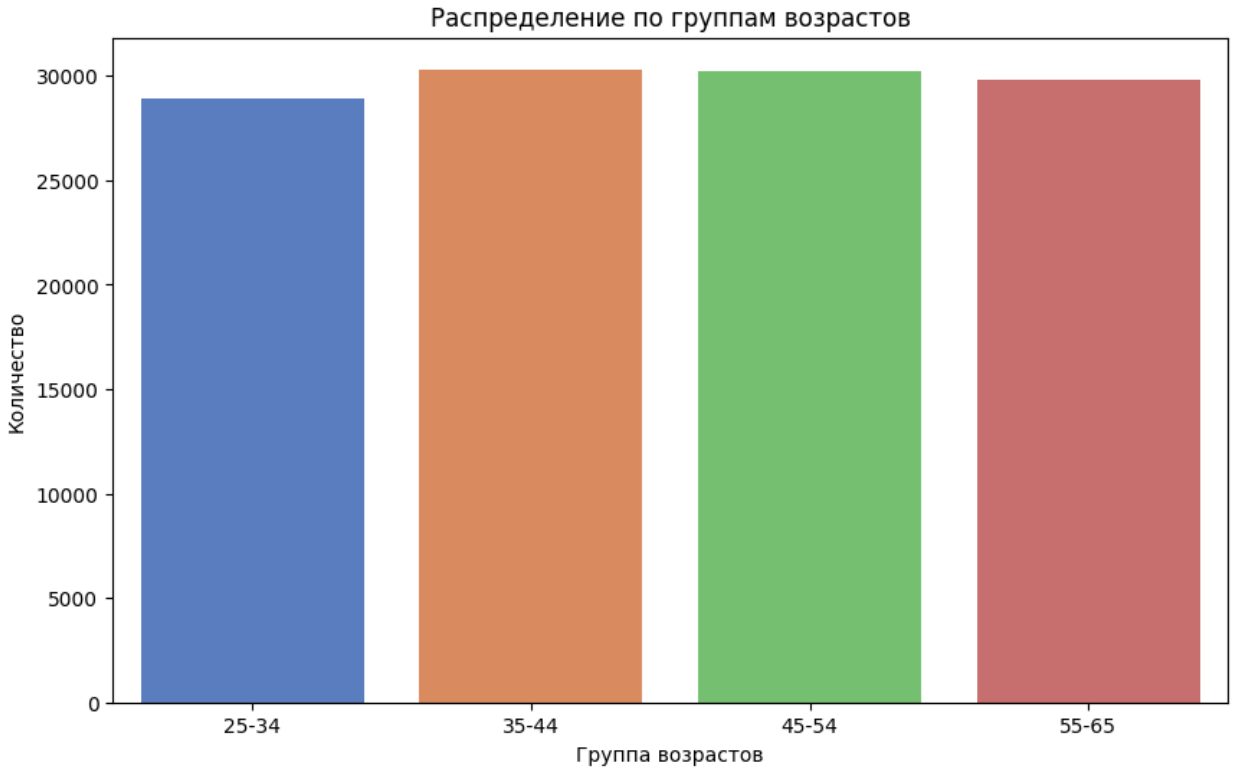
```
['45-54', '55-65', '35-44', '25-34', NaN]
Categories (4, object): ['25-34' < '35-44' < '45-54' < '55-65']
```

```
df_clean = df_clean.dropna(subset=['age_group'])
```

```
df_clean['age_group'].value_counts().sort_index()
```

```
age_group
25-34    28941
35-44    30313
45-54    30239
55-65    29832
Name: count, dtype: int64
```

```
# Создаем график
plt.figure(figsize=(10, 6))

# Строим столбчатую диаграмму для групп возрастов
sb.countplot(x='age_group', data=df_clean, palette='muted')

# Добавляем заголовок и подписи
plt.title('Распределение по группам возрастов')
plt.xlabel('Группа возрастов')
plt.ylabel('Количество')

# Отображаем график
plt.show()
```



Как и видно, что все возрасты группы примерно равномерны одинаковы.

