Data-Driven Prediction of Fe and Silica Concentrate using Machine Learning and Mean Process Conditions

Import Libraries

In [29]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.utils import resample
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_sco
import joblib
import os
from sklearn.linear_model import Lasso
```

Data Analysis

In [2]:
```
data = pd.read_csv(r'C:\Users\USER\Desktop\saint martins\Mining\Dataset.csv
```

In [3]:
```
data
```

Out[3]:

	date	% Iron Feed	% Silica Feed	Starch Flow	Amina Flow	Ore Pulp Flow	Ore Pulp pH	Ore Pulp Density	Flotation Column 01 Air Flow	Flotati Colu 02 Fl
0	2017-03-10 01:00:00	55,2	16,98	3019,53	557,434	395,713	10,0664	1,74	249,214	253,2
1	2017-03-10 01:00:00	55,2	16,98	3024,41	563,965	397,383	10,0672	1,74	249,719	250,5
2	2017-03-10 01:00:00	55,2	16,98	3043,46	568,054	399,668	10,068	1,74	249,741	247,8
3	2017-03-10 01:00:00	55,2	16,98	3047,36	568,665	397,939	10,0689	1,74	249,917	254,4
4	2017-03-10 01:00:00	55,2	16,98	3033,69	558,167	400,254	10,0697	1,74	250,203	252,1
...	...	...	...	...	...	...	...	...	...	
737448	2017-09-09 23:00:00	49,75	23,2	2710,94	441,052	386,57	9,62129	1,65365	302,344	298,7
737449	2017-09-09 23:00:00	49,75	23,2	2692,01	473,436	384,939	9,62063	1,65352	303,013	301,8
737450	2017-09-09 23:00:00	49,75	23,2	2692,2	500,488	383,496	9,61874	1,65338	303,662	307,3
737451	2017-09-09 23:00:00	49,75	23,2	1164,12	491,548	384,976	9,61686	1,65324	302,55	301,9
737452	2017-09-09 23:00:00	49,75	23,2	1164,12	468,019	384,801	9,61497	1,6531	300,355	292,8

737453 rows × 24 columns

In [4]:
```
data.head()
```

Out[4]:

	date	% Iron Feed	% Silica Feed	Starch Flow	Amina Flow	Ore Pulp Flow	Ore Pulp pH	Ore Pulp Density	Flotation Column 01 Air Flow	Flotation Column 02 Air Flow	..
0	2017-03-10 01:00:00	55,2	16,98	3019,53	557,434	395,713	10,0664	1,74	249,214	253,235	..
1	2017-03-10 01:00:00	55,2	16,98	3024,41	563,965	397,383	10,0672	1,74	249,719	250,532	..
2	2017-03-10 01:00:00	55,2	16,98	3043,46	568,054	399,668	10,068	1,74	249,741	247,874	..
3	2017-03-10 01:00:00	55,2	16,98	3047,36	568,665	397,939	10,0689	1,74	249,917	254,487	..
4	2017-03-10 01:00:00	55,2	16,98	3033,69	558,167	400,254	10,0697	1,74	250,203	252,136	..

5 rows × 24 columns

In [5]:
```
data.tail()
```

Out[5]:

	date	% Iron Feed	% Silica Feed	Starch Flow	Amina Flow	Ore Pulp Flow	Ore Pulp pH	Ore Pulp Density	Flotation Column 01 Air Flow	Flotati Colu 02 Fl
737448	2017-09-09 23:00:00	49,75	23,2	2710,94	441,052	386,57	9,62129	1,65365	302,344	298,7
737449	2017-09-09 23:00:00	49,75	23,2	2692,01	473,436	384,939	9,62063	1,65352	303,013	301,8
737450	2017-09-09 23:00:00	49,75	23,2	2692,2	500,488	383,496	9,61874	1,65338	303,662	307,3
737451	2017-09-09 23:00:00	49,75	23,2	1164,12	491,548	384,976	9,61686	1,65324	302,55	301,9
737452	2017-09-09 23:00:00	49,75	23,2	1164,12	468,019	384,801	9,61497	1,6531	300,355	292,8

5 rows × 24 columns

In [6]:
```
data.describe()
```

Out[6]:

	date	% Iron Feed	% Silica Feed	Starch Flow	Amina Flow	Ore Pulp Flow	Ore Pulp pH	Ore Pulp Density	Flotation Column 01 Air Flow	Flot Co (
count	737453	737453	737453	737453	737453	737453	737453	737453	737453	73
unique	4097	278	293	409317	319416	180189	131143	105805	43675	8
top	2017-06-28 10:00:00	64,03	6,26	2562,5	534,668	402,246	10,0591	1,75	299,927	25
freq	180	142560	142560	690	959	1735	1509	3214	13683	

4 rows × 24 columns

In [7]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 737453 entries, 0 to 737452
Data columns (total 24 columns):
date                            737453 non-null object
% Iron Feed                     737453 non-null object
% Silica Feed                   737453 non-null object
Starch Flow                     737453 non-null object
Amina Flow                      737453 non-null object
Ore Pulp Flow                   737453 non-null object
Ore Pulp pH                     737453 non-null object
Ore Pulp Density                737453 non-null object
Flotation Column 01 Air Flow    737453 non-null object
Flotation Column 02 Air Flow    737453 non-null object
Flotation Column 03 Air Flow    737453 non-null object
Flotation Column 04 Air Flow    737453 non-null object
Flotation Column 05 Air Flow    737453 non-null object
Flotation Column 06 Air Flow    737453 non-null object
Flotation Column 07 Air Flow    737453 non-null object
Flotation Column 01 Level       737453 non-null object
Flotation Column 02 Level       737453 non-null object
Flotation Column 03 Level       737453 non-null object
Flotation Column 04 Level       737453 non-null object
Flotation Column 05 Level       737453 non-null object
Flotation Column 06 Level       737453 non-null object
Flotation Column 07 Level       737453 non-null object
% Iron Concentrate              737453 non-null object
% Silica Concentrate            737453 non-null object
dtypes: object(24)
memory usage: 135.0+ MB
```

Data Preprocessing

In [8]:
```
data.isnull().sum()
```

Out[8]:
```
date                          0
% Iron Feed                   0
% Silica Feed                 0
Starch Flow                   0
Amina Flow                    0
Ore Pulp Flow                 0
Ore Pulp pH                   0
Ore Pulp Density              0
Flotation Column 01 Air Flow  0
Flotation Column 02 Air Flow  0
Flotation Column 03 Air Flow  0
Flotation Column 04 Air Flow  0
Flotation Column 05 Air Flow  0
Flotation Column 06 Air Flow  0
Flotation Column 07 Air Flow  0
Flotation Column 01 Level     0
Flotation Column 02 Level     0
Flotation Column 03 Level     0
Flotation Column 04 Level     0
Flotation Column 05 Level     0
Flotation Column 06 Level     0
Flotation Column 07 Level     0
% Iron Concentrate            0
% Silica Concentrate          0
dtype: int64
```

In [9]:
```
data.shape
```

Out[9]:
```
(737453, 24)
```

In [10]:
```
df = pd.DataFrame(data)

# Replace commas with dots
df = df.replace(',', '.', regex=True)

# Convert columns to numeric types (ignore 'date' for now)
numeric_columns = df.columns[1:]  # exclude 'date' column
df[numeric_columns] = df[numeric_columns].apply(pd.to_numeric, errors='coer

# Convert 'date' column to datetime type
df['date'] = pd.to_datetime(df['date'])
df['day'] = df['date'].dt.day
df['month'] = df['date'].dt.month
df['year'] = df['date'].dt.year

df.drop(columns=['date'], inplace=True)

print(df.dtypes)  # Check the data types of the columns
```

```
% Iron Feed                     float64
% Silica Feed                   float64
Starch Flow                     float64
Amina Flow                      float64
Ore Pulp Flow                   float64
Ore Pulp pH                     float64
Ore Pulp Density                float64
Flotation Column 01 Air Flow    float64
Flotation Column 02 Air Flow    float64
Flotation Column 03 Air Flow    float64
Flotation Column 04 Air Flow    float64
Flotation Column 05 Air Flow    float64
Flotation Column 06 Air Flow    float64
Flotation Column 07 Air Flow    float64
Flotation Column 01 Level       float64
Flotation Column 02 Level       float64
Flotation Column 03 Level       float64
Flotation Column 04 Level       float64
Flotation Column 05 Level       float64
Flotation Column 06 Level       float64
Flotation Column 07 Level       float64
% Iron Concentrate              float64
% Silica Concentrate            float64
day                               int64
month                             int64
year                              int64
dtype: object
```

In [11]:
```
df
```

Out[11]:

	% Iron Feed	% Silica Feed	Starch Flow	Amina Flow	Ore Pulp Flow	Ore Pulp pH	Ore Pulp Density	Flotation Column 01 Air Flow	Flotation Column 02 Air Flow	Flota Col 0
0	55.20	16.98	3019.53	557.434	395.713	10.06640	1.74000	249.214	253.235	25
1	55.20	16.98	3024.41	563.965	397.383	10.06720	1.74000	249.719	250.532	25
2	55.20	16.98	3043.46	568.054	399.668	10.06800	1.74000	249.741	247.874	25
3	55.20	16.98	3047.36	568.665	397.939	10.06890	1.74000	249.917	254.487	25
4	55.20	16.98	3033.69	558.167	400.254	10.06970	1.74000	250.203	252.136	24
...	...	...	...	...	...	...	...	...	...	
737448	49.75	23.20	2710.94	441.052	386.570	9.62129	1.65365	302.344	298.786	29
737449	49.75	23.20	2692.01	473.436	384.939	9.62063	1.65352	303.013	301.879	29
737450	49.75	23.20	2692.20	500.488	383.496	9.61874	1.65338	303.662	307.397	29
737451	49.75	23.20	1164.12	491.548	384.976	9.61686	1.65324	302.550	301.959	29
737452	49.75	23.20	1164.12	468.019	384.801	9.61497	1.65310	300.355	292.865	29

