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

**Лабораторная работа №2**

# «Обработка признаков (часть 1)»

**ИСПОЛНИТЕЛЬ:**

Костян Алина Алексеевна
Группа ИУ5-21М

_______________________

Москва 2021

## Задание:

1. Выбрать набор данных (датасет), содержащий категориальные и числовые признаки и пропуски в данных. Для выполнения следующих пунктов можно использовать несколько различных наборов данных (один для обработки пропусков, другой для категориальных признаков и т.д.) Просьба не использовать датасет, на котором данная задача решалась в лекции.
2. Для выбранного датасета (датасетов) на основе материалов лекций решить следующие задачи:
   - устранение пропусков в данных;
   - кодирование категориальных признаков;
   - нормализацию числовых признаков.

## Текст программы:

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as stats
```

In [3]:
```
data = pd.read_csv('healthcare-dataset-stroke-data.csv')
```

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

Out[4]:

| | id | gender | age | hypertension | heart_disease | ever_married | work_type | Residence_type | avg_glucose_level | bmi | smoking_status | stroke |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 9046 | Male | 67.0 | 0 | 1 | Yes | Private | Urban | 228.69 | 36.6 | formerly smoked | 1 |
| 1 | 51676 | Female | 61.0 | 0 | 0 | Yes | Self-employed | Rural | 202.21 | NaN | never smoked | 1 |
| 2 | 31112 | Male | 80.0 | 0 | 1 | Yes | Private | Rural | 105.92 | 32.5 | never smoked | 1 |
| 3 | 60182 | Female | 49.0 | 0 | 0 | Yes | Private | Urban | 171.23 | 34.4 | smokes | 1 |
| 4 | 1665 | Female | 79.0 | 1 | 0 | Yes | Self-employed | Rural | 174.12 | 24.0 | never smoked | 1 |

In [5]:
```
data = data.drop('id', 1)
data.head()
```

Out[5]:

| | gender | age | hypertension | heart_disease | ever_married | work_type | Residence_type | avg_glucose_level | bmi | smoking_status | stroke |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | Male | 67.0 | 0 | 1 | Yes | Private | Urban | 228.69 | 36.6 | formerly smoked | 1 |
| 1 | Female | 61.0 | 0 | 0 | Yes | Self-employed | Rural | 202.21 | NaN | never smoked | 1 |
| 2 | Male | 80.0 | 0 | 1 | Yes | Private | Rural | 105.92 | 32.5 | never smoked | 1 |
| 3 | Female | 49.0 | 0 | 0 | Yes | Private | Urban | 171.23 | 34.4 | smokes | 1 |
| 4 | Female | 79.0 | 1 | 0 | Yes | Self-employed | Rural | 174.12 | 24.0 | never smoked | 1 |

In [6]:
```
data_features = list(zip(
# признаки
[i for i in data.columns],
zip(
    # типы колонок
    [str(i) for i in data.dtypes],
    # проверим есть ли пропущенные значения
    [i for i in data.isnull().sum()]
)))
# Признаки с типом данных и количеством пропусков
data_features
```

Out[6]:
```
[('gender', ('object', 0)),
 ('age', ('float64', 0)),
 ('hypertension', ('int64', 0)),
 ('heart_disease', ('int64', 0)),
 ('ever_married', ('object', 0)),
 ('work_type', ('object', 0)),
 ('Residence_type', ('object', 0)),
 ('avg_glucose_level', ('float64', 0)),
 ('bmi', ('float64', 201)),
 ('smoking_status', ('object', 0)),
 ('stroke', ('int64', 0))]
```

## Устранение пропусков

In [7]:
```
# Доля (процент) пропусков
[(c, data[c].isnull().mean()) for c in data.columns]
```

Out[7]:
```
[('gender', 0.0),
 ('age', 0.0),
 ('hypertension', 0.0),
 ('heart_disease', 0.0),
 ('ever_married', 0.0),
 ('work_type', 0.0),
 ('Residence_type', 0.0),
 ('avg_glucose_level', 0.0),
 ('bmi', 0.03933463796477495),
 ('smoking_status', 0.0),
 ('stroke', 0.0)]
```

In [8]:
```
# Заполним пропуски
data.dropna(subset=['age'], inplace=True)
