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 и системы управления и искусственный интеллект

КАФЕДРА Системы обработки информации и управления

Лабораторная работа №2 по курсу «Методы машинного обучения в автоматизированных системах обработки информации и управления»

Подготовили:

У Жун

ИУ5И-25М

20.04.2024

Проверил:

Гапанюк Ю. Е.

2024 г.

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

Набор данных о диабете в sklearn - один из стандартных наборов данных, используемых для машинного обучения и анализа данных.

Набор данных включает следующие функции (независимые переменные):

1. Возраст (age)
2. Пол (sex)
3. Индекс массы тела (BMI)
4. Артериальное давление (BP)
5. S1-S6: серия результатов анализа крови
6. Количественный показатель заболевания через год после диагностики (target)

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.datasets import load_diabetes
from sklearn.impute import SimpleImputer

diabetes = load_diabetes()
X = diabetes.data
y = diabetes.target
feature_names = diabetes.feature_names

df = pd.DataFrame(data=X, columns=feature_names)
df['target'] = y

np.random.seed(42)
missing_indices = np.random.choice(range(df.shape[0]), size=50, replace=False)
for col in df.columns:
    df.loc[missing_indices, col] = np.nan

missing_values = df.isnull().sum()
print("Количество пропущенных значений:\n", missing_values)

imputer = SimpleImputer(strategy='mean')
X_imputed = imputer.fit_transform(df.drop(columns=['target']))

df_imputed = pd.DataFrame(data=X_imputed, columns=feature_names)

plt.figure(figsize=(12, 6))
```

```
missing_values = df.isnull().sum()
print("Количество пропущенных значений:\n", missing_values)

imputer = SimpleImputer(strategy='mean')
X_imputed = imputer.fit_transform(df.drop(columns=['target']))

df_imputed = pd.DataFrame(data=X_imputed, columns=feature_names)

plt.figure(figsize=(12, 6))

for i, feature in enumerate(feature_names):
    plt.subplot(2, 5, i+1)
    sns.histplot(df[feature], color='blue', kde=True, label='До обработки', alpha=0.5)
    sns.histplot(df_imputed[feature], color='green', kde=True, label='После обработки', alpha=0.5)
    plt.title(feature)

plt.tight_layout()
plt.legend()
plt.show()
```

```
Количество пропущенных значений:
 age       50
sex        50
bmi        50
bp         50
s1         50
s2         50
s3         50
s4         50
s5         50
s6         50
target     50
dtype: int64
```

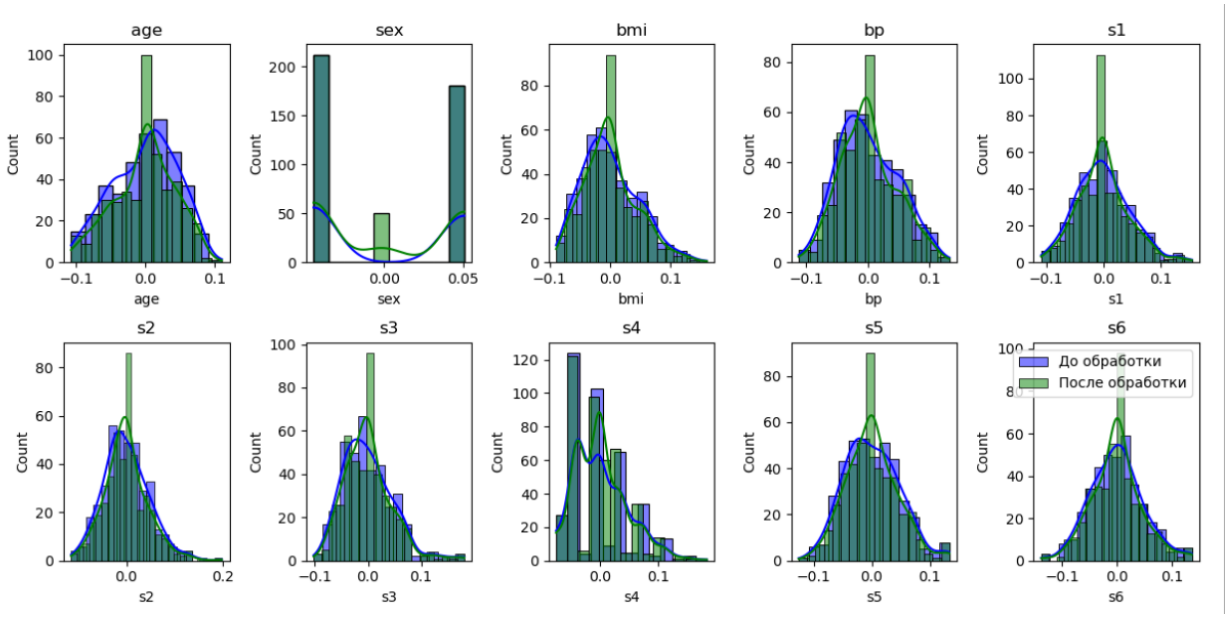


Рис.1-Устранение пропусков в данных

КОДИРОВАНИЕ КАТЕГОРИАЛЬНЫХ ПРИЗНАКОВ

Я использую кодирование категорий наборами бинарных значений - one-hot encoding.

One-hot encoding предполагает, что значение категории заменяется на отдельную колонку, которая содержит бинарные значения.

```
import pandas as pd
from sklearn.preprocessing import OneHotEncoder

data = pd.DataFrame({
    'gender': ['Male', 'Female', 'Female', 'Male', 'Female'],
    'city': ['Moscow', 'Paris', 'London', 'New York', 'Tokyo'],
    'age': [30, 25, 35, 40, 45]
})

encoder = OneHotEncoder(sparse_output=False)
encoded_data = encoder.fit_transform(data[['gender', 'city']])
encoded_df = pd.DataFrame(encoded_data, columns=encoder.get_feature_names_out(['gender', 'city']))
final_df = pd.concat([encoded_df, data[['age']]], axis=1)
print(final_df)
```

```
   gender_Female  gender_Male  city_London  city_Moscow  city_New York  \
0            0.0          1.0          0.0          1.0            0.0
1            1.0          0.0          0.0          0.0            0.0
2            1.0          0.0          1.0          0.0            0.0
3            0.0          1.0          0.0          0.0            1.0
4            1.0          0.0          0.0          0.0            0.0

   city_Paris  city_Tokyo  age
0         0.0         0.0   30
1         1.0         0.0   25
2         0.0         0.0   35
3         0.0         0.0   40
4         0.0         1.0   45
```

Рис. 2 – Кодирование категориальных признаков

НОРМАЛИЗАЦИЯ ЧИСЛОВЫХ ПРИЗНАКОВ.

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

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

from sklearn.datasets import load_diabetes
diabetes = load_diabetes()
X = diabetes.data
y = diabetes.target
feature_names = diabetes.feature_names

data = pd.DataFrame(data=X, columns=feature_names)
data['target'] = y


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

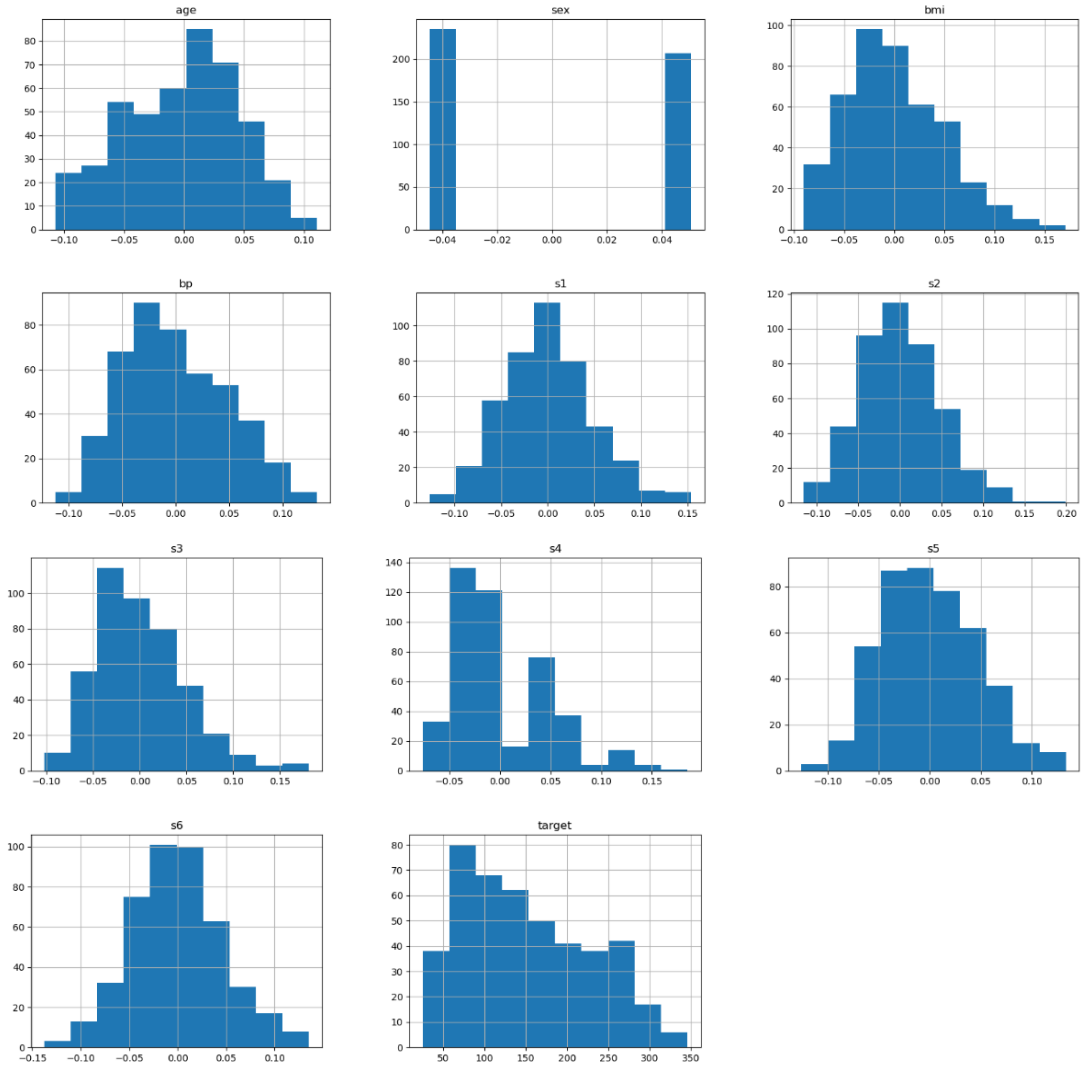
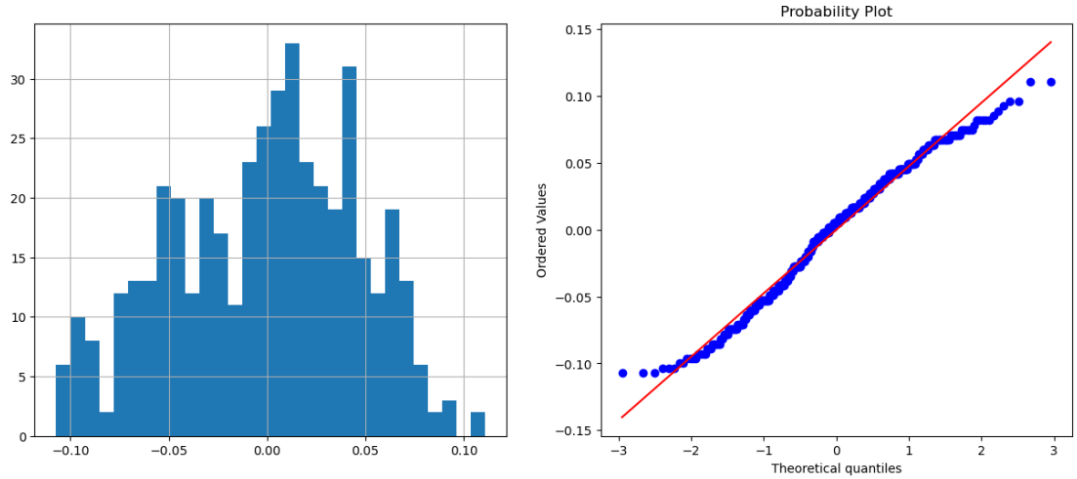


Рис.3– Загрузка и первичный анализ данных

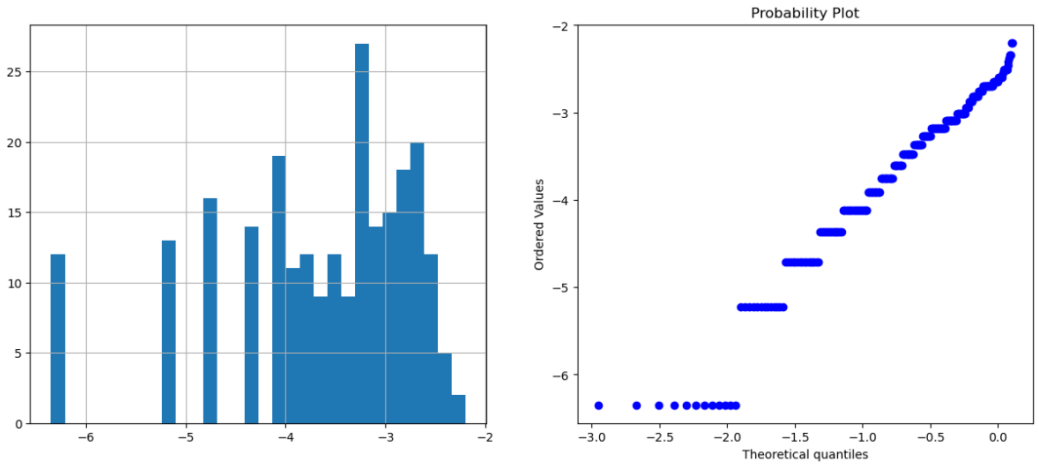
- Исходное распределение

```
diagnostic_plots(data, 'age')
```



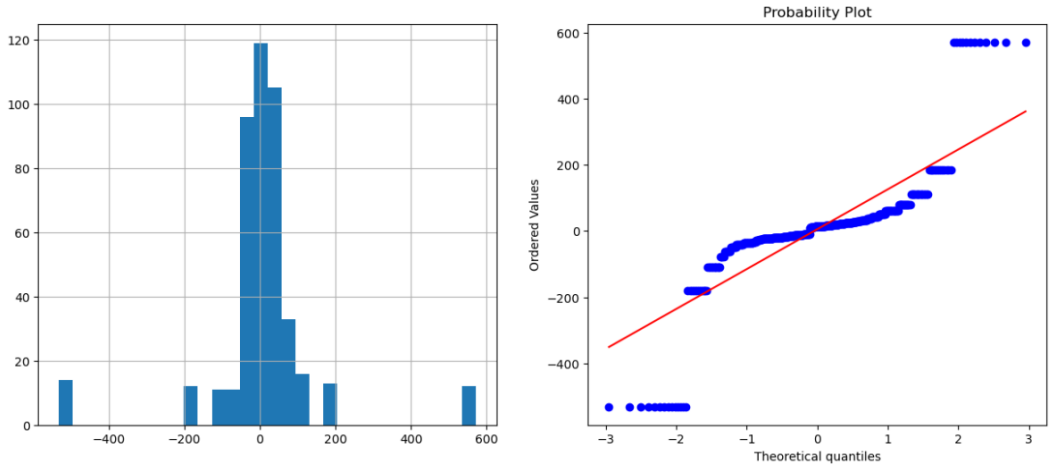
- Логарифмическое преобразование:

```
data['age_log'] = np.log(data['age'])
diagnostic_plots(data, 'age_log')
```



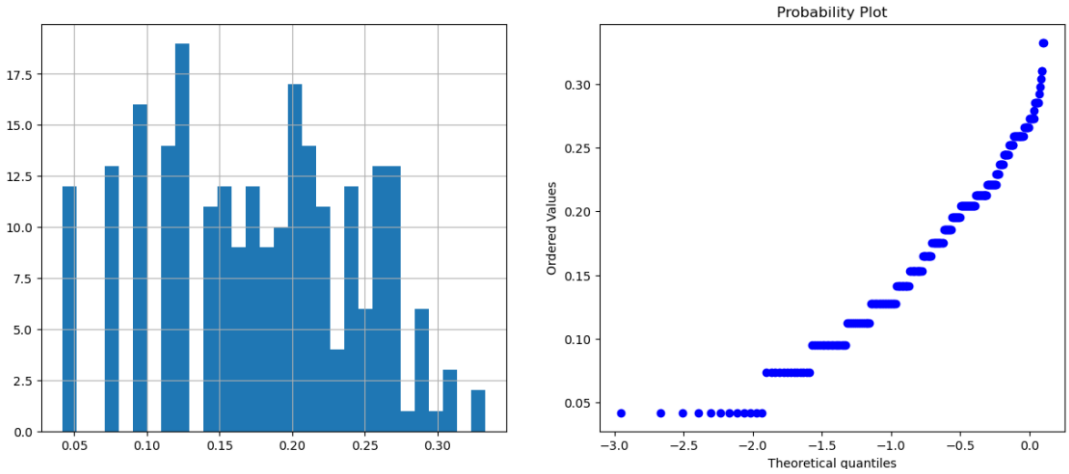
- Обратное преобразование

```
data['age_reciprocal'] = 1 / (data['age'])
diagnostic_plots(data, 'age_reciprocal')
```



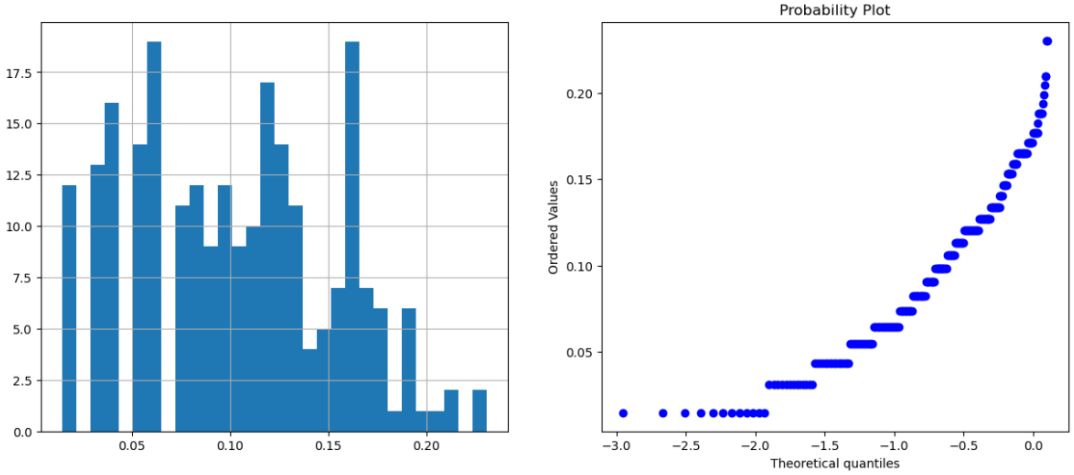
- Квадратный корень

```
data['age_sqr'] = data['age']**(1/2)
diagnostic_plots(data, 'age_sqr')
```



- Возведение в степень

```
data['age_exp'] = data['age']**(1/1.5)
diagnostic_plots(data, 'age_exp')
```



- Преобразование Йео-Джонсона

Очень хороший результат.

```
pt = PowerTransformer(method='yeo-johnson', standardize=False)
data['yeojohnson'] = pt.fit_transform(data[['age']])
diagnostic_plots(data, 'yeojohnson')
```