737453 rows × 26 columns

In [12]:
```
df.info()
```

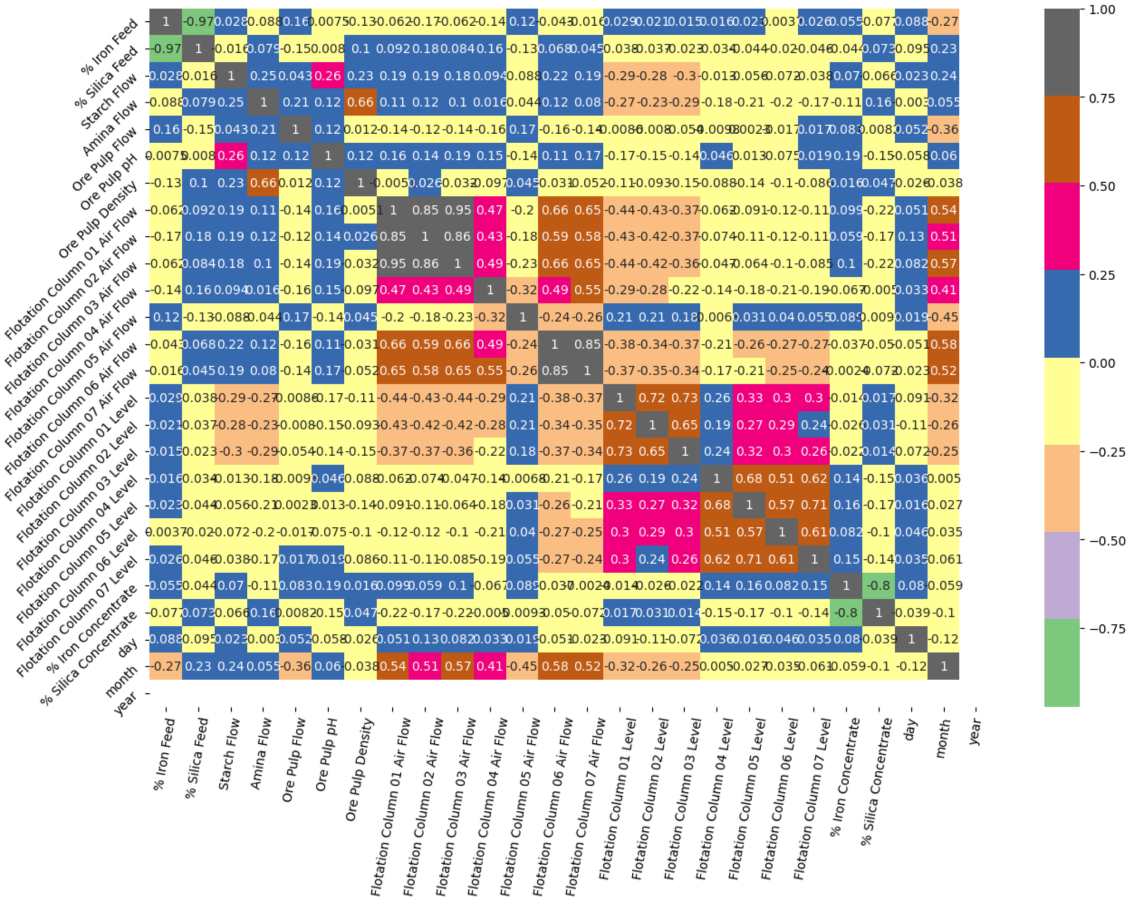
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 737453 entries, 0 to 737452
Data columns (total 26 columns):
% Iron Feed                     737453 non-null float64
% Silica Feed                   737453 non-null float64
Starch Flow                     737453 non-null float64
Amina Flow                      737453 non-null float64
Ore Pulp Flow                   737453 non-null float64
Ore Pulp pH                     737453 non-null float64
Ore Pulp Density                737453 non-null float64
Flotation Column 01 Air Flow    737453 non-null float64
Flotation Column 02 Air Flow    737453 non-null float64
Flotation Column 03 Air Flow    737453 non-null float64
Flotation Column 04 Air Flow    737453 non-null float64
Flotation Column 05 Air Flow    737453 non-null float64
Flotation Column 06 Air Flow    737453 non-null float64
Flotation Column 07 Air Flow    737453 non-null float64
Flotation Column 01 Level       737453 non-null float64
Flotation Column 02 Level       737453 non-null float64
Flotation Column 03 Level       737453 non-null float64
Flotation Column 04 Level       737453 non-null float64
Flotation Column 05 Level       737453 non-null float64
Flotation Column 06 Level       737453 non-null float64
Flotation Column 07 Level       737453 non-null float64
% Iron Concentrate              737453 non-null float64
% Silica Concentrate            737453 non-null float64
day                             737453 non-null int64
month                           737453 non-null int64
year                            737453 non-null int64
dtypes: float64(23), int64(3)
memory usage: 146.3 MB
```

In [13]:
```
df.columns
```

Out[13]:
```
Index(['% Iron Feed', '% Silica Feed', 'Starch Flow', 'Amina Flow',
       'Ore Pulp Flow', 'Ore Pulp pH', 'Ore Pulp Density',
       'Flotation Column 01 Air Flow', 'Flotation Column 02 Air Flow',
       'Flotation Column 03 Air Flow', 'Flotation Column 04 Air Flow',
       'Flotation Column 05 Air Flow', 'Flotation Column 06 Air Flow',
       'Flotation Column 07 Air Flow', 'Flotation Column 01 Level',
       'Flotation Column 02 Level', 'Flotation Column 03 Level',
       'Flotation Column 04 Level', 'Flotation Column 05 Level',
       'Flotation Column 06 Level', 'Flotation Column 07 Level',
       '% Iron Concentrate', '% Silica Concentrate', 'day', 'month', 'yea
r'],
      dtype='object')
```

HeatMap

In [14]:
```
plt.figure(figsize=(15,10))
sns.heatmap(df.corr(),cmap = 'Accent',annot = True)
plt.xticks(rotation = 80)
plt.yticks(rotation = 45)
plt.show()
```



In [15]:
```
x = df.drop(['% Iron Concentrate','% Silica Concentrate'], axis = 1)
x
```

Out[15]:

	% Iron Feed	% Silica Feed	Starch Flow	Amina Flow	Ore Pulp Flow	Ore Pulp pH	Ore Pulp Density	Flotation Column 01 Air Flow	Flotation Column 02 Air Flow	Flota Col 0
0	55.20	16.98	3019.53	557.434	395.713	10.06640	1.74000	249.214	253.235	250
1	55.20	16.98	3024.41	563.965	397.383	10.06720	1.74000	249.719	250.532	250
2	55.20	16.98	3043.46	568.054	399.668	10.06800	1.74000	249.741	247.874	250
3	55.20	16.98	3047.36	568.665	397.939	10.06890	1.74000	249.917	254.487	250
4	55.20	16.98	3033.69	558.167	400.254	10.06970	1.74000	250.203	252.136	249
...	...	...	...	...	...	...	...	...	...	
737448	49.75	23.20	2710.94	441.052	386.570	9.62129	1.65365	302.344	298.786	299
737449	49.75	23.20	2692.01	473.436	384.939	9.62063	1.65352	303.013	301.879	299
737450	49.75	23.20	2692.20	500.488	383.496	9.61874	1.65338	303.662	307.397	299
737451	49.75	23.20	1164.12	491.548	384.976	9.61686	1.65324	302.550	301.959	298
737452	49.75	23.20	1164.12	468.019	384.801	9.61497	1.65310	300.355	292.865	298

737453 rows × 24 columns

In [16]:
```
y = df[['% Iron Concentrate','% Silica Concentrate']]
```

In [17]:
```
y
```

Out[17]:

	% Iron Concentrate	% Silica Concentrate
0	66.91	1.31
1	66.91	1.31
2	66.91	1.31
3	66.91	1.31
4	66.91	1.31
...	...	...
737448	64.27	1.71
737449	64.27	1.71
737450	64.27	1.71
737451	64.27	1.71
737452	64.27	1.71

737453 rows × 2 columns

Data Splitting

In [18]:
```
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size = 0.30,
```

In [19]:
```
x_train
```

Out[19]:

	% Iron Feed	% Silica Feed	Starch Flow	Amina Flow	Ore Pulp Flow	Ore Pulp pH	Ore Pulp Density	Flotation Column 01 Air Flow	Flotat Colu 02 Fl
23233	59.09	10.44	3457.420000	518.127	403.389000	9.340970	1.70031	250.378	248.2
632302	52.12	21.39	3269.680000	437.931	417.079461	9.735860	1.71255	299.710	296.7
482508	52.33	16.95	3331.020000	392.686	391.331000	9.659040	1.69239	248.531	248.2
61287	56.43	13.32	603.572077	424.940	400.782000	9.380000	1.71192	252.588	249.9
601805	48.81	25.31	1302.036042	475.389	378.920503	9.659080	1.67078	299.032	299.6
...	...	...	...	...	...	...	...	...	
259178	64.03	6.26	3392.500000	485.835	397.630000	10.139300	1.66726	298.552	297.8
365838	64.48	3.85	676.264000	565.488	395.330000	9.652550	1.68474	300.267	297.6
131932	55.17	14.35	1001.368837	423.757	399.142000	9.711390	1.57788	250.235	247.9
671155	58.26	12.88	3218.910000	529.337	380.179400	8.753917	1.71273	299.707	298.6
121958	53.51	16.52	2463.000000	509.122	404.123000	9.745650	1.68193	249.668	251.0

516217 rows × 24 columns

In [20]:
```
y_train
```

Out[20]:

	% Iron Concentrate	% Silica Concentrate
23233	66.49	1.81
632302	65.79	1.14
482508	64.48	0.98
61287	65.46	2.64
601805	65.04	2.21
...	...	...
259178	65.27	3.25
365838	65.51	1.73
131932	62.13	5.11
671155	65.72	1.21
121958	65.92	2.59

516217 rows × 2 columns

In [21]:
```
x_test
```

Out[21]:

	% Iron Feed	% Silica Feed	Starch Flow	Amina Flow	Ore Pulp Flow	Ore Pulp pH	Ore Pulp Density	Flotation Column 01 Air Flow	Fl C
198870	56.65	14.83	2969.800000	474.569000	397.085000	9.62077	1.710860	249.935	2
18768	60.24	8.87	3751.370000	552.399000	405.806000	9.22732	1.760000	250.697	2
19259	60.24	8.87	3770.700000	518.311000	401.499000	9.26991	1.750000	250.455	2
616058	48.81	25.31	3633.310000	335.078000	378.750310	10.10850	1.673780	299.487	2
5905	59.89	8.98	794.536195	561.523000	400.664000	9.39431	1.760000	248.236	2
...	...	...	...	...	...	...	...	...	
570827	57.46	10.80	1770.238038	382.540909	378.818391	10.05910	1.574942	300.366	2
124864	50.22	23.80	2000.420000	503.474000	396.925000	9.59433	1.686480	250.217	2
401478	49.69	26.68	822.945708	494.901000	400.375000	9.01198	1.674900	299.479	2
504525	58.87	9.27	1271.136250	579.130000	406.081000	10.46300	1.728470	249.900	2
111673	57.17	10.91	1859.960000	518.192000	400.626000	9.62070	1.741720	219.883	2

221236 rows × 24 columns

In [22]:
```
y_test
```

Out[22]:

	% Iron Concentrate	% Silica Concentrate
198870	63.53	3.54
18768	65.68	1.30
19259	65.61	2.08
616058	67.19	1.45
5905	65.15	2.97
...	...	...
570827	65.44	2.08
124864	65.53	2.23
401478	63.93	3.21
504525	65.64	1.17
111673	65.18	1.52

221236 rows × 2 columns

In [23]:
```
x_train.shape
```

Out[23]: (516217, 24)

In [24]:
```
y_train.shape
```

Out[24]: (516217, 2)

Performance Evaluation

In [25]:
```
a = []
b = []
c = []

def performance_metrics(algorithm, predict, testY):
    mse = mean_squared_error(testY, predict)
    mae = mean_absolute_error(testY, predict)
    r2 = r2_score(testY, predict)
    a.append(mse)
    b.append(mae)
    c.append(r2)
    print(algorithm + ' Mean Squared Error: {:.4f}'.format(mse))
    print(algorithm + ' Mean Absolute Error: {:.4f}'.format(mae))
    print(algorithm + ' R^2 Score: {:.4f}'.format(r2))

    # Plotting best-fit line
    plt.scatter(testY, predict)
    plt.plot([testY.min(), testY.max()], [testY.min(), testY.max()], '--r',
    plt.xlabel('True Values')
    plt.ylabel('Predictions')
    plt.title(algorithm + ' Predictions vs True Values')
    plt.show()
```

XGBoost Regressor

In [26]:
```
import xgboost as xgb

if os.path.exists('model/XGBoostRegressor_weights.pkl'):
    # Load the model from the pkl file
    regressor = joblib.load('model/XGBoostRegressor_weights.pkl')
    predict = regressor.predict(x_test)
    performance_metrics("model/XGBoost Regressor", predict, y_test)
else:
    # Train the regressor on the training data
    regressor = xgb.XGBRegressor()
    regressor.fit(x_train, y_train)
    # Make predictions on the test data
    predict = regressor.predict(x_test)
    # Save the model weights to a pkl file
    joblib.dump(regressor, 'model/XGBoostRegressor_weights.pkl')
    print("XGBoost Regressor model trained and model weights saved.")
    performance_metrics("XGBoost Regressor", predict, y_test)
```

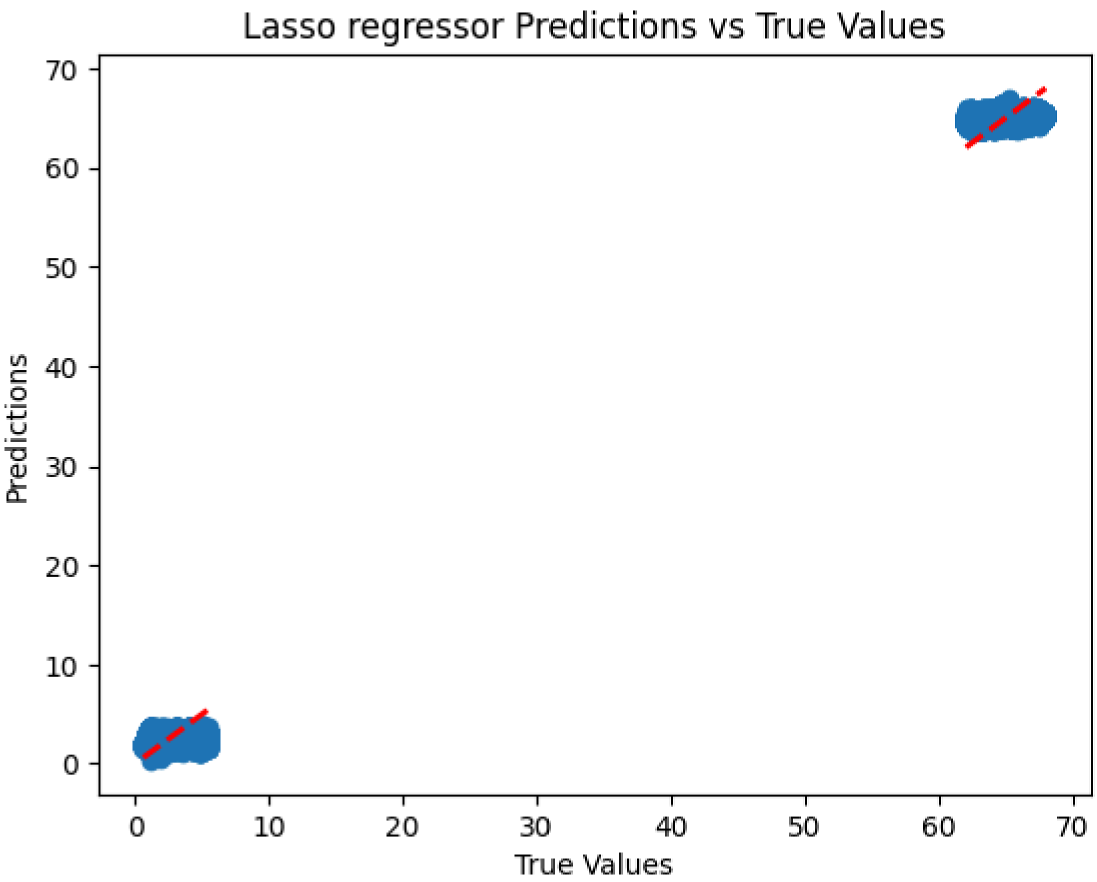
```
model/XGBoost Regressor Mean Squared Error: 0.2859
model/XGBoost Regressor Mean Absolute Error: 0.3857
model/XGBoost Regressor R^2 Score: 0.7725
```

Lasso Regressor

In [31]:
```
if os.path.exists('model/Lasso_weights.pkl'):
    # Load the model from the pkl file
    Lasso_regressor = joblib.load('model/Lasso_weights.pkl')
else:
    # Train the regressor on the training data
    Lasso_regressor = Lasso(alpha=0.1)  # You can adjust the alpha paramete
    Lasso_regressor.fit(x_train, y_train)
    # Save the model weights to a pkl file
    joblib.dump(Lasso_regressor, 'model/Lasso_weights.pkl')

predict = Lasso_regressor.predict(x_test)
performance_metrics("Lasso regressor", predict, y_test)
```

```
Lasso regressor Mean Squared Error: 1.1130
Lasso regressor Mean Absolute Error: 0.8397
Lasso regressor R^2 Score: 0.1144
```



In [30]:
```
test_data = resample(x_test, replace=True, n_samples=100, random_state=42)
test_data
```

Out[30]:

	% Iron Feed	% Silica Feed	Starch Flow	Amina Flow	Ore Pulp Flow	Ore Pulp pH	Ore Pulp Density	Flotation Column 01 Air Flow
516776	55.73	14.36	4936.070000	623.152000	401.410000	10.807536	1.751370	299.828
533781	57.46	10.80	5140.400000	700.763333	403.932659	10.166100	1.702380	250.705
439480	53.26	16.29	4279.300000	462.036000	402.832000	10.530300	1.660020	298.169
728557	56.09	15.79	1881.378505	524.534000	417.855176	9.577050	1.674620	299.840
661982	53.67	19.11	2731.750000	568.848000	380.873000	10.306400	1.720450	300.157
...	...	...	...	...	...	...	...	...
125296	55.17	14.35	2532.260000	360.335000	396.307000	9.435980	1.521894	249.969
547067	57.46	10.80	3507.880000	558.782000	401.879000	10.624000	1.705480	251.834
578011	51.34	23.16	3620.460000	362.959000	380.664224	9.828380	1.669300	299.911
599661	48.81	25.31	2895.820000	328.272000	376.972757	9.956840	1.682930	298.173
475462	53.79	16.57	2562.500000	534.668000	402.246000	9.980470	1.722900	300.806

100 rows × 24 columns

In [32]:
```
predict = regressor.predict(test_data)
predict
```

Out[32]:
```
array([[64.9935419 ,  2.28012383],
       [65.05709819,  2.58868403],
       [65.35330552,  1.94079025],
       [64.81548764,  2.63624686],
       [64.81615915,  2.42487445],
       [65.01634519,  2.4515897 ],
       [64.5616651 ,  2.53572341],
       [65.35194007,  2.04750067],
       [65.15831562,  2.4648116 ],
       [65.32535842,  2.35451584],
       [64.9462003 ,  2.48085631],
       [65.34164727,  1.93622326],
       [64.85538282,  2.34807583],
       [64.95195631,  2.48830279],
       [64.99313902,  2.54770241],
       [65.25935933,  2.27069345],
       [64.95441154,  2.45455106],
       [65.07703703,  2.09994377],
       [64.56708185,  2.43948639],
       [65.55752725,  1.61534454],
       [65.25404812,  1.98851677],
       [64.68847563,  2.36587441],
       [65.23815054,  2.35166888],
       [64.98627375,  2.58899066],
       [64.75952107,  2.92624441],
       [64.91461291,  2.61599554],
       [65.22989839,  2.43154344],
       [65.3452286 ,  1.86420833],
       [64.77022111,  2.83534223],
       [65.12229   ,  2.50701803],
       [65.37863659,  2.00247918],
       [64.62934738,  3.20487759],
       [65.32955209,  1.7944594 ],
       [65.15491449,  2.32547347],
       [65.30184045,  1.94478863],
       [65.14760513,  2.07879891],
       [64.95833681,  2.26368345],
       [65.37898911,  2.15555603],
       [65.43591716,  1.80758772],
       [65.25284951,  2.33264898],
       [65.25178478,  1.91505826],
       [64.6136913 ,  2.51351264],
       [65.23710077,  2.35341321],
       [64.98533499,  2.51978606],
       [65.32575029,  2.42569113],
       [64.72682607,  2.30454239],
       [64.96121373,  2.45014548],
       [64.65648964,  2.51020281],
       [64.94794399,  2.21294366],
       [66.29930596,  0.75275416],
       [65.48351817,  2.07326423],
       [65.5349245 ,  1.63656435],
       [65.36428678,  2.12031254],
       [65.26028498,  2.46807785],
       [64.63307656,  2.46647905],
       [65.01984731,  2.24554181],
       [64.94970952,  2.40951548],
       [65.22976306,  2.29266701],
       [64.84160781,  2.71356016],
       [65.19447535,  2.06672062],
       [65.08474544,  2.21356924],
```

```
[64.66130713,  2.80501479],
[64.6079531 ,  2.98090366],
[65.19026557,  2.33058796],
[64.83585152,  2.11414775],
[65.02076066,  2.28425463],
[65.41217011,  1.91574738],
[64.66160579,  2.68831314],
[64.70435428,  2.88069818],
[65.08012569,  2.495932  ],
[65.09326406,  2.19502071],
[65.04082999,  2.19485674],
[65.39655816,  1.96478231],
[64.92113434,  2.4287235 ],
[65.0923288 ,  2.42069753],
[64.92129008,  2.27682273],
[64.97993695,  2.5091074 ],
[65.15869479,  2.01425914],
[65.19258974,  2.09289733],
[64.79964578,  2.62541555],
[64.85210982,  2.31600201],
[65.26156438,  2.14242056],
[64.92019311,  2.68643863],
[65.15876   ,  2.40232293],
[65.49366348,  2.0294738 ],
[65.38269658,  2.27734297],
[65.27186932,  1.91128312],
[65.06123928,  2.16135413],
[64.99747575,  2.33278481],
[65.16679725,  2.35501093],
[64.94338067,  2.23202446],
[65.32041137,  2.09133543],
[65.11050294,  2.43590867],
[64.71510432,  2.43515508],
[65.45081845,  1.85672751],
[64.97087911,  2.47806803],
[64.68486452,  2.75371159],
[64.92307581,  2.06984704],
[65.02690588,  2.00841004],
[65.26195557,  2.07250101]])
```

In [ ]:

In [ ]: