Министерство науки и высшего образования
Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
"Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)"
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ _ИНФОРМАТИКА, ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ И СИСТЕМЫ УПРАВЛЕНИЯ

КАФЕДРА ___СИСТЕМЫ ОБРАБОТКИ ИНФОРМАЦИИ И УПРАВЛЕНИЯ (ИУ5)_____

Отчет по лабораторной работе №2

««Обработка признаков (Часть 1)»

по курсу «Методы машинного обучения».

ИСПОЛНИТЕЛЬ: Группа ИУ5-24М
Уралова Е.А.
ФИО
_____
подпись

"12" марта 2024 г.

ПРЕПОДАВАТЕЛЬ: Гапанюк Ю.Е.
ФИО
_____
подпись

"__"_____2024 г.

Москва – 2024

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;
- нормализация числовых признаков.

3) Сформировать отчет и разместить его в своем репозитории на github.

Описание:

Этот набор данных содержит 76 атрибутов, включая предсказанный атрибут, но все опубликованные эксперименты относятся к использованию подмножества из 14 из них. Поле "цель" относится к наличию сердечного заболевания у пациента. Это целое значение отсутствие заболевания = 0, болезнь = 1.

1) age – возраст;

2) sex – пол;

3) chest pain type (4 values) – тип боли в груди;

4) resting blood pressure – кровяное давление в состоянии покоя;

5) serum cholestoral in mg/dl – холесторал в крови;

6) fasting blood sugar > 120 mg/dl – сахар в крови;

7) resting electrocardiographic results (values 0,1,2) – результаты электрокардиографии в состоянии покоя;

8) maximum heart rate achieved – максимальная частота сердцебиения;

9) exercise induced angina – стенокардия, вызванная физической нагрузкой;

10) oldpeak = ST depression induced by exercise relative to rest – стресс из-за физической нагрузки, в сравнении с покоем;

11) the slope of the peak exercise ST segment – наклон сегмента ST пикового упражнения;

12) number of major vessels (0-3) colored by flourosopy thal: 0 = normal; 1 = fixed defect; 2 = reversable defect – количество крупных сосудов, окрашенных специальным маркером.

Выполнение:

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 [2]:
```
data = pd.read_csv('heart.csv')
```

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

Out[3]:

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAr
0	40	M	ATA	140	289.0	0	Normal	172	
1	49	F	NAP	160	180.0	0	Normal	156	
2	37	M	ATA	130	283.0	0	ST	98	
3	48	F	ASY	138	214.0	0	Normal	108	
4	54	M	NAP	150	NaN	0	Normal	122	

In [4]:
```
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() ]
)))
data_features
```

Out[4]:
```
[('Age', ('int64', 0)),
 ('Sex', ('object', 0)),
 ('ChestPainType', ('object', 0)),
 ('RestingBP', ('int64', 0)),
 ('Cholesterol', ('float64', 1)),
 ('FastingBS', ('int64', 0)),
 ('RestingECG', ('object', 3)),
 ('MaxHR', ('int64', 0)),
 ('ExerciseAngina', ('object', 1)),
 ('Oldpeak', ('float64', 0)),
 ('ST_Slope', ('object', 0)),
 ('HeartDisease', ('int64', 0))]
```

Устранение пропусков

In [5]:
```
# процент пропусков
[(c, data[c].isnull().mean()) for c in data.columns]
```

Out[5]:
```
[('Age', 0.0),
 ('Sex', 0.0),
 ('ChestPainType', 0.0),
 ('RestingBP', 0.0),
 ('Cholesterol', 0.0010893246187363835),
 ('FastingBS', 0.0),
 ('RestingECG', 0.0032679738562091504),
 ('MaxHR', 0.0),
 ('ExerciseAngina', 0.0010893246187363835),
 ('Oldpeak', 0.0),
 ('ST_Slope', 0.0),
 ('HeartDisease', 0.0)]
```

In [6]:
```
#заполнение
data['Cholesterol'] = data['Cholesterol'].astype(str).str[0]
```

In [7]:
```
data['RestingECG'] = data['RestingECG'].astype(str).str[0]
data['ExerciseAngina'] = data['ExerciseAngina'].astype(str).str[0]
```

In [8]:
```
data.isnull().sum()
```

Out[8]:
```
Age               0
Sex               0
ChestPainType     0
RestingBP         0
Cholesterol       0
FastingBS         0
RestingECG        0
MaxHR             0
ExerciseAngina    0
Oldpeak           0
ST_Slope          0
HeartDisease      0
dtype: int64
```

In [9]:
```
data.head()
```

Out[9]:

ex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST_Slope	HeartDisease
M	ATA	140	2	0	N	172	N	0.0	Up	0
F	NAP	160	1	0	N	156	N	1.0	Flat	1
M	ATA	130	2	0	S	98	N	0.0	Up	0
F	ASY	138	2	0	N	108	Y	1.5	Flat	1
M	NAP	150	n	0	N	122	N	0.0	Up	0

Кодирование категориальных признаков

In [10]:
```
from sklearn.preprocessing import LabelEncoder
```

In [11]:
```
le = LabelEncoder()
cat_enc_le = le.fit_transform(data['ChestPainType'])
```

In [12]:
```
data['ChestPainType'].unique()
```

Out[12]:
```
array(['ATA', 'NAP', 'ASY', 'TA'], dtype=object)
```

In [13]:
```
np.unique(cat_enc_le)
```

Out[13]:
```
array([0, 1, 2, 3])
```

In [14]:
```
le.inverse_transform([0,1,2,3])
```

Out[14]:
```
array(['ASY', 'ATA', 'NAP', 'TA'], dtype=object)
```

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

Out[15]:
```
array(['Up', 'Flat', 'Down'], dtype=object)
```

In [16]:
```
#TargetEncoder
from category_encoders.target_encoder import TargetEncoder as ce_TargetEncoder
```

In [19]:
```
getEncoder()
ncoder1.fit_transform(data[data.columns.difference(['HeartDisease'])], data['HeartDisease'])
```

In [20]:
```
data_MEAN_ENC.head()
```

Out[20]:

	Age	ChestPainType	Cholesterol	ExerciseAngina	FastingBS	MaxHR	Oldpeak	RestingBP	RestingECG	ST_Slope	S
0	40	0.138728	0.488889	0.351648	0	172	0.0	140	0.515483	0.197468	0.6317
1	49	0.354680	0.395834	0.351648	0	156	1.0	160	0.515483	0.828261	0.2590
2	37	0.138728	0.488889	0.351648	0	98	0.0	130	0.657303	0.197468	0.6317
3	48	0.790323	0.488889	0.851752	0	108	1.5	138	0.515483	0.828261	0.2590
4	54	0.354680	0.481378	0.351648	0	122	0.0	150	0.515483	0.197468	0.6317

In [21]:
```
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['HeartDisease']) / data_filter.shape[0]
            print(s, '-' , prob)
```

In [22]:
```
check_mean_encoding('Sex')
```

```
M - 0.6317241379310344
F - 0.25906735751295334
```

In [23]:
```
check_mean_encoding('ChestPainType')
```

```
ATA - 0.13872832369942195
NAP - 0.35467980295566504
ASY - 0.7903225806451613
TA - 0.43478260869565216
```

In [24]:
```
check_mean_encoding('ST_Slope')
```

```
Up - 0.19746835443037974
Flat - 0.8282608695652174
Down - 0.7777777777777778
```

In [25]:
```
from category_encoders.woe import WOEEncoder as ce_WOEEncoder
```

In [26]:
```
)EEncoder()
:ncoder1.fit_transform(data[data.columns.difference(['HeartDisease'])], data['HeartDisease'])
```

In [27]:
```
data_WOE_ENC.head()
```

Out[27]:

	Age	ChestPainType	Cholesterol	ExerciseAngina	FastingBS	MaxHR	Oldpeak	RestingBP	RestingECG	ST_Slope	S
0	40	-2.005147	-0.257660	-0.822815	0	172	0.0	140	-0.151662	-1.605991	0.3246
1	49	-0.805730	-0.630281	-0.822815	0	156	1.0	160	-0.151662	1.350007	-1.2513
2	37	-2.005147	-0.257660	-0.822815	0	98	0.0	130	0.430163	-1.605991	0.3246
3	48	1.106462	-0.257660	1.520163	0	108	1.5	138	-0.151662	1.350007	-1.2513
4	54	-0.805730	0.000000	-0.822815	0	122	0.0	150	-0.151662	-1.605991	0.3246

In [28]:
```
def check_woe_encoding(field):
    data_ones = data[data['HeartDisease'] == 1].shape[0]
    data_zeros = data[data['HeartDisease'] == 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['HeartDisease'] == 1].shape[0]
            filter_data_zeros = data_filter[data_filter['HeartDisease'] == 0].shape[0]

            good = filter_data_ones / data_ones
            bad = filter_data_zeros / data_zeros

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

In [29]:
```
check_woe_encoding('Sex')
```

In [29]:
```
check_woe_encoding('Sex')
```

```
M - 0.32529623783380696
F - -1.2651459127118896
```

In [30]:
```
check_woe_encoding('ChestPainType')
```

```
ATA - -2.040216763477642
NAP - -0.8128554920652247
ASY - 1.1125466527689614
TA - -0.4766885523476195
```

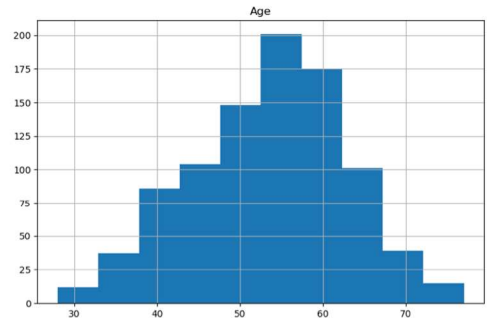
In [31]:
```
check_woe_encoding('ST_Slope')
```

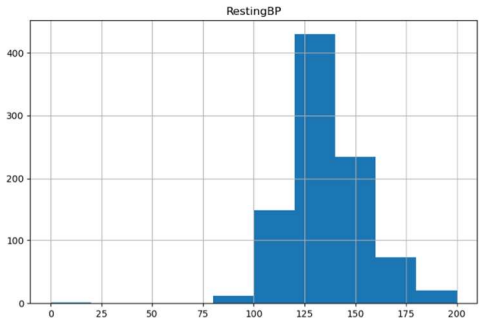
```
Up - -1.6165172350678174
Flat - 1.3590272347795511
Down - 1.0384386806152395
```

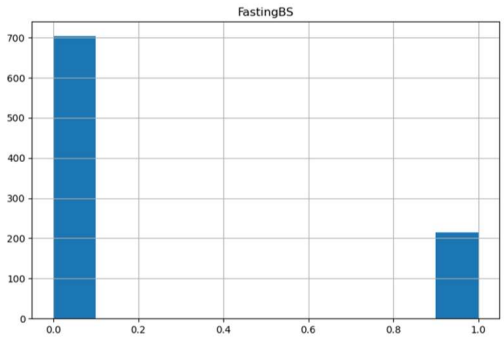
Нормализация числовых признаков

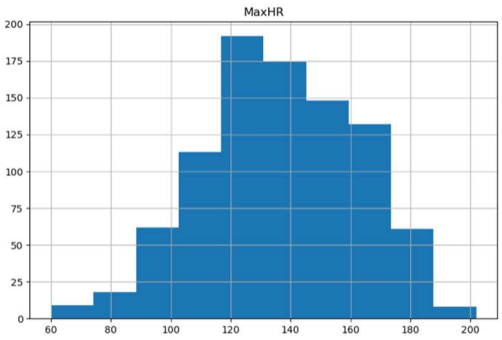
In [32]:
```
def diagnostic_plots(df, variable):
    plt.figure(figsize=(15,6))
    # гистограмма
    plt.subplot(1, 2, 1)
    df[variable].hist(bins=30)
    ## Q-Q plot
    plt.subplot(1, 2, 2)
    stats.probplot(df[variable], dist="norm", plot=plt)
    plt.show()
```

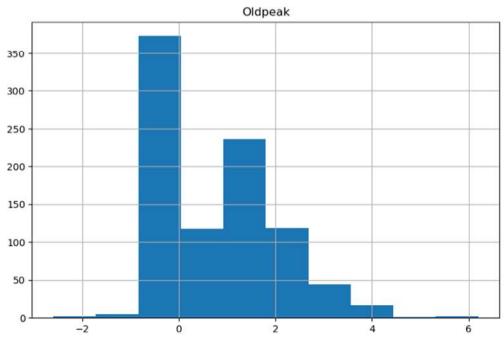
In [33]:
```
data.hist(figsize=(20,20))
plt.show()
```

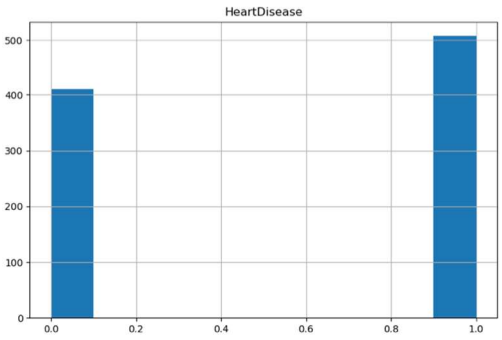




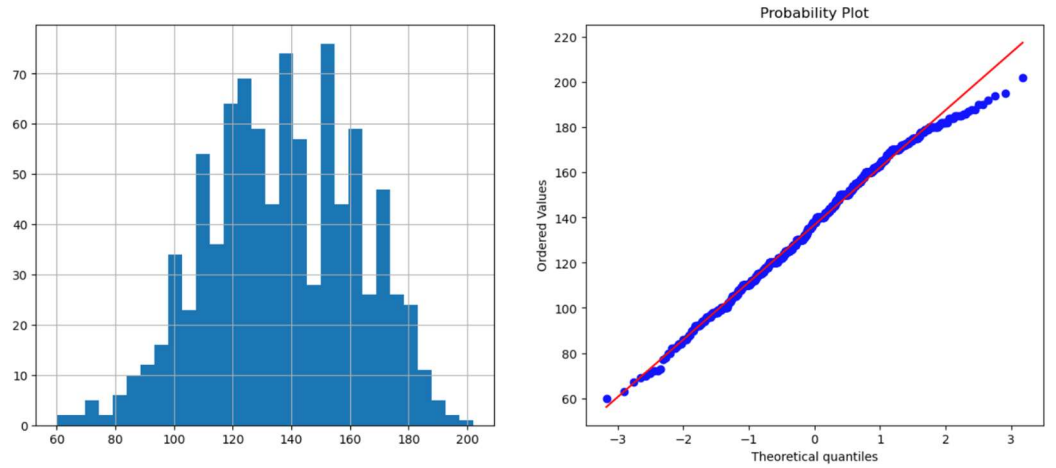




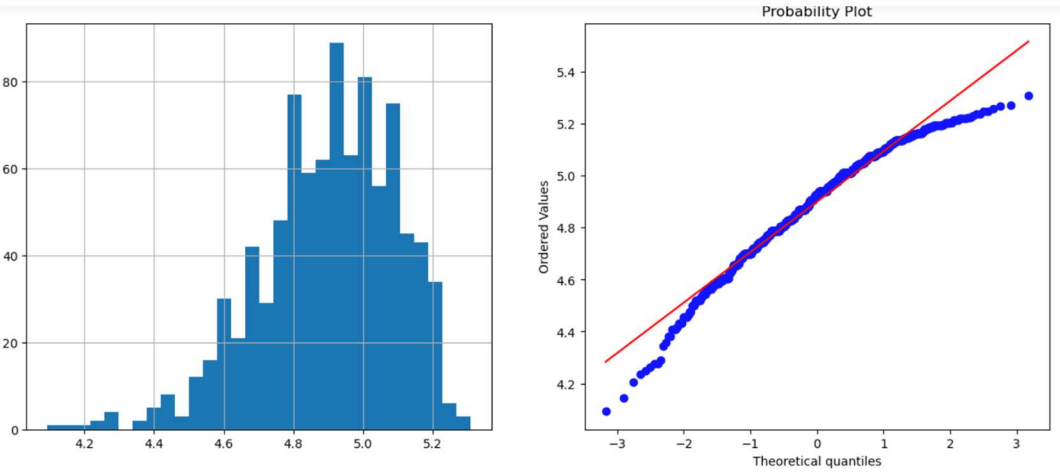




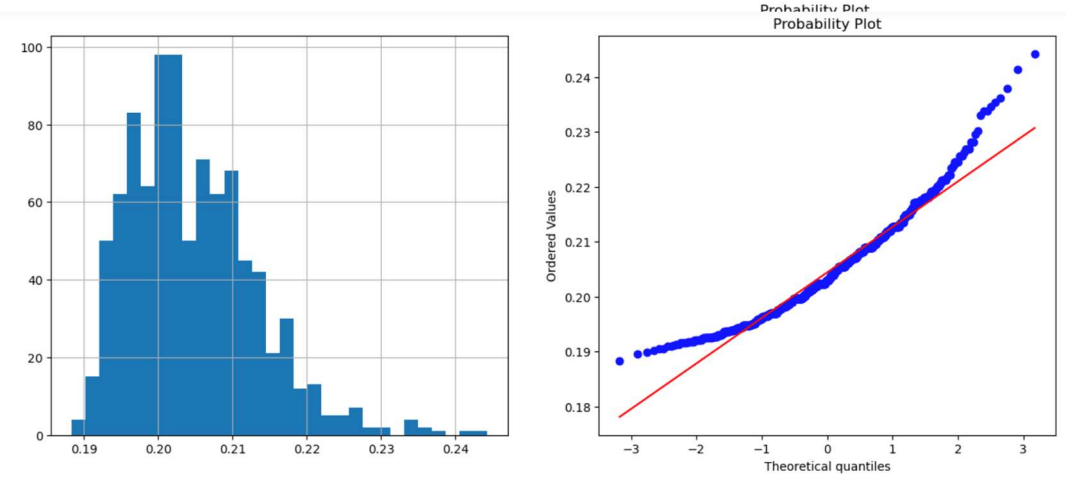
In [35]:
```
diagnostic_plots(data, 'MaxHR')
```



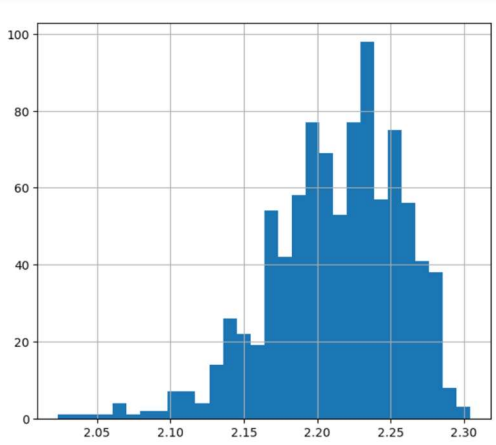
In [36]:
```
#Логарифмическое преобразование
data['MaxHR'] = np.log(data['MaxHR'])
diagnostic_plots(data, 'MaxHR')
```

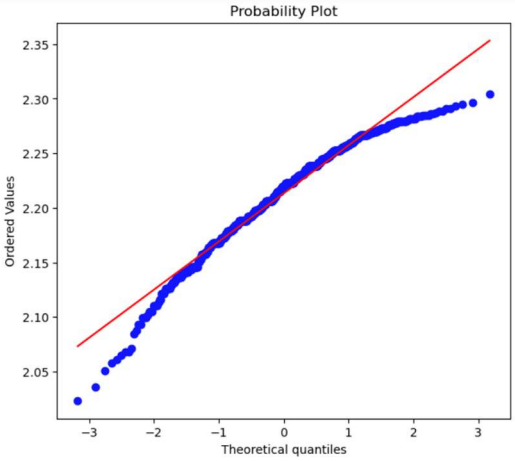


In [38]:
```
#Обратное преобразование
data['MaxHR_rec'] = 1 / (data['MaxHR'])
diagnostic_plots(data, 'MaxHR_rec')
```

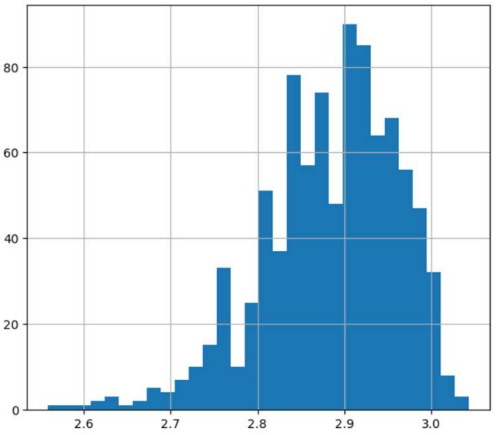


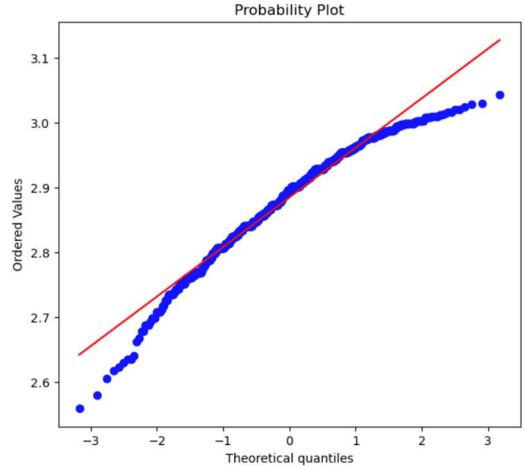
In [39]:
```
#Квадратный корень
data['MaxHR_sqr'] = data['MaxHR']**(1/2)
diagnostic_plots(data, 'MaxHR_sqr')
```



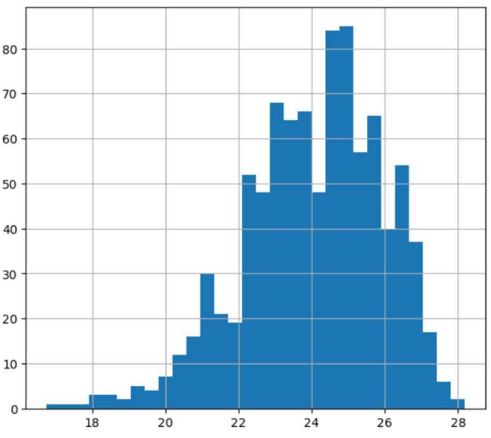


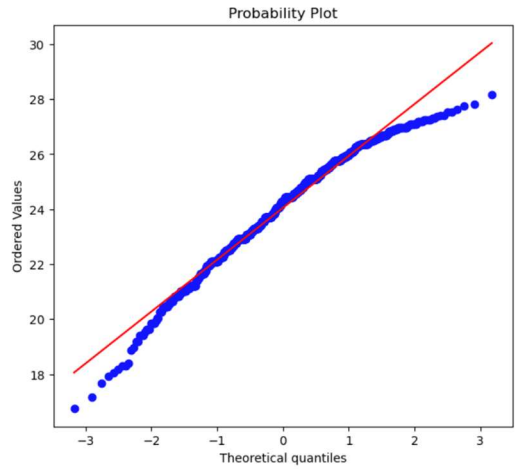
In [40]:
```
#Возведение в степень
data['MaxHR_exp1'] = data['MaxHR']**(1/1.5)
diagnostic_plots(data, 'MaxHR_exp1')
```



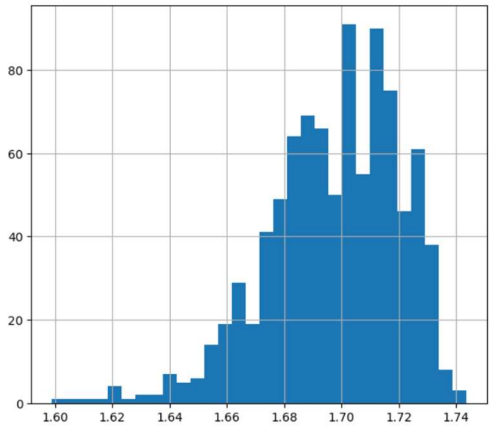


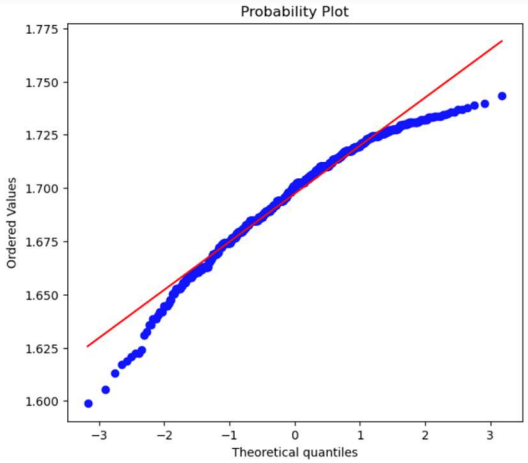
In [41]:
```
data['MaxHR_exp2'] = data['MaxHR']**(2)
diagnostic_plots(data, 'MaxHR_exp2')
```





In [42]:
```
data['MaxHR_exp3'] = data['MaxHR']**(0.333)
diagnostic_plots(data, 'MaxHR_exp3')
```





In [43]:
```
#Преобразованиея Бокса-Кокса
data['MaxHR_boxcox'], param = stats.boxcox(data['MaxHR'])
print('Оптимальное значение λ = {}'.format(param))
diagnostic_plots(data, 'MaxHR_boxcox')
```

```
Оптимальное значение λ = 6.875338794133852
```