Рубежный контроль №1

ИУ5-24М

Желтова Александра

Вариант **3**

Для студентов группы ИУ5-24М - для произвольной колонки данных построить график "Скрипичная диаграмма (violin plot)".

Задача №3.

Для набора данных проведите кодирование одного (произвольного) категориального признака с использованием метода "weight of evidence (WoE) encoding".

Задача №23.

Для набора данных для одного (произвольного) числового признака проведите обнаружение и удаление выбросов на основе правила трех сигм.

---

[6]

5 сек.

```
pip install category_encoders
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting category_encoders
  Downloading category_encoders-2.6.0-py2.py3-none-any.whl (81 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 81.2/81.2
KB 4.7 MB/s eta 0:00:00
Requirement already satisfied: pandas>=1.0.5 in /usr/local/lib/python3.9/dist-packages (from
category_encoders) (1.4.4)
Requirement already satisfied: statsmodels>=0.9.0 in /usr/local/lib/python3.9/dist-packages (from
category_encoders) (0.13.5)
Requirement already satisfied: numpy>=1.14.0 in /usr/local/lib/python3.9/dist-packages (from
category_encoders) (1.22.4)
Requirement already satisfied: scipy>=1.0.0 in /usr/local/lib/python3.9/dist-packages (from
category_encoders) (1.10.1)
Requirement already satisfied: patsy>=0.5.1 in /usr/local/lib/python3.9/dist-packages (from
category_encoders) (0.5.3)
Requirement already satisfied: scikit-learn>=0.20.0 in /usr/local/lib/python3.9/dist-packages (from
category_encoders) (1.2.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.9/dist-packages (from
pandas>=1.0.5->category_encoders) (2022.7.1)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.9/dist-packages (from
pandas>=1.0.5->category_encoders) (2.8.2)
Requirement already satisfied: six in /usr/local/lib/python3.9/dist-packages (from patsy>=0.5.1-
>category_encoders) (1.16.0)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.9/dist-packages (from scikit-
learn>=0.20.0->category_encoders) (1.1.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.9/dist-packages (from scikit-
learn>=0.20.0->category_encoders) (3.1.0)
Requirement already satisfied: packaging>=21.3 in /usr/local/lib/python3.9/dist-packages (from
statsmodels>=0.9.0->category_encoders) (23.0)
Installing collected packages: category_encoders
Successfully installed category_encoders-2.6.0
```

[123]

0 сек.

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
sns.set(style="ticks")
from category_encoders.woe import WOEEncoder as ce_WOEEncoder
```

[124]

2 сек.

```
from google.colab import drive
drive.mount('/content/drive')
```

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).

Загрузка набора данных

[125]

0 сек.

```
data = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/MMO/sport.csv', sep=",")
# Первые 5 строк датасета
data = data.drop(['ID'], axis=1)
data.head()
```

	Name	Sex	Age	Teal	NOC	Games	Year	Season	City	Sport	Event	Medal
0	A Dijiang	1	24.0	China	CHN	1992 Summer	1992	Summer	Barcelona	Basketball	Basketball Men's Basketball	NaN
1	A Lamusi	1	23.0	China	CHN	2012 Summer	2012	Summer	London	Judo	Judo Men's Extra-Lightweight	NaN
2	Gunnar Nielsen Aaby	1	24.0	Denmark	DEN	1920 Summer	1920	Summer	Antwerpen	Football	Football Men's Football	NaN
3	Edgar Lindenau Aabye	1	34.0	Denmark/Sweden	DEN	1900 Summer	1900	Summer	Paris	Tug-Of-War	Tug-Of-War Men's Tug-Of-War	Gold
4	Jyri Tapani Aalto	1	31.0	Finland	FIN	2000 Summer	2000	Summer	Sydney	Badminton	Badminton Men's Singles	NaN

Кодирование категориального признака

```
ce_WOEEncoder1 = ce_WOEEncoder()
data_WOE_ENC = ce_WOEEncoder1.fit_transform(data[data.columns.difference(['Sex'])], data['Sex'])
data_WOE_ENC
```

	Age	City	Event	Games	Medal	NOC	Name	Season	Sport	Teal	Year
0	24.0	0.884288	0.373462	0.884288	0.153897	-0.725150	0.000000	0.020748	-0.319685	-0.725150	1992
1	23.0	-0.106111	-0.725150	-1.371777	0.153897	-0.725150	0.000000	0.020748	-0.032003	-0.725150	2012
2	24.0	0.255679	1.066610	0.255679	0.153897	-0.725150	0.000000	0.020748	0.373462	0.000000	1920
3	34.0	-1.130615	0.000000	-1.823762	0.022064	-0.725150	0.000000	0.020748	0.000000	0.000000	1900
4	31.0	-0.175104	0.000000	-0.175104	0.153897	0.335722	0.000000	0.020748	0.000000	0.335722	2000
...	...	...	...	...	...	...	...	...	...	...	...
286	19.0	-0.319685	-0.437468	-0.319685	0.153897	-1.130615	-0.725150	0.020748	-0.186153	-1.130615	1988
287	23.0	0.884288	-0.437468	0.884288	0.153897	-1.130615	-0.725150	0.020748	-0.186153	-1.130615	1992
288	22.0	-1.743719	-2.922374	-1.743719	0.153897	-1.130615	-2.922374	0.020748	-0.619789	-1.130615	2008
289	26.0	-0.106111	-2.922374	-1.371777	0.153897	-1.130615	-2.922374	0.020748	-0.619789	-1.130615	2012
290	25.0	0.884288	-0.437468	0.884288	0.153897	0.000000	0.000000	0.020748	-0.186153	0.000000	1992

291 rows × 11 columns

```
data['NOC'].unique()
```

```
array(['CHN', 'DEN', 'FIN', 'NOR', 'ROU', 'NED', 'EST', 'MAR', 'ESP',
       'FRA', 'EGY', 'IRI', 'BUL', 'ITA', 'CHA', 'AZE', 'SUD', 'RUS',
       'ARG', 'CUB', 'CMR', 'TUR', 'CHI', 'GRE', 'MEX', 'URS', 'NCA',
       'HUN', 'NGR', 'ALG', 'KUW', 'BRN', 'PAK', 'IRQ', 'UAR', 'LIB',
       'QAT', 'MAS', 'GER', 'RSA', 'CAN', 'USA', 'TAN', 'TUN', 'LBA',
       'DJI'], dtype=object)
```

```
[130] data_WOE_ENC['NOC'].unique()
```

```
array([-0.72514991,  0.33572205, -0.35742513,  0.        , -0.72514991,
       -2.22922731, -1.8237622 , -0.11901411,  0.12214795,  0.16036916,
       -0.03200273,  0.37346238, -1.8237622 , -3.43320011, -1.41829709,
       -0.21432429, -1.13061502,  1.17197007])
```

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

            good = filter_data_ones / data_ones
            bad = filter_data_zeros / data_zeros

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

```
check_woe_encoding('Medal')
```

```
Gold - 0.33135713595444244
Bronze - -1.0185695809945734
Silver - -1.8658674413817768
```

Обнаружение и удаление выбросов на основе правила трех сигм

[148]

0 сек.

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import scipy.stats as stats
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")
```

[149]

0 сек.

```
def diagnostic_plots(df, variable, title):
    fig, ax = plt.subplots(figsize=(10,7))
    # гистограмма
    plt.subplot(2, 2, 1)
    df[variable].hist(bins=30)
    ## Q-Q plot
    plt.subplot(2, 2, 2)
    stats.probplot(df[variable], dist="norm", plot=plt)
    # ящик с усами
```

```
plt.subplot(2, 2, 3)
sns.violinplot(x=df[variable])
# ящик с усами
plt.subplot(2, 2, 4)
sns.boxplot(x=df[variable])
fig.suptitle(title)
plt.show()
```

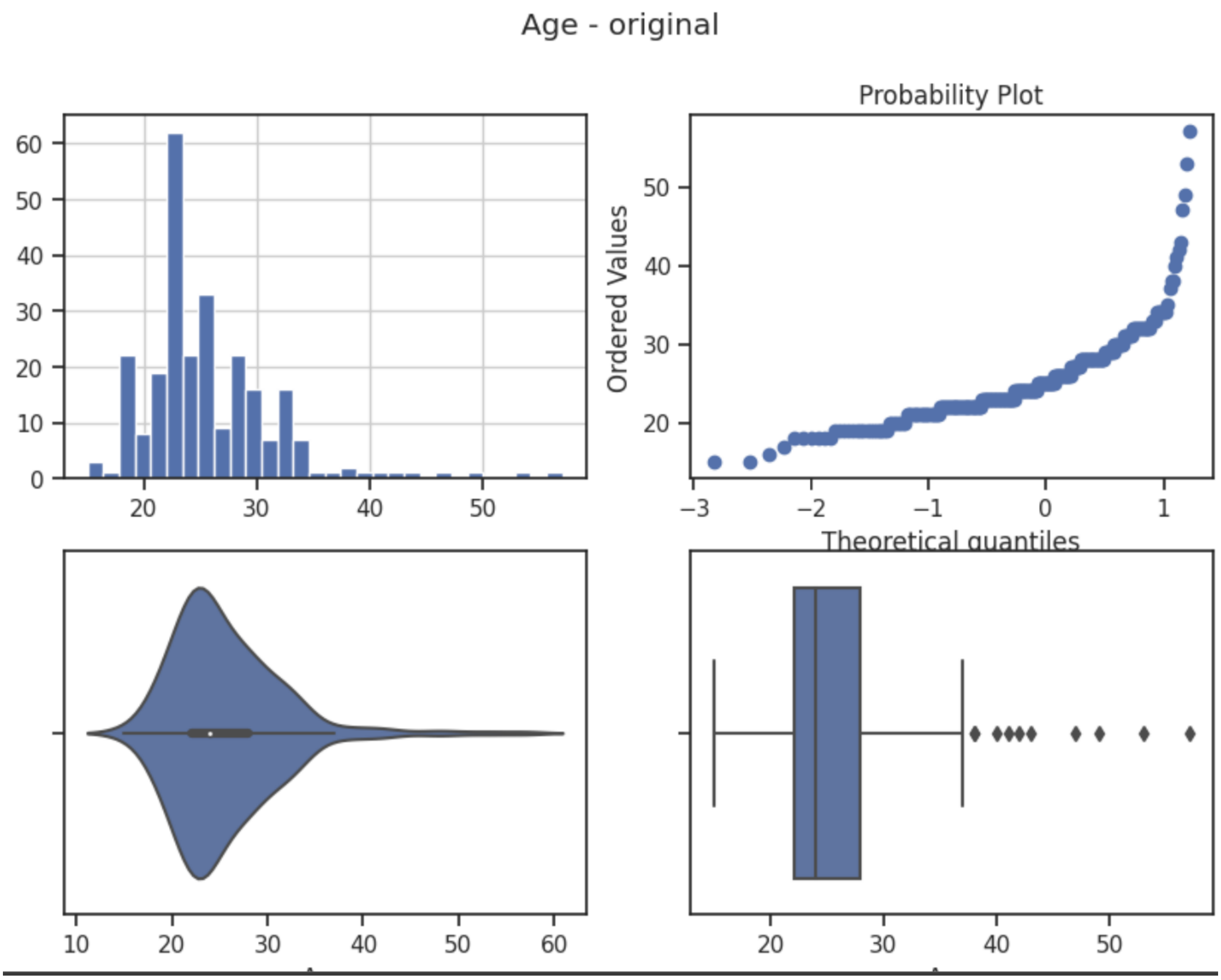
[150]

3 сек.

```
diagnostic_plots(data, 'Age', 'Age - original')
```

[151]

0 сек.




```
# Функция вычисления верхней и нижней границы выбросов
def get_outlier_boundaries(df, col):

  K1 = 3
  lower_boundary = df[col].mean() - (K1 * df[col].std())
  upper_boundary = df[col].mean() + (K1 * df[col].std())
```

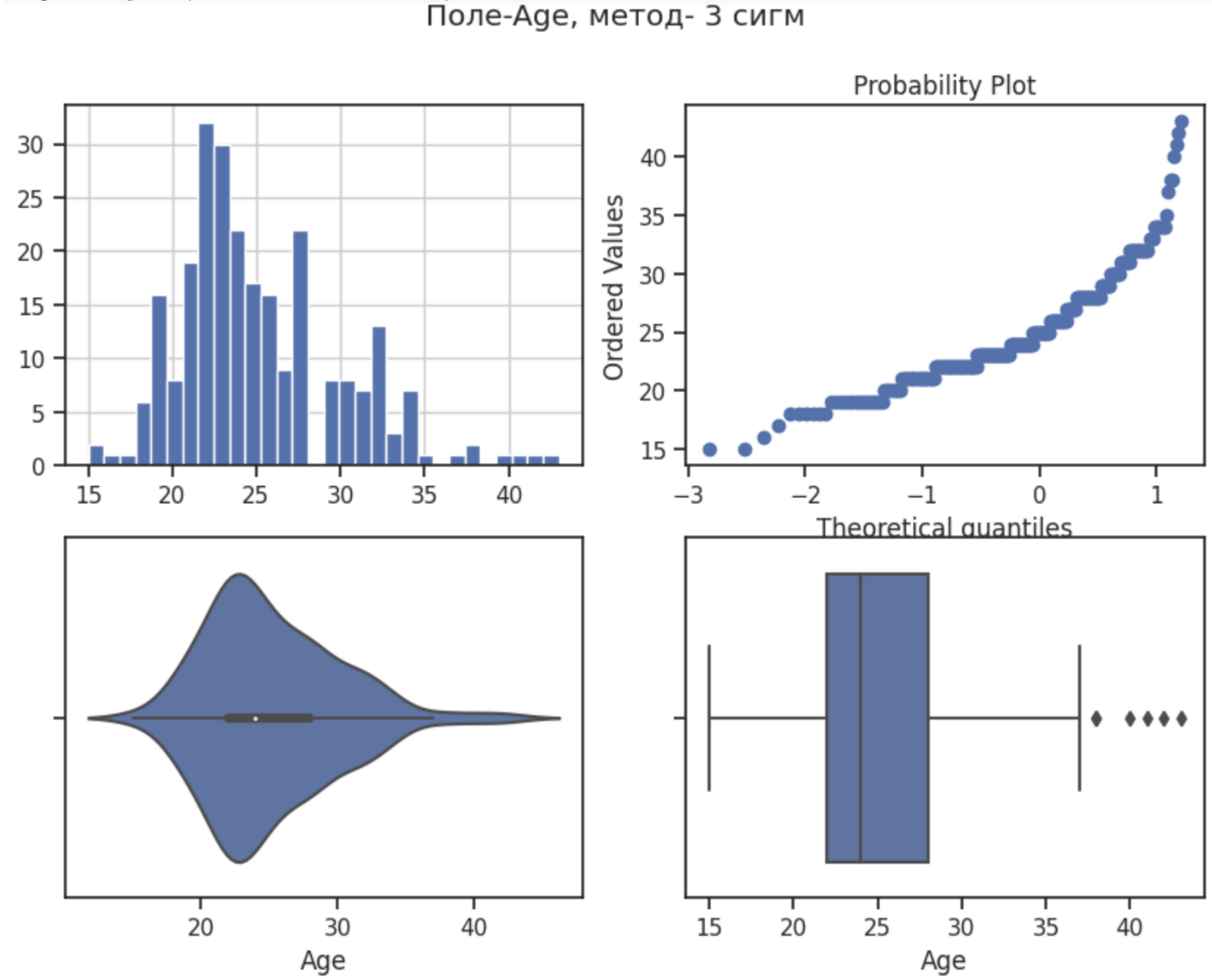
```
    return lower_boundary, upper_boundary
```

Удаление выбросов

[156]

1 сек.

```
col = 'Age'
# Вычисление верхней и нижней границы
lower_boundary, upper_boundary = get_outlier_boundaries(data, col)
# Флаги для удаления выбросов
outliers_temp = np.where(data[col] > upper_boundary, True,
                np.where(data[col] < lower_boundary, True, False))
# Удаление данных на основе флага
data_trimmed = data.loc[~(outliers_temp), ]
title = 'Поле-{}, метод- 3 сигм'.format(col,data_trimmed.shape[0])
diagnostic_plots(data_trimmed, col, title)
```



Задание группы ИУ5-24М

Для произвольной колонки данных построить график "Скрипичная диаграмма (violin plot)".

[135]

0 сек.

```
sns.violinplot(x='Year', data=data)
```

```
<Axes: xlabel='Year'>
```