

Лабораторная работа №3 "Предобработка данных"

Вариант №12

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

Импорт библиотек

Загрузка данных

Задание 1. Загрузите данные

1. Загрузите данные файла planets.csv о планетах в разных планетных системах галактики (планеты одной системы идут в списке рядом). Поля данных: способ обнаружения, число планет в системе своего «солнца», период обращения вокруг своего «солнца», масса планеты, расстояние от центра планетной системы до нашего Солнца, год обнаружения.

```
planets = pd.read_csv("data/planets.csv", sep=";")
planets.head()
```

	method	number	orbital_period	mass	distance	year
0	Radial Velocity	1	269.300	7.10	77.40	2006
1	Radial Velocity	1	874.774	2.21	56.95	2008
2	Radial Velocity	1	763.000	2.60	19.84	2011
3	Radial Velocity	1	326.030	19.40	110.62	2007
4	Radial Velocity	1	516.220	10.50	119.47	2009

Данные загружены правильно. Для проверки можно посмотреть на первые пять записей.

Описательная статистика

Задание 2. Описание данных

1. Получите описание данных: число записей, число полей.

```
print(f"Число записей: {planets.shape[0]}")
print(f"Число полей: {planets.shape[1]}")
```

```
Число записей: 1035
Число полей: 6
```

Задание 3. Описание полей

1. По каждому полю определите: тип поля: категориальный, порядковый, числовой, текстовый) число имеющихся значений, число уникальных значений, число отсутствующих значений.

Типы полей:

- method - категориальный
- number - порядковый
- orbital_period - числовой
- mass - числовой
- distance - числовой
- year - порядковый

```
print("Число имеющихся значений Non-Null Count: \n")
planets.info()
```

Число имеющихся значений Non-Null Count:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1035 entries, 0 to 1034
Data columns (total 6 columns):
#   Column                Non-Null Count  Dtype
---  -
0   method                1035 non-null  object
1   number                1035 non-null  int64
2   orbital_period        992 non-null   float64
3   mass                  513 non-null   float64
4   distance               808 non-null   float64
5   year                  1035 non-null  int64
dtypes: float64(3), int64(2), object(1)
memory usage: 48.6+ KB
```

```
print("Число уникальных значений: \n")
planets.nunique()
```

Число уникальных значений:

```
method      10
number       7
orbital_period  988
mass        381
distance    552
year        23
dtype: int64
```

```
print("Число отсутствующих значений: \n")
planets.isnull().sum()
```

Число отсутствующих значений:

```
method      0
number      0
orbital_period  43
mass        522
distance     227
year         0
dtype: int64
```

Можно заметить, что в характеристиках планет много пропусков

Задание 4. Мин, макс, медиана, среднеквадратическое отклонение

1. По каждому числовому полю определите: мин, макс, среднее, медиану, среднеквадратическое отклонение.

```
planets.loc[:,["orbital_period", "mass", "distance"]].describe()
```

	orbital_period	mass	distance
count	992.000000	513.000000	808.000000
mean	2002.917596	2.638161	264.069282
std	26014.728304	3.818617	733.116493
min	0.090706	0.003600	1.350000
25%	5.442540	0.229000	32.560000
50%	39.979500	1.260000	55.250000
75%	526.005000	3.040000	178.500000
max	730000.000000	25.000000	8500.000000

Можно заметить, что среди этих признаков самым широким диапазоном значений (и наибольшей дисперсией) обладает признак период обращения. Далее за ним идёт расстояние до планеты, далее масса.

Задание 5. Статистики по категориям

1. По каждой категории: число планет и средние значения остальных признаков.

Здесь, и в дальнейших заданиях, в качестве категорий можно было бы использовать не только метод обнаружения, а что-то более интересное. Например выделить категории "планета сильно удаленна от земли" "слабо удаленна" или выделить категории на основе массы: "карлик", "земля", "суперземля", "мегаземля", "гигант"...

Это может помочь выделить некоторые закономерности

```
planets.groupby('method').agg(
    count=('method', 'size'), # количество планет
    orbital_period=('orbital_period', 'mean'), # средний период
    обращения
    mass=('mass', 'mean'), # средняя масса
```

```
distance=('distance', 'mean') # среднее расстройство
).reset_index()
```

	method	count	orbital_period	mass
distance				
0	Astrometry	2	631.180000	NaN
17.875000				
1	Eclipse Timing Variations	9	4751.644444	5.125000
315.360000				
2	Imaging	38	118247.737500	NaN
67.715937				
3	Microlensing	23	3153.571429	NaN
4144.000000				
4	Orbital Brightness Modulation	3	0.709307	NaN
1180.000000				
5	Pulsar Timing	5	7343.021201	NaN
1200.000000				
6	Pulsation Timing Variations	1	1170.000000	NaN
NaN				
7	Radial Velocity	553	823.354680	2.630699
51.600208				
8	Transit	397	21.102073	1.470000
599.298080				
9	Transit Timing Variations	4	79.783500	NaN
1104.333333				

Для выполнения задания используется метод groupby по атрибуту "method". Метод groupby позволяет сгруппировать значения с агрегированием. В данном случае агрегирование - подсчёт суммы значений и подсчёт среднего значения.

Задание 6. Пропуски данных

1. Определите число записей, имеющих пропуски данных и общее число пропусков.

```
print(f"Число записей с пропусками: {planets.isnull().any(axis=1).sum()}")
print(f"Общее число пропусков: {planets.isnull().sum().sum()}")
```

Число записей с пропусками: 537
Общее число пропусков: 792

Задание 7. Ещё пропуски

1. Для каждого числа возможных пропусков в отдельной записи (от 0 до укажите, в скольких записях столько пропусков.

```
_ = planets.isnull().sum(axis=1).value_counts().reset_index()
_.columns = ['Количество пропусков в записи', 'Число таких записей']
_
```

	Количество пропусков в записи	Число таких записей
0	0	498
1	1	293
2	2	233
3	3	11

Здесь мы определяем отсутствующие значения, далее получаем сумму по строкам, далее группируем по количеству пропусков в записи и агрегируем число таких записей

Задание 8. Среднее количество пропусков для планет по годам

1. Для каждого года обнаружения планеты укажите среднее число пропусков данных для одной планеты.

```
_ = planets.loc[:, "year"].reset_index() # Копия датасета с колонкой
год
_["nulls"] = planets.isnull().sum(axis=1) # Считаем пропуски по каждой
планете
_.groupby("year").agg(
    null_mean=('nulls', 'mean')
)
```

	null_mean
year	
1989	0.000000
1992	2.000000
1994	2.000000
1995	0.000000
1996	0.333333
1997	0.000000
1998	0.000000
1999	0.066667
2000	0.125000
2001	0.083333
2002	0.062500
2003	0.160000
2004	0.692308
2005	0.205128
2006	0.548387
2007	0.566038
2008	0.621622
2009	0.418367
2010	0.745098
2011	0.675676
2012	1.235714
2013	1.237288
2014	1.807692

Здесь вычисляем количество пропусков данных по каждой планете, затем агрегируем по колонке год и вычисляем среднее по пропускам

Видно, что есть года, где или ровно 2 пропуска в каждой планете, или ровно 0 пропусков. Видимо, там было открыто немного планет. Также отсутствуют некоторые года - в отсутствующих годах не было планетарных открытий.

Задание 9. Среднее количество пропусков для планет по категориям

1. Для каждой категории укажите среднее число пропусков данных для одной планеты.

```
_ = planets.loc[:, "method"].reset_index() # Копия датасета с колонкой
метод обнаружения (категория)
_["nulls"] = planets.isnull().sum(axis=1) # Считаем пропуски по каждой
планете
_.groupby("method").agg(
    null_mean=('nulls', 'mean')
)
```

	null_mean
method	
Astrometry	1.000000
Eclipse Timing Variations	1.333333
Imaging	1.842105
Microlensing	2.260870
Orbital Brightness Modulation	1.333333
Pulsar Timing	1.800000
Pulsation Timing Variations	2.000000
Radial Velocity	0.119349
Transit	1.433249
Transit Timing Variations	1.500000

Очевидно, что разные методы обнаружения возволяют получать разные характеристики с разной вероятностью

Очистка датасета

Задание 10. Удаляем записи где больше 2 пропусков

1. Удалите записи с числом пропусков более 2-х.

```
planets.loc[planets.isnull().sum(axis=1) > 2, :]
```

	method	number	orbital_period	mass	distance
year					
33	Imaging	1	NaN	NaN	NaN
2008					
90	Imaging	1	NaN	NaN	NaN
2006					
749	Transit Timing Variations	3	NaN	NaN	NaN
2014					
902	Microlensing	1	NaN	NaN	NaN

31	Imaging	1	NaN	NaN	140.00	2004
34	Imaging	1	NaN	NaN	145.00	2013
35	Imaging	1	NaN	NaN	139.00	2004

Вот записи с пропусками

```
# находим средние по всему набору
mean_orbital_period = filtered_planets.dropna(
    subset=['orbital_period']
)['orbital_period'].mean()
mean_mass = filtered_planets.dropna(
    subset=['mass']
)['mass'].mean()
mean_distance = filtered_planets.dropna(
    subset=['distance']
)['distance'].mean()

print(f"Среднее по периоду: {mean_orbital_period:.02f}")
print(f"Среднее по массе: {mean_mass:.02f}")
print(f"Среднее по расстоянию: {mean_distance:.02f}")

Среднее по периоду: 2002.92
Среднее по массе: 2.64
Среднее по расстоянию: 264.07
```

Заменяем значения

```
non_null_planets = filtered_planets.copy()
non_null_planets["orbital_period"] =
non_null_planets["orbital_period"].fillna(mean_orbital_period)
non_null_planets["mass"] = non_null_planets["mass"].fillna(mean_mass)
non_null_planets["distance"] =
non_null_planets["distance"].fillna(mean_distance)
non_null_planets.loc[29:36]
```

	method	number	orbital_period	mass
distance \				
29	Imaging	1	2002.917596	2.638161
45.520000				
30	Imaging	1	2002.917596	2.638161
165.000000				
31	Imaging	1	2002.917596	2.638161
140.000000				
32	Eclipse Timing Variations	1	10220.000000	6.050000
264.069282				
34	Imaging	1	2002.917596	2.638161
145.000000				
35	Imaging	1	2002.917596	2.638161
139.000000				

36		Imaging	1	2002.917596	2.638161
18.390000					
	year				
29	2005				
30	2007				
31	2004				
32	2009				
34	2013				
35	2004				
36	2006				

Видим, что у строк 29, 30, 31, 34, 35 значения признаков orbital_period и mass заменены на среднее

Задание 12. Добавляем код планеты

1. Добавьте поле «код планеты», куда внесите составной код, типа B74, построенный по правилу: в качестве числа уникальный номер планетной системы, где обнаружена планета (признак принадлежности нескольких планет к одной системе: расположение в списке рядом, одинаковое число планет в системе, одинаковое расстояние до Солнца), а в качестве буквы – номер планеты от центра системы (1-A, 2-B, 3-C, 4-D, ...), который определяется по возрастанию периода обращения.

```
# сначала отделим планетные системы
non_null_planets["planet_systems"] = [-1]*non_null_planets.shape[0]
last_number = -1
last_distance = -1
last_planet_systems = -1
for i, row in non_null_planets.iterrows():
    if row.number != 1 and \
        row.number == last_number and \
        round(row.distance) == round(last_distance):
        # Одна планетная система
        non_null_planets.loc[i, "planet_systems"] =
last_planet_systems
    else:
        # Новая планетная система
        non_null_planets.loc[i, "planet_systems"] =
last_planet_systems + 1

    last_number = non_null_planets.loc[i, "number"]
    last_distance = non_null_planets.loc[i, "distance"]
    last_planet_systems = non_null_planets.loc[i, "planet_systems"]

non_null_planets.loc[non_null_planets["planet_systems"] == 102,]

method number orbital_period mass distance year \
130 Radial Velocity 6 7.2004 0.018 6.8 2011
```

131	Radial Velocity	6	28.1400	0.012	6.8	2011
132	Radial Velocity	6	91.6100	0.016	6.8	2013
133	Radial Velocity	6	62.2400	0.008	6.8	2013
134	Radial Velocity	6	39.0260	0.008	6.8	2013
135	Radial Velocity	6	256.2000	0.014	6.8	2013

	planet_systems
130	102
131	102
132	102
133	102
134	102
135	102

Здесь мы в цикле бежим по строкам сверху вниз, при этом, если:

1. планета в своей системе не одинока
2. и количество планет в её системе равно количеству планет в системе предыдущей планеты
3. и расстояние до неё примерно равно расстоянию до предыдущей планеты
То мы считаем что это одна планетарная система. Каждой планете сначала присваивается номер планетарной системы.

Иначе считаем что это другая планетарная система.

```
# Сортируем
sorted_planets = non_null_planets.sort_values(by=["planet_systems",
"orbital_period"], ascending=[True, True])
sorted_planets[sorted_planets["planet_systems"] ==
102].reset_index(inplace=True)

# Далее присваиваем код
sorted_planets["code"] = ["X0"]*sorted_planets.shape[0]
last_planet_systems = -1
last_code = 65
for i, row in sorted_planets.iterrows():
    if row.planet_systems == last_planet_systems:
        # Одна планетная система, повышаем букву алфавита
        sorted_planets.loc[i, "code"] = f"{chr(last_code+1)}"
    {row.planet_systems}"
        last_code += 1
    else:
        # Новая планетная система, начинаем с буквы A
        last_code = 65
        sorted_planets.loc[i, "code"] = f"{chr(last_code)}"
    {row.planet_systems}"
        last_planet_systems = row.planet_systems
```

```
df = sorted_planets.copy()
# пример кодов
df.loc[sorted_planets["planet_systems"] == 102,]
```

	method	number	orbital_period	mass	distance	year	\
130	Radial Velocity	6	7.2004	0.018	6.8	2011	
131	Radial Velocity	6	28.1400	0.012	6.8	2011	
134	Radial Velocity	6	39.0260	0.008	6.8	2013	
133	Radial Velocity	6	62.2400	0.008	6.8	2013	
132	Radial Velocity	6	91.6100	0.016	6.8	2013	
135	Radial Velocity	6	256.2000	0.014	6.8	2013	

	planet_systems	code
130	102	A102
131	102	B102
134	102	C102
133	102	D102
132	102	E102
135	102	F102

Здесь мы имея номер планетарной системы присваиваем планетам (предварительно отсортировав планеты по планетарной системе, периоду) код планеты, используя таблицу символов ascii.

Визуализация данных

На мой взгляд порядок заданий в работе неверный. Т.к. для верной обработки пропусков нужно сначала проанализировать распределения признаков. Распределения ниже искажены тем, что мы обработали пропуски.

Задание 13. Строим гистограммы

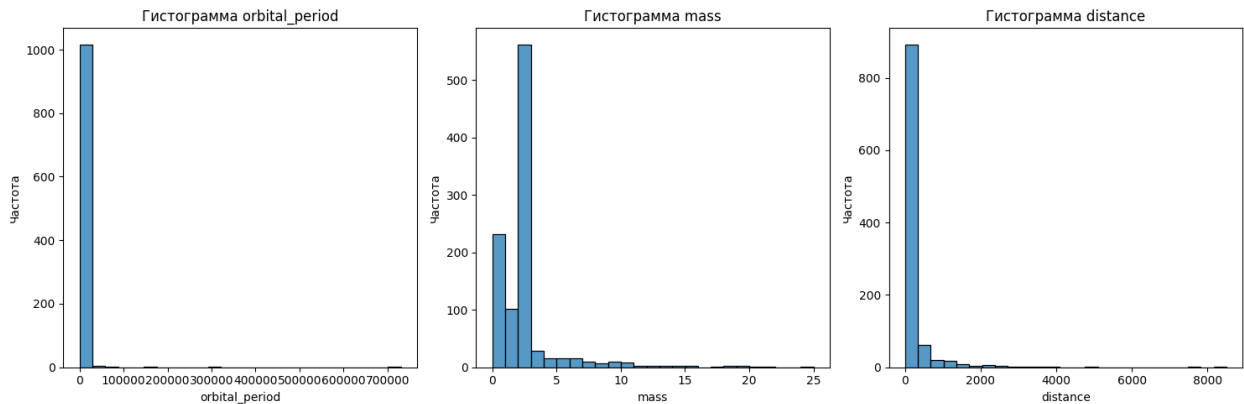
- Для каждого нового набора постройте диаграммы четырех типов:
 - гистограмма каждого числового признака по всему датасету и по отдельным категориям,
 - для каждого числового признака набор ящиков с усами по всем категориям,
 - тепловую карту корреляций для числовых признаков (номер планеты в системе, число планет в системе, период обращения, масса, расстояние, год обнаружения) для всего датасета, для самой массовой и для самой редкой категории.
 - парные диаграммы рассеяния для всех пар числовых признаков, имеющих более 50 уникальных значений, по всему датасету.

```
numeric_features = ["orbital_period", "mass", "distance"]

plt.figure(figsize=(15, 5))
for i, feature in enumerate(numeric_features, 1):
    plt.subplot(1, 3, i)
    sns.histplot(df[feature], bins=25, kde=False)
    plt.title(f'Гистограмма {feature}')
```

```
plt.xlabel(feature)
plt.ylabel('Частота')

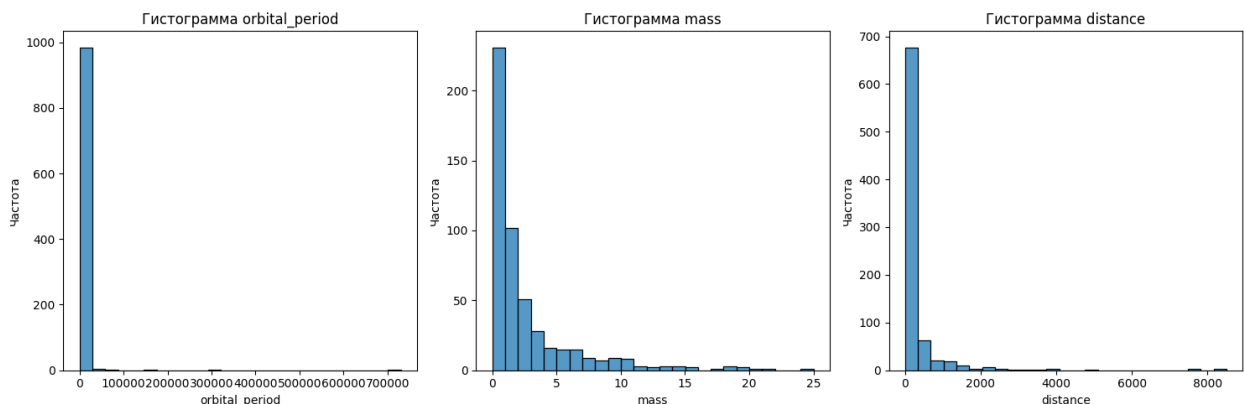
plt.tight_layout()
plt.show()
```



Видно, что некоторые значения выделяются, видимо, из-за замены пропусков. Давайте посмотрим на гистограммы датасета до замены

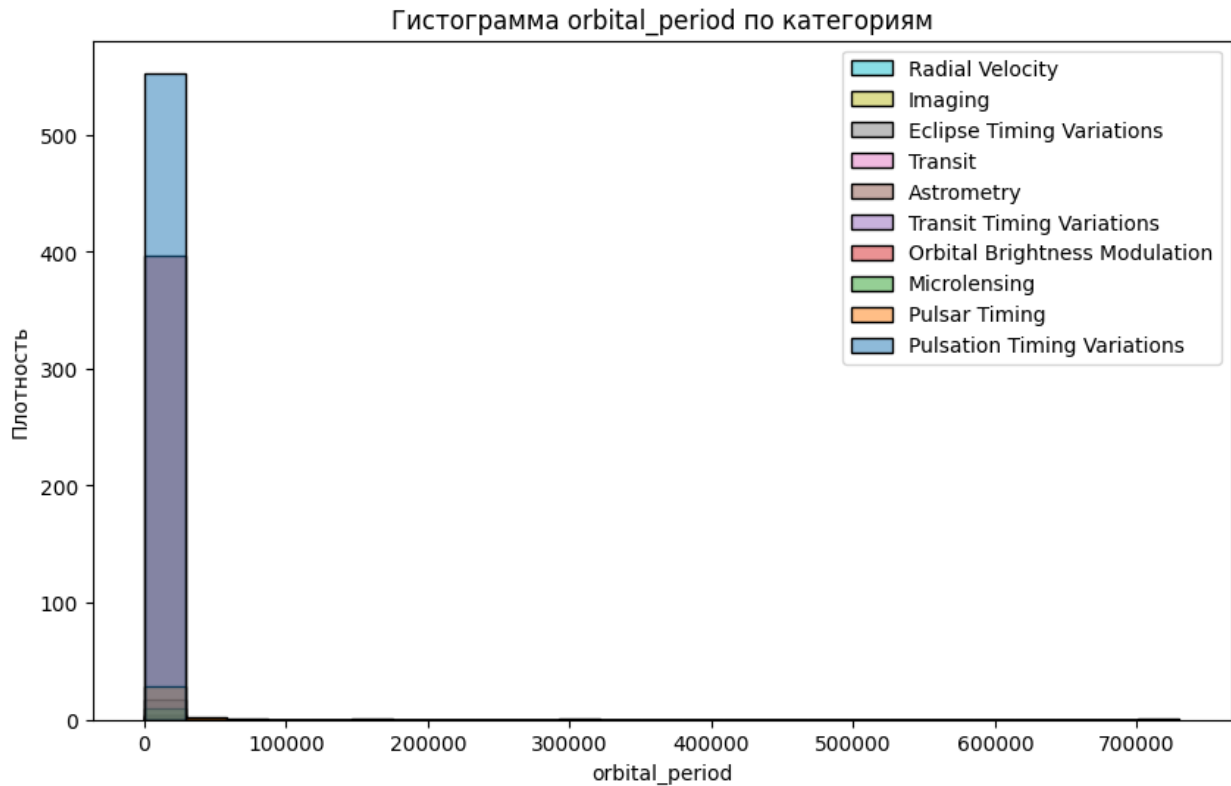
```
plt.figure(figsize=(15, 5))
for i, feature in enumerate(numeric_features, 1):
    plt.subplot(1, 3, i)
    sns.histplot(planets[feature], bins=25, kde=False)
    plt.title(f'Гистограмма {feature}')
    plt.xlabel(feature)
    plt.ylabel('Частота')

