ЛР3

Задание

Для выбранного датасета (датасетов) на основе материалов лекций решить следующие задачи:

- масштабирование признаков (не менее чем тремя способами);
- обработку выбросов для числовых признаков (по одному способу для удаления выбросов и для замены выбросов);
- обработку по крайней мере одного нестандартного признака (который не является числовым или категориальным);
- отбор признаков:
  - один метод из группы методов фильтрации (filter methods);
  - один метод из группы методов обертывания (wrapper methods);
  - один метод из группы методов вложений (embedded methods).

Описание датасета

Космическая обсерватория Кеплер - это спутник, созданный НАСА, который был запущен в 2009 году. Телескоп должен был искать экзопланеты в звездных системах, помимо нашей, с конечной целью, возможно, найти другие обитаемые планеты. По состоянию на май 2016 года Кеплер проверил 1284 новые экзопланеты. По состоянию на октябрь 2017 года насчитывается уже более 3000 подтвержденных экзопланет (с использованием всех методов обнаружения, включая наземные). Телескоп все еще активен и продолжает собирать новые данные о своей расширенной миссии.

Датасет содержит записи всех наблюдаемых Кеплером объектов - 10 000 кандидатов в экзопланеты, за которыми Кеплер наблюдал. Суммарно 9564 строки и 50 колонок.

In [17]:
```
# This Python 3 environment comes with many helpful analytics libraries insta
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will li

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that g
# You can also write temporary files to /kaggle/temp/, but they won't be save

pd.set_option('max_colwidth', 800)
pd.set_option('display.max_columns', None)

import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
from sklearn.datasets import load_boston
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import RobustScaler
from sklearn.preprocessing import MaxAbsScaler
```

```
/kaggle/input/kepler-exoplanet-search-results/cumulative.csv
```

In [2]:
```
data = pd.read_csv('/kaggle/input/kepler-exoplanet-search-results/cumulative.
```

In [3]:
```
data.head()
```

Out[3]:

	rowid	kepid	kepoi_name	kepler_name	koi_disposition	koi_pdisposition	koi_score	koi_
0	1	10797460	K00752.01	Kepler-227 b	CONFIRMED	CANDIDATE	1.000	
1	2	10797460	K00752.02	Kepler-227 c	CONFIRMED	CANDIDATE	0.969	
2	3	10811496	K00753.01	NaN	FALSE POSITIVE	FALSE POSITIVE	0.000	
3	4	10848459	K00754.01	NaN	FALSE POSITIVE	FALSE POSITIVE	0.000	
4	5	10854555	K00755.01	Kepler-664 b	CONFIRMED	CANDIDATE	1.000	

Препроцессинг - Устранение пропусков в данных

В данных присутствуют пропуски

In [4]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9564 entries, 0 to 9563
Data columns (total 50 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   rowid              9564 non-null   int64  
 1   kepid              9564 non-null   int64  
 2   kepoi_name         9564 non-null   object 
 3   kepler_name        2294 non-null   object 
 4   koi_disposition    9564 non-null   object 
 5   koi_pdisposition   9564 non-null   object 
 6   koi_score          8054 non-null   float64
 7   koi_fpflag_nt      9564 non-null   int64  
 8   koi_fpflag_ss      9564 non-null   int64  
 9   koi_fpflag_co      9564 non-null   int64  
 10  koi_fpflag_ec      9564 non-null   int64  
 11  koi_period         9564 non-null   float64
 12  koi_period_err1    9110 non-null   float64
 13  koi_period_err2    9110 non-null   float64
 14  koi_time0bk        9564 non-null   float64
 15  koi_time0bk_err1   9110 non-null   float64
 16  koi_time0bk_err2   9110 non-null   float64
 17  koi_impact         9201 non-null   float64
 18  koi_impact_err1    9110 non-null   float64
 19  koi_impact_err2    9110 non-null   float64
 20  koi_duration       9564 non-null   float64
 21  koi_duration_err1  9110 non-null   float64
 22  koi_duration_err2  9110 non-null   float64
 23  koi_depth          9201 non-null   float64
 24  koi_depth_err1     9110 non-null   float64
 25  koi_depth_err2     9110 non-null   float64
 26  koi_prad           9201 non-null   float64
 27  koi_prad_err1      9201 non-null   float64
 28  koi_prad_err2      9201 non-null   float64
 29  koi_teq            9201 non-null   float64
 30  koi_teq_err1       0 non-null      float64
 31  koi_teq_err2       0 non-null      float64
 32  koi_insol          9243 non-null   float64
 33  koi_insol_err1     9243 non-null   float64
 34  koi_insol_err2     9243 non-null   float64
 35  koi_model_snr      9201 non-null   float64
 36  koi_tce_plnt_num   9218 non-null   float64
 37  koi_tce_delivname  9218 non-null   object 
 38  koi_steff          9201 non-null   float64
 39  koi_steff_err1     9096 non-null   float64
 40  koi_steff_err2     9081 non-null   float64
 41  koi_slogg          9201 non-null   float64
 42  koi_slogg_err1     9096 non-null   float64
 43  koi_slogg_err2     9096 non-null   float64
 44  koi_srad           9201 non-null   float64
 45  koi_srad_err1      9096 non-null   float64
 46  koi_srad_err2      9096 non-null   float64
 47  ra                 9564 non-null   float64
 48  dec                9564 non-null   float64
 49  koi_kepmag         9563 non-null   float64
dtypes: float64(39), int64(6), object(5)
memory usage: 3.6+ MB
```

In [128…
```
dict_null = dict()
for c in data.columns:
    dict_null[c] = data[c].isnull().sum()*100.0/9564.0
```

Удалю признаки с количесвтом пропуском >5%

In [6]:
```
data.drop(axis=1, columns=list({k for (k,v) in dict_null.items() if v>5}), in
```

In [7]:
```
data.shape
```

Out[7]: (9564, 45)

Пропуски в числовых признаках буду заполнять медианой, а пропуски в категориальных признаках буду заполнять специальным значением - "fill"

In [9]:
```
object_col = []
for i in list({k for (k,v) in dict_null.items() if v>0 and v<5}):
    if data.dtypes[i] == 'int64' or data.dtypes[i] == 'float64':
        data[i] = data[i].fillna(data[i].median())
    else:
        object_col.append(i)
        data[i] = data[i].fillna('fill')
```

Кодирование категориальных признаков

Кодирование категориальных признаков буду производить с помощью разных энкодеров: kepoi_name, уникальные значения, - sklearn.LabelEncoder остальные - category_encoders.CountEncoder

In [10]:
```
object_col = ["kepoi_name", "koi_disposition", "koi_pdisposition", 'koi_tce_de
```

In [14]:
```
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

from category_encoders.count import CountEncoder as ce_CountEncoder

ce = ce_CountEncoder(normalize=True)

for i in data.columns:
    if data.dtypes[i] == 'object':
        if i == 'kepoi_name':
            data[i] = le.fit_transform(data[i])
        else:
            data[i] = ce.fit_transform(data[i])
```

In [15]:
```
data.head()
```

Out[15]:

	rowid	kepid	kepoi_name	koi_disposition	koi_pdisposition	koi_fpflag_nt	koi_fpflag_ss
0	1	10797460	1080	0.239753	0.470096	0	0

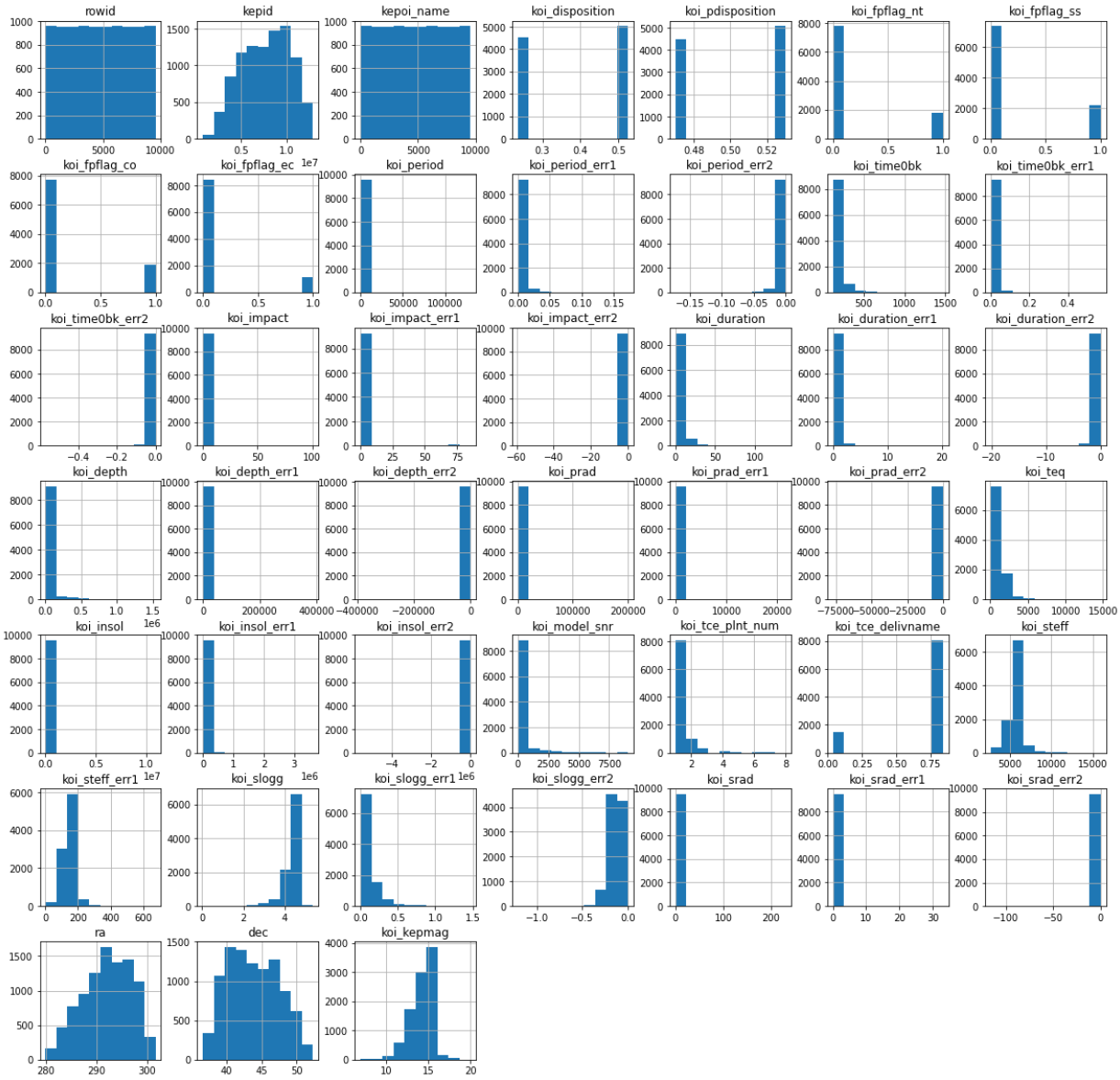
	rowid	kepid	kepoi_name	koi_disposition	koi_pdisposition	koi_fpflag_nt	koi_fpflag_ss
1	2	10797460	1081	0.239753	0.470096	0	0
2	3	10811496	1082	0.525199	0.529904	0	1
3	4	10848459	1083	0.525199	0.529904	0	1
4	5	10854555	1084	0.239753	0.470096	0	0

В датасете не осталось признаков с типом object

Гистограммы признаков до масштабирования

In [16]:
```
import matplotlib.pyplot as plt
data.hist(figsize=(20,20))
plt.show()
```

