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:

Чжан Чжиси

ИУ5И-25М

01.05.2024

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

Используемый набор данных - классический набор данных "Взрослые", который обычно применяется для задач классификации и прогнозирования. Он содержит ряд сведений о характеристиках взрослых людей, а также целевую переменную, указывающую, превышает ли годовой доход каждого человека 50 000 долларов. Характеристики в наборе данных включают возраст, категорию работы, образование, семейное положение, профессию, расу, пол, доход от капитала, потери капитала и количество часов, отработанных в неделю.

```
#1
import numpy as np
import pandas as pd

url = "https://archive.ics.uci.edu/ml/machine-learning-databases/adult/adult.data"
column_names = ['age', 'workclass', 'fnlwgt', 'education', 'education-num', 'marital-status',
                'occupation', 'relationship', 'race', 'sex', 'capital-gain', 'capital-loss',
                'hours-per-week', 'native-country', 'income']
data = pd.read_csv(url, names=column_names, na_values=' ?')

data.head()
```

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	income
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States	<=50K
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States	<=50K
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States	<=50K
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States	<=50K
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	Cuba	<=50K

```
print("Размер набора данных: ", data.shape)

missing_values = data.isnull().sum()
print("\nКоличество отсутствующих значений: \n", missing_values)

print("\nТип данных: \n", data.dtypes)
```

```
Размер набора данных:  (32561, 15)

Количество отсутствующих значений:
 age                0
workclass          1836
fnlwgt                0
education             0
education-num         0
marital-status        0
occupation         1843
relationship          0
race                  0
sex                   0
capital-gain          0
capital-loss          0
hours-per-week        0
native-country      583
income                0
dtype: int64
```

```
Тип данных:
 age                int64
workclass          object
fnlwgt              int64
education          object
education-num       int64
marital-status     object
occupation         object
relationship       object
race               object
sex                object
capital-gain        int64
capital-loss        int64
hours-per-week      int64
native-country     object
income             object
dtype: object
```

```
data_cleaned = data.dropna()

for column in ['workclass', 'occupation', 'native-country']:
    mode_value = data_cleaned[column].mode()[0]
    data_cleaned[column].fillna(mode_value, inplace=True)

missing_values_cleaned = data_cleaned.isnull().sum()
print("Количество отсутствующих значений после обработки: \n", missing_values_cleaned)

data_cleaned.head()
```

```
Количество отсутствующих значений после обработки:
 age               0
workclass          0
fnlwgt             0
education          0
education-num      0
marital-status     0
occupation         0
relationship       0
race               0
sex                0
capital-gain       0
capital-loss       0
hours-per-week     0
native-country     0
income             0
dtype: int64
```

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	income
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States	<=50K
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States	<=50K
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States	<=50K
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States	<=50K
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	Cuba	<=50K

```
import matplotlib.pyplot as plt

def plot_hist_diff(old_ds, new_ds, cols):
    """
    Постройте график разницы между распределениями до и после удаления не
    """
    for c in cols:
        fig = plt.figure()
        ax = fig.add_subplot(111)
        ax.set_title('Field - ' + str(c))
        old_ds[c].hist(bins=50, ax=ax, density=True, color='green', alpha=0.5, label='Before')
        new_ds[c].hist(bins=50, ax=ax, color='blue', density=True, alpha=0.5, label='After')
        ax.legend()
        plt.show()

plot_hist_diff(data, data_cleaned, ['workclass', 'occupation', 'native-country'])
```

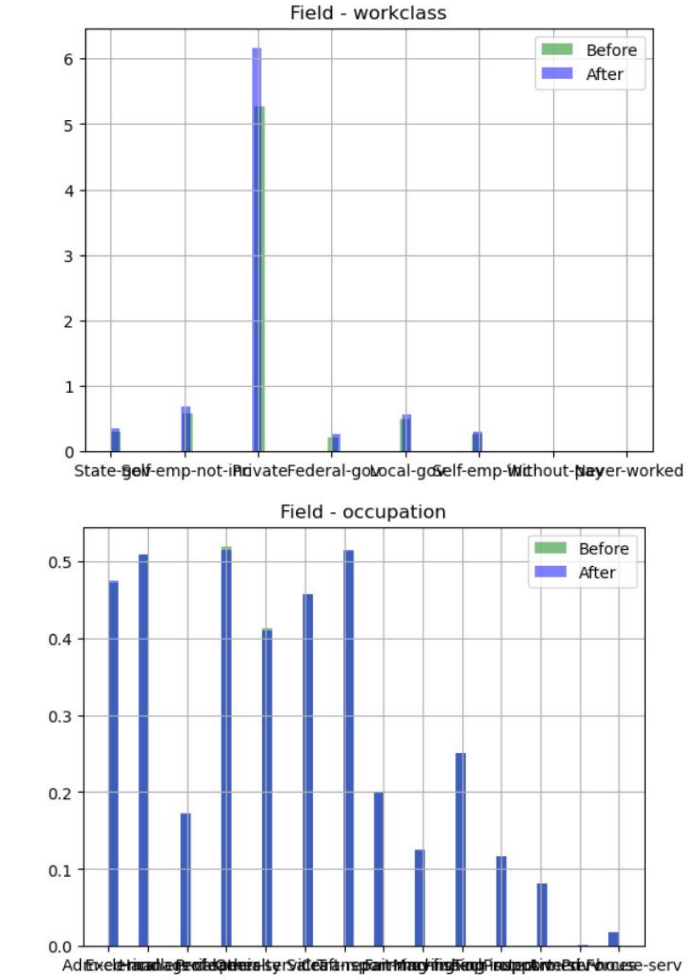

Рис.1-Устранение пропусков в данных

Кодирование категориальных признаков

```
#2
from sklearn.preprocessing import OneHotEncoder

categorical_features = ['workclass', 'marital-status', 'occupation', 'relationship', 'race', 'sex', 'native-country', 'income']

onehot_encoder = OneHotEncoder(sparse=False, drop='first')

encoded_features = onehot_encoder.fit_transform(data_cleaned[categorical_features])

encoded_column_names = onehot_encoder.get_feature_names_out(categorical_features)

encoded_df = pd.DataFrame(encoded_features, columns=encoded_column_names)

encoded_df.head()
```

	workclass_ Local-gov	workclass_ Private	workclass_ Self-emp-inc	workclass_ Self-emp-not-inc	workclass_ State-gov	workclass_ Without-pay	marital-status_ Married-AF-spouse	marital-status_ Married-civ-spouse	marital-status_ Married-spouse-absent	marital-status_ Never-married	...	native-country_ Puerto-Rico	native-country_ Scotland	native-country_ South	native-country_ Taiwan
0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	...	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	...	0.0	0.0	0.0	0.0
2	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
3	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	...	0.0	0.0	0.0	0.0
4	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	...	0.0	0.0	0.0	0.0

5 rows × 76 columns

Рис. 2 – Кодирование категориальных признаков

Нормализация числовых признаков.

```
#3
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
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)
    plt.title('Гистограмма')

    # Q-Q график
    plt.subplot(1, 2, 2)
    stats.probplot(df[variable], dist="norm", plot=plt)
    plt.title('Q-Q График')

    plt.show()

# Давайте имитируем процесс нормализации для переменной "age" в вашем набо
diagnostic_plots(data_cleaned, 'age')
```

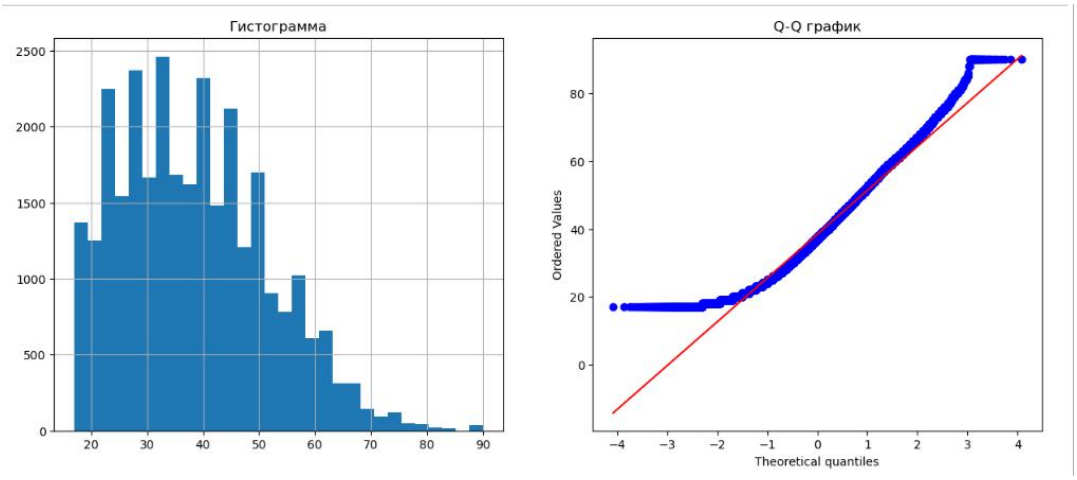

Рис.3- Исходное распределение

```
# Методы преобразования
# Логарифмическое преобразование
data_cleaned['age_log'] = np.log(data_cleaned['age'])
diagnostic_plots(data_cleaned, 'age_log')

# Обратное преобразование
data_cleaned['age_inverse'] = 1 / data_cleaned['age']
diagnostic_plots(data_cleaned, 'age_inverse')

# Квадратный корень
data_cleaned['age_sqrt'] = np.sqrt(data_cleaned['age'])
diagnostic_plots(data_cleaned, 'age_sqrt')

# Возведение в степень
data_cleaned['age_power'] = np.power(data_cleaned['age'], 2)
diagnostic_plots(data_cleaned, 'age_power')

# Преобразование Бокса-Кокса
from scipy.stats import boxcox
data_cleaned['age_boxcox'], _ = boxcox(data_cleaned['age'])
diagnostic_plots(data_cleaned, 'age_boxcox')

# Преобразование Йео-Джонсона
from scipy.stats import yeojohnson
data_cleaned['age_yeojohnson'], _ = yeojohnson(data_cleaned['age'])
diagnostic_plots(data_cleaned, 'age_yeojohnson')
```

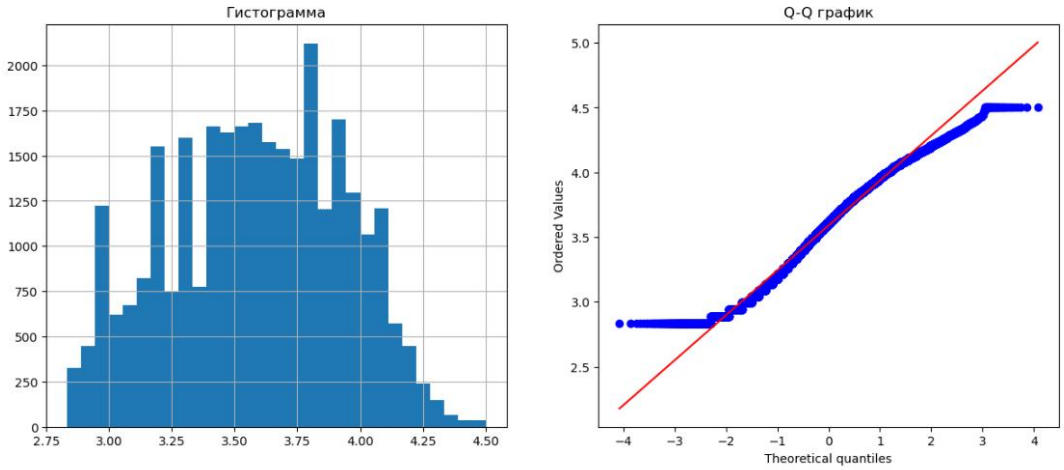

Рис.4- Логарифмическое преобразование

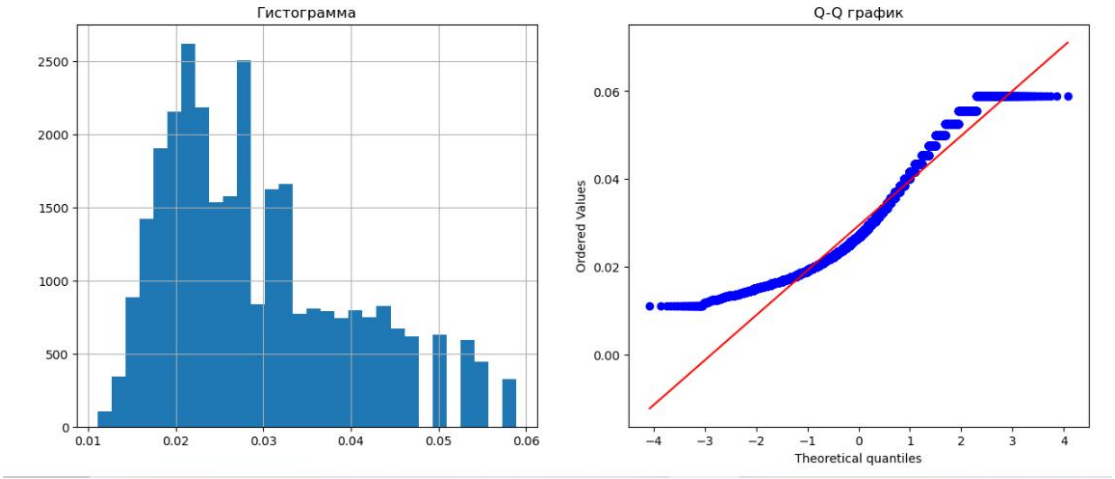

Рис.5- Обратное преобразование

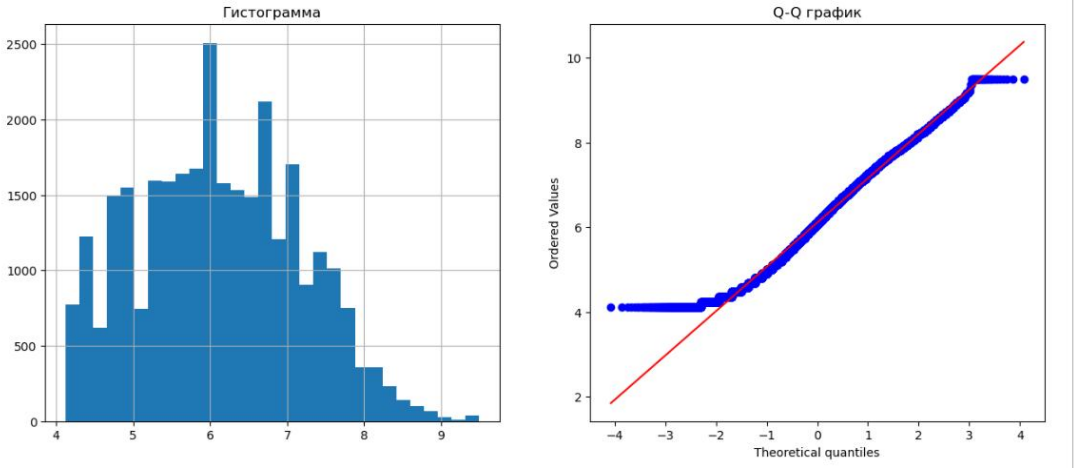

Рис.6- Квадратный корень

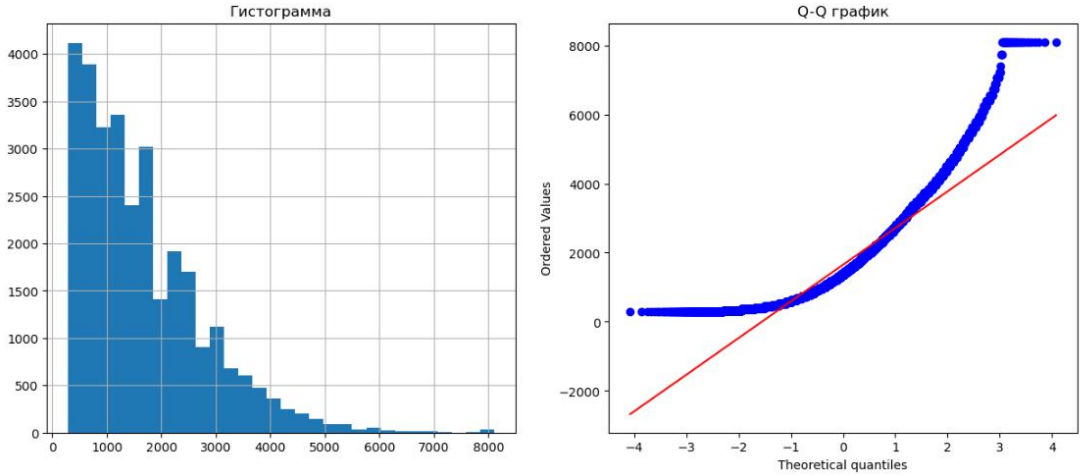

Рис.7- Возведение в степень

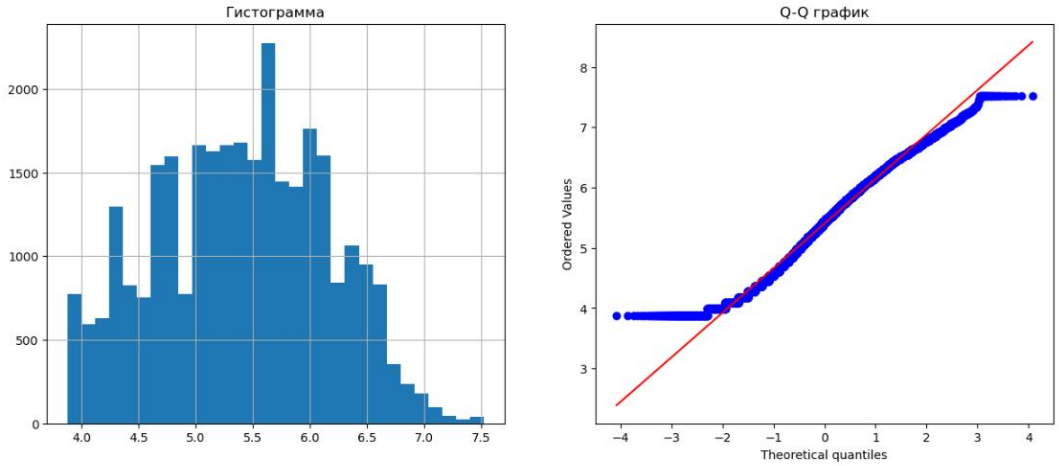

Рис.8- Преобразование Бокса-Кокса

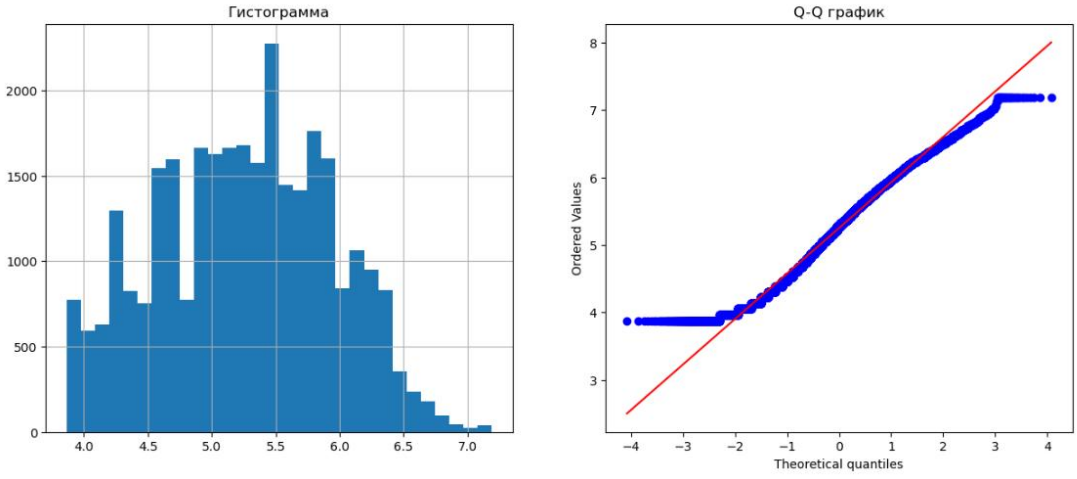

Рис.9- Преобразование Йео-Джонсона