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



Судя по гистограмме mass можно предположить, что распределение признака mass похоже на логнормальное. Во всех трёх признаках есть явно доминирующий первый диапазон.

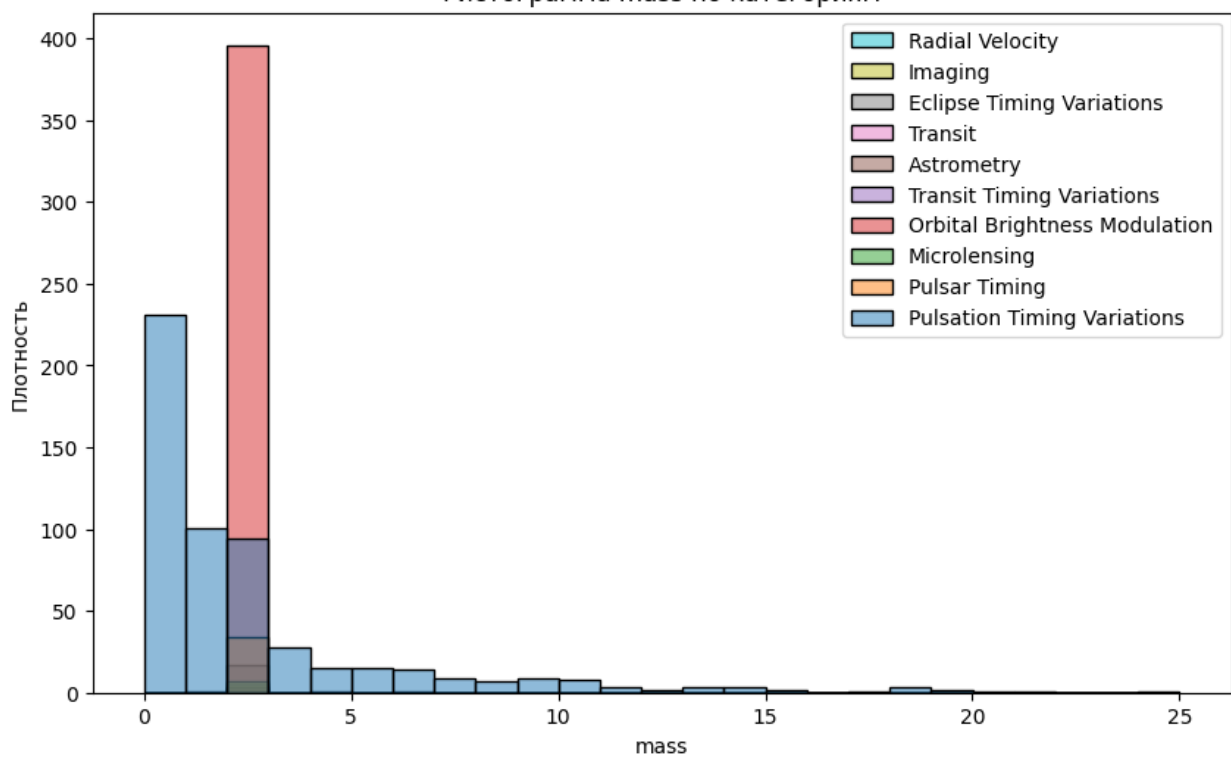
```
# Строим гистограммы по категориям
for feature in numeric_features:
    plt.figure(figsize=(10, 6))
    sns.histplot(data=df, x=feature, hue='method', bins=25)
    plt.title(f'Гистограмма {feature} по категориям')
    plt.xlabel(feature)
    plt.ylabel('Плотность')
    plt.legend(df["method"].unique())
    plt.show()
```



Плотность

mass

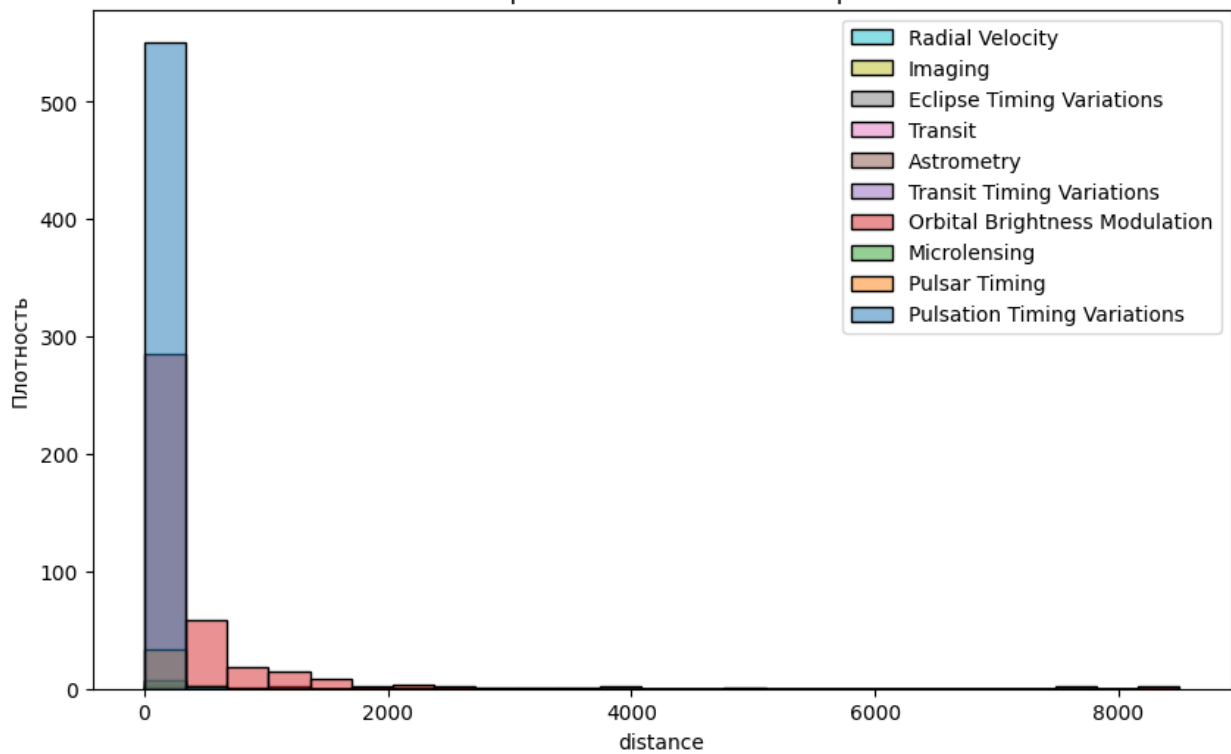
- Radial Velocity
- Imaging
- Eclipse Timing Variations
- Transit
- Astrometry
- Transit Timing Variations
- Orbital Brightness Modulation
- Microlensing
- Pulsar Timing
- Pulsation Timing Variations



Плотность

distance

- Radial Velocity
- Imaging
- Eclipse Timing Variations
- Transit
- Astrometry
- Transit Timing Variations
- Orbital Brightness Modulation
- Microlensing
- Pulsar Timing
- Pulsation Timing Variations

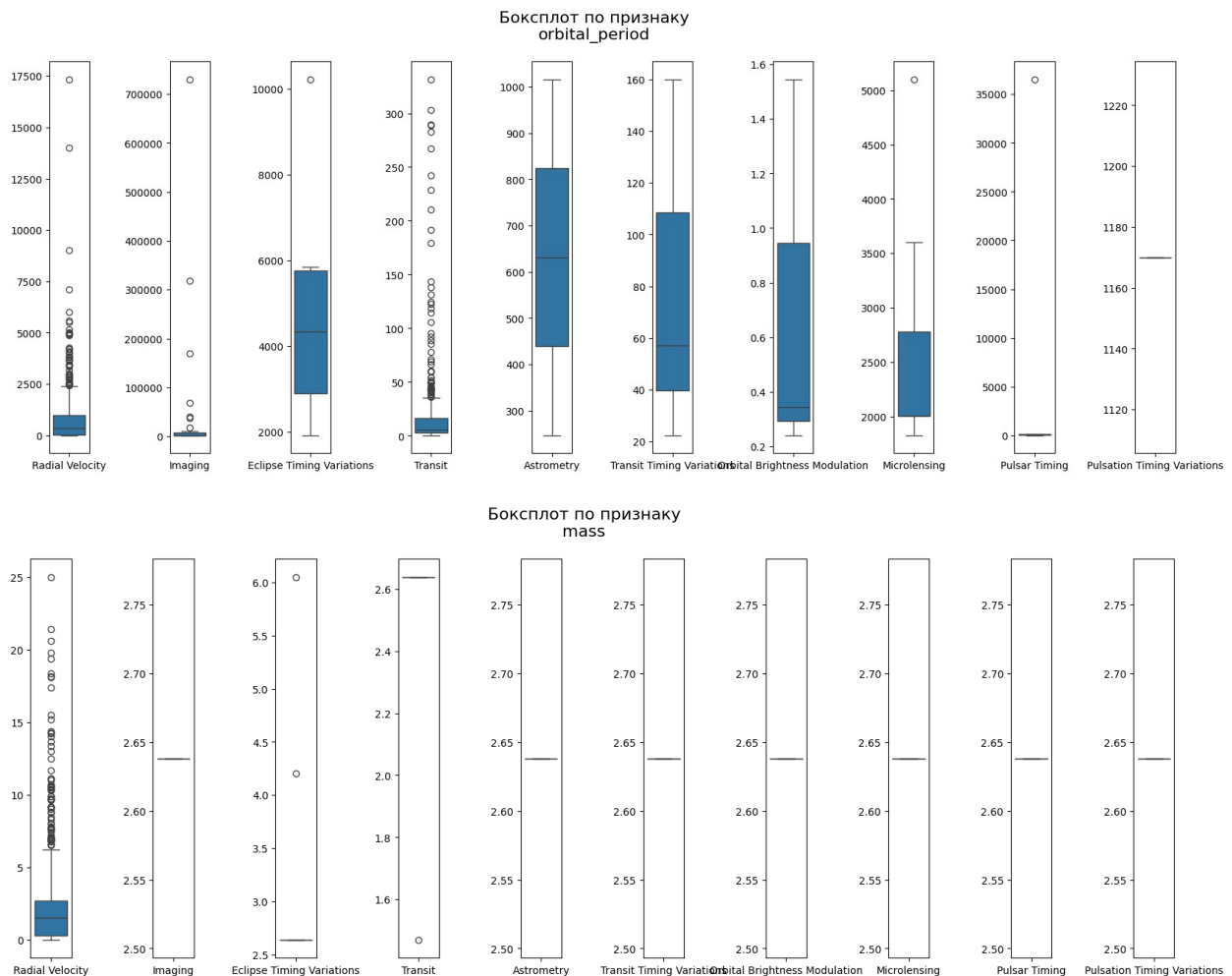


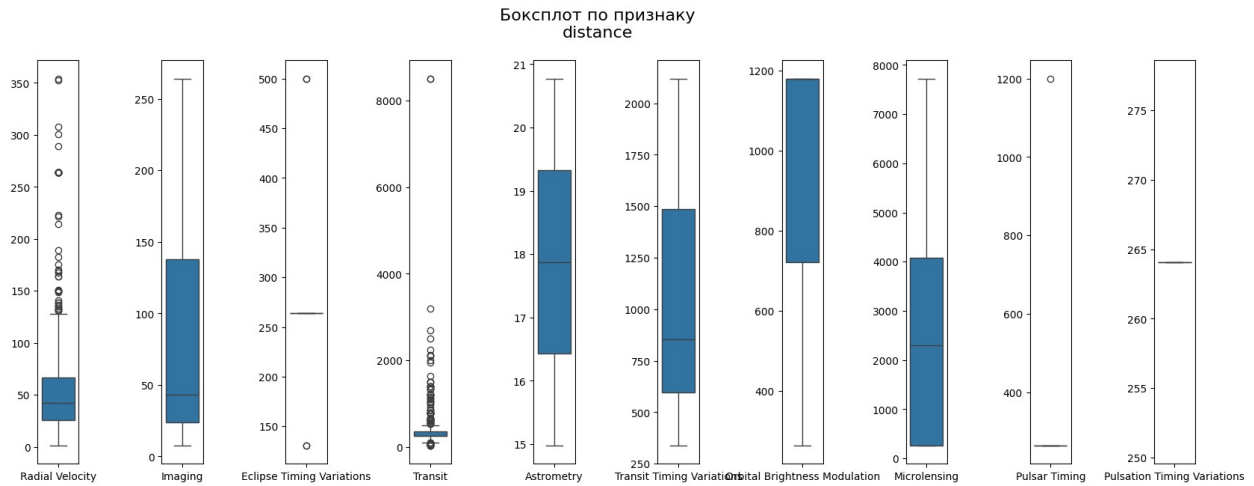
Боксплоты

Строим боксплоты по категориям

```
categories = df.method.unique()
for feature in numeric_features:
    fig, axs = plt.subplots(1, len(categories), figsize=(20, 7))
    for i, category in enumerate(categories):
        sns.boxplot(x='method', y=feature, data=df[df.method ==
category], ax=axs[i])
        axs[i].set_xlabel('')
        axs[i].set_ylabel('')

plt.subplots_adjust(wspace=2)
fig.suptitle(f'Боксплот по признаку\n{feature}', fontsize=16)
plt.show()
```



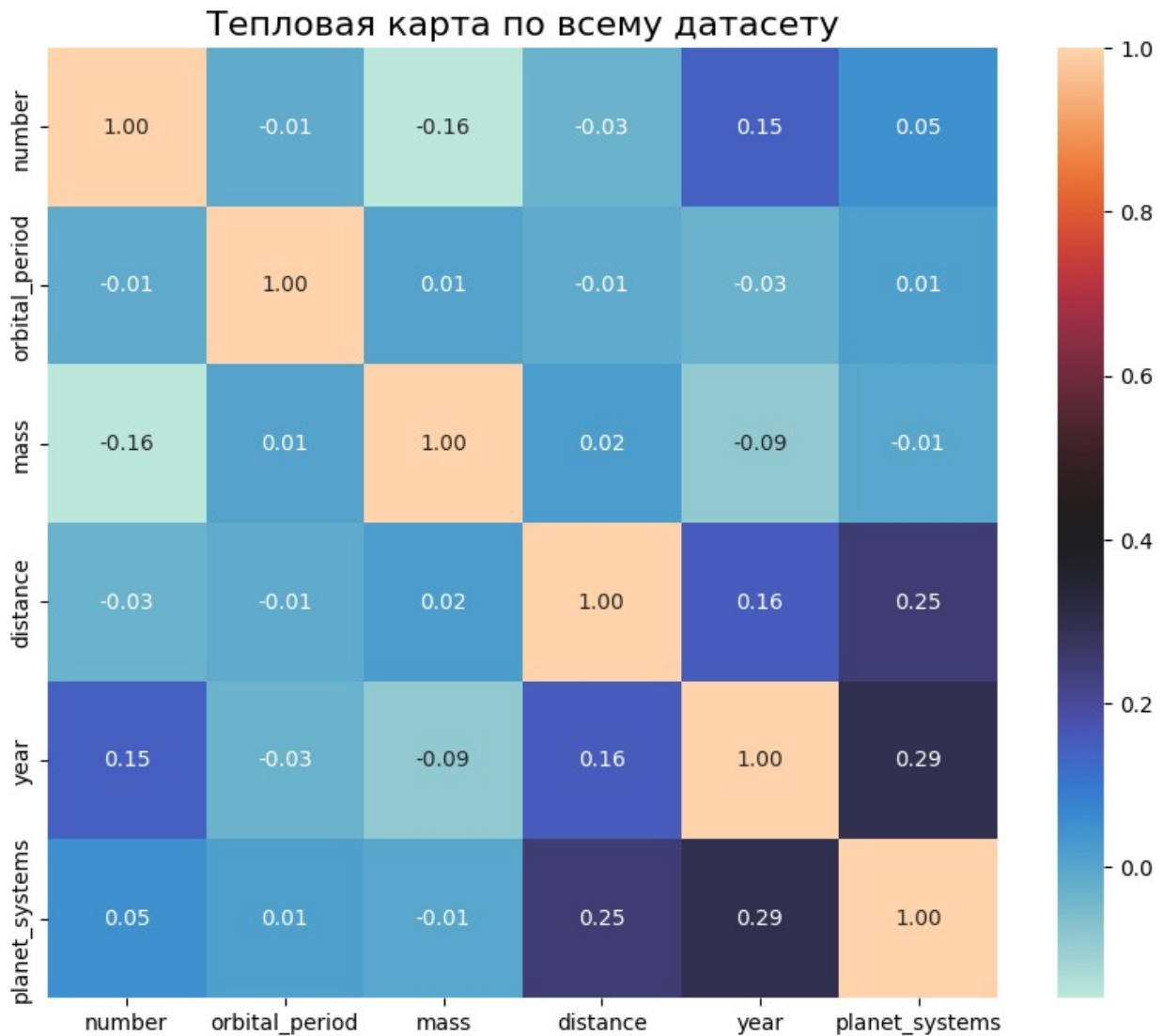


Видно, что есть малочисленные категории

Тепловая карта корреляции

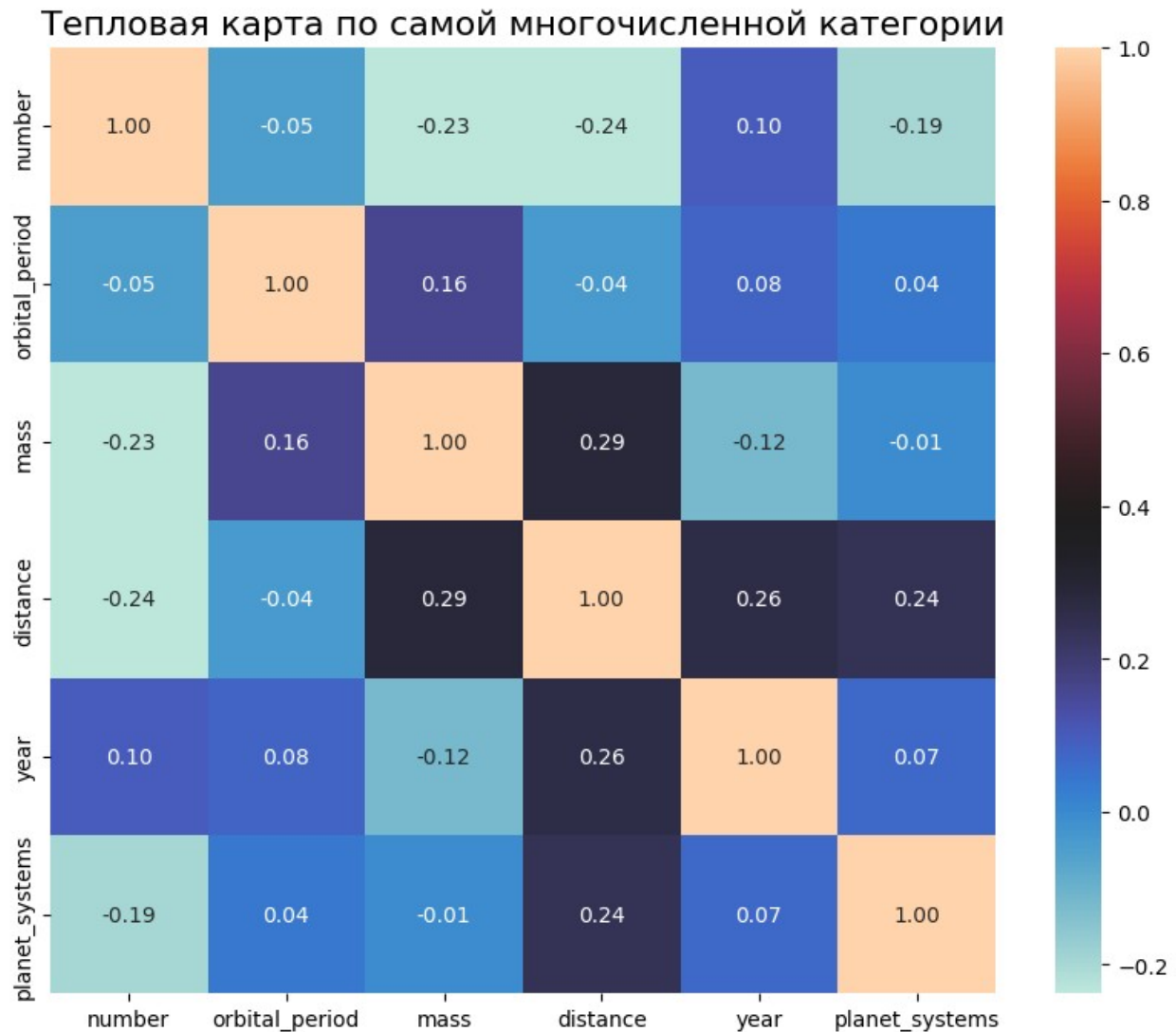
```
cov_matrix = df.loc[:,["number", "orbital_period", "mass", "distance",
"year", "planet_systems"]].corr()
plt.figure(figsize=(10, 8))
plt.title("Тепловая карта по всему датасету", fontsize=16)
sns.heatmap(cov_matrix, annot=True, fmt=".2f", cmap='icefire',
square=True)
```

```
<Axes: title={'center': 'Тепловая карта по всему датасету'}>
```

```
cov_matrix = df.loc[df.method == df.method.value_counts().idxmax(), \
                    ["number", "orbital_period", "mass", "distance",
                    "year", "planet_systems"]].corr()
plt.figure(figsize=(10, 8))
plt.title("Тепловая карта по самой многочисленной категории",
          fontsize=16)
sns.heatmap(cov_matrix, annot=True, fmt=".2f", cmap='icefire',
            square=True)
```

```
<Axes: title={'center': 'Тепловая карта по самой многочисленной
категории'}>
```



```

cov_matrix = df.loc[df.method == df.method.value_counts().idxmin(), \
                    ["number", "orbital_period", "mass", "distance",
"year", "planet_systems"]].corr()
plt.figure(figsize=(10, 8))
plt.title("Тепловая карта по самой малочисленной категории",
          fontsize=16)
sns.heatmap(cov_matrix, annot=True, fmt=".2f", cmap='icefire',
            square=True)

```

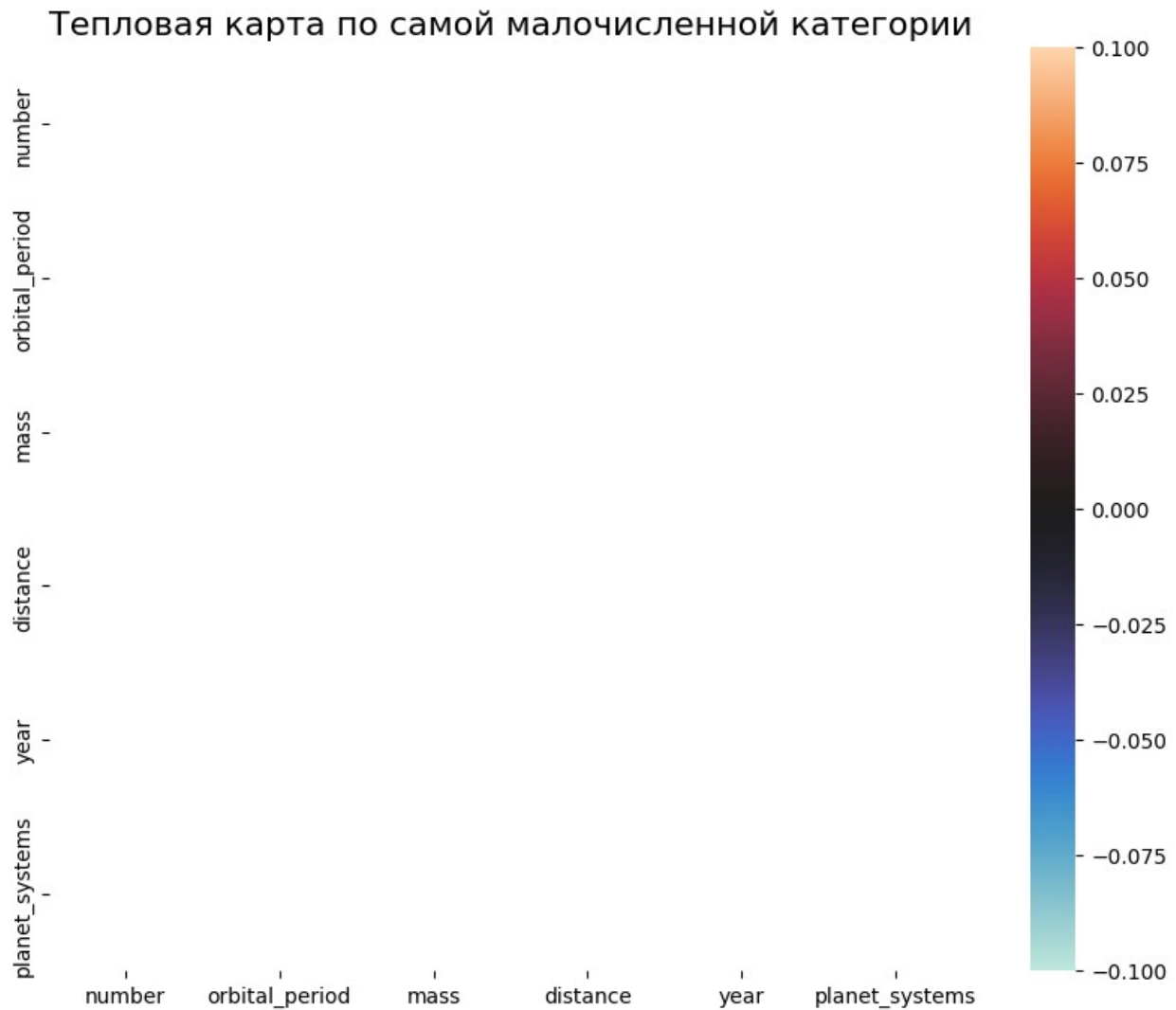
c:\Users\Sai\AppData\Local\Programs\Python\Python312\Lib\site-packages\seaborn\matrix.py:202: RuntimeWarning: All-NaN slice encountered

```
vmin = np.nanmin(calc_data)
```

c:\Users\Sai\AppData\Local\Programs\Python\Python312\Lib\site-packages\seaborn\matrix.py:207: RuntimeWarning: All-NaN slice encountered

```
encountered
vmax = np.nanmax(calc_data)
```

```
<Axes: title={'center': 'Тепловая карта по самой малочисленной
категории'}>
```



У малочисленной категории нечего отрисовывать

```
df.loc[df.method == df.method.value_counts().idxmin(), \
       ["number", "orbital_period", "mass", "distance",
"year", "planet_systems"]]
```

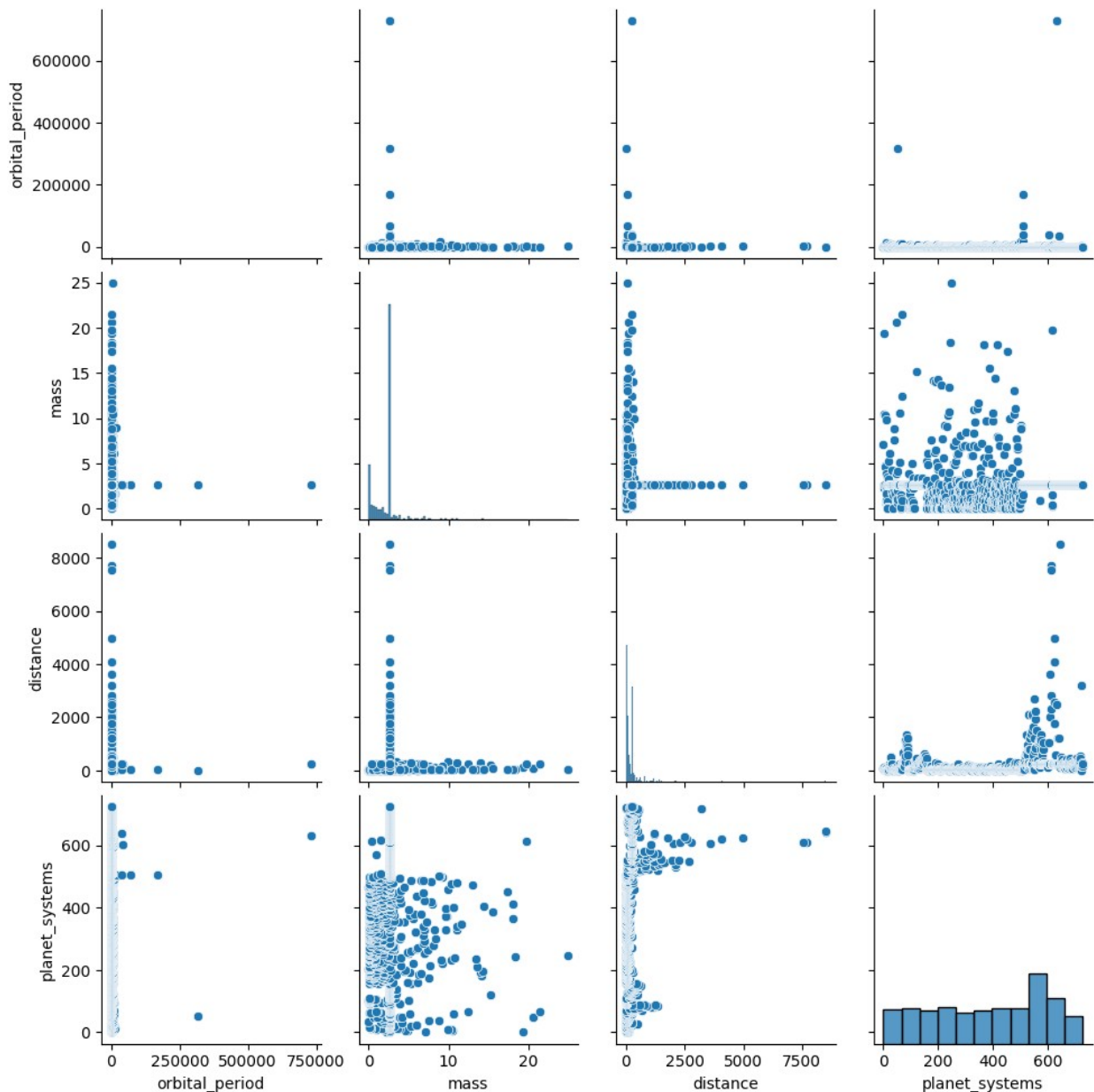
	number	orbital_period	mass	distance	year
planet_systems					
958	1	1170.0	2.638161	264.069282	2007
650					

Строго говоря тут даже корреляцию использовать нельзя, ведь, тут всего одна запись

Диаграмма рассеяния

```
filtered_cols = [col for col in df.columns if df[col].nunique() > 50]

sns.pairplot(df[filtered_cols])
plt.show()
```



На диаграмме рассеяния хорошо видно, что отдельные значения сильно выбиваются из общего распределения

Анализ данных

Задание 14. Планетные системы

1. Укажите, у скольких звезд обнаружены планетные системы.

```
ans = df.code.apply(lambda x: x[1:]).nunique()  
print(f"Количество панетных систем: {ans}")
```

Количество панетных систем: 727

15. Выводы

1. Лучше бы выполнять работу в другом порядке. Сначала строим графики, потом обрабатываем пропуски. А ещё лучше бы выделить другие категории, а не только категории метода обнаружения.
2. Метод "Radial Velocity" позволяет лучше всего определять массу планеты.
3. Среди открытых планет, большая часть имеет массу больше Земли.
4. Распределение признака массы близко к логнормальному.
5. Представленные признаки попарно слабо коррелируют.
6. Можно выделить 7 планет с аномально большим периодом
7. У планет, которые значительно удалены от Земли, неизвестна масса. Это говорит о том, что пропуски имеют систематический характер.