ЛР2

Задание

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

- устранение пропусков в данных;
- кодирование категориальных признаков;
- нормализацию числовых признаков.

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 [1]:
```
# 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)
```

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

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

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	

1. 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)
```

In [5]:

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

{k:v for (k,v) in dict_null.items() if v>0}
```

Out[5]:

```
{'kepler_name': 76.0142199916353,
 'koi_score': 15.788373065662903,
 'koi_period_err1': 4.746967795901297,
 'koi_period_err2': 4.746967795901297,
 'koi_time0bk_err1': 4.746967795901297,
 'koi_time0bk_err2': 4.746967795901297,
 'koi_impact': 3.795483061480552,
 'koi_impact_err1': 4.746967795901297,
 'koi_impact_err2': 4.746967795901297,
 'koi_duration_err1': 4.746967795901297,
 'koi_duration_err2': 4.746967795901297,
 'koi_depth': 3.795483061480552,
 'koi_depth_err1': 4.746967795901297,
 'koi_depth_err2': 4.746967795901297,
 'koi_prad': 3.795483061480552,
 'koi_prad_err1': 3.795483061480552,
 'koi_prad_err2': 3.795483061480552,
 'koi_teq': 3.795483061480552,
 'koi_teq_err1': 100.0,
 'koi_teq_err2': 100.0,
 'koi_insol': 3.35633626097867,
 'koi_insol_err1': 3.35633626097867,
 'koi_insol_err2': 3.35633626097867,
 'koi_model_snr': 3.795483061480552,
 'koi_tce_plnt_num': 3.617733166039314,
 'koi_tce_delivname': 3.617733166039314,
 'koi_steff': 3.795483061480552,
 'koi_steff_err1': 4.893350062735257,
 'koi_steff_err2': 5.050188205771644,
 'koi_slogg': 3.795483061480552,
 'koi_slogg_err1': 4.893350062735257,
 'koi_slogg_err2': 4.893350062735257,
 'koi_srad': 3.795483061480552,
 'koi_srad_err1': 4.893350062735257,
 'koi_srad_err2': 4.893350062735257,
 'koi_kepmag': 0.010455876202425763}
```

Удалю прзнаки 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 [8]:
```
data.dtypes
```

Out[8]:
```
rowid                 int64
kepid                 int64
kepoi_name           object
koi_disposition      object
koi_pdisposition     object
koi_fpflag_nt         int64
koi_fpflag_ss         int64
koi_fpflag_co         int64
koi_fpflag_ec         int64
koi_period          float64
koi_period_err1     float64
koi_period_err2     float64
koi_time0bk         float64
koi_time0bk_err1    float64
koi_time0bk_err2    float64
koi_impact          float64
koi_impact_err1     float64
koi_impact_err2     float64
koi_duration        float64
koi_duration_err1   float64
koi_duration_err2   float64
koi_depth           float64
koi_depth_err1      float64
koi_depth_err2      float64
koi_prad            float64
koi_prad_err1       float64
koi_prad_err2       float64
koi_teq             float64
koi_insol           float64
koi_insol_err1      float64
koi_insol_err2      float64
koi_model_snr       float64
koi_tce_plnt_num    float64
koi_tce_delivname    object
koi_steff           float64
koi_steff_err1      float64
koi_slogg           float64
koi_slogg_err1      float64
koi_slogg_err2      float64
koi_srad            float64
koi_srad_err1       float64
koi_srad_err2       float64
ra                  float64
dec                 float64
koi_kepmag          float64
dtype: object
```

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')
```

In [10]:
```
data.isnull().sum()
```

Out[10]:
```
rowid                0
kepid                0
kepoi_name           0
koi_disposition      0
koi_pdisposition     0
koi_fpflag_nt        0
koi_fpflag_ss        0
koi_fpflag_co        0
koi_fpflag_ec        0
koi_period           0
koi_period_err1      0
koi_period_err2      0
koi_time0bk          0
koi_time0bk_err1     0
koi_time0bk_err2     0
koi_impact           0
koi_impact_err1      0
koi_impact_err2      0
koi_duration         0
koi_duration_err1    0
koi_duration_err2    0
koi_depth            0
koi_depth_err1       0
koi_depth_err2       0
koi_prad             0
koi_prad_err1        0
koi_prad_err2        0
koi_teq              0
koi_insol            0
koi_insol_err1       0
koi_insol_err2       0
koi_model_snr        0
koi_tce_plnt_num     0
koi_tce_delivname    0
koi_steff            0
koi_steff_err1       0
koi_slogg            0
koi_slogg_err1       0
koi_slogg_err2       0
koi_srad             0
koi_srad_err1        0
koi_srad_err2        0
ra                   0
dec                  0
koi_kepmag           0
dtype: int64
```

2. Кодирование категориальных признаков

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

category_encoders.CountEncoder

In [24]:
```
object_col = ["kepoi_name", "koi_disposition", "koi_pdisposition", 'koi_tce_de
```

In [27]:
```
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 [28]:
```
data.head()
```

Out[28]:

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

In [18]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9564 entries, 0 to 9563
Data columns (total 45 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   koi_disposition    9564 non-null   object 
 4   koi_pdisposition   9564 non-null   object 
 5   koi_fpflag_nt      9564 non-null   int64  
 6   koi_fpflag_ss      9564 non-null   int64  
 7   koi_fpflag_co      9564 non-null   int64  
 8   koi_fpflag_ec      9564 non-null   int64  
 9   koi_period         9564 non-null   float64
 10  koi_period_err1    9564 non-null   float64
 11  koi_period_err2    9564 non-null   float64
 12  koi_time0bk        9564 non-null   float64
 13  koi_time0bk_err1   9564 non-null   float64
 14  koi_time0bk_err2   9564 non-null   float64
 15  koi_impact         9564 non-null   float64
 16  koi_impact_err1    9564 non-null   float64
```

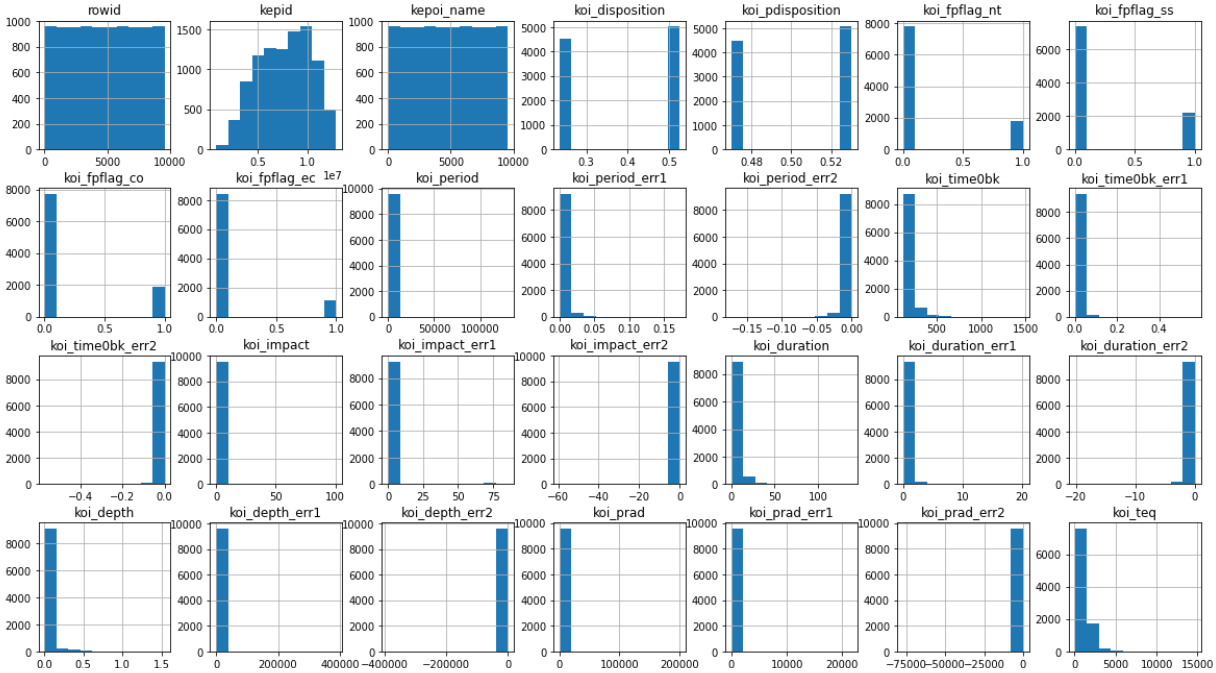
```
 17  koi_impact_err2    9564 non-null   float64
 18  koi_duration       9564 non-null   float64
 19  koi_duration_err1  9564 non-null   float64
 20  koi_duration_err2  9564 non-null   float64
 21  koi_depth          9564 non-null   float64
 22  koi_depth_err1     9564 non-null   float64
 23  koi_depth_err2     9564 non-null   float64
 24  koi_prad           9564 non-null   float64
 25  koi_prad_err1      9564 non-null   float64
 26  koi_prad_err2      9564 non-null   float64
 27  koi_teq            9564 non-null   float64
 28  koi_insol          9564 non-null   float64
 29  koi_insol_err1     9564 non-null   float64
 30  koi_insol_err2     9564 non-null   float64
 31  koi_model_snr      9564 non-null   float64
 32  koi_tce_plnt_num   9564 non-null   float64
 33  koi_tce_delivname  9564 non-null   float64
 34  koi_steff          9564 non-null   float64
 35  koi_steff_err1     9564 non-null   float64
 36  koi_slogg          9564 non-null   float64
 37  koi_slogg_err1     9564 non-null   float64
 38  koi_slogg_err2     9564 non-null   float64
 39  koi_srad           9564 non-null   float64
 40  koi_srad_err1      9564 non-null   float64
 41  koi_srad_err2      9564 non-null   float64
 42  ra                 9564 non-null   float64
 43  dec                9564 non-null   float64
 44  koi_kepmag         9564 non-null   float64
dtypes: float64(36), int64(6), object(3)
memory usage: 3.3+ MB
```

3. Нормализация числовых признаков

Для нормальзации числовых признаков буду использовать преобразование Бокса-Кокса. Так как этот метод работает только с положительными значениями, необходимо сдвинуть все значения на константу

In [30]:
```
import matplotlib.pyplot as plt
data.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 [31]:

```
import scipy.stats as stats
import seaborn as sns

for col in data.drop(columns = ['rowid']+object_col).columns:
    if len(data[data[col] <= 0]) > 0:
        data[col] = data[col].sub(data[col].min() - 0.0000001)
    data[col], param = stats.boxcox(data[col])
    print(col, 'Оптимальное значение λ = {}'.format(param))
```

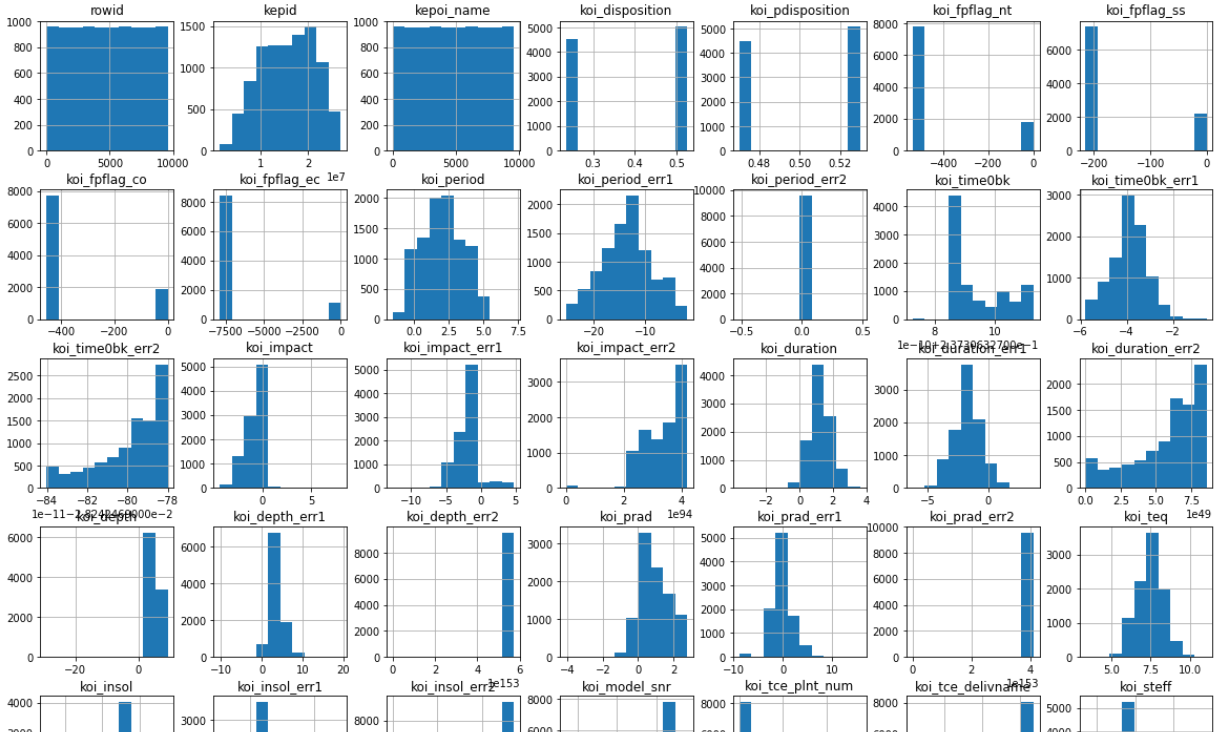
```
kepid Оптимальное значение λ = 1.0476015746957192
koi_fpflag_nt Оптимальное значение λ = -0.31978135073569386
koi_fpflag_ss Оптимальное значение λ = -0.24780480356697157
koi_fpflag_co Оптимальное значение λ = -0.3070874475171977
koi_fpflag_ec Оптимальное значение λ = -0.51581980017193
koi_period Оптимальное значение λ = -0.09124675843700877
koi_period_err1 Оптимальное значение λ = -0.05269924084467775
koi_period_err2 Оптимальное значение λ = 56.68224846736839
koi_time0bk Оптимальное значение λ = -4.213962635898045
koi_time0bk_err1 Оптимальное значение λ = 0.1385847617019599
koi_time0bk_err2 Оптимальное значение λ = 35.40766815561219
koi_impact Оптимальное значение λ = 0.22317263754826602
koi_impact_err1 Оптимальное значение λ = 0.031353188088270416
koi_impact_err2 Оптимальное значение λ = 54.34096947201991
koi_duration Оптимальное значение λ = -0.13011710707428928
koi_duration_err1 Оптимальное значение λ = 0.14374602343138945
koi_duration_err2 Оптимальное значение λ = 39.47670891609475
koi_depth Оптимальное значение λ = -0.0674280905190754
koi_depth_err1 Оптимальное значение λ = 0.05954698028711861
koi_depth_err2 Оптимальное значение λ = 27.766659055000016
koi_prad Оптимальное значение λ = -0.3545556018630248
koi_prad_err1 Оптимальное значение λ = 0.0845922993473487
koi_prad_err2 Оптимальное значение λ = 31.736911460778163
koi_teq Оптимальное значение λ = 0.028622825709666642
/opt/conda/lib/python3.7/site-packages/numpy/core/_methods.py:205: RuntimeWarn
ing: overflow encountered in multiply
  x = um.multiply(x, x, out=x)
/opt/conda/lib/python3.7/site-packages/numpy/core/_methods.py:216: RuntimeWarn
ing: overflow encountered in reduce
  ret = umr_sum(x, axis, dtype, out, keepdims)
```

```
/opt/conda/lib/python3.7/site-packages/numpy/core/_methods.py:205: RuntimeWarn
ing: overflow encountered in multiply
  x = um.multiply(x, x, out=x)
/opt/conda/lib/python3.7/site-packages/numpy/core/_methods.py:216: RuntimeWarn
ing: overflow encountered in reduce