3.8.2. Группировка по уровня дохода

Создадим функцию, которая группирует данных по уровня дохода и по категории "Низкий", "Средний" и "Высокий" на основе заданных порогов.

```
print('Максимальный доход', df_clean['income'].max())
print('Средний доход', df_clean['income'].mean())
print('Медианный доход', df_clean['income'].median())
print('Минимальный доход', df_clean['income'].min())
print('Минимальный доход', df_clean['income'].std())
```

```
Максимальный доход 124731.0
Средний доход 93467.7581646763
Медианный доход 94313.0
Минимальный доход 55880.0
Минимальный доход 13767.512415596104
```

```
def categorize_income(df, low_threshold, high_threshold):
    # low_threshold: Порог для низкого дохода
    # high_threshold: Порог для высокого дохода

    df['income_level'] = pd.cut(df['income'], bins=[float('-inf'), low_threshold, high_threshold, float('inf')],
                                labels=['Низкий', 'Средний', 'Высокий'], include_lowest=True, right=False)
    return df
```

```
high_threshold = round(df_clean['income'].max() - df_clean['income'].std(), 3)
print('Порог для высокого дохода:', high_threshold)
```

```
Порог для высокого дохода: 110969.338
```

```
low_threshold = round(df_clean['income'].min() + df_clean['income'].std(), 3)
print('Порог для низкого дохода:', low_threshold)
```

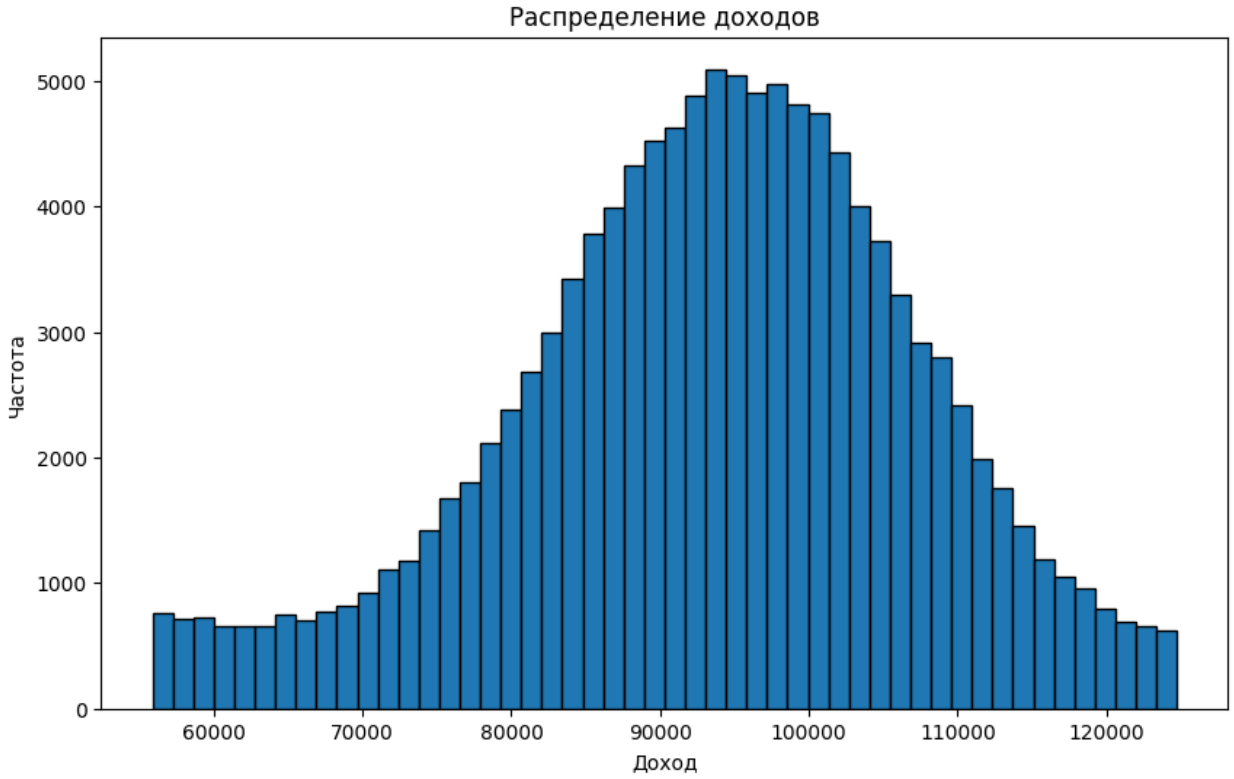
```
Порог для низкого дохода: 69641.662
```

```
df_clean = categorize_income(df_clean, low_threshold=low_threshold, high_threshold=high_threshold)
```

```
df_clean.income_level.value_counts()
```

