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 ИУ5И-24М
Хэ Синьчэнь

Москва-2024 г.

Цель лабораторной работы:

изучение продвинутых способов предварительной обработки данных для дальнейшего формирования моделей.

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:
    i. устранение пропусков в данных;
    ii. кодирование категориальных признаков;
    iii. нормализация числовых признаков.

Устранение пропусков в данных.

```
[ ] #Устранение пропусков в данных.
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.impute import SimpleImputer
from sklearn.impute import MissingIndicator
from sklearn.impute import KNNImputer
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import Lasso
from sklearn.pipeline import Pipeline
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestRegressor
from sklearn.experimental import enable_iterative_imputer
from sklearn.impute import IterativeImputer
from IPython.display import Image
%matplotlib inline
sns.set(style="ticks")
```

```
[48] # Будем использовать только обучающую выборку
hdata_loaded = pd.read_csv('/content/Heart.csv', sep=',')
```

```
[49] hdata_loaded.shape
```

```
(303, 15)
```

```
[50] hdata = hdata_loaded
```

```
[51] #Удаление пропущенных значений
list(zip(hdata.columns, [i for i in hdata.dtypes]))
```

```
[('Unnamed: 0', dtype('int64')),
 ('Age', dtype('int64')),
 ('Sex', dtype('int64')),
 ('ChestPain', dtype('O')),
 ('RestBP', dtype('int64')),
 ('Chol', dtype('int64')),
 ('Fbs', dtype('int64')),
 ('RestECG', dtype('int64')),
 ('MaxHR', dtype('int64')),
 ('ExAng', dtype('int64')),
 ('Oldpeak', dtype('float64')),
 ('Slope', dtype('int64')),
 ('Ca', dtype('float64')),
 ('Thal', dtype('O')),
 ('AHD', dtype('O'))]
```

```
[52] # Колонки с пропусками
hcols_with_na = [c for c in hdata.columns if hdata[c].isnull().sum() > 0]
hcols_with_na
```

```
['Ca', 'Thal']
```

```
[53] hdata.shape
```

```
(303, 15)
```

```
[54] # Количество пропусков
[(c, hdata[c].isnull().sum()) for c in hcols_with_na]
```

```
[('Ca', 4), ('Thal', 2)]
```

```
[55] # Доля (процент) пропусков
[(c, hdata[c].isnull().mean()) for c in hcols_with_na]
```

```
[('Ca', 0.013201320132013201), ('Thal', 0.006600660066006601)]
```

Кодирование категориальных признаков

```
[62] #第二部分Кодирование категориальных признаков
     import numpy as np
     import pandas as pd
     import seaborn as sns
     import matplotlib.pyplot as plt
     %matplotlib inline
     sns.set(style="ticks")
```

```
[63] # Будем использовать только обучающую выборку
     data_loaded = pd.read_csv('/content/Heart.csv', sep=",")
```

```
[64] # размер набора данных
     data_loaded.shape
```

```
(303, 15)
```

```
[65] data_loaded.head()
```

	Unnamed: 0	Age	Sex	ChestPain	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	Oldpeak	Slope	Ca	Thal	AHD
0	1	63	1	typical	145	233	1	2	150	0	2.3	3	0.0	fixed	No
1	2	67	1	asymptomatic	160	286	0	2	108	1	1.5	2	3.0	normal	Yes
2	3	67	1	asymptomatic	120	229	0	2	129	1	2.6	2	2.0	reversable	Yes
3	4	37	1	nonanginal	130	250	0	0	187	0	3.5	3	0.0	normal	No
4	5	41	0	nontypical	130	204	0	2	172	0	1.4	1	0.0	normal	No

In [45]:
```
# Колонки с пропусками
cols_with_na = [c for c in data.columns if data[c].isnull().sum() > 0]
cols_with_na
```

Out[45]:
```
['Gender',
 'Married',
 'Dependents',
 'Self_Employed',
 'LoanAmount',
 'Loan_Amount_Term',
 'Credit_History']
```

In [46]:
```
# Доля (процент) пропусков
[(c, data[c].isnull().mean()) for c in cols_with_na]
```

Out[46]:
```
[('Gender', 0.021172638436482084),
 ('Married', 0.004885993485342019),
 ('Dependents', 0.024429967426710098),
 ('Self_Employed', 0.05211726384364821),
 ('LoanAmount', 0.035830618892508145),
 ('Loan_Amount_Term', 0.02280130293159609),
 ('Credit_History', 0.08143322475570032)]
```

```
##Датасет достаточно маленький, а доля пропущенных значений не очень большая. В таком случае можно поработать с внедрением пропущенных значений.
```

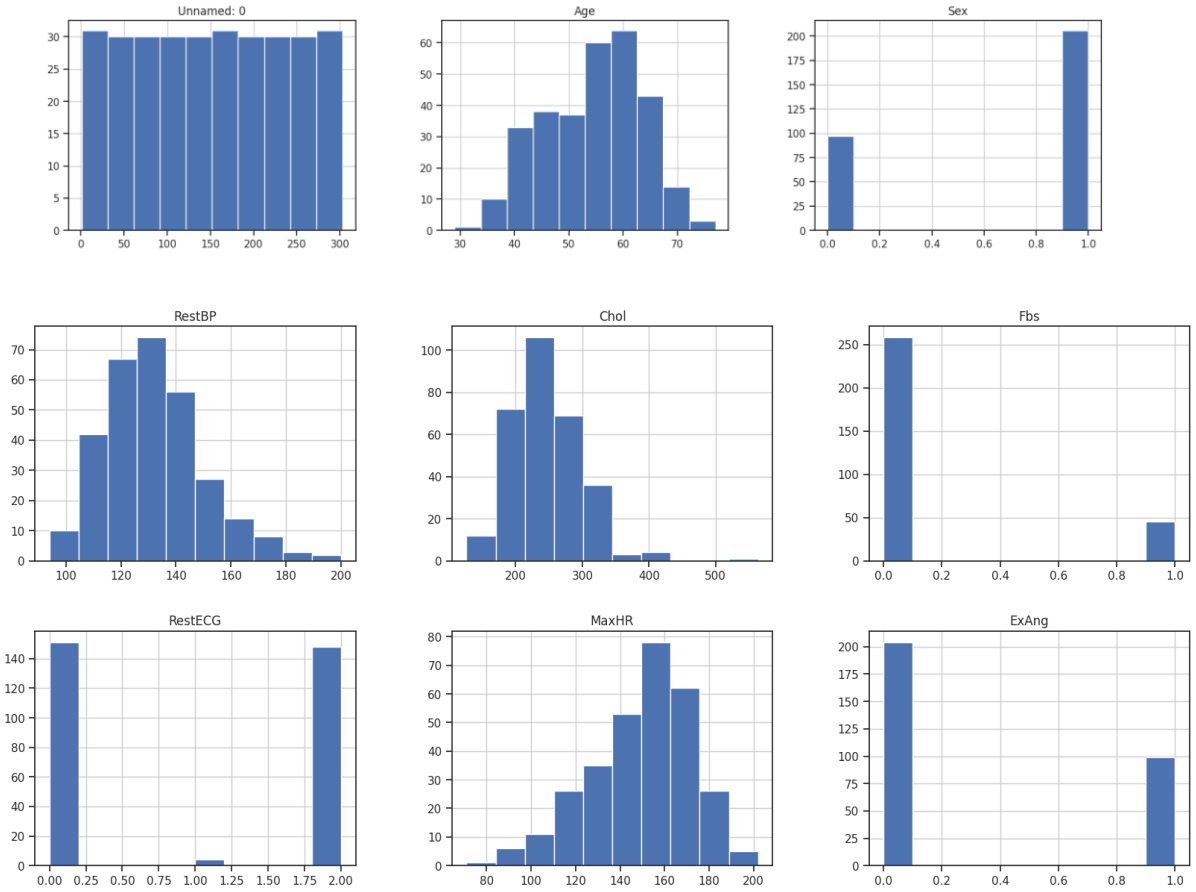
Нормализация числовых признаков

```
[74] import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
     import scipy.stats as stats
```

```
[75] 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()
```

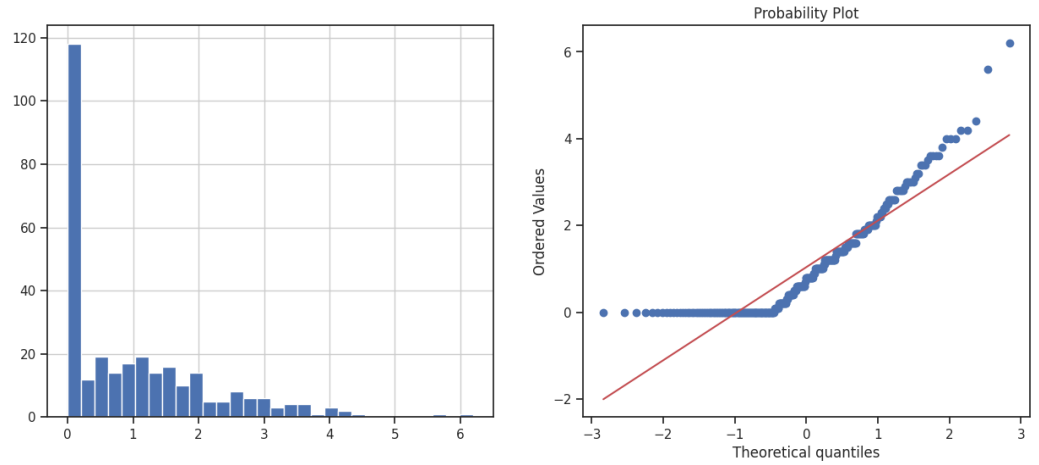
```
[76] # Будем использовать только обучающую выборку
     data = pd.read_csv('/content/Heart.csv', sep=',')
```

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



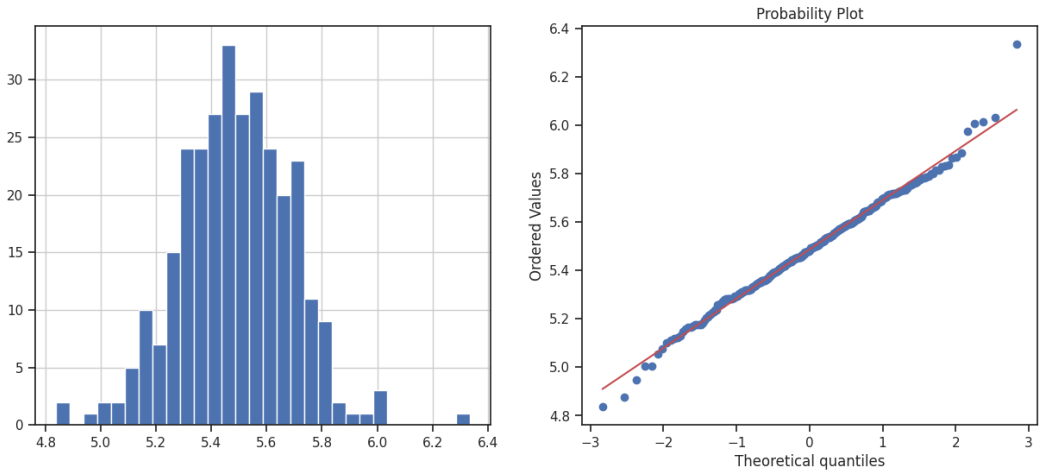
Исходное распределение

```
[79] diagnostic_plots(data, 'Oldpeak')
```



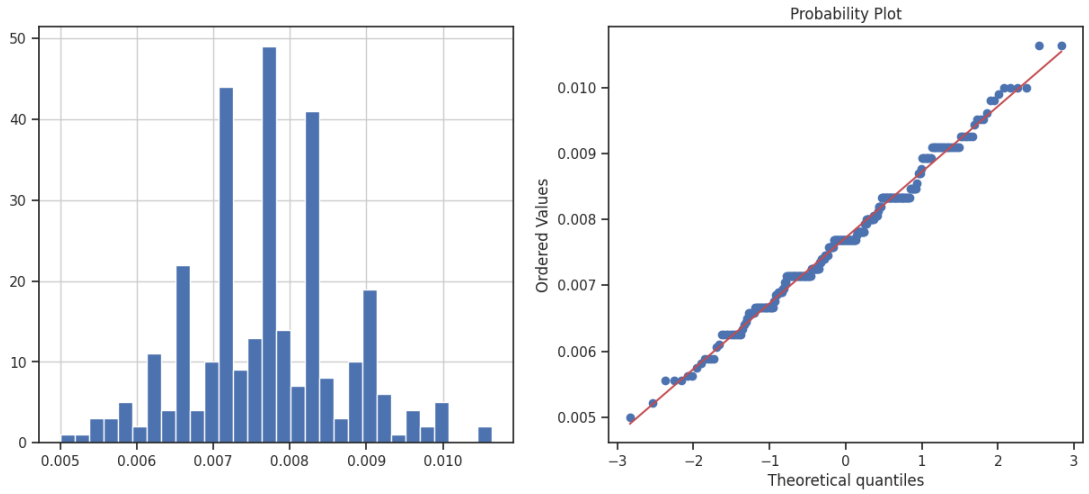
Логарифмическое преобразование

```
[81] data['Chol'] = np.log(data['Chol'])
     diagnostic_plots(data, 'Chol')
```



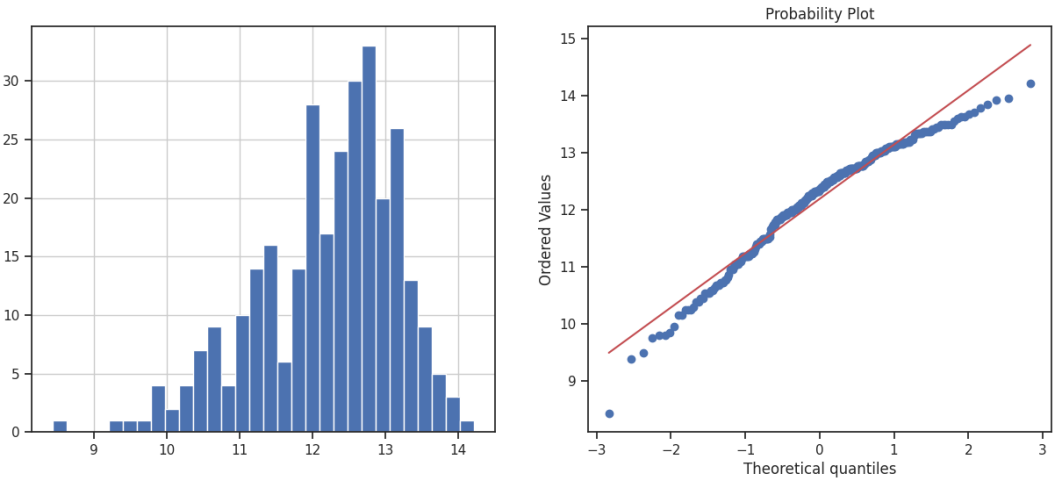
Обратное преобразование

```
[82] data['RestBP'] = 1 / (data['RestBP'])
     diagnostic_plots(data, 'RestBP')
```



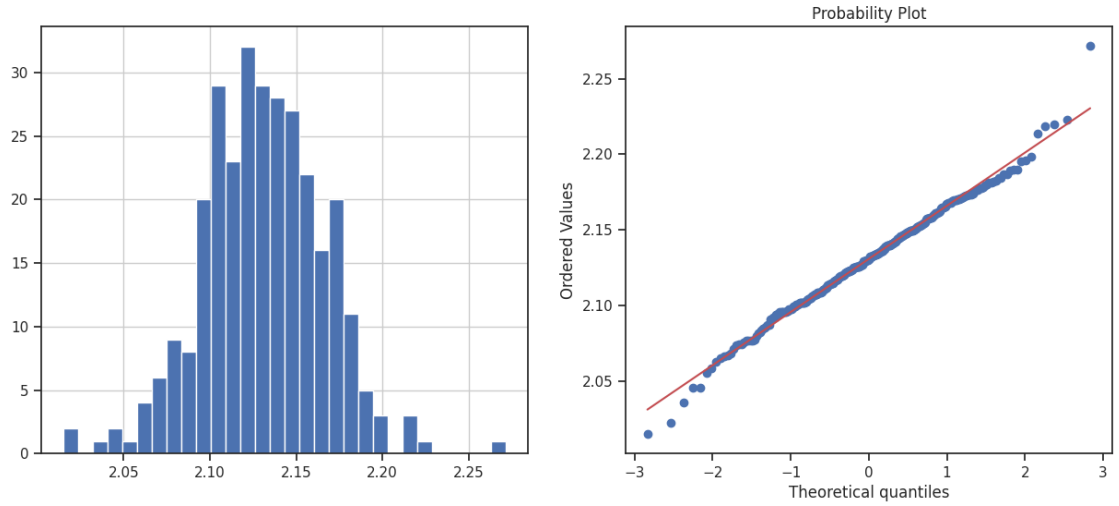
Квадратный корень

```
[83] data['MaxHR'] = data['MaxHR']**(1/2)
     diagnostic_plots(data, 'MaxHR')
```



Возведение в степень

```
[86] data['MaxHR_exp1'] = data['MaxHR']**(1/1.5)
     diagnostic_plots(data, 'MaxHR_exp1')
```



Преобразование Бокса-Кокса

```
data['MaxHR_boxcox'], param = stats.boxcox(data['MaxHR'])
print('Оптимальное значение λ = {}'.format(param))
diagnostic_plots(data, 'MaxHR_boxcox')
```

Преобразование Йео-Джонсона

```
# Необходимо преобразовать данные к действительному типу
data['MaxHR'] = data['MaxHR'].astype('float')
data['MaxHR_yeojohnson'], param = stats.yeojohnson(data['MaxHR'])
print('Оптимальное значение λ = {}'.format(param))
diagnostic_plots(data, 'MaxHR_yeojohnson')
```

```
Оптимальное значение λ = 0.7373660842633347
```