Фонин Максим Алексеевич, ИУ5-25М

Рубежный контроль №1: "Методы обработки данных"

Номер варианта: 8

Номер задачи №1: 8

Номер задачи №2: 28

Задача №8. Для набора данных проведите устранение пропусков для одного (произвольного) числового признака с использованием метода заполнения модой.

Задача №28. Для набора данных для одного (произвольного) числового признака проведите обнаружение и замену (найденными верхними и нижними границами) выбросов на основе межквартильного размаха.

Дополнительные требования по группам:

Для студентов группы **ИУ5-25М**, ИУ5И-25М, ИУ5-25МВ - для произвольной колонки данных построить парные диаграммы (pairplot).

Описание набора данных cleaned_nutrition_dataset - https://www.kaggle.com/datasets/tharunmss/nutritional-breakdown-of-foods/data

Этот набор данных содержит полную разбивку по питательной ценности более 3500 продуктов питания, включая подробную информацию о макроэлементах, микроэлементах и калорийности. Он идеально подходит для использования в науке о данных, аналитике здоровья, планировании питания и проектах машинного обучения, ориентированных на еду, питание и благополучие.

Содержит следующую основную информацию:

- Vitamin C - количество витамина C (аскорбиновой кислоты), присутствующего в продукте питания.
- Vitamin B11 - количество витамина B11 (фолиевой кислоты) в продукте питания.
- Sodium - содержание натрия, как правило, влияет на уровень соли.
- Calcium - количество кальция, необходимого для костей и зубов.
- Carbohydrates - общее количество углеводов, включая крахмалы и сахара.
- food - первоначальное название продукта питания, как оно было записано.
- Iron - содержание железа важно для транспорта крови и кислорода.
- Caloric Value - общая энергия, содержащаяся в продукте питания (в килокалориях).
- Sugars - простые сахара включены в общее количество углеводов.
- Dietary Fiber - неперевариваемые углеводы, способствующие пищеварению.

Задача №8.

In [44]:
```
import pandas as pd
import seaborn as sns
import numpy as np
import scipy.stats as stats
import matplotlib.pyplot as plt
from sklearn.impute import SimpleImputer
from sklearn.impute import MissingIndicator
```

In [45]:
```
# Install dependencies as needed:
# pip install kagglehub[pandas-datasets]
import kagglehub
from kagglehub import KaggleDatasetAdapter

# Set the path to the file you'd like to load
file_path = "cleaned_nutrition_dataset.csv"

# Load the latest version
df = kagglehub.load_dataset(
  KaggleDatasetAdapter.PANDAS,
  "tharunmss/nutritional-breakdown-of-foods",
  file_path,
  # Provide any additional arguments like
  # sql_query or pandas_kwargs. See the
  # documenation for more information:
  # https://github.com/Kaggle/kagglehub/blob/main/README.md#kaggledatasetadapter
```

```
)
df.shape
```

```
<ipython-input-45-e0d2a6dc8e91>:10: DeprecationWarning: load_dataset is deprecate
d and will be removed in future version.
  df = kagglehub.load_dataset(
```

Out[45]: (3454, 13)

In [46]:
```
df.head()
```

Out[46]:

	Vitamin C	Vitamin B11	Sodium	Calcium	Carbohydrates	food	Iron	Caloric Value	Suga
0	0.082	0.086	0.018	2.8	0.073	margarine with yoghurt	0.027	88.0	0
1	0.400	0.005	0.065	10.2	3.700	sunflower seed butter	0.700	99.0	1
2	0.000	0.000	0.000	0.0	0.000	hazelnut oil	0.000	120.0	0
3	0.000	0.000	0.000	0.0	0.000	menhaden fish oil	0.000	1966.0	0
4	0.000	0.000	0.000	0.0	0.000	cod liver fish oil	0.000	123.0	0

In [47]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3454 entries, 0 to 3453
Data columns (total 13 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Vitamin C        3454 non-null   float64
 1   Vitamin B11      3454 non-null   float64
 2   Sodium           3454 non-null   float64
 3   Calcium          3454 non-null   float64
 4   Carbohydrates    3454 non-null   float64
 5   food             3454 non-null   object
 6   Iron             3454 non-null   float64
 7   Caloric Value    3454 non-null   float64
 8   Sugars           3454 non-null   float64
 9   Dietary Fiber    3454 non-null   float64
 10  Fat              3454 non-null   float64
 11  Protein          3454 non-null   float64
 12  food_normalized  3454 non-null   object
dtypes: float64(11), object(2)
memory usage: 350.9+ KB
```

In [48]:
```
df.describe()
```

Out[48]:

	Vitamin C	Vitamin B11	Sodium	Calcium	Carbohydrates	Iron
count	3454.000000	3454.000000	3454.000000	3454.000000	3454.000000	3454.000000
mean	9.501021	11.447140	141.437806	53.953960	18.724570	1.627954
std	70.387542	33.849817	973.168394	103.339195	26.232674	4.384574
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.029000	0.057000	4.647500	1.700000	0.200000
50%	0.300000	0.075000	0.200000	21.940000	9.110000	0.700000
75%	5.382500	4.327500	21.372500	60.600000	26.400000	1.600000
max	3872.000000	550.520000	14174.590000	1283.500000	390.200000	121.200000

In [49]:
```
# Проверка на пропуски
df.isnull().sum()
```

Out[49]:

	0
Vitamin C	0
Vitamin B11	0
Sodium	0
Calcium	0
Carbohydrates	0
food	0
Iron	0
Caloric Value	0
Sugars	0
Dietary Fiber	0
Fat	0
Protein	0
food_normalized	0

dtype: int64

In [50]:
```
# Создадим фиктивные пропуски для выполнения задания
# Пропуски в 'Caloric Value' там, где 'Sugars' и 'Dietary Fiber' низкие (почти н
np.random.seed(42)
subset = df[(df['Sugars'] < 2) & (df['Dietary Fiber'] < 1)]
missing_indices = np.random.choice(subset.index, min(20, len(subset)), replace=F
df.loc[missing_indices, 'Caloric Value'] = np.nan

df['Caloric Value'].isnull().sum()
```

Out[50]: np.int64(20)

In [51]:
```
mode_calories = df['Caloric Value'].mode()[0]
mode_calories
# df['Caloric Value'].fillna(mode_calories, inplace=True)
```

Out[51]: np.float64(2.0)

In [52]:
```
def impute_column(dataset, column, strategy_param, fill_value_param=None):
    """
    Заполнение пропусков в одном признаке
    """
    temp_data = dataset[[column]].values
    size = temp_data.shape[0]

    indicator = MissingIndicator()
    mask_missing_values_only = indicator.fit_transform(temp_data)

    imputer = SimpleImputer(strategy=strategy_param,
                            fill_value=fill_value_param)
    all_data = imputer.fit_transform(temp_data)

    missed_data = temp_data[mask_missing_values_only]
    filled_data = all_data[mask_missing_values_only]

    return all_data.reshape((size,)), filled_data, missed_data
```

In [53]:
```
all_data, filled_data, missed_data = impute_column(df, 'Caloric Value', 'most_fr
all_data
```

Out[53]: array([88., 99., 120., ..., 300., 320., 350.])

In [54]:
```
df['Caloric Value'].mode()
```

Out[54]:

	Caloric Value
0	2.0

dtype: float64

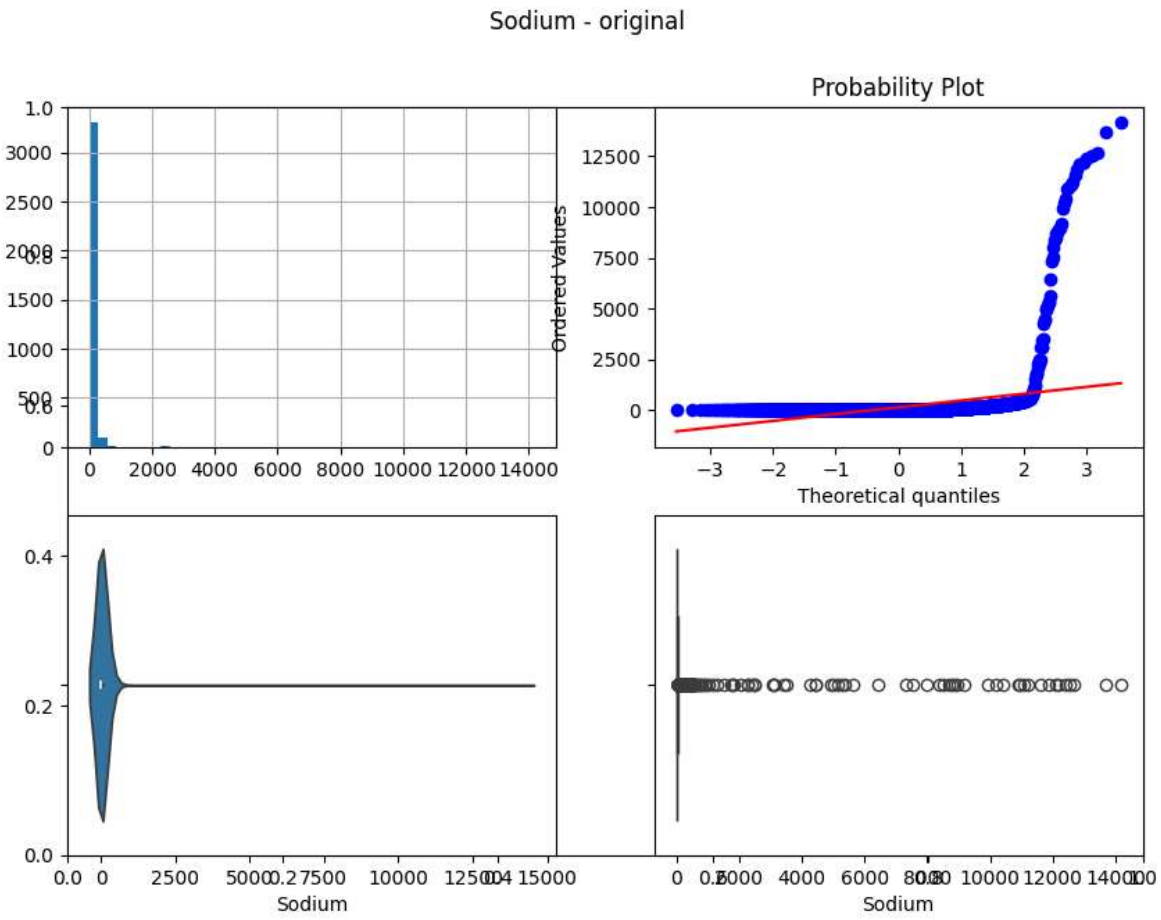
Задача №28.

Обработаем выбросы в 'Sodium'

In [55]:
```
def diagnostic_plots(df, variable, title):
    fig, ax = plt.subplots(figsize=(10,7))
    # гистограмма
    plt.subplot(2, 2, 1)
    df[variable].hist(bins=50)
    ## Q-Q plot
    plt.subplot(2, 2, 2)
    stats.probplot(df[variable], dist="norm", plot=plt)
    # ящик с усами
    plt.subplot(2, 2, 3)
    sns.violinplot(x=df[variable])
    # ящик с усами
```