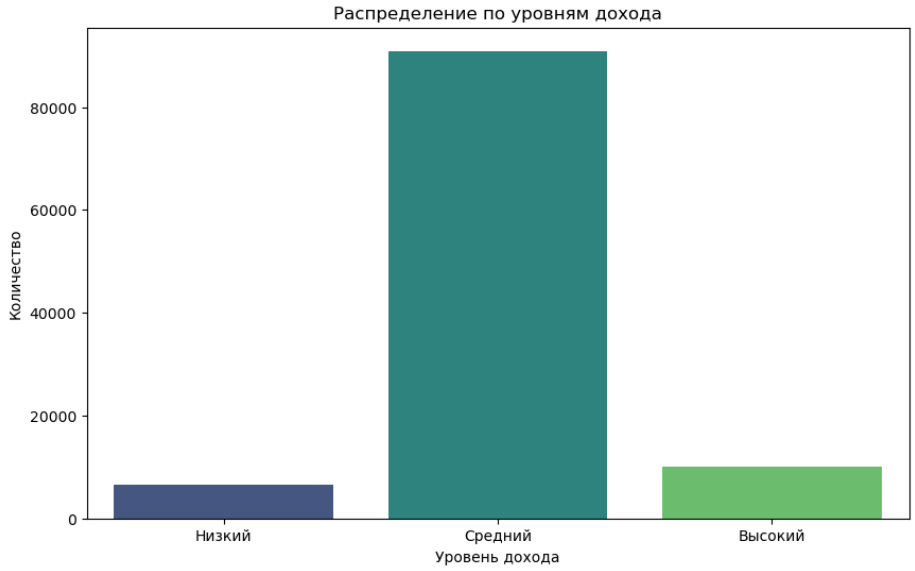
```
Средний    90912
Высокий    10005
Низкий      6472
Name: income_level, dtype: int64
```

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

plt.hist(df_clean['income'], bins=50, edgecolor='black')

plt.title('Распределение доходов')
plt.xlabel('Доход')
plt.ylabel('Частота')

plt.show()
```



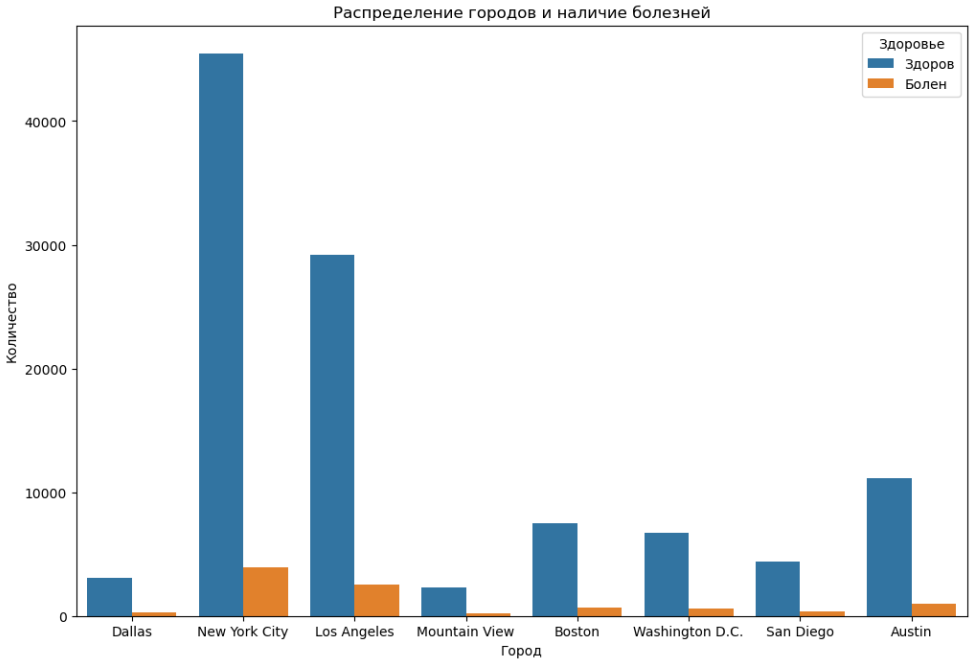
Изобразим еще один график созданного нового признака.