```

In [9]:
```
data['gender'] = data['gender'].astype(str).str[0]
```

In [10]:
```
# Заполним пропуски возраста средними значениями
def impute_na(df, variable, value):
    df[variable].fillna(value, inplace=True)
impute_na(data, 'bmi', data['bmi'].mean())
```

In [11]:
```
# Убедимся что нет пустых значений
data.isnull().sum()
```

Out[11]:
```
gender               0
age                  0
hypertension         0
heart_disease        0
ever_married         0
work_type            0
Residence_type       0
avg_glucose_level    0
bmi                  0
smoking_status       0
stroke               0
dtype: int64
```

In [12]:
```
data.head()
```

Out[12]:

| | gender | age | hypertension | heart_disease | ever_married | work_type | Residence_type | avg_glucose_level | bmi | smoking_status | stroke |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | M | 67.0 | 0 | 1 | Yes | Private | Urban | 228.69 | 36.600000 | formerly smoked | 1 |
| **1** | F | 61.0 | 0 | 0 | Yes | Self-employed | Rural | 202.21 | 28.893237 | never smoked | 1 |
| **2** | M | 80.0 | 0 | 1 | Yes | Private | Rural | 105.92 | 32.500000 | never smoked | 1 |
| **3** | F | 49.0 | 0 | 0 | Yes | Private | Urban | 171.23 | 34.400000 | smokes | 1 |
| **4** | F | 79.0 | 1 | 0 | Yes | Self-employed | Rural | 174.12 | 24.000000 | never smoked | 1 |

## Кодирование категориальных признаков

In [13]:
```
from sklearn.preprocessing import LabelEncoder
```

In [14]:
```
le = LabelEncoder()
cat_enc_le = le.fit_transform(data['work_type'])
```

In [15]:
```
data['work_type'].unique()
```

Out[15]:
```
array(['Private', 'Self-employed', 'Govt_job', 'children', 'Never_worked'],
      dtype=object)
```

In [16]:
```
np.unique(cat_enc_le)
```

Out[16]:
```
array([0, 1, 2, 3, 4])
```

In [17]:
```
le.inverse_transform([0, 1, 2, 3,4])
```

Out[17]:
```
array(['Govt_job', 'Never_worked', 'Private', 'Self-employed', 'children'],
      dtype=object)
```

In [18]:
```
data['smoking_status'].unique()
```

Out[18]:
```
array(['formerly smoked', 'never smoked', 'smokes', 'Unknown'],
      dtype=object)
```

In [22]:
```
#TargetEncoder
from category_encoders.target_encoder import TargetEncoder as ce_TargetEncoder
```

In [23]:
```
ce_TargetEncoder1 = ce_TargetEncoder()
data_MEAN_ENC = ce_TargetEncoder1.fit_transform(data[data.columns.difference(['stroke'])], data['stroke'])
```

In [24]:
```
data_MEAN_ENC.head()
```

Out[24]:

| | Residence_type | age | avg_glucose_level | bmi | ever_married | gender | heart_disease | hypertension | smoking_status | work_type |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0.052003 | 67.0 | 228.69 | 36.600000 | 0.065613 | 0.051064 | 1 | 0 | 0.079096 | 0.050940 |
| **1** | 0.045346 | 61.0 | 202.21 | 28.893237 | 0.065613 | 0.047094 | 0 | 0 | 0.047569 | 0.079365 |
| **2** | 0.045346 | 80.0 | 105.92 | 32.500000 | 0.065613 | 0.051064 | 1 | 0 | 0.047569 | 0.050940 |
| **3** | 0.052003 | 49.0 | 171.23 | 34.400000 | 0.065613 | 0.047094 | 0 | 0 | 0.053232 | 0.050940 |
| **4** | 0.045346 | 79.0 | 174.12 | 24.000000 | 0.065613 | 0.047094 | 0 | 1 | 0.047569 | 0.079365 |

In [25]:
```
def check_mean_encoding(field):
    for s in data[field].unique():
        data_filter = data[data[field]==s]
        if data_filter.shape[0] > 0:
            prob = sum(data_filter['stroke']) / data_filter.shape[0]
            print(s, '-' , prob)
```

In [26]:
```
check_mean_encoding('gender')
```

```
M - 0.05106382978723404
F - 0.047094188376753505
O - 0.0
```

In [27]:
```
check_mean_encoding('smoking_status')
```

