Содержание

- 1 Подготовка данных
  - 1.1 Изучение данных
  - 1.2 Рассчет эффективности rougher.output.recovery (MAE)
  - 1.3 Анализ признаков, недоступных в тестовой выборке
  - 1.4 Предобработка данных
- 2 Анализ данных
  - 2.1 Концентрация металлов (Au, Ag, Pb) на различных этапах очистки
    - 2.1.1 Золото
    - 2.1.2 Серебро
    - 2.1.3 Свинец
  - 2.2 Сравнение распределения размеров гранул сырья на обучающей и тестовой выборках
    - 2.2.1 Этап флотации
    - 2.2.2 Этап очистки
  - 2.3 Исследование суммарной концентрации всех веществ на разных стадиях
- 3 Модель
  - 3.1 Функция для вычисления итоговой sMAPE
  - 3.2 Подготовка данных для обучения
  - 3.3 Поиск лучшей модели
    - 3.3.1 Константная модель
    - 3.3.2 Decision Tree (Дерево решений)
    - 3.3.3 Random Forest (Случайный лес)
    - 3.3.4 Linear Regression (Линейная регрессия)
    - 3.3.5 Определим лучшую модель
  - 3.4 Тестирование модели
- 4 Общий вывод
- 5 Чек-лист готовности проекта

Восстановление золота из руды

Необходимо подготовить прототип модели машинного обучения. Компания разрабатывает решения для эффективной работы промышленных предприятий.

Модель должна предсказать коэффициент восстановления золота из золотосодержащей руды. Используем предоставленные данные с параметрами добычи и очистки.

Модель поможет оптимизировать производство, чтобы не запускать предприятие с убыточными характеристиками.

Нам нужно:

1. Подготовить данные;
2. Провести исследовательский анализ данных;
3. Построить и обучить модель.

Чтобы выполнить проект, используем библиотеки *pandas*, *matplotlib* и *sklearn.*

Цель исследования — подготовить прототип модели машинного обучения. Модель должна предсказать коэффициент восстановления золота из золотосодержащей руды.

Модель поможет оптимизировать производство, чтобы не запускать предприятие с убыточными характеристиками.

Данные с параметрами добычи и очистки получим из трех файлов:

- gold_recovery_train_new.csv — обучающая выборка
- gold_recovery_test_new.csv — тестовая выборка
- gold_recovery_full_new.csv — исходные данные

Исследование пройдёт в три этапа:

1. Подготовка данных
2. Анализ данных
3. Построение модели

Подготовка данных

In [4]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

from numpy.random import RandomState

from tqdm import tqdm

from sklearn.metrics import make_scorer
from sklearn.metrics import mean_absolute_error as mae
```

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import cross_val_score
from sklearn.preprocessing import StandardScaler
```

In [5]:
```
r_state = 12345
state = RandomState(r_state)
```

In [6]:
```
data_train = pd.read_csv('datasets/gold_industry_train.csv')
data_test = pd.read_csv('datasets/gold_industry_test.csv')
data_full = pd.read_csv('datasets/gold_industry_full.csv')

datas = {'Обучающая выборка' : data_train, 'Тестовая выборка' : data_test, 'Исходные данные' : data_full}
```

Изучение данных

In [7]:
```
for i in datas:
    print('\033[1mОбщие сведения "{}":'.format(i))
    print('\033[0m')
    datas[i].info()
    display(datas[i].sample(5, random_state=r_state))
    display(datas[i].head())

    print('Количество пропусков по столбцам:')
    print()

    for col in datas[i].columns:
        nmv = datas[i][col].isna().sum()
        pmv = nmv/len(datas[i])

        if pmv == 0:
            print('\033[0m{} - {} шт. - {:.2%}'.format(col, nmv, pmv))
        elif pmv <= 0.1:
            print('\033[0m{} - \033[43m{} шт.\033[0m - \033[43m{:.2%}'.format(col, nmv, pmv))
        else:
            print('\033[0m{} - \033[41m{} шт.\033[0m - \033[41m{:.2%}'.format(col, nmv, pmv))
        print('\033[0m')

    print('Количество явных дубликатов:', datas[i].duplicated().sum())
    print()
```

Общие сведения "Обучающая выборка":

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14579 entries, 0 to 14578
Data columns (total 87 columns):
 #   Column                                      Non-Null Count  Dtype
---  ------                                      --------------  -----
 0   date                                        14579 non-null  object
 1   rougher.input.feed_au                       14579 non-null  float64
 2   rougher.input.feed_ag                       14579 non-null  float64
 3   rougher.input.feed_pb                       14507 non-null  float64
 4   rougher.input.feed_sol                      14502 non-null  float64
 5   rougher.input.feed_rate                     14572 non-null  float64
 6   rougher.input.feed_size                     14478 non-null  float64
 7   rougher.input.floatbank10_sulfate           14548 non-null  float64
 8   rougher.input.floatbank10_xanthate          14572 non-null  float64
 9   rougher.state.floatbank10_a_air             14579 non-null  float64
 10  rougher.state.floatbank10_a_level           14579 non-null  float64
 11  rougher.state.floatbank10_b_air             14579 non-null  float64
 12  rougher.state.floatbank10_b_level           14579 non-null  float64
 13  rougher.state.floatbank10_c_air             14579 non-null  float64
 14  rougher.state.floatbank10_c_level           14579 non-null  float64
 15  rougher.state.floatbank10_d_air             14579 non-null  float64
 16  rougher.state.floatbank10_d_level           14579 non-null  float64
 17  rougher.state.floatbank10_e_air             14150 non-null  float64
 18  rougher.state.floatbank10_e_level           14579 non-null  float64
 19  rougher.state.floatbank10_f_air             14579 non-null  float64
 20  rougher.state.floatbank10_f_level           14579 non-null  float64
 21  rougher.input.floatbank11_sulfate           14543 non-null  float64
 22  rougher.input.floatbank11_xanthate          14172 non-null  float64
 23  rougher.calculation.sulfate_to_au_concentrate  14578 non-null  float64
 24  rougher.calculation.floatbank10_sulfate_to_au_feed  14578 non-null  float64
 25  rougher.calculation.floatbank11_sulfate_to_au_feed  14578 non-null  float64
 26  rougher.calculation.au_pb_ratio             14579 non-null  float64
 27  rougher.output.concentrate_au               14579 non-null  float64
 28  rougher.output.concentrate_ag               14579 non-null  float64
 29  rougher.output.concentrate_pb               14579 non-null  float64
 30  rougher.output.concentrate_sol              14561 non-null  float64
 31  rougher.output.recovery                     14579 non-null  float64
 32  rougher.output.tail_au                      14579 non-null  float64
 33  rougher.output.tail_ag                      14578 non-null  float64
 34  rougher.output.tail_pb                      14579 non-null  float64
```

```
 35  rougher.output.tail_sol                     14579 non-null  float64
 36  primary_cleaner.input.sulfate               14556 non-null  float64
 37  primary_cleaner.input.depressant            14551 non-null  float64
 38  primary_cleaner.input.feed_size             14579 non-null  float64
 39  primary_cleaner.input.xanthate              14518 non-null  float64
 40  primary_cleaner.state.floatbank8_a_air      14576 non-null  float64
 41  primary_cleaner.state.floatbank8_a_level    14579 non-null  float64
 42  primary_cleaner.state.floatbank8_b_air      14576 non-null  float64
 43  primary_cleaner.state.floatbank8_b_level    14579 non-null  float64
 44  primary_cleaner.state.floatbank8_c_air      14579 non-null  float64
 45  primary_cleaner.state.floatbank8_c_level    14579 non-null  float64
 46  primary_cleaner.state.floatbank8_d_air      14578 non-null  float64
 47  primary_cleaner.state.floatbank8_d_level    14579 non-null  float64
 48  primary_cleaner.output.concentrate_au       14579 non-null  float64
 49  primary_cleaner.output.concentrate_ag       14579 non-null  float64
 50  primary_cleaner.output.concentrate_pb       14491 non-null  float64
 51  primary_cleaner.output.concentrate_sol      14314 non-null  float64
 52  primary_cleaner.output.tail_au              14579 non-null  float64
 53  primary_cleaner.output.tail_ag              14575 non-null  float64
 54  primary_cleaner.output.tail_pb              14573 non-null  float64
 55  primary_cleaner.output.tail_sol             14534 non-null  float64
 56  secondary_cleaner.state.floatbank2_a_air    14485 non-null  float64
 57  secondary_cleaner.state.floatbank2_a_level  14579 non-null  float64
 58  secondary_cleaner.state.floatbank2_b_air    14557 non-null  float64
 59  secondary_cleaner.state.floatbank2_b_level  14579 non-null  float64
 60  secondary_cleaner.state.floatbank3_a_air    14567 non-null  float64
 61  secondary_cleaner.state.floatbank3_a_level  14579 non-null  float64
 62  secondary_cleaner.state.floatbank3_b_air    14579 non-null  float64
 63  secondary_cleaner.state.floatbank3_b_level  14579 non-null  float64
 64  secondary_cleaner.state.floatbank4_a_air    14574 non-null  float64
 65  secondary_cleaner.state.floatbank4_a_level  14579 non-null  float64
 66  secondary_cleaner.state.floatbank4_b_air    14579 non-null  float64
 67  secondary_cleaner.state.floatbank4_b_level  14579 non-null  float64
 68  secondary_cleaner.state.floatbank5_a_air    14579 non-null  float64
 69  secondary_cleaner.state.floatbank5_a_level  14579 non-null  float64
 70  secondary_cleaner.state.floatbank5_b_air    14579 non-null  float64
 71  secondary_cleaner.state.floatbank5_b_level  14579 non-null  float64
 72  secondary_cleaner.state.floatbank6_a_air    14578 non-null  float64
 73  secondary_cleaner.state.floatbank6_a_level  14579 non-null  float64
 74  secondary_cleaner.output.tail_au            14579 non-null  float64
 75  secondary_cleaner.output.tail_ag            14578 non-null  float64
 76  secondary_cleaner.output.tail_pb            14575 non-null  float64
 77  secondary_cleaner.output.tail_sol           13659 non-null  float64
 78  final.output.concentrate_au                 14579 non-null  float64
 79  final.output.concentrate_ag                 14578 non-null  float64
 80  final.output.concentrate_pb                 14578 non-null  float64
 81  final.output.concentrate_sol                14387 non-null  float64
 82  final.output.recovery                       14579 non-null  float64
 83  final.output.tail_au                        14579 non-null  float64
 84  final.output.tail_ag                        14578 non-null  float64
 85  final.output.tail_pb                        14504 non-null  float64
 86  final.output.tail_sol                       14574 non-null  float64
dtypes: float64(86), object(1)
memory usage: 9.7+ MB
```

	date	rougher.input.feed_au	rougher.input.feed_ag	rougher.input.feed_pb	rougher.input.feed_sol	rougher.input.feed_rate	rougher.input.feed_size	ro
11066	2017-07-04 14:59:59	11.311617	11.508350	5.760956	38.012769	457.315357	54.984357	
7524	2017-01-10 09:59:59	7.195038	7.840737	3.649247	39.212813	504.674931	53.906820	
10618	2017-06-14 19:59:59	7.411768	7.712749	3.392968	31.869068	439.050267	46.446289	
6773	2016-12-08 22:59:59	9.658762	8.672438	3.715044	29.747256	173.531215	93.257746	
853	2016-02-22 23:00:00	4.639804	7.166969	1.823448	24.950583	543.416648	55.266248	

5 rows × 87 columns

	date	rougher.input.feed_au	rougher.input.feed_ag	rougher.input.feed_pb	rougher.input.feed_sol	rougher.input.feed_rate	rougher.input.feed_size	rougher
0	2016-01-15 00:00:00	6.486150	6.100378	2.284912	36.808594	523.546326	55.486599	
1	2016-01-15 01:00:00	6.478583	6.161113	2.266033	35.753385	525.290581	57.278666	

	date	rougher.input.feed_au	rougher.input.feed_ag	rougher.input.feed_pb	rougher.input.feed_sol	rougher.input.feed_rate	rougher.input.feed_size	rougher
2	2016-01-15 02:00:00	6.362222	6.116455	2.159622	35.971630	530.026610	57.510649	
3	2016-01-15 03:00:00	6.118189	6.043309	2.037807	36.862241	542.590390	57.792734	
4	2016-01-15 04:00:00	5.663707	6.060915	1.786875	34.347666	540.531893	56.047189	

5 rows × 87 columns

Количество пропусков по столбцам:

date - 0 шт. - 0.00%

rougher.input.feed_au - 0 шт. - 0.00%

rougher.input.feed_ag - 0 шт. - 0.00%

rougher.input.feed_pb - 72 шт. - 0.49%

rougher.input.feed_sol - 77 шт. - 0.53%

rougher.input.feed_rate - 7 шт. - 0.05%

rougher.input.feed_size - 101 шт. - 0.69%

rougher.input.floatbank10_sulfate - 31 шт. - 0.21%

rougher.input.floatbank10_xanthate - 7 шт. - 0.05%

rougher.state.floatbank10_a_air - 0 шт. - 0.00%

rougher.state.floatbank10_a_level - 0 шт. - 0.00%

rougher.state.floatbank10_b_air - 0 шт. - 0.00%

rougher.state.floatbank10_b_level - 0 шт. - 0.00%

rougher.state.floatbank10_c_air - 0 шт. - 0.00%

rougher.state.floatbank10_c_level - 0 шт. - 0.00%

rougher.state.floatbank10_d_air - 0 шт. - 0.00%

rougher.state.floatbank10_d_level - 0 шт. - 0.00%

rougher.state.floatbank10_e_air - 429 шт. - 2.94%

rougher.state.floatbank10_e_level - 0 шт. - 0.00%

rougher.state.floatbank10_f_air - 0 шт. - 0.00%

rougher.state.floatbank10_f_level - 0 шт. - 0.00%

rougher.input.floatbank11_sulfate - 36 шт. - 0.25%

rougher.input.floatbank11_xanthate - 407 шт. - 2.79%

rougher.calculation.sulfate_to_au_concentrate - 1 шт. - 0.01%

rougher.calculation.floatbank10_sulfate_to_au_feed - 1 шт. - 0.01%

rougher.calculation.floatbank11_sulfate_to_au_feed - 1 шт. - 0.01%

rougher.calculation.au_pb_ratio - 0 шт. - 0.00%

rougher.output.concentrate_au - 0 шт. - 0.00%

rougher.output.concentrate_ag - 0 шт. - 0.00%

rougher.output.concentrate_pb - 0 шт. - 0.00%

rougher.output.concentrate_sol - 18 шт. - 0.12%

rougher.output.recovery - 0 шт. - 0.00%

rougher.output.tail_au - 0 шт. - 0.00%

rougher.output.tail_ag - 1 шт. - 0.01%

rougher.output.tail_pb - 0 шт. - 0.00%

rougher.output.tail_sol - 0 шт. - 0.00%

primary_cleaner.input.sulfate - 23 шт. - 0.16%

primary_cleaner.input.depressant - 28 шт. - 0.19%

primary_cleaner.input.feed_size - 0 шт. - 0.00%

primary_cleaner.input.xanthate - 61 шт. - 0.42%

primary_cleaner.state.floatbank8_a_air - 3 шт. - 0.02%

primary_cleaner.state.floatbank8_a_level - 0 шт. - 0.00%

primary_cleaner.state.floatbank8_b_air - 3 шт. - 0.02%

primary_cleaner.state.floatbank8_b_level - 0 шт. - 0.00%

primary_cleaner.state.floatbank8_c_air - 0 шт. - 0.00%

primary_cleaner.state.floatbank8_c_level - 0 шт. - 0.00%

primary_cleaner.state.floatbank8_d_air - 1 шт. - 0.01%

primary_cleaner.state.floatbank8_d_level - 0 шт. - 0.00%

primary_cleaner.output.concentrate_au - 0 шт. - 0.00%

primary_cleaner.output.concentrate_ag - 0 шт. - 0.00%

primary_cleaner.output.concentrate_pb - 88 шт. - 0.60%

primary_cleaner.output.concentrate_sol - 265 шт. - 1.82%

primary_cleaner.output.tail_au - 0 шт. - 0.00%

primary_cleaner.output.tail_ag - 4 шт. - 0.03%

primary_cleaner.output.tail_pb - 6 шт. - 0.04%

primary_cleaner.output.tail_sol - 45 шт. - 0.31%

secondary_cleaner.state.floatbank2_a_air - 94 шт. - 0.64%

secondary_cleaner.state.floatbank2_a_level - 0 шт. - 0.00%

secondary_cleaner.state.floatbank2_b_air - 22 шт. - 0.15%

secondary_cleaner.state.floatbank2_b_level - 0 шт. - 0.00%

secondary_cleaner.state.floatbank3_a_air - 12 шт. - 0.08%

secondary_cleaner.state.floatbank3_a_level - 0 шт. - 0.00%

secondary_cleaner.state.floatbank3_b_air - 0 шт. - 0.00%

secondary_cleaner.state.floatbank3_b_level - 0 шт. - 0.00%

secondary_cleaner.state.floatbank4_a_air - 5 шт. - 0.03%

secondary_cleaner.state.floatbank4_a_level - 0 шт. - 0.00%

secondary_cleaner.state.floatbank4_b_air - 0 шт. - 0.00%

secondary_cleaner.state.floatbank4_b_level - 0 шт. - 0.00%

secondary_cleaner.state.floatbank5_a_air - 0 шт. - 0.00%

secondary_cleaner.state.floatbank5_a_level - 0 шт. - 0.00%

secondary_cleaner.state.floatbank5_b_air - 0 шт. - 0.00%

secondary_cleaner.state.floatbank5_b_level - 0 шт. - 0.00%

secondary_cleaner.state.floatbank6_a_air - 1 шт. - 0.01%

secondary_cleaner.state.floatbank6_a_level - 0 шт. - 0.00%

secondary_cleaner.output.tail_au - 0 шт. - 0.00%

secondary_cleaner.output.tail_ag - 1 шт. - 0.01%

secondary_cleaner.output.tail_pb - 4 шт. - 0.03%

secondary_cleaner.output.tail_sol - 920 шт. - 6.31%

final.output.concentrate_au - 0 шт. - 0.00%

final.output.concentrate_ag - 1 шт. - 0.01%

final.output.concentrate_pb - 1 шт. - 0.01%

final.output.concentrate_sol - 192 шт. - 1.32%

final.output.recovery - 0 шт. - 0.00%

final.output.tail_au - 0 шт. - 0.00%

final.output.tail_ag - 1 шт. - 0.01%

final.output.tail_pb - 75 шт. - 0.51%

final.output.tail_sol - 5 шт. - 0.03%

Количество явных дубликатов: 0

Общие сведения "Тестовая выборка":

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4860 entries, 0 to 4859
Data columns (total 53 columns):

#	Column	Non-Null Count	Dtype
0	date	4860 non-null	object
1	rougher.input.feed_au	4860 non-null	float64
2	rougher.input.feed_ag	4860 non-null	float64
3	rougher.input.feed_pb	4832 non-null	float64
4	rougher.input.feed_sol	4838 non-null	float64
5	rougher.input.feed_rate	4856 non-null	float64
6	rougher.input.feed_size	4816 non-null	float64
7	rougher.input.floatbank10_sulfate	4857 non-null	float64
8	rougher.input.floatbank10_xanthate	4859 non-null	float64
9	rougher.state.floatbank10_a_air	4859 non-null	float64
10	rougher.state.floatbank10_a_level	4859 non-null	float64
11	rougher.state.floatbank10_b_air	4859 non-null	float64
12	rougher.state.floatbank10_b_level	4859 non-null	float64
13	rougher.state.floatbank10_c_air	4859 non-null	float64
14	rougher.state.floatbank10_c_level	4859 non-null	float64
15	rougher.state.floatbank10_d_air	4860 non-null	float64
16	rougher.state.floatbank10_d_level	4860 non-null	float64
17	rougher.state.floatbank10_e_air	4853 non-null	float64
18	rougher.state.floatbank10_e_level	4860 non-null	float64
19	rougher.state.floatbank10_f_air	4860 non-null	float64
20	rougher.state.floatbank10_f_level	4860 non-null	float64
21	rougher.input.floatbank11_sulfate	4852 non-null	float64
22	rougher.input.floatbank11_xanthate	4814 non-null	float64
23	primary_cleaner.input.sulfate	4859 non-null	float64
24	primary_cleaner.input.depressant	4851 non-null	float64
25	primary_cleaner.input.feed_size	4860 non-null	float64
26	primary_cleaner.input.xanthate	4817 non-null	float64
27	primary_cleaner.state.floatbank8_a_air	4859 non-null	float64
28	primary_cleaner.state.floatbank8_a_level	4859 non-null	float64
29	primary_cleaner.state.floatbank8_b_air	4859 non-null	float64
30	primary_cleaner.state.floatbank8_b_level	4859 non-null	float64
31	primary_cleaner.state.floatbank8_c_air	4858 non-null	float64
32	primary_cleaner.state.floatbank8_c_level	4859 non-null	float64
33	primary_cleaner.state.floatbank8_d_air	4858 non-null	float64
34	primary_cleaner.state.floatbank8_d_level	4859 non-null	float64
35	secondary_cleaner.state.floatbank2_a_air	4734 non-null	float64
36	secondary_cleaner.state.floatbank2_a_level	4859 non-null	float64
37	secondary_cleaner.state.floatbank2_b_air	4859 non-null	float64
38	secondary_cleaner.state.floatbank2_b_level	4859 non-null	float64
39	secondary_cleaner.state.floatbank3_a_air	4859 non-null	float64
40	secondary_cleaner.state.floatbank3_a_level	4859 non-null	float64
41	secondary_cleaner.state.floatbank3_b_air	4859 non-null	float64
42	secondary_cleaner.state.floatbank3_b_level	4859 non-null	float64
43	secondary_cleaner.state.floatbank4_a_air	4859 non-null	float64
44	secondary_cleaner.state.floatbank4_a_level	4859 non-null	float64
45	secondary_cleaner.state.floatbank4_b_air	4859 non-null	float64
46	secondary_cleaner.state.floatbank4_b_level	4859 non-null	float64
47	secondary_cleaner.state.floatbank5_a_air	4859 non-null	float64
48	secondary_cleaner.state.floatbank5_a_level	4859 non-null	float64
49	secondary_cleaner.state.floatbank5_b_air	4859 non-null	float64
50	secondary_cleaner.state.floatbank5_b_level	4859 non-null	float64

```
 51  secondary_cleaner.state.floatbank6_a_air    4859 non-null   float64
 52  secondary_cleaner.state.floatbank6_a_level  4859 non-null   float64
dtypes: float64(52), object(1)
memory usage: 2.0+ MB
```

	date	rougher.input.feed_au	rougher.input.feed_ag	rougher.input.feed_pb	rougher.input.feed_sol	rougher.input.feed_rate	rougher.input.feed_size	rou
3781	2018-06-21 06:59:59	10.962657	10.755106	3.924509	32.126409	401.822539	43.579452	
716	2018-01-24 04:59:59	3.668504	6.713862	0.490494	40.820509	488.690973	69.750987	
4597	2018-08-06 06:59:59	7.795887	7.602602	2.351350	37.849031	505.387278	47.812838	
4178	2018-07-11 00:59:59	5.621863	6.874989	2.868029	23.838692	392.516344	46.011460	
3547	2018-06-10 16:59:59	8.501001	7.810126	4.028249	27.271661	394.868960	39.985807	

5 rows × 53 columns

	date	rougher.input.feed_au	rougher.input.feed_ag	rougher.input.feed_pb	rougher.input.feed_sol	rougher.input.feed_rate	rougher.input.feed_size	roughe
0	2017-12-09 14:59:59	4.365491	6.158718	3.875727	39.135119	555.820208	94.544358	
1	2017-12-09 15:59:59	4.362781	6.048130	3.902537	39.713906	544.731687	123.742430	
2	2017-12-09 16:59:59	5.081681	6.082745	4.564078	37.208683	558.155110	82.610855	
3	2017-12-09 17:59:59	5.145949	6.084374	4.768124	36.808874	539.713765	77.984784	
4	2017-12-09 18:59:59	5.735249	6.165220	4.512346	37.810642	558.713584	86.434874	

5 rows × 53 columns

Количество пропусков по столбцам:

date - 0 шт. - 0.00%

rougher.input.feed_au - 0 шт. - 0.00%

rougher.input.feed_ag - 0 шт. - 0.00%

rougher.input.feed_pb - 28 шт. - 0.58%

rougher.input.feed_sol - 22 шт. - 0.45%

rougher.input.feed_rate - 4 шт. - 0.08%

rougher.input.feed_size - 44 шт. - 0.91%

rougher.input.floatbank10_sulfate - 3 шт. - 0.06%

rougher.input.floatbank10_xanthate - 1 шт. - 0.02%

rougher.state.floatbank10_a_air - 1 шт. - 0.02%

rougher.state.floatbank10_a_level - 1 шт. - 0.02%

rougher.state.floatbank10_b_air - 1 шт. - 0.02%

rougher.state.floatbank10_b_level - 1 шт. - 0.02%

rougher.state.floatbank10_c_air - 1 шт. - 0.02%

rougher.state.floatbank10_c_level - 1 шт. - 0.02%

rougher.state.floatbank10_d_air - 0 шт. - 0.00%

rougher.state.floatbank10_d_level - 0 шт. - 0.00%

rougher.state.floatbank10_e_air - 7 шт. - 0.14%

rougher.state.floatbank10_e_level - 0 шт. - 0.00%

rougher.state.floatbank10_f_air - 0 шт. - 0.00%

rougher.state.floatbank10_f_level - 0 шт. - 0.00%

rougher.input.floatbank11_sulfate - 8 шт. - 0.16%

rougher.input.floatbank11_xanthate - 46 шт. - 0.95%

primary_cleaner.input.sulfate - 1 шт. - 0.02%

primary_cleaner.input.depressant - 9 шт. - 0.19%

primary_cleaner.input.feed_size - 0 шт. - 0.00%

primary_cleaner.input.xanthate - 43 шт. - 0.88%

primary_cleaner.state.floatbank8_a_air - 1 шт. - 0.02%

primary_cleaner.state.floatbank8_a_level - 1 шт. - 0.02%

primary_cleaner.state.floatbank8_b_air - 1 шт. - 0.02%

primary_cleaner.state.floatbank8_b_level - 1 шт. - 0.02%

primary_cleaner.state.floatbank8_c_air - 2 шт. - 0.04%

primary_cleaner.state.floatbank8_c_level - 1 шт. - 0.02%

primary_cleaner.state.floatbank8_d_air - 2 шт. - 0.04%

primary_cleaner.state.floatbank8_d_level - 1 шт. - 0.02%

secondary_cleaner.state.floatbank2_a_air - 126 шт. - 2.59%

secondary_cleaner.state.floatbank2_a_level - 1 шт. - 0.02%

secondary_cleaner.state.floatbank2_b_air - 1 шт. - 0.02%

secondary_cleaner.state.floatbank2_b_level - 1 шт. - 0.02%

secondary_cleaner.state.floatbank3_a_air - 1 шт. - 0.02%

secondary_cleaner.state.floatbank3_a_level - 1 шт. - 0.02%

secondary_cleaner.state.floatbank3_b_air - 1 шт. - 0.02%

secondary_cleaner.state.floatbank3_b_level - 1 шт. - 0.02%

secondary_cleaner.state.floatbank4_a_air - 1 шт. - 0.02%

secondary_cleaner.state.floatbank4_a_level - 1 шт. - 0.02%

secondary_cleaner.state.floatbank4_b_air - 1 шт. - 0.02%

secondary_cleaner.state.floatbank4_b_level - 1 шт. - 0.02%

secondary_cleaner.state.floatbank5_a_air - 1 шт. - 0.02%

secondary_cleaner.state.floatbank5_a_level - 1 шт. - 0.02%

secondary_cleaner.state.floatbank5_b_air - 1 шт. - 0.02%

secondary_cleaner.state.floatbank5_b_level - 1 шт. - 0.02%

secondary_cleaner.state.floatbank6_a_air - 1 шт. - 0.02%

secondary_cleaner.state.floatbank6_a_level - 1 шт. - 0.02%

Количество явных дубликатов: 0

Общие сведения "Исходные данные":

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19439 entries, 0 to 19438
Data columns (total 87 columns):
 #   Column                                      Non-Null Count  Dtype  
---  ------                                      --------------  -----  
 0   date                                        19439 non-null  object 
 1   rougher.input.feed_au                       19439 non-null  float64
 2   rougher.input.feed_ag                       19439 non-null  float64
 3   rougher.input.feed_pb                       19339 non-null  float64
```

4 rougher.input.feed_sol 19340 non-null float64
5 rougher.input.feed_rate 19428 non-null float64
6 rougher.input.feed_size 19294 non-null float64
7 rougher.input.floatbank10_sulfate 19405 non-null float64
8 rougher.input.floatbank10_xanthate 19431 non-null float64
9 rougher.state.floatbank10_a_air 19438 non-null float64
10 rougher.state.floatbank10_a_level 19438 non-null float64
11 rougher.state.floatbank10_b_air 19438 non-null float64
12 rougher.state.floatbank10_b_level 19438 non-null float64
13 rougher.state.floatbank10_c_air 19438 non-null float64
14 rougher.state.floatbank10_c_level 19438 non-null float64
15 rougher.state.floatbank10_d_air 19439 non-null float64
16 rougher.state.floatbank10_d_level 19439 non-null float64
17 rougher.state.floatbank10_e_air 19003 non-null float64
18 rougher.state.floatbank10_e_level 19439 non-null float64
19 rougher.state.floatbank10_f_air 19439 non-null float64
20 rougher.state.floatbank10_f_level 19439 non-null float64
21 rougher.input.floatbank11_sulfate 19395 non-null float64
22 rougher.input.floatbank11_xanthate 18986 non-null float64
23 rougher.calculation.sulfate_to_au_concentrate 19437 non-null float64
24 rougher.calculation.floatbank10_sulfate_to_au_feed 19437 non-null float64
25 rougher.calculation.floatbank11_sulfate_to_au_feed 19437 non-null float64
26 rougher.calculation.au_pb_ratio 19439 non-null float64
27 rougher.output.concentrate_au 19439 non-null float64
28 rougher.output.concentrate_ag 19439 non-null float64
29 rougher.output.concentrate_pb 19439 non-null float64
30 rougher.output.concentrate_sol 19416 non-null float64
31 rougher.output.recovery 19439 non-null float64
32 rougher.output.tail_au 19439 non-null float64
33 rougher.output.tail_ag 19438 non-null float64
34 rougher.output.tail_pb 19439 non-null float64
35 rougher.output.tail_sol 19439 non-null float64
36 primary_cleaner.input.sulfate 19415 non-null float64
37 primary_cleaner.input.depressant 19402 non-null float64
38 primary_cleaner.input.feed_size 19439 non-null float64
39 primary_cleaner.input.xanthate 19335 non-null float64
40 primary_cleaner.state.floatbank8_a_air 19435 non-null float64
41 primary_cleaner.state.floatbank8_a_level 19438 non-null float64
42 primary_cleaner.state.floatbank8_b_air 19435 non-null float64
43 primary_cleaner.state.floatbank8_b_level 19438 non-null float64
44 primary_cleaner.state.floatbank8_c_air 19437 non-null float64
45 primary_cleaner.state.floatbank8_c_level 19438 non-null float64
46 primary_cleaner.state.floatbank8_d_air 19436 non-null float64
47 primary_cleaner.state.floatbank8_d_level 19438 non-null float64
48 primary_cleaner.output.concentrate_au 19439 non-null float64
49 primary_cleaner.output.concentrate_ag 19439 non-null float64
50 primary_cleaner.output.concentrate_pb 19323 non-null float64
51 primary_cleaner.output.concentrate_sol 19069 non-null float64
52 primary_cleaner.output.tail_au 19439 non-null float64
53 primary_cleaner.output.tail_ag 19435 non-null float64
54 primary_cleaner.output.tail_pb 19418 non-null float64
55 primary_cleaner.output.tail_sol 19377 non-null float64
56 secondary_cleaner.state.floatbank2_a_air 19219 non-null float64
57 secondary_cleaner.state.floatbank2_a_level 19438 non-null float64
58 secondary_cleaner.state.floatbank2_b_air 19416 non-null float64
59 secondary_cleaner.state.floatbank2_b_level 19438 non-null float64
60 secondary_cleaner.state.floatbank3_a_air 19426 non-null float64
61 secondary_cleaner.state.floatbank3_a_level 19438 non-null float64
62 secondary_cleaner.state.floatbank3_b_air 19438 non-null float64
63 secondary_cleaner.state.floatbank3_b_level 19438 non-null float64
64 secondary_cleaner.state.floatbank4_a_air 19433 non-null float64
65 secondary_cleaner.state.floatbank4_a_level 19438 non-null float64
66 secondary_cleaner.state.floatbank4_b_air 19438 non-null float64
67 secondary_cleaner.state.floatbank4_b_level 19438 non-null float64
68 secondary_cleaner.state.floatbank5_a_air 19438 non-null float64
69 secondary_cleaner.state.floatbank5_a_level 19438 non-null float64
70 secondary_cleaner.state.floatbank5_b_air 19438 non-null float64
71 secondary_cleaner.state.floatbank5_b_level 19438 non-null float64
72 secondary_cleaner.state.floatbank6_a_air 19437 non-null float64
73 secondary_cleaner.state.floatbank6_a_level 19438 non-null float64
74 secondary_cleaner.output.tail_au 19439 non-null float64
75 secondary_cleaner.output.tail_ag 19437 non-null float64
76 secondary_cleaner.output.tail_pb 19427 non-null float64
77 secondary_cleaner.output.tail_sol 17691 non-null float64
78 final.output.concentrate_au 19439 non-null float64
79 final.output.concentrate_ag 19438 non-null float64
80 final.output.concentrate_pb 19438 non-null float64
81 final.output.concentrate_sol 19228 non-null float64
82 final.output.recovery 19439 non-null float64
83 final.output.tail_au 19439 non-null float64
84 final.output.tail_ag 19438 non-null float64
85 final.output.tail_pb 19338 non-null float64
86 final.output.tail_sol 19433 non-null float64

86 final.output.tail_sol 19433 non-null float64
dtypes: float64(86), object(1)
memory usage: 12.9+ MB

	date	rougher.input.feed_au	rougher.input.feed_ag	rougher.input.feed_pb	rougher.input.feed_sol	rougher.input.feed_rate	rougher.input.feed_size	ro
5264	2016-10-04 14:59:59	8.739823	9.714287	3.521696	35.384010	423.588713	99.643338	
12699	2017-09-17 21:59:59	9.380200	9.108785	4.671233	39.726142	559.553945	73.045854	
4329	2016-08-16 14:59:59	11.690742	11.019713	4.596352	37.109778	428.658948	43.129930	
558	2016-02-09 18:00:00	6.765993	8.284344	2.733397	35.785207	499.238419	55.105532	
3300	2016-06-27 02:59:59	11.876232	11.504955	4.358123	40.850050	447.195722	58.545776	

5 rows × 87 columns

	date	rougher.input.feed_au	rougher.input.feed_ag	rougher.input.feed_pb	rougher.input.feed_sol	rougher.input.feed_rate	rougher.input.feed_size	rougher
0	2016-01-15 00:00:00	6.486150	6.100378	2.284912	36.808594	523.546326	55.486599	
1	2016-01-15 01:00:00	6.478583	6.161113	2.266033	35.753385	525.290581	57.278666	
2	2016-01-15 02:00:00	6.362222	6.116455	2.159622	35.971630	530.026610	57.510649	
3	2016-01-15 03:00:00	6.118189	6.043309	2.037807	36.862241	542.590390	57.792734	
4	2016-01-15 04:00:00	5.663707	6.060915	1.786875	34.347666	540.531893	56.047189	

5 rows × 87 columns

Количество пропусков по столбцам:

date - 0 шт. - 0.00%

rougher.input.feed_au - 0 шт. - 0.00%

rougher.input.feed_ag - 0 шт. - 0.00%

rougher.input.feed_pb - 100 шт. - 0.51%

rougher.input.feed_sol - 99 шт. - 0.51%

rougher.input.feed_rate - 11 шт. - 0.06%

rougher.input.feed_size - 145 шт. - 0.75%

rougher.input.floatbank10_sulfate - 34 шт. - 0.17%

rougher.input.floatbank10_xanthate - 8 шт. - 0.04%

rougher.state.floatbank10_a_air - 1 шт. - 0.01%

rougher.state.floatbank10_a_level - 1 шт. - 0.01%

rougher.state.floatbank10_b_air - 1 шт. - 0.01%

rougher.state.floatbank10_b_level - 1 шт. - 0.01%

rougher.state.floatbank10_c_air - 1 шт. - 0.01%

rougher.state.floatbank10_c_level - 1 шт. - 0.01%

rougher.state.floatbank10_d_air - 0 шт. - 0.00%

rougher.state.floatbank10_d_level - 0 шт. - 0.00%

rougher.state.floatbank10_e_air - 436 шт. - 2.24%

rougher.state.floatbank10_e_level - 0 шт. - 0.00%

rougher.state.floatbank10_f_air - 0 шт. - 0.00%

rougher.state.floatbank10_f_level - 0 шт. - 0.00%

rougher.input.floatbank11_sulfate - 44 шт. - 0.23%

rougher.input.floatbank11_xanthate - 453 шт. - 2.33%

rougher.calculation.sulfate_to_au_concentrate - 2 шт. - 0.01%

rougher.calculation.floatbank10_sulfate_to_au_feed - 2 шт. - 0.01%

rougher.calculation.floatbank11_sulfate_to_au_feed - 2 шт. - 0.01%

rougher.calculation.au_pb_ratio - 0 шт. - 0.00%

rougher.output.concentrate_au - 0 шт. - 0.00%

rougher.output.concentrate_ag - 0 шт. - 0.00%

rougher.output.concentrate_pb - 0 шт. - 0.00%

rougher.output.concentrate_sol - 23 шт. - 0.12%

rougher.output.recovery - 0 шт. - 0.00%

rougher.output.tail_au - 0 шт. - 0.00%

rougher.output.tail_ag - 1 шт. - 0.01%

rougher.output.tail_pb - 0 шт. - 0.00%

rougher.output.tail_sol - 0 шт. - 0.00%

primary_cleaner.input.sulfate - 24 шт. - 0.12%

primary_cleaner.input.depressant - 37 шт. - 0.19%

primary_cleaner.input.feed_size - 0 шт. - 0.00%

primary_cleaner.input.xanthate - 104 шт. - 0.54%

primary_cleaner.state.floatbank8_a_air - 4 шт. - 0.02%

primary_cleaner.state.floatbank8_a_level - 1 шт. - 0.01%

primary_cleaner.state.floatbank8_b_air - 4 шт. - 0.02%

primary_cleaner.state.floatbank8_b_level - 1 шт. - 0.01%

primary_cleaner.state.floatbank8_c_air - 2 шт. - 0.01%

primary_cleaner.state.floatbank8_c_level - 1 шт. - 0.01%

primary_cleaner.state.floatbank8_d_air - 3 шт. - 0.02%

primary_cleaner.state.floatbank8_d_level - 1 шт. - 0.01%

primary_cleaner.output.concentrate_au - 0 шт. - 0.00%

primary_cleaner.output.concentrate_ag - 0 шт. - 0.00%

primary_cleaner.output.concentrate_pb - 116 шт. - 0.60%

primary_cleaner.output.concentrate_sol - 370 шт. - 1.90%

primary_cleaner.output.tail_au - 0 шт. - 0.00%

primary_cleaner.output.tail_ag - 4 шт. - 0.02%

primary_cleaner.output.tail_pb - 21 шт. - 0.11%

primary_cleaner.output.tail_sol - 62 шт. - 0.32%

secondary_cleaner.state.floatbank2_a_air - 220 шт. - 1.13%

secondary_cleaner.state.floatbank2_a_level - 1 шт. - 0.01%

secondary_cleaner.state.floatbank2_b_air - 23 шт. - 0.12%

secondary_cleaner.state.floatbank2_b_level - 1 шт. - 0.01%

secondary_cleaner.state.floatbank3_a_air - 13 шт. - 0.07%

secondary_cleaner.state.floatbank3_a_air - 13 шт. - 0.07%

secondary_cleaner.state.floatbank3_a_level - 1 шт. - 0.01%

secondary_cleaner.state.floatbank3_b_air - 1 шт. - 0.01%

secondary_cleaner.state.floatbank3_b_level - 1 шт. - 0.01%

secondary_cleaner.state.floatbank4_a_air - 6 шт. - 0.03%

secondary_cleaner.state.floatbank4_a_level - 1 шт. - 0.01%

secondary_cleaner.state.floatbank4_b_air - 1 шт. - 0.01%

secondary_cleaner.state.floatbank4_b_level - 1 шт. - 0.01%

secondary_cleaner.state.floatbank5_a_air - 1 шт. - 0.01%

secondary_cleaner.state.floatbank5_a_level - 1 шт. - 0.01%

secondary_cleaner.state.floatbank5_b_air - 1 шт. - 0.01%

secondary_cleaner.state.floatbank5_b_level - 1 шт. - 0.01%

secondary_cleaner.state.floatbank6_a_air - 2 шт. - 0.01%

secondary_cleaner.state.floatbank6_a_level - 1 шт. - 0.01%

secondary_cleaner.output.tail_au - 0 шт. - 0.00%

secondary_cleaner.output.tail_ag - 2 шт. - 0.01%

secondary_cleaner.output.tail_pb - 12 шт. - 0.06%

secondary_cleaner.output.tail_sol - 1748 шт. - 8.99%

final.output.concentrate_au - 0 шт. - 0.00%

final.output.concentrate_ag - 1 шт. - 0.01%

final.output.concentrate_pb - 1 шт. - 0.01%

final.output.concentrate_sol - 211 шт. - 1.09%

final.output.recovery - 0 шт. - 0.00%

final.output.tail_au - 0 шт. - 0.00%

final.output.tail_ag - 1 шт. - 0.01%

final.output.tail_pb - 101 шт. - 0.52%

final.output.tail_sol - 6 шт. - 0.03%

Количество явных дубликатов: 0

Описание данных

Технологический процесс

- Rougher feed — исходное сырье
- Rougher additions (или reagent additions) — флотационные реагенты: Xanthate, Sulphate, Depressant
  - Xanthate **— ксантогенат (промотер, или активатор флотации);
  - Sulphate — сульфат (на данном производстве сульфид натрия);
  - Depressant — депрессант (силикат натрия).
- Rougher process (англ. «грубый процесс») — флотация
- Rougher tails — отвальные хвосты
- Float banks — флотационная установка
- Cleaner process — очистка
- Rougher Au — черновой концентрат золота
- Final Au — финальный концентрат золота

Параметры этапов

- air amount — объём воздуха
- fluid levels — уровень жидкости
- feed size — размер гранул сырья
- feed rate — скорость подачи

Наименование признаков

Наименование признаков должно быть такое:

[этап].[тип_параметра].[название_параметра]

Пример: rougher.input.feed_ag

Возможные значения для блока [этап]:

- rougher — флотация
- primary_cleaner — первичная очистка
- secondary_cleaner — вторичная очистка
- final — финальные характеристики

Возможные значения для блока [тип_параметра]:

- input — параметры сырья
- output — параметры продукта
- state — параметры, характеризующие текущее состояние этапа
- calculation — расчётные характеристики

"Обучающая выборка" (data_train):

- Состоит из 14149 объектов
- Имеет 87 признаков
- Явные дубликаты отсутствуют
- Имеет пропуски в большинстве признаков
  - В основном в районе 1% от общего количества данных
  - В признаке "secondary_cleaner.output.tail_sol" (отвальные хвосты солей после вторичной очистки) пропусков больше 10%

"Тестовая выборка" (data_test):

- Состоит из 5290 объектов
- Имеет 53 признаков
- Явные дубликаты отсутствуют
- Имеет незначительные пропуски (все меньше 1% от общего количества данных)

"Исходные данные" (data_full):

- Состоит из 19439 объектов (соответствует общей сумме объектов обучающей и тестовой выборок)
- Имеет 87 признаков
- Явные дубликаты отсутствуют
- Имеет пропуски в большинстве признаков
  - В основном меньше 1% от общего количества данных
  - В признаке "secondary_cleaner.output.tail_sol" (отвальные хвосты солей после вторичной очистки) пропусков около 9%

Рассчет эффективности rougher.output.recovery (MAE)

Эффективность обогащения рассчитывается по формуле:

$$\text{$Recovery = \dfrac{C \times (F-T)}{F \times (C-T)} \times 100\% $}$$
где:

- C — доля золота в концентрате после флотации/очистки;
- F — доля золота в сырье/концентрате до флотации/очистки;
- T — доля золота в отвальных хвостах после флотации/очистки.

In [8]:
```
C = data_train['rougher.output.concentrate_au']
F = data_train['rougher.input.feed_au']
```

```
T = data_train['rougher.output.tail_au']
recovery = (C * (F-T)) / (F * (C - T)) * 100
```

In [9]:
```
print("Средняя рассчитанная эффективность:", recovery.mean())
print("Средняя указанная эффективность:", data_train['rougher.output.recovery'].mean())
print()
print('MAE = {:.16f}'.format(mae(data_train['rougher.output.recovery'], recovery)))
```

```
Средняя рассчитанная эффективность: 82.52119968211305
Средняя указанная эффективность: 82.52119968211304

MAE = 0.0000000000000098
```
Средняя абсолютная ошибка мала - эффективность обогащения рассчитана правильно

Анализ признаков, недоступных в тестовой выборке

In [10]:
```
print('Столбцы, отсутствующие в "data_test"')
display(set(data_full.columns) - set(data_test.columns))
```

```
Столбцы, отсутствующие в "data_test"
{'final.output.concentrate_ag',
 'final.output.concentrate_au',
 'final.output.concentrate_pb',
 'final.output.concentrate_sol',
 'final.output.recovery',
 'final.output.tail_ag',
 'final.output.tail_au',
 'final.output.tail_pb',
 'final.output.tail_sol',
 'primary_cleaner.output.concentrate_ag',
 'primary_cleaner.output.concentrate_au',
 'primary_cleaner.output.concentrate_pb',
 'primary_cleaner.output.concentrate_sol',
 'primary_cleaner.output.tail_ag',
 'primary_cleaner.output.tail_au',
 'primary_cleaner.output.tail_pb',
 'primary_cleaner.output.tail_sol',
 'rougher.calculation.au_pb_ratio',
 'rougher.calculation.floatbank10_sulfate_to_au_feed',
 'rougher.calculation.floatbank11_sulfate_to_au_feed',
 'rougher.calculation.sulfate_to_au_concentrate',
 'rougher.output.concentrate_ag',
 'rougher.output.concentrate_au',
 'rougher.output.concentrate_pb',
 'rougher.output.concentrate_sol',
 'rougher.output.recovery',
 'rougher.output.tail_ag',
 'rougher.output.tail_au',
 'rougher.output.tail_pb',
 'rougher.output.tail_sol',
 'secondary_cleaner.output.tail_ag',
 'secondary_cleaner.output.tail_au',
 'secondary_cleaner.output.tail_pb',
 'secondary_cleaner.output.tail_sol'}
```

- "rougher.output.recovery" и "final.output.recovery" - целевые признаки для нашей модели
- Остальные отсутствующие признаки содержат информацию:
  - Либо о параметрах продукта после всех этапов флотации и очисток
  - Либо расчетные характеристики
- Данные признаки не нужны для тестирования нашей модели

Предобработка данных

Судя по признаку "date" данные обновляются каждый час. Соседние объекты в каждом признаке примерно одинаковы, таким образом можем заполнить пропуски методом "ffill" (нулевые значения заменяются данными из предыдущеей строки).

Заполним пропуски в обучающей и тестовой выборках.

Исходные данные оставим без изменений.

In [11]:
```
for i in datas:
    print('Количество пропусков до замены в "{}": {} шт.'.format(i, datas[i].isna().sum().sum()))
```

```
Количество пропусков до замены в "Обучающая выборка": 3050 шт.
Количество пропусков до замены в "Тестовая выборка": 375 шт.
Количество пропусков до замены в "Исходные данные": 4481 шт.
```

In [12]:
```
data_train = data_train.ffill().bfill()
data_test = data_test.ffill().bfill()
```

In [13]:
```
datas = {'Обучающая выборка' : data_train, 'Тестовая выборка' : data_test, 'Исходные данные' : data_full}
for i in datas:
    print('Количество пропусков после замены в "{}": {} шт.'.format(i, datas[i].isna().sum().sum()))
```

Количество пропусков после замены в "Обучающая выборка": 0 шт.
Количество пропусков после замены в "Тестовая выборка": 0 шт.
Количество пропусков после замены в "Исходные данные": 4481 шт.

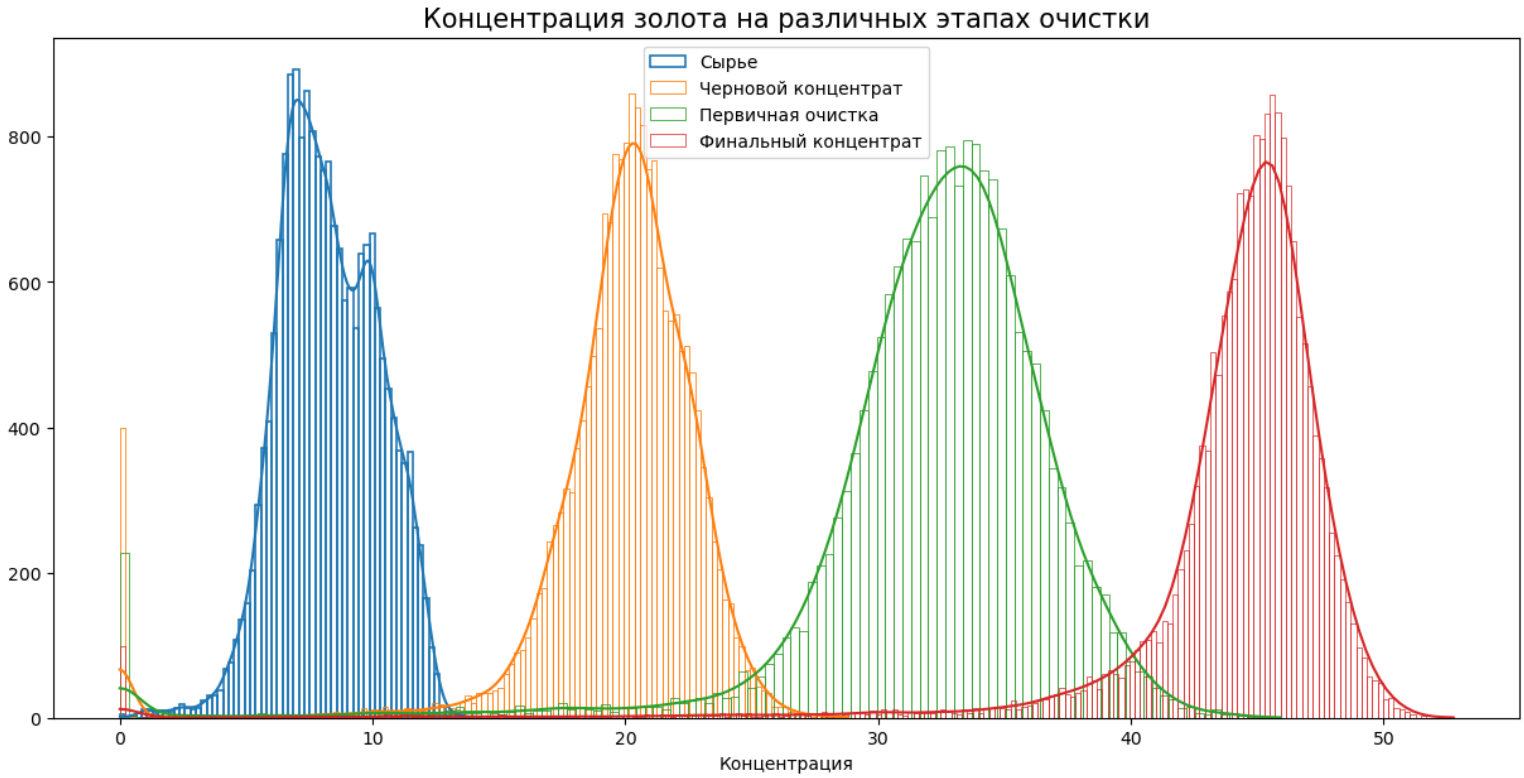
Анализ данных

Концентрация металлов (Au, Ag, Pb) на различных этапах очистки

In [14]:
```
def concentrate_hist(chem_element, title):
    x1 = data_full[f'rougher.input.feed_{chem_element}'].ffill().bfill()
    x2 = data_full[f'rougher.output.concentrate_{chem_element}'].ffill().bfill()
    x3 = data_full[f'primary_cleaner.output.concentrate_{chem_element}'].ffill().bfill()
    x4 = data_full[f'final.output.concentrate_{chem_element}'].ffill().bfill()

    f, ax = plt.subplots(figsize=(15, 7))

    sns.histplot(x1, fill=False, label="Сырье", kde=True);
    sns.histplot(x2, fill=False, label="Черновой концентрат", kde=True);
    sns.histplot(x3, fill=False, label="Первичная очистка", kde=True);
    sns.histplot(x4, fill=False, label="Финальный концентрат", kde=True);

    plt.legend();

    plt.title(label=f'Концентрация {title} на различных этапах очистки', fontsize=15)
    plt.xlabel('Концентрация')
    plt.ylabel(' ')
```

Золото

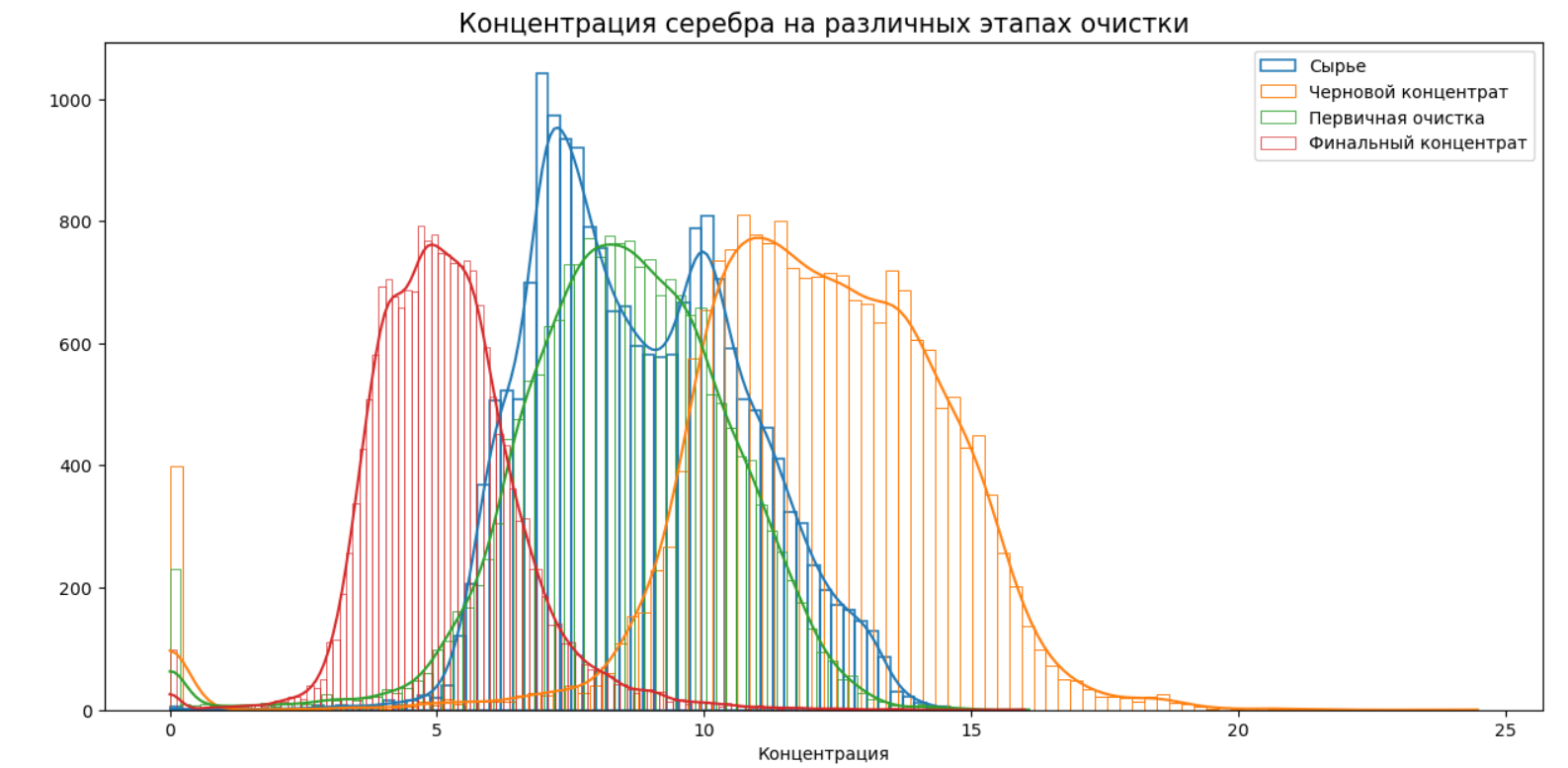
In [15]:
```
concentrate_hist('au', 'золота')
```



Концентрация золота увеличивается после каждого этапа очистки

Серебро

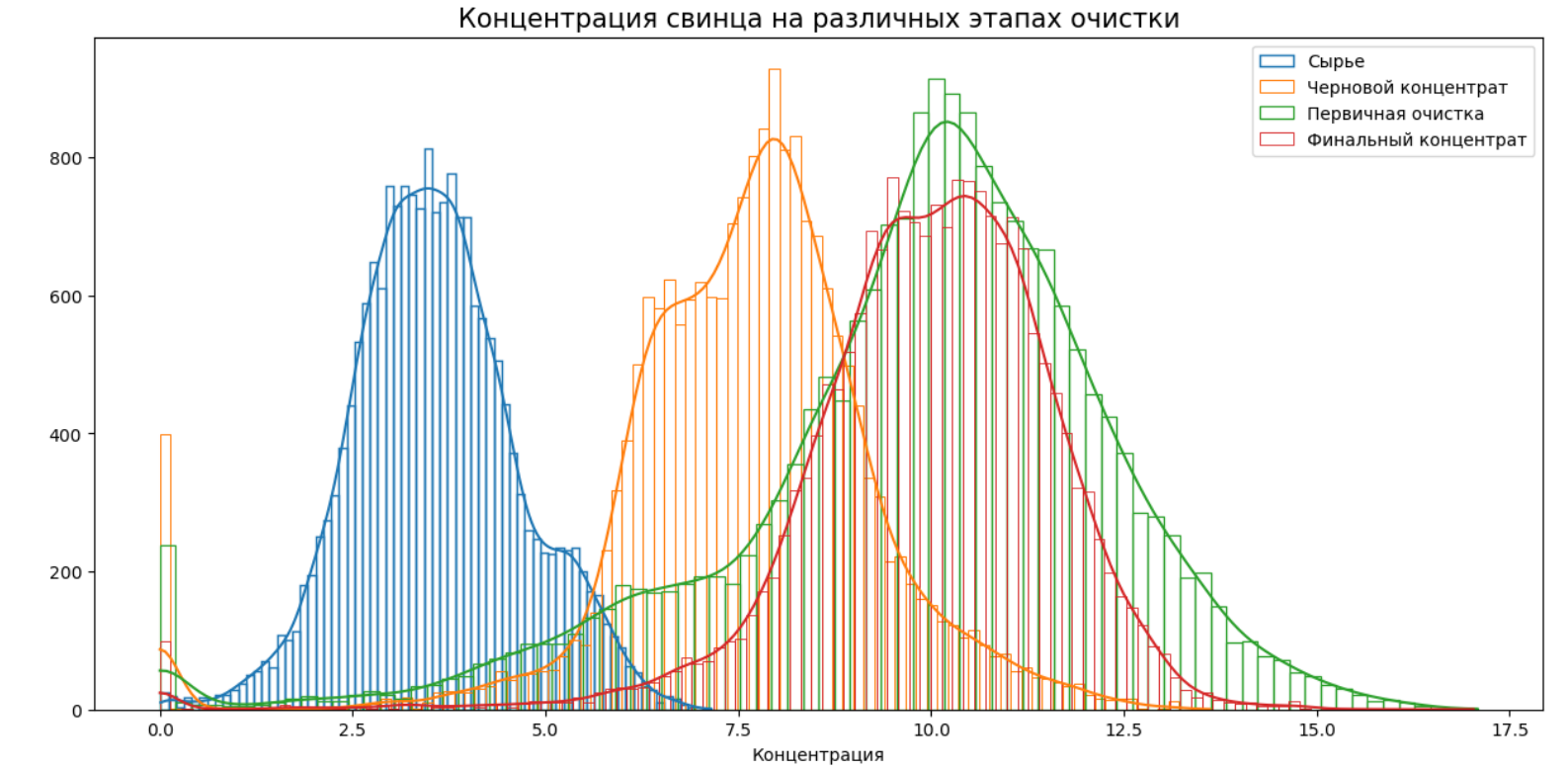
In [16]:
```
concentrate_hist('ag', 'серебра')
```



Концентрация серебра увеличивается после флотации и резко уменьшается после второго этапа очистки

Свинец

In [17]:
```
concentrate_hist('pb', 'свинца')
```



Концентрация свинца значительно увеличивается после флотации и незначительно увеличивается на оставшихся этапах

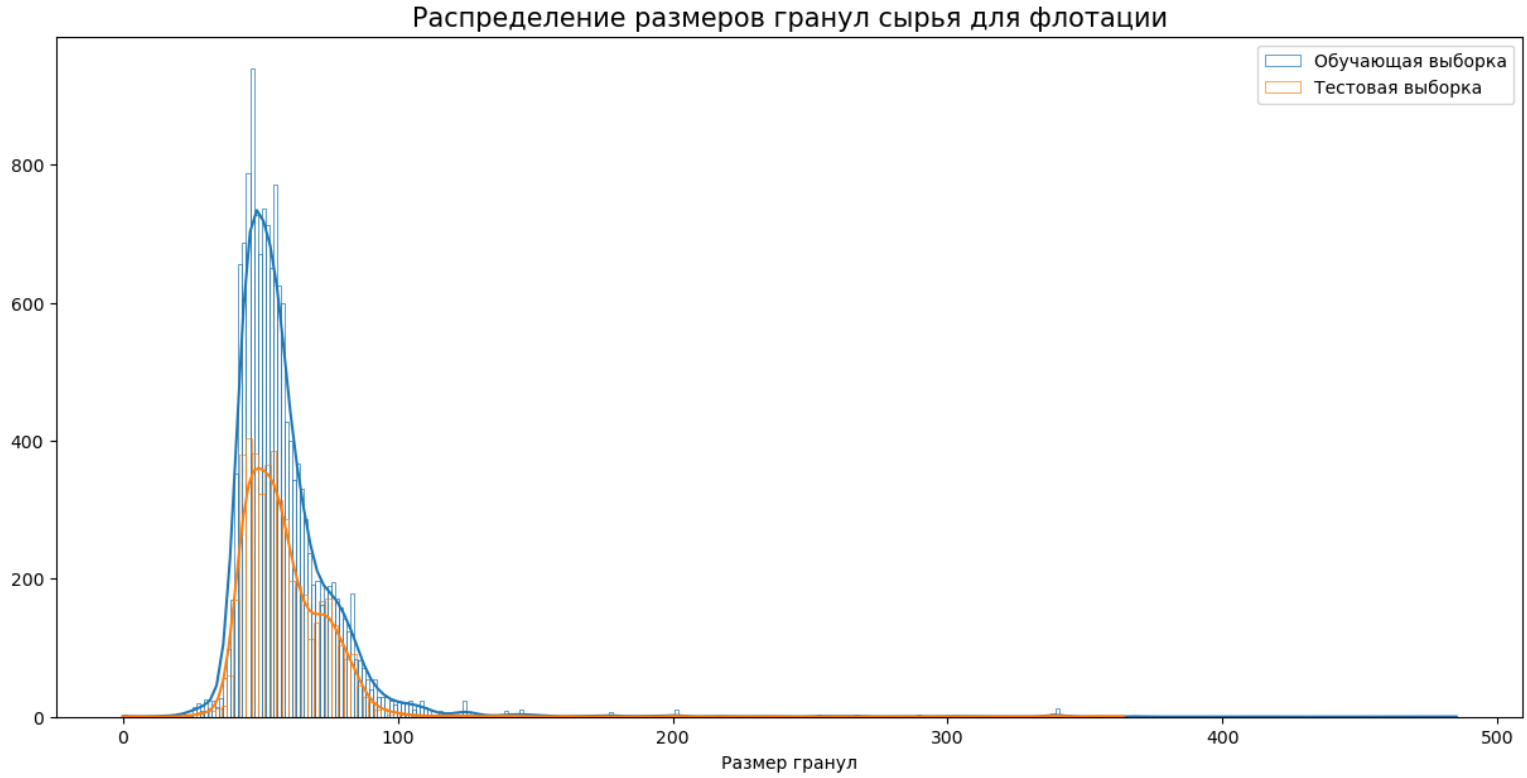
Сравнение распределения размеров гранул сырья на обучающей и тестовой выборках

In [18]:
```
def granules_hist(stage, title):
    x1 = data_train[f'{stage}.input.feed_size']
    x2 = data_test[f'{stage}.input.feed_size']

    f, ax = plt.subplots(figsize=(15, 7))

    sns.histplot(x1, fill=False, label="Обучающая выборка", kde=True);
    sns.histplot(x2, fill=False, label="Тестовая выборка", kde=True);

    plt.legend();

    plt.title(label=f'Распределение размеров гранул сырья для {title}', fontsize=15)
    plt.xlabel('Размер гранул')
```

```
    plt.ylabel(' ')
```

Этап флотации

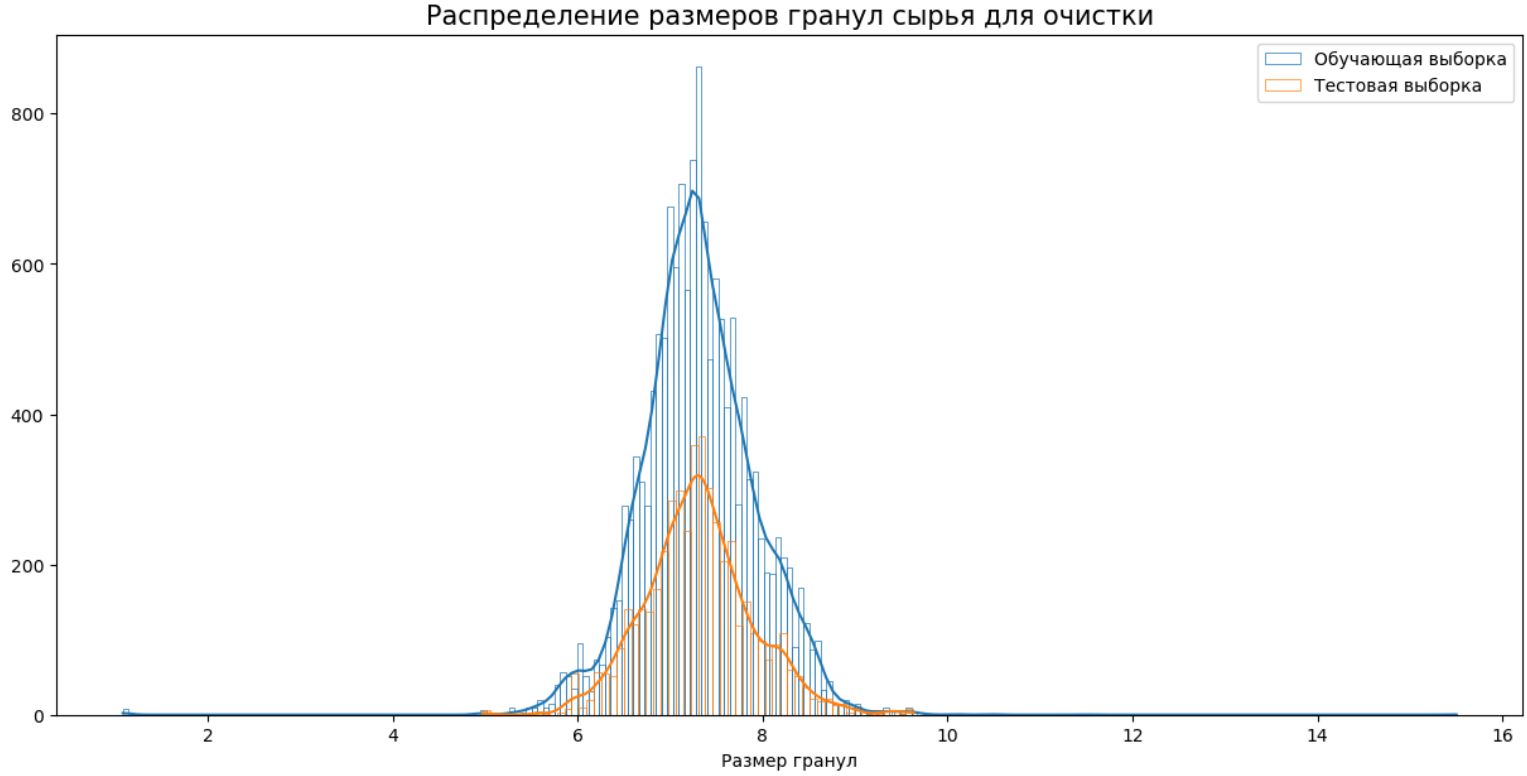
In [19]:
```
granules_hist('rougher', 'флотации')
```



Распределение размеров гранул примерно одинаково на обеих выборках

Этап очистки

In [20]:
```
granules_hist('primary_cleaner', 'очистки')
```

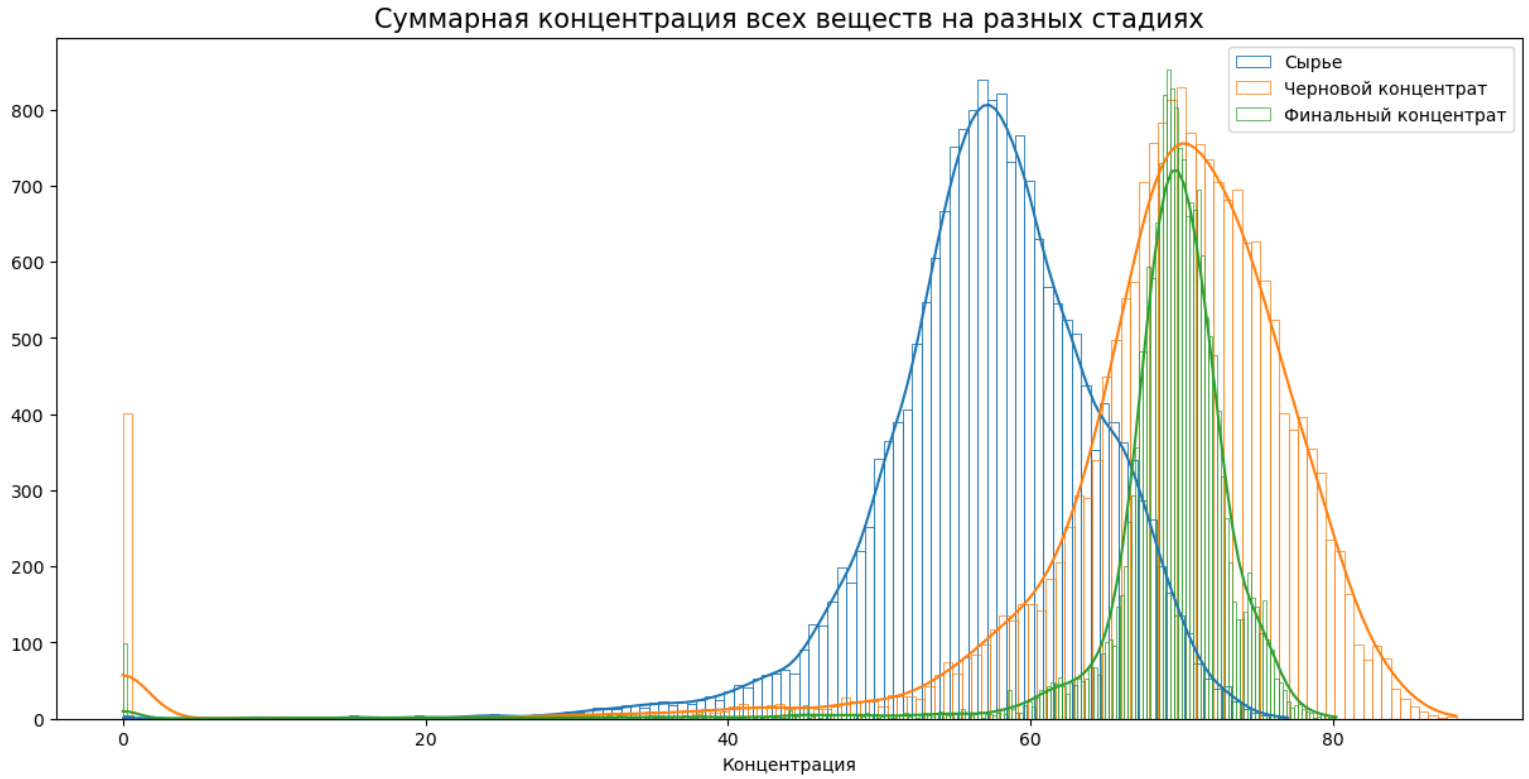


Распределение размеров гранул примерно одинаково на обеих выборках

Исследование суммарной концентрации всех веществ на разных стадиях

In [21]:
```
stages = ['rougher.input.feed_', 'rougher.output.concentrate_', 'final.output.concentrate_']
chem_element = ['au', 'ag', 'pb', 'sol']

for stage in stages:
    data_full[stage + 'sum'] = 0
    for el in chem_element:
        data_full[stage + 'sum'] += data_full[stage + el]
```

In [22]:
```
x1 = data_full['rougher.input.feed_sum'].ffill().bfill()
x2 = data_full['rougher.output.concentrate_sum'].ffill().bfill()
x3 = data_full['final.output.concentrate_sum'].ffill().bfill()

f, ax = plt.subplots(figsize=(15, 7))

sns.histplot(x1, fill=False, label="Сырье", kde=True);
sns.histplot(x2, fill=False, label="Черновой концентрат", kde=True);
sns.histplot(x3, fill=False, label="Финальный концентрат", kde=True);

plt.legend();

plt.title(label='Суммарная концентрация всех веществ на разных стадиях', fontsize=15);
plt.xlabel('Концентрация');
plt.ylabel(' ');
```



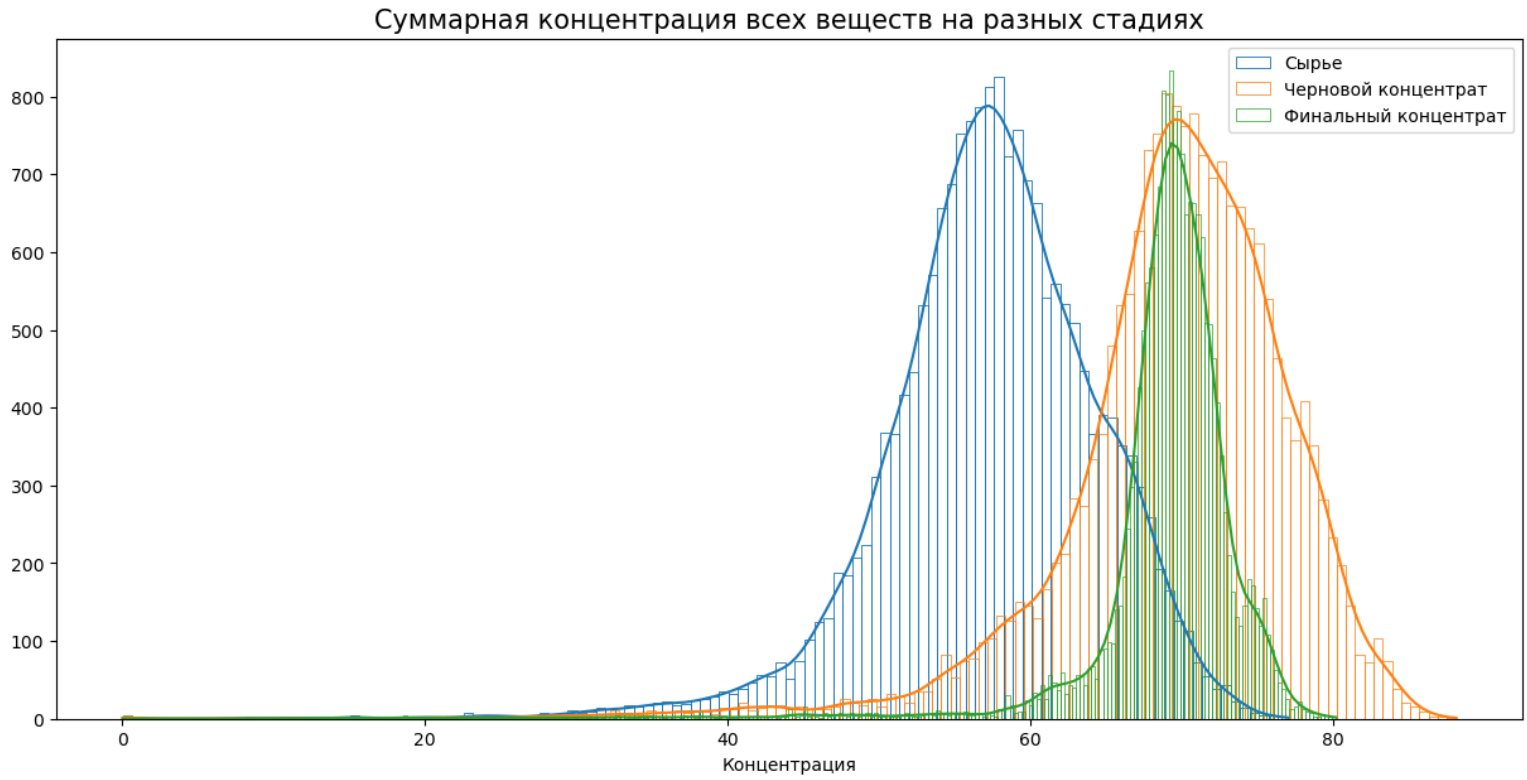
- Суммарная концентрация всех веществ после этапа флотации увеличивается
- После этапов очистки сокращается интервал распределения
- В данных присутствуют аномалии в виде суммарной концентрации веществ равной нулю
  - Удалим аномалии из исходных данных и обеих выборок

In [23]:
```
anomaly_date = data_full['date'][(data_full['rougher.input.feed_sum']==0)|(
    data_full['rougher.output.concentrate_sum']==0)|(data_full['final.output.concentrate_sum']==0)]

print('Общее количество аномалий:', anomaly_date.count())
print()

datas = {'Обучающая выборка' : data_train, 'Тестовая выборка' : data_test, 'Исходные данные' : data_full}
for i in datas:
    print('Количество аномалий в {} - {} шт.'.format(i, datas[i]['date'][datas[i]['date'].isin(anomaly_date)].count()))

data_train = data_train[~data_train['date'].isin(anomaly_date)]
data_test = data_test[~data_test['date'].isin(anomaly_date)]
data_full = data_full[~data_full['date'].isin(anomaly_date)]

print()

datas = {'Обучающая выборка' : data_train, 'Тестовая выборка' : data_test, 'Исходные данные' : data_full}
for i in datas:
    print('Количество аномалий в {} после удаления - {} шт.'.format(
        i, datas[i]['date'][datas[i]['date'].isin(anomaly_date)].count()))
```

```
Общее количество аномалий: 490

Количество аномалий в Обучающая выборка - 490 шт.
Количество аномалий в Тестовая выборка - 0 шт.
Количество аномалий в Исходные данные - 490 шт.

Количество аномалий в Обучающая выборка после удаления - 0 шт.
Количество аномалий в Тестовая выборка после удаления - 0 шт.
Количество аномалий в Исходные данные после удаления - 0 шт.
```

In [24]:
```
x1 = data_full['rougher.input.feed_sum'].ffill().bfill()
```

```
x2 = data_full['rougher.output.concentrate_sum'].ffill().bfill()
x3 = data_full['final.output.concentrate_sum'].ffill().bfill()

f, ax = plt.subplots(figsize=(15, 7))

sns.histplot(x1, fill=False, label="Сырье", kde=True);
sns.histplot(x2, fill=False, label="Черновой концентрат", kde=True);
sns.histplot(x3, fill=False, label="Финальный концентрат", kde=True);

plt.legend();

plt.title(label='Суммарная концентрация всех веществ на разных стадиях', fontsize=15);
plt.xlabel('Концентрация');
plt.ylabel(' ');
```



Аномалии в данных отсутствуют

Модель

Функция для вычисления итоговой sMAPE

Для решения задачи используем метрику качества — sMAPE (англ. Symmetric Mean Absolute Percentage Error, «симметричное среднее абсолютное процентное отклонение»).

Метрика sMAPE рассчитывается по формуле:

$$\text{$sMAPE = \dfrac{1}{N}\sum_{i=1}^{N} \dfrac{|y_i - \hat y_i|}{(|y_i| + |\hat y_i|)\div 2} \times 100\% $}$$
где:

- y_i — Значение целевого признака для объекта с порядковым номером i в выборке, на которой измеряется качество
- $\hat y_i$ — Значение предсказания для объекта с порядковым номером i, например, в тестовой выборке
- N — Количество объектов в выборке
- $\sum_{i=1}^{N}$ — Суммирование по всем объектам выборки (i меняется от 1 до N)

Нашей модели нужно спрогнозировать сразу две величины:

1. Эффективность обогащения чернового концентрата (**rougher.output.recovery**)
2. Эффективность обогащения финального концентрата (**final.output.recovery**)

Итоговая метрика складывается из двух величин:

$$\text{${Итоговое}\:{sMAPE} = 25\% \times sMAPE(rougher) + 75\% \times sMAPE(final)$}$$

In [25]:
```
def smape(target, pred):
    smape = abs(target - pred) / ((abs(target) + abs(pred)) / 2) * 100
    smape = smape.fillna(value=0)
    smape = sum(smape) / len(smape)
    return smape

def smape_final(smape_rougher, smape_final):
    return 0.25 * smape_rougher + 0.75 * smape_final

smape_score = make_scorer(smape, greater_is_better=False)
```

Подготовка данных для обучения

In [26]:
```
target_rougher_test = pd.merge(data_test, data_full, how='left', on=['date'])
target_rougher_test = target_rougher_test['rougher.output.recovery']
target_final_test = pd.merge(data_test, data_full, how='left', on=['date'])
target_final_test = target_final_test['final.output.recovery']

data_test = data_test.drop(['date'], axis=1)

col = data_test.columns
features = data_train[col]
target_rougher = data_train['rougher.output.recovery']
target_final = data_train['final.output.recovery']

features_list = {'features' : features}
targets_list = {'target_rougher' : target_rougher, 'target_final' : target_final}

print('Вспомогательные признаки:')
print()
for i in features_list:
    sh = features_list[i].shape
    psh = sh[0]/len(data_train)
    print('{} - Объектов: {} шт., признаков: {} шт. - {:.2%}'.format(i, sh[0], sh[1], psh))

print()

print('Целевые признаки:')
print()
for i in targets_list:
    sh = targets_list[i].shape
    psh = sh[0]/len(data_train)
    print('{} - Объектов: {} шт. - {:.2%}'.format(i, sh[0], psh))
```

```
Вспомогательные признаки:

features - Объектов: 14089 шт., признаков: 52 шт. - 100.00%

Целевые признаки:

target_rougher - Объектов: 14089 шт. - 100.00%
target_final - Объектов: 14089 шт. - 100.00%
```

Стандартизируем численные признаки в выборках, так как в данных присутствуют значения в разных масштабах

In [27]:
```
pd.options.mode.chained_assignment = None

features_list = {'features' : features, 'data_test' : data_test}

scaler = StandardScaler()
scaler.fit(features[col])

for i in features_list:
    features_list[i][col] = scaler.transform(features_list[i][col])
    print('{}:'.format(i))
    display(features_list[i].sample(5, random_state=r_state))
    print()
```

```
features:
```

	rougher.input.feed_au	rougher.input.feed_ag	rougher.input.feed_pb	rougher.input.feed_sol	rougher.input.feed_rate	rougher.input.feed_size	rougher.input
4471	1.216444	0.193767	-0.049684	-0.272741	-0.368005	-0.606031	
8621	-0.654844	-1.100730	-0.354170	-2.171698	-0.050536	-0.227404	
5910	0.712961	0.788081	0.145395	0.481891	-0.176740	-0.567061	
7012	0.629188	0.739351	0.829257	0.665510	-0.311252	-0.252994	
8363	-1.240525	-0.985350	-0.698040	-0.159647	0.332162	-0.345366	

5 rows × 52 columns

```
data_test:
```

	rougher.input.feed_au	rougher.input.feed_ag	rougher.input.feed_pb	rougher.input.feed_sol	rougher.input.feed_rate	rougher.input.feed_size	rougher.input
3781	1.353402	0.982176	0.315215	-0.854309	-0.685387	-0.675210	
716	-2.418251	-1.125227	-2.898236	0.815251	0.119995	0.461862	
4597	-0.284068	-0.661772	-1.156902	0.244627	0.274791	-0.491283	
4178	-1.408210	-1.041203	-0.673409	-2.445829	-0.771668	-0.569547	
3547	0.080532	-0.553554	0.412291	-1.786583	-0.749856	-0.831343	

5 rows × 52 columns

Поиск лучшей модели

Константная модель

In [28]:
```
smape_rougher_const = smape(target_rougher, target_rougher.mean())
smape_final_const = smape(target_final, target_final.mean())
score_const = smape_final(smape_rougher_const, smape_final_const)
print('Итоговое sMAPE константной модели:', score_const)
```

```
Итоговое sMAPE константной модели: 9.109564280483204
```

Decision Tree (Дерево решений)

In [29]:
```
best_score_DT = 1000
best_depth_DT = 0
for depth in tqdm(range(1,21)):
    model_decision_tree = DecisionTreeRegressor(random_state=r_state, max_depth=depth)
    smape_r = abs(cross_val_score(model_decision_tree, features, target_rougher, cv=5, scoring=smape_score))
    smape_f = abs(cross_val_score(model_decision_tree, features, target_final, cv=5, scoring=smape_score))
    final_score = smape_final(smape_r, smape_f).mean()
    if final_score < best_score_DT:
        best_score_DT = final_score
        best_depth_DT = depth

print()
print('Лучший результат:')
print('Глубина =', best_depth_DT, ', Итоговое sMAPE =', best_score_DT)
```

```
100%|████████████████████| 20/20 [03:07<00:00,  9.35s/it]
Лучший результат:
Глубина = 2 , Итоговое sMAPE = 8.779830899334977
```

Random Forest (Случайный лес)

In [30]:
```
best_score_RF = 1000
best_est_RF = 0
best_depth_RF = 0

for est in range(1, 101, 20):
    for depth in tqdm(range(1, 21, 5)):
        model_random_forest = RandomForestRegressor(random_state=r_state, n_estimators=est, max_depth=depth)
        smape_r = abs(cross_val_score(model_random_forest, features, target_rougher, cv=5, scoring=smape_score))
        smape_f = abs(cross_val_score(model_random_forest, features, target_final, cv=5, scoring=smape_score))
        final_score = smape_final(smape_r, smape_f).mean()
        if final_score < best_score_RF:
            best_score_RF = final_score
            best_est_RF = est
            best_depth_RF = depth

print()
print('Лучший результат:')
print('Количество деревьев =', best_est_RF,', глубина =', best_depth_RF, ', Итоговое sMAPE =', best_score_RF)
```

```
100%|████████████████████| 4/4 [00:26<00:00,  6.60s/it]
100%|████████████████████| 4/4 [07:01<00:00, 105.36s/it]
100%|████████████████████| 4/4 [12:52<00:00, 193.21s/it]
100%|████████████████████| 4/4 [19:10<00:00, 287.71s/it]
100%|████████████████████| 4/4 [25:53<00:00, 388.50s/it]
Лучший результат:
Количество деревьев = 61 , глубина = 6 , Итоговое sMAPE = 8.689526885467895
```

Linear Regression (Линейная регрессия)

In [31]:
```
best_score_LR = 1000
best_fit_intercept_LR = ''
fit_intercept = ['True', 'False']

for fit in tqdm(fit_intercept):
    model_linear_regression = LinearRegression(fit_intercept=fit_intercept)
    smape_r = abs(cross_val_score(model_linear_regression, features, target_rougher, cv=5, scoring=smape_score))
    smape_f = abs(cross_val_score(model_linear_regression, features, target_final, cv=5, scoring=smape_score))
    final_score = smape_final(smape_r, smape_f).mean()
    if final_score < best_score_LR:
        best_score_LR = final_score
        best_fit_intercept_LR = fit

print()
print('Лучший результат:')
print('fit_intercept =', best_fit_intercept_LR, ', Итоговое sMAPE =', best_score_LR)
```

```
100%|████████████████████| 2/2 [00:01<00:00,  1.18it/s]
Лучший результат:
fit_intercept = True , Итоговое sMAPE = 8.90180956927614
```

Определим лучшую модель

In [32]:
```
max_score = best_score_DT
if max_score > best_score_RF:
    max_score = best_score_RF
if max_score > best_score_LR:
    max_score = best_score_LR

if max_score == best_score_DT:
    print('Лучшая модель: Decision Tree (Дерево решений), с гиперпараметром max_depth =', best_depth_DT)
if max_score == best_score_RF:
    print('Лучшая модель: Random Forest (Случайный лес), с гиперпараметрами n_estimators =', best_est_RF,
      ', max_depth =', best_depth_RF)
if max_score == best_score_LR:
    print('Лучшая модель: Linear Regression (Линейная регрессия), с гиперпараметрами fit_intercept =', best_fit_intercept_LR)
print('Итоговое sMAPE =', max_score)

if max_score < score_const:
    print('Итоговое sMAPE модели меньше константной, модель подходит для обучения')
else:
    print('Итоговое sMAPE модели больше константной, модель не подходит для обучения')
```

```
Лучшая модель: Random Forest (Случайный лес), с гиперпараметрами n_estimators = 61 , max_depth = 6
Итоговое sMAPE = 8.689526885467895
Итоговое sMAPE модели меньше константной, модель подходит для обучения
```

Тестирование модели

In [33]:
```
model = RandomForestRegressor(random_state=r_state, n_estimators=best_est_RF, max_depth=best_depth_RF)
model.fit(features, target_rougher)
predictions_rougher_test = pd.Series(model.predict(data_test))
model.fit(features, target_final)
predictions_final_test = pd.Series(model.predict(data_test))
smape_r = abs(smape(target_rougher_test, predictions_rougher_test))
smape_f = abs(smape(target_final_test, predictions_final_test))
final_score = smape_final(smape_r, smape_f)

print('sMAPE на тестовой выборке:', final_score)
```

```
sMAPE на тестовой выборке: 6.334999771849558
```

- sMAPE на тестовой выборке меньше sMAPE лучшей модели
- sMAPE на тестовой выборке меньше константной
- Модель справилась с обучением

Общий вывод

Проведено исследование с целью подготовить прототип модели машинного обучения для компании «Цифры». Модель должна предсказать коэффициент восстановления золота из золотосодержащей руды.

Модель поможет оптимизировать производство, чтобы не запускать предприятие с убыточными характеристиками.

Данные с параметрами добычи и очистки получим из трех файлов:

- gold_recovery_train_new.csv — обучающая выборка
- gold_recovery_test_new.csv — тестовая выборка
- gold_recovery_full_new.csv — исходные данные

Исследование проходило в три этапа:

1. Подготовка данных
    - Изучение данных:
        - "Обучающая выборка" (data_train):
            - Состоит из 14149 объектов
            - Имеет 87 признаков
            - Явные дубликаты отсутствуют
            - Имеет пропуски в большинстве признаков
                - В основном в районе 1% от общего количества данных
                - В признаке "secondary_cleaner.output.tail_sol" (отвальные хвосты солей после вторичной очистки) пропусков больше 10%
        - "Тестовая выборка" (data_test):
            - Состоит из 5290 объектов
            - Имеет 53 признаков
            - Явные дубликаты отсутствуют
            - Имеет незначительные пропуски (все меньше 1% от общего количества данных)
        - "Исходные данные" (data_full):
            - Состоит из 19439 объектов (соответствует общей сумме объектов обучающей и тестовой выборок)
            - Имеет 87 признаков
            - Явные дубликаты отсутствуют

        - Явные дубликаты отсутствуют
        - Имеет пропуски в большинстве признаков
            - В основном меньше 1% от общего количества данных
            - В признаке "secondary_cleaner.output.tail_sol" (отвальные хвосты солей после вторичной очистки) пропусков около 9%
- Рассчет эффективности rougher.output.recovery (MAE):
    - Средняя рассчитанная эффективность: 82.70450164550293
    - Средняя указанная эффективность: 82.70450164550293
    - MAE = 0.0000000000000097
    - Средняя абсолютная ошибка мала - эффективность обогащения рассчитана правильно
- Анализ признаков, недоступных в тестовой выборке:
    - "rougher.output.recovery" и "final.output.recovery" - целевые признаки для нашей модели
    - Остальные отсутствующие признаки содержат информацию:
        - Либо о параметрах продукта после всех этапов флотации и очисток
        - Либо расчетные характеристики
    - Данные признаки не нужны для тестирования нашей модели
- Предобработка данных:
    - Судя по признаку "date" данные обновляются каждый час
    - Соседние объекты в каждом признаке примерно одинаковы
    - Мы заполнили пропуски в обучающей и тестовой выборках методом "ffill" (нулевые значения заменяются данными из предыдущеей строки)
    - Исходные данные оставили без изменений

2. Анализ данных
    - Концентрация металлов (Au, Ag, Pb) на различных этапах очистки:
        - Концентрация золота увеличивается после каждого этапа очистки
        - Концентрация серебра увеличивается после флотации и резко уменьшается после второго этапа очистки
        - Концентрация свинца значительно увеличивается после флотации и незначительно увеличивается на оставшихся этапах
    - Сравнение распределения размеров гранул сырья на обучающей и тестовой выборках:
        - Распределение размеров гранул на этапе флотации примерно одинаково на обеих выборках
        - Распределение размеров гранул на этапе очистки примерно одинаково на обеих выборках
    - Исследование суммарной концентрации всех веществ на разных стадиях:
        - Суммарная концентрация всех веществ после этапа флотации увеличивается
        - После этапов очистки сокращается интервал распределения
        - В данных присутствовали аномалии в виде суммарной концентрации веществ равной нулю
            - Мы удалили аномалии из исходных данных и обеих выборок

3. Построение модели
    - Для решения задачи мы использовали метрику качества — sMAPE (англ. Symmetric Mean Absolute Percentage Error, «симметричное среднее абсолютное процентное отклонение»)
    - Подготовка данных для обучения:
        - Удалили признак 'date' из тестовой и обучающей выборок, так как он не влияет на обучение
        - Стандартизировали численные признаки в выборках, так как в данных присутствовали значения в разных масштабах
    - Поиск лучшей модели:
        - Итоговое sMAPE константной модели: 9.51200558511382
        - Лучшая модель:
            - Random Forest (Случайный лес), с гиперпараметрами:
                - n_estimators = 81
                - max_depth = 6
            - Итоговое sMAPE = 8.715139034269836
        - Итоговое sMAPE лучшей модели меньше константной, модель подходит для обучения
        - Тестирование модели:
            - sMAPE на тестовой выборке: 7.290369716327468
            - sMAPE на тестовой выборке меньше sMAPE лучшей модели
            - sMAPE на тестовой выборке меньше константной
            - Модель справилась с обучением