```
/opt/conda/lib/python3.7/site-packages/pandas/plotting/_matplotlib/tools.py:400: MatplotlibDeprecationWarning:
The is_first_col function was deprecated in Matplotlib 3.4 and will be removed two minor releases later. Use ax.get_subplotspec().is_first_col() instead.
  if ax.is_first_col():
```



1. Масштабирование признаков

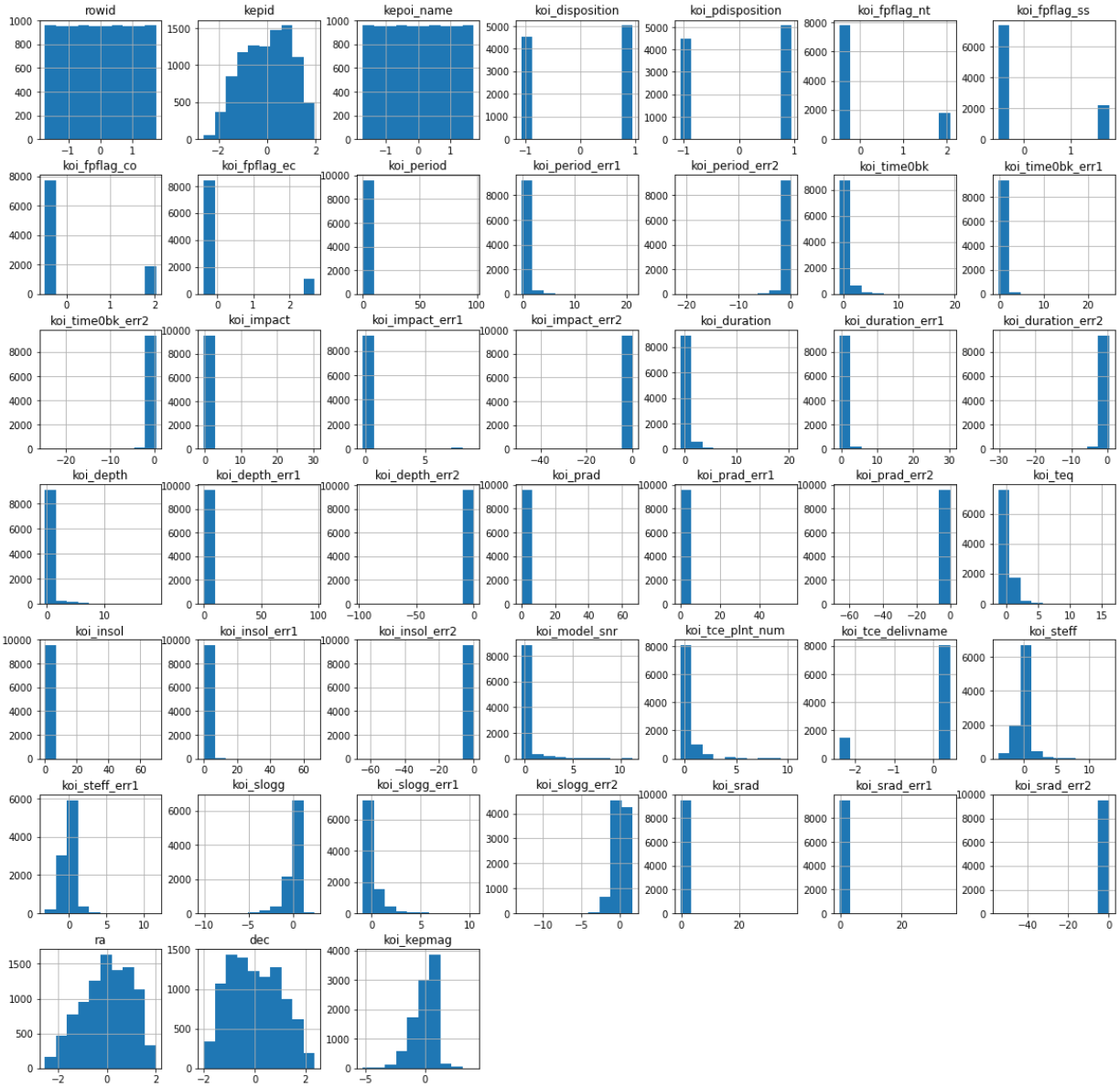
1.1. Масштабирование данных на основе Z-оценки

Реализван в StandardScaler

In [18]:
```
z_scale = StandardScaler()
data_z_scale = z_scale.fit_transform(data)
```

In [20]:
```
pd.DataFrame(data_z_scale, columns=data.columns).hist(figsize=(20,20))
plt.show()
```

