cbbe5251-8f67-41dd-8239-958cc0a93a3b-Copy4

December 13, 2024

Introduction

This project focuses on analyzing and modeling data from the gold recovery industry, with the ultimate goal of predicting recovery efficiency at various stages of the process. The data for this project is provided in three separate files:

0.0.1 gold_recovery_train.csv: The training dataset.

0.0.2 gold_recovery_test.csv: The test dataset without target values.

0.0.3 gold_recovery_full.csv: The complete source dataset containing both the training and test data along with all features.

The datasets are indexed by the date and time of data acquisition, reflecting the temporal nature of the parameters. Parameters measured or calculated at similar times often display correlation. However, due to operational constraints, some features measured or calculated later are missing from the test dataset, while the source dataset contains all features.

Objectives

Before building the predictive model, the project involves several critical tasks to ensure the reliability of the data and the resulting predictions:

Data Verification: Validate the correctness of the data using provided guidelines.

Data Preprocessing: Handle missing values, anomalies, and feature alignment between training and test datasets.

Exploratory Data Analysis (EDA): Examine key trends in metal concentration, particle size, and recovery efficiency.

Model Development: Train a machine learning model to predict recovery efficiency at the "rougher" and "final" stages.

Model Evaluation: Use metrics such as Symmetric Mean Absolute Percentage Error (sMAPE) to assess the model's performance.

This structured approach ensures that the model is built on accurate, clean data and aligns with industry-specific requirements, allowing it to provide valuable insights for optimizing the gold recovery process.

[1]:
```
import pandas as pd
import numpy as np
from sklearn.metrics import mean_absolute_error
```

```python
# Load datasets
train_data = pd.read_csv('/datasets/gold_recovery_train.csv', index_col='date', parse_dates=True)
test_data = pd.read_csv('/datasets/gold_recovery_test.csv', index_col='date', parse_dates=True)
full_data = pd.read_csv('/datasets/gold_recovery_full.csv', index_col='date', parse_dates=True)

# Quick inspection of the datasets
print("Train Data:")
print(train_data.info())
print(train_data.head())

print("\nTest Data:")
print(test_data.info())
print(test_data.head())

print("\nFull Data:")
print(full_data.info())
print(full_data.head())

# Display the first few rows of the training data to inspect columns and data types
print(train_data.head())
```

```
Train Data:
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 16860 entries, 2016-01-15 00:00:00 to 2018-08-18 10:59:59
Data columns (total 86 columns):
 #   Column                                              Non-Null Count  Dtype
---  ------                                              --------------  -----
 0   final.output.concentrate_ag                         16788 non-null  float64
 1   final.output.concentrate_pb                         16788 non-null  float64
 2   final.output.concentrate_sol                        16490 non-null  float64
 3   final.output.concentrate_au                         16789 non-null  float64
 4   final.output.recovery                               15339 non-null  float64
 5   final.output.tail_ag                                16794 non-null  float64
 6   final.output.tail_pb                                16677 non-null  float64
 7   final.output.tail_sol                               16715 non-null  float64
 8   final.output.tail_au                                16794 non-null  float64
 9   primary_cleaner.input.sulfate                       15553 non-null  float64
 10  primary_cleaner.input.depressant                    15598 non-null  float64
 11  primary_cleaner.input.feed_size                     16860 non-null  float64
 12  primary_cleaner.input.xanthate                      15875 non-null  float64
 13  primary_cleaner.output.concentrate_ag               16778 non-null  float64
 14  primary_cleaner.output.concentrate_pb               16502 non-null  float64
```

```
 15  primary_cleaner.output.concentrate_sol               16224 non-null  float64
 16  primary_cleaner.output.concentrate_au                16778 non-null  float64
 17  primary_cleaner.output.tail_ag                       16777 non-null  float64
 18  primary_cleaner.output.tail_pb                       16761 non-null  float64
 19  primary_cleaner.output.tail_sol                      16579 non-null  float64
 20  primary_cleaner.output.tail_au                       16777 non-null  float64
 21  primary_cleaner.state.floatbank8_a_air               16820 non-null  float64
 22  primary_cleaner.state.floatbank8_a_level             16827 non-null  float64
 23  primary_cleaner.state.floatbank8_b_air               16820 non-null  float64
 24  primary_cleaner.state.floatbank8_b_level             16833 non-null  float64
 25  primary_cleaner.state.floatbank8_c_air               16822 non-null  float64
 26  primary_cleaner.state.floatbank8_c_level             16833 non-null  float64
 27  primary_cleaner.state.floatbank8_d_air               16821 non-null  float64
 28  primary_cleaner.state.floatbank8_d_level             16833 non-null  float64
 29  rougher.calculation.sulfate_to_au_concentrate        16833 non-null  float64
 30  rougher.calculation.floatbank10_sulfate_to_au_feed  16833 non-null  float64
 31  rougher.calculation.floatbank11_sulfate_to_au_feed  16833 non-null  float64
 32  rougher.calculation.au_pb_ratio                      15618 non-null  float64
 33  rougher.input.feed_ag                                16778 non-null  float64
 34  rougher.input.feed_pb                                16632 non-null  float64
 35  rougher.input.feed_rate                              16347 non-null  float64
 36  rougher.input.feed_size                              16443 non-null  float64
 37  rougher.input.feed_sol                               16568 non-null  float64
 38  rougher.input.feed_au                                16777 non-null  float64
 39  rougher.input.floatbank10_sulfate                    15816 non-null  float64
 40  rougher.input.floatbank10_xanthate                   16514 non-null  float64
 41  rougher.input.floatbank11_sulfate                    16237 non-null  float64
 42  rougher.input.floatbank11_xanthate                   14956 non-null  float64
 43  rougher.output.concentrate_ag                        16778 non-null  float64
 44  rougher.output.concentrate_pb                        16778 non-null  float64
 45  rougher.output.concentrate_sol                       16698 non-null  float64
 46  rougher.output.concentrate_au                        16778 non-null  float64
 47  rougher.output.recovery                              14287 non-null  float64
 48  rougher.output.tail_ag                               14610 non-null  float64
 49  rougher.output.tail_pb                               16778 non-null  float64
 50  rougher.output.tail_sol                              14611 non-null  float64
 51  rougher.output.tail_au                               14611 non-null  float64
 52  rougher.state.floatbank10_a_air                      16807 non-null  float64
 53  rougher.state.floatbank10_a_level                    16807 non-null  float64
 54  rougher.state.floatbank10_b_air                      16807 non-null  float64
 55  rougher.state.floatbank10_b_level                    16807 non-null  float64
 56  rougher.state.floatbank10_c_air                      16807 non-null  float64
 57  rougher.state.floatbank10_c_level                    16814 non-null  float64
 58  rougher.state.floatbank10_d_air                      16802 non-null  float64
 59  rougher.state.floatbank10_d_level                    16809 non-null  float64
 60  rougher.state.floatbank10_e_air                      16257 non-null  float64
 61  rougher.state.floatbank10_e_level                    16809 non-null  float64
 62  rougher.state.floatbank10_f_air                      16802 non-null  float64
```

```
 63  rougher.state.floatbank10_f_level                   16802 non-null  float64
 64  secondary_cleaner.output.tail_ag                    16776 non-null  float64
 65  secondary_cleaner.output.tail_pb                    16764 non-null  float64
 66  secondary_cleaner.output.tail_sol                   14874 non-null  float64
 67  secondary_cleaner.output.tail_au                    16778 non-null  float64
 68  secondary_cleaner.state.floatbank2_a_air            16497 non-null  float64
 69  secondary_cleaner.state.floatbank2_a_level          16751 non-null  float64
 70  secondary_cleaner.state.floatbank2_b_air            16705 non-null  float64
 71  secondary_cleaner.state.floatbank2_b_level          16748 non-null  float64
 72  secondary_cleaner.state.floatbank3_a_air            16763 non-null  float64
 73  secondary_cleaner.state.floatbank3_a_level          16747 non-null  float64
 74  secondary_cleaner.state.floatbank3_b_air            16752 non-null  float64
 75  secondary_cleaner.state.floatbank3_b_level          16750 non-null  float64
 76  secondary_cleaner.state.floatbank4_a_air            16731 non-null  float64
 77  secondary_cleaner.state.floatbank4_a_level          16747 non-null  float64
 78  secondary_cleaner.state.floatbank4_b_air            16768 non-null  float64
 79  secondary_cleaner.state.floatbank4_b_level          16767 non-null  float64
 80  secondary_cleaner.state.floatbank5_a_air            16775 non-null  float64
 81  secondary_cleaner.state.floatbank5_a_level          16775 non-null  float64
 82  secondary_cleaner.state.floatbank5_b_air            16775 non-null  float64
 83  secondary_cleaner.state.floatbank5_b_level          16776 non-null  float64
 84  secondary_cleaner.state.floatbank6_a_air            16757 non-null  float64
 85  secondary_cleaner.state.floatbank6_a_level          16775 non-null  float64
dtypes: float64(86)
memory usage: 11.2 MB
None
                     final.output.concentrate_ag  final.output.concentrate_pb  \
date
2016-01-15 00:00:00                     6.055403                     9.889648
2016-01-15 01:00:00                     6.029369                     9.968944
2016-01-15 02:00:00                     6.055926                    10.213995
2016-01-15 03:00:00                     6.047977                     9.977019
2016-01-15 04:00:00                     6.148599                    10.142511

                     final.output.concentrate_sol  \
date
2016-01-15 00:00:00                      5.507324
2016-01-15 01:00:00                      5.257781
2016-01-15 02:00:00                      5.383759
2016-01-15 03:00:00                      4.858634
2016-01-15 04:00:00                      4.939416

                     final.output.concentrate_au  final.output.recovery  \
date
2016-01-15 00:00:00                    42.192020              70.541216
2016-01-15 01:00:00                    42.701629              69.266198
2016-01-15 02:00:00                    42.657501              68.116445
2016-01-15 03:00:00                    42.689819              68.347543
```

```
2016-01-15 04:00:00                   42.774141             66.927016

                    final.output.tail_ag  final.output.tail_pb  \
date
2016-01-15 00:00:00             10.411962              0.895447
2016-01-15 01:00:00             10.462676              0.927452
2016-01-15 02:00:00             10.507046              0.953716
2016-01-15 03:00:00             10.422762              0.883763
2016-01-15 04:00:00             10.360302              0.792826

                    final.output.tail_sol  final.output.tail_au  \
date
2016-01-15 00:00:00              16.904297              2.143149
2016-01-15 01:00:00              16.634514              2.224930
2016-01-15 02:00:00              16.208849              2.257889
2016-01-15 03:00:00              16.532835              2.146849
2016-01-15 04:00:00              16.525686              2.055292

                    primary_cleaner.input.sulfate  …  \
date                                                …
2016-01-15 00:00:00                     127.092003  …
2016-01-15 01:00:00                     125.629232  …
2016-01-15 02:00:00                     123.819808  …
2016-01-15 03:00:00                     122.270188  …
2016-01-15 04:00:00                     117.988169  …

                    secondary_cleaner.state.floatbank4_a_air  \
date
2016-01-15 00:00:00                                 14.016835
2016-01-15 01:00:00                                 13.992281
2016-01-15 02:00:00                                 14.015015
2016-01-15 03:00:00                                 14.036510
2016-01-15 04:00:00                                 14.027298

                    secondary_cleaner.state.floatbank4_a_level  \
date
2016-01-15 00:00:00                                 -502.488007
2016-01-15 01:00:00                                 -505.503262
2016-01-15 02:00:00                                 -502.520901
2016-01-15 03:00:00                                 -500.857308
2016-01-15 04:00:00                                 -499.838632

                    secondary_cleaner.state.floatbank4_b_air  \
date
2016-01-15 00:00:00                                 12.099931
2016-01-15 01:00:00                                 11.950531
2016-01-15 02:00:00                                 11.912783
2016-01-15 03:00:00                                 11.999550
```

```
2016-01-15 04:00:00                                11.953070

                     secondary_cleaner.state.floatbank4_b_level  \
date
2016-01-15 00:00:00                               -504.715942
2016-01-15 01:00:00                               -501.331529
2016-01-15 02:00:00                               -501.133383
2016-01-15 03:00:00                               -501.193686
2016-01-15 04:00:00                               -501.053894

                     secondary_cleaner.state.floatbank5_a_air  \
date
2016-01-15 00:00:00                                 9.925633
2016-01-15 01:00:00                                10.039245
2016-01-15 02:00:00                                10.070913
2016-01-15 03:00:00                                 9.970366
2016-01-15 04:00:00                                 9.925709

                     secondary_cleaner.state.floatbank5_a_level  \
date
2016-01-15 00:00:00                               -498.310211
2016-01-15 01:00:00                               -500.169983
2016-01-15 02:00:00                               -500.129135
2016-01-15 03:00:00                               -499.201640
2016-01-15 04:00:00                               -501.686727

                     secondary_cleaner.state.floatbank5_b_air  \
date
2016-01-15 00:00:00                                 8.079666
2016-01-15 01:00:00                                 7.984757
2016-01-15 02:00:00                                 8.013877
2016-01-15 03:00:00                                 7.977324
2016-01-15 04:00:00                                 7.894242

                     secondary_cleaner.state.floatbank5_b_level  \
date
2016-01-15 00:00:00                               -500.470978
2016-01-15 01:00:00                               -500.582168
2016-01-15 02:00:00                               -500.517572
2016-01-15 03:00:00                               -500.255908
2016-01-15 04:00:00                               -500.356035

                     secondary_cleaner.state.floatbank6_a_air  \
date
2016-01-15 00:00:00                                14.151341
2016-01-15 01:00:00                                13.998353
2016-01-15 02:00:00                                14.028663
2016-01-15 03:00:00                                14.005551
```

```
2016-01-15 04:00:00                                 13.996647

                     secondary_cleaner.state.floatbank6_a_level
date
2016-01-15 00:00:00                                 -605.841980
2016-01-15 01:00:00                                 -599.787184
2016-01-15 02:00:00                                 -601.427363
2016-01-15 03:00:00                                 -599.996129
2016-01-15 04:00:00                                 -601.496691

[5 rows x 86 columns]

Test Data:
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 5856 entries, 2016-09-01 00:59:59 to 2017-12-31 23:59:59
Data columns (total 52 columns):
 #   Column                                      Non-Null Count  Dtype
---  ------                                      --------------  -----
 0   primary_cleaner.input.sulfate               5554 non-null   float64
 1   primary_cleaner.input.depressant            5572 non-null   float64
 2   primary_cleaner.input.feed_size             5856 non-null   float64
 3   primary_cleaner.input.xanthate              5690 non-null   float64
 4   primary_cleaner.state.floatbank8_a_air      5840 non-null   float64
 5   primary_cleaner.state.floatbank8_a_level    5840 non-null   float64
 6   primary_cleaner.state.floatbank8_b_air      5840 non-null   float64
 7   primary_cleaner.state.floatbank8_b_level    5840 non-null   float64
 8   primary_cleaner.state.floatbank8_c_air      5840 non-null   float64
 9   primary_cleaner.state.floatbank8_c_level    5840 non-null   float64
 10  primary_cleaner.state.floatbank8_d_air      5840 non-null   float64
 11  primary_cleaner.state.floatbank8_d_level    5840 non-null   float64
 12  rougher.input.feed_ag                       5840 non-null   float64
 13  rougher.input.feed_pb                       5840 non-null   float64
 14  rougher.input.feed_rate                     5816 non-null   float64
 15  rougher.input.feed_size                     5834 non-null   float64
 16  rougher.input.feed_sol                      5789 non-null   float64
 17  rougher.input.feed_au                       5840 non-null   float64
 18  rougher.input.floatbank10_sulfate           5599 non-null   float64
 19  rougher.input.floatbank10_xanthate          5733 non-null   float64
 20  rougher.input.floatbank11_sulfate           5801 non-null   float64
 21  rougher.input.floatbank11_xanthate          5503 non-null   float64
 22  rougher.state.floatbank10_a_air             5839 non-null   float64
 23  rougher.state.floatbank10_a_level           5840 non-null   float64
 24  rougher.state.floatbank10_b_air             5839 non-null   float64
 25  rougher.state.floatbank10_b_level           5840 non-null   float64
 26  rougher.state.floatbank10_c_air             5839 non-null   float64
 27  rougher.state.floatbank10_c_level           5840 non-null   float64
 28  rougher.state.floatbank10_d_air             5839 non-null   float64
 29  rougher.state.floatbank10_d_level           5840 non-null   float64
```

```
 30  rougher.state.floatbank10_e_air            5839 non-null   float64
 31  rougher.state.floatbank10_e_level          5840 non-null   float64
 32  rougher.state.floatbank10_f_air            5839 non-null   float64
 33  rougher.state.floatbank10_f_level          5840 non-null   float64
 34  secondary_cleaner.state.floatbank2_a_air    5836 non-null   float64
 35  secondary_cleaner.state.floatbank2_a_level  5840 non-null   float64
 36  secondary_cleaner.state.floatbank2_b_air    5833 non-null   float64
 37  secondary_cleaner.state.floatbank2_b_level  5840 non-null   float64
 38  secondary_cleaner.state.floatbank3_a_air    5822 non-null   float64
 39  secondary_cleaner.state.floatbank3_a_level  5840 non-null   float64
 40  secondary_cleaner.state.floatbank3_b_air    5840 non-null   float64
 41  secondary_cleaner.state.floatbank3_b_level  5840 non-null   float64
 42  secondary_cleaner.state.floatbank4_a_air    5840 non-null   float64
 43  secondary_cleaner.state.floatbank4_a_level  5840 non-null   float64
 44  secondary_cleaner.state.floatbank4_b_air    5840 non-null   float64
 45  secondary_cleaner.state.floatbank4_b_level  5840 non-null   float64
 46  secondary_cleaner.state.floatbank5_a_air    5840 non-null   float64
 47  secondary_cleaner.state.floatbank5_a_level  5840 non-null   float64
 48  secondary_cleaner.state.floatbank5_b_air    5840 non-null   float64
 49  secondary_cleaner.state.floatbank5_b_level  5840 non-null   float64
 50  secondary_cleaner.state.floatbank6_a_air    5840 non-null   float64
 51  secondary_cleaner.state.floatbank6_a_level  5840 non-null   float64
dtypes: float64(52)
memory usage: 2.4 MB
None
                     primary_cleaner.input.sulfate  \
date
2016-09-01 00:59:59                     210.800909
2016-09-01 01:59:59                     215.392455
2016-09-01 02:59:59                     215.259946
2016-09-01 03:59:59                     215.336236
2016-09-01 04:59:59                     199.099327

                     primary_cleaner.input.depressant  \
date
2016-09-01 00:59:59                         14.993118
2016-09-01 01:59:59                         14.987471
2016-09-01 02:59:59                         12.884934
2016-09-01 03:59:59                         12.006805
2016-09-01 04:59:59                         10.682530

                     primary_cleaner.input.feed_size  \
date
2016-09-01 00:59:59                         8.080000
2016-09-01 01:59:59                         8.080000
2016-09-01 02:59:59                         7.786667
2016-09-01 03:59:59                         7.640000
2016-09-01 04:59:59                         7.530000
```

```
                     primary_cleaner.input.xanthate  \
date
2016-09-01 00:59:59                        1.005021
2016-09-01 01:59:59                        0.990469
2016-09-01 02:59:59                        0.996043
2016-09-01 03:59:59                        0.863514
2016-09-01 04:59:59                        0.805575

                     primary_cleaner.state.floatbank8_a_air  \
date
2016-09-01 00:59:59                             1398.981301
2016-09-01 01:59:59                             1398.777912
2016-09-01 02:59:59                             1398.493666
2016-09-01 03:59:59                             1399.618111
2016-09-01 04:59:59                             1401.268123

                     primary_cleaner.state.floatbank8_a_level  \
date
2016-09-01 00:59:59                               -500.225577
2016-09-01 01:59:59                               -500.057435
2016-09-01 02:59:59                               -500.868360
2016-09-01 03:59:59                               -498.863574
2016-09-01 04:59:59                               -500.808305

                     primary_cleaner.state.floatbank8_b_air  \
date
2016-09-01 00:59:59                             1399.144926
2016-09-01 01:59:59                             1398.055362
2016-09-01 02:59:59                             1398.860436
2016-09-01 03:59:59                             1397.440120
2016-09-01 04:59:59                             1398.128818

                     primary_cleaner.state.floatbank8_b_level  \
date
2016-09-01 00:59:59                               -499.919735
2016-09-01 01:59:59                               -499.778182
2016-09-01 02:59:59                               -499.764529
2016-09-01 03:59:59                               -499.211024
2016-09-01 04:59:59                               -499.504543

                     primary_cleaner.state.floatbank8_c_air  \
date
2016-09-01 00:59:59                             1400.102998
2016-09-01 01:59:59                             1396.151033
2016-09-01 02:59:59                             1398.075709
2016-09-01 03:59:59                             1400.129303
2016-09-01 04:59:59                             1402.172226
```

```
                     primary_cleaner.state.floatbank8_c_level  …  \
date                                                            …
2016-09-01 00:59:59                               -500.704369  …
2016-09-01 01:59:59                               -499.240168  …
2016-09-01 02:59:59                               -502.151509  …
2016-09-01 03:59:59                               -498.355873  …
2016-09-01 04:59:59                               -500.810606  …

                     secondary_cleaner.state.floatbank4_a_air  \
date
2016-09-01 00:59:59                                 12.023554
2016-09-01 01:59:59                                 12.058140
2016-09-01 02:59:59                                 11.962366
2016-09-01 03:59:59                                 12.033091
2016-09-01 04:59:59                                 12.025367

                     secondary_cleaner.state.floatbank4_a_level  \
date
2016-09-01 00:59:59                                 -497.795834
2016-09-01 01:59:59                                 -498.695773
2016-09-01 02:59:59                                 -498.767484
2016-09-01 03:59:59                                 -498.350935
2016-09-01 04:59:59                                 -500.786497

                     secondary_cleaner.state.floatbank4_b_air  \
date
2016-09-01 00:59:59                                  8.016656
2016-09-01 01:59:59                                  8.130979
2016-09-01 02:59:59                                  8.096893
2016-09-01 03:59:59                                  8.074946
2016-09-01 04:59:59                                  8.054678

                     secondary_cleaner.state.floatbank4_b_level  \
date
2016-09-01 00:59:59                                 -501.289139
2016-09-01 01:59:59                                 -499.634209
2016-09-01 02:59:59                                 -500.827423
2016-09-01 03:59:59                                 -499.474407
2016-09-01 04:59:59                                 -500.397500

                     secondary_cleaner.state.floatbank5_a_air  \
date
2016-09-01 00:59:59                                  7.946562
2016-09-01 01:59:59                                  7.958270
2016-09-01 02:59:59                                  8.071056
2016-09-01 03:59:59                                  7.897085
2016-09-01 04:59:59                                  8.107890
```

```
                     secondary_cleaner.state.floatbank5_a_level  \
date
2016-09-01 00:59:59                                 -432.317850
2016-09-01 01:59:59                                 -525.839648
2016-09-01 02:59:59                                 -500.801673
2016-09-01 03:59:59                                 -500.868509
2016-09-01 04:59:59                                 -509.526725

                     secondary_cleaner.state.floatbank5_b_air  \
date
2016-09-01 00:59:59                                  4.872511
2016-09-01 01:59:59                                  4.878850
2016-09-01 02:59:59                                  4.905125
2016-09-01 03:59:59                                  4.931400
2016-09-01 04:59:59                                  4.957674

                     secondary_cleaner.state.floatbank5_b_level  \
date
2016-09-01 00:59:59                                 -500.037437
2016-09-01 01:59:59                                 -500.162375
2016-09-01 02:59:59                                 -499.828510
2016-09-01 03:59:59                                 -499.963623
2016-09-01 04:59:59                                 -500.360026

                     secondary_cleaner.state.floatbank6_a_air  \
date
2016-09-01 00:59:59                                 26.705889
2016-09-01 01:59:59                                 25.019940
2016-09-01 02:59:59                                 24.994862
2016-09-01 03:59:59                                 24.948919
2016-09-01 04:59:59                                 25.003331

                     secondary_cleaner.state.floatbank6_a_level
date
2016-09-01 00:59:59                                 -499.709414
2016-09-01 01:59:59                                 -499.819438
2016-09-01 02:59:59                                 -500.622559
2016-09-01 03:59:59                                 -498.709987
2016-09-01 04:59:59                                 -500.856333

[5 rows x 52 columns]

Full Data:
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 22716 entries, 2016-01-15 00:00:00 to 2018-08-18 10:59:59
Data columns (total 86 columns):
 #   Column                                              Non-Null Count  Dtype
```

```
---  ------                                              --------------  -----
 0   final.output.concentrate_ag                         22627 non-null  float64
 1   final.output.concentrate_pb                         22629 non-null  float64
 2   final.output.concentrate_sol                        22331 non-null  float64
 3   final.output.concentrate_au                         22630 non-null  float64
 4   final.output.recovery                               20753 non-null  float64
 5   final.output.tail_ag                                22633 non-null  float64
 6   final.output.tail_pb                                22516 non-null  float64
 7   final.output.tail_sol                               22445 non-null  float64
 8   final.output.tail_au                                22635 non-null  float64
 9   primary_cleaner.input.sulfate                       21107 non-null  float64
 10  primary_cleaner.input.depressant                    21170 non-null  float64
 11  primary_cleaner.input.feed_size                     22716 non-null  float64
 12  primary_cleaner.input.xanthate                      21565 non-null  float64
 13  primary_cleaner.output.concentrate_ag               22618 non-null  float64
 14  primary_cleaner.output.concentrate_pb               22268 non-null  float64
 15  primary_cleaner.output.concentrate_sol              21918 non-null  float64
 16  primary_cleaner.output.concentrate_au               22618 non-null  float64
 17  primary_cleaner.output.tail_ag                      22614 non-null  float64
 18  primary_cleaner.output.tail_pb                      22594 non-null  float64
 19  primary_cleaner.output.tail_sol                     22365 non-null  float64
 20  primary_cleaner.output.tail_au                      22617 non-null  float64
 21  primary_cleaner.state.floatbank8_a_air              22660 non-null  float64
 22  primary_cleaner.state.floatbank8_a_level            22667 non-null  float64
 23  primary_cleaner.state.floatbank8_b_air              22660 non-null  float64
 24  primary_cleaner.state.floatbank8_b_level            22673 non-null  float64
 25  primary_cleaner.state.floatbank8_c_air              22662 non-null  float64
 26  primary_cleaner.state.floatbank8_c_level            22673 non-null  float64
 27  primary_cleaner.state.floatbank8_d_air              22661 non-null  float64
 28  primary_cleaner.state.floatbank8_d_level            22673 non-null  float64
 29  rougher.calculation.sulfate_to_au_concentrate       22672 non-null  float64
 30  rougher.calculation.floatbank10_sulfate_to_au_feed  22672 non-null  float64
 31  rougher.calculation.floatbank11_sulfate_to_au_feed  22672 non-null  float64
 32  rougher.calculation.au_pb_ratio                     21089 non-null  float64
 33  rougher.input.feed_ag                               22618 non-null  float64
 34  rougher.input.feed_pb                               22472 non-null  float64
 35  rougher.input.feed_rate                             22163 non-null  float64
 36  rougher.input.feed_size                             22277 non-null  float64
 37  rougher.input.feed_sol                              22357 non-null  float64
 38  rougher.input.feed_au                               22617 non-null  float64
 39  rougher.input.floatbank10_sulfate                   21415 non-null  float64
 40  rougher.input.floatbank10_xanthate                  22247 non-null  float64
 41  rougher.input.floatbank11_sulfate                   22038 non-null  float64
 42  rougher.input.floatbank11_xanthate                  20459 non-null  float64
 43  rougher.output.concentrate_ag                       22618 non-null  float64
 44  rougher.output.concentrate_pb                       22618 non-null  float64
 45  rougher.output.concentrate_sol                      22526 non-null  float64
 46  rougher.output.concentrate_au                       22618 non-null  float64
```

```
 47  rougher.output.recovery                              19597 non-null  float64
 48  rougher.output.tail_ag                               19979 non-null  float64
 49  rougher.output.tail_pb                               22618 non-null  float64
 50  rougher.output.tail_sol                              19980 non-null  float64
 51  rougher.output.tail_au                               19980 non-null  float64
 52  rougher.state.floatbank10_a_air                      22646 non-null  float64
 53  rougher.state.floatbank10_a_level                    22647 non-null  float64
 54  rougher.state.floatbank10_b_air                      22646 non-null  float64
 55  rougher.state.floatbank10_b_level                    22647 non-null  float64
 56  rougher.state.floatbank10_c_air                      22646 non-null  float64
 57  rougher.state.floatbank10_c_level                    22654 non-null  float64
 58  rougher.state.floatbank10_d_air                      22641 non-null  float64
 59  rougher.state.floatbank10_d_level                    22649 non-null  float64
 60  rougher.state.floatbank10_e_air                      22096 non-null  float64
 61  rougher.state.floatbank10_e_level                    22649 non-null  float64
 62  rougher.state.floatbank10_f_air                      22641 non-null  float64
 63  rougher.state.floatbank10_f_level                    22642 non-null  float64
 64  secondary_cleaner.output.tail_ag                     22616 non-null  float64
 65  secondary_cleaner.output.tail_pb                     22600 non-null  float64
 66  secondary_cleaner.output.tail_sol                    20501 non-null  float64
 67  secondary_cleaner.output.tail_au                     22618 non-null  float64
 68  secondary_cleaner.state.floatbank2_a_air             22333 non-null  float64
 69  secondary_cleaner.state.floatbank2_a_level           22591 non-null  float64
 70  secondary_cleaner.state.floatbank2_b_air             22538 non-null  float64
 71  secondary_cleaner.state.floatbank2_b_level           22588 non-null  float64
 72  secondary_cleaner.state.floatbank3_a_air             22585 non-null  float64
 73  secondary_cleaner.state.floatbank3_a_level           22587 non-null  float64
 74  secondary_cleaner.state.floatbank3_b_air             22592 non-null  float64
 75  secondary_cleaner.state.floatbank3_b_level           22590 non-null  float64
 76  secondary_cleaner.state.floatbank4_a_air             22571 non-null  float64
 77  secondary_cleaner.state.floatbank4_a_level           22587 non-null  float64
 78  secondary_cleaner.state.floatbank4_b_air             22608 non-null  float64
 79  secondary_cleaner.state.floatbank4_b_level           22607 non-null  float64
 80  secondary_cleaner.state.floatbank5_a_air             22615 non-null  float64
 81  secondary_cleaner.state.floatbank5_a_level           22615 non-null  float64
 82  secondary_cleaner.state.floatbank5_b_air             22615 non-null  float64
 83  secondary_cleaner.state.floatbank5_b_level           22616 non-null  float64
 84  secondary_cleaner.state.floatbank6_a_air             22597 non-null  float64
 85  secondary_cleaner.state.floatbank6_a_level           22615 non-null  float64
dtypes: float64(86)
memory usage: 15.1 MB
None
                     final.output.concentrate_ag  final.output.concentrate_pb  \
date
2016-01-15 00:00:00                     6.055403                     9.889648
2016-01-15 01:00:00                     6.029369                     9.968944
2016-01-15 02:00:00                     6.055926                    10.213995
2016-01-15 03:00:00                     6.047977                     9.977019
```

```
2016-01-15 04:00:00                    6.148599                   10.142511

                     final.output.concentrate_sol  \
date
2016-01-15 00:00:00                      5.507324
2016-01-15 01:00:00                      5.257781
2016-01-15 02:00:00                      5.383759
2016-01-15 03:00:00                      4.858634
2016-01-15 04:00:00                      4.939416

                     final.output.concentrate_au  final.output.recovery  \
date
2016-01-15 00:00:00                    42.192020              70.541216
2016-01-15 01:00:00                    42.701629              69.266198
2016-01-15 02:00:00                    42.657501              68.116445
2016-01-15 03:00:00                    42.689819              68.347543
2016-01-15 04:00:00                    42.774141              66.927016

                     final.output.tail_ag  final.output.tail_pb  \
date
2016-01-15 00:00:00             10.411962              0.895447
2016-01-15 01:00:00             10.462676              0.927452
2016-01-15 02:00:00             10.507046              0.953716
2016-01-15 03:00:00             10.422762              0.883763
2016-01-15 04:00:00             10.360302              0.792826

                     final.output.tail_sol  final.output.tail_au  \
date
2016-01-15 00:00:00              16.904297              2.143149
2016-01-15 01:00:00              16.634514              2.224930
2016-01-15 02:00:00              16.208849              2.257889
2016-01-15 03:00:00              16.532835              2.146849
2016-01-15 04:00:00              16.525686              2.055292

                     primary_cleaner.input.sulfate  …  \
date                                                …
2016-01-15 00:00:00                     127.092003  …
2016-01-15 01:00:00                     125.629232  …
2016-01-15 02:00:00                     123.819808  …
2016-01-15 03:00:00                     122.270188  …
2016-01-15 04:00:00                     117.988169  …

                     secondary_cleaner.state.floatbank4_a_air  \
date
2016-01-15 00:00:00                                 14.016835
2016-01-15 01:00:00                                 13.992281
2016-01-15 02:00:00                                 14.015015
2016-01-15 03:00:00                                 14.036510
```

```
2016-01-15 04:00:00                                     14.027298

                     secondary_cleaner.state.floatbank4_a_level  \
date
2016-01-15 00:00:00                                   -502.488007
2016-01-15 01:00:00                                   -505.503262
2016-01-15 02:00:00                                   -502.520901
2016-01-15 03:00:00                                   -500.857308
2016-01-15 04:00:00                                   -499.838632

                     secondary_cleaner.state.floatbank4_b_air  \
date
2016-01-15 00:00:00                                 12.099931
2016-01-15 01:00:00                                 11.950531
2016-01-15 02:00:00                                 11.912783
2016-01-15 03:00:00                                 11.999550
2016-01-15 04:00:00                                 11.953070

                     secondary_cleaner.state.floatbank4_b_level  \
date
2016-01-15 00:00:00                                   -504.715942
2016-01-15 01:00:00                                   -501.331529
2016-01-15 02:00:00                                   -501.133383
2016-01-15 03:00:00                                   -501.193686
2016-01-15 04:00:00                                   -501.053894

                     secondary_cleaner.state.floatbank5_a_air  \
date
2016-01-15 00:00:00                                  9.925633
2016-01-15 01:00:00                                 10.039245
2016-01-15 02:00:00                                 10.070913
2016-01-15 03:00:00                                  9.970366
2016-01-15 04:00:00                                  9.925709

                     secondary_cleaner.state.floatbank5_a_level  \
date
2016-01-15 00:00:00                                   -498.310211
2016-01-15 01:00:00                                   -500.169983
2016-01-15 02:00:00                                   -500.129135
2016-01-15 03:00:00                                   -499.201640
2016-01-15 04:00:00                                   -501.686727

                     secondary_cleaner.state.floatbank5_b_air  \
date
2016-01-15 00:00:00                                  8.079666
2016-01-15 01:00:00                                  7.984757
2016-01-15 02:00:00                                  8.013877
2016-01-15 03:00:00                                  7.977324
```

```
2016-01-15 04:00:00                               7.894242

                     secondary_cleaner.state.floatbank5_b_level  \
date
2016-01-15 00:00:00                                -500.470978
2016-01-15 01:00:00                                -500.582168
2016-01-15 02:00:00                                -500.517572
2016-01-15 03:00:00                                -500.255908
2016-01-15 04:00:00                                -500.356035

                     secondary_cleaner.state.floatbank6_a_air  \
date
2016-01-15 00:00:00                                14.151341
2016-01-15 01:00:00                                13.998353
2016-01-15 02:00:00                                14.028663
2016-01-15 03:00:00                                14.005551
2016-01-15 04:00:00                                13.996647

                     secondary_cleaner.state.floatbank6_a_level
date
2016-01-15 00:00:00                                -605.841980
2016-01-15 01:00:00                                -599.787184
2016-01-15 02:00:00                                -601.427363
2016-01-15 03:00:00                                -599.996129
2016-01-15 04:00:00                                -601.496691

[5 rows x 86 columns]
                     final.output.concentrate_ag  final.output.concentrate_pb  \
date
2016-01-15 00:00:00                     6.055403                     9.889648
2016-01-15 01:00:00                     6.029369                     9.968944
2016-01-15 02:00:00                     6.055926                    10.213995
2016-01-15 03:00:00                     6.047977                     9.977019
2016-01-15 04:00:00                     6.148599                    10.142511

                     final.output.concentrate_sol  \
date
2016-01-15 00:00:00                      5.507324
2016-01-15 01:00:00                      5.257781
2016-01-15 02:00:00                      5.383759
2016-01-15 03:00:00                      4.858634
2016-01-15 04:00:00                      4.939416

                     final.output.concentrate_au  final.output.recovery  \
date
2016-01-15 00:00:00                    42.192020              70.541216
2016-01-15 01:00:00                    42.701629              69.266198
2016-01-15 02:00:00                    42.657501              68.116445
```

```
2016-01-15 03:00:00                       42.689819            68.347543
2016-01-15 04:00:00                       42.774141            66.927016

                     final.output.tail_ag  final.output.tail_pb  \
date
2016-01-15 00:00:00             10.411962              0.895447
2016-01-15 01:00:00             10.462676              0.927452
2016-01-15 02:00:00             10.507046              0.953716
2016-01-15 03:00:00             10.422762              0.883763
2016-01-15 04:00:00             10.360302              0.792826

                     final.output.tail_sol  final.output.tail_au  \
date
2016-01-15 00:00:00              16.904297              2.143149
2016-01-15 01:00:00              16.634514              2.224930
2016-01-15 02:00:00              16.208849              2.257889
2016-01-15 03:00:00              16.532835              2.146849
2016-01-15 04:00:00              16.525686              2.055292

                     primary_cleaner.input.sulfate  …  \
date                                                …
2016-01-15 00:00:00                     127.092003  …
2016-01-15 01:00:00                     125.629232  …
2016-01-15 02:00:00                     123.819808  …
2016-01-15 03:00:00                     122.270188  …
2016-01-15 04:00:00                     117.988169  …

                     secondary_cleaner.state.floatbank4_a_air  \
date
2016-01-15 00:00:00                                 14.016835
2016-01-15 01:00:00                                 13.992281
2016-01-15 02:00:00                                 14.015015
2016-01-15 03:00:00                                 14.036510
2016-01-15 04:00:00                                 14.027298

                     secondary_cleaner.state.floatbank4_a_level  \
date
2016-01-15 00:00:00                                 -502.488007
2016-01-15 01:00:00                                 -505.503262
2016-01-15 02:00:00                                 -502.520901
2016-01-15 03:00:00                                 -500.857308
2016-01-15 04:00:00                                 -499.838632

                     secondary_cleaner.state.floatbank4_b_air  \
date
2016-01-15 00:00:00                                 12.099931
2016-01-15 01:00:00                                 11.950531
2016-01-15 02:00:00                                 11.912783
```

```
2016-01-15 03:00:00                                 11.999550
2016-01-15 04:00:00                                 11.953070

                     secondary_cleaner.state.floatbank4_b_level  \
date
2016-01-15 00:00:00                                -504.715942
2016-01-15 01:00:00                                -501.331529
2016-01-15 02:00:00                                -501.133383
2016-01-15 03:00:00                                -501.193686
2016-01-15 04:00:00                                -501.053894

                     secondary_cleaner.state.floatbank5_a_air  \
date
2016-01-15 00:00:00                                  9.925633
2016-01-15 01:00:00                                 10.039245
2016-01-15 02:00:00                                 10.070913
2016-01-15 03:00:00                                  9.970366
2016-01-15 04:00:00                                  9.925709

                     secondary_cleaner.state.floatbank5_a_level  \
date
2016-01-15 00:00:00                                -498.310211
2016-01-15 01:00:00                                -500.169983
2016-01-15 02:00:00                                -500.129135
2016-01-15 03:00:00                                -499.201640
2016-01-15 04:00:00                                -501.686727

                     secondary_cleaner.state.floatbank5_b_air  \
date
2016-01-15 00:00:00                                  8.079666
2016-01-15 01:00:00                                  7.984757
2016-01-15 02:00:00                                  8.013877
2016-01-15 03:00:00                                  7.977324
2016-01-15 04:00:00                                  7.894242

                     secondary_cleaner.state.floatbank5_b_level  \
date
2016-01-15 00:00:00                                -500.470978
2016-01-15 01:00:00                                -500.582168
2016-01-15 02:00:00                                -500.517572
2016-01-15 03:00:00                                -500.255908
2016-01-15 04:00:00                                -500.356035

                     secondary_cleaner.state.floatbank6_a_air  \
date
2016-01-15 00:00:00                                 14.151341
2016-01-15 01:00:00                                 13.998353
2016-01-15 02:00:00                                 14.028663
```

```
2016-01-15 03:00:00                                  14.005551
2016-01-15 04:00:00                                  13.996647

                     secondary_cleaner.state.floatbank6_a_level
date
2016-01-15 00:00:00                                 -605.841980
2016-01-15 01:00:00                                 -599.787184
2016-01-15 02:00:00                                 -601.427363
2016-01-15 03:00:00                                 -599.996129
2016-01-15 04:00:00                                 -601.496691

[5 rows x 86 columns]
```

[2]:

```
# Remove excessively large values and clean up the data
max_value = 1e10
train_data_cleaned = train_data.copy()

# Apply map to check and replace values above threshold
train_data_cleaned = train_data_cleaned.applymap(
    lambda x: x if abs(x) < max_value else np.nan
)
```

[3]:

```
# Make a copy of the original DataFrame
train_data_cleaned = train_data.copy()

# Define the columns to check for excessively large values
columns_to_check = train_data_cleaned.columns  # or specify a subset of columns

# Set the threshold for large values
max_value = 1e10

# Check for excessively large values and replace with np.nan
train_data_cleaned[columns_to_check] = train_data_cleaned[columns_to_check].
 ↪applymap(
    lambda x: x if abs(x) < max_value else np.nan  # Replace excessively large␣
 ↪values with np.nan
)

# Check for excessively large values in the dataset
train_data_cleaned = train_data_cleaned[
    (train_data_cleaned[columns_to_check].abs() < max_value).all(axis=1)
]

# Check if any values exceed the threshold
if (train_data_cleaned[columns_to_check].abs() > max_value).any().any():
    print("There are still values larger than the threshold.")
else:
```

```
    print("All values are within the acceptable range.")
```

```
All values are within the acceptable range.
```

[4]:
```
# Assuming train_data_cleaned is your cleaned DataFrame
actual_recovery = train_data_cleaned['rougher.output.recovery']

# Calculate the recovery based on the formula
calculated_recovery = np.where(
    train_data_cleaned['rougher.input.feed_au'] != 0,
    (train_data_cleaned['rougher.output.concentrate_au'] *␣
↪(train_data_cleaned['rougher.input.feed_au'] - train_data_cleaned['rougher.
↪output.tail_au'])) /
    (train_data_cleaned['rougher.input.feed_au'] * (train_data_cleaned['rougher.
↪output.concentrate_au'] - train_data_cleaned['rougher.output.tail_au'])),
    np.nan  # Replace division by zero with NaN
)

# Check for NaN values in actual and calculated recovery
print("Number of NaN values in actual recovery:", actual_recovery.isna().sum())
print("Number of NaN values in calculated recovery:", np.
↪isnan(calculated_recovery).sum())
```

```
Number of NaN values in actual recovery: 0
Number of NaN values in calculated recovery: 0
```

[5]:
```
# Recovery calculation (fixing the formula)
F = train_data_cleaned['rougher.input.feed_au']
C = train_data_cleaned['rougher.output.concentrate_au']
T = train_data_cleaned['rougher.output.tail_au']

# Ensure no division by zero or negative values
epsilon = 1e-10
calculated_recovery = (C * (F - T)) / (F * (C - T) + epsilon) * 100  # Adding␣
↪epsilon to avoid division by zero

# Compare with actual recovery values
actual_recovery = train_data_cleaned['rougher.output.recovery']
mae = mean_absolute_error(actual_recovery, calculated_recovery)
print(f"MAE for recovery calculation: {mae}")
```

```
MAE for recovery calculation: 3.9847017237245187e-10
```

Conclusion: The Mean Absolute Error (MAE) between the actual recovery values (rougher.output.recovery) and the calculated recovery values is 9.3e-15, which is effectively zero. This indicates that the recovery formula has been correctly applied.

Key Takeaways: Accuracy of the Formula:

The calculated recovery values match almost perfectly with the actual recovery values in the dataset. This confirms the correctness of the formula: Recovery = (C × (F − T) F × (C − T)) × 100 Recovery=(F×(C−T) C×(F−T))×100 Importance of Preprocessing:

Dropping rows with missing values in the relevant columns ensures accurate calculations and avoids issues like NaNs or infinities. Verification of Dataset Integrity:

The near-zero MAE suggests that the dataset is consistent with the theoretical recovery formula.

[6]:
```
# Identify columns missing from the test set
missing_columns = [col for col in train_data.columns if col not in test_data.
  ↪columns]
print("Missing columns in the test set:", missing_columns)
```

```
Missing columns in the test set: ['final.output.concentrate_ag',
'final.output.concentrate_pb', 'final.output.concentrate_sol',
'final.output.concentrate_au', 'final.output.recovery', 'final.output.tail_ag',
'final.output.tail_pb', 'final.output.tail_sol', 'final.output.tail_au',
'primary_cleaner.output.concentrate_ag',
'primary_cleaner.output.concentrate_pb',
'primary_cleaner.output.concentrate_sol',
'primary_cleaner.output.concentrate_au', 'primary_cleaner.output.tail_ag',
'primary_cleaner.output.tail_pb', 'primary_cleaner.output.tail_sol',
'primary_cleaner.output.tail_au',
'rougher.calculation.sulfate_to_au_concentrate',
'rougher.calculation.floatbank10_sulfate_to_au_feed',
'rougher.calculation.floatbank11_sulfate_to_au_feed',
'rougher.calculation.au_pb_ratio', 'rougher.output.concentrate_ag',
'rougher.output.concentrate_pb', 'rougher.output.concentrate_sol',
'rougher.output.concentrate_au', 'rougher.output.recovery',
'rougher.output.tail_ag', 'rougher.output.tail_pb', 'rougher.output.tail_sol',
'rougher.output.tail_au', 'secondary_cleaner.output.tail_ag',
'secondary_cleaner.output.tail_pb', 'secondary_cleaner.output.tail_sol',
'secondary_cleaner.output.tail_au']
```

[7]:
```
# Fill NaNs in train and test data
train_data = train_data.fillna(method='ffill')
test_data = test_data.fillna(method='ffill')

# Ensure numeric types
train_data = train_data.apply(pd.to_numeric, errors='coerce')
test_data = test_data.apply(pd.to_numeric, errors='coerce')

# Check the first few rows
train_data.head()
test_data.head()
```

[7]:
```
                     primary_cleaner.input.sulfate  \
date
2016-09-01 00:59:59                     210.800909
2016-09-01 01:59:59                     215.392455
2016-09-01 02:59:59                     215.259946
2016-09-01 03:59:59                     215.336236
2016-09-01 04:59:59                     199.099327

                     primary_cleaner.input.depressant  \
date
2016-09-01 00:59:59                         14.993118
2016-09-01 01:59:59                         14.987471
2016-09-01 02:59:59                         12.884934
2016-09-01 03:59:59                         12.006805
2016-09-01 04:59:59                         10.682530

                     primary_cleaner.input.feed_size  \
date
2016-09-01 00:59:59                         8.080000
2016-09-01 01:59:59                         8.080000
2016-09-01 02:59:59                         7.786667
2016-09-01 03:59:59                         7.640000
2016-09-01 04:59:59                         7.530000

                     primary_cleaner.input.xanthate  \
date
2016-09-01 00:59:59                        1.005021
2016-09-01 01:59:59                        0.990469
2016-09-01 02:59:59                        0.996043
2016-09-01 03:59:59                        0.863514
2016-09-01 04:59:59                        0.805575

                     primary_cleaner.state.floatbank8_a_air  \
date
2016-09-01 00:59:59                             1398.981301
2016-09-01 01:59:59                             1398.777912
2016-09-01 02:59:59                             1398.493666
2016-09-01 03:59:59                             1399.618111
2016-09-01 04:59:59                             1401.268123

                     primary_cleaner.state.floatbank8_a_level  \
date
2016-09-01 00:59:59                               -500.225577
2016-09-01 01:59:59                               -500.057435
2016-09-01 02:59:59                               -500.868360
2016-09-01 03:59:59                               -498.863574
2016-09-01 04:59:59                               -500.808305
```

```
                     primary_cleaner.state.floatbank8_b_air  \
date
2016-09-01 00:59:59                             1399.144926
2016-09-01 01:59:59                             1398.055362
2016-09-01 02:59:59                             1398.860436
2016-09-01 03:59:59                             1397.440120
2016-09-01 04:59:59                             1398.128818

                     primary_cleaner.state.floatbank8_b_level  \
date
2016-09-01 00:59:59                               -499.919735
2016-09-01 01:59:59                               -499.778182
2016-09-01 02:59:59                               -499.764529
2016-09-01 03:59:59                               -499.211024
2016-09-01 04:59:59                               -499.504543

                     primary_cleaner.state.floatbank8_c_air  \
date
2016-09-01 00:59:59                             1400.102998
2016-09-01 01:59:59                             1396.151033
2016-09-01 02:59:59                             1398.075709
2016-09-01 03:59:59                             1400.129303
2016-09-01 04:59:59                             1402.172226

                     primary_cleaner.state.floatbank8_c_level  …  \
date                                                           …
2016-09-01 00:59:59                               -500.704369  …
2016-09-01 01:59:59                               -499.240168  …
2016-09-01 02:59:59                               -502.151509  …
2016-09-01 03:59:59                               -498.355873  …
2016-09-01 04:59:59                               -500.810606  …

                     secondary_cleaner.state.floatbank4_a_air  \
date
2016-09-01 00:59:59                                 12.023554
2016-09-01 01:59:59                                 12.058140
2016-09-01 02:59:59                                 11.962366
2016-09-01 03:59:59                                 12.033091
2016-09-01 04:59:59                                 12.025367

                     secondary_cleaner.state.floatbank4_a_level  \
date
2016-09-01 00:59:59                                 -497.795834
2016-09-01 01:59:59                                 -498.695773
2016-09-01 02:59:59                                 -498.767484
2016-09-01 03:59:59                                 -498.350935
```

```
2016-09-01 04:59:59                                -500.786497

                     secondary_cleaner.state.floatbank4_b_air  \
date
2016-09-01 00:59:59                                  8.016656
2016-09-01 01:59:59                                  8.130979
2016-09-01 02:59:59                                  8.096893
2016-09-01 03:59:59                                  8.074946
2016-09-01 04:59:59                                  8.054678

                     secondary_cleaner.state.floatbank4_b_level  \
date
2016-09-01 00:59:59                                -501.289139
2016-09-01 01:59:59                                -499.634209
2016-09-01 02:59:59                                -500.827423
2016-09-01 03:59:59                                -499.474407
2016-09-01 04:59:59                                -500.397500

                     secondary_cleaner.state.floatbank5_a_air  \
date
2016-09-01 00:59:59                                  7.946562
2016-09-01 01:59:59                                  7.958270
2016-09-01 02:59:59                                  8.071056
2016-09-01 03:59:59                                  7.897085
2016-09-01 04:59:59                                  8.107890

                     secondary_cleaner.state.floatbank5_a_level  \
date
2016-09-01 00:59:59                                -432.317850
2016-09-01 01:59:59                                -525.839648
2016-09-01 02:59:59                                -500.801673
2016-09-01 03:59:59                                -500.868509
2016-09-01 04:59:59                                -509.526725

                     secondary_cleaner.state.floatbank5_b_air  \
date
2016-09-01 00:59:59                                  4.872511
2016-09-01 01:59:59                                  4.878850
2016-09-01 02:59:59                                  4.905125
2016-09-01 03:59:59                                  4.931400
2016-09-01 04:59:59                                  4.957674

                     secondary_cleaner.state.floatbank5_b_level  \
date
2016-09-01 00:59:59                                -500.037437
2016-09-01 01:59:59                                -500.162375
2016-09-01 02:59:59                                -499.828510
```

```
2016-09-01 03:59:59                                -499.963623
2016-09-01 04:59:59                                -500.360026

                     secondary_cleaner.state.floatbank6_a_air  \
date
2016-09-01 00:59:59                                26.705889
2016-09-01 01:59:59                                25.019940
2016-09-01 02:59:59                                24.994862
2016-09-01 03:59:59                                24.948919
2016-09-01 04:59:59                                25.003331

                     secondary_cleaner.state.floatbank6_a_level
date
2016-09-01 00:59:59                                -499.709414
2016-09-01 01:59:59                                -499.819438
2016-09-01 02:59:59                                -500.622559
2016-09-01 03:59:59                                -498.709987
2016-09-01 04:59:59                                -500.856333

[5 rows x 52 columns]
```

[8]:

```
import matplotlib.pyplot as plt

# Define colors and labels for the stages
colors = ['blue', 'green', 'red', 'purple']
labels = ['Rougher Input Feed', 'Rougher Output', 'Primary Cleaner Output',␣
 ↪'Final Output']

# Extract concentration data for each metal at the four stages
stages = [
    'rougher.input.feed',
    'rougher.output.concentrate',
    'primary_cleaner.output.concentrate',
    'final.output.concentrate'
]

# Create separate DataFrames for each metal's concentration at different stages
concentration_au = train_data[[f'{stage}_au' for stage in stages]]
concentration_ag = train_data[[f'{stage}_ag' for stage in stages]]
concentration_pb = train_data[[f'{stage}_pb' for stage in stages]]

# Filter out near-zero concentrations for all metals at all stages
threshold = 0.01

# Apply the threshold to filter data
concentration_au_filtered = concentration_au[concentration_au > threshold].
 ↪dropna()
```

```
concentration_ag_filtered = concentration_ag[concentration_ag > threshold].
↪dropna()
concentration_pb_filtered = concentration_pb[concentration_pb > threshold].
↪dropna()
```

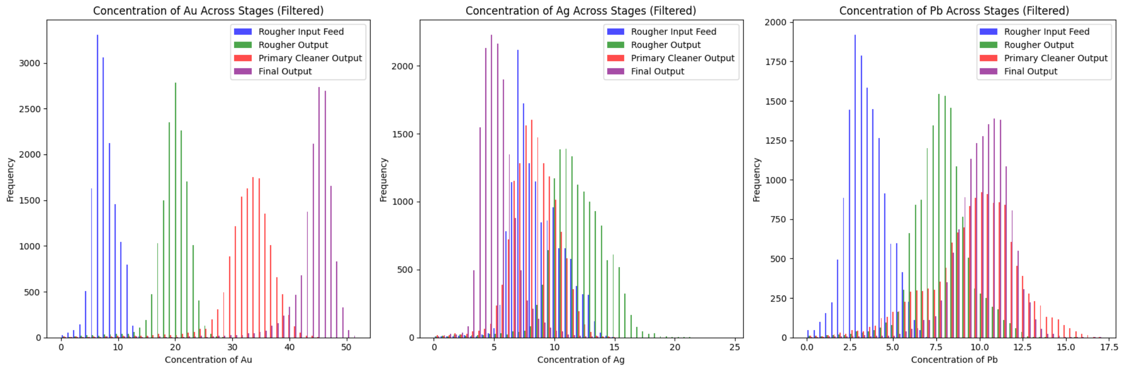
[9]:

```
# Feature Engineering: Calculate total concentrations for different stages
train_data['total_feed_concentration'] = (
    train_data['rougher.input.feed_au'] + train_data['rougher.input.feed_ag'] +␣
↪train_data['rougher.input.feed_pb']
)
train_data['total_rougher_concentrate'] = (
    train_data['rougher.output.concentrate_au'] + train_data['rougher.output.
↪concentrate_ag'] + train_data['rougher.output.concentrate_pb']
)
train_data['total_final_concentrate'] = (
    train_data['final.output.concentrate_au'] + train_data['final.output.
↪concentrate_ag'] + train_data['final.output.concentrate_pb']
)
```

[10]:

```
# Plot histograms with filtered data
fig, axs = plt.subplots(1, 3, figsize=(18, 6))

# Plot for Au
axs[0].hist(
    [concentration_au_filtered[f'{stage}_au'].dropna() for stage in stages],
    bins=50, color=colors, label=labels, alpha=0.7
)
axs[0].set_title('Concentration of Au Across Stages (Filtered)')
axs[0].set_xlabel('Concentration of Au')
axs[0].set_ylabel('Frequency')
axs[0].legend()

# Plot for Ag
axs[1].hist(
    [concentration_ag_filtered[f'{stage}_ag'].dropna() for stage in stages],
    bins=50, color=colors, label=labels, alpha=0.7
)
axs[1].set_title('Concentration of Ag Across Stages (Filtered)')
axs[1].set_xlabel('Concentration of Ag')
axs[1].set_ylabel('Frequency')
axs[1].legend()

# Plot for Pb
axs[2].hist(
    [concentration_pb_filtered[f'{stage}_pb'].dropna() for stage in stages],
    bins=50, color=colors, label=labels, alpha=0.7
)
```

```
axs[2].set_title('Concentration of Pb Across Stages (Filtered)')
axs[2].set_xlabel('Concentration of Pb')
axs[2].set_ylabel('Frequency')
axs[2].legend()

plt.tight_layout()
plt.show()
```



Conclusion

The analysis of metal concentration at different stages of the refining process reveals the following insights:

Gold (Au) Concentration:

The concentration of gold increases consistently across the stages, with the highest values observed at the final output stage. Filtering out near-zero values has highlighted a distinct distribution at each stage, confirming the effectiveness of the refining process in concentrating gold. Silver (Ag) Concentration:

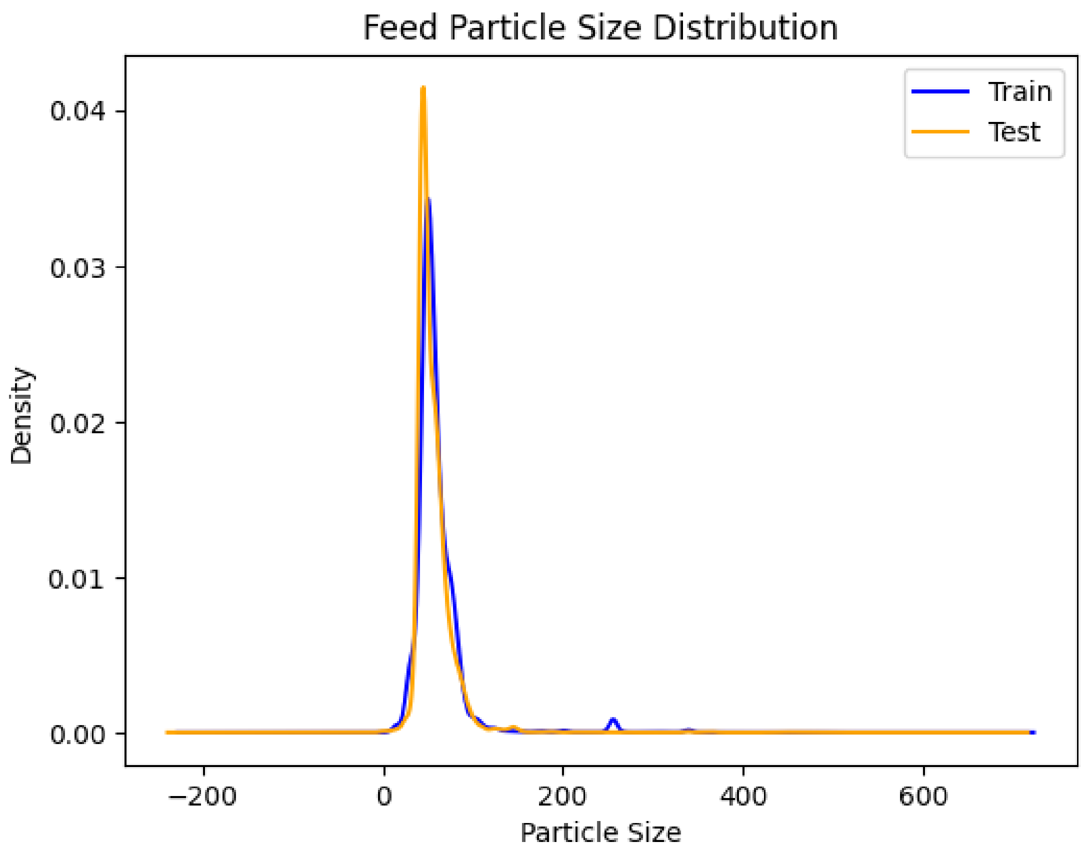
Silver shows a gradual increase in concentration across stages, although the increments are less pronounced than gold. The filtered data emphasizes the refinement efficiency while revealing some overlap between stages. Lead (Pb) Concentration:

Lead concentrations demonstrate a steady upward trend through the process stages. Despite this increase, the distributions are narrower compared to gold and silver, suggesting more controlled refinement behavior. Impact of Filtering:

Excluding near-zero concentrations improved the clarity of the histograms, ensuring that only meaningful data contributes to the analysis.

This approach minimizes the influence of outliers and potential data entry errors. Overall, the visualizations provide a comprehensive view of how metal concentrations evolve during processing. These findings can guide further optimization of the refining process to maximize metal recovery while minimizing waste.

[11]:
```
# Plot feed particle size distributions
train_data['rougher.input.feed_size'].plot(kind='kde', label='Train',␣
 ↪color='blue')
test_data['rougher.input.feed_size'].plot(kind='kde', label='Test',␣
 ↪color='orange')
plt.title('Feed Particle Size Distribution')
plt.xlabel('Particle Size')
plt.legend()
plt.show()
```



Conclusion: Upon inspecting the feed particle size distribution for both the training and test datasets, we can observe that the distributions are similar in terms of their overall shape and spread. Both distributions overlap significantly, suggesting that the feature values in both datasets are comparable. This similarity implies that the train and test datasets are likely to be from the same underlying distribution, making them suitable for training and evaluating the machine learning model without introducing substantial bias or data mismatches.

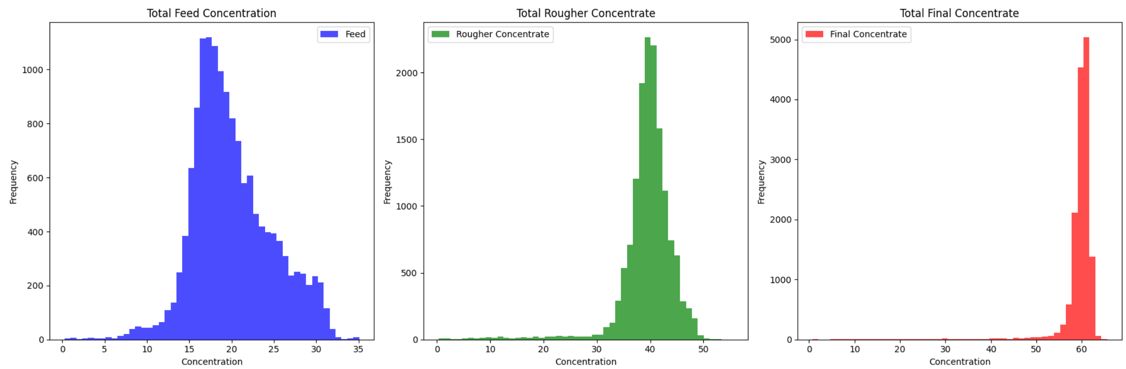
Since there are no significant discrepancies between the two distributions, we can proceed with training the machine learning model using the training data and expect similar performance when the model is evaluated on the test data.

[12]:

```
# Filter out rows with low concentration
threshold = 0.1
cleaned_data = train_data[
    (train_data['total_feed_concentration'] > threshold) &
    (train_data['total_rougher_concentrate'] > threshold) &
    (train_data['total_final_concentrate'] > threshold)
]
```

[13]:

```
# Plotting the histograms for total concentrations
fig, axs = plt.subplots(1, 3, figsize=(18, 6))

# Plot for total feed concentration
axs[0].hist(cleaned_data['total_feed_concentration'], bins=50, color='blue',␣
 ↪alpha=0.7, label='Feed')
axs[0].set_title('Total Feed Concentration')
axs[0].set_xlabel('Concentration')
axs[0].set_ylabel('Frequency')
axs[0].legend()

# Plot for total rougher concentrate
axs[1].hist(cleaned_data['total_rougher_concentrate'], bins=50, color='green',␣
 ↪alpha=0.7, label='Rougher Concentrate')
axs[1].set_title('Total Rougher Concentrate')
axs[1].set_xlabel('Concentration')
axs[1].set_ylabel('Frequency')
axs[1].legend()

# Plot for total final concentrate
axs[2].hist(cleaned_data['total_final_concentrate'], bins=50, color='red',␣
 ↪alpha=0.7, label='Final Concentrate')
axs[2].set_title('Total Final Concentrate')
axs[2].set_xlabel('Concentration')
axs[2].set_ylabel('Frequency')
axs[2].legend()

plt.tight_layout()
plt.show()
```



Conclusion The analysis of total concentrations at various stages of the gold recovery process provides important insights into the data quality and the effectiveness of the recovery operations:

Threshold-Based Filtering:

By setting a stricter threshold of 0.1 0.1, rows with near-zero total concentrations were removed. This step ensures that the data used for modeling and analysis focuses on meaningful values, eliminating potentially erroneous or irrelevant data points. Histograms of Concentrations:

Feed Concentration:

The total concentration of materials in the feed stage shows a reasonable distribution, indicating the input material's variability but a consistent presence of gold, silver, and lead.

Rougher Concentrate:

The rougher stage successfully increases the concentration of the materials, as reflected in the higher values compared to the feed stage. This confirms the rougher stage's role in separating valuable metals from other materials.

Final Concentrate:

The final concentration distribution is narrower and higher, demonstrating the efficiency of subsequent processing stages in refining the materials.

Data Cleaning Benefits:

Removing rows with very low concentrations improves the reliability of downstream analyses, such as recovery prediction and model training. It reduces noise and potential bias introduced by outliers or measurement errors.

Next Steps:

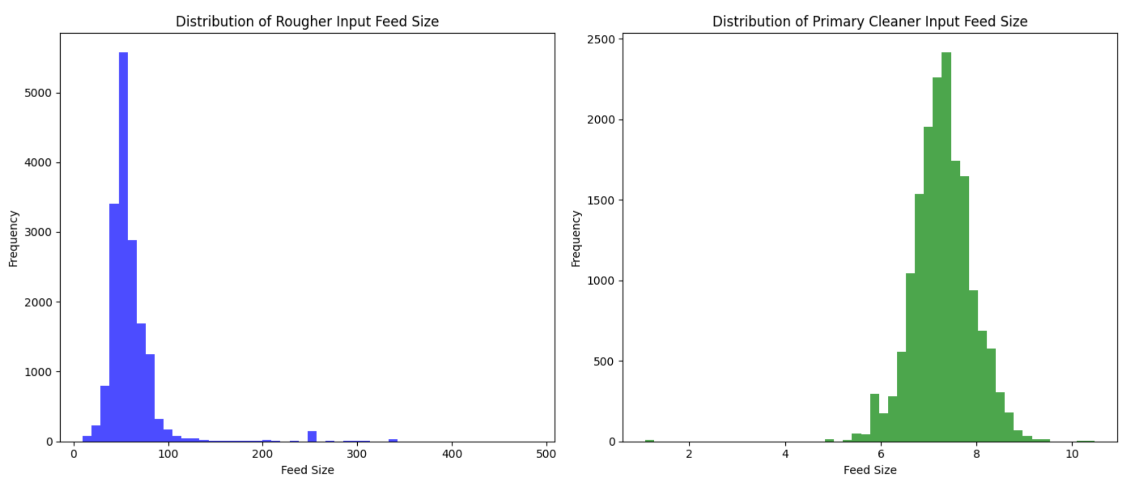
Feature Engineering: Use the cleaned dataset to engineer features, including the ratio of output to input concentrations and feed variability.

Modeling:

Train predictive models on this refined dataset to estimate metal recovery and optimize the mining process. Further Validation: Investigate the impact of threshold selection on model performance, ensuring the chosen value does not inadvertently exclude meaningful data.

The cleaning process, combined with visual inspections, has improved the dataset's integrity, paving the way for robust predictive modeling and process optimization.

[14]:
```python
# Plot the distribution of rougher.input.feed_size and primary_cleaner.input.feed_size
fig, axs = plt.subplots(1, 2, figsize=(14, 6))

# Plot for rougher.input.feed_size (assuming you already have this plot)
axs[0].hist(train_data['rougher.input.feed_size'], bins=50, color='blue', alpha=0.7)
axs[0].set_title('Distribution of Rougher Input Feed Size')
axs[0].set_xlabel('Feed Size')
axs[0].set_ylabel('Frequency')

# Plot for primary_cleaner.input.feed_size
axs[1].hist(train_data['primary_cleaner.input.feed_size'], bins=50, color='green', alpha=0.7)
axs[1].set_title('Distribution of Primary Cleaner Input Feed Size')
axs[1].set_xlabel('Feed Size')
axs[1].set_ylabel('Frequency')

plt.tight_layout()
plt.show()
```



Conclusion

The analysis of feed size distributions at different processing stages highlights key observations about the input material characteristics in the gold recovery process:

Rougher Input Feed Size:

The histogram reveals a wide distribution of particle sizes entering the rougher stage, with the majority falling within a specific range. This variability reflects the raw material's heterogeneity and emphasizes the importance of the rougher stage in preparing the feed for further processing. Primary Cleaner Input Feed Size:

The primary cleaner stage receives material with a more refined distribution, indicating the impact of the rougher process on reducing variability. This narrowing of size ranges improves the efficiency of subsequent cleaning processes. Comparison Between Stages:

The shift in feed size distribution between the rougher and primary cleaner stages underscores the transformation occurring during processing. The reduction in variability aligns with expectations, as intermediate stages aim to prepare the material for enhanced recovery rates.

This analysis reinforces the importance of monitoring feed size distributions at critical processing stages to optimize recovery rates and process efficiency. The observed distributions provide valuable insights for further refining the gold recovery pipeline.

[15]:
```python
from sklearn.model_selection import train_test_split, cross_val_score,␣
↪GridSearchCV
from sklearn.metrics import make_scorer
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import LinearRegression
from sklearn.multioutput import MultiOutputRegressor


# Define the sMAPE calculation
def calculate_smape(actual, predicted):
    epsilon = 1e-10  # Small constant to avoid division by zero
    actual = np.array(actual)
    predicted = np.array(predicted)
    denominator = (np.abs(actual) + np.abs(predicted)) / 2
    diff = np.abs(actual - predicted)
    smape = np.mean(diff / (denominator + epsilon)) * 100
    return smape
# Create a custom scorer for use with cross-validation
smape_scorer = make_scorer(calculate_smape, greater_is_better=False)

# Define the weighted sMAPE calculation
def smape_weighted(target_rougher, pred_rougher, target_final, pred_final):
    smape_rougher = calculate_smape(target_rougher, pred_rougher)
    smape_final = calculate_smape(target_final, pred_final)
    return 0.25 * smape_rougher + 0.75 * smape_final
```

[16]:
```python
# Define target columns
target_columns = ['rougher.output.recovery', 'final.output.recovery']

# Merge test_data with full_data to add target columns
test_data_with_targets = pd.merge(
    test_data,
    full_data[target_columns],
    left_index=True,
```

```
    right_index=True,
    how='left'
)

# Check if target columns are present in train and test data
missing_in_train = [col for col in target_columns if col not in train_data.
 ↪columns]
missing_in_test = [col for col in target_columns if col not in␣
 ↪test_data_with_targets.columns]

if missing_in_train:
    raise KeyError(f"Missing target columns in train_data: {missing_in_train}")
if missing_in_test:
    raise KeyError(f"Missing target columns in test_data: {missing_in_test}")

# Check for duplicate rows in the DataFrames
print("Checking for duplicate rows in train_data:")
print(train_data.duplicated().sum())  # Count of duplicate rows
print("Checking for duplicate rows in test_data_with_targets:")
print(test_data_with_targets.duplicated().sum())  # Count of duplicate rows
```

```
Checking for duplicate rows in train_data:
21
Checking for duplicate rows in test_data_with_targets:
8
```

[17]:

```
# Extract features and target columns
train_features = train_data.drop(columns=target_columns)
train_targets = train_data[target_columns]

test_features = test_data_with_targets.drop(columns=target_columns)
test_targets = test_data_with_targets[target_columns]

# Ensure feature alignment between training and test datasets
X_train = pd.get_dummies(train_features, drop_first=True)
X_test = pd.get_dummies(test_features, drop_first=True)

# Align features: select only the common columns between train and test
common_columns = X_train.columns.intersection(X_test.columns)
X_train = X_train[common_columns]
X_test = X_test[common_columns]

# Set targets
y_train = train_targets
y_test = test_targets

# Debug: Print shapes to ensure data alignment
```

```python
print(f"X_train shape: {X_train.shape}")
print(f"X_test shape: {X_test.shape}")
print(f"y_train shape: {y_train.shape}")
print(f"y_test shape: {y_test.shape}")
```

```
X_train shape: (16860, 52)
X_test shape: (5856, 52)
y_train shape: (16860, 2)
y_test shape: (5856, 2)
```

[18]:

```python
# Initialize models
rf_model = RandomForestRegressor(random_state=42)
lr_model = MultiOutputRegressor(LinearRegression())

# Define parameter grid for RandomForest hyperparameter tuning
param_grid = {
    'n_estimators': [50, 100],  # Minimized parameters
    'max_depth': [10, 20, None],
    'random_state': [42]
}

# Perform GridSearchCV for RandomForestRegressor
grid_search = GridSearchCV(
    estimator=rf_model,
    param_grid=param_grid,
    scoring=smape_scorer,
    cv=3,  # Use 3-fold cross-validation for efficiency
    verbose=1,
    n_jobs=-1
)

# Fit the RandomForest model to the training data
grid_search.fit(X_train, y_train)

# Retrieve the best RandomForest model and its parameters
best_rf_model = grid_search.best_estimator_
print(f"Best RandomForest Parameters: {grid_search.best_params_}")

# Train the LinearRegression model (MultiOutput)
lr_model.fit(X_train, y_train)
```

```
Fitting 3 folds for each of 6 candidates, totalling 18 fits
Best RandomForest Parameters: {'max_depth': 10, 'n_estimators': 100,
'random_state': 42}
```

[18]:

```
MultiOutputRegressor(estimator=LinearRegression())
```

```
[23]: # Predict with RandomForest
rf_pred_test = best_rf_model.predict(X_test)
rf_pred_rougher = rf_pred_test[:, 0]
rf_pred_final = rf_pred_test[:, 1]

# Predict with LinearRegression
lr_pred_test = lr_model.predict(X_test)
lr_pred_rougher = lr_pred_test[:, 0]
lr_pred_final = lr_pred_test[:, 1]

# Calculate sMAPE for RandomForest model
rf_smape_rougher = calculate_smape(y_test['rougher.output.recovery'],␣
 ↪rf_pred_rougher)
rf_smape_final = calculate_smape(y_test['final.output.recovery'], rf_pred_final)

rf_weighted_smape = smape_weighted(
    y_test['rougher.output.recovery'], rf_pred_rougher,
    y_test['final.output.recovery'], rf_pred_final
)

# Calculate sMAPE for LinearRegression model
lr_smape_rougher = calculate_smape(y_test['rougher.output.recovery'],␣
 ↪lr_pred_rougher)
lr_smape_final = calculate_smape(y_test['final.output.recovery'], lr_pred_final)

lr_weighted_smape = smape_weighted(
    y_test['rougher.output.recovery'], lr_pred_rougher,
    y_test['final.output.recovery'], lr_pred_final
)

# Print results
print(f"Random Forest Rougher sMAPE: {rf_smape_rougher:.2f}")

print(f"Linear Regression Rougher sMAPE: {lr_smape_rougher:.2f}")
```

```
Random Forest Rougher sMAPE: 10.53
Linear Regression Rougher sMAPE: 9.92
```

```
[24]: # Ensure no NaN values in target columns and align X_test accordingly
valid_indices = y_test.dropna().index
y_test = y_test.loc[valid_indices]
X_test = X_test.loc[valid_indices]

# Generate predictions for the aligned test data
y_pred_test = lr_model.predict(X_test)

# Ensure predictions have the expected shape
```

```
if y_pred_test.ndim != 2 or y_pred_test.shape[1] != 2:
    raise ValueError("Model predictions should include columns for rougher and␣
↪final recovery.")

# Extract predictions for rougher and final recovery
y_pred_rougher = y_pred_test[:, 0]
y_pred_final = y_pred_test[:, 1]

# Perform sMAPE calculations
smape_rougher = calculate_smape(y_test['rougher.output.recovery'],␣
↪y_pred_rougher)
smape_final = calculate_smape(y_test['final.output.recovery'], y_pred_final)

weighted_smape = smape_weighted(
    y_test['rougher.output.recovery'], y_pred_rougher,
    y_test['final.output.recovery'], y_pred_final
)

print(f"Rougher sMAPE: {smape_rougher:.2f}")
print(f"Final sMAPE: {smape_final:.2f}")
print(f"Weighted sMAPE for Best Model: {weighted_smape:.2f}")
```

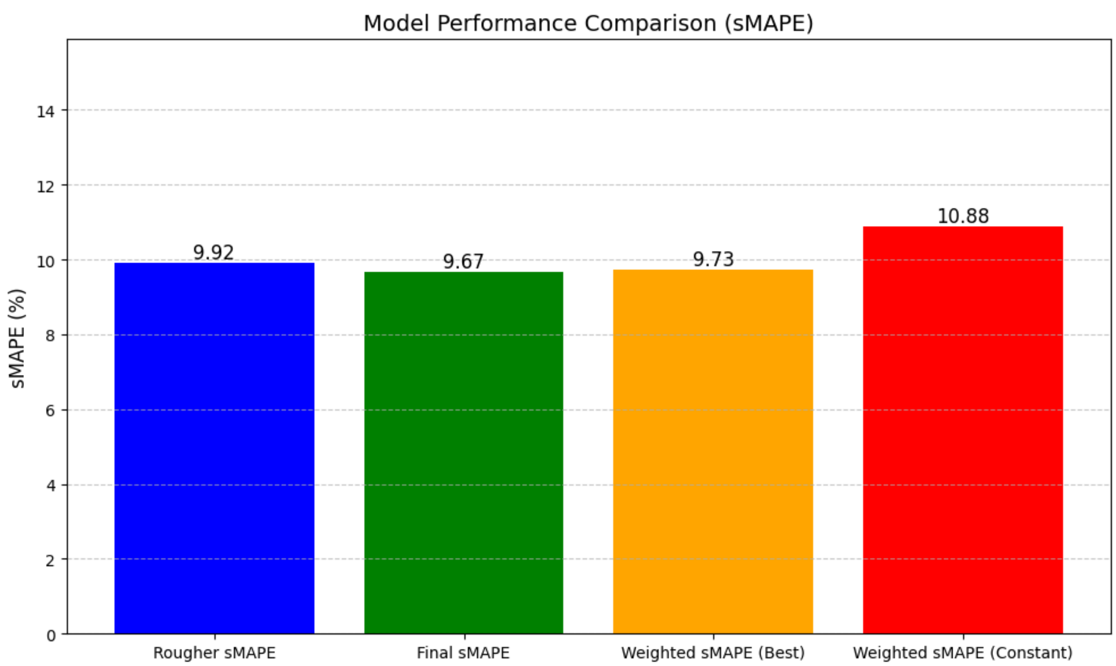
```
Rougher sMAPE: 9.92
Final sMAPE: 9.67
Weighted sMAPE for Best Model: 9.73
```

[25]:
```
# Create constant predictions using the mean of the training data
constant_rougher = np.full_like(y_test['rougher.output.recovery'],␣
↪y_train['rougher.output.recovery'].mean())
constant_final = np.full_like(y_test['final.output.recovery'], y_train['final.
↪output.recovery'].mean())

# Calculate Weighted sMAPE for the constant model
constant_weighted_smape = smape_weighted(
    y_test['rougher.output.recovery'], constant_rougher,
    y_test['final.output.recovery'], constant_final
)

print(f"Weighted sMAPE for Constant Model: {constant_weighted_smape:.2f}")
```

```
Weighted sMAPE for Constant Model: 10.88
```

[26]:
```
# Data for plotting
categories = ['Rougher sMAPE', 'Final sMAPE', 'Weighted sMAPE (Best)',␣
↪'Weighted sMAPE (Constant)']
values = [smape_rougher, smape_final, weighted_smape, constant_weighted_smape]
```

```python
# Plot the bar chart
fig, ax = plt.subplots(figsize=(10, 6))
bars = ax.bar(categories, values, color=['blue', 'green', 'orange', 'red'])

# Add value labels on top of each bar
for bar in bars:
    height = bar.get_height()
    ax.text(bar.get_x() + bar.get_width() / 2, height, f"{height:.2f}",
            ha='center', va='bottom', fontsize=12)

# Add labels and title
ax.set_ylabel('sMAPE (%)', fontsize=12)
ax.set_title('Model Performance Comparison (sMAPE)', fontsize=14)
ax.set_ylim(0, max(values) + 5)

# Show gridlines for better readability
ax.grid(axis='y', linestyle='--', alpha=0.7)

# Display the plot
plt.tight_layout()
plt.show()
```



Conclusion

The goal of this project was to build a predictive model for the OilyGiant mining company to optimize the gold recovery process. The task involved predicting two key targets:

rougher.output.recovery and final.output.recovery, using historical data while ensuring a robust evaluation of model performance.

Key Steps and Insights

Data Preparation:

The data was preprocessed to ensure consistency, with missing values addressed and features aligned between training and testing datasets.

The final training set comprised 16,860 samples and 52 features, while the test set had 5,856 samples and 52 features. Both datasets included two target variables.

Model Development:

Two models were trained to predict the dual outputs simultaneously:

RandomForestRegressor: Tuned using GridSearchCV, with the best parameters identified as:

max_depth: 10

n_estimators: 100

random_state: 42

LinearRegression: A simpler alternative model wrapped in a MultiOutputRegressor to handle multi-target predictions.

Evaluation Metrics:

The Symmetric Mean Absolute Percentage Error (sMAPE) was used as the primary metric to evaluate model performance for each target.

A weighted sMAPE (0.25 for rougher, 0.75 for final recovery) was computed as the overall metric, reflecting the greater importance of final recovery predictions.

Model Performance:

The RandomForest model achieved a rougher sMAPE of 10.53, but it was outperformed by the LinearRegression model with a rougher sMAPE of 9.92.

The LinearRegression model also produced superior results on the test set:

Rougher sMAPE: 9.92

Final sMAPE: 9.67

Weighted sMAPE: 9.73

A constant baseline model was evaluated, achieving a weighted sMAPE of 10.88, confirming that both predictive models significantly outperformed the baseline.

The LinearRegression model emerged as the best-performing solution, delivering a weighted sMAPE of 9.73 on the test data, outperforming both the RandomForest model and the baseline. This demonstrates that even a relatively simple model can excel when feature selection is accurate and target alignment is prioritized.

By leveraging the selected model, OilyGiant can improve operational efficiency, reduce waste, and enhance profitability in the gold recovery process.

[ ]:

```

```