РК1

Елхимова Ирина Сергеевна И5-22М

Номер по списку группы - 2

Вариант задачи №1 - 2

Для набора данных проведите кодирование одного (произвольного) категориального признака с использованием метода "target (mean) encoding".

Вариант задачи №2 - 22

Для набора данных проведите масштабирование данных для одного (произвольного) числового признака с использованием масштабирования по максимальному значению.

Дополнительное задание

Для произвольной колонки данных построить гистограмму.

In [26]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as stats
%matplotlib inline
sns.set(style="ticks")
from sklearn.impute import SimpleImputer
from sklearn.impute import MissingIndicator
```

In [28]:
```
data = pd.read_csv('heart2.csv')
```

In [29]:
```
data.head()
```

Out[29]:

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_gluc
0	9046	Male	NaN	0	1	Yes	Private	Urban	
1	51676	Female	61.0	0	0	Yes	Self-employed	Rural	
2	31112	Male	80.0	0	1	Yes	Private	Rural	
3	60182	Female	49.0	0	0	Yes	Private	Urban	
4	1665	Female	NaN	1	0	Yes	Self-employed	Rural	

Задача 1 (2)

Для набора данных проведите кодирование одного (произвольного) категориального признака с использованием метода "target (mean) encoding".

In [30]:
```
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[30]:
```
[('id', ('int64', 0)),
 ('gender', ('object', 0)),
 ('age', ('float64', 16)),
 ('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))]
```

In [31]:
```
# процент пропусков
[(c, data[c].isnull().mean()) for c in data.columns]
```

Out[31]:
```
[('id', 0.0),
 ('gender', 0.0),
 ('age', 0.0031311154598825833),
 ('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 [32]:
```
# Заполним пропуски bmi средними значениями
def impute_na(df, variable, value):
    df[variable].fillna(value, inplace=True)
impute_na(data, 'bmi', data['bmi'].mean())
```

In [33]:
```
# Удалим данные, где возраст незаполнен, так как таких данных мало, и удаление не повлия
data.dropna(subset=['age'], inplace=True)
```

In [34]:
```
data.isnull().sum()
```

Out[34]:
```
id                   0
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 [35]:
```
#TargetEncoder
from category_encoders.target_encoder import TargetEncoder as ce_TargetEncoder
```

In [36]:
```
ce_TargetEncoder1 = ce_TargetEncoder()
data_MEAN_ENC = ce_TargetEncoder1.fit_transform(data[data.columns.difference(['stroke'])
```

In [37]:
```
data_MEAN_ENC.head()
```

Out[37]:

	Residence_type	age	avg_glucose_level	bmi	ever_married	gender	heart_disease	hypertens
1	0.044258	61.0	202.21	28.893237	0.063136	0.045896	0	
2	0.044258	80.0	105.92	32.500000	0.063136	0.048387	1	
3	0.049497	49.0	171.23	34.400000	0.063136	0.045896	0	
5	0.049497	81.0	186.21	29.000000	0.063136	0.048387	0	
6	0.044258	74.0	70.09	27.400000	0.063136	0.048387	1	

In [38]:
```
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 [39]:
```
check_mean_encoding('gender')
```

```
Female - 0.04589614740368509
Male - 0.04838709677419355
Other - 0.0
```

In [40]:
```
check_mean_encoding('gender')
```

```
Female - 0.04589614740368509
Male - 0.04838709677419355
Other - 0.0
```

In [41]:
```
check_mean_encoding('work_type')
```

```
Self-employed - 0.07607361963190185
Private - 0.04874699622382424
Govt_job - 0.0502283105022831
children - 0.002911208151382824
Never_worked - 0.0
```

In [42]:
```
data = data.drop('id', 1)
data.head()
```

```
/var/folders/z0/w0jmy3853hl1_sp6r8x9hswm0000gn/T/ipykernel_2576/3892771371.py:1: FutureW
arning: In a future version of pandas all arguments of DataFrame.drop except for the arg
ument 'labels' will be keyword-only.
  data = data.drop('id', 1)
```

Out[42]:

	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_leve
1	Female	61.0	0	0	Yes	Self-employed	Rural	202.2
2	Male	80.0	0	1	Yes	Private	Rural	105.9
3	Female	49.0	0	0	Yes	Private	Urban	171.2
5	Male	81.0	0	0	Yes	Private	Urban	186.2
6	Male	74.0	1	1	Yes	Private	Rural	70.0

Задача 2 (22)

Для набора данных проведите масштабирование данных для одного (произвольного) числового признака с использованием масштабирования по максимальному значению.

In [43]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MaxAbsScaler
```

In [44]:
```
data.describe()
```

Out[44]:

	age	hypertension	heart_disease	avg_glucose_level	bmi	stroke
count	5094.000000	5094.000000	5094.000000	5094.000000	5094.000000	5094.000000
mean	43.182960	0.097173	0.053592	106.074751	28.886542	0.046918
std	22.601491	0.296222	0.225234	45.216297	7.697727	0.211484
min	0.080000	0.000000	0.000000	55.120000	10.300000	0.000000
25%	25.000000	0.000000	0.000000	77.265000	23.800000	0.000000
50%	45.000000	0.000000	0.000000	91.850000	28.400000	0.000000
75%	61.000000	0.000000	0.000000	114.017500	32.800000	0.000000
max	82.000000	1.000000	1.000000	271.740000	97.600000	1.000000

In [45]:
```
X_ALL = data.drop(['stroke', 'gender', 'ever_married', 'work_type','Residence_type','smo
```

In [46]:
```
# Функция для восстановления датафрейма
# на основе масштабированных данных
def arr_to_df(arr_scaled):
    res = pd.DataFrame(arr_scaled, columns=X_ALL.columns)
    return res
```

In [47]:
```
# Разделим выборку на обучающую и тестовую
X_train, X_test, y_train, y_test = train_test_split(X_ALL, data['stroke'],
                                                    test_size=0.2,
                                                    random_state=1)
# Преобразуем массивы в DataFrame
X_train_df = arr_to_df(X_train)
X_test_df = arr_to_df(X_test)

X_train_df.shape, X_test_df.shape
```

Out[47]:
```
((4075, 5), (1019, 5))
```

In [48]:
```
cs31 = MaxAbsScaler()
data_cs31_scaled_temp = cs31.fit_transform(X_ALL)
# формируем DataFrame на основе массива
data_cs31_scaled = arr_to_df(data_cs31_scaled_temp)
data_cs31_scaled.describe()
```

Out[48]:

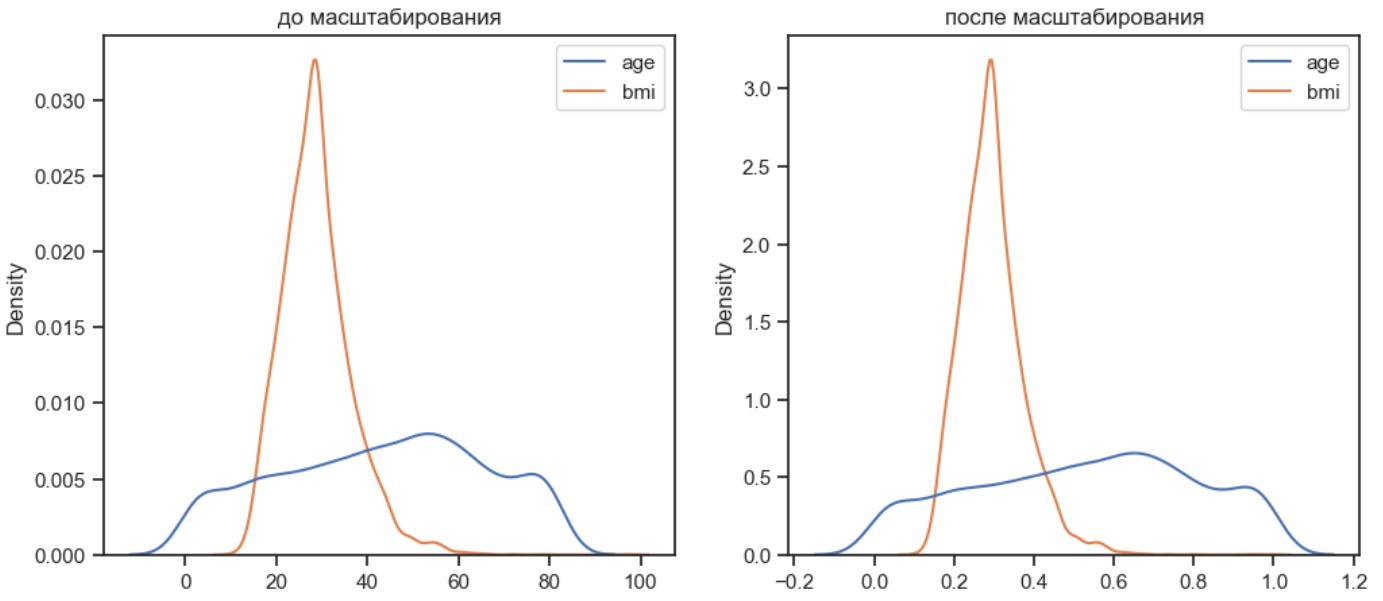
	age	hypertension	heart_disease	avg_glucose_level	bmi
count	5094.000000	5094.000000	5094.000000	5094.000000	5094.000000
mean	0.526621	0.097173	0.053592	0.390354	0.295969
std	0.275628	0.296222	0.225234	0.166395	0.078870
min	0.000976	0.000000	0.000000	0.202841	0.105533
25%	0.304878	0.000000	0.000000	0.284334	0.243852
50%	0.548780	0.000000	0.000000	0.338007	0.290984

75%	0.743902	0.000000	0.000000	0.419583	0.336066
max	1.000000	1.000000	1.000000	1.000000	1.000000

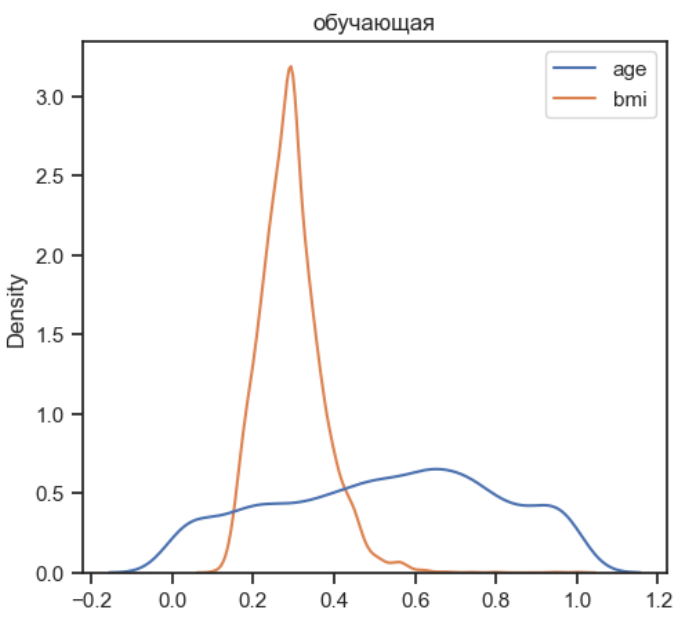
In [49]:
```
cs32 = MaxAbsScaler()
cs32.fit(X_train)
data_cs32_scaled_train_temp = cs32.transform(X_train)
data_cs32_scaled_test_temp = cs32.transform(X_test)
# формируем DataFrame на основе массива
data_cs32_scaled_train = arr_to_df(data_cs32_scaled_train_temp)
data_cs32_scaled_test = arr_to_df(data_cs32_scaled_test_temp)
```

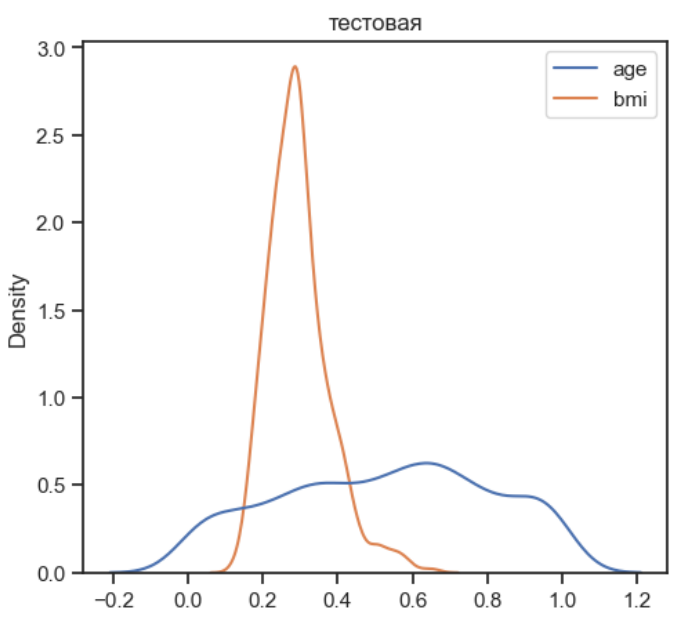
In [51]:
```
# Построение плотности распределения
def draw_kde(col_list, df1, df2, label1, label2):
    fig, (ax1, ax2) = plt.subplots(
        ncols=2, figsize=(12, 5))
    # первый график
    ax1.set_title(label1)
    sns.kdeplot(data=df1[col_list], ax=ax1)
    # второй график
    ax2.set_title(label2)
    sns.kdeplot(data=df2[col_list], ax=ax2)
    plt.show()
```

In [52]:
```
draw_kde(['age', 'bmi'], data, data_cs31_scaled, 'до масштабирования', 'после масштабиро
```



In [55]:
```
draw_kde(['age', 'bmi'], data_cs32_scaled_train, data_cs32_scaled_test, 'обучающая', 'те
```

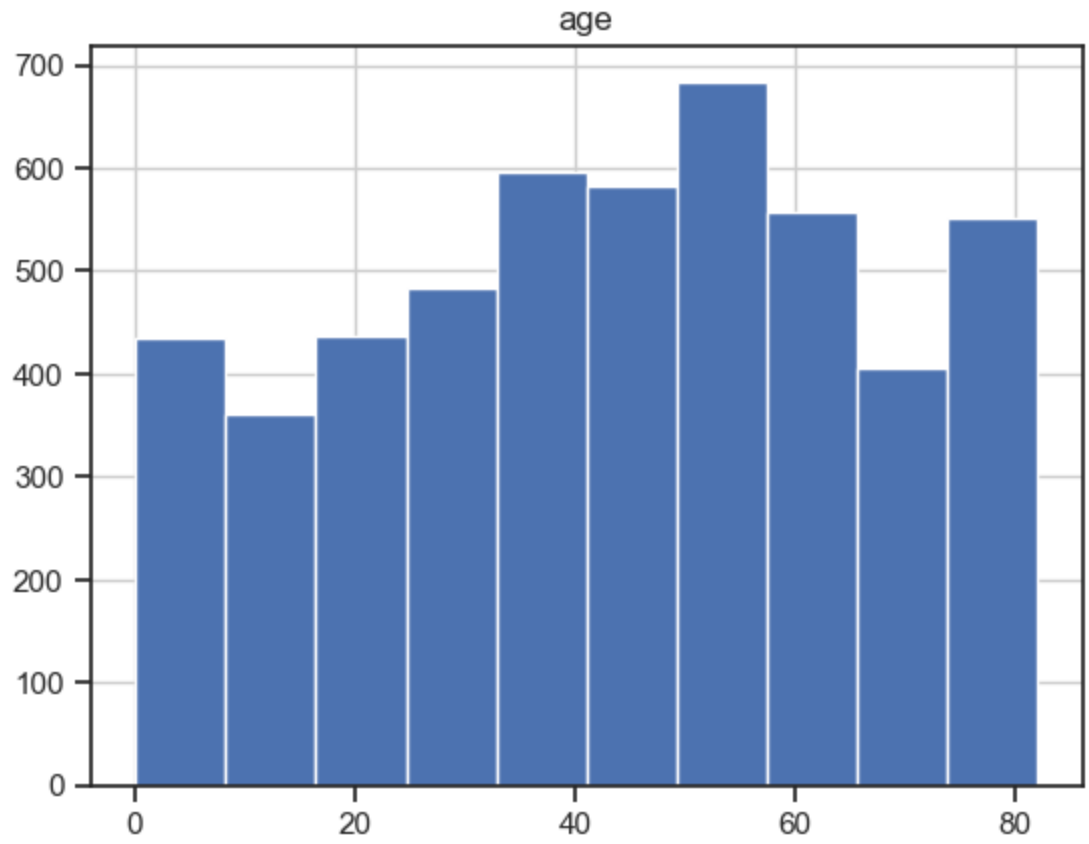




Дополнительное задание

Для произвольной колонки данных построить гистограмму.

In [54]:
```
data.hist('age')
plt.show()
```



In [ ]: