

Рубежный контроль №2

Назаров М.М. ИУ5-63Б

Вариант №15

Задание: Для заданного набора данных (по Вашему варианту) постройте модели классификации или регрессии (в зависимости от конкретной задачи, рассматриваемой в наборе данных). Для построения моделей используйте методы 1 и 2 (по варианту для Вашей группы). Оцените качество моделей на основе подходящих метрик качества (не менее двух метрик). Какие метрики качества Вы использовали и почему? Какие выводы Вы можете сделать о качестве построенных моделей? Для построения моделей необходимо выполнить требуемую предобработку данных: заполнение пропусков, кодирование категориальных признаков, и т.д.

Методы для ИУ5-63Б. Метод №1: "Дерево решений". Метод №2: "Случайный лес".

Набор данных: [U.S. Education Datasets: Unification Project](#)

Импорт библиотек

In [107...]

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
```

Загрузка и первичная обработка набора данных

In [108...]

```
# загрузка набора данных
data=pd.read_csv('states_all_extended.csv', sep=",")
# размер набора данных
data.shape
```

Out[108...](1715, 266)

In [109...]

```
# первые 5 строк набора данных
data.head()
```

Out[109...]

	PRIMARY_KEY	STATE	YEAR	ENROLL	TOTAL_REVENUE	FEDERAL_REVENUE	STATE_REVENUE
0	1992_ALABAMA	ALABAMA	1992	NaN	2678885.0	304177.0	1659028.0
1	1992_ALASKA	ALASKA	1992	NaN	1049591.0	106780.0	720711.0
2	1992_ARIZONA	ARIZONA	1992	NaN	3258079.0	297888.0	1369815.0
3	1992_ARKANSAS	ARKANSAS	1992	NaN	1711959.0	178571.0	958785.0
4	1992_CALIFORNIA	CALIFORNIA	1992	NaN	26260025.0	2072470.0	16546514.0

5 rows × 266 columns

```
In [110... parts = np.split(data, [13], axis=1)
data = parts[0]
```

```
In [111... # список колонок с типами данных
data.dtypes
```

```
Out[111... PRIMARY_KEY          object
STATE              object
YEAR              object
ENROLL             object
TOTAL_REVENUE      object
FEDERAL_REVENUE    object
STATE_REVENUE      object
LOCAL_REVENUE      object
TOTAL_EXPENDITURE   object
INSTRUCTION_EXPENDITURE object
SUPPORT_SERVICES_EXPENDITURE object
OTHER_EXPENDITURE   object
CAPITAL_OUTLAY_EXPENDITURE object
dtype: object
```

```
In [112... data.drop(['ENROLL','PRIMARY_KEY'], axis = 1, inplace = True)
```

Обработка пропусков данных

```
In [113... le = LabelEncoder()
le.fit(data.STATE.drop_duplicates())
data.STATE = le.transform(data.STATE)
```

```
In [114... # Удаление строк, содержащих пустые значения
data_new = data.dropna(axis=0, how='any')
(data.shape, data_new.shape)
```

```
Out[114... ((1715, 11), (1224, 11))
```

```
In [115... data=data_new
```

```
In [116... data.isnull().sum()
```

```
Out[116... STATE          0
YEAR           0
TOTAL_REVENUE 0
FEDERAL_REVENUE 0
STATE_REVENUE 0
LOCAL_REVENUE 0
TOTAL_EXPENDITURE 0
INSTRUCTION_EXPENDITURE 0
SUPPORT_SERVICES_EXPENDITURE 0
OTHER_EXPENDITURE 0
CAPITAL_OUTLAY_EXPENDITURE 0
dtype: int64
```

```
In [117... data.head()
```

Out[117...]

	STATE	YEAR	TOTAL_REVENUE	FEDERAL_REVENUE	STATE_REVENUE	LOCAL_REVENUE	TOTAL_EXPENSES
51	0	1993	2.82739e+06	331409	1.7293e+06	766687	2.83e+06
52	1	1993	1.1914e+06	176150	775829	239419	1.19e+06
53	2	1993	3.42798e+06	318465	1.41541e+06	1.6941e+06	3.62e+06
54	3	1993	1.34691e+06	128196	771079	447634	1.37e+06
55	4	1993	2.80433e+07	2.15116e+06	1.70641e+07	8.82804e+06	2.83e+07

Масштабирование данных:

MinMax масштабирование

In [118...]