```
    plt.subplot(2, 2, 4)
    sns.boxplot(x=df[variable])
    fig.suptitle(title)
    plt.show()
```

In [56]:
```
diagnostic_plots(df, 'Sodium', title="Sodium - original")
```



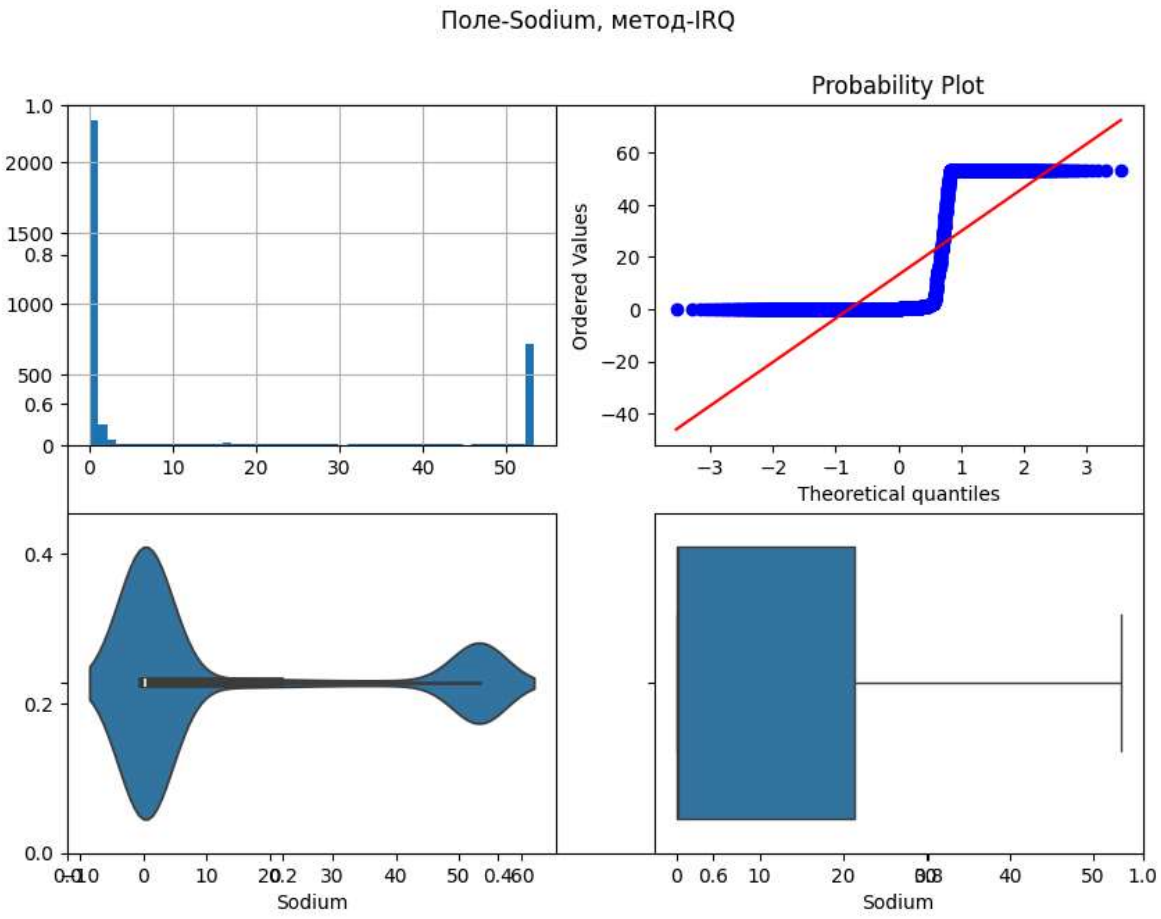
In [41]:
```
sodium = df['Sodium']

# Вычислим границы IQR
Q1 = sodium.quantile(0.25)
Q3 = sodium.quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5*IQR
upper_bound = Q3 + 1.5*IQR
lower_bound, upper_bound
```

Out[41]: (np.float64(-31.91625), np.float64(53.345749999999995))

In [57]:
```
# Флаги для изменения выбросов
df['Sodium'] = np.where(sodium > upper_bound, upper_bound,
                        np.where(sodium < lower_bound, lower_bound, sodium))
title = 'Поле-{}, метод-{}'.format('Sodium', 'IRQ')
```

In [60]:
```
diagnostic_plots(df, 'Sodium', title)
```



Дополнительные требования по группам:

Для столбца 'Sodium' построим парные диаграммы с другими численными столбцами

In [61]:
```
numerical_cols = [
    c for c in df.columns
    if df[c].dtype in ['float64', 'int64']
]
numerical_cols
```

Out[61]:
```
['Vitamin C',
 'Vitamin B11',
 'Sodium',
 'Calcium',
 'Carbohydrates',
 'Iron',
 'Caloric Value',
 'Sugars',
 'Dietary Fiber',
 'Fat',
 'Protein']
```

In [62]:
```
sns.pairplot(
    data=df,
    x_vars=numerical_cols,
    y_vars=['Sodium'],
    plot_kws={'alpha': 0.4, 's': 20},
    height=4,
    aspect=1.2
```

```
)
plt.suptitle('Sodium vs numerical features')
plt.show()
```