```
/opt/conda/lib/python3.7/site-packages/pandas/plotting/_matplotlib/tools.py:400: MatplotlibDeprecationWarning:
The is_first_col function was deprecated in Matplotlib 3.4 and will be removed two minor releases later. Use ax.get_subplotspec().is_first_col() instead.
  if ax.is_first_col():
```



В результате применения z-оценки средние значения были приведены к 0, формы исходных распределений сохраняются, максимальные и минимальные значения варьируются, выбросы созранились.

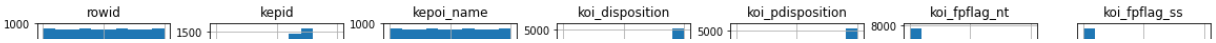
1.2. MinMax-масштабирование

Реализван в MinMaxScaler

In [21]:
```
minmax_scaler = MinMaxScaler()
data_minmax = minmax_scaler.fit_transform(data)
```

In [22]:
```
pd.DataFrame(data_minmax, columns=data.columns).hist(figsize=(20,20))
plt.show()
```

```
/opt/conda/lib/python3.7/site-packages/pandas/plotting/_matplotlib/tools.py:40
0: MatplotlibDeprecationWarning:
The is_first_col function was deprecated in Matplotlib 3.4 and will be removed
two minor releases later. Use ax.get_subplotspec().is_first_col() instead.
  if ax.is_first_col():
```



В результате применения MinMaxScaler средние значения варьируются, формы распределений существенно не изменились, максимальные и минимальные значения находятся в диапазоне [0;1], выбросы сохранились.

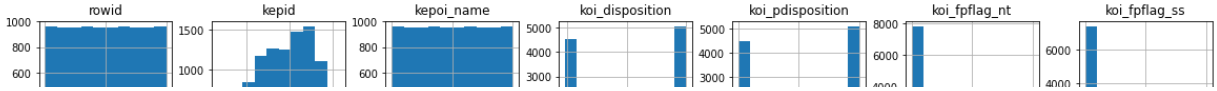
1.3. Масштабирование по медиане

Реализовано в RobustScaler

In [23]:
```
rob_scaler = RobustScaler()
data_rob_scaler = rob_scaler.fit_transform(data)
```

In [24]:
```
pd.DataFrame(data_rob_scaler, columns=data.columns).hist(figsize=(20,20))
plt.show()
```

```
/opt/conda/lib/python3.7/site-packages/pandas/plotting/_matplotlib/tools.py:40
0: MatplotlibDeprecationWarning:
The is_first_col function was deprecated in Matplotlib 3.4 and will be removed
two minor releases later. Use ax.get_subplotspec().is_first_col() instead.
  if ax.is_first_col():
```



In [28]:
```
print(data.koi_slogg_err2.median())
pd.DataFrame(data_rob_scaler, columns=data.columns).koi_slogg_err2.median()
```

```
-0.128
```

Out[28]:
```
0.0
```

В результате применения Robust_scaler медиана распределений была приведена приводится к 0, форма исходных распределений изменилась, **выбросы устранились**.

2. Обработка выбросов для числовых признаков

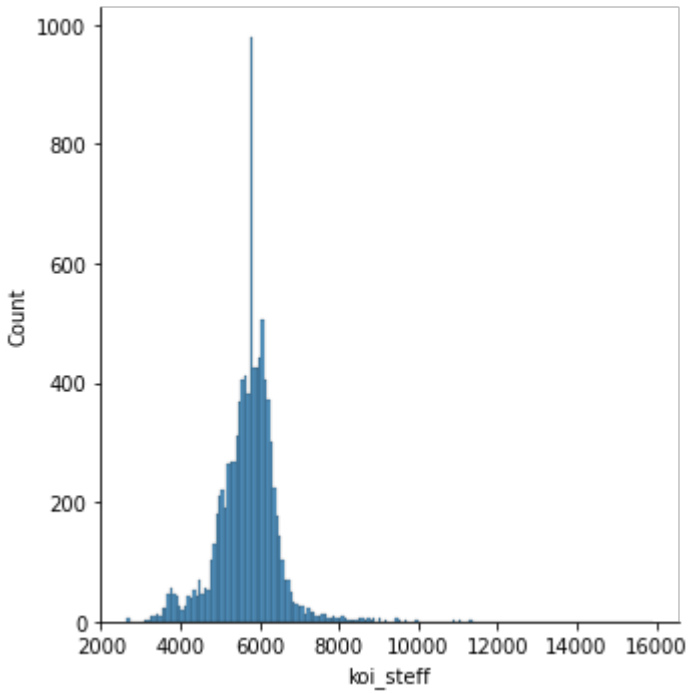
2.1. Замена выбросов на основе 5% и 95% квантилей

Обнаружение и замена выбросов будет производиться для признака koi_steff

In [29]:
```
import matplotlib.pyplot as plt
import seaborn as sns

sns.displot(data, x="koi_steff")
```

Out[29]:
```
<seaborn.axisgrid.FacetGrid at 0x7f13c74f6b50>
```

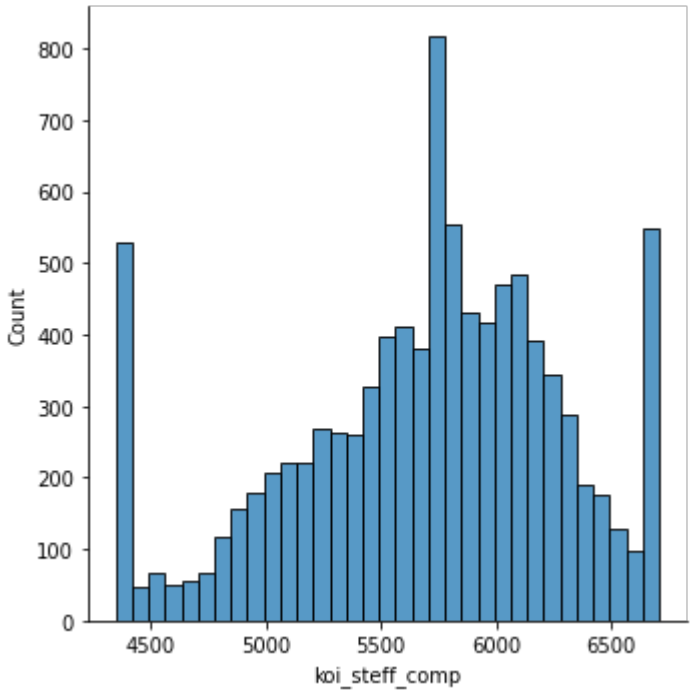


In [30]:
```
q5 = data.koi_steff.quantile(0.05)
q95 = data.koi_steff.quantile(0.95)
print(q5, q95)

data['koi_steff_comp'] = np.where(data.koi_steff < q5, q5,
                                  np.where(data.koi_steff > q95, q95, data.koi_steff))
sns.displot(data, x="koi_steff_comp")
```

```
4348.0 6706.0
```

Out[30]:
```
<seaborn.axisgrid.FacetGrid at 0x7f13c7331150>
```



2.2. Удаление с использованием межквартильного размаха

Обнаружение и удаление выбросов будет производиться для признака koi_steff

In [32]:
```
q25, q75
```

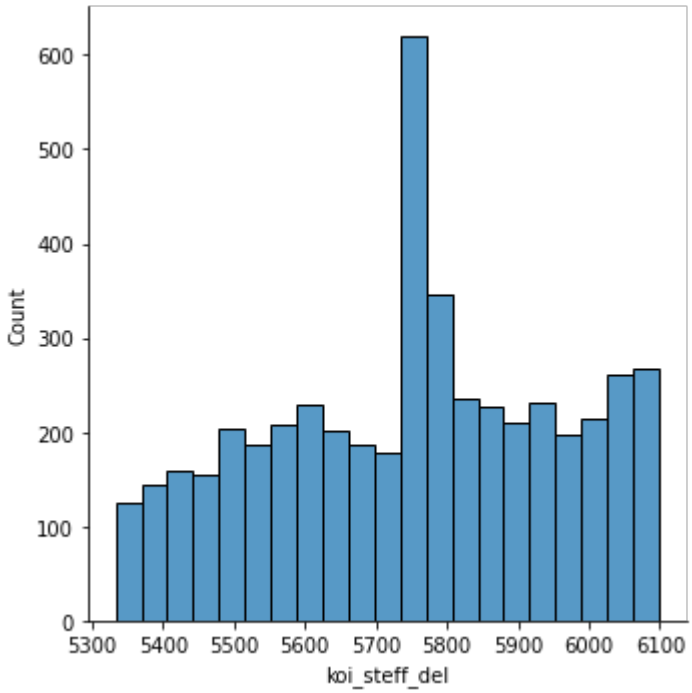
Out[32]:
```
(5333.0, 6099.0)
```

In [31]:
```
K = 1.5
q25 = data.koi_steff.quantile(0.25)
q75 = data.koi_steff.quantile(0.75)


IQR = q75 - q25
lower_boundary = q25 - (K * IQR)
upper_boundary = q75 + (K * IQR)

data['koi_steff_del'] = data.koi_steff.loc[~(np.where(data.koi_steff < q25, T
                         np.where(data.koi_steff > q75, True, False))), ]
sns.displot(data, x="koi_steff_del")
```

Out[31]: <seaborn.axisgrid.FacetGrid at 0x7f13c7baedd0>



3. Обработка нестандартного признака

В датасете не представлено нестандартных признаков, поэтому добавлю колонку со случайными датами и создам с ее помощью производные признаки - день недели, год, месяц, неделя

In [121…
```
from random import randrange
from datetime import timedelta
from datetime import datetime

d1 = datetime.strptime('1/1/2009 1:30 PM', '%m/%d/%Y %I:%M %p')
d2 = datetime.strptime('1/1/2017 4:50 AM', '%m/%d/%Y %I:%M %p')


def random_date(d,start=d1, end=d2):

    delta = end - start
    int_delta = (delta.days * 24 * 60 * 60) + delta.seconds
    random_second = randrange(int_delta)
    return start + timedelta(seconds=random_second)
```

In [122…
```
data['random_date'] = data.koi_steff.apply(random_date)
```

In [123…
```
data.random_date
```

Out[123…
```
0       2013-09-26 00:44:46
1       2013-10-05 06:57:29
2       2015-02-08 21:16:00
3       2009-06-18 18:25:12
4       2014-09-01 18:24:03
                ...
9559    2010-11-03 02:47:14
9560    2009-12-09 12:25:36
9561    2010-07-24 05:46:35
9562    2014-01-06 20:21:24
9563    2011-05-18 01:45:23
Name: random_date, Length: 9564, dtype: datetime64[ns]
```

In [125…
```
data['random_date_weekday'] = data['random_date'].apply(lambda x: x.date().wee
data['random_date_year'] = data['random_date'].dt.year
data['random_date_month'] = data['random_date'].dt.month
data['random_date_week'] = data['random_date'].dt.week
```

```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:4: FutureWarning:
Series.dt.weekofyear and Series.dt.week have been deprecated.  Please use Seri
es.dt.isocalendar().week instead.
  after removing the cwd from sys.path.
```

Результат

In [127…
```
data[['random_date','random_date_weekday', 'random_date_year', 'random_date_mo
```

Out[127…

	random_date	random_date_weekday	random_date_year	random_date_month	random_dat
0	2013-09-26 00:44:46	3	2013	9	
1	2013-10-05 06:57:29	5	2013	10	
2	2015-02-08 21:16:00	6	2015	2	

	random_date	random_date_weekday	random_date_year	random_date_month	random_dat
3	2009-06-18 18:25:12	3	2009	6	
4	2014-09-01 18:24:03	0	2014	9	
...	...	...	...	...	
9559	2010-11-03 02:47:14	2	2010	11	
9560	2009-12-09 12:25:36	2	2009	12	
9561	2010-07-24 05:46:35	5	2010	7	
9562	2014-01-06 20:21:24	0	2014	1	
9563	2011-05-18 01:45:23	2	2011	5	

4. Отбор признаков

4.1. Группа методов фильтрации - Удаление константных и псевдоконстантных признаков

VarianceThreshold

In [35]:
```
from sklearn.feature_selection import VarianceThreshold
from sklearn.feature_selection import mutual_info_classif, mutual_info_regress
from sklearn.feature_selection import SelectKBest, SelectPercentile
```

In [39]:
```
filter_method = VarianceThreshold(threshold=0.15)
filter_method.fit(data)
# Значения дисперсий для каждого признака
dict(zip(data.columns, filter_method.variances_))
```

Out[39]:
```
{'rowid': 7622507.916666667,
 'kepid': 7040108912356.319,
 'kepoi_name': 7622507.916666667,
 'koi_disposition': 0.020653609281324942,
 'koi_pdisposition': 0.0008910389660237114,
 'koi_fpflag_nt': 0.15278435916367686,
 'koi_fpflag_ss': 0.1779601827465011,
 'koi_fpflag_co': 0.15691248427245552,
 'koi_fpflag_ec': 0.10562542757143273,
 'koi_period': 1781355.3927057358,
 'koi_period_err1': 6.48122663732829e-05,
 'koi_period_err2': 6.48122663732829e-05,
 'koi_time0bk': 4612.502742103357,
 'koi_time0bk_err1': 0.0005096325609101442,
 'koi_time0bk_err2': 0.0005096325609101442,
 'koi_impact': 10.789285214755926,
 'koi_impact_err1': 84.68564850929324,
 'koi_impact_err2': 1.4884690937431888,
```

```
 'koi_duration': 41.876628747831056,
 'koi_duration_err1': 0.42908967780327,
 'koi_duration_err2': 0.42908967780327,
 'koi_depth': 6526373509.186501,
 'koi_depth_err1': 16109426.799172996,
 'koi_depth_err2': 16109426.799172996,
 'koi_prad': 9111738.100988487,
 'koi_prad_err1': 147177.37117065108,
 'koi_prad_err2': 1370313.6638827673,
 'koi_teq': 706997.3790234023,
 'koi_insol': 24494649657.21997,
 'koi_insol_err1': 2928294468.9121385,
 'koi_insol_err2': 7549982970.327816,
 'koi_model_snr': 611253.9985340558,
 'koi_tce_plnt_num': 0.42770301673237554,
 'koi_tce_delivname': 0.08109291530427024,
 'koi_steff': 610947.7655723025,
 'koi_steff_err1': 2112.46794260498,
 'koi_slogg': 0.18062238166505257,
 'koi_slogg_err1': 0.016900074811684904,
 'koi_slogg_err2': 0.006958651789236123,
 'koi_srad': 36.132941429704886,
 'koi_srad_err1': 0.8246036986678139,
 'koi_srad_err2': 4.474362888040906,
 'ra': 22.71864485303107,
 'dec': 12.967592845888982,
 'koi_kepmag': 1.919071933565731,
 'koi_steff_comp': 353139.3178495407,
 'koi_steff_del': 41448.129036563834}
```

Здесь присутствуют маловариативные признаки. Удалим константные и псевдоконстантные признаки - из 52 осталось 37

In [42]:
```
len(filter_method.transform(data)[0])
```

Out[42]: 37

4.2. Группа методов обертывания - Алгоритм полного перебора

ExhaustiveFeatureSelector

Будет решаться задача классификации, целевой признак - koi_disposition. Проведу заново обработку датасета

In [43]:
```
data = pd.read_csv('/kaggle/input/kepler-exoplanet-search-results/cumulative.
```

In [45]:
```
data.head()
```

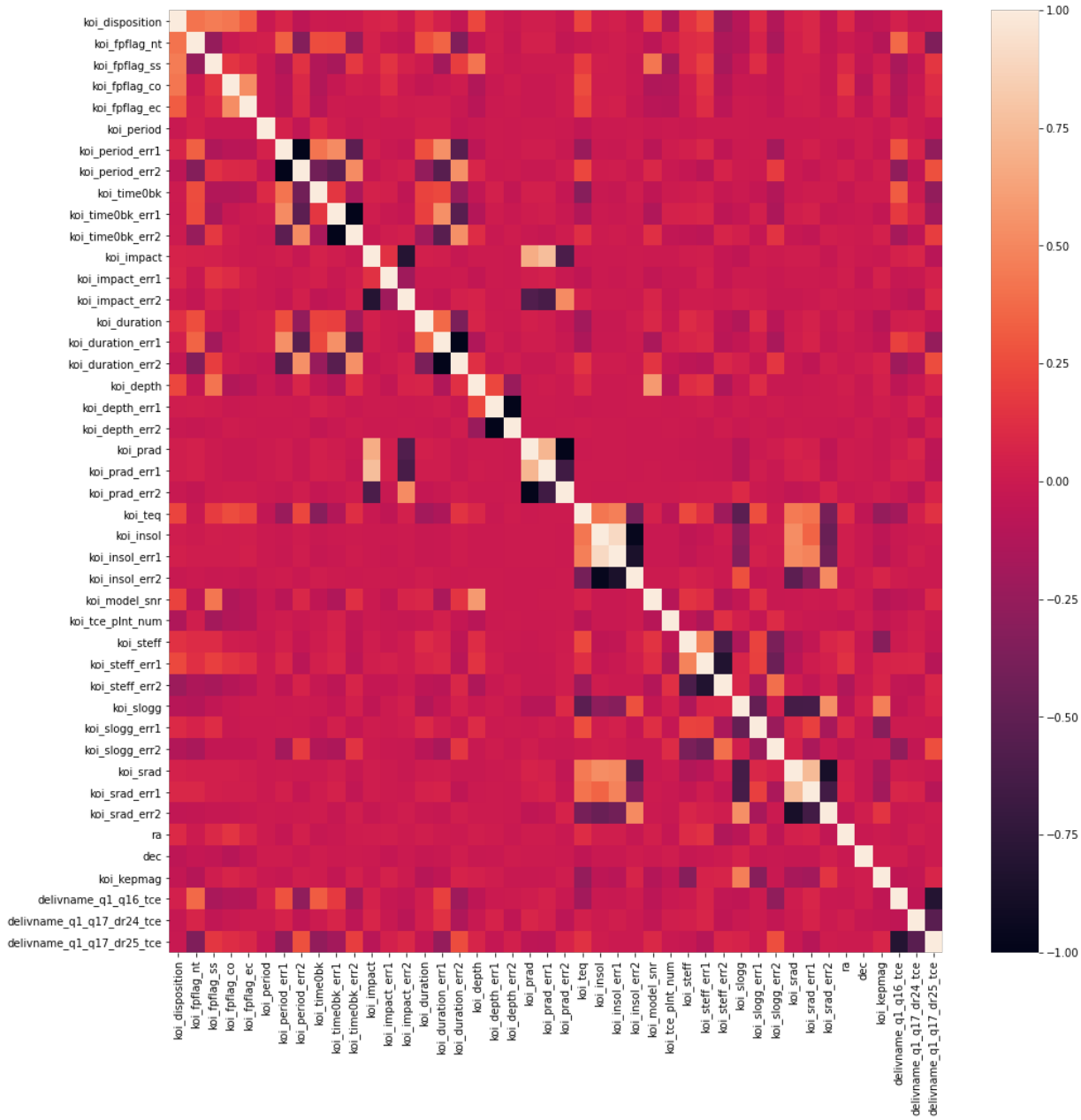
Out[45]:

	rowid	kepid	kepoi_name	kepler_name	koi_disposition	koi_pdisposition	koi_score	koi_
0	1	10797460	K00752.01	Kepler-227 b	CONFIRMED	CANDIDATE	1.000	
1	2	10797460	K00752.02	Kepler-227 c	CONFIRMED	CANDIDATE	0.969	
2	3	10811496	K00753.01	NaN	FALSE POSITIVE	FALSE POSITIVE	0.000	

	rowid	kepid	kepoi_name	kepler_name	koi_disposition	koi_pdisposition	koi_score	koi_
3	4	10848459	K00754.01	NaN	FALSE	FALSE POSITIVE	0.000	

In [77]:
```
plt.figure(figsize = (16,16))
sns.heatmap(df.corr(),fmt='.3f')
```

Out[77]: <AxesSubplot:>



In [ ]:
```
ТАК ЖЕ НАДО УДАЛИТЬ 'koi_period_err1', 'koi_time0bk_err1', 'koi_impact_err1',
```

In [66]:
```
data.columns
```

Out[66]:
```
Index(['rowid', 'kepid', 'kepoi_name', 'kepler_name', 'koi_disposition',
       'koi_pdisposition', 'koi_score', 'koi_fpflag_nt', 'koi_fpflag_ss',
       'koi_fpflag_co', 'koi_fpflag_ec', 'koi_period', 'koi_period_err1',
```

```
       'koi_period_err2', 'koi_time0bk', 'koi_time0bk_err1',
       'koi_time0bk_err2', 'koi_impact', 'koi_impact_err1', 'koi_impact_err2',
       'koi_duration', 'koi_duration_err1', 'koi_duration_err2', 'koi_depth',
       'koi_depth_err1', 'koi_depth_err2', 'koi_prad', 'koi_prad_err1',
       'koi_prad_err2', 'koi_teq', 'koi_teq_err1', 'koi_teq_err2', 'koi_insol
',
       'koi_insol_err1', 'koi_insol_err2', 'koi_model_snr', 'koi_tce_plnt_num
',
       'koi_tce_delivname', 'koi_steff', 'koi_steff_err1', 'koi_steff_err2',
       'koi_slogg', 'koi_slogg_err1', 'koi_slogg_err2', 'koi_srad',
       'koi_srad_err1', 'koi_srad_err2', 'ra', 'dec', 'koi_kepmag'],
```

Сокращу пространство признаков

In [93]:

```
feature = [
       'koi_fpflag_nt', 'koi_fpflag_ss',
       'koi_fpflag_co', 'koi_fpflag_ec', 'koi_period',
       'koi_duration',  'koi_depth',
        'koi_prad',  'koi_teq',
        'koi_model_snr', 'koi_tce_plnt_num',
       'koi_steff',
        'ra', 'dec', 'koi_kepmag']
```

In [90]:

```
plt.figure(figsize = (16,16))
sns.heatmap(df[feature].corr(),fmt='.3f')
```

Out[90]: <AxesSubplot:>

In [94]:

```
# Удалю неинформативные признаки, например rowid - номер строки датасета
df = data.copy()
df = df.drop(['rowid', 'kepid', 'kepoi_name', 'kepler_name', 'koi_pdisposition

# Удалю пустые признаки
df = df.drop(['koi_teq_err1', 'koi_teq_err2'], axis=1)

# Заполнение пропусков
df['koi_tce_delivname'] = df['koi_tce_delivname'].fillna(df['koi_tce_delivname
for column in df.columns[df.isna().sum() > 0]:
    df[column] = df[column].fillna(df[column].mean())

# One-hot encode koi_tce_delivname
delivname_dummies = pd.get_dummies(df['koi_tce_delivname'], prefix='delivname
df = pd.concat([df, delivname_dummies], axis=1)
df = df.drop('koi_tce_delivname', axis=1)

from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

from category_encoders.count import CountEncoder as ce_CountEncoder

ce = ce_CountEncoder(normalize=True)

for i in data.columns:
    if data.dtypes[i] == 'object':
        if i == 'koi_disposition':
            data[i] = le.fit_transform(data[i])
        else:
            data[i] = ce.fit_transform(data[i])

# Разделение датасета признаки и целевой признак
y = df['koi_disposition']
X = df.drop('koi_disposition', axis=1)[feature]

# Scale X
scaler = StandardScaler()
scaler.fit(X)
```

Out[94]:

```
StandardScaler()
```

In [95]:

```
X.shape
```

Out[95]:

```
(9564, 15)
```

In [96]:
```
from mlxtend.feature_selection import ExhaustiveFeatureSelector as EFS
from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier(n_neighbors=10)
```

In [97]:
```
efs = EFS(knn,
          min_features=10,
          max_features=15,
          scoring='accuracy',
          print_progress=True,
          cv=5)

efs = efs.fit(X, y, custom_feature_names=X.columns)

print('Best accuracy score: %.2f' % efs.best_score_)
print('Best subset (indices):', efs.best_idx_)
print('Best subset (corresponding names):', efs.best_feature_names_)
```

```
Features: 4944/4944
Best accuracy score: 0.69
Best subset (indices): (0, 1, 2, 3, 4, 5, 7, 9, 10, 14)
Best subset (corresponding names): ('koi_fpflag_nt', 'koi_fpflag_ss', 'koi_fpf
lag_co', 'koi_fpflag_ec', 'koi_period', 'koi_duration', 'koi_prad', 'koi_model
_snr', 'koi_tce_plnt_num', 'koi_kepmag')
```

с 44 признаками accuracy была 0.64

с 10 признаками - 0.69

Самые важные признаки: 'koi_fpflag_nt', 'koi_fpflag_ss', 'koi_fpflag_co', 'koi_fpflag_ec', 'koi_period', 'koi_duration', 'koi_prad', 'koi_model_snr', 'koi_tce_plnt_num', 'koi_kepmag'

4.3. Метод вложений - логистическая регрессия

SelectFromModel

In [101…
```
from sklearn.linear_model import LogisticRegression
from sklearn.feature_selection import SelectFromModel
# Используем L1-регуляризацию
e_lr1 = LogisticRegression(C=1000, solver='liblinear', penalty='l1', max_iter=
e_lr1.fit(X[feature], y)
# Коэффициенты регрессии
print(e_lr1.coef_)

sel_e_lr1 = SelectFromModel(e_lr1)
sel_e_lr1.fit(X[feature], y)
sel_e_lr1.get_support()
```

```
/opt/conda/lib/python3.7/site-packages/sklearn/svm/_base.py:986: ConvergenceWa
rning: Liblinear failed to converge, increase the number of iterations.
  "the number of iterations.", ConvergenceWarning)
[[-6.47158220e+00 -3.26939741e+00 -5.65273814e+00 -5.53012534e+00
   5.44456617e-03 -2.20041433e-02 -4.76437604e-07 -1.91389521e-05
   4.04323824e-04 -1.19837194e-03 -1.89306658e-01  2.36813700e-05
  -6.01357410e-05 -3.96824420e-04  2.42063908e-02]
```

```
 [-3.13991044e+00 -2.69956798e+00 -4.22768460e+00 -4.79837974e+00
  -7.94341703e-03  2.39875698e-02 -6.50135215e-05 -3.89668458e-02
  -6.12760623e-04  1.43505813e-03  2.43323440e-01 -2.50797768e-04
  -1.53152624e-04  3.93731686e-02 -5.21035106e-02]
 [ 7.74769167e+00  5.87574184e+00  8.27748836e+00  8.62625095e+00
   2.86774405e-06  8.06484296e-02  3.49358128e-05  3.14715118e-04
   2.12060258e-04 -4.24890713e-04 -4.69407250e-01  2.13522588e-04
```

```
/opt/conda/lib/python3.7/site-packages/sklearn/svm/_base.py:986: ConvergenceWa
rning: Liblinear failed to converge, increase the number of iterations.
  "the number of iterations.", ConvergenceWarning)
```

Out[101…
```
array([ True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True])
```

Все ризнаки оказались важными

In [ ]: