Exploratory Data Analysis Project

Introduction

This is data set from UCI repository.https://archive.ics.uci.edu/ml/datasets/Chemical+Composition+of+Ceramic+Samples# it has percentage composition of various metallic oxides in various ceramics.It has 19 columns and 88 datapoints

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
composition = pd.read_csv('composition.csv')
from scipy.stats import skew

composition.head()
```

Out[1]:

	Ceramic Name	Part	Na2O	MgO	Al2O3	SiO2	K2O	CaO	TiO2	Fe2O3	MnO	CuO	ZnO	PbO2	Rb2O	SrO	Y2O3	ZrO2	P2
0	FLQ-1-b	Body	0.62	0.38	19.61	71.99	4.84	0.31	0.07	1.18	630	10	70	10	430	0	40	80	
1	FLQ-2-b	Body	0.57	0.47	21.19	70.09	4.98	0.49	0.09	1.12	380	20	80	40	430	-10	40	100	1
2	FLQ-3-b	Body	0.49	0.19	18.60	74.70	3.47	0.43	0.06	1.07	420	20	50	50	380	40	40	80	2
3	FLQ-4-b	Body	0.89	0.30	18.01	74.19	4.01	0.27	0.09	1.23	460	20	70	60	380	10	40	70	2
4	FLQ-5-b	Body	0.03	0.36	18.41	73.99	4.33	0.65	0.05	1.19	380	40	90	40	360	10	30	80	1

In [2]:

```
print(f"There are {composition.shape[0]} rows and {composition.shape[1]} columns in datas
et.")
```

```
There are 88 rows and 19 columns in dataset.
```

In [3]:

```
composition.dtypes
```

Out[3]:

```
Ceramic Name     object
Part             object
Na2O            float64
MgO             float64
Al2O3           float64
SiO2            float64
K2O             float64
CaO             float64
TiO2            float64
Fe2O3           float64
MnO               int64
CuO               int64
ZnO               int64
PbO2              int64
Rb2O              int64
SrO               int64
Y2O3              int64
ZrO2              int64
P2O5              int64
```

```
dtype: object
```

In [4]:

```
composition.isnull().sum()
```

Out[4]:

```
Ceramic Name    0
Part            0
Na2O            0
MgO             0
Al2O3           0
SiO2            0
K2O             0
CaO             0
TiO2            0
Fe2O3           0
MnO             0
CuO             0
ZnO             0
PbO2            0
Rb2O            0
SrO             0
Y2O3            0
ZrO2            0
P2O5            0
dtype: int64
```

Plan for analysis and exploration

Some variables are in ppm(parts per million) and some are in percentage. so first i have converted into percentage for easire operation. then i have calculated sum of all the composition og]f the ceramics to check if they do not deviate too much from 100(>98.5%)

In [5]:

```
ppm_percent=['MnO','CuO','ZnO','PbO2','Rb2O','SrO','Y2O3','ZrO2','P2O5']
composition[ppm_percent] = composition[ppm_percent]/10000
```

In [6]:

```
composition.head()
```

Out[6]:

	Ceramic Name	Part	Na2O	MgO	Al2O3	SiO2	K2O	CaO	TiO2	Fe2O3	MnO	CuO	ZnO	PbO2	Rb2O	SrO	Y2O3	ZrO2
0	FLQ-1-b	Body	0.62	0.38	19.61	71.99	4.84	0.31	0.07	1.18	0.063	0.001	0.007	0.001	0.043	0.000	0.004	0.008
1	FLQ-2-b	Body	0.57	0.47	21.19	70.09	4.98	0.49	0.09	1.12	0.038	0.002	0.008	0.004	0.043	-0.001	0.004	0.010
2	FLQ-3-b	Body	0.49	0.19	18.60	74.70	3.47	0.43	0.06	1.07	0.042	0.002	0.005	0.005	0.038	0.004	0.004	0.008
3	FLQ-4-b	Body	0.89	0.30	18.01	74.19	4.01	0.27	0.09	1.23	0.046	0.002	0.007	0.006	0.038	0.001	0.004	0.007
4	FLQ-5-b	Body	0.03	0.36	18.41	73.99	4.33	0.65	0.05	1.19	0.038	0.004	0.009	0.004	0.036	0.001	0.003	0.008

In [7]:

```
list=['Na2O','MgO','Al2O3','SiO2','K2O','CaO','TiO2','Fe2O3','MnO','CuO','ZnO','PbO2','R
b2O','SrO','Y2O3','ZrO2','P2O5']
```

In [8]:

```
composition
```

Out[8]:

	Ceramic Name	Part	Na2O	MgO	Al2O3	SiO2	K2O	CaO	TiO2	Fe2O3	MnO	CuO	ZnO	PbO2	Rb2O	SrO	Y2O3	ZrO
0	FLQ-1-b	Body	0.62	0.38	19.61	71.99	4.84	0.31	0.07	1.18	0.063	0.001	0.007	0.001	0.043	0.000	0.004	0.0
1	FLQ-2-b	Body	0.57	0.47	21.19	70.09	4.98	0.49	0.09	1.12	0.038	0.002	0.008	0.004	0.043	-0.001	0.004	0.0
2	FLQ-3-b	Body	0.49	0.19	18.60	74.70	3.47	0.43	0.06	1.07	0.042	0.002	0.005	0.005	0.038	0.004	0.004	0.0
3	FLQ-4-b	Body	0.89	0.30	18.01	74.19	4.01	0.27	0.09	1.23	0.046	0.002	0.007	0.006	0.038	0.001	0.004	0.0
4	FLQ-5-b	Body	0.03	0.36	18.41	73.99	4.33	0.65	0.05	1.19	0.038	0.004	0.009	0.004	0.036	0.001	0.003	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
83	DY-M-3-g	Glaze	0.34	0.55	12.37	70.70	5.33	8.06	0.06	1.61	0.125	0.001	0.009	0.003	0.025	0.052	0.003	0.0
84	DY-QC-1-g	Glaze	0.72	0.34	12.20	72.19	6.19	6.06	0.04	1.27	0.170	0.006	0.011	0.001	0.027	0.054	0.004	0.0
85	DY-QC-2-g	Glaze	0.23	0.24	12.99	71.81	5.25	7.15	0.05	1.29	0.075	0.004	0.010	0.000	0.024	0.047	0.004	0.0
86	DY-QC-3-g	Glaze	0.14	0.46	12.62	69.16	4.34	11.03	0.05	1.20	0.092	0.004	0.009	0.002	0.023	0.047	0.004	0.0
87	DY-QC-4-g	Glaze	0.14	0.63	14.25	71.55	4.87	6.43	0.08	1.05	0.080	0.004	0.009	0.002	0.022	0.041	0.004	0.0

88 rows × 19 columns

Here i have checked the interquartile ranges for the complete data.

In [9]:

```
pd.set_option('display.float_format', lambda x: '%.3f' % x)
composition.describe().T
```

Out[9]:

	count	mean	std	min	25%	50%	75%	max
Na2O	88.000	0.472	0.349	0.030	0.247	0.375	0.642	1.880
MgO	88.000	0.430	0.215	0.070	0.270	0.405	0.530	1.320
Al2O3	88.000	17.461	4.703	11.300	13.008	16.205	21.707	26.480
SiO2	88.000	69.825	2.754	63.880	67.737	69.990	71.840	75.950
K2O	88.000	4.978	0.879	2.730	4.338	5.065	5.590	6.740
CaO	88.000	4.172	4.306	0.120	0.180	2.690	7.912	13.690
TiO2	88.000	0.101	0.053	0.040	0.070	0.080	0.130	0.290
Fe2O3	88.000	1.562	0.604	0.580	1.098	1.510	1.925	3.110
MnO	88.000	0.082	0.061	0.018	0.038	0.059	0.098	0.297
CuO	88.000	0.003	0.002	0.000	0.002	0.003	0.004	0.008
ZnO	88.000	0.010	0.003	0.002	0.007	0.009	0.011	0.023
PbO2	88.000	0.004	0.003	0.000	0.002	0.003	0.006	0.010
Rb2O	88.000	0.031	0.007	0.018	0.025	0.032	0.037	0.045
SrO	88.000	0.023	0.026	-0.001	0.001	0.007	0.048	0.078
Y2O3	88.000	0.004	0.001	0.002	0.003	0.004	0.005	0.008
ZrO2	88.000	0.015	0.006	0.005	0.010	0.014	0.017	0.039
P2O5	88.000	0.044	0.040	0.005	0.010	0.036	0.070	0.161

In [10]:

In [10]:

```
composition['Part'].unique().tolist()
```

Out[10]:

```
['Body', 'Glaze']
```

Now i will be dividing the dataset into two parts because the properties of ceramic vary if it is body or glaze. this will let me see the trend of composition of compounds of ceramics classified as body or glaze

In [11]:

```
composition_b=composition[composition['Part']=="Body"]
composition_g=composition[composition['Part']=="Glaze"]
```

In [12]:

```
composition_g.insert(0, 'id', range(1, 1 + len(composition_g)))
composition_g
```

Out[12]:

	id	Ceramic Name	Part	Na2O	MgO	Al2O3	SiO2	K2O	CaO	TiO2	Fe2O3	MnO	CuO	ZnO	PbO2	Rb2O	SrO	Y
44	1	FLQ-1-g	Glaze	0.970	0.070	11.420	74.410	5.700	5.340	0.050	1.040	0.055	0.002	0.006	0.002	0.031	0.015	0.
45	2	FLQ-2-g	Glaze	1.460	0.470	12.960	68.790	4.850	8.880	0.110	1.490	0.095	0.003	0.004	0.000	0.035	0.025	0.
46	3	FLQ-3-g	Glaze	1.050	0.230	13.640	69.900	4.460	8.430	0.070	1.220	0.059	0.002	0.009	0.004	0.037	0.009	0.
47	4	FLQ-4-g	Glaze	0.140	0.410	12.420	67.240	4.290	12.860	0.060	1.580	0.096	0.008	0.007	0.004	0.033	0.016	0.
48	5	FLQ-5-g	Glaze	0.370	1.030	13.150	68.980	5.580	7.910	0.080	1.900	0.080	0.006	0.012	0.000	0.032	0.008	0.
49	6	FLQ-6-g	Glaze	1.090	0.500	13.470	68.510	5.970	7.230	0.190	2.050	0.087	0.002	0.005	0.004	0.036	0.024	0.
50	7	FLQ-7-g	Glaze	1.160	0.580	13.830	71.370	5.140	5.990	0.080	0.850	0.105	0.006	0.009	0.002	0.038	0.024	0.
51	8	FLQ-8-g	Glaze	1.010	0.100	11.840	71.130	3.840	9.400	0.100	1.580	0.052	0.004	0.005	0.004	0.018	0.016	0.
52	9	FLQ-9-g	Glaze	1.880	0.580	12.950	67.580	2.980	10.280	0.120	2.610	0.059	0.008	0.009	0.003	0.023	0.019	0.
53	10	FLQ-10-g	Glaze	0.730	0.250	13.000	71.010	5.780	6.430	0.100	1.710	0.109	0.004	0.007	0.002	0.033	0.018	0.
54	11	FLQ-11-g	Glaze	0.680	0.270	12.740	71.930	6.160	6.100	0.090	1.040	0.081	0.002	0.002	0.000	0.036	0.020	0.
55	12	FLQ-12-g	Glaze	1.290	0.320	12.830	68.810	5.800	7.920	0.070	1.970	0.068	0.003	0.004	0.002	0.034	0.021	0.
56	13	FLQ-13-g	Glaze	1.270	0.540	13.010	73.110	5.460	4.120	0.130	1.360	0.066	0.001	0.004	0.002	0.040	0.016	0.
57	14	DY-BS-1-g	Glaze	0.280	0.520	14.760	68.650	3.630	10.460	0.070	0.640	0.200	0.003	0.012	0.010	0.021	0.057	0.
58	15	DY-BS-2-g	Glaze	0.340	0.970	13.760	65.530	3.570	13.690	0.060	1.070	0.219	0.002	0.007	0.002	0.018	0.078	0.
59	16	DY-BS-3-g	Glaze	0.500	0.660	11.300	69.900	3.880	11.720	0.060	0.980	0.127	0.003	0.013	0.002	0.022	0.068	0.
60	17	DY-BS-4-g	Glaze	0.510	0.400	13.540	71.350	4.140	8.210	0.070	0.780	0.091	0.002	0.012	0.003	0.019	0.063	0.
61	18	DY-BS-5-g	Glaze	0.140	0.500	14.150	68.910	5.100	9.210	0.080	0.910	0.141	0.001	0.005	0.003	0.025	0.066	0.
62	19	DY-BS-6-g	Glaze	0.380	1.040	12.370	67.700	3.890	12.170	0.070	1.380	0.183	0.004	0.014	0.003	0.023	0.054	0.
63	20	DY-BS-7-g	Glaze	0.250	0.470	14.090	68.460	4.710	9.810	0.070	1.140	0.146	0.001	0.010	0.002	0.027	0.061	0.
64	21	DY-NS-1-g	Glaze	0.200	0.530	12.830	72.240	5.030	6.920	0.070	1.180	0.095	0.001	0.005	0.001	0.025	0.052	0.
65	22	DY-NS-	Glaze	0.190	0.570	13.610	70.060	4.700	9.140	0.070	0.660	0.064	0.002	0.007	0.000	0.022	0.064	0.

2-g

	id	Ceramic Name	Part	Na2O	MgO	Al2O3	SiO2	K2O	CaO	TiO2	Fe2O3	MnO	CuO	ZnO	PbO2	Rb2O	SrO	Y:
66	23	DY-NS-3-g	Glaze	0.690	0.350	13.860	71.380	4.940	6.710	0.160	0.910	0.077	0.003	0.012	0.006	0.023	0.046	0.
67	24	DY-NS-4-g	Glaze	0.650	0.780	13.810	70.370	5.910	6.140	0.080	1.270	0.150	0.003	0.011	0.003	0.029	0.049	0.
68	25	DY-NS-5-g	Glaze	0.030	1.010	13.050	72.280	5.560	5.770	0.070	1.240	0.175	0.002	0.015	0.003	0.027	0.033	0.
69	26	DY-NS-6-g	Glaze	0.610	0.350	12.730	71.600	5.790	6.960	0.070	0.880	0.174	0.004	0.014	0.007	0.024	0.055	0.
70	27	DY-NS-7-g	Glaze	0.030	0.540	13.970	67.300	3.560	12.530	0.100	0.970	0.134	0.004	0.007	0.003	0.018	0.069	0.
71	28	DY-NS-8-g	Glaze	0.310	0.530	14.630	68.220	3.600	11.060	0.060	0.580	0.195	0.004	0.023	0.008	0.018	0.070	0.
72	29	DY-Y-1-g	Glaze	0.250	0.500	12.930	71.590	5.500	6.990	0.060	1.180	0.129	0.002	0.009	0.004	0.025	0.052	0.
73	30	DY-Y-2-g	Glaze	0.110	0.320	11.330	75.950	5.870	4.370	0.090	0.960	0.160	0.005	0.020	0.005	0.030	0.030	0.
74	31	DY-Y-3-g	Glaze	0.240	0.390	12.640	74.080	5.110	5.760	0.080	0.710	0.062	0.004	0.010	0.003	0.025	0.061	0.
75	32	DY-Y-4-g	Glaze	0.150	0.330	13.140	73.760	4.870	6.030	0.050	0.680	0.070	0.002	0.011	0.000	0.022	0.056	0.
76	33	DY-Y-5-g	Glaze	1.000	0.520	13.700	70.520	6.740	5.340	0.050	1.110	0.151	0.007	0.014	0.005	0.032	0.061	0.
77	34	DY-Y-6-g	Glaze	0.660	0.530	12.950	72.160	6.570	4.290	0.140	1.690	0.225	0.005	0.009	0.006	0.028	0.040	0.
78	35	DY-Y-7-g	Glaze	0.320	0.580	12.890	70.600	4.270	8.700	0.050	1.610	0.088	0.002	0.007	0.000	0.023	0.047	0.
79	36	DY-Y-8-g	Glaze	0.710	0.570	11.610	71.040	5.310	8.130	0.060	1.580	0.245	0.004	0.015	0.002	0.024	0.048	0.
80	37	DY-Y-9-g	Glaze	0.400	0.470	14.380	66.590	4.160	12.160	0.070	0.770	0.087	0.000	0.008	0.003	0.021	0.065	0.
81	38	DY-M-1-g	Glaze	0.030	1.320	13.550	67.660	5.410	8.910	0.110	2.000	0.297	0.006	0.018	0.009	0.022	0.065	0.
82	39	DY-M-2-g	Glaze	0.370	0.470	13.560	72.770	6.540	4.120	0.080	1.090	0.256	0.002	0.008	0.001	0.029	0.064	0.
83	40	DY-M-3-g	Glaze	0.340	0.550	12.370	70.700	5.330	8.060	0.060	1.610	0.125	0.001	0.009	0.003	0.025	0.052	0.
84	41	DY-QC-1-g	Glaze	0.720	0.340	12.200	72.190	6.190	6.060	0.040	1.270	0.170	0.006	0.011	0.001	0.027	0.054	0.
85	42	DY-QC-2-g	Glaze	0.230	0.240	12.990	71.810	5.250	7.150	0.050	1.290	0.075	0.004	0.010	0.000	0.024	0.047	0.
86	43	DY-QC-3-g	Glaze	0.140	0.460	12.620	69.160	4.340	11.030	0.050	1.200	0.092	0.004	0.009	0.002	0.023	0.047	0.
87	44	DY-QC-4-g	Glaze	0.140	0.630	14.250	71.550	4.870	6.430	0.080	1.050	0.080	0.004	0.009	0.002	0.022	0.041	0.

In [13]:

```
composition_b.insert(0, 'id', range(1, 1 + len(composition_b)))
composition_b
```

Out[13]:

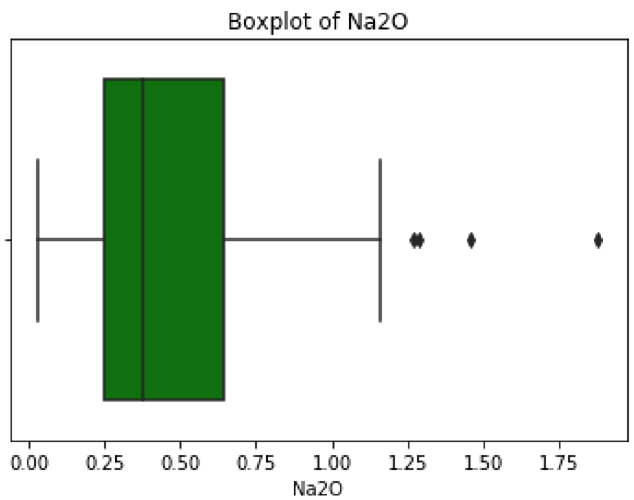
	id	Ceramic Name	Part	Na2O	MgO	Al2O3	SiO2	K2O	CaO	TiO2	Fe2O3	MnO	CuO	ZnO	PbO2	Rb2O	SrO	Y2C
0	1	FLQ-1-b	Body	0.620	0.380	19.610	71.990	4.840	0.310	0.070	1.180	0.063	0.001	0.007	0.001	0.043	0.000	0.0(
1	2	FLQ-2-b	Body	0.570	0.470	21.190	70.090	4.980	0.490	0.090	1.120	0.038	0.002	0.008	0.004	0.043	-0.001	0.0(

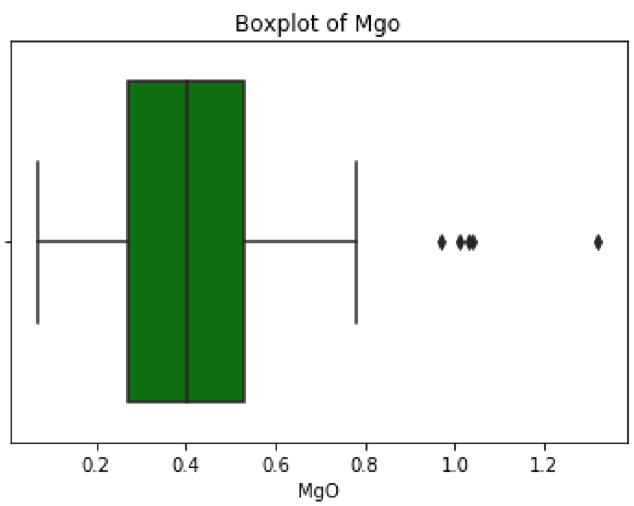
	id	Ceramic Name	Part	Na2O	MgO	Al2O3	SiO2	K2O	CaO	TiO2	Fe2O3	MnO	CuO	ZnO	PbO2	Rb2O	SrO	Y2O
2	3	FLQ-3-b	Body	0.490	0.190	18.600	74.700	3.470	0.490	0.060	1.070	0.042	0.002	0.005	0.005	0.038	0.004	0.0
3	4	FLQ-4-b	Body	0.890	0.300	18.010	74.190	4.010	0.270	0.090	1.230	0.046	0.002	0.007	0.006	0.038	0.001	0.0
4	5	FLQ-5-b	Body	0.030	0.360	18.410	73.990	4.330	0.650	0.050	1.190	0.038	0.004	0.009	0.004	0.036	0.001	0.0
5	6	FLQ-6-b	Body	0.620	0.180	18.820	73.790	4.280	0.300	0.040	0.960	0.035	0.002	0.008	0.001	0.039	0.001	0.0
6	7	FLQ-7-b	Body	0.450	0.330	17.650	74.990	3.530	0.700	0.070	1.280	0.065	0.002	0.009	0.009	0.041	0.003	0.0
7	8	FLQ-8-b	Body	0.590	0.450	21.420	71.460	3.470	0.350	0.050	1.200	0.050	0.001	0.007	0.005	0.038	0.007	0.0
8	9	FLQ-9-b	Body	0.420	0.530	23.120	67.410	3.810	0.740	0.160	2.810	0.034	0.004	0.012	0.003	0.037	0.002	0.0
9	10	FLQ-10-b	Body	0.560	0.490	19.860	72.000	4.510	0.250	0.230	1.100	0.033	0.002	0.007	0.002	0.035	0.001	0.0
10	11	FLQ-11-b	Body	0.350	0.230	19.530	72.870	4.620	0.280	0.070	1.050	0.032	0.007	0.004	0.002	0.045	0.001	0.0
11	12	FLQ-12-b	Body	0.430	0.700	19.350	71.210	4.770	1.260	0.040	1.230	0.042	0.000	0.009	0.003	0.043	0.001	0.0
12	13	FLQ-13-b	Body	0.760	0.440	19.450	72.520	3.940	0.580	0.070	1.240	0.042	0.005	0.010	0.001	0.039	0.003	0.0
13	14	DY-BS-1-b	Body	0.030	0.260	18.340	73.260	5.110	0.140	0.120	1.740	0.048	0.000	0.010	0.009	0.037	0.002	0.0
14	15	DY-BS-2-b	Body	0.710	0.310	24.470	65.200	6.160	0.170	0.090	1.890	0.043	0.004	0.011	0.002	0.036	0.001	0.0
15	16	DY-BS-3-b	Body	0.250	0.240	23.070	67.370	5.800	0.180	0.140	1.940	0.028	0.000	0.007	0.006	0.037	0.001	0.0
16	17	DY-BS-4-b	Body	0.430	0.470	20.670	70.070	5.270	0.130	0.200	1.770	0.030	0.001	0.008	0.004	0.035	0.004	0.0
17	18	DY-BS-5-b	Body	0.280	0.220	20.890	70.770	4.790	0.150	0.130	1.770	0.023	0.006	0.009	0.006	0.029	0.000	0.0
18	19	DY-BS-6-b	Body	0.250	0.490	20.790	69.920	5.370	0.180	0.150	1.860	0.047	0.003	0.012	0.008	0.037	0.001	0.0
19	20	DY-BS-7-b	Body	0.160	0.340	23.770	66.310	6.050	0.160	0.160	2.050	0.028	0.004	0.008	0.008	0.040	0.002	0.0
20	21	DY-NS-1-b	Body	0.030	0.360	25.130	64.580	6.560	0.170	0.070	2.110	0.040	0.002	0.009	0.004	0.038	0.001	0.0
21	22	DY-NS-2-b	Body	0.240	0.550	22.810	66.310	5.590	0.200	0.180	3.110	0.035	0.002	0.011	0.007	0.033	0.003	0.0
22	23	DY-NS-3-b	Body	0.290	0.330	23.490	67.940	4.460	0.170	0.160	2.150	0.031	0.000	0.008	0.002	0.023	0.002	0.0
23	24	DY-NS-4-b	Body	0.450	0.460	25.800	64.420	4.570	0.150	0.190	2.970	0.024	0.006	0.011	0.004	0.027	0.000	0.0
24	25	DY-NS-5-b	Body	0.290	0.230	22.260	68.930	4.190	0.120	0.290	2.680	0.025	0.005	0.012	0.006	0.027	0.002	0.0
25	26	DY-NS-6-b	Body	0.410	0.250	19.480	71.970	5.030	0.280	0.040	1.530	0.049	0.008	0.011	0.006	0.032	0.001	0.0
26	27	DY-NS-7-b	Body	0.340	0.470	22.480	67.800	5.450	0.490	0.140	1.830	0.031	0.001	0.012	0.004	0.032	0.001	0.0
27	28	DY-NS-8-b	Body	0.710	0.250	19.490	70.850	5.430	0.270	0.140	1.880	0.037	0.003	0.011	0.007	0.033	0.003	0.0
28	29	DY-Y-1-b	Body	0.300	0.370	25.000	65.090	6.170	0.180	0.070	1.830	0.036	0.003	0.008	0.010	0.037	0.001	0.0
29	30	DY-Y-2-b	Body	0.500	0.320	25.150	65.370	5.340	0.120	0.100	2.100	0.040	0.003	0.009	0.001	0.038	0.000	0.0
30	31	DY-Y-3-b	Body	0.310	0.410	22.770	67.750	4.500	0.180	0.240	2.850	0.035	0.001	0.010	0.000	0.031	0.003	0.0
31	32	DY-Y-4-b	Body	0.200	0.490	24.070	66.180	4.660	0.200	0.210	2.980	0.041	0.002	0.012	0.009	0.033	0.002	0.0
32	33	DY-Y-5-b	Body	0.260	0.320	26.480	63.880	5.620	0.130	0.150	2.170	0.037	0.004	0.007	0.007	0.039	0.000	0.0

	id	Ceramic Name	Part	Na2O	MgO	Al2O3	SiO2	K2O	CaO	TiO2	Fe2O3	MnO	CuO	ZnO	PbO2	Rb2O	SrO	Y2O
33	34	DY-Y-6-b	Body	0.860	0.190	21.540	68.950	5.590	0.160	0.080	1.630	0.062	0.003	0.012	0.007	0.039	0.002	0.0
34	35	DY-Y-7-b	Body	0.170	0.530	21.400	71.020	2.730	0.140	0.280	2.730	0.018	0.003	0.008	0.001	0.029	0.001	0.0
35	36	DY-Y-8-b	Body	0.320	0.220	22.340	68.860	5.280	0.130	0.120	1.730	0.030	0.000	0.010	0.006	0.037	0.000	0.0
36	37	DY-Y-9-b	Body	0.390	0.350	23.200	66.400	6.250	0.210	0.100	2.090	0.047	0.004	0.011	0.008	0.039	0.003	0.0
37	38	DY-M-1-b	Body	0.180	0.180	23.250	67.860	5.370	0.140	0.110	1.920	0.039	0.002	0.012	0.006	0.034	0.000	0.0
38	39	DY-M-2-b	Body	0.420	0.180	22.090	69.030	5.170	0.170	0.070	1.860	0.051	0.005	0.012	0.003	0.031	0.001	0.0
39	40	DY-M-3-b	Body	0.290	0.210	24.350	65.430	6.070	0.130	0.100	2.410	0.042	0.002	0.010	0.004	0.038	0.001	0.0
40	41	DY-QC-1-b	Body	0.550	0.270	21.580	69.910	4.610	0.130	0.100	1.860	0.033	0.004	0.011	0.001	0.031	0.000	0.0
41	42	DY-QC-2-b	Body	0.640	0.190	21.310	69.340	4.900	0.220	0.140	2.270	0.042	0.004	0.012	0.006	0.032	0.002	0.0
42	43	DY-QC-3-b	Body	0.140	0.270	24.010	66.700	5.470	0.230	0.090	2.080	0.042	0.003	0.007	0.007	0.032	0.003	0.0
43	44	DY-QC-4-b	Body	0.310	0.280	23.230	67.080	5.630	0.160	0.130	2.180	0.036	0.002	0.009	0.003	0.032	0.001	0.0

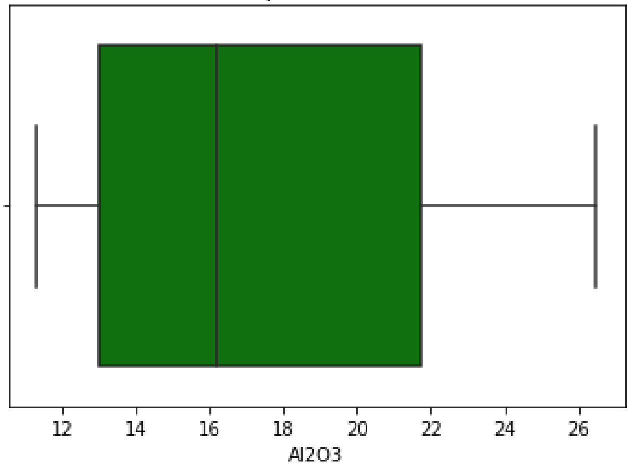
In [14]:

```
for column in composition[list]:
    fig, ax = plt.subplots()
    sns.boxplot(x=composition[column], ax=ax, color="green");
    plt.title(f'Boxplot of {column.title().replace("_", " ")}')
```

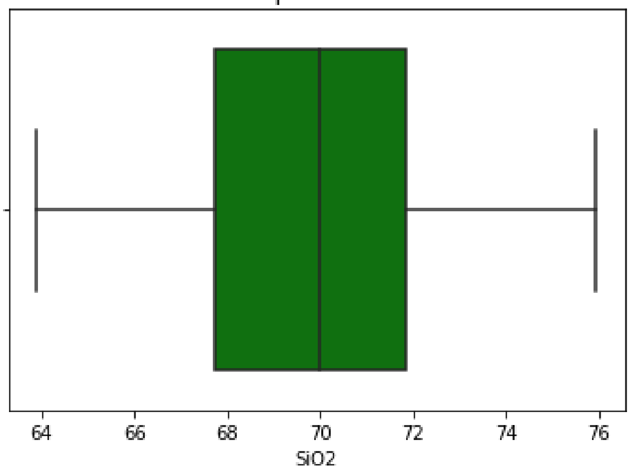




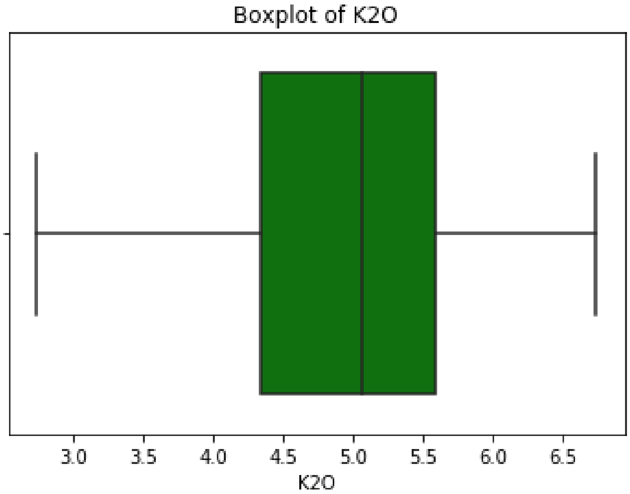
Boxplot of Al2O3



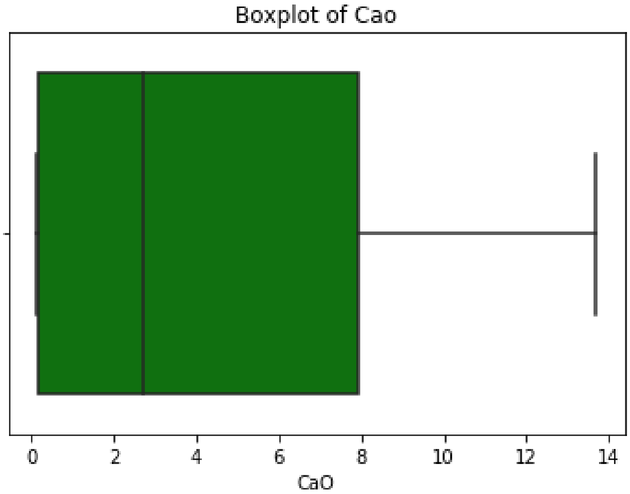
Boxplot of Sio2



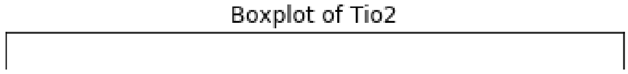
Boxplot of K2O

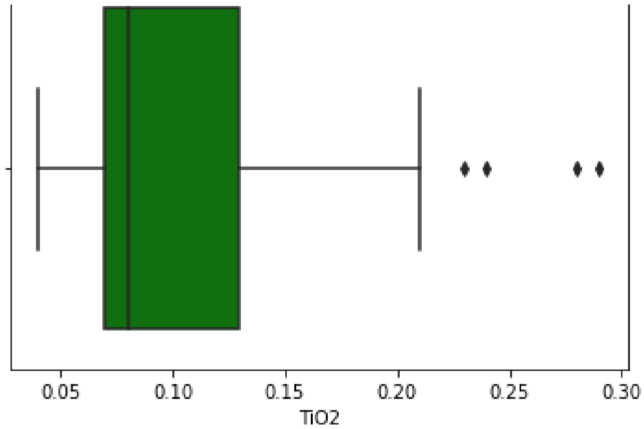


Boxplot of Cao

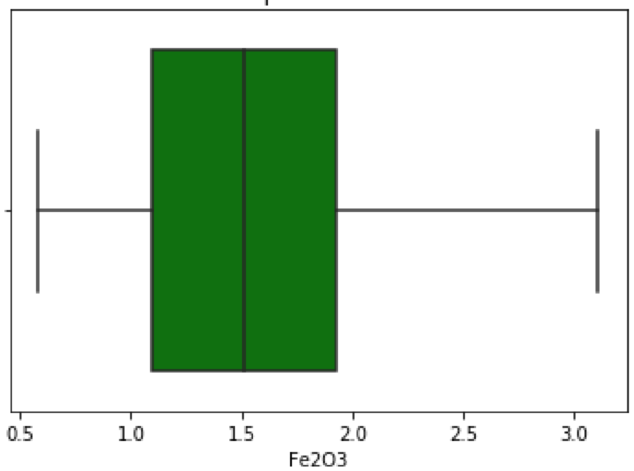


Boxplot of Tio2