```
from sklearn.preprocessing import MinMaxScaler, StandardScaler, Normalizer
```

In [119...]

```
# Числовые колонки для масштабирования
scale_cols = num_cols
```

In [120...]

```
sc1 = MinMaxScaler()
sc1_data = sc1.fit_transform(data[scale_cols])
```

In [121...]

```
# Добавим масштабированные данные в набор данных
for i in range(len(scale_cols)):
    col = scale_cols[i]
    new_col_name = col + '_scaled'
    data[new_col_name] = sc1_data[:,i]
```

<ipython-input-121-29357899d09f>:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
data[new_col_name] = sc1_data[:,i]

In [122...]

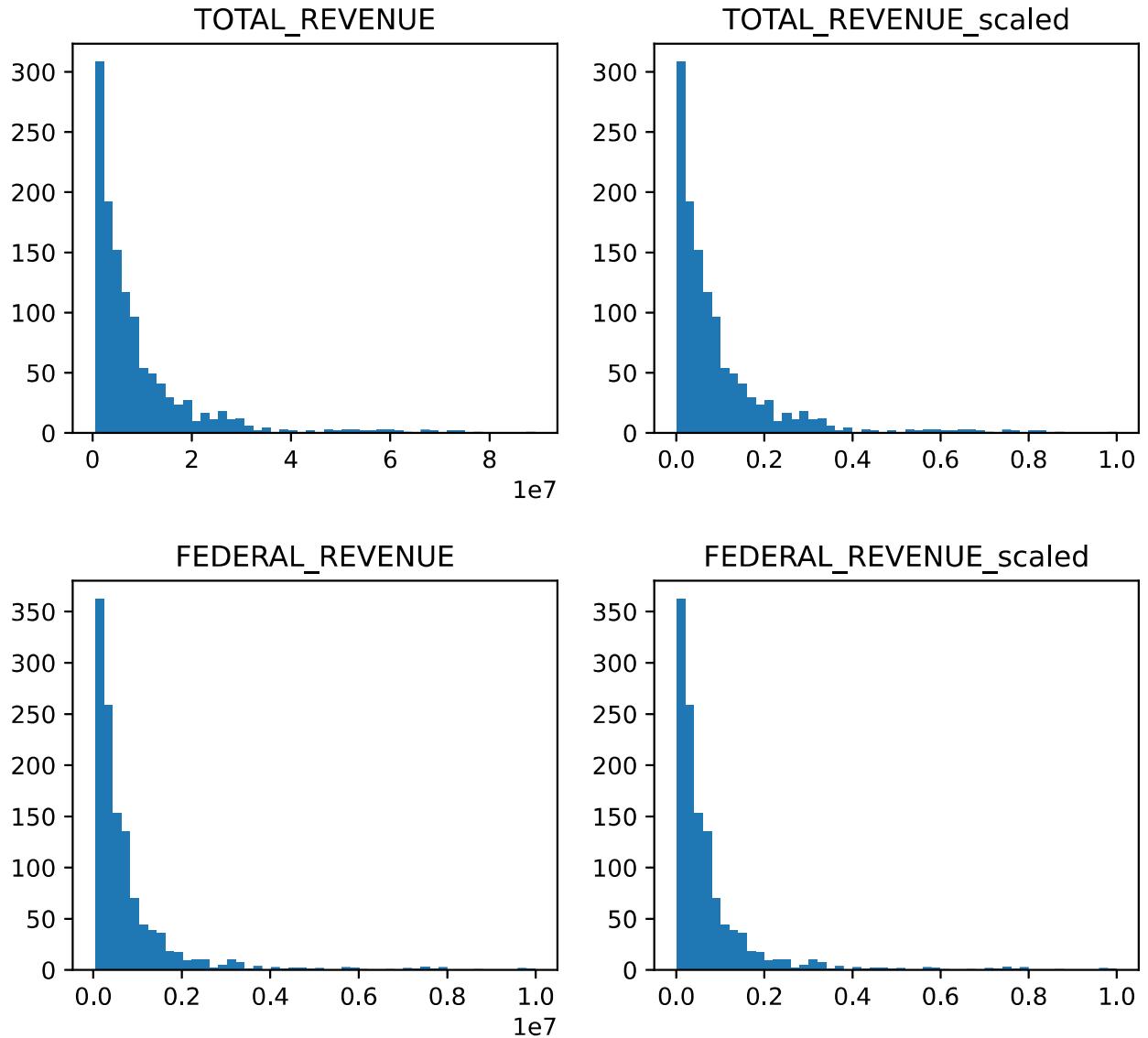
```
data.head()
```

Out[122...]

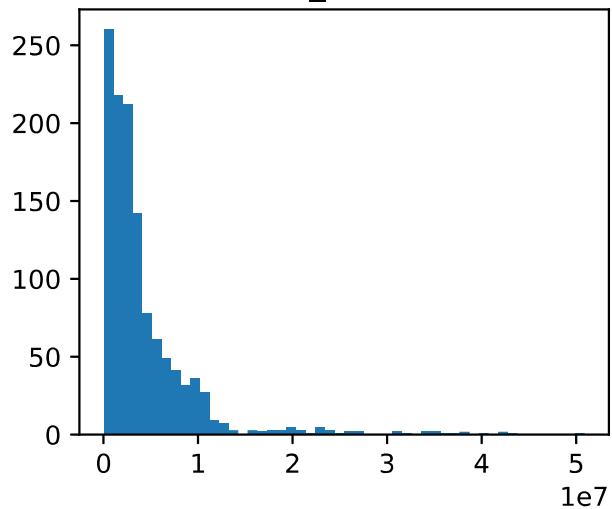
	STATE	YEAR	TOTAL_REVENUE	FEDERAL_REVENUE	STATE_REVENUE	LOCAL_REVENUE	TOTAL_EXPENSES
51	0	1993	2.82739e+06	331409	1.7293e+06	766687	2.83e+06
52	1	1993	1.1914e+06	176150	775829	239419	1.19e+06
53	2	1993	3.42798e+06	318465	1.41541e+06	1.6941e+06	3.62e+06
54	3	1993	1.34691e+06	128196	771079	447634	1.37e+06
55	4	1993	2.80433e+07	2.15116e+06	1.70641e+07	8.82804e+06	2.83e+07

```
for col in scale_cols:
    col_scaled = col + '_scaled'
```

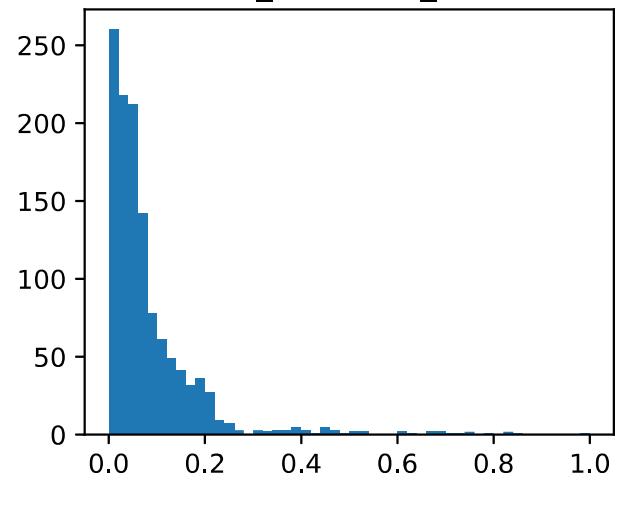
```
fig, ax = plt.subplots(1, 2, figsize=(8,3))
ax[0].hist(data[col], 50)
ax[1].hist(data[col_scaled], 50)
ax[0].title.set_text(col)
ax[1].title.set_text(col_scaled)
plt.show()
```



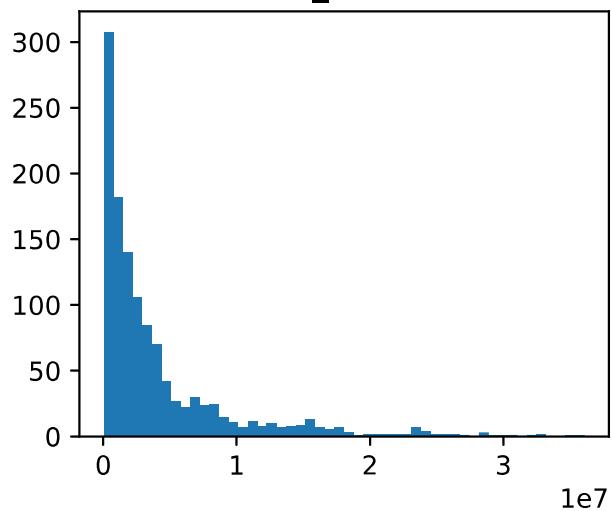
STATE_REVENUE



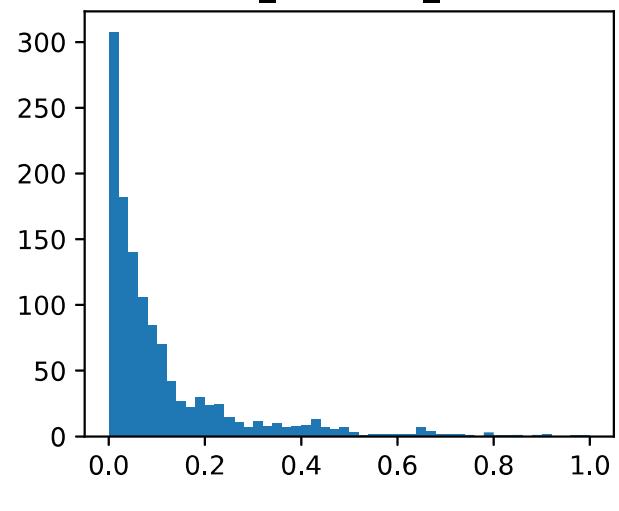
STATE_REVENUE_scaled



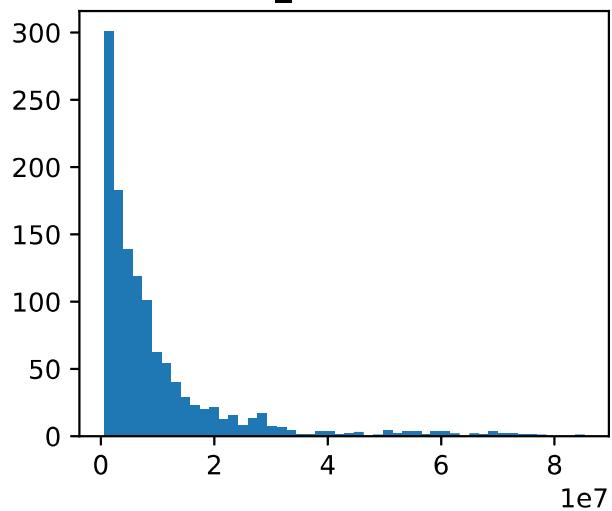
LOCAL_REVENUE



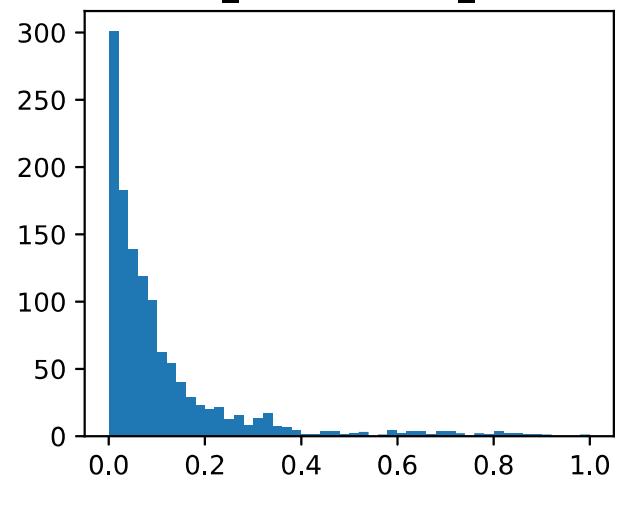
LOCAL_REVENUE_scaled

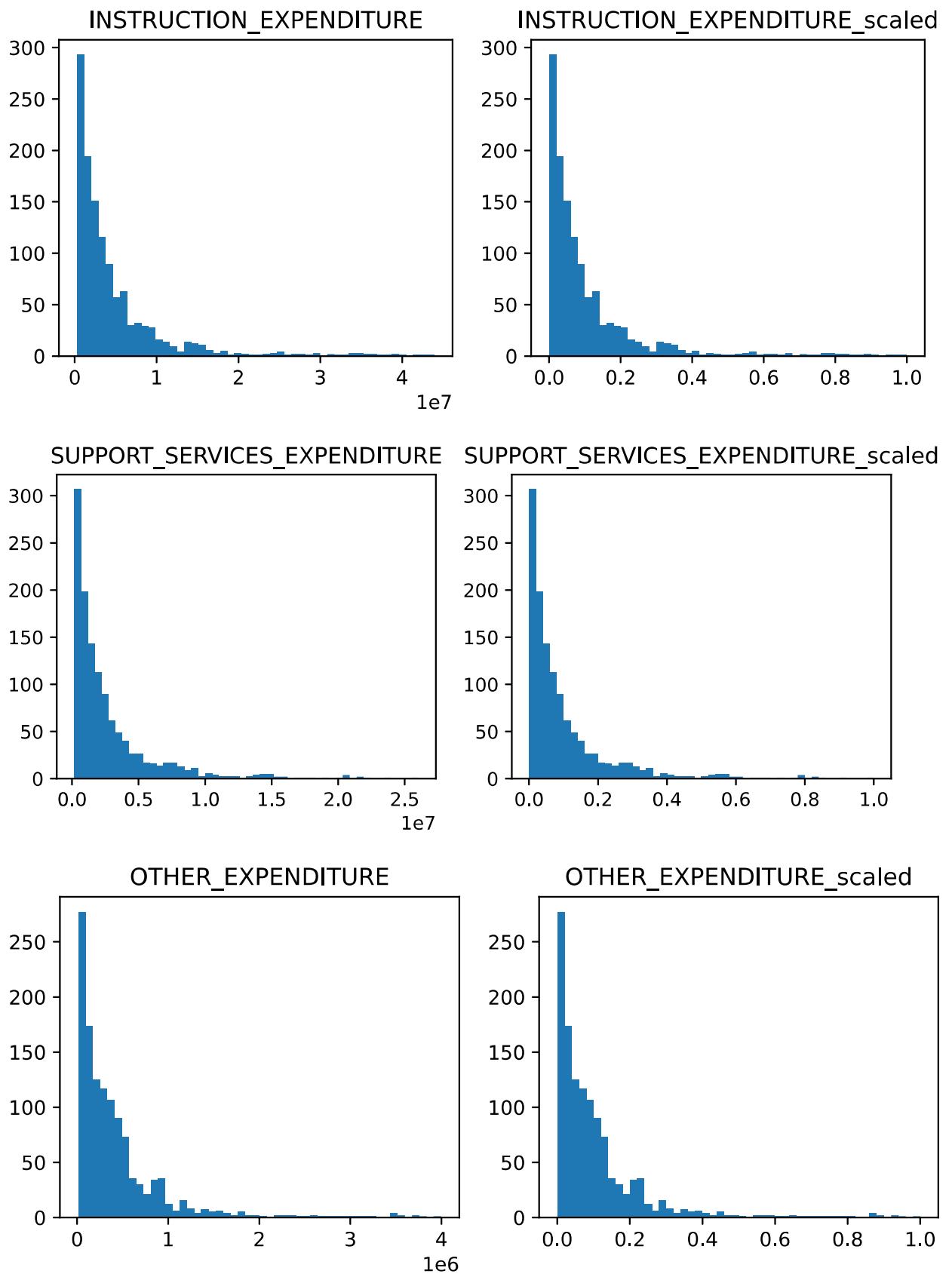


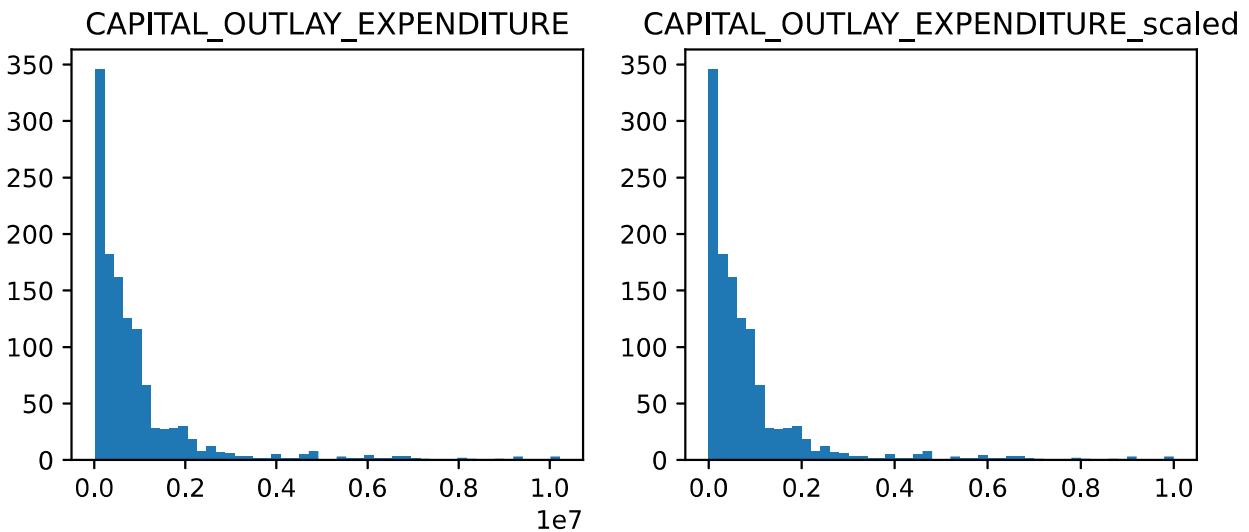
TOTAL_EXPENDITURE



TOTAL_EXPENDITURE_scaled







In [124...]: `data.drop(['TOTAL_REVENUE', 'FEDERAL_REVENUE', 'STATE_REVENUE', 'LOCAL_REVENUE', 'TOTAL_EXP`

D:\Anaconda3\lib\site-packages\pandas\core\frame.py:4163: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: <https://pandas.pydata.org/pandas-docs/stable/user-guide/indexing.html#returning-a-view-versus-a-copy>
return super().drop()

In [125...]: `data.head()`

Out[125...]:

	STATE	YEAR	TOTAL_REVENUE_scaled	FEDERAL_REVENUE_scaled	STATE_REVENUE_scaled	LOCAL_REV
51	0	1993	0.026611	0.029904	0.033971	
52	1	1993	0.008177	0.014310	0.015241	
53	2	1993	0.033378	0.028604	0.027805	
54	3	1993	0.009929	0.009494	0.015148	
55	4	1993	0.310729	0.212673	0.335218	

Построение моделей:

In [126...]:

```
X = data.drop(['TOTAL_EXPENDITURE_scaled'], axis = 1)
Y = data.TOTAL_EXPENDITURE_scaled
print('Входные данные:\n\n', X.head(), '\n\nВыходные данные:\n\n', Y.head())
```

Входные данные:

51	0	1993	0.026611	0.029904
52	1	1993	0.008177	0.014310
53	2	1993	0.033378	0.028604
54	3	1993	0.009929	0.009494
55	4	1993	0.310729	0.212673

51	0.033971	0.020586
52	0.015241	0.005973

```

53          0.027805      0.046289
54          0.015148      0.011743
55          0.335218      0.244007

    INSTRUCTION_EXPENDITURE_scaled SUPPORT_SERVICES_EXPENDITURE_scaled \
51                  0.029726      0.025240
52                  0.005249      0.011337
53                  0.030054      0.033218
54                  0.011836      0.009513
55                  0.343614      0.338551

    OTHER_EXPENDITURE_scaled CAPITAL_OUTLAY_EXPENDITURE_scaled
51          0.056641      0.018754
52          0.006212      0.012054
53          0.038285      0.065364
54          0.014325      0.008336
55          0.400805      0.189214

```

Выходные данные:

```

51  0.027721
52  0.007600
53  0.037038
54  0.010542
55  0.325670
Name: TOTAL_EXPENDITURE_scaled, dtype: float64

```

```
In [159...]: X_train, X_test, Y_train, Y_test = train_test_split(X, Y, random_state = 0, test_size = 0.2)
print('Входные параметры обучающей выборки:\n\n', X_train.head(), '\n\n')
print('Входные параметры тестовой выборки:\n\n', X_test.head(), '\n\n')
print('Выходные параметры обучающей выборки:\n\n', Y_train.head(), '\n\n')
print('Выходные параметры тестовой выборки:\n\n', Y_test.head())

```

Входные параметры обучающей выборки:

```

STATE  YEAR  TOTAL_REVENUE_scaled  FEDERAL_REVENUE_scaled \
1147   26   2014          0.110319      0.086583
632    21   2004          0.097551      0.055026
446    40   2000          0.187321      0.099322
239    37   1996          0.131909      0.067389
546    38   2002          0.043997      0.046619

STATE_REVENUE_scaled  LOCAL_REVENUE_scaled \
1147          0.083825      0.140513
632          0.067513      0.140830
446          0.121088      0.273833
239          0.094911      0.183276
546          0.046084      0.041650

INSTRUCTION_EXPENDITURE_scaled SUPPORT_SERVICES_EXPENDITURE_scaled \
1147          0.113907      0.117444
632          0.107498      0.097751
446          0.195715      0.183160
239          0.137901      0.150714
546          0.044124      0.050143

OTHER_EXPENDITURE_scaled CAPITAL_OUTLAY_EXPENDITURE_scaled
1147          0.142063      0.088804
632          0.095904      0.055697
446          0.168206      0.180041
239          0.130640      0.081339
546          0.108980      0.032226

```

Входные параметры тестовой выборки:

```

      STATE  YEAR  TOTAL_REVENUE_scaled  FEDERAL_REVENUE_scaled  \
904      39    2009            0.064147            0.062534
56       5    1993            0.029213            0.011462
963     47    2010            0.014470            0.014047
175     23    1995            0.135598            0.066180
1049    31    2012            0.028141            0.015424

      STATE_REVENUE_scaled  LOCAL_REVENUE_scaled  \
904            0.061238            0.065442
56            0.024405            0.045571
963            0.026059            0.006262
175            0.158210            0.103379
1049           0.020221            0.047746

      INSTRUCTION_EXPENDITURE_scaled  SUPPORT_SERVICES_EXPENDITURE_scaled  \
904                  0.069241            0.076449
56                  0.029112            0.031409
963                  0.014304            0.014264
175                  0.134942            0.145689
1049                 0.031788            0.028110

      OTHER_EXPENDITURE_scaled  CAPITAL_OUTLAY_EXPENDITURE_scaled
904            0.050988            0.077406
56            0.020234            0.029992
963            0.009045            0.005195
175            0.234797            0.072560
1049           0.016727            0.014430

```

Выходные параметры обучающей выборки:

```

1147    0.116016
632     0.099486
446     0.204862
239     0.135009
546     0.046987
Name: TOTAL_EXPENDITURE_scaled, dtype: float64

```

Выходные параметры тестовой выборки:

```

904    0.073473
56     0.030018
963    0.014595
175    0.141400
1049   0.029232
Name: TOTAL_EXPENDITURE_scaled, dtype: float64

```

Модель "Дерево решений"

```
In [160...]: from sklearn.tree import DecisionTreeRegressor
```

```
In [162...]: dtc = DecisionTreeRegressor(random_state=1).fit(X_train, Y_train)
data_test_predicted_dtc = dtc.predict(X_test)
```

Модель "Случайный лес"

```
In [163...]: from sklearn.ensemble import RandomForestRegressor
```

```
In [164...]: RF = RandomForestRegressor(random_state=1).fit(X_train, Y_train)
data_test_predicted_rf = RF.predict(X_test)
```

```
In [165...]: from sklearn.metrics import mean_squared_error, r2_score
```

Оценка качества моделей:

В качестве метрик для оценки качества моделей я использую Mean squared error (средняя квадратичная ошибка), как наиболее часто используемую метрику для оценки качества регрессии, и метрику R^2 (коэффициент детерминации), потому что эта метрика является нормированной.

In [166...]

```
# Mean squared error - средняя квадратичная ошибка
print('Метрика MSE:\nДерево решений: {} \nСлучайный лес: {}'.format(mean_squared_error(Y
```

Метрика MSE:

Дерево решений: 0.0001797353469085379

Случайный лес: 0.0001170196395991741

In [167...]

```
# 4) Метрика R2 или коэффициент детерминации
print('Метрика R2:\nДерево решений: {} \nСлучайный лес: {}'.format(r2_score(Y_test,
```

Метрика R^2 :

Дерево решений: 0.9933087725656454

Случайный лес: 0.9956435668536433

Вывод:

Исходя из оценки качества построенных моделей можно увидеть, что модели "Случайный лес" и "Дерево решений" одинаково хорошо справились с поставленной задачей