```
formerly smoked - 0.07909604519774012
never smoked - 0.04756871035940803
smokes - 0.053231939163498096
Unknown - 0.03044041450777202
```

In [28]:
```
check_mean_encoding('work_type')
```

```
Private - 0.05094017094017094
Self-employed - 0.07936507936507936
Govt_job - 0.0502283105022831
children - 0.002911208151382824
Never_worked - 0.0
```

In [29]:
```
#Weight of evidence (WoE) encoding
from category_encoders.woe import WOEEncoder as ce_WOEEncoder
```

In [30]:
```
ce_WOEEncoder1 = ce_WOEEncoder()
data_WOE_ENC = ce_WOEEncoder1.fit_transform(data[data.columns.difference(['stroke'])], data['stroke'])
```

In [31]:
```
data_WOE_ENC.head()
```

Out[31]:

| | Residence_type | age | avg_glucose_level | bmi | ever_married | gender | heart_disease | hypertension | smoking_status | work_type |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0.067883 | 67.0 | 228.69 | 36.600000 | 0.312055 | 0.050411 | 1 | 0 | 0.522223 | 0.045467 |
| **1** | -0.074751 | 61.0 | 202.21 | 28.893237 | 0.312055 | -0.036692 | 0 | 0 | -0.022390 | 0.526895 |
| **2** | -0.074751 | 80.0 | 105.92 | 32.500000 | 0.312055 | 0.050411 | 1 | 0 | -0.022390 | 0.045467 |
| **3** | 0.067883 | 49.0 | 171.23 | 34.400000 | 0.312055 | -0.036692 | 0 | 0 | 0.107755 | 0.045467 |
| **4** | -0.074751 | 79.0 | 174.12 | 24.000000 | 0.312055 | -0.036692 | 0 | 1 | -0.022390 | 0.526895 |

In [32]:
```
def check_woe_encoding(field):
    data_ones = data[data['stroke'] == 1].shape[0]
    data_zeros = data[data['stroke'] == 0].shape[0]

    for s in data[field].unique():
        data_filter = data[data[field]==s]
        if data_filter.shape[0] > 0:

            filter_data_ones = data_filter[data_filter['stroke'] == 1].shape[0]
            filter_data_zeros = data_filter[data_filter['stroke'] == 0].shape[0]

            good = filter_data_ones / data_ones
            bad = filter_data_zeros / data_zeros

            woe = np.log(good/bad)
            print(s, '-' , woe)
```

In [33]:
```
check_woe_encoding('gender')
```

```
M - 0.04928143890862949
F - -0.035819900254583675
O - -inf
```

In [34]:
```
check_woe_encoding('smoking_status')
```

```
formerly smoked - 0.5168536893892441
never smoked - -0.02529620725727031
smokes - 0.09315099373129877
Unknown - -0.48952422212882185
```

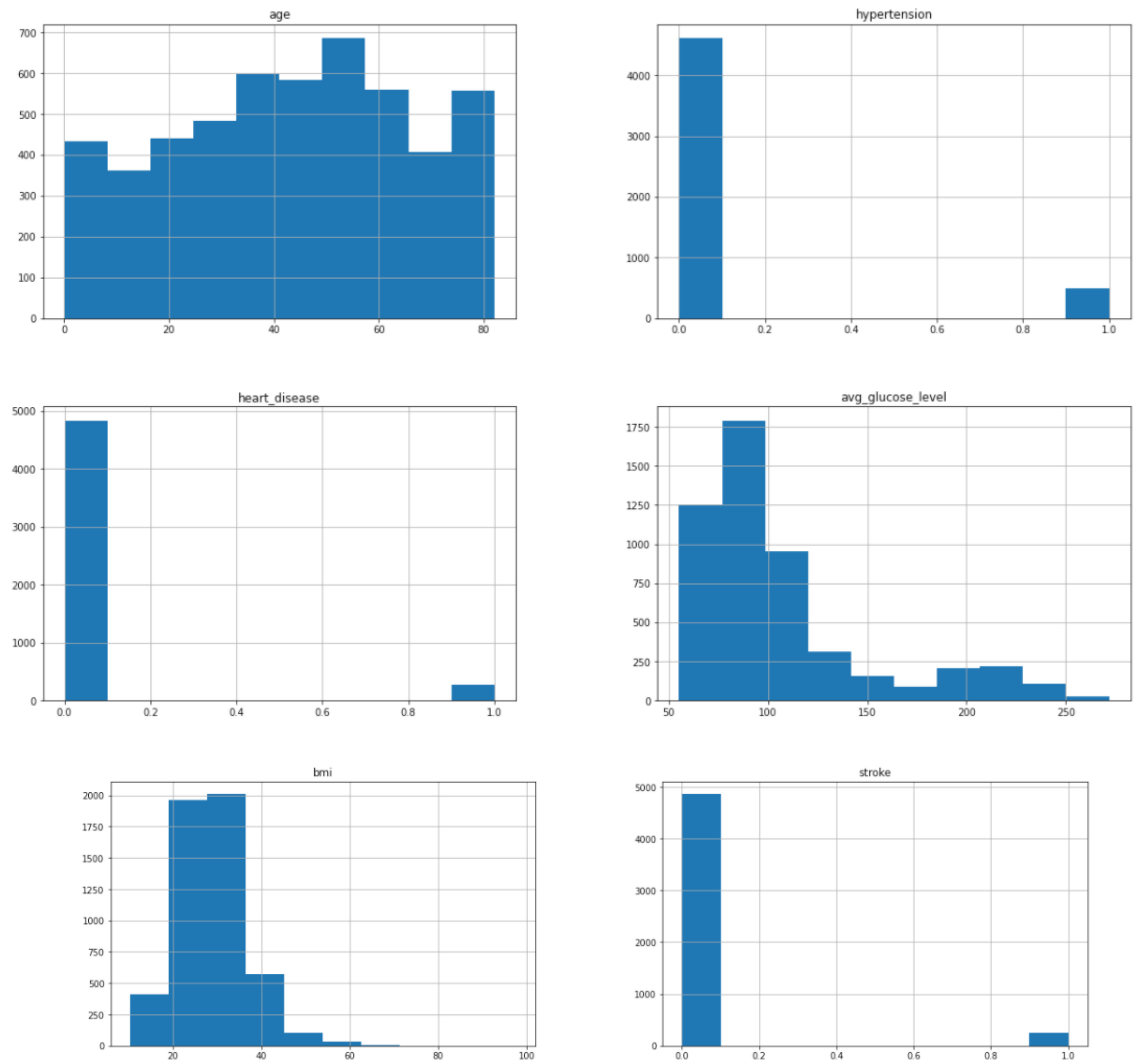
In [35]:
```
check_woe_encoding('work_type')
```

```
Private - 0.046726544899511835
Self-employed - 0.520541462468429
Govt_job - 0.031903753677800545
children - -2.864725097121532
Never_worked - -inf
```

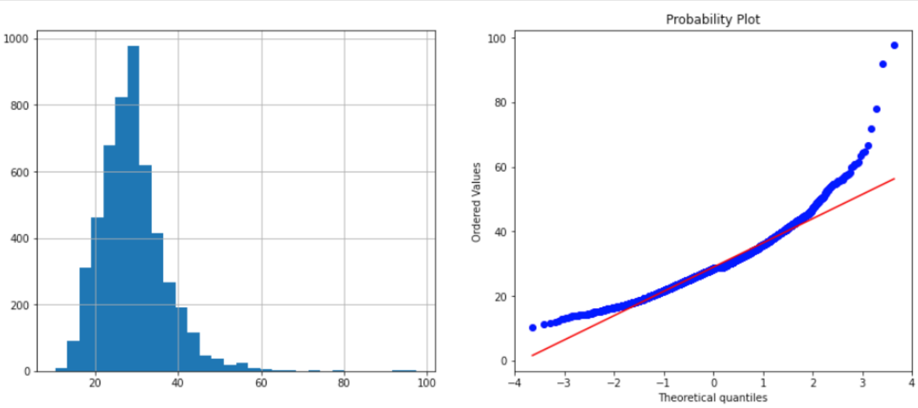
## Нормализация числовых признаков

In [36]:
```
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()
```

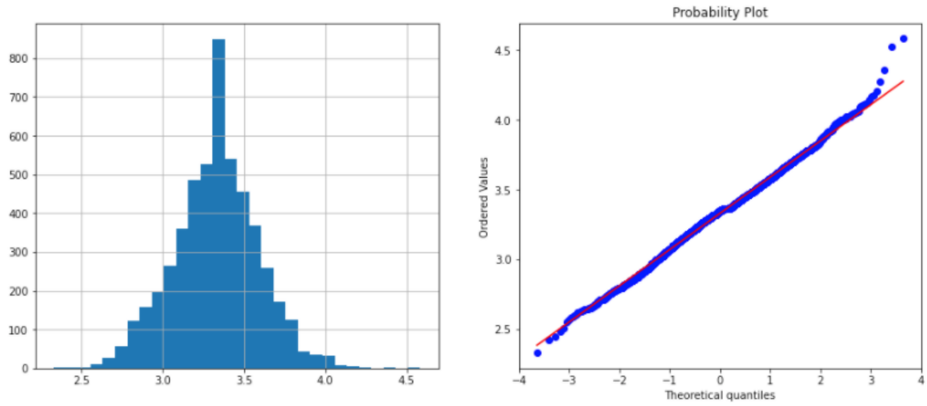
In [37]:
```
data.hist(figsize=(20,20))
plt.show()
```



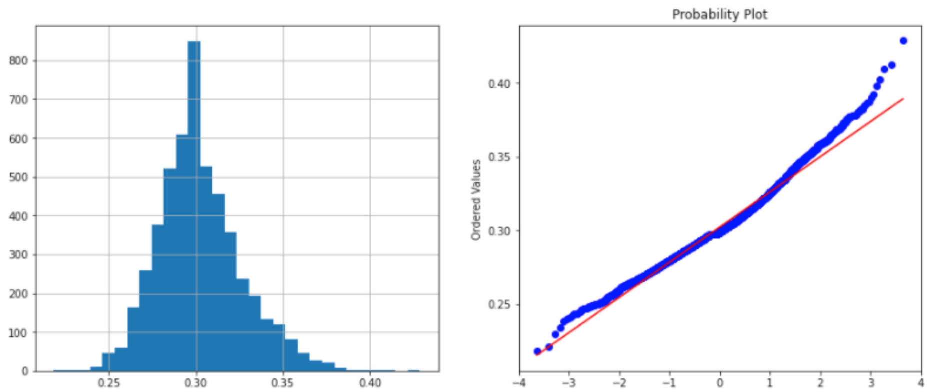

In [38]:
```
diagnostic_plots(data, 'bmi')
```



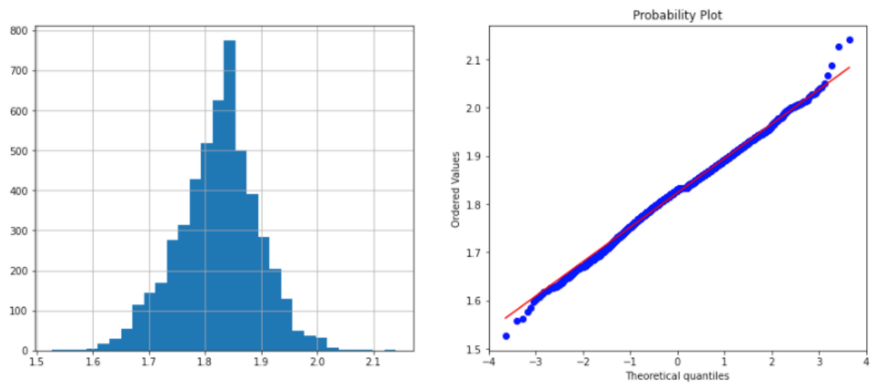
In [39]:
```python
#Логарифмическое преобразование
data['bmi'] = np.log(data['bmi'])
diagnostic_plots(data, 'bmi')
```



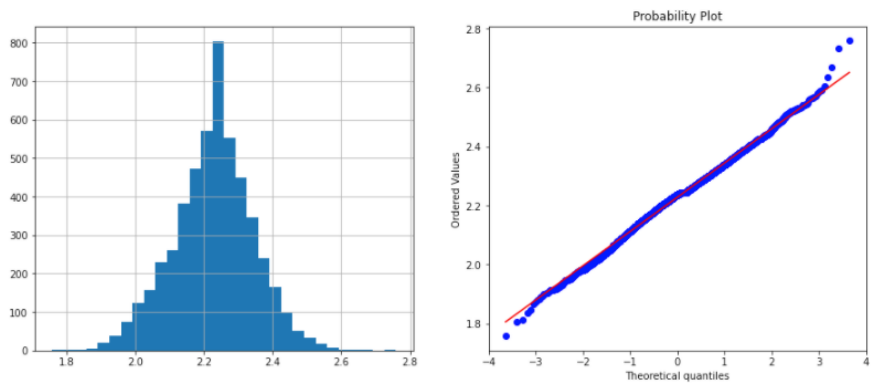

In [40]:
```python
#Обратное преобразование
data['bmi_reciprocal'] = 1 / (data['bmi'])
diagnostic_plots(data, 'bmi_reciprocal')
```



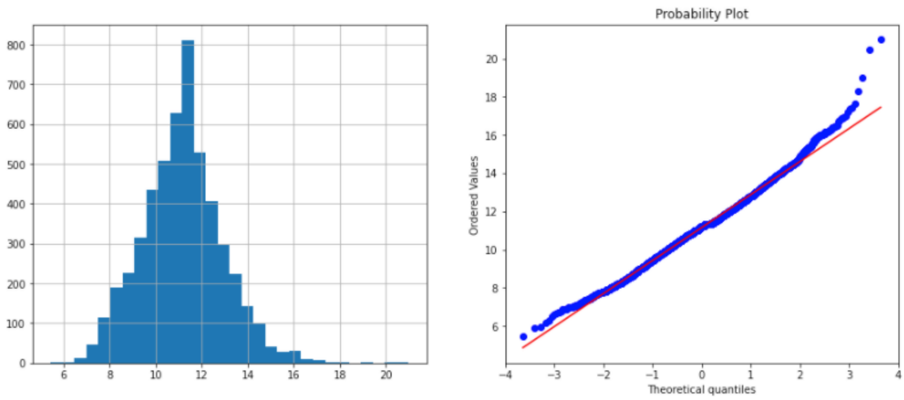

In [41]:
```python
#Квадратный корень
data['bmi_sqr'] = data['bmi']**(1/2)
diagnostic_plots(data, 'bmi_sqr')
```



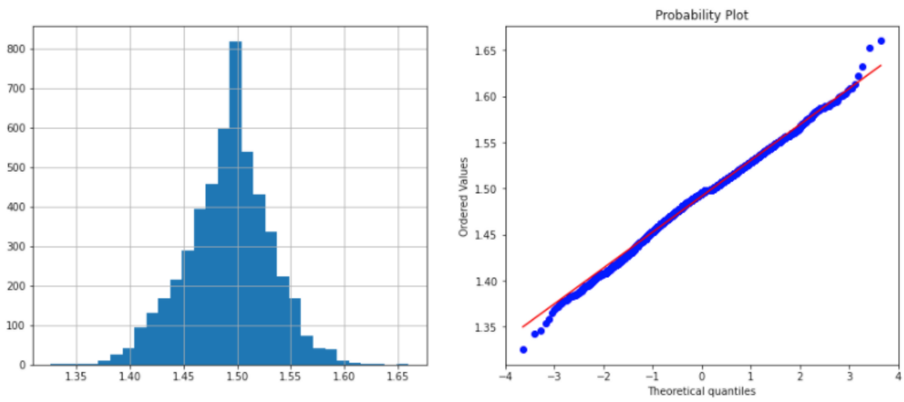

In [42]:
```python
#Возведение в степень
data['bmi_exp1'] = data['bmi']**(1/1.5)
diagnostic_plots(data, 'bmi_exp1')
```



In [43]:
```python
data['bmi_exp2'] = data['bmi']**(2)
diagnostic_plots(data, 'bmi_exp2')
```




In [44]:
```python
data['bmi_exp3'] = data['bmi']**(0.333)
diagnostic_plots(data, 'bmi_exp3')
```




In [45]:
```python
#Преобразованиея Бокса-Кокса
data['bmi_boxcox'], param = stats.boxcox(data['bmi'])
print('Оптимальное значение λ = {}'.format(param))
diagnostic_plots(data, 'bmi_boxcox')
```

```
Оптимальное значение λ = 1.0876864575037113
```