

Лабораторная работа 0-1 по курсу "Машинное обучение"

Студент Цыкин

Группа 301

О датасете

Миссия "Кеплер" предоставила точные данные для проведения комплексного астеросейсмического анализа эволюционировавших звезд. В этой работе мы систематически характеризуем подобные солнцу колебания и грануляцию для 16094 колеблющихся красных гигантов, используя данные о длительной частоте в конце миссии. Благодаря большому образцу красных гигантов мы подтверждаем, что звезды с ветвью красного гиганта (RGB) и звезды с гелиевым ядром (HeB) в совокупности отличаются распределением амплитуды колебаний, мощностью грануляции и шириной превышения мощности, что в основном связано с разницей масс.

ссылка: <https://www.kaggle.com/datasets/fernandolima23/asteroseismology-of-16000-kepler-red-giants>

Содержание

Kp_{mag} - видимая величина Кеплера

numax - Частота максимальной мощности колебаний

e_numax - Неопределенность

Delnu - Разделение акустических состояний по средней частоте

e_Delnu - Неопределенность in Delnu

A - Амплитуда колебаний в радиальном режиме

e_A - Неопределенность in A

Width - Ширина превышения мощности

e_Width - Неопределенность в ширине

Teff - Эффективная температура

e_Teff - Неопределенность в Teff

log(g) - Логарифмическая сила тяжести поверхности из этой работы

e_log(g) - Неопределенность в логарифме(g)

Fe/H - Металличность

e_Fe/H - Неопределенность в Fe/H

Gran - Мощность грануляции

e_Gran - Неопределенность in Gran

NoCorM - Корректирующее массовое решение

e_NoCorM - Неопределенность in NoCorM

NoCorR - Правка радиуса коррекции

e_NoCorR - Неопределенность in NoCorR

RGBcorM - RGB скорректированное массовое решение

e_RGBcorM - Неопределенность in RGBcorM

RGBcorR - GB решение с исправленным радиусом

e_RGBcorR - Неопределенность in RGBcorR

CICorM - скорректированный массовый раствор

e_CICorM - Неопределенность in CICorM

CICorR - Решение с исправленным радиусом

e_CICorR - Неопределенность в CICorR

Phase - Эволюционная фаза

Фазы следующим образом:

2 = Фаза HeB;

1 = Фаза RGB;

0 = неклассифицированная фаза

RGB (Red-Giant-Branch)

HeB (Helium Burning)

Загрузка данных

```
In [1]: #Подключение библиотек  
import numpy as np  
import pandas as pd  
import matplotlib.pyplot as plt  
import seaborn as sns  
sns.__version__
```

```
Out[1]: '0.11.2'
```

```
In [2]: df = pd.read_csv('Data_classified_phase_.csv')
df.shape
```

```
Out[2]: (15388, 28)
```

```
In [3]: df_un = pd.read_csv('Data_unclassified_phase_.csv')
df_un.shape
```

```
Out[3]: (706, 28)
```

```
In [4]: df.head(5)
```

```
Out[4]:
```

	Kpmag	numax	e_numax	Delnu	e_Delnu	A	e_A	Width	e_Width	Teff	...	e_No
0	9.20	29.99	0.60	3.399	0.011	104.9	4.6	12.3	1.5	4751	...	
1	13.23	29.48	0.48	3.962	0.116	149.7	8.3	12.0	3.9	5188	...	
2	12.58	41.39	0.54	4.311	0.013	86.1	4.6	15.3	1.6	4728	...	
3	12.14	41.17	0.90	4.414	0.061	63.8	2.9	24.8	2.7	5072	...	
4	11.74	36.91	0.71	3.991	0.064	116.1	9.6	14.0	2.4	4718	...	

5 rows × 28 columns

```
In [5]: df.columns
```

```
Out[5]: Index(['Kpmag', 'numax', 'e_numax', 'Delnu', 'e_Delnu', 'A', 'e_A', 'Width',
              'e_Width', 'Teff', 'e_Teff', 'log(g)', 'e_log(g)', '[Fe/H]', 'e_[Fe/H]',
              'NoCorM', 'e_NoCorM', 'NoCorR', 'e_NoCorR', 'RGBcorM', 'e_RGBcorM',
              'RGBcorR', 'e_RGBcorR', 'ClcorM', 'e_ClcorM', 'ClcorR', 'e_ClcorR',
              'Phase'],
              dtype='object')
```

```
In [6]: df_un.columns
```

```
Out[6]: Index(['Kpmag', 'numax', 'e_numax', 'Delnu', 'e_Delnu', 'A', 'e_A', 'Width',
              'e_Width', 'Teff', 'e_Teff', 'log(g)', 'e_log(g)', '[Fe/H]', 'e_[Fe/H]',
              'NoCorM', 'e_NoCorM', 'NoCorR', 'e_NoCorR', 'RGBcorM', 'e_RGBcorM',
              'RGBcorR', 'e_RGBcorR', 'ClcorM', 'e_ClcorM', 'ClcorR', 'e_ClcorR',
              'Phase'],
              dtype='object')
```

```
In [7]: df.index
```

```
Out[7]: RangeIndex(start=0, stop=15388, step=1)
```

```
In [8]: df_un.index
```

```
Out[8]: RangeIndex(start=0, stop=706, step=1)
```

```
In [9]: df.dtypes
```

```
Out[9]: Kpmag          float64
numax          float64
e_numax        float64
Delnu          float64
e_Delnu        float64
A              object
e_A            object
Width          object
e_Width        object
Teff           int64
e_Teff         int64
log(g)         float64
e_log(g)       float64
[Fe/H]         float64
e_[Fe/H]       float64
NoCorM         float64
e_NoCorM       float64
NoCorR         float64
e_NoCorR       float64
RGBcorM        float64
e_RGBcorM      float64
RGBcorR        float64
e_RGBcorR      float64
ClcorM         float64
e_ClcorM       float64
ClcorR         float64
e_ClcorR       float64
Phase          int64
dtype: object
```

```
In [10]: df_un.dtypes
```

```
Out[10]: Kpmag          float64
numax          float64
e_numax        float64
Delnu          float64
e_Delnu        float64
A              float64
e_A            float64
Width          float64
e_Width        float64
Teff           int64
e_Teff         int64
log(g)         float64
e_log(g)       float64
[Fe/H]         float64
e_[Fe/H]       float64
NoCorM         float64
e_NoCorM       float64
NoCorR         float64
e_NoCorR       float64
RGBcorM        float64
e_RGBcorM      float64
RGBcorR        float64
e_RGBcorR      float64
ClcorM         float64
```

```
e_ClcorM      float64
ClcorR        float64
e_ClcorR      float64
Phase         int64
dtype: object
```

Все параметры в представленном датасете - численные

Подготовка данных

Как видно, что некоторый тип данных на тестовых и тренировочных выборках не схожи. Необходимо перевести к одному типу. Удобнее Object => float64

```
In [11]: arr = []
         for column in df.columns:
             if df.dtypes[column] == 'object':
                 arr.append(column)
         print(arr)
```

```
['A', 'e_A', 'Width', 'e_Width']
```

Убрать пробелы данных на nan

```
In [12]: for i in range(1, 20):
         df[df == i*' '] = np.nan
         df_un[df == i*' '] = np.nan
```

```
In [13]: for i in arr:
         df[i] = pd.to_numeric(df[i])
```

Результат после преобразования типов

```
In [14]: df.dtypes
```

```
Out[14]: Kpmag      float64
         numax      float64
         e_numax    float64
         Delnu      float64
         e_Delnu    float64
         A          float64
         e_A        float64
         Width      float64
         e_Width    float64
         Teff       int64
         e_Teff     int64
         log(g)     float64
         e_log(g)   float64
         [Fe/H]     float64
         e_[Fe/H]   float64
         NoCorM     float64
         e_NoCorM   float64
         NoCorR     float64
         e_NoCorR   float64
         RGBcorM    float64
         e_RGBcorM  float64
         RGBcorR    float64
         e_RGBcorR  float64
         ClcorM     float64
```

```
e_ClcorM      float64
ClcorR        float64
e_ClcorR      float64
Phase         int64
dtype: object
```

In [15]: `df.describe()`

	Kpmag	numax	e_numax	Delnu	e_Delnu	A
count	15388.000000	15388.000000	15388.000000	15388.000000	15388.000000	14851.000000
mean	12.632402	64.673195	0.903276	6.16765	0.047327	95.059174
std	1.211778	50.958690	0.891080	3.56072	0.069763	51.897428
min	6.240000	3.970000	0.040000	0.74700	0.008000	10.700000
25%	11.900000	31.680000	0.560000	3.97675	0.017000	59.900000
50%	12.850000	42.260000	0.720000	4.54050	0.028000	90.800000
75%	13.510000	79.522500	1.000000	7.27000	0.050000	120.900000
max	16.850000	273.160000	70.270000	19.29400	1.338000	745.200000

8 rows × 7 columns

Работаем с потерями

Здесь мы видим, какие данные были утеряны. Это необходимо исправить

In [16]:

```
#Для df
for i in df.columns:
    a = df[i].isnull().sum()
    if a > 0:
        print(i, '-', a)
print('end')
```

```
A - 537
e_A - 537
Width - 537
e_Width - 537
end
```

In [17]:

```
for i in df_un.columns:
    a = df_un[i].isnull().sum()
    if a > 0:
        print(i, '-', a)
print('end')
```

end

Выделим подтаблицу с параметрами в которых нет некоторых значений

In [18]:

```
df_nan = df[df.isna().any(axis=1)]
df_nan
```

	Kpmag	numax	e_numax	Delnu	e_Delnu	A	e_A	Width	e_Width	Teff	...	e
33	13.537	234.61	3.17	16.733	0.070	NaN	NaN	NaN	NaN	5018	...	e
69	12.931	212.06	2.91	15.984	0.035	NaN	NaN	NaN	NaN	4979	...	e

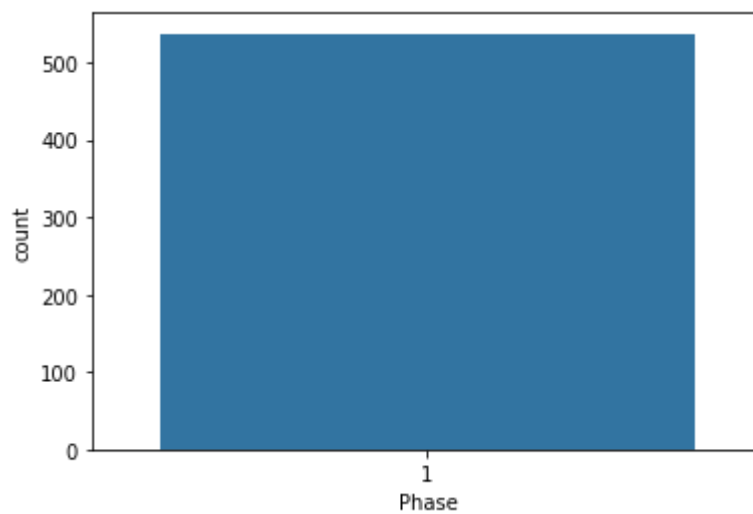
102	12.688	215.81	3.19	16.108	0.023	NaN	NaN	NaN	NaN	5008	...
124	12.381	224.38	1.56	16.205	0.041	NaN	NaN	NaN	NaN	4908	...
222	13.875	225.74	2.64	17.622	0.050	NaN	NaN	NaN	NaN	4893	...
...
15207	13.109	200.05	2.28	15.573	0.056	NaN	NaN	NaN	NaN	5035	...
15239	13.842	259.05	9.97	18.603	0.104	NaN	NaN	NaN	NaN	5080	...
15259	13.785	211.95	1.26	16.658	0.048	NaN	NaN	NaN	NaN	5098	...
15351	13.227	214.32	1.73	16.808	0.045	NaN	NaN	NaN	NaN	5049	...
15379	13.785	249.27	4.08	19.004	0.128	NaN	NaN	NaN	NaN	4996	...

537 rows × 28 columns

```
In [19]: df_nan.shape
```

```
Out[19]: (537, 28)
```

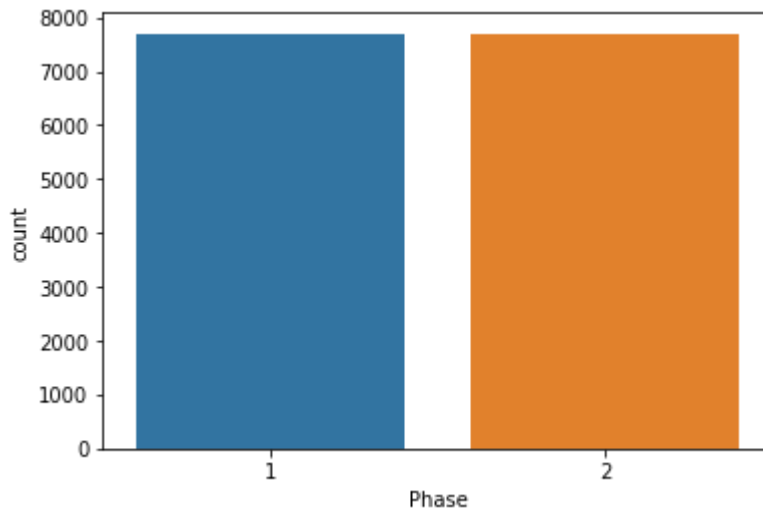
```
In [20]: sns.countplot(x='Phase', data=df_nan)
pass
```



Как видно, все выпавшие варианты представляют 1 фазу

Выведем соотношение числа фаз

```
In [21]: sns.countplot(x='Phase', data=df)
pass
```



```
In [22]: df_new = df.copy()
df_new["A"] = df_new["A"].fillna(df_new["e_A"].median())
df_new["e_A"] = df_new["e_A"].fillna(df_new["e_Width"].median())
df_new["Width"] = df_new["Width"].fillna(df_new["e_A"].median())
df_new["e_Width"] = df_new["e_Width"].fillna(df_new["e_Width"].median())
print(df.shape, df_new.shape)
```

```
(15388, 28) (15388, 28)
```

```
In [23]: f, axes = plt.subplots(2, 2, figsize = (17,10))
sns.histplot(df['A'], ax = axes[0][0])
sns.histplot(df_new['A'], ax = axes[0][1])
sns.boxplot(df['A'], ax = axes[1][0])
sns.boxplot(df_new['A'], ax = axes[1][1])
```

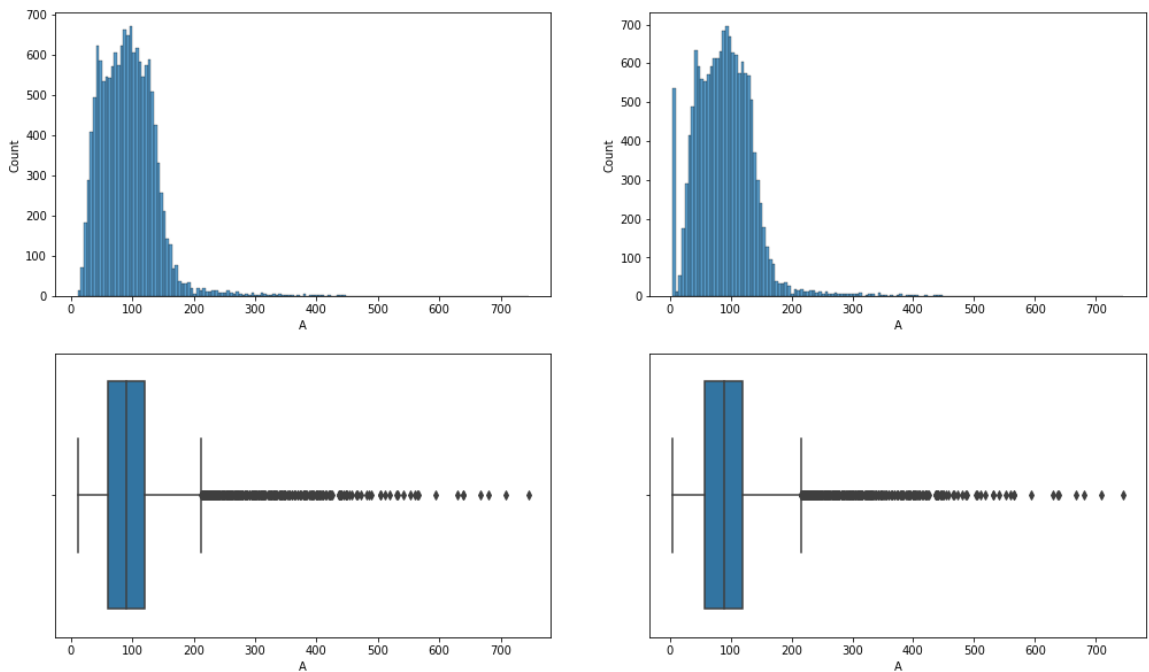
C:\Users\itsyk\Python\lib\site-packages\seaborn_decorators.py:36: Future Warning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.

```
warnings.warn(
```

C:\Users\itsyk\Python\lib\site-packages\seaborn_decorators.py:36: Future Warning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.

```
warnings.warn(
```

```
Out[23]: <AxesSubplot:xlabel='A'>
```

In [24]:

```
f, axes = plt.subplots(2, 2, figsize = (17,10))
sns.histplot(df['e_A'], ax = axes[0][0])
sns.histplot(df_new['e_A'], ax = axes[0][1])
sns.boxplot(df['e_A'], ax = axes[1][0])
sns.boxplot(df_new['e_A'], ax = axes[1][1])
```

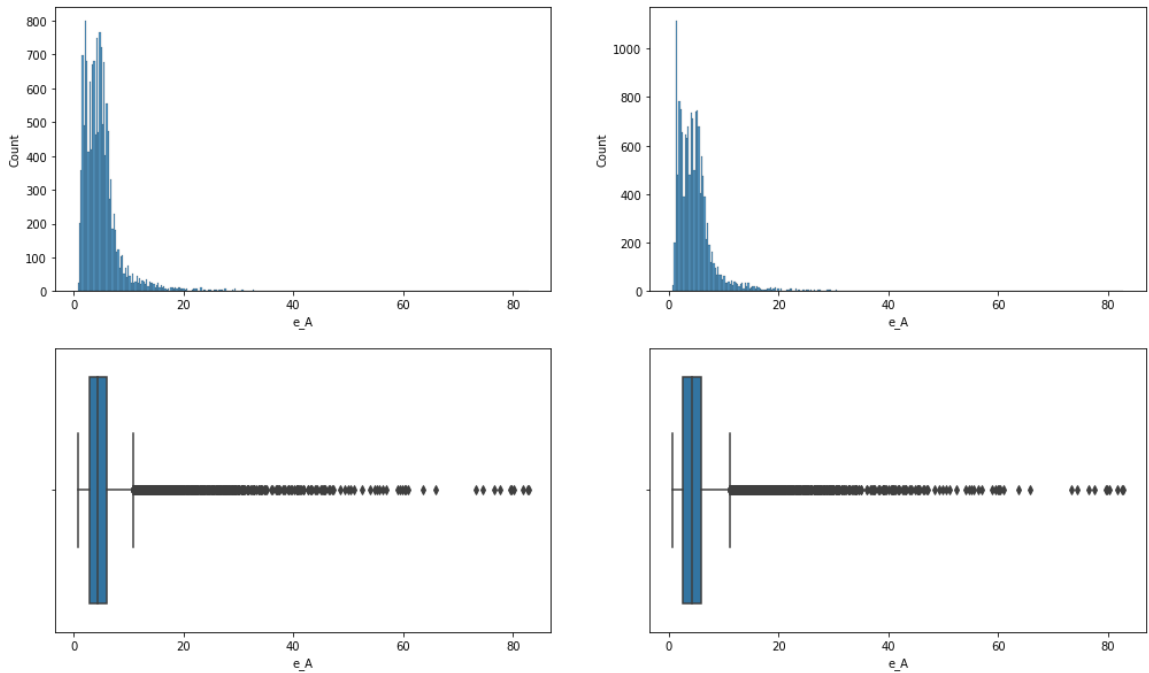
C:\Users\itsyk\Python\lib\site-packages\seaborn_decorators.py:36: Future Warning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.

warnings.warn(

C:\Users\itsyk\Python\lib\site-packages\seaborn_decorators.py:36: Future Warning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.

warnings.warn(

Out[24]: <AxesSubplot:xlabel='e_A'>



In [25]:

```
f, axes = plt.subplots(2, 2, figsize = (17,10))
sns.histplot(df['Width'], ax = axes[0][0])
sns.histplot(df_new['Width'], ax = axes[0][1])
sns.boxplot(df['Width'], ax = axes[1][0])
sns.boxplot(df_new['Width'], ax = axes[1][1])
```

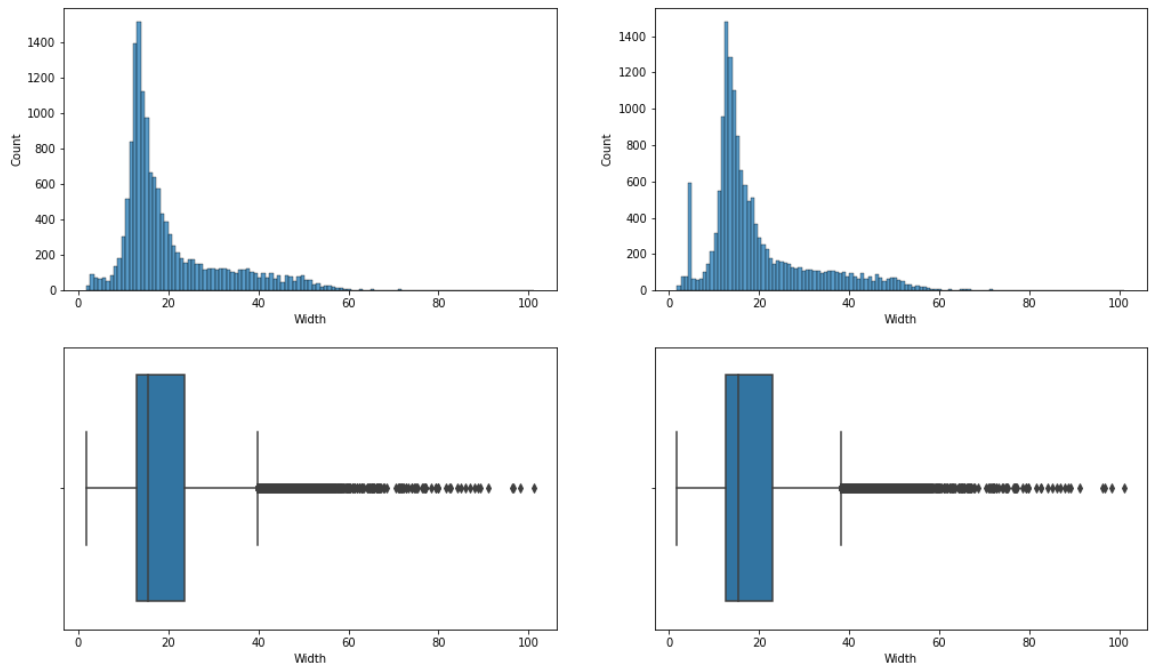
C:\Users\itsyk\Python\lib\site-packages\seaborn_decorators.py:36: Future Warning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.

warnings.warn(

C:\Users\itsyk\Python\lib\site-packages\seaborn_decorators.py:36: Future Warning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.

warnings.warn(

Out[25]: <AxesSubplot:xlabel='Width'>



In [26]:

```
f, axes = plt.subplots(2, 2, figsize = (17,10))
sns.histplot(df['e_Width'], ax = axes[0][0])
sns.histplot(df_new['e_Width'], ax = axes[0][1])
sns.boxplot(df['e_Width'], ax = axes[1][0])
sns.boxplot(df_new['e_Width'], ax = axes[1][1])
```

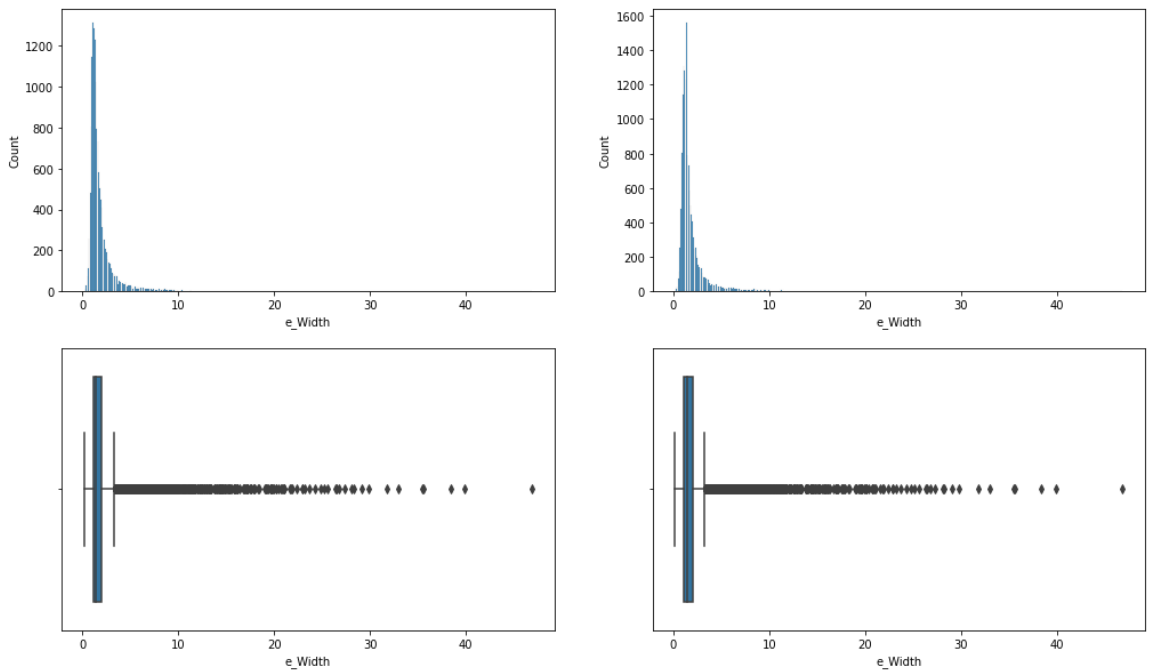
C:\Users\itsyk\Python\lib\site-packages\seaborn_decorators.py:36: Future Warning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.

warnings.warn(

C:\Users\itsyk\Python\lib\site-packages\seaborn_decorators.py:36: Future Warning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.

warnings.warn(

Out[26]: <AxesSubplot:xlabel='e_Width'>



Здесь я попробывал вместо nap значений написать усредненные, но некоторые знаения сильна выбиваются. Думаю, что лучше просто от них отказаться, так как данных достаточно.

```
In [27]: df = df[df['A'].notna()]
df.shape
```

```
Out[27]: (14851, 28)
```

```
In [28]: for i in df.columns:
a = df[i].isnull().sum()
if a > 0:
    print(i, '-', a)
print('end')
```

```
end
```

```
In [29]: df.head()
```

```
Out[29]:
```

	Kpmag	numax	e_numax	Delnu	e_Delnu	A	e_A	Width	e_Width	Teff	...	e_No
0	9.20	29.99	0.60	3.399	0.011	104.9	4.6	12.3	1.5	4751	...	
1	13.23	29.48	0.48	3.962	0.116	149.7	8.3	12.0	3.9	5188	...	
2	12.58	41.39	0.54	4.311	0.013	86.1	4.6	15.3	1.6	4728	...	
3	12.14	41.17	0.90	4.414	0.061	63.8	2.9	24.8	2.7	5072	...	
4	11.74	36.91	0.71	3.991	0.064	116.1	9.6	14.0	2.4	4718	...	

5 rows × 28 columns

```
In [30]: df.describe()
```

```
Out[30]:
```

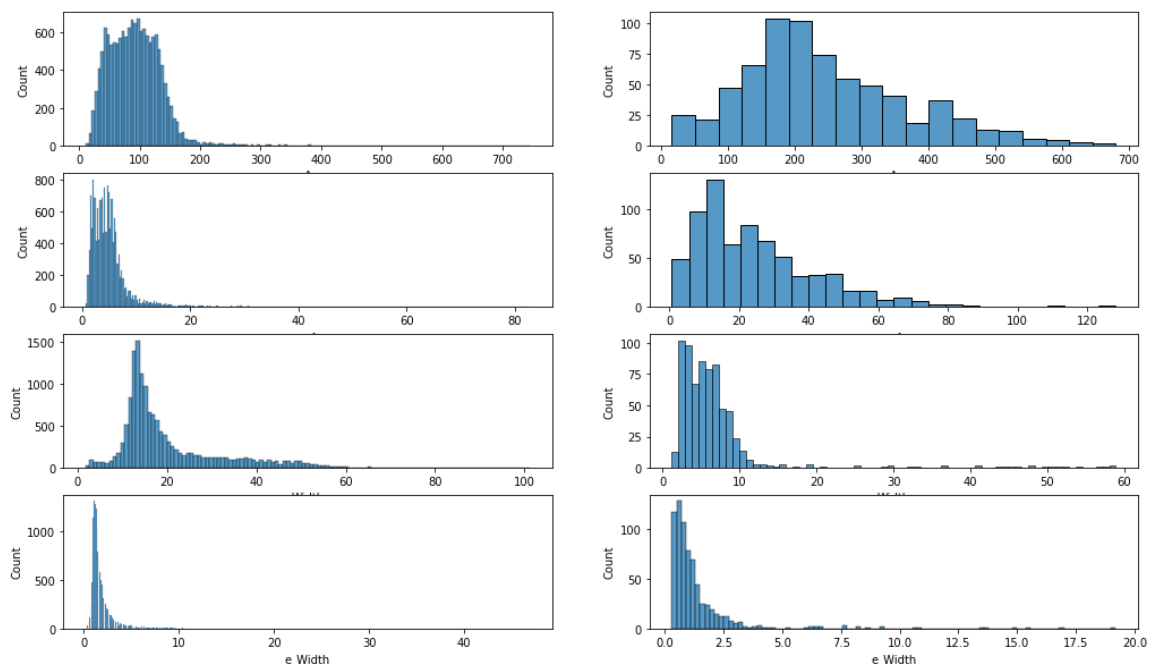
	Kpmag	numax	e_numax	Delnu	e_Delnu	A
--	-------	-------	---------	-------	---------	---

count	14851.000000	14851.000000	14851.000000	14851.000000	14851.000000	14851.000000
mean	12.619928	58.995670	0.856245	5.783544	0.047270	95.059174
std	1.216108	41.937888	0.613275	2.979246	0.070172	51.897428
min	6.240000	3.970000	0.040000	0.747000	0.008000	10.700000
25%	11.880000	31.500000	0.550000	3.961000	0.016000	59.900000
50%	12.840000	40.960000	0.710000	4.457000	0.027000	90.800000
75%	13.500000	72.070000	0.960000	6.711500	0.050000	120.900000
max	16.850000	200.000000	25.370000	16.745000	1.338000	745.200000

8 rows × 28 columns

```
In [31]: f, axes = plt.subplots(4, 2, figsize = (17,10))
sns.histplot(df['A'], ax = axes[0][0])
sns.histplot(df_un['A'], ax = axes[0][1])
sns.histplot(df['e_A'], ax = axes[1][0])
sns.histplot(df_un['e_A'], ax = axes[1][1])
sns.histplot(df['Width'], ax = axes[2][0])
sns.histplot(df_un['Width'], ax = axes[2][1])
sns.histplot(df['e_Width'], ax = axes[3][0])
sns.histplot(df_un['e_Width'], ax = axes[3][1])
```

Out[31]: <AxesSubplot:xlabel='e_Width', ylabel='Count'>

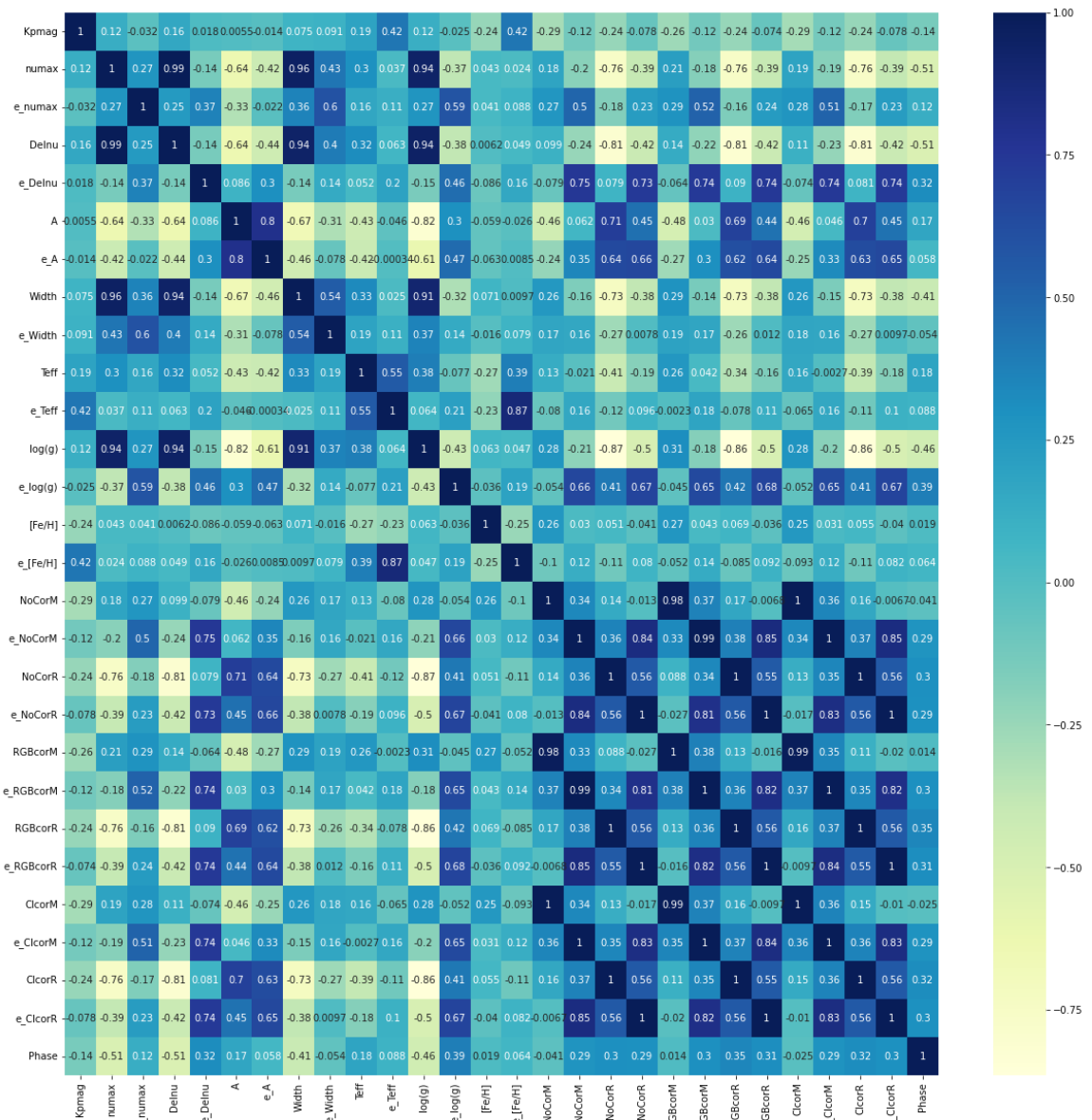


Как можно отметить, данных достаточно, чтобы определить фазы на неопределенных данных

Матрица корреляции

```
In [32]: plt.figure(figsize = (20, 20))
corr_matrix = df.corr()
sns.heatmap(corr_matrix, annot=True, cmap="YlGnBu")
```

Out[32]: <AxesSubplot:>



```
In [33]: df_corr = df.corr().unstack().reset_index()
df_corr
```

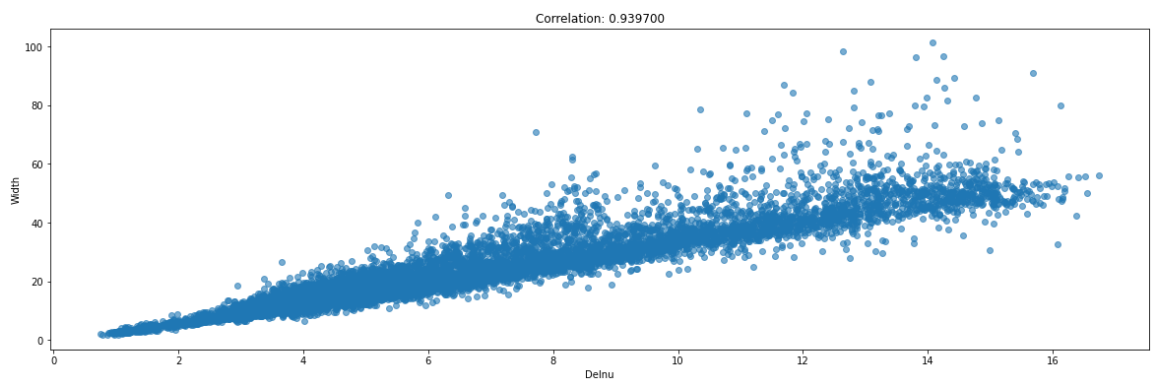
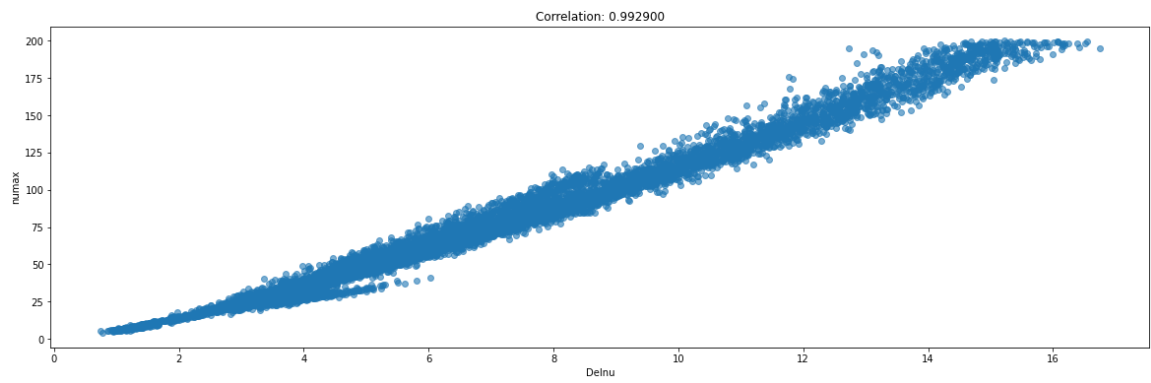
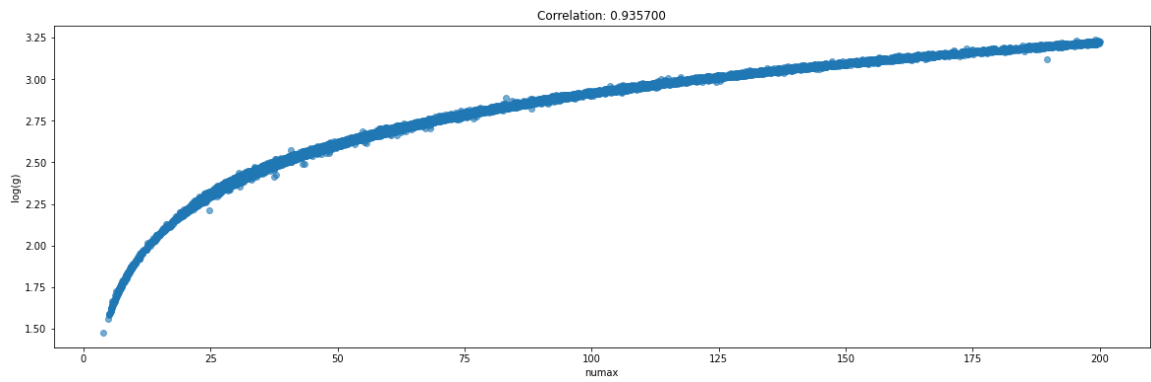
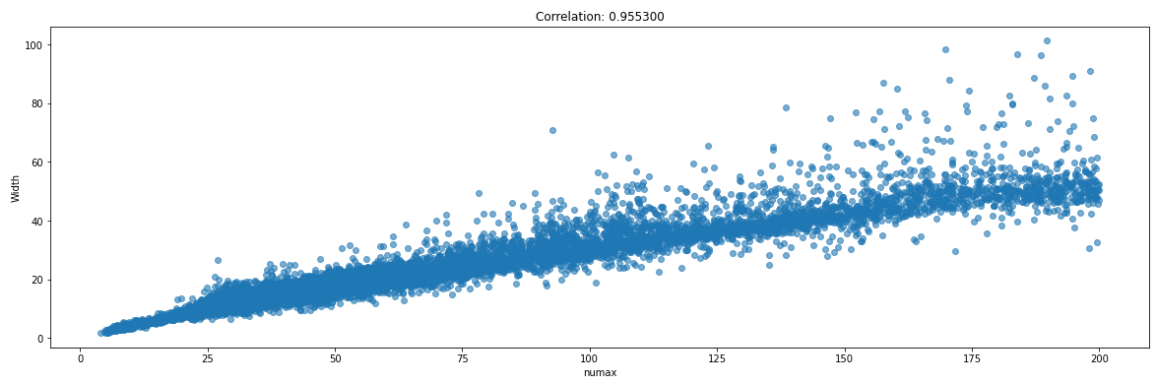
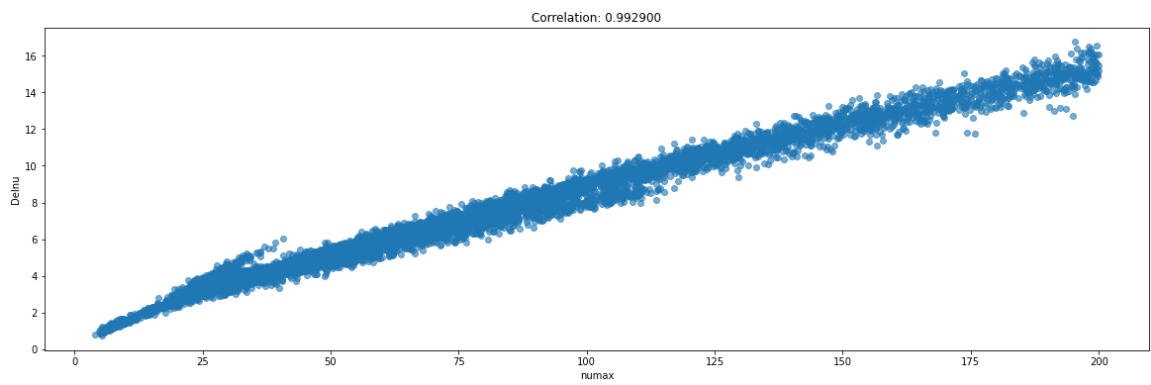
	level_0	level_1	0
0	Kpmag	Kpmag	1.000000
1	Kpmag	numax	0.118883
2	Kpmag	e_numax	-0.031973
3	Kpmag	Delnu	0.158125
4	Kpmag	e_Delnu	0.018239
...
779	Phase	ClcorM	-0.024521
780	Phase	e_ClcorM	0.288437
781	Phase	ClcorR	0.315257
782	Phase	e_ClcorR	0.299135
783	Phase	Phase	1.000000

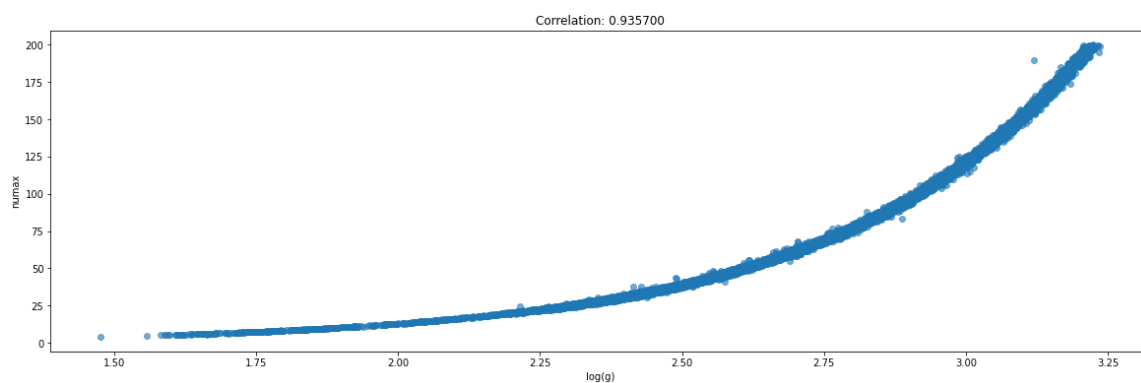
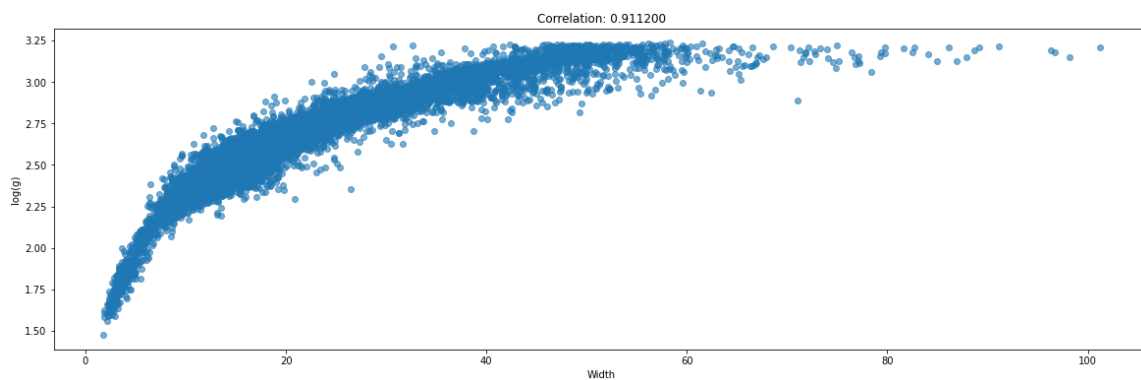
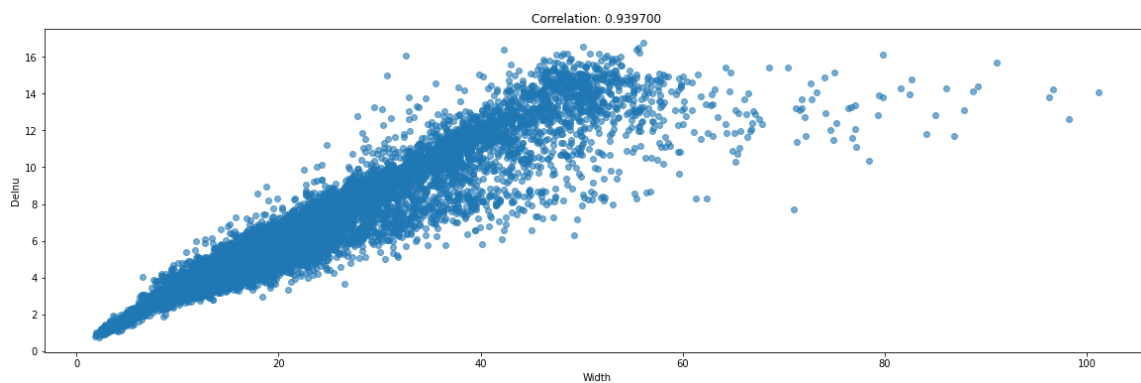
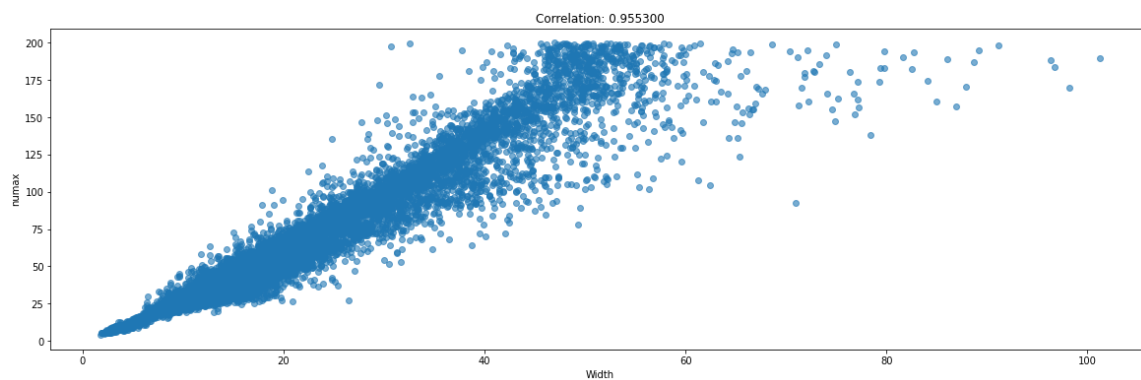
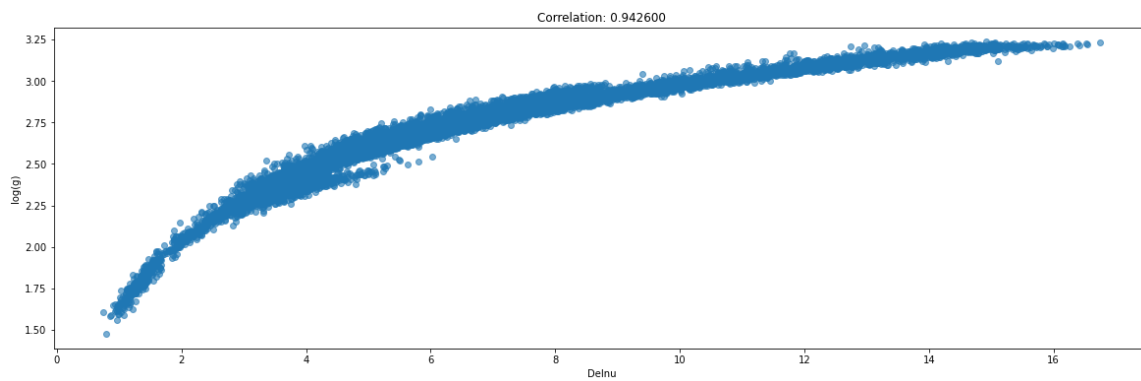
784 rows × 3 columns

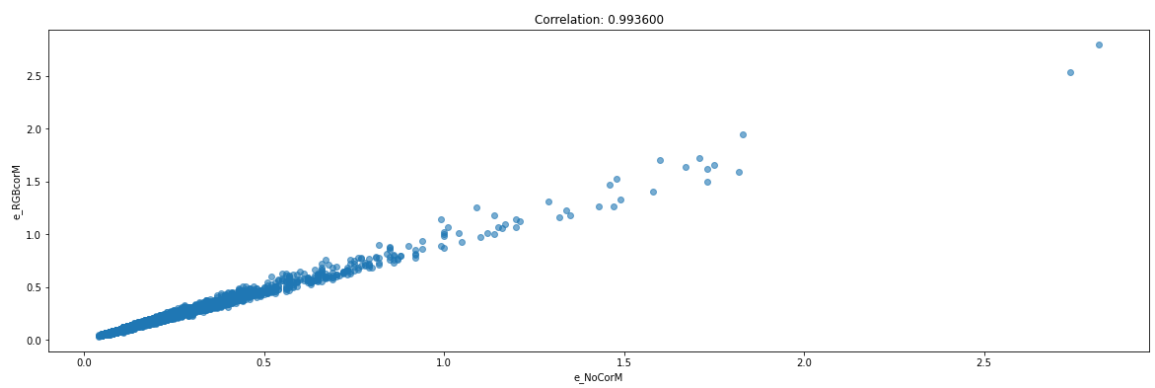
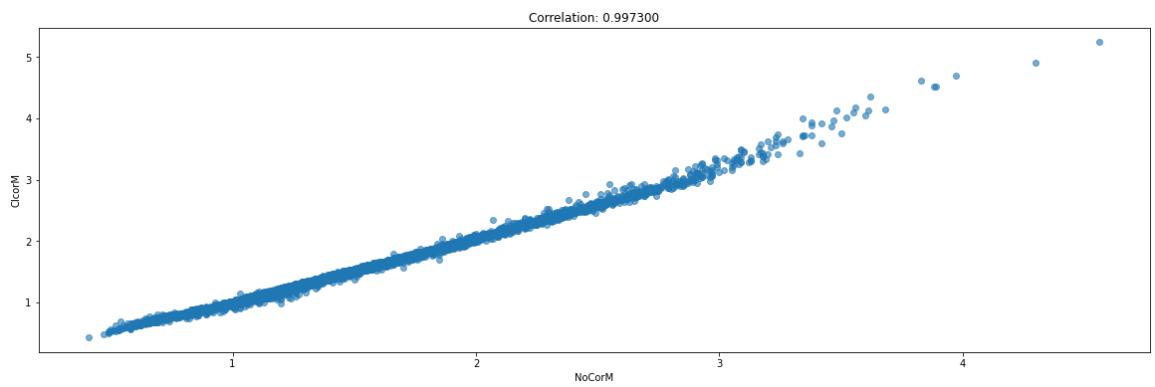
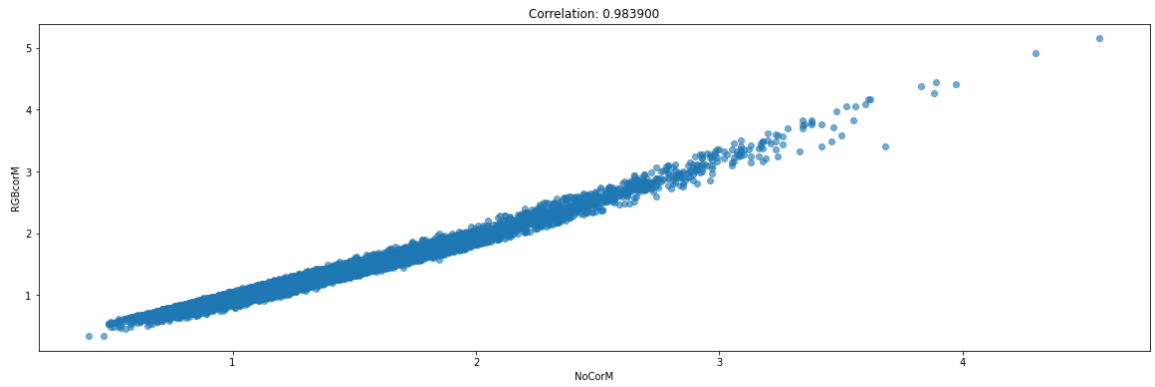
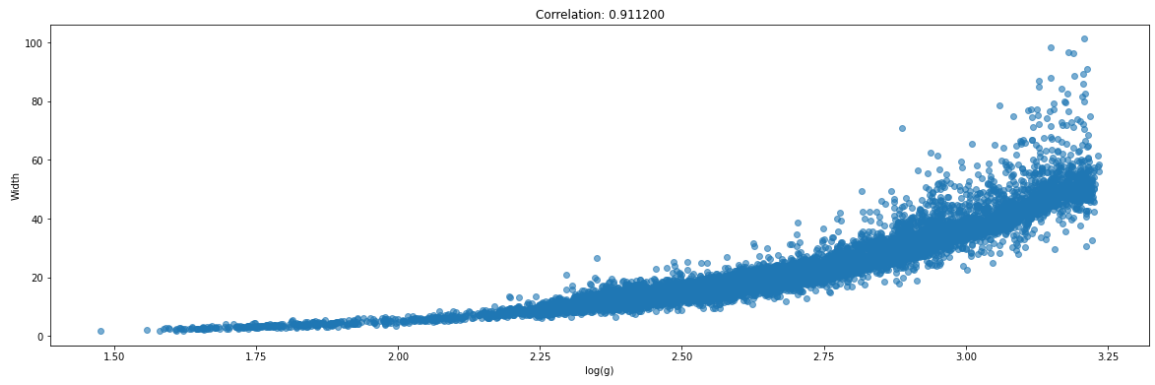
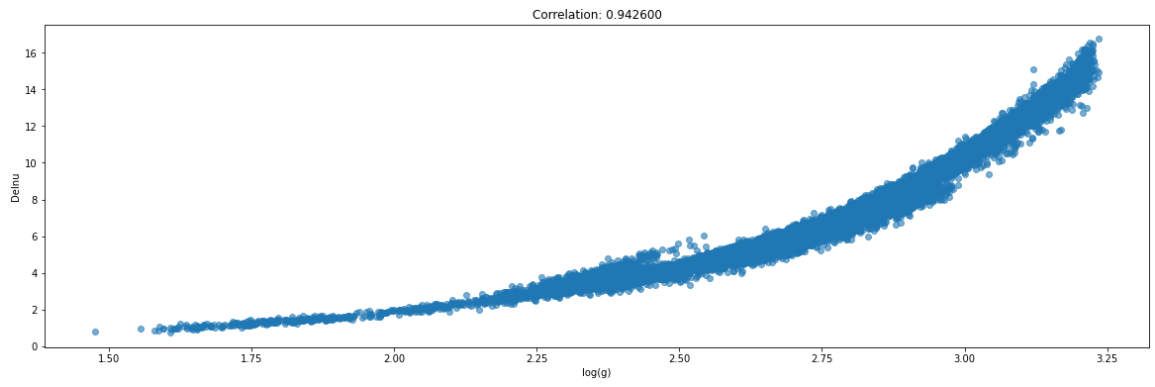
```
In [34]: high_corr_features = []
         for i in range(len(df_corr)):
             if df_corr[0][i] > 0.9 and df_corr['level_0'][i] != df_corr['level_1']:
                 high_corr_features.append([df_corr['level_0'][i], df_corr['level_1']])
         high_corr_features
```

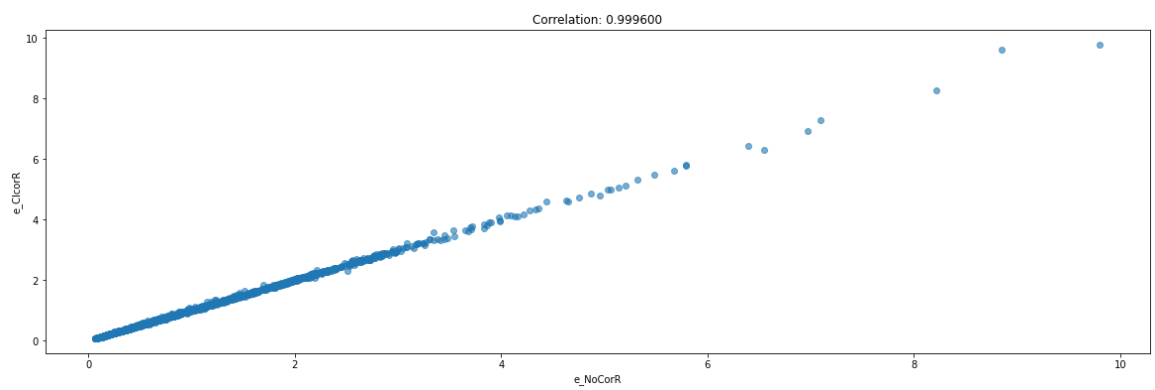
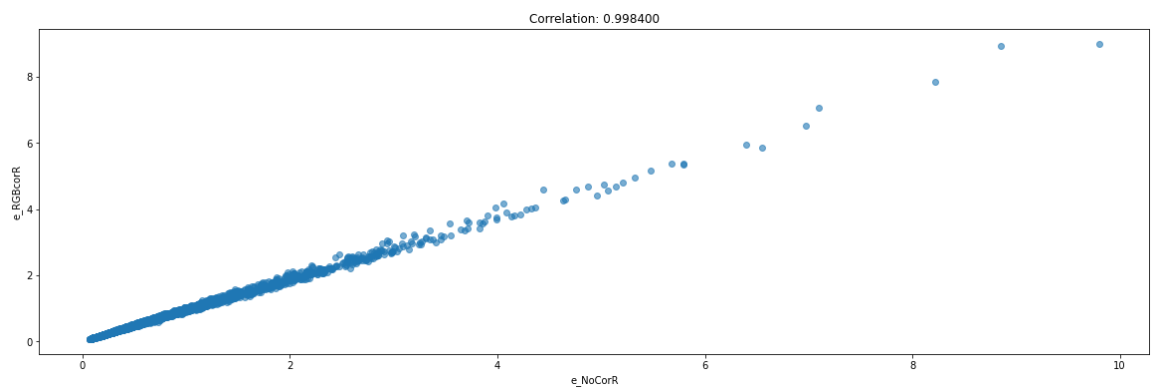
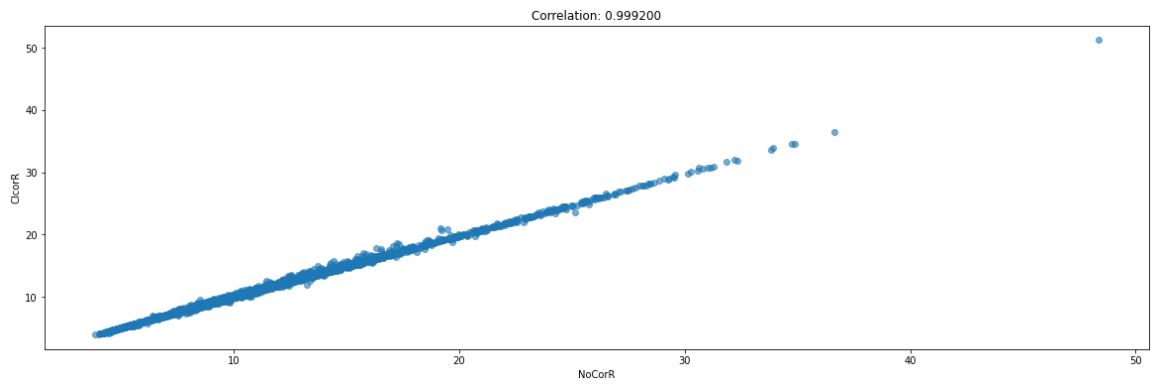
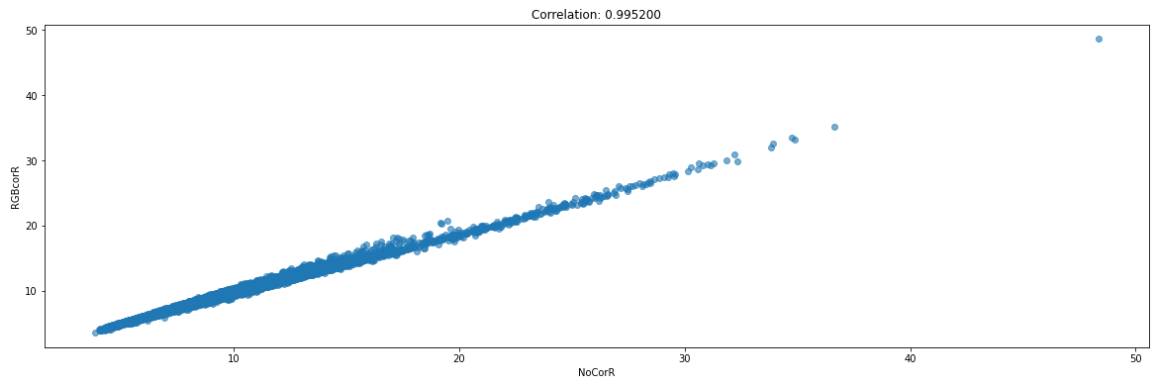
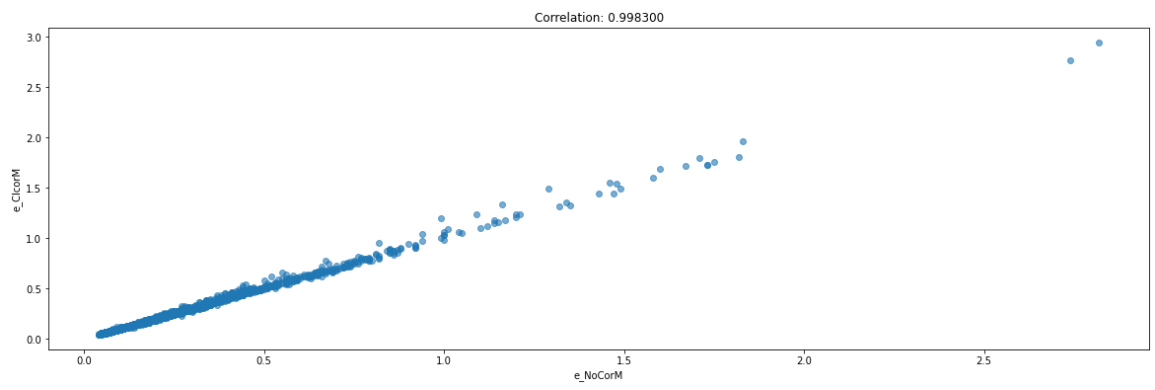
```
Out[34]: [['numax', 'Delnu', 0.9929377587760945],
          ['numax', 'Width', 0.9552979711189871],
          ['numax', 'log(g)', 0.935695230273852],
          ['Delnu', 'numax', 0.9929377587760945],
          ['Delnu', 'Width', 0.9396805418304727],
          ['Delnu', 'log(g)', 0.942644750007283],
          ['Width', 'numax', 0.9552979711189871],
          ['Width', 'Delnu', 0.9396805418304727],
          ['Width', 'log(g)', 0.9112383054900803],
          ['log(g)', 'numax', 0.935695230273852],
          ['log(g)', 'Delnu', 0.942644750007283],
          ['log(g)', 'Width', 0.9112383054900803],
          ['NoCorM', 'RGBcorM', 0.9839076557216867],
          ['NoCorM', 'ClcorM', 0.9972906059411837],
          ['e_NoCorM', 'e_RGBcorM', 0.9935568667387731],
          ['e_NoCorM', 'e_ClcorM', 0.9982558829325169],
          ['NoCorR', 'RGBcorR', 0.995156784951965],
          ['NoCorR', 'ClcorR', 0.9992449435255716],
          ['e_NoCorR', 'e_RGBcorR', 0.998362812237733],
          ['e_NoCorR', 'e_ClcorR', 0.9996361917355046],
          ['RGBcorM', 'NoCorM', 0.9839076557216867],
          ['RGBcorM', 'ClcorM', 0.9899052085865482],
          ['e_RGBcorM', 'e_NoCorM', 0.9935568667387731],
          ['e_RGBcorM', 'e_ClcorM', 0.99623587429077],
          ['RGBcorR', 'NoCorR', 0.995156784951965],
          ['RGBcorR', 'ClcorR', 0.9969752898077152],
          ['e_RGBcorR', 'e_NoCorR', 0.998362812237733],
          ['e_RGBcorR', 'e_ClcorR', 0.9990123064794945],
          ['ClcorM', 'NoCorM', 0.9972906059411837],
          ['ClcorM', 'RGBcorM', 0.9899052085865482],
          ['e_ClcorM', 'e_NoCorM', 0.9982558829325169],
          ['e_ClcorM', 'e_RGBcorM', 0.99623587429077],
          ['ClcorR', 'NoCorR', 0.9992449435255716],
          ['ClcorR', 'RGBcorR', 0.9969752898077152],
          ['e_ClcorR', 'e_NoCorR', 0.9996361917355046],
          ['e_ClcorR', 'e_RGBcorR', 0.9990123064794945]]
```

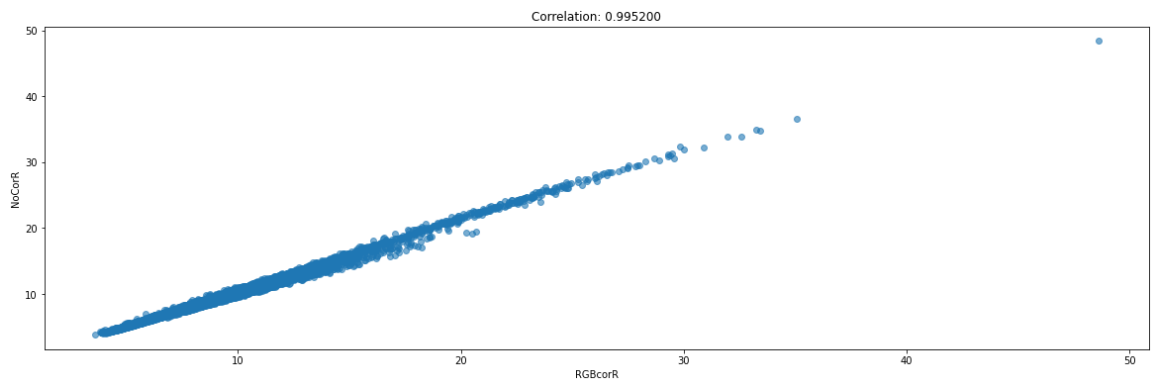
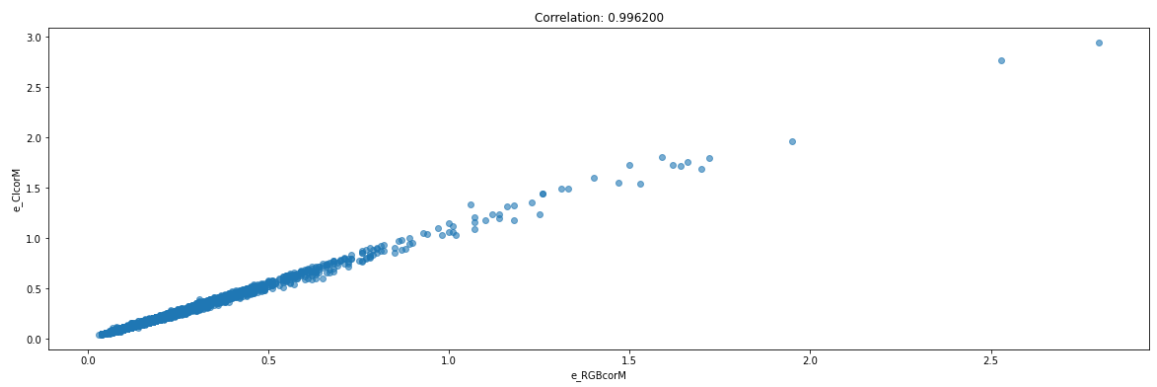
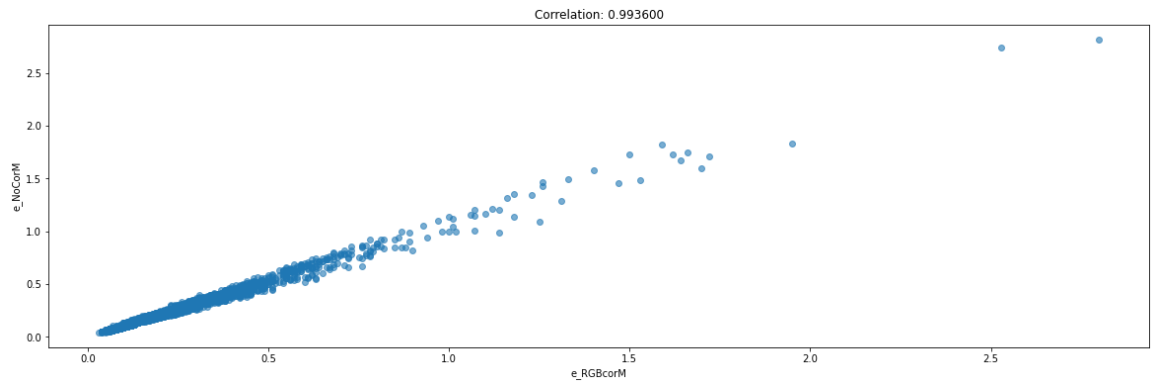
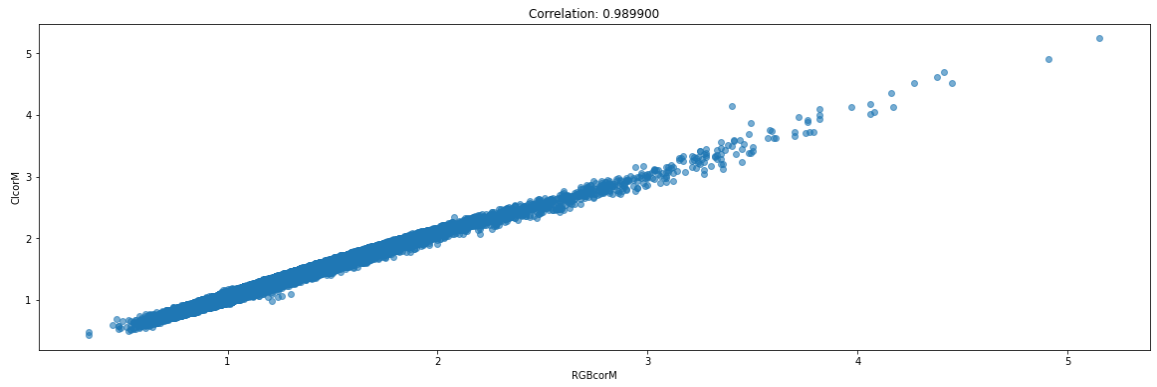
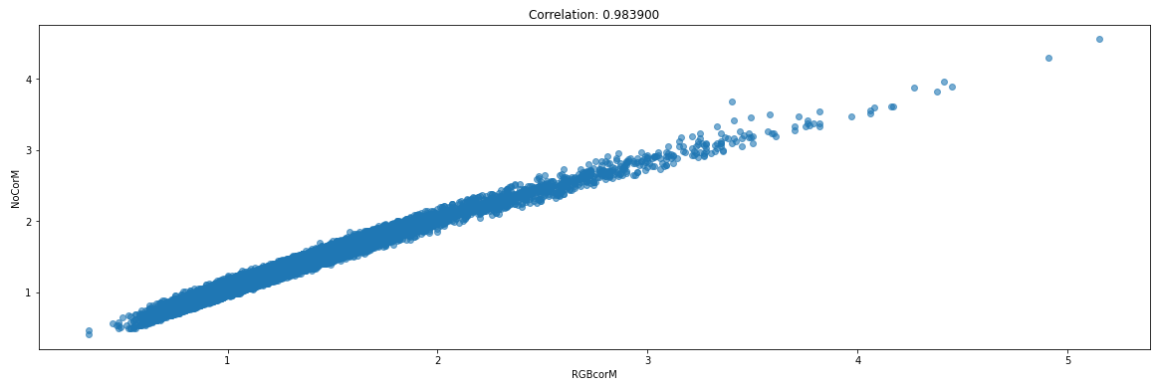
```
In [35]: for i in high_corr_features:
         plt.figure(figsize = (20, 6))
         plt.plot(df[i[0]], df[i[1]], 'o', alpha = 0.6)
         plt.title('Correlation: %f' % round(i[2], 4))
         plt.xlabel(i[0])
         plt.ylabel(i[1])
         plt.show()
```

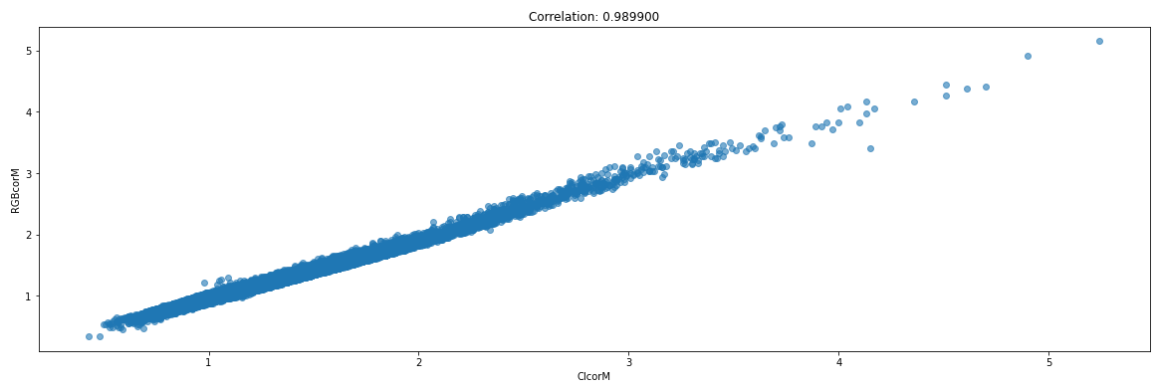
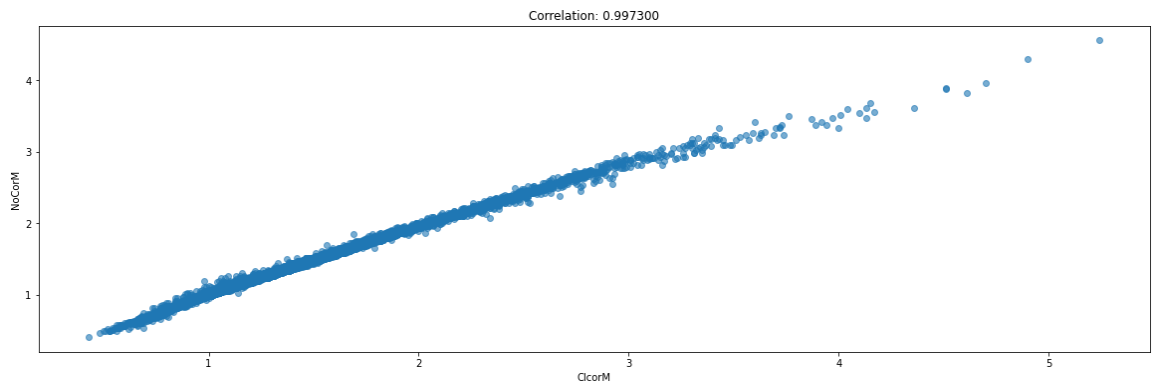
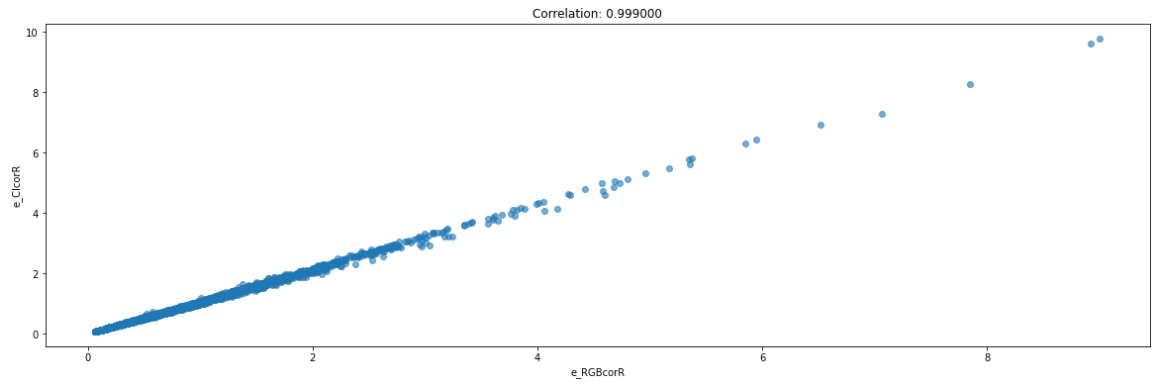
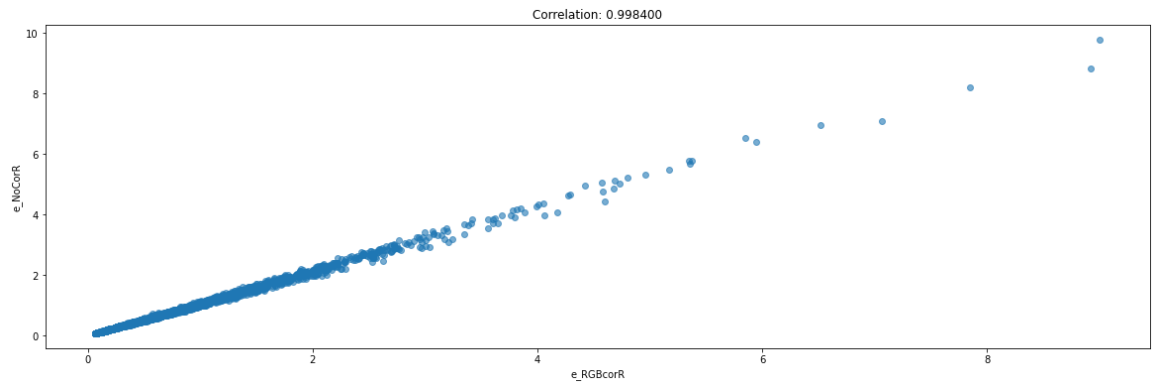
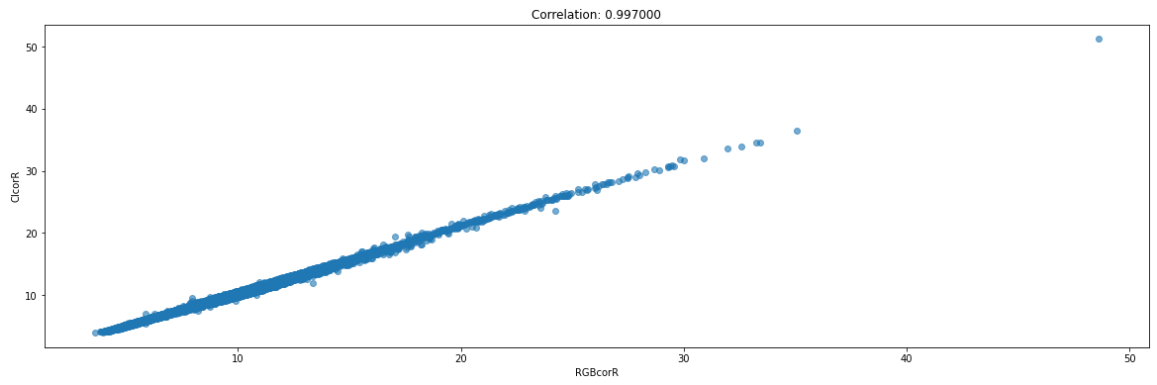


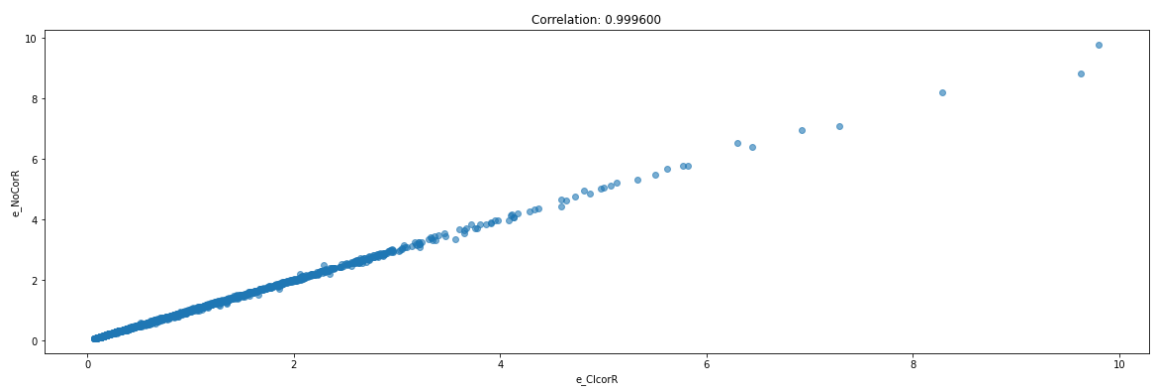
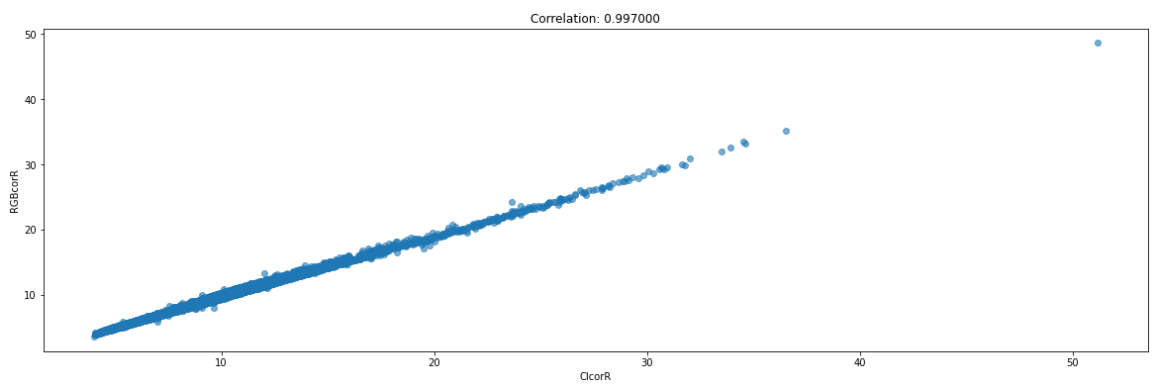
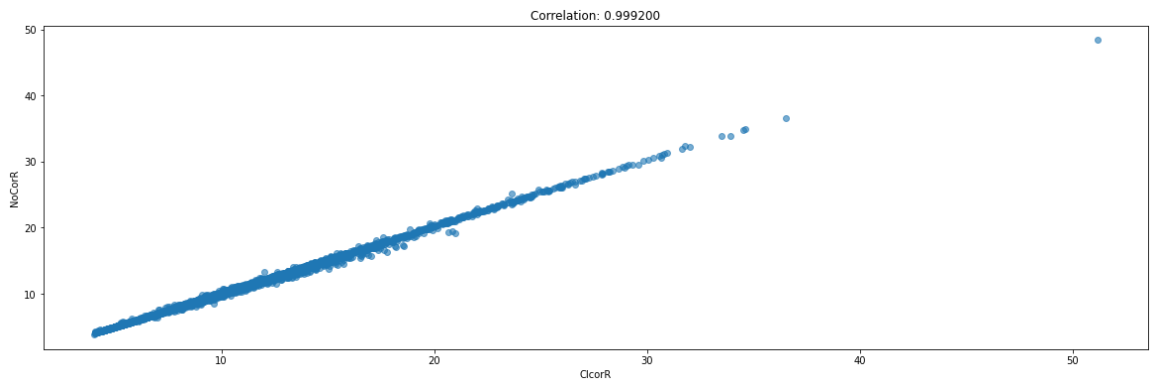
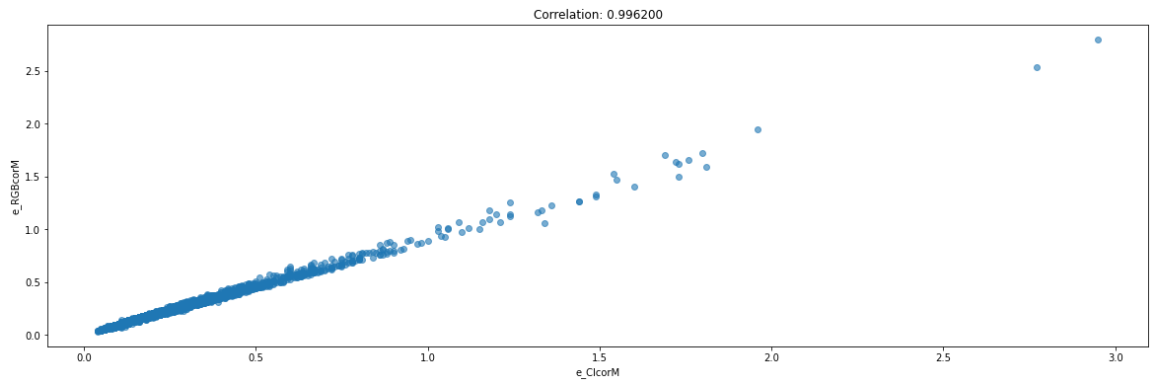
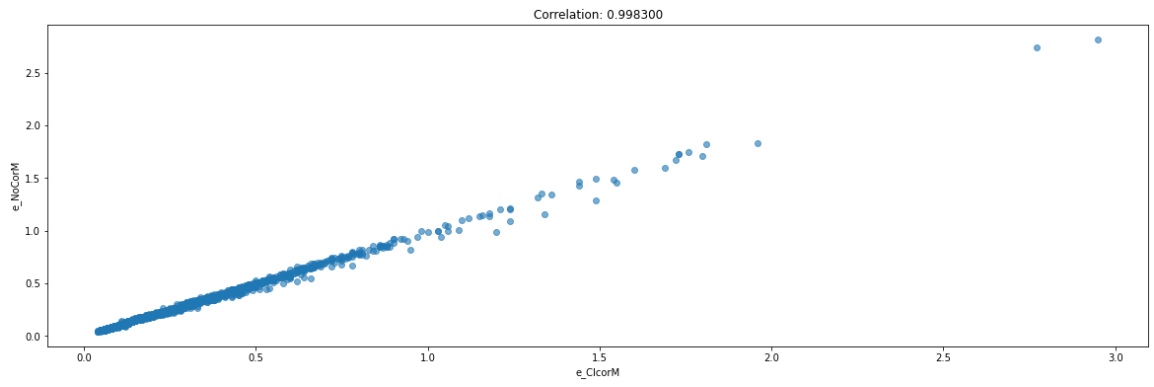


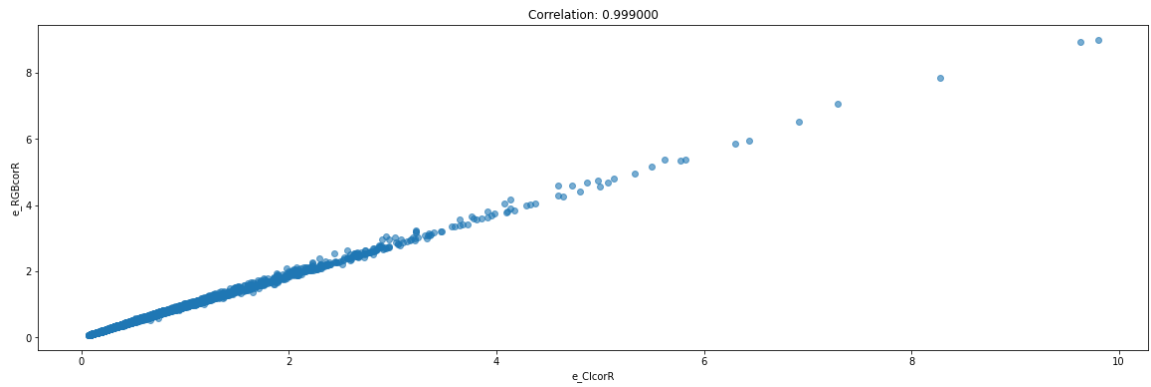












Из полученной матрицы корреляции и графиков коррелиции можно выявить, что есть параметры которые можно отбросить, $\text{corr}[A, B]$ которых = 1

```
In [36]: par = ["NoCorM", "e_NoCorM", "NoCorR", "e_NoCorR", "RGBcorM", "e_RGBcorM", "NoCorR", "e_NoCorR", "RGBcorM", "e_RGBcorM", "NoCorR", "e_NoCorR", "RGBcorR", "e_RGBcorR", "NoCorR", "e_NoCorR", "RGBcorR", "e_RGBcorR"]
```

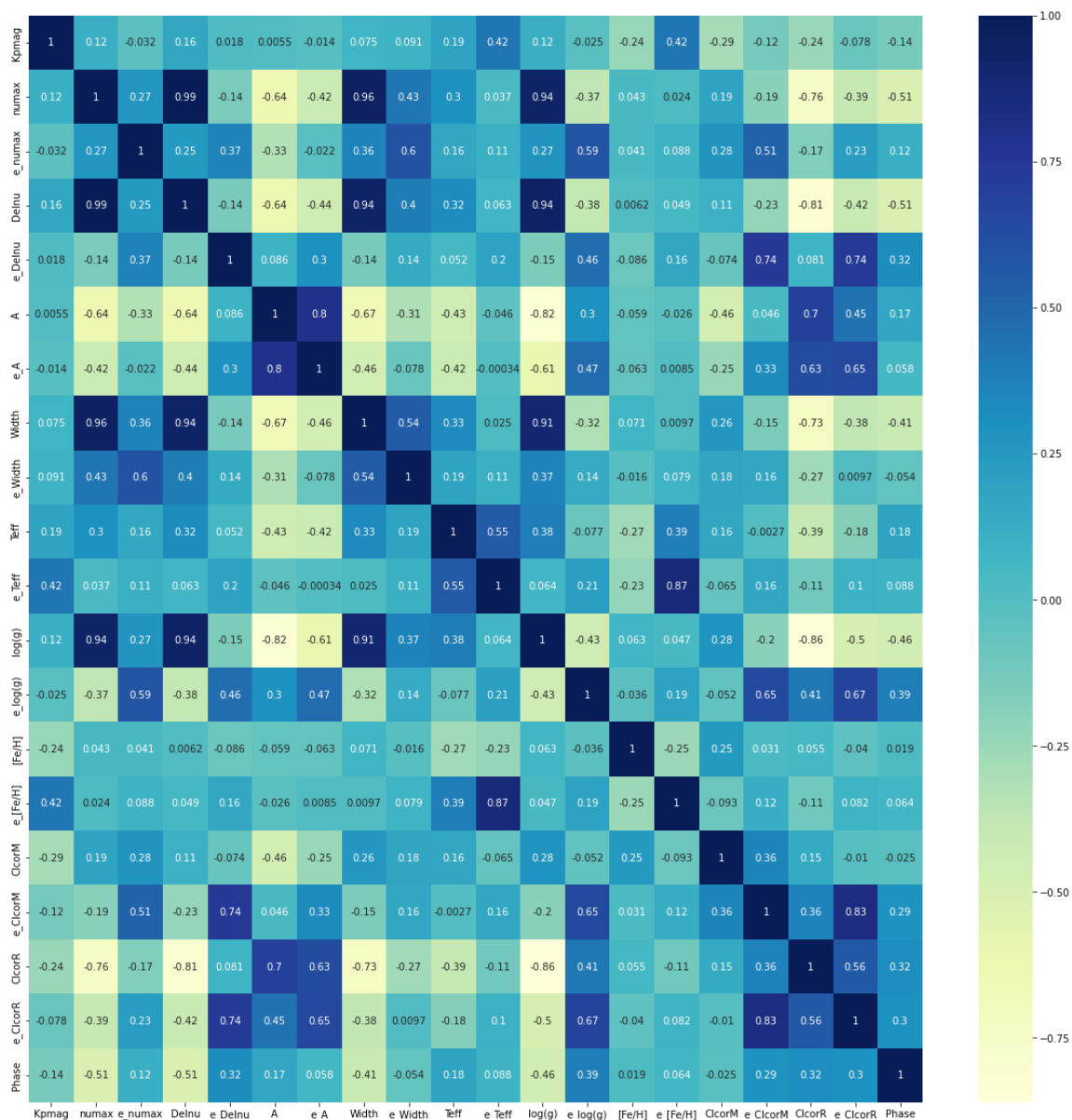
```
Out[36]: ['NoCorM',
'e_NoCorM',
'NoCorR',
'e_NoCorR',
'RGBcorM',
'e_RGBcorM',
'RGBcorR',
'e_RGBcorR']
```

```
In [37]: df = df.drop(par, axis = 1)
df_un = df_un.drop(par, axis = 1)
print(df.shape, df_un.shape)
```

```
(14851, 20) (706, 20)
```

```
In [38]: plt.figure(figsize = (20, 20))
corr_matrix = df.corr()
sns.heatmap(corr_matrix, annot=True, cmap="YlGnBu")
```

```
Out[38]: <AxesSubplot:>
```



Вывод по 0 части

В данной части лабораторной работы мной был найден датасет для определения фазы эволюционировавших звезд (Фаза HeB, Фаза RGB). В задании требуется по полученным данным для звезд, у которых не определена фаза, определить ее. В этой работе мной был реализован эта PRE-PROCESSING для подготовки данных к задачам машинного обучения. Я подготовил данные для решения задачи классификации по параметрам.