Московский государственный технический университет им. Н.Э. Баумана
Кафедра «Системы обработки информации и управления»

Лабораторная работа №2
по дисциплине
«Методы машинного обучения»
на тему
«Обработка признаков (часть 1)»

Выполнил:
студент группы
Пряхин В. Г.

Москва — 2021 г.

1. Цель лабораторной

Изучение продвинутых способов предварительной обработки данных для дальнейшего формирования моделей.

2. Задание

Требуется выполнить следующие действия:

- Выбрать набор данных (датасет) содержащий категориальные и числовые признаки и пропуски в данных.
- Для выбранного датасета решить следующие задачи:
    1. устранение пропусков в данных;
    2. кодирование категориальных признаков;
    3. нормализацию числовых признаков.

3. Текст программы и экранные формы

3.1. Загрузка и предобработка данных

Воспользуемся датасетом с первой лабораторной работы для решения вышеперечисленных задач.

[1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
data_loaded = pd.read_csv('who_life_exp.csv', sep=",")
```

Выведем количество строк/столбцов и часть набора данных.

[2]:
```
data_loaded.shape
```

[2]:
```
(3111, 32)
```

[3]:
```
data_loaded.head()
```

[3]:
```
  country country_code  region  year  life_expect  life_exp60  \
0  Angola          AGO  Africa  2000     47.33730    14.73400
1  Angola          AGO  Africa  2001     48.19789    14.95963
2  Angola          AGO  Africa  2002     49.42569    15.20010
3  Angola          AGO  Africa  2003     50.50266    15.39144
4  Angola          AGO  Africa  2004     51.52863    15.56860

   adult_mortality  infant_mort  age1-4mort  alcohol  …  che_gdp  ␣
↪une_pop  \
0         383.5583     0.137985    0.025695  1.47439  …  1.90860  16395.473
1         372.3876     0.133675    0.024500  1.94025  …  4.48352  16945.753
2         354.5147     0.128320    0.023260  2.07512  …  3.32946  17519.417
3         343.2169     0.122040    0.021925  2.20275  …  3.54797  18121.479
```

```
4         333.8711     0.115700    0.020545  2.41274  …  3.96720  18758.145

   une_infant  une_life  une_hiv  une_gni  une_poverty  une_edu_spend  \
0       122.2    46.522      1.0   2530.0         32.3        2.60753
1       118.9    47.059      1.1   2630.0          NaN            NaN
2       115.1    47.702      1.2   3180.0          NaN            NaN
3       110.8    48.440      1.3   3260.0          NaN            NaN
4       106.2    49.263      1.3   3560.0          NaN            NaN

   une_literacy  une_school
0           NaN         NaN
1      67.40542         NaN
2           NaN         NaN
3           NaN         NaN
4           NaN         NaN

[5 rows x 32 columns]
```

[4]:
```
# Вычислим процент пропущенных значений
[(c, data_loaded[c].isnull().mean() * 100) for c in data_loaded]
```

[4]:
```
[('country', 0.0),
 ('country_code', 0.0),
 ('region', 0.0),
 ('year', 0.0),
 ('life_expect', 0.0),
 ('life_exp60', 0.0),
 ('adult_mortality', 0.0),
 ('infant_mort', 0.0),
 ('age1-4mort', 0.0),
 ('alcohol', 1.6072002571520412),
 ('bmi', 1.092896174863388),
 ('age5-19thinness', 1.092896174863388),
 ('age5-19obesity', 1.092896174863388),
 ('hepatitis', 18.289938926390228),
 ('measles', 0.6107360977177756),
 ('polio', 0.6107360977177756),
 ('diphtheria', 0.6107360977177756),
 ('basic_water', 1.0286081645773062),
 ('doctors', 42.783670845387334),
 ('hospitals', 95.8212793314047),
 ('gni_capita', 21.92221150755384),
 ('gghe-d', 3.2144005143040824),
 ('che_gdp', 3.7608486017357765),
 ('une_pop', 1.1893281902925106),
 ('une_infant', 0.0),
 ('une_life', 0.0),
 ('une_hiv', 23.81870781099325),
 ('une_gni', 3.7608486017357765),
 ('une_poverty', 70.65252330440373),
```

```
 ('une_edu_spend', 41.3371906139505),
 ('une_literacy', 81.64577306332369),
 ('une_school', 74.12407585985214)]
```

Данные представляют собой выгрузку с серверов «Глобальной обсерватории здравоохранения»(GHO) и ЮНЕСКО за шестнадь лет (2000 - 2016). Воспользуемся данными с серверов GHO, так как они наиболее полные и параметром une_gni из набора данных ЮНЕСКО.

[5]:
```
data = data_loaded[['country',␣
↪'country_code','region','year','life_expect','adult_mortality',␣
↪'infant_mort','alcohol','bmi','basic_water','doctors','hospitals','une_gni']]
```

[6]:
```
data.head()
```

[6]:

	country	country_code	region	year	life_expect	adult_mortality	infant_mort	alcohol	bmi	basic_water	doctors	hospitals	une_gni
0	Angola	AGO	Africa	2000	47.33730	383.5583	0.137985	1.47439	21.7	41.14431	NaN	NaN	2530.0
1	Angola	AGO	Africa	2001	48.19789	372.3876	0.133675	1.94025	21.8	42.25467	NaN	NaN	2630.0
2	Angola	AGO	Africa	2002	49.42569	354.5147	0.128320	2.07512	21.9	43.37680	NaN	NaN	3180.0
3	Angola	AGO	Africa	2003	50.50266	343.2169	0.122040	2.20275	22.0	44.36387	NaN	NaN	3260.0
4	Angola	AGO	Africa	2004	51.52863	333.8711	0.115700	2.41274	22.2	45.35134	0.621	NaN	3560.0

3.2. Устранение пропусков в данных

[7]:
```
# Вычислим процент пропущенных значений
[(c, data[c].isnull().mean() * 100) for c in data]
```

[7]:
```
[('country', 0.0),
 ('country_code', 0.0),
 ('region', 0.0),
 ('year', 0.0),
 ('life_expect', 0.0),
 ('adult_mortality', 0.0),
 ('infant_mort', 0.0),
 ('alcohol', 1.6072002571520412),
 ('bmi', 1.092896174863388),
 ('basic_water', 1.0286081645773062),
 ('doctors', 42.783670845387334),
 ('hospitals', 95.8212793314047),
 ('une_gni', 3.7608486017357765)]
```

Удалим колонки с преобладающими пропусками

[8]:
```
data = data.drop(['doctors','hospitals'],axis='columns')
```

Заполним пропущенные данные для параметров alcohol, bmi, basic_water одним из показателей центра распределения — средним значением.

[9]:
```
data['alcohol'].fillna((data['alcohol'].mean()), inplace=True)
```

[10]:
```
data['bmi'].fillna((data['alcohol'].mean()), inplace=True)
```

[11]:
```
data['basic_water'].fillna((data['alcohol'].mean()), inplace=True)
```

[12]:
```
data['une_gni'].fillna((data['une_gni'].mean()), inplace=True)
```

[13]:
```
# Вычислим процент пропущенных значений
[(c, data[c].isnull().mean() * 100) for c in data]
```

[13]:
```
[('country', 0.0),
 ('country_code', 0.0),
 ('region', 0.0),
 ('year', 0.0),
 ('life_expect', 0.0),
 ('adult_mortality', 0.0),
 ('infant_mort', 0.0),
 ('alcohol', 0.0),
 ('bmi', 0.0),
 ('basic_water', 0.0),
 ('une_gni', 0.0)]
```

[14]:
```
data.head()
```

[14]:

	country	country_code	region	year	life_expect	adult_mortality	infant_mort	alcohol	bmi	basic_water	une_gni
0	Angola	AGO	Africa	2000	47.33730	383.5583	0.137985	1.47439	21.7	41.14431	2530.0
1	Angola	AGO	Africa	2001	48.19789	372.3876	0.133675	1.94025	21.8	42.25467	2630.0
2	Angola	AGO	Africa	2002	49.42569	354.5147	0.128320	2.07512	21.9	43.37680	3180.0
3	Angola	AGO	Africa	2003	50.50266	343.2169	0.122040	2.20275	22.0	44.36387	3260.0
4	Angola	AGO	Africa	2004	51.52863	333.8711	0.115700	2.41274	22.2	45.35134	3560.0

[15]:
```
data.shape
```

[15]:
```
(3111, 11)
```

3.3. Кодирование категориальных признаков

Выполним кодирование категорий целочисленными значениями для колонки “Регион”

[16]:
```
from sklearn.preprocessing import LabelEncoder
```

[17]:
```
le = LabelEncoder()
cat_enc_le = le.fit_transform(data['region'])
```

[18]:
```
data['region'].unique()
```

[18]:
```
array(['Africa', 'Americas', 'Eastern Mediterranean', 'Europe',
       'South-East Asia', 'Western Pacific'], dtype=object)
```

[19]:
```
np.unique(cat_enc_le)
```

[19]:
```
array([0, 1, 2, 3, 4, 5])
```

[20]:
```
le.inverse_transform([0, 1, 2, 3])
```

[20]:
```
array(['Africa', 'Americas', 'Eastern Mediterranean', 'Europe'],
      dtype=object)
```

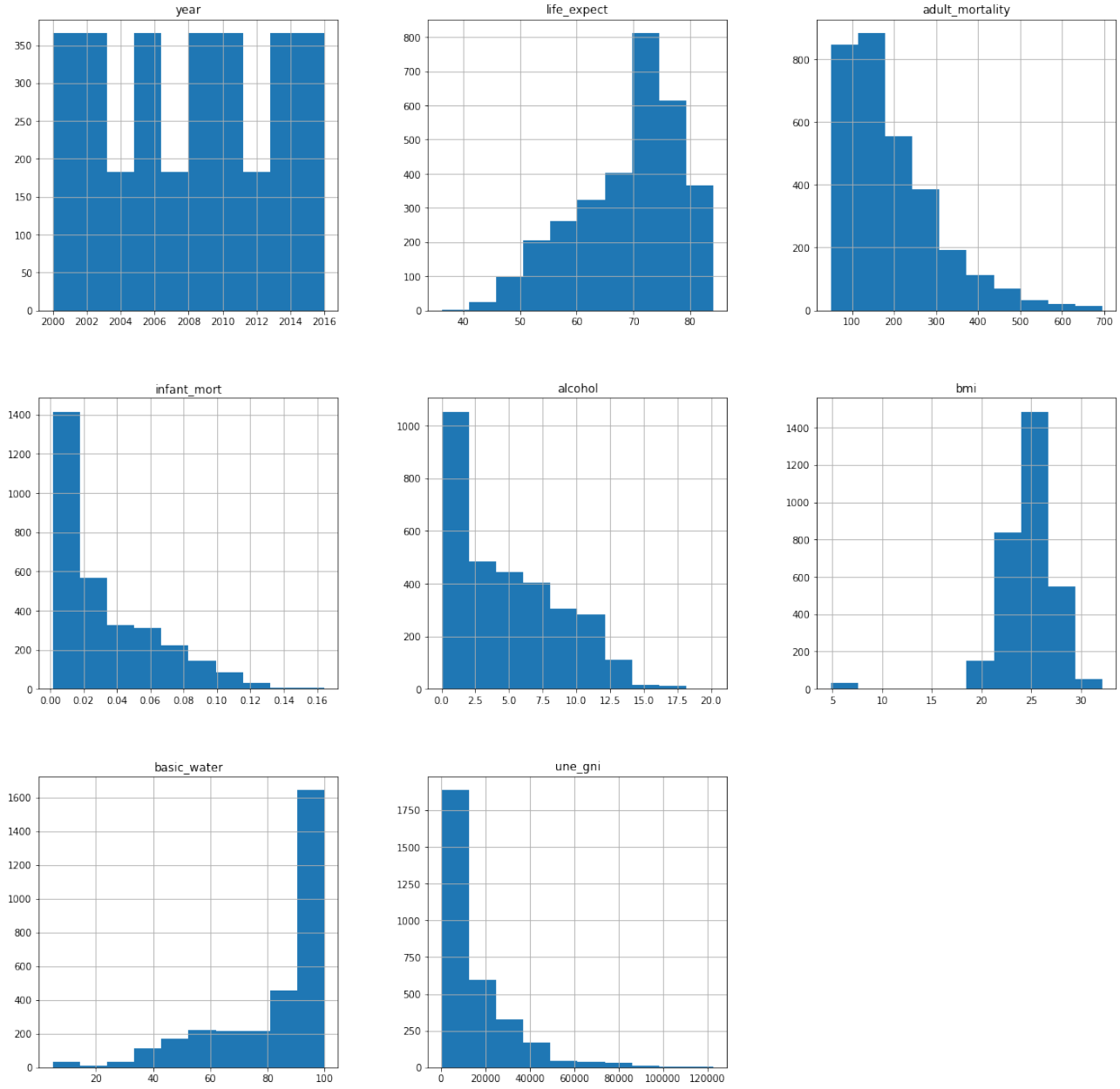
В результате 0 регион соответствует ‘Africa’, 1 - ‘Americas’, 2 - ‘Eastern Mediterranean’ и 3 - ‘Europe’.

3.4. Нормализация числовых признаков

Вывыдем гистаграммы плотности.

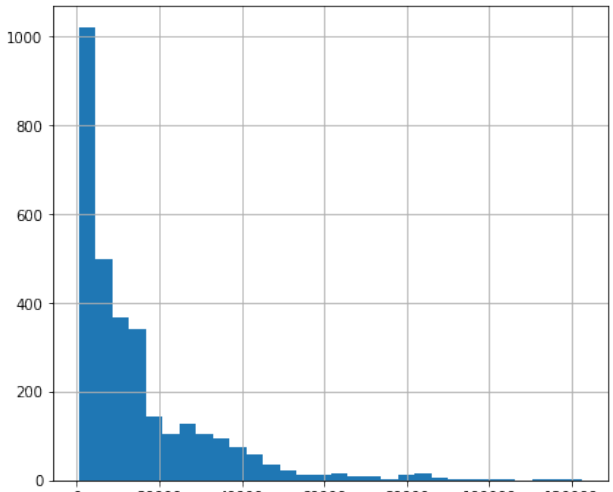
[21]:
```
import scipy.stats as stats
def diagnostic_plots(df, variable):
    plt.figure(figsize=(15,6))
    # гистограмма
    plt.subplot(1, 2, 1)
    df[variable].hist(bins=30)
    ## Q-Q plot
    plt.subplot(1, 2, 2)
    stats.probplot(df[variable], dist="norm", plot=plt)
    plt.show()
```

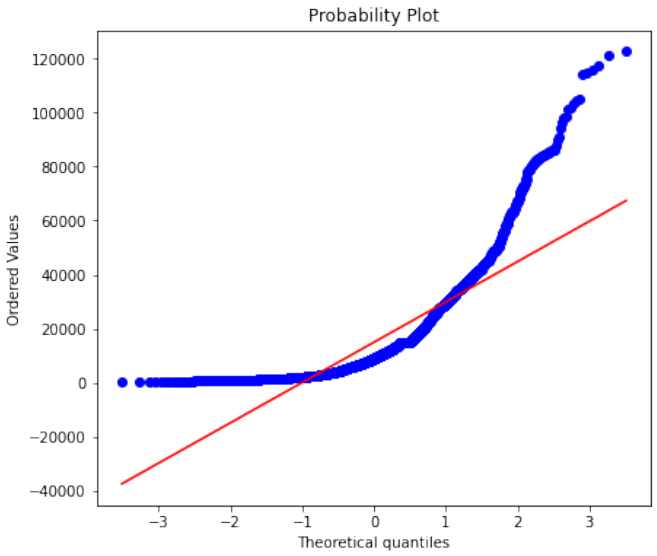
[22]:
```
data.hist(figsize=(20,20))
plt.show()
```



Отобразим график квантиль-квантиль для визуального оценки близости распределения к нормальному.

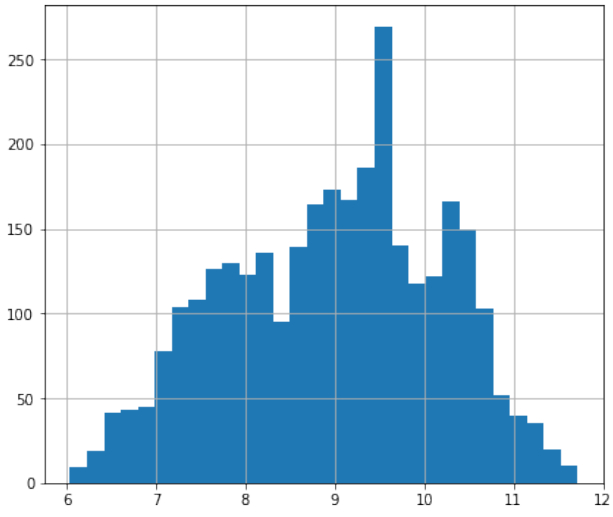
[23]:
```
diagnostic_plots(data, 'une_gni')
```

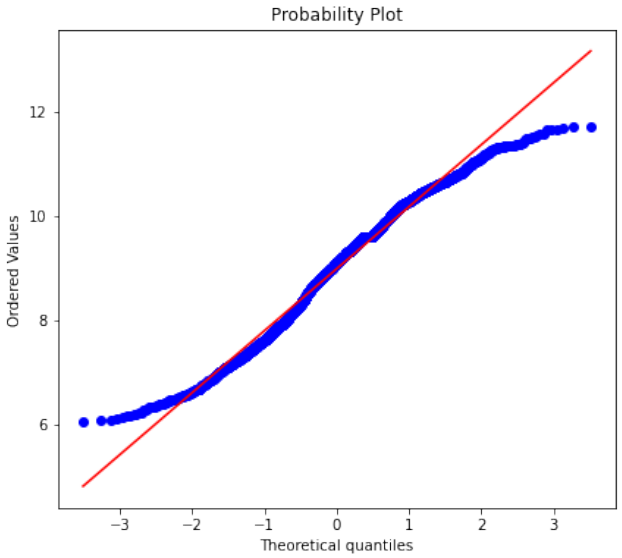




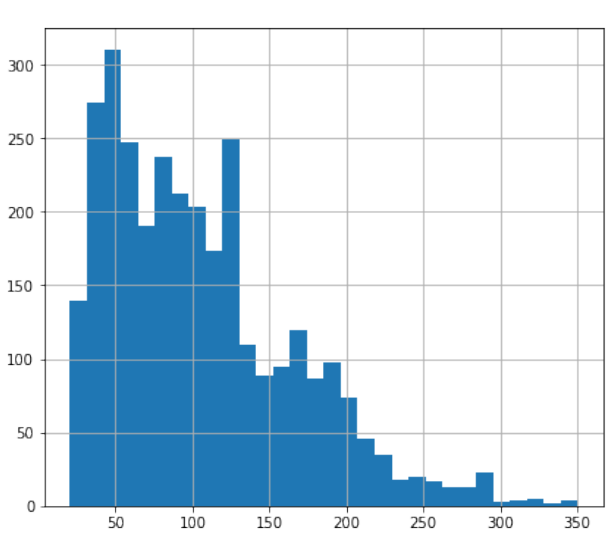
Выведем наиболее удачные методы нормализация числового признака ‘une_gni’

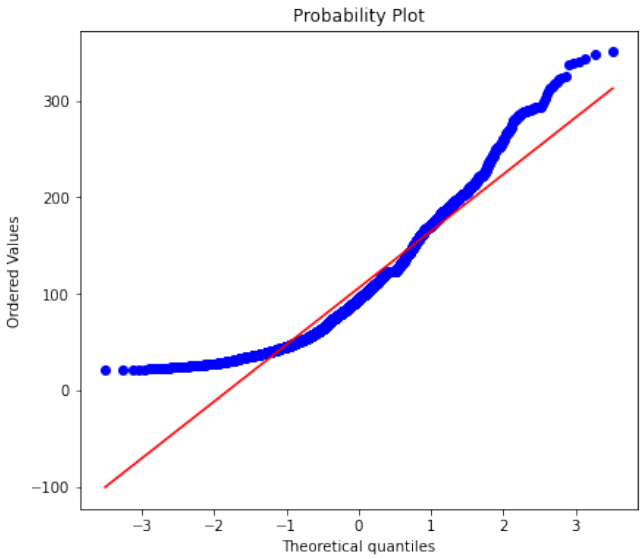
[33]:
```
#логарифмируем
data['GrLivArea_log'] = np.log(data['une_gni'])
diagnostic_plots(data, 'GrLivArea_log')
```



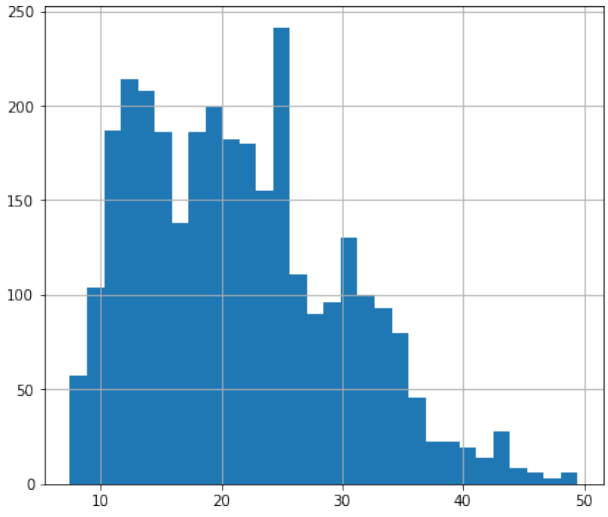


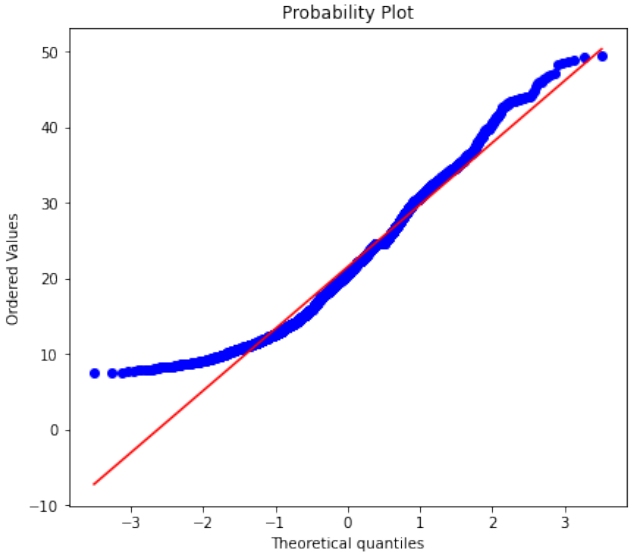
[35]:
```
# квадратный корень
data['GrLivArea_sqr'] = data['une_gni']**(1/2)
diagnostic_plots(data, 'GrLivArea_sqr')
```





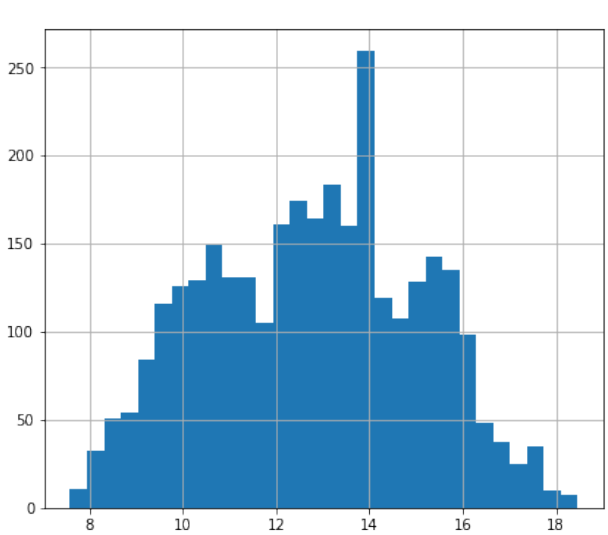
[38]:
```
#возведение в степень(кубический корень)
data['GrLivArea_exp3'] = data['une_gni']**(0.333)
diagnostic_plots(data, 'GrLivArea_exp3')
```

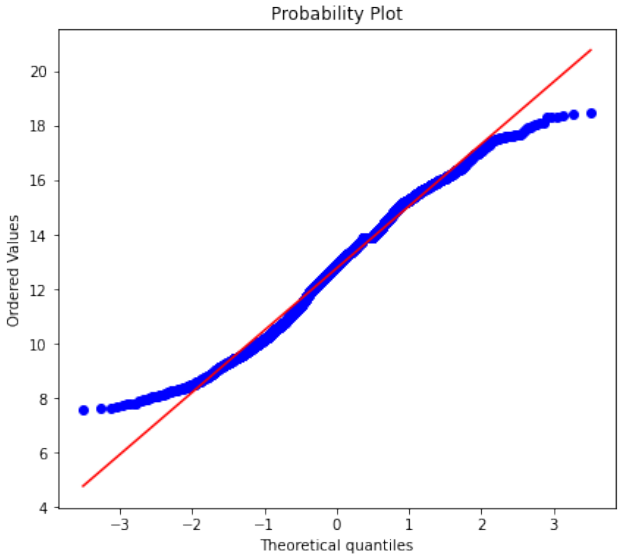




[39]:
```
#преобразование Бокса-Кокса
data['GrLivArea_boxcox'], param = stats.boxcox(data['une_gni'])
print('Оптимальное значение λ = {}'.format(param))
diagnostic_plots(data, 'GrLivArea_boxcox')
```

```
Оптимальное значение λ = 0.07254153648769908
```





Для параметра une_gni(Валовый Национальный Доход на душу населения) наиболее удачные методы нормализации - логарифмирование и преобразование Бокса-Кокса. ВНД сильно раличается в разных странах, поэтому при логарифмировании эти различия уменьшаются.