```
# Создаем график
plt.figure(figsize=(10, 6))

sb.countplot(x='income_level', data=df_clean, palette='viridis')

plt.title('Распределение по уровням дохода')
plt.xlabel('Уровень дохода')
plt.ylabel('Количество')

plt.show()
```



3.9. Распределение городов и наличие болезней

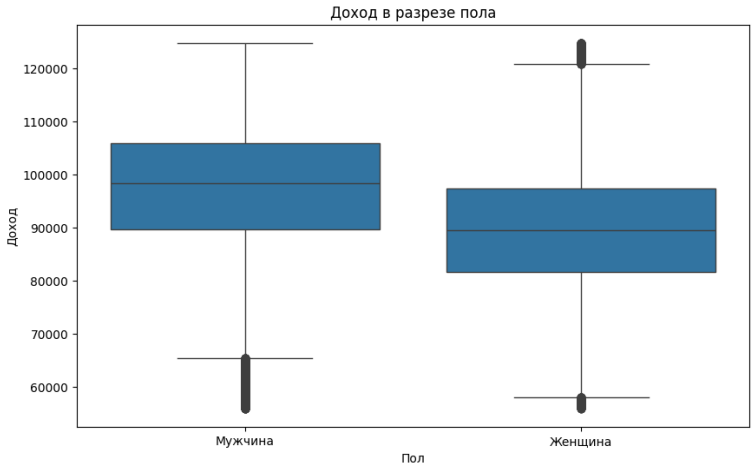
```
plt.figure(figsize=(12, 8))
sb.countplot(x='city', hue='illness', data=df_clean)
plt.title('Распределение городов и наличие болезней')
plt.xlabel('Город')
plt.ylabel('Количество')
# Добавление подписей
plt.legend(title='Здоровье', labels=['Здоров', 'Болен'])
plt.show()
```



Как видим, что в густонаселенных городах мы часто сталкиваемся с риском по шансу схватиться с вирусными заболеваниями.

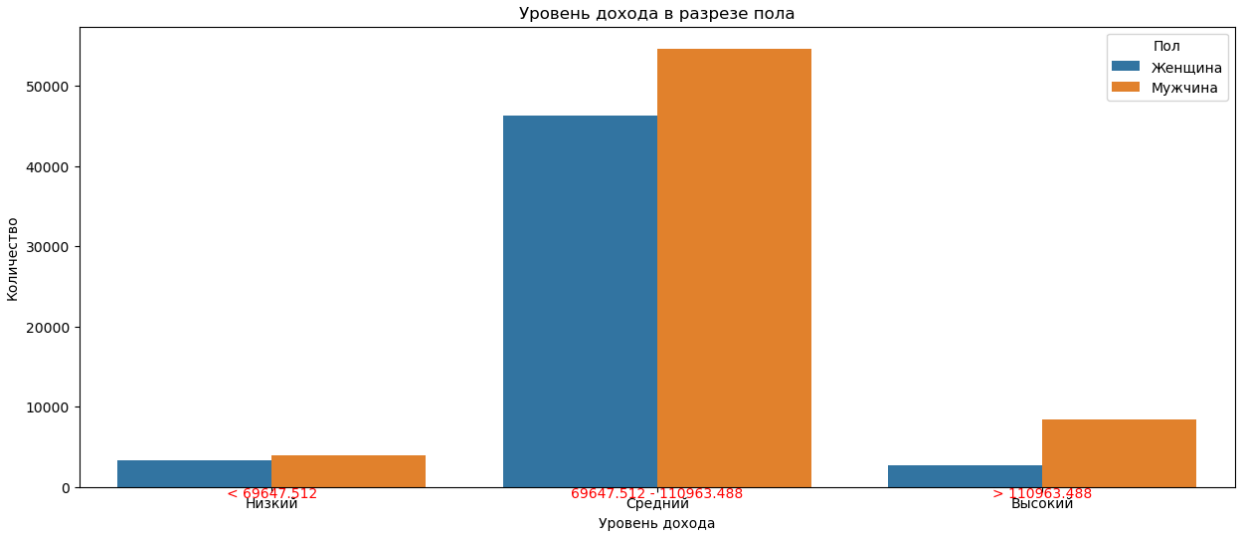
3.10. Доход в разрезе пола

```
plt.figure(figsize=(10, 6))
sb.boxplot(x=df_clean['gender'].replace({True: 'Мужчина', False: 'Женщина'}), y='income', data=df)
plt.title('Доход в разрезе пола')
plt.xlabel('Пол')
plt.ylabel('Доход')
plt.show()
```



Будет проще рассмотреть по группировкам.

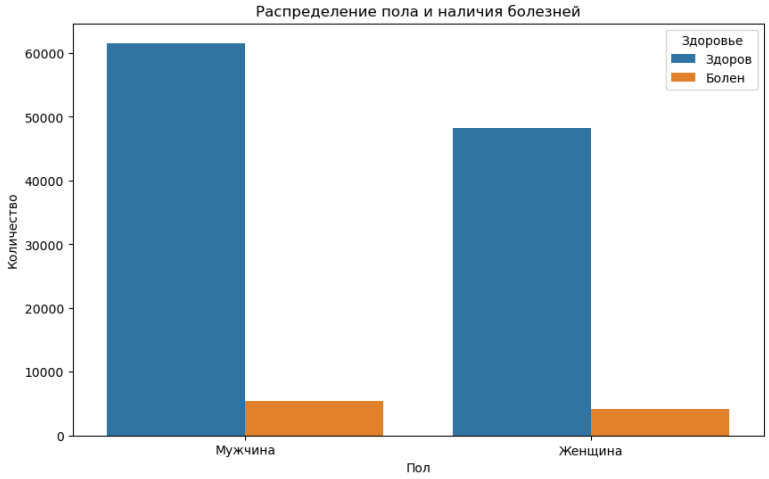
```
plt.figure(figsize=(15, 6))
sb.countplot(x='income_level', hue='gender', data=df_clean, order=['Низкий', 'Средний', 'Высокий'])
plt.title('Уровень дохода в разрезе пола')
plt.xlabel('Уровень дохода')
plt.ylabel('Количество')
plt.legend(title='Пол', labels=['Женщина', 'Мужчина'])

# Добавляем подписи под графиком
plt.text(0.0, 0, f'< {low_threshold}', ha='center', va='top', fontsize=10, color='red')
plt.text(1.0, 0, f'{low_threshold} - {high_threshold}', ha='center', va='top', fontsize=10, color='red')
plt.text(2.0, 0, f'> {high_threshold}', ha='center', va='top', fontsize=10, color='red')

plt.show()
```



Сразу видно, что мужчины преобладает по большей заработной платы нежели представительницы прекрасного пола.

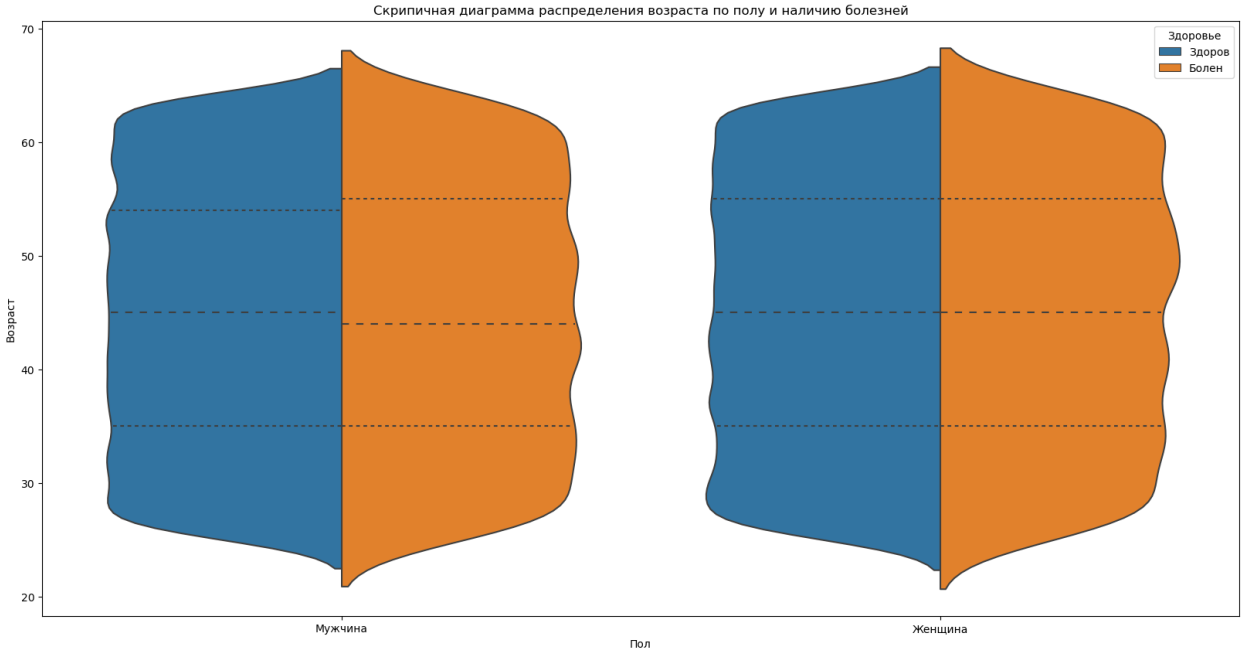
3.11. Распределения пола и наличия болезней

```
# Создание столбчатой диаграммы для распределения пола и наличия болезней
plt.figure(figsize=(10, 6))
sb.countplot(x=df_clean['gender'].replace({True: 'Мужчина', False: 'Женщина'}), hue='illness', data=df_clean)
plt.title('Распределение пола и наличия болезней')
plt.xlabel('Пол')
plt.ylabel('Количество')

# Добавление подписей
plt.legend(title='Здоровье', labels=['Здоров', 'Болен'])
plt.show()
```



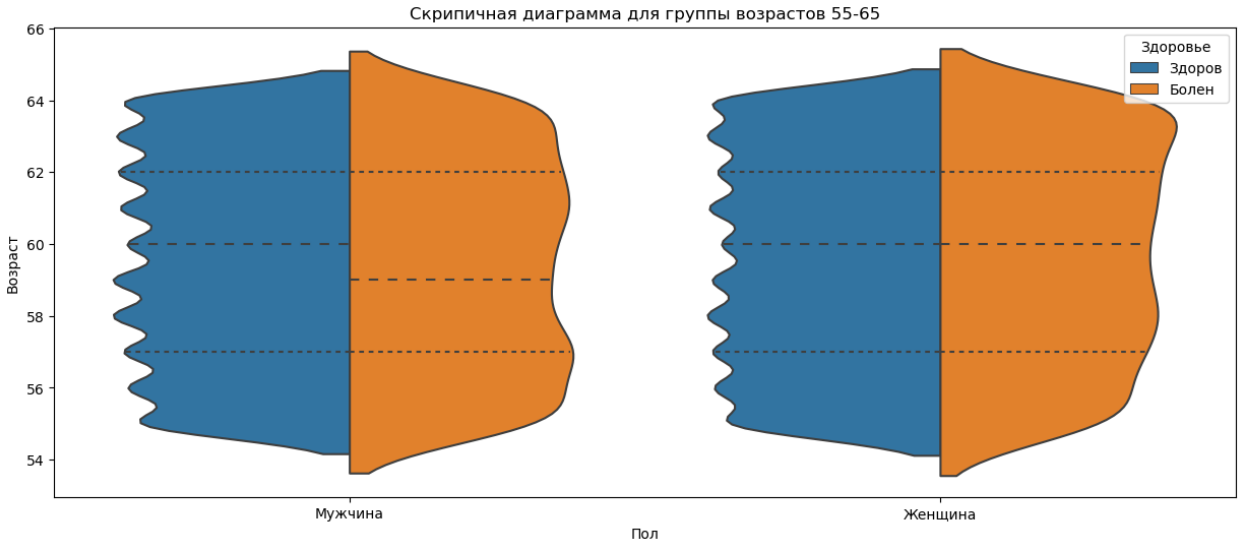
Как и видим, что почти примерны одинаковы по уровня здоровья у мужчин и женщин. Лучше всего рассмотреть на скрипичной диаграммой.

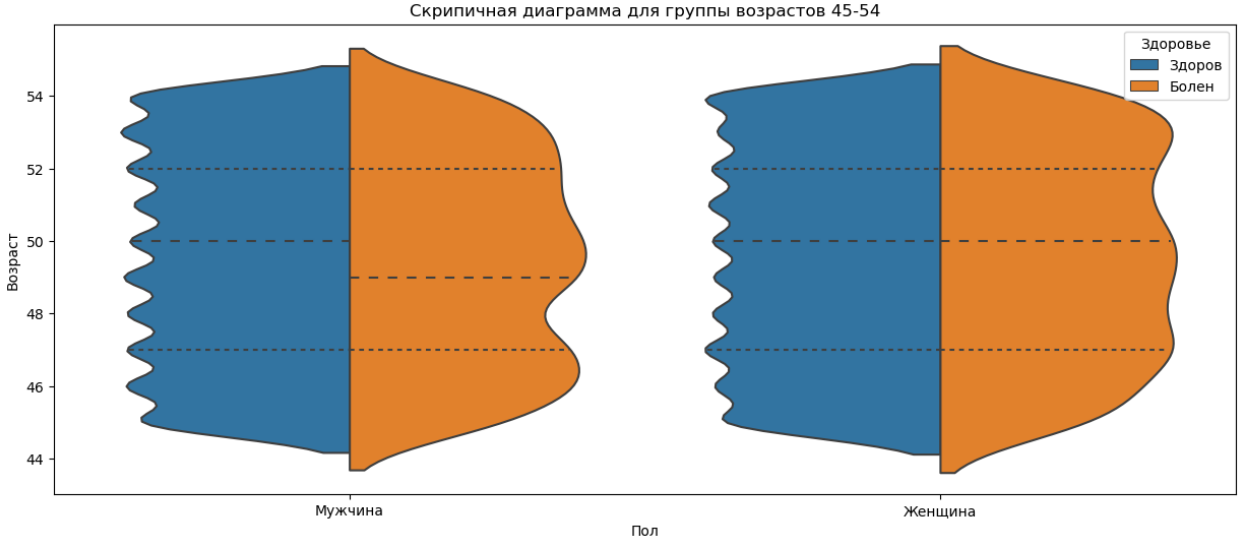
```
plt.figure(figsize=(20, 10))
sb.violinplot(
    x=df_clean['gender'].replace({True: 'Мужчина', False: 'Женщина'}),
    y=df_clean['age'],
    hue=df_clean['illness'].replace({True: 'Болен', False: 'Здоров'}),
    split=True,
    inner='quartile'
)
plt.title('Скрипичная диаграмма распределения возраста по полу и наличию болезней')
plt.xlabel('Пол')
plt.ylabel('Возраст')
# Добавление подписей
plt.legend(title='Здоровье')
plt.show()
```

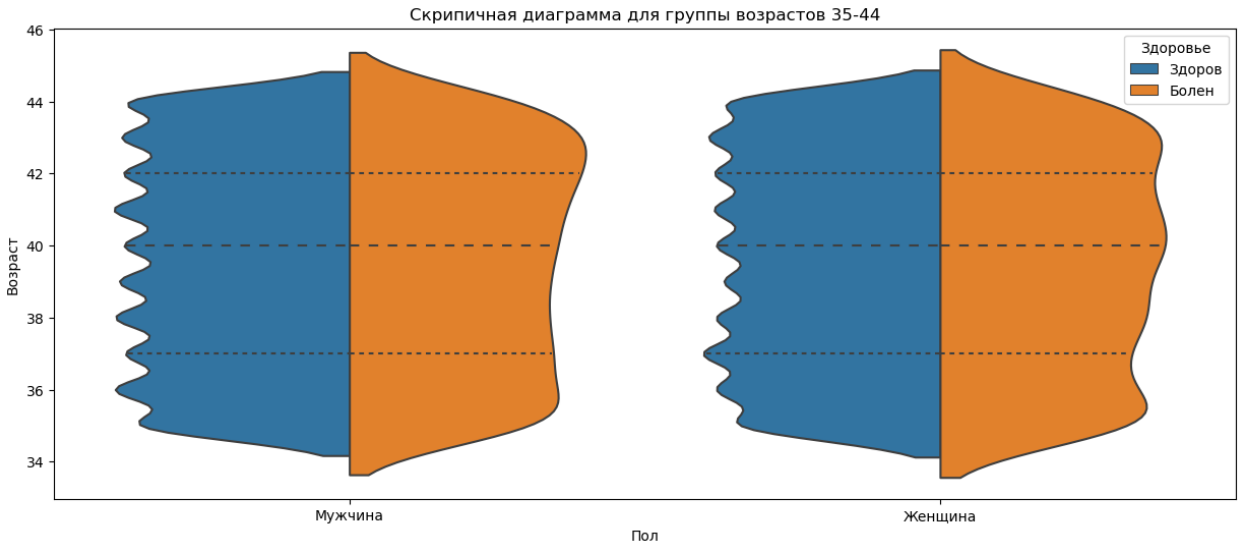


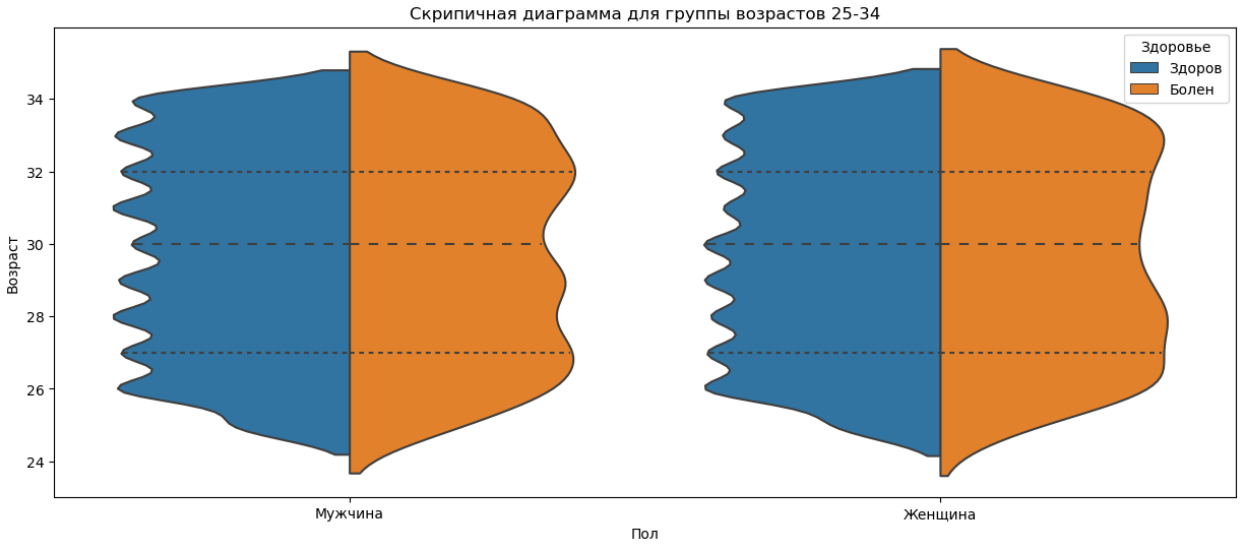
Как и видим, что в молодом возрасте женщины реже заболевает нежели мужчин. А что если рассмотреть по группам возраста?

```
for age_group in df_clean['age_group'].unique().sort_values(ascending=False):
    plt.figure(figsize=(15, 6))
    sb.violinplot(
        x=df_clean['gender'].replace({True: 'Мужчина', False: 'Женщина'}),
        y='age',
        hue=df_clean['illness'].replace({True: 'Болен', False: 'Здоров'}),
        split=True,
        inner='quartile',
        data=df_clean[df_clean['age_group'] == age_group]
    )
    plt.title(f'Скрипичная диаграмма для группы возрастов {age_group}')
    plt.xlabel('Пол')
    plt.ylabel('Возраст')
    plt.legend(title='Здоровье')
    plt.show()
```


Скрипичная диаграмма для группы возрастов 55-65
Здоровье
Здоров
Болен
Возраст
66
64
62
60
58
56
54
Мужчина
Женщина
Пол


Скрипичная диаграмма для группы возрастов 45-54
Здоровье
Здоров
Болен
Возраст
54
52
50
48
46
44
Мужчина
Женщина
Пол


Скрипичная диаграмма для группы возрастов 35-44
Здоровье
Здоров
Болен
Возраст
46
44
42
40
38
36
34
Мужчина
Женщина
Пол



По графику можно примерно поставить оценку по уровня заболеваемости. В преклонных возрастах женщины заболевают чаще мужчин. В остальном примерно одинаковы получилось.