Содержание

Восстановление золота из руды

В нашем распоряжении данные золото обогатительного предприятия.

Наша задача подготовить модель предсказания коэффициент восстановления золота из золотосодержащей руды.

В ходе работы необходимо:

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

In [1]
```
# импортируем библиотеки
import pandas as pd
from sklearn.metrics import mean_absolute_error
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import make_scorer
from sklearn.model_selection import GridSearchCV, KFold
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import LinearRegression
from sklearn.dummy import DummyRegressor
from sklearn.model_selection import cross_val_score
```

Подготовка данных

Обзор данных

Данные находятся в трёх файлах:

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

Данные индексируются датой и временем получения информации (признак `date`). Соседние по времени параметры часто похожи.

Откроим файлы с данными, изучим общую информацию.

In [2]
```
# читаем данные
try:
    data_train = pd.read_csv('/datasets/gold_recovery_train_new.csv')
    data_test = pd.read_csv('/datasets/gold_recovery_test_new.csv')
    data_full = pd.read_csv('/datasets/gold_recovery_full_new.csv')
except:
    data_train = pd.read_csv('https://restricted/datasets/gold_recovery_train_ne
    data_test = pd.read_csv('https://restricted/datasets/gold_recovery_test_new.
    data_full = pd.read_csv('https://restricted/datasets/gold_recovery_full_new.
```

In [3]
```
# Выведем общую инф о df
def df_info(df):
    display(df.head())
    display(df.info())
    display(df.describe())
    print("Количество дубликатов:", df.duplicated().sum())
```

Посмотрим на данные по обучающей выборке

In [4]
```
df_info(data_train)
```

	date	final.output.concentrate_ag	final.output.concentrate_pb	final.output.concentra
0	2016-01-15 00:00:00	6.055403	9.889648	5.5
1	2016-01-15 01:00:00	6.029369	9.968944	5.2
2	2016-01-15 02:00:00	6.055926	10.213995	5.3
3	2016-01-15 03:00:00	6.047977	9.977019	4.8
4	2016-01-15 04:00:00	6.148599	10.142511	4.9

5 rows × 87 columns

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14149 entries, 0 to 14148
Data columns (total 87 columns):
 #   Column                                              Non-Null Count  Dtype  
---  ------                                              --------------  -----  
 0   date                                                14149 non-null  object 
 1   final.output.concentrate_ag                         14148 non-null  float64
 2   final.output.concentrate_pb                         14148 non-null  float64
 3   final.output.concentrate_sol                        13938 non-null  float64
 4   final.output.concentrate_au                         14149 non-null  float64
 5   final.output.recovery                               14149 non-null  float64
 6   final.output.tail_ag                                14149 non-null  float64
 7   final.output.tail_pb                                14049 non-null  float64
 8   final.output.tail_sol                               14144 non-null  float64
 9   final.output.tail_au                                14149 non-null  float64
 10  primary_cleaner.input.sulfate                       14129 non-null  float64
 11  primary_cleaner.input.depressant                    14117 non-null  float64
 12  primary_cleaner.input.feed_size                     14149 non-null  float64
 13  primary_cleaner.input.xanthate                      14049 non-null  float64
 14  primary_cleaner.output.concentrate_ag               14149 non-null  float64
 15  primary_cleaner.output.concentrate_pb               14063 non-null  float64
 16  primary_cleaner.output.concentrate_sol              13863 non-null  float64
 17  primary_cleaner.output.concentrate_au               14149 non-null  float64
 18  primary_cleaner.output.tail_ag                      14148 non-null  float64
 19  primary_cleaner.output.tail_pb                      14134 non-null  float64
 20  primary_cleaner.output.tail_sol                     14103 non-null  float64
 21  primary_cleaner.output.tail_au                      14149 non-null  float64
 22  primary_cleaner.state.floatbank8_a_air              14145 non-null  float64
 23  primary_cleaner.state.floatbank8_a_level            14148 non-null  float64
 24  primary_cleaner.state.floatbank8_b_air              14145 non-null  float64
 25  primary_cleaner.state.floatbank8_b_level            14148 non-null  float64
 26  primary_cleaner.state.floatbank8_c_air              14147 non-null  float64
 27  primary_cleaner.state.floatbank8_c_level            14148 non-null  float64
 28  primary_cleaner.state.floatbank8_d_air              14146 non-null  float64
 29  primary_cleaner.state.floatbank8_d_level            14148 non-null  float64
 30  rougher.calculation.sulfate_to_au_concentrate       14148 non-null  float64
 31  rougher.calculation.floatbank10_sulfate_to_au_feed  14148 non-null  float64
 32  rougher.calculation.floatbank11_sulfate_to_au_feed  14148 non-null  float64
 33  rougher.calculation.au_pb_ratio                     14149 non-null  float64
 34  rougher.input.feed_ag                               14149 non-null  float64
 35  rougher.input.feed_pb                               14049 non-null  float64
 36  rougher.input.feed_rate                             14141 non-null  float64
 37  rougher.input.feed_size                             14005 non-null  float64
 38  rougher.input.feed_sol                              14071 non-null  float64
 39  rougher.input.feed_au                               14149 non-null  float64
 40  rougher.input.floatbank10_sulfate                   14120 non-null  float64
 41  rougher.input.floatbank10_xanthate                  14141 non-null  float64
 42  rougher.input.floatbank11_sulfate                   14113 non-null  float64
 43  rougher.input.floatbank11_xanthate                  13721 non-null  float64
 44  rougher.output.concentrate_ag                       14149 non-null  float64
 45  rougher.output.concentrate_pb                       14149 non-null  float64
 46  rougher.output.concentrate_sol                      14127 non-null  float64
 47  rougher.output.concentrate_au                       14149 non-null  float64
 48  rougher.output.recovery                             14149 non-null  float64
 49  rougher.output.tail_ag                              14148 non-null  float64
 50  rougher.output.tail_pb                              14149 non-null  float64
 51  rougher.output.tail_sol                             14149 non-null  float64
 52  rougher.output.tail_au                              14149 non-null  float64
 53  rougher.state.floatbank10_a_air                     14148 non-null  float64
 54  rougher.state.floatbank10_a_level                   14148 non-null  float64
```

```
 55  rougher.state.floatbank10_b_air                     14148 non-null  float64
 56  rougher.state.floatbank10_b_level                   14148 non-null  float64
 57  rougher.state.floatbank10_c_air                     14148 non-null  float64
 58  rougher.state.floatbank10_c_level                   14148 non-null  float64
 59  rougher.state.floatbank10_d_air                     14149 non-null  float64
 60  rougher.state.floatbank10_d_level                   14149 non-null  float64
 61  rougher.state.floatbank10_e_air                     13713 non-null  float64
 62  rougher.state.floatbank10_e_level                   14149 non-null  float64
 63  rougher.state.floatbank10_f_air                     14149 non-null  float64
 64  rougher.state.floatbank10_f_level                   14149 non-null  float64
 65  secondary_cleaner.output.tail_ag                    14147 non-null  float64
 66  secondary_cleaner.output.tail_pb                    14139 non-null  float64
 67  secondary_cleaner.output.tail_sol                   12544 non-null  float64
 68  secondary_cleaner.output.tail_au                    14149 non-null  float64
 69  secondary_cleaner.state.floatbank2_a_air            13932 non-null  float64
 70  secondary_cleaner.state.floatbank2_a_level          14148 non-null  float64
 71  secondary_cleaner.state.floatbank2_b_air            14128 non-null  float64
 72  secondary_cleaner.state.floatbank2_b_level          14148 non-null  float64
 73  secondary_cleaner.state.floatbank3_a_air            14145 non-null  float64
 74  secondary_cleaner.state.floatbank3_a_level          14148 non-null  float64
 75  secondary_cleaner.state.floatbank3_b_air            14148 non-null  float64
 76  secondary_cleaner.state.floatbank3_b_level          14148 non-null  float64
 77  secondary_cleaner.state.floatbank4_a_air            14143 non-null  float64
 78  secondary_cleaner.state.floatbank4_a_level          14148 non-null  float64
 79  secondary_cleaner.state.floatbank4_b_air            14148 non-null  float64
 80  secondary_cleaner.state.floatbank4_b_level          14148 non-null  float64
 81  secondary_cleaner.state.floatbank5_a_air            14148 non-null  float64
 82  secondary_cleaner.state.floatbank5_a_level          14148 non-null  float64
 83  secondary_cleaner.state.floatbank5_b_air            14148 non-null  float64
 84  secondary_cleaner.state.floatbank5_b_level          14148 non-null  float64
 85  secondary_cleaner.state.floatbank6_a_air            14147 non-null  float64
 86  secondary_cleaner.state.floatbank6_a_level          14148 non-null  float64
dtypes: float64(86), object(1)
memory usage: 9.4+ MB
None
```

	final.output.concentrate_ag	final.output.concentrate_pb	final.output.concentrate_so
count	14148.000000	14148.000000	13938.000000
mean	5.142034	10.132960	9.202849
std	1.369586	1.654930	2.790516
min	0.000000	0.000000	0.000000
25%	4.211620	9.297355	7.484645
50%	4.994652	10.297144	8.845462
75%	5.859540	11.170603	10.487508
max	16.001945	17.031899	18.124851

8 rows × 86 columns

Данные представлены в 87 столбцах и 14149 строках, дубликаты отсутствуют. Есть пропуски в данных

Посмотрим на данные по тестовой выборке

In [5]
```
df_info(data_test)
```

	date	primary_cleaner.input.sulfate	primary_cleaner.input.depressant	primary_cleane
0	2016-09-01 00:59:59	210.800909	14.993118	
1	2016-09-01 01:59:59	215.392455	14.987471	
2	2016-09-01 02:59:59	215.259946	12.884934	
3	2016-09-01 03:59:59	215.336236	12.006805	
4	2016-09-01 04:59:59	199.099327	10.682530	

5 rows × 53 columns

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5290 entries, 0 to 5289
Data columns (total 53 columns):
 #   Column                                      Non-Null Count  Dtype  
---  ------                                      --------------  -----  
 0   date                                        5290 non-null   object 
 1   primary_cleaner.input.sulfate               5286 non-null   float64
 2   primary_cleaner.input.depressant            5285 non-null   float64
 3   primary_cleaner.input.feed_size             5290 non-null   float64
 4   primary_cleaner.input.xanthate              5286 non-null   float64
 5   primary_cleaner.state.floatbank8_a_air      5290 non-null   float64
 6   primary_cleaner.state.floatbank8_a_level    5290 non-null   float64
 7   primary_cleaner.state.floatbank8_b_air      5290 non-null   float64
 8   primary_cleaner.state.floatbank8_b_level    5290 non-null   float64
 9   primary_cleaner.state.floatbank8_c_air      5290 non-null   float64
 10  primary_cleaner.state.floatbank8_c_level    5290 non-null   float64
 11  primary_cleaner.state.floatbank8_d_air      5290 non-null   float64
 12  primary_cleaner.state.floatbank8_d_level    5290 non-null   float64
 13  rougher.input.feed_ag                       5290 non-null   float64
 14  rougher.input.feed_pb                       5290 non-null   float64
 15  rougher.input.feed_rate                     5287 non-null   float64
 16  rougher.input.feed_size                     5289 non-null   float64
 17  rougher.input.feed_sol                      5269 non-null   float64
 18  rougher.input.feed_au                       5290 non-null   float64
 19  rougher.input.floatbank10_sulfate           5285 non-null   float64
 20  rougher.input.floatbank10_xanthate          5290 non-null   float64
 21  rougher.input.floatbank11_sulfate           5282 non-null   float64
 22  rougher.input.floatbank11_xanthate          5265 non-null   float64
 23  rougher.state.floatbank10_a_air             5290 non-null   float64
 24  rougher.state.floatbank10_a_level           5290 non-null   float64
 25  rougher.state.floatbank10_b_air             5290 non-null   float64
 26  rougher.state.floatbank10_b_level           5290 non-null   float64
 27  rougher.state.floatbank10_c_air             5290 non-null   float64
 28  rougher.state.floatbank10_c_level           5290 non-null   float64
 29  rougher.state.floatbank10_d_air             5290 non-null   float64
 30  rougher.state.floatbank10_d_level           5290 non-null   float64
 31  rougher.state.floatbank10_e_air             5290 non-null   float64
 32  rougher.state.floatbank10_e_level           5290 non-null   float64
 33  rougher.state.floatbank10_f_air             5290 non-null   float64
 34  rougher.state.floatbank10_f_level           5290 non-null   float64
 35  secondary_cleaner.state.floatbank2_a_air    5287 non-null   float64
 36  secondary_cleaner.state.floatbank2_a_level  5290 non-null   float64
 37  secondary_cleaner.state.floatbank2_b_air    5288 non-null   float64
 38  secondary_cleaner.state.floatbank2_b_level  5290 non-null   float64
 39  secondary_cleaner.state.floatbank3_a_air    5281 non-null   float64
 40  secondary_cleaner.state.floatbank3_a_level  5290 non-null   float64
 41  secondary_cleaner.state.floatbank3_b_air    5290 non-null   float64
 42  secondary_cleaner.state.floatbank3_b_level  5290 non-null   float64
 43  secondary_cleaner.state.floatbank4_a_air    5290 non-null   float64
 44  secondary_cleaner.state.floatbank4_a_level  5290 non-null   float64
 45  secondary_cleaner.state.floatbank4_b_air    5290 non-null   float64
 46  secondary_cleaner.state.floatbank4_b_level  5290 non-null   float64
 47  secondary_cleaner.state.floatbank5_a_air    5290 non-null   float64
 48  secondary_cleaner.state.floatbank5_a_level  5290 non-null   float64
 49  secondary_cleaner.state.floatbank5_b_air    5290 non-null   float64
 50  secondary_cleaner.state.floatbank5_b_level  5290 non-null   float64
 51  secondary_cleaner.state.floatbank6_a_air    5290 non-null   float64
 52  secondary_cleaner.state.floatbank6_a_level  5290 non-null   float64
dtypes: float64(52), object(1)
memory usage: 2.1+ MB
```

None

	primary_cleaner.input.sulfate	primary_cleaner.input.depressant	primary_cleaner.inpu
count	5286.000000	5285.000000	5
mean	174.839652	8.683596	
std	43.027080	3.072050	
min	2.566156	0.003839	
25%	147.121401	6.489555	
50%	177.828489	8.052207	
75%	208.125438	10.027764	
max	265.983123	40.000000	

8 rows × 52 columns

Данные представлены в 53 столбцах и 5290 строках, дубликаты отсутствуют. Есть пропуски в данных

Посмотрим информацию по исходным данным

In [6]
```
df_info(data_full)
```

	date	final.output.concentrate_ag	final.output.concentrate_pb	final.output.concentra
0	2016-01-15 00:00:00	6.055403	9.889648	5.5
1	2016-01-15 01:00:00	6.029369	9.968944	5.2
2	2016-01-15 02:00:00	6.055926	10.213995	5.3
3	2016-01-15 03:00:00	6.047977	9.977019	4.8
4	2016-01-15 04:00:00	6.148599	10.142511	4.9

5 rows × 87 columns

```
<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   final.output.concentrate_ag                         19438 non-null  float64
 2   final.output.concentrate_pb                         19438 non-null  float64
 3   final.output.concentrate_sol                        19228 non-null  float64
 4   final.output.concentrate_au                         19439 non-null  float64
 5   final.output.recovery                               19439 non-null  float64
 6   final.output.tail_ag                                19438 non-null  float64
 7   final.output.tail_pb                                19338 non-null  float64
 8   final.output.tail_sol                               19433 non-null  float64
 9   final.output.tail_au                                19439 non-null  float64
 10  primary_cleaner.input.sulfate                       19415 non-null  float64
 11  primary_cleaner.input.depressant                    19402 non-null  float64
 12  primary_cleaner.input.feed_size                     19439 non-null  float64
 13  primary_cleaner.input.xanthate                      19335 non-null  float64
 14  primary_cleaner.output.concentrate_ag               19439 non-null  float64
 15  primary_cleaner.output.concentrate_pb               19323 non-null  float64
 16  primary_cleaner.output.concentrate_sol              19069 non-null  float64
 17  primary_cleaner.output.concentrate_au               19439 non-null  float64
 18  primary_cleaner.output.tail_ag                      19435 non-null  float64
 19  primary_cleaner.output.tail_pb                      19418 non-null  float64
 20  primary_cleaner.output.tail_sol                     19377 non-null  float64
 21  primary_cleaner.output.tail_au                      19439 non-null  float64
 22  primary_cleaner.state.floatbank8_a_air              19435 non-null  float64
 23  primary_cleaner.state.floatbank8_a_level            19438 non-null  float64
 24  primary_cleaner.state.floatbank8_b_air              19435 non-null  float64
 25  primary_cleaner.state.floatbank8_b_level            19438 non-null  float64
 26  primary_cleaner.state.floatbank8_c_air              19437 non-null  float64
 27  primary_cleaner.state.floatbank8_c_level            19438 non-null  float64
 28  primary_cleaner.state.floatbank8_d_air              19436 non-null  float64
 29  primary_cleaner.state.floatbank8_d_level            19438 non-null  float64
 30  rougher.calculation.sulfate_to_au_concentrate       19437 non-null  float64
 31  rougher.calculation.floatbank10_sulfate_to_au_feed  19437 non-null  float64
 32  rougher.calculation.floatbank11_sulfate_to_au_feed  19437 non-null  float64
 33  rougher.calculation.au_pb_ratio                     19439 non-null  float64
 34  rougher.input.feed_ag                               19439 non-null  float64
 35  rougher.input.feed_pb                               19339 non-null  float64
 36  rougher.input.feed_rate                             19428 non-null  float64
 37  rougher.input.feed_size                             19294 non-null  float64
 38  rougher.input.feed_sol                              19340 non-null  float64
 39  rougher.input.feed_au                               19439 non-null  float64
 40  rougher.input.floatbank10_sulfate                   19405 non-null  float64
 41  rougher.input.floatbank10_xanthate                  19431 non-null  float64
 42  rougher.input.floatbank11_sulfate                   19395 non-null  float64
 43  rougher.input.floatbank11_xanthate                  18986 non-null  float64
 44  rougher.output.concentrate_ag                       19439 non-null  float64
 45  rougher.output.concentrate_pb                       19439 non-null  float64
 46  rougher.output.concentrate_sol                      19416 non-null  float64
 47  rougher.output.concentrate_au                       19439 non-null  float64
 48  rougher.output.recovery                             19439 non-null  float64
 49  rougher.output.tail_ag                              19438 non-null  float64
 50  rougher.output.tail_pb                              19439 non-null  float64
 51  rougher.output.tail_sol                             19439 non-null  float64
 52  rougher.output.tail_au                              19439 non-null  float64
 53  rougher.state.floatbank10_a_air                     19438 non-null  float64
 54  rougher.state.floatbank10_a_level                   19438 non-null  float64
```

```
 55  rougher.state.floatbank10_b_air                     19438 non-null  float64
 56  rougher.state.floatbank10_b_level                   19438 non-null  float64
 57  rougher.state.floatbank10_c_air                     19438 non-null  float64
 58  rougher.state.floatbank10_c_level                   19438 non-null  float64
 59  rougher.state.floatbank10_d_air                     19439 non-null  float64
 60  rougher.state.floatbank10_d_level                   19439 non-null  float64
 61  rougher.state.floatbank10_e_air                     19003 non-null  float64
 62  rougher.state.floatbank10_e_level                   19439 non-null  float64
 63  rougher.state.floatbank10_f_air                     19439 non-null  float64
 64  rougher.state.floatbank10_f_level                   19439 non-null  float64
 65  secondary_cleaner.output.tail_ag                    19437 non-null  float64
 66  secondary_cleaner.output.tail_pb                    19427 non-null  float64
 67  secondary_cleaner.output.tail_sol                   17691 non-null  float64
 68  secondary_cleaner.output.tail_au                    19439 non-null  float64
 69  secondary_cleaner.state.floatbank2_a_air            19219 non-null  float64
 70  secondary_cleaner.state.floatbank2_a_level          19438 non-null  float64
 71  secondary_cleaner.state.floatbank2_b_air            19416 non-null  float64
 72  secondary_cleaner.state.floatbank2_b_level          19438 non-null  float64
 73  secondary_cleaner.state.floatbank3_a_air            19426 non-null  float64
 74  secondary_cleaner.state.floatbank3_a_level          19438 non-null  float64
 75  secondary_cleaner.state.floatbank3_b_air            19438 non-null  float64
 76  secondary_cleaner.state.floatbank3_b_level          19438 non-null  float64
 77  secondary_cleaner.state.floatbank4_a_air            19433 non-null  float64
 78  secondary_cleaner.state.floatbank4_a_level          19438 non-null  float64
 79  secondary_cleaner.state.floatbank4_b_air            19438 non-null  float64
 80  secondary_cleaner.state.floatbank4_b_level          19438 non-null  float64
 81  secondary_cleaner.state.floatbank5_a_air            19438 non-null  float64
 82  secondary_cleaner.state.floatbank5_a_level          19438 non-null  float64
 83  secondary_cleaner.state.floatbank5_b_air            19438 non-null  float64
 84  secondary_cleaner.state.floatbank5_b_level          19438 non-null  float64
 85  secondary_cleaner.state.floatbank6_a_air            19437 non-null  float64
 86  secondary_cleaner.state.floatbank6_a_level          19438 non-null  float64
dtypes: float64(86), object(1)
memory usage: 12.9+ MB
None
```

	final.output.concentrate_ag	final.output.concentrate_pb	final.output.concentrate_so
count	19438.000000	19438.000000	19228.000000
mean	5.168470	9.978895	9.501224
std	1.372348	1.669240	2.787537
min	0.000000	0.000000	0.000000
25%	4.251240	9.137262	7.722820
50%	5.066094	10.102433	9.218961
75%	5.895527	11.035769	10.947813
max	16.001945	17.031899	19.615720

8 rows × 86 columns

Данные представлены в 87 столбцах и 19439 строках, дубликаты отсутствуют. Есть пропуски в данных

Размеры обучающей и тестовой выборок различаются (в части признаков), в тестовой выбороке нет целевых признаков. Есть пропуски в данных
Требуется предобработка данных.

Рассчет эффективности обогащения

По условиям зададачи необходимо проверьть правильность расчета эффективности обогащения. Для этого вычислим её на обучающей выборке для признака `rougher.output.recovery`. Найдем `MAE` между расчётным и фактическим значением признака

Эффективность обогащения рассчитывается по формуле
$ Recovery = (C*(F-T))/(F*(C-T)) * 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 [7]
```
# Эффективность обогащения
def calc_enrichment(C, F, T):
    recovery = (C*(F-T))/(F*(C-T)) * 100
    return recovery
```

In [8]
```
recovery = calc_enrichment(
    data_train['rougher.output.concentrate_au'],
    data_train['rougher.input.feed_au'],
    data_train['rougher.output.tail_au'],
)

# Найдем MAE между расчётным и фактическим значением признака
mean_absolute_error(data_train['rougher.output.recovery'], recovery)
```

Out[8]
```
9.73512347450521e-15
```

Средняя абсолютная ошибка между рассчитанной по формуле эффективностью обогащения и признаком из датасета близко к нулю, а значит эффективность обогащения в датасете рассчитана правильно.

Анализ признаков

Как выявили ранее, размеры обучающей и тестовой выборок в части признаков различаются.
Проанализируем признаки, недоступные в тестовой выборке

In [9]
```
feature_diff = data_train.columns.difference(data_test.columns)
feature_diff
```

Out[9]
```
Index(['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'],
      dtype='object')
```

По описанию технологического процесса, признаки именуются следующим образом:
`[этап].[тип_параметра].[название_параметра]`

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

Таким образом признаки отсутствующие в тестовой выборке характеризуют:

- финальные характеристики продукта
- характеристики продукта при первичной очистке
- расчетные характеристики флотации
- характеристики продукта при флотации
- характеристики продукта при вторичной очистке

Наряду с этим в тестовой выборке отсутствуют целевые признаки `final.output.recovery` и `rougher.output.recovery`

Предобработка данных

Пропуски присутствуют во всех трех выборках

В тестовой выборке заполнить пропуски средним значением или каким-то другим мы не можем - это исказит результаты тестирования модели. Поэтому принимаем решение - удалить строки с пропущенными значениями.

In [10]
```
data_test = data_test.dropna()
print('Количество пропусков', data_test.isna().sum().sum())
```

```
Количество пропусков 0
```

В обучающей выборке пропуски заполним методом interpolate() (расчитывает значение между предыдущей и следующей строкой), т.к. по условиям техпроцесса - соседние по времени параметры часто похожи и с целью оставить как можно больше данных для обучения модели

In [11]
```
data_train = data_train.interpolate()
print('Количество пропусков', data_train.isna().sum().sum())
```

```
Количество пропусков 0
```

Аналогичным образом поступим и с исходными данными

In [12]
```
data_full = data_full.interpolate()
print('Количество пропусков', data_full.isna().sum().sum())
```

```
Количество пропусков 0
```

В ходе данного шага:

- провели обзор данных;
- проверили правильность расчета эффективности обогащения;
- выявили признаки недоступные в тестовой выборке;
- обработали пропуски в данных.

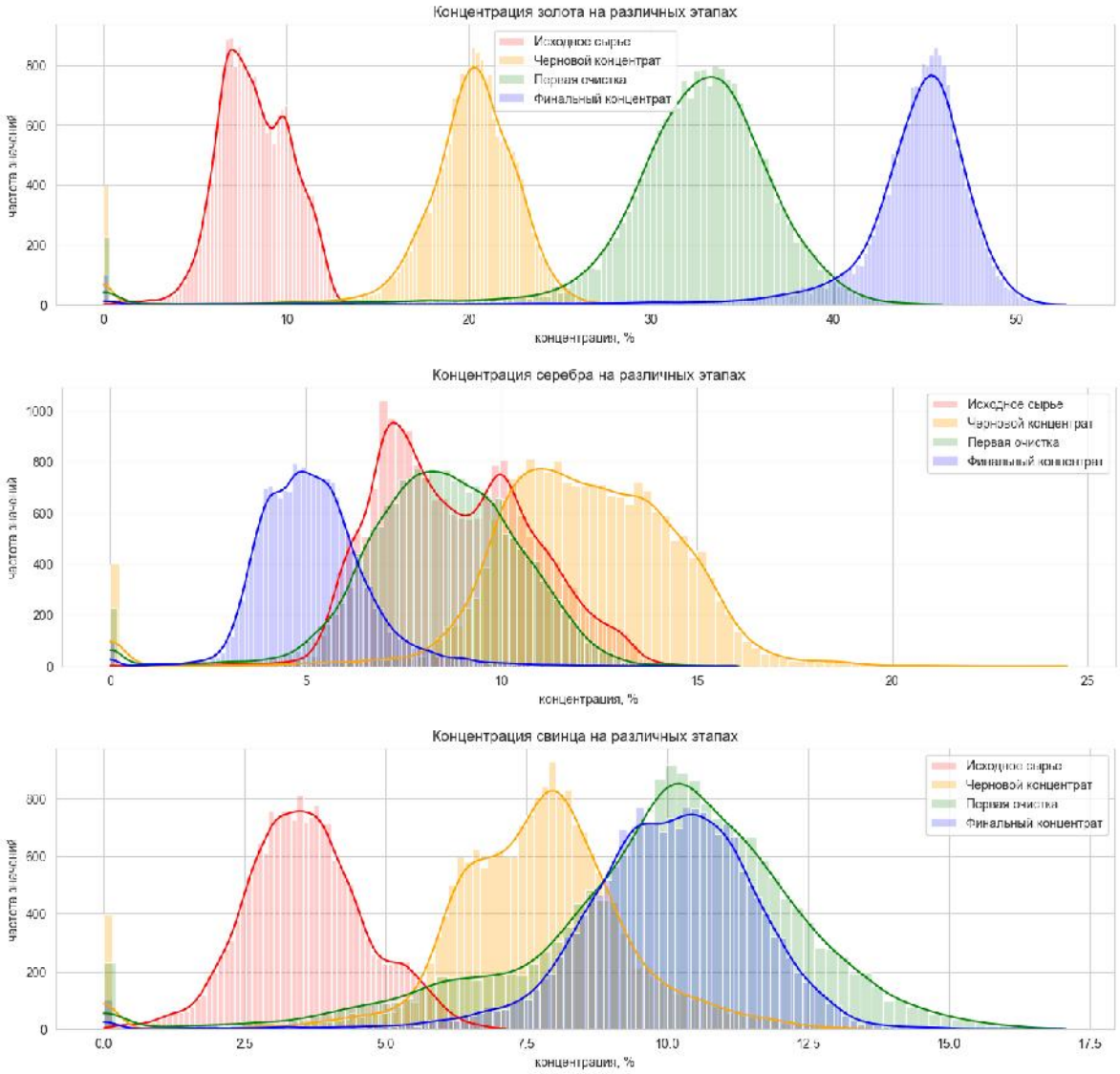
Анализ данных

Посмотрим, как меняется концентрация металлов (Au, Ag, Pb) на различных этапах

In [13]
```
for metall, metall_name in zip(['au','ag','pb'], ['золота', 'серебра', 'свинца']
    plt.figure(figsize=(15,4))
    sns.set_style('whitegrid')
    sns.histplot(data_full['rougher.input.feed_' + metall], color="red", kde=Tru
    sns.histplot(data_full['rougher.output.concentrate_' + metall], color="orang
    sns.histplot(data_full['primary_cleaner.output.concentrate_' + metall], colo
    sns.histplot(data_full['final.output.concentrate_' + metall], color='blue',

    plt.legend()
    # plt.grid()
    plt.title(f'Концентрация {metall_name} на различных этапах')
```

```
    plt.xlabel('концентрация, %')
    plt.ylabel('частота значений')
    plt.show()
```



Что можно понять из графиков:

- концентрация золота повышается с каждым этапом. Цель флотации и очисток увеличить концентрацию золота;
- концентрация серебра понижается;
- концентрация свинца повышается. Это связано с химическим процесами очищения золота.

Так же на графиках отчетливо видны нулевые значения концентраций. Это явная ошибка. Скорее всего это связано со сбоем измерительного оборудования. В дальнейшем уберем эти значения из обучающих данных

Сравнение распределения размеров гранул

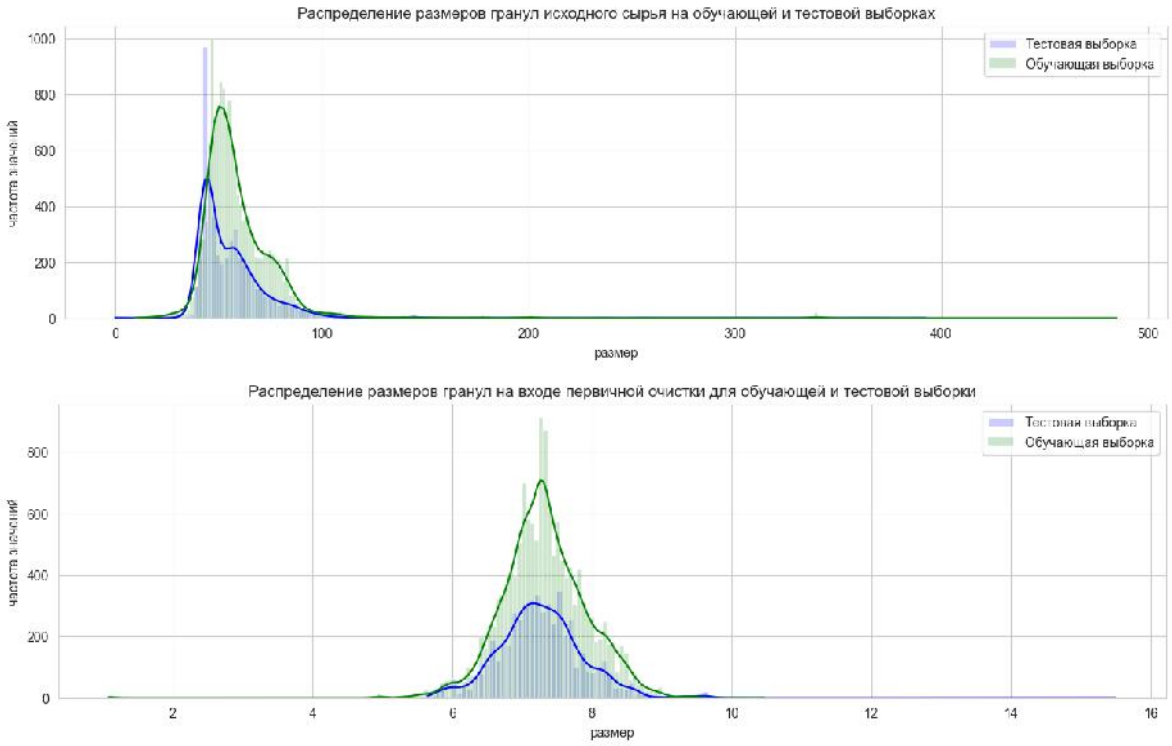
Сравним распределение размеров гранул исходного сырья и на входе при первичной очистке для обучающей и тестовой выборки

In [14]:
```
# Распределение размеров гранул исходного сырья
plt.figure(figsize=(15,4))
sns.histplot(data_test['rougher.input.feed_size'], color="blue", kde=True, label
sns.histplot(data_train['rougher.input.feed_size'], color="green", kde=True, lab

plt.legend()
# plt.grid()
plt.title('Распределение размеров гранул исходного сырья на обучающей и тестовой
plt.xlabel('размер')
plt.ylabel('частота значений')
plt.show()

# Распределение размеров гранул на входе первичной очистки
plt.figure(figsize=(15,4))
sns.histplot(data_test['primary_cleaner.input.feed_size'], color="blue", kde=Tru
sns.histplot(data_train['primary_cleaner.input.feed_size'], color="green", kde=T

plt.legend()
# plt.grid()
plt.title('Распределение размеров гранул на входе первичной очистки для обучающе
plt.xlabel('размер')
plt.ylabel('частота значений')
plt.show()
```



Видно, что распределения похожи, а значит оценка модели будет правильной.

Исследуем суммарную концентрацию металлов на разных стадиях

In [15]:
```
plt.figure(figsize=(15,4))
sns.set_style('whitegrid')
for stage, stage_name, color in zip(['rougher.input.feed_','rougher.output.conce
                                     ['Исходное сырье', 'Черновой концентрат', 'Первая о
                                     ['b', 'y', 'g', 'r']):
    sns.histplot(data_full[[stage+'au', stage+'ag',stage+'pb']].sum(axis=1),colo
```

```
plt.title('Суммарная концентрация металлов на разных стадиях')
plt.xlabel('Концентрация, %')
plt.ylabel('частота значений')
plt.legend()
plt.show()
```

Отчетливо видно, что концентрация металлов повышается с каждым этапом.
Так же просматриваются нулевые значения концентраций.

В ходе данного шага:

- оценили изменения концентрации металлов (Au, Ag, Pb) на различных этапах
- сравнили распределение размеров гранул исходного сырья и на входе при первичной очистке для обучающей и тестовой выборки
- исследовали суммарную концентрацию металлов на разных стадиях

Модель

Подготовка данных

Уберем нулевые значения из обучающих данных, выявленные на предыдущем этапе

In [16]
```
for col in ['rougher.input.feed_','rougher.output.concentrate_','primary_cleaner
    data_train = data_train[data_train[col + 'au']>0]
    data_train = data_train[data_train[col + 'ag']>0]
    data_train = data_train[data_train[col + 'pb']>0]
```

Добавим в тестовый набор целевые признаки

In [17]
```
# Сделаем индексом датафреймов поле date
data_full.set_index(data_full['date'], drop=True, inplace=True)
data_train.set_index(data_train['date'], drop=True, inplace=True)
data_test.set_index(data_test['date'], drop=True, inplace=True)

data_train = data_train.drop('date', axis=1)
data_test = data_test.drop('date', axis=1)

#Добавим в тестовый набор целевые признаки
data_test = data_test.join(data_full.loc[data_test.index][['final.output.recover
data_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 5229 entries, 2016-09-01 00:59:59 to 2017-12-31 23:59:59
Data columns (total 54 columns):
 #   Column                                      Non-Null Count  Dtype  
---  ------                                      --------------  -----  
 0   primary_cleaner.input.sulfate               5229 non-null   float64
 1   primary_cleaner.input.depressant            5229 non-null   float64
 2   primary_cleaner.input.feed_size             5229 non-null   float64
 3   primary_cleaner.input.xanthate              5229 non-null   float64
 4   primary_cleaner.state.floatbank8_a_air      5229 non-null   float64
 5   primary_cleaner.state.floatbank8_a_level    5229 non-null   float64
 6   primary_cleaner.state.floatbank8_b_air      5229 non-null   float64
 7   primary_cleaner.state.floatbank8_b_level    5229 non-null   float64
 8   primary_cleaner.state.floatbank8_c_air      5229 non-null   float64
 9   primary_cleaner.state.floatbank8_c_level    5229 non-null   float64
 10  primary_cleaner.state.floatbank8_d_air      5229 non-null   float64
 11  primary_cleaner.state.floatbank8_d_level    5229 non-null   float64
 12  rougher.input.feed_ag                       5229 non-null   float64
 13  rougher.input.feed_pb                       5229 non-null   float64
 14  rougher.input.feed_rate                     5229 non-null   float64
 15  rougher.input.feed_size                     5229 non-null   float64
 16  rougher.input.feed_sol                      5229 non-null   float64
 17  rougher.input.feed_au                       5229 non-null   float64
 18  rougher.input.floatbank10_sulfate           5229 non-null   float64
 19  rougher.input.floatbank10_xanthate          5229 non-null   float64
 20  rougher.input.floatbank11_sulfate           5229 non-null   float64
 21  rougher.input.floatbank11_xanthate          5229 non-null   float64
 22  rougher.state.floatbank10_a_air             5229 non-null   float64
 23  rougher.state.floatbank10_a_level           5229 non-null   float64
 24  rougher.state.floatbank10_b_air             5229 non-null   float64
 25  rougher.state.floatbank10_b_level           5229 non-null   float64
 26  rougher.state.floatbank10_c_air             5229 non-null   float64
 27  rougher.state.floatbank10_c_level           5229 non-null   float64
 28  rougher.state.floatbank10_d_air             5229 non-null   float64
 29  rougher.state.floatbank10_d_level           5229 non-null   float64
 30  rougher.state.floatbank10_e_air             5229 non-null   float64
 31  rougher.state.floatbank10_e_level           5229 non-null   float64
 32  rougher.state.floatbank10_f_air             5229 non-null   float64
 33  rougher.state.floatbank10_f_level           5229 non-null   float64
 34  secondary_cleaner.state.floatbank2_a_air    5229 non-null   float64
 35  secondary_cleaner.state.floatbank2_a_level  5229 non-null   float64
 36  secondary_cleaner.state.floatbank2_b_air    5229 non-null   float64
 37  secondary_cleaner.state.floatbank2_b_level  5229 non-null   float64
 38  secondary_cleaner.state.floatbank3_a_air    5229 non-null   float64
 39  secondary_cleaner.state.floatbank3_a_level  5229 non-null   float64
 40  secondary_cleaner.state.floatbank3_b_air    5229 non-null   float64
 41  secondary_cleaner.state.floatbank3_b_level  5229 non-null   float64
 42  secondary_cleaner.state.floatbank4_a_air    5229 non-null   float64
 43  secondary_cleaner.state.floatbank4_a_level  5229 non-null   float64
 44  secondary_cleaner.state.floatbank4_b_air    5229 non-null   float64
 45  secondary_cleaner.state.floatbank4_b_level  5229 non-null   float64
 46  secondary_cleaner.state.floatbank5_a_air    5229 non-null   float64
 47  secondary_cleaner.state.floatbank5_a_level  5229 non-null   float64
 48  secondary_cleaner.state.floatbank5_b_air    5229 non-null   float64
 49  secondary_cleaner.state.floatbank5_b_level  5229 non-null   float64
 50  secondary_cleaner.state.floatbank6_a_air    5229 non-null   float64
 51  secondary_cleaner.state.floatbank6_a_level  5229 non-null   float64
 52  final.output.recovery                       5229 non-null   float64
 53  rougher.output.recovery                     5229 non-null   float64
```

```
dtypes: float64(54)
memory usage: 2.3+ MB
```

Так как размеры обучающей и тестовой выборок различаются (в части признаков) приведем обучающую выборку к одному размеру с тестовой

In [18]
```
data_train = data_train[data_test.columns]
```

Подготовим признаки

In [19]
```
target_signs = ['rougher.output.recovery', 'final.output.recovery']

# Подготовим признаки для обучения
target_train_rougher = data_train[target_signs[0]]
target_train_final = data_train[target_signs[1]]
features_train = data_train.drop(target_signs, axis=1)

# Подготовим признаки для теста
target_test_rougher = data_test[target_signs[0]]
target_test_final = data_test[target_signs[1]]
features_test = data_test.drop(target_signs, axis=1)

print("Размерность тренеровочных данных", features_train.shape)
print("Размерность тестовых данных", features_test.shape)
```

```
Размерность тренеровочных данных (13725, 52)
Размерность тестовых данных (5229, 52)
```

Построение модели и её оценка

In [20]
```
# функция для расчета sMAPE
def smape(target, predictions):
    smape = 100/len(target) * sum(2*abs(target - predictions) / (abs(target)+abs
    return smape
# функция для расчета итогового sMAPE
def final_smape(r_smape, f_smape):
    return abs(0.25*r_smape + 0.75*f_smape)
```

In [21]
```
# Создадим метрику для поиска оптимального решения
scorer = make_scorer(smape, greater_is_better=False)
```

In [22]
```
# используем кросс-валидацию
cv = KFold(n_splits=3, shuffle=False)
```

Модель на основе Логистической регрессии

In [23]
```
# Создадим модель, объявим переменные
model_lgr = LinearRegression()
lgr_rougher = cross_val_score(model_lgr, features_train, target_train_rougher,
                              scoring=scorer, cv=cv, n_jobs=-1).mean()
lgr_final = cross_val_score(model_lgr, features_train, target_train_final,
                              scoring=scorer, cv=cv, n_jobs=-1).mean()
print('Итоговое sMAPE', final_smape(lgr_rougher, lgr_final))
```

```
Итоговое sMAPE 12.649360186573364
```

Модель на основе Случайного леса

In [24]:
```
# Создадим модель, объявим переменные
model_rfc = RandomForestRegressor(random_state=12345)

param_grid_rfc = {
    'n_estimators': range(10, 510, 50),
    'max_depth' : [None] + [i for i in range(2, 11)]
}

# Обучение
cv_rfc = GridSearchCV(estimator=model_rfc,
                      param_grid=param_grid_rfc,
                      cv=cv,
                      n_jobs=-1,
                      scoring=scorer,
                      verbose=10
                     )

cv_rfc_rougher = cv_rfc.fit(features_train, target_train_rougher)
cv_rfc_final = cv_rfc.fit(features_train, target_train_final)

print('Итоговое sMAPE', final_smape(cv_rfc_rougher.best_score_, cv_rfc_final.bes
```

```
Fitting 3 folds for each of 100 candidates, totalling 300 fits
Fitting 3 folds for each of 100 candidates, totalling 300 fits
Итоговое sMAPE 10.144783143487258
```

Таким образом наиболее лучшей моделью является Модель на основе Случайного леса, итоговое sMAPE 10.14

Проверка модели на тестовой выборке

In [25]:
```
predict_test_rougher = cv_rfc_rougher.best_estimator_.predict(features_test)
predict_test_final = cv_rfc_final.best_estimator_.predict(features_test)

test_smape_r = smape(target_test_rougher, predict_test_rougher)
test_smape_f = smape(target_test_final, predict_test_final)

print('Итоговое sMAPE', final_smape(test_smape_r, test_smape_f))
```

```
Итоговое sMAPE 13.135973875472796
```

Проверка модели на адекватность

In [26]:
```
dummy_model = DummyRegressor(strategy='mean')

dummy_rougher = dummy_model.fit(features_train, target_train_rougher)
dummy_final = dummy_model.fit(features_train, target_train_final)

dummy_predict_test_rougher = dummy_rougher.predict(features_test)
dummy_predict_test_final = dummy_final.predict(features_test)

dummy_smape_r = smape(target_test_rougher, dummy_predict_test_rougher)
dummy_smape_f = smape(target_test_final, dummy_predict_test_final)

print('Итоговое sMAPE', final_smape(dummy_smape_r, dummy_smape_f))
```

```
Итоговое sMAPE 14.503509689970741
```

Так как `Итоговое sMAPE` у модели ниже чем у `DummyRegressor`, то модель можно признать адекватной

Вывод

В ходе проделанной работы было выполнено:

В рамках подготовки данных:

- проверена формула вычисления эффективности обогащения;
- проанализированы признаки, недоступные в тестовой выборке;
- проведена предобработка данных.

В рамках анализа данных:

- исследовано изменение концентрации элементов на каждом этапе;
- проанализированы распределения размеров гранул на обучающей и тестовой выборках;
- исследованы суммарные концентрации.

Написана функция для вычисления итогового sMAPE. Построено и обучено нескольео моделей прогнозирования. Выбрана лучшая модель, её качество проверено на тестовой выборке и прошло проверку DummyRegressor.