  ret = umr_sum(x, axis, dtype, out, keepdims)
koi_insol Оптимальное значение λ = 0.009951261393936406
koi_insol_err1 Оптимальное значение λ = 0.0758758057353181
koi_insol_err2 Оптимальное значение λ = 22.938604073068987
koi_model_snr Оптимальное значение λ = -0.04095136912484287
koi_tce_plnt_num Оптимальное значение λ = -7.219257450161952
koi_steff Оптимальное значение λ = 0.5844731837985878
koi_steff_err1 Оптимальное значение λ = 0.7282234171583486
koi_slogg Оптимальное значение λ = 6.794280502626673
/opt/conda/lib/python3.7/site-packages/numpy/core/_methods.py:205: RuntimeWarn
ing: overflow encountered in multiply
  x = um.multiply(x, x, out=x)
/opt/conda/lib/python3.7/site-packages/scipy/optimize/optimize.py:2116: Runtim
eWarning: invalid value encountered in double_scalars
  tmp2 = (x - v) * (fx - fw)
/opt/conda/lib/python3.7/site-packages/numpy/core/_methods.py:216: RuntimeWarn
ing: overflow encountered in reduce
  ret = umr_sum(x, axis, dtype, out, keepdims)
koi_slogg_err1 Оптимальное значение λ = 0.12209230283685704
koi_slogg_err2 Оптимальное значение λ = 4.04934312774165
koi_srad Оптимальное значение λ = -0.5430299861550587
koi_srad_err1 Оптимальное значение λ = 0.12328169500452747
koi_srad_err2 Оптимальное значение λ = 74.94270767007492
ra Оптимальное значение λ = 9.285452880367943
dec Оптимальное значение λ = -0.30062277993109815
koi_kepmag Оптимальное значение λ = 3.260941835278969
/opt/conda/lib/python3.7/site-packages/numpy/core/_methods.py:205: RuntimeWarn
ing: overflow encountered in multiply
  x = um.multiply(x, x, out=x)
/opt/conda/lib/python3.7/site-packages/scipy/optimize/optimize.py:2116: Runtim
eWarning: invalid value encountered in double_scalars
  tmp2 = (x - v) * (fx - fw)
/opt/conda/lib/python3.7/site-packages/numpy/core/_methods.py:216: RuntimeWarn
ing: overflow encountered in reduce
  ret = umr_sum(x, axis, dtype, out, keepdims)
```

In [32]:

```
data.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 получили датасет без пропусков с нормализванными значениями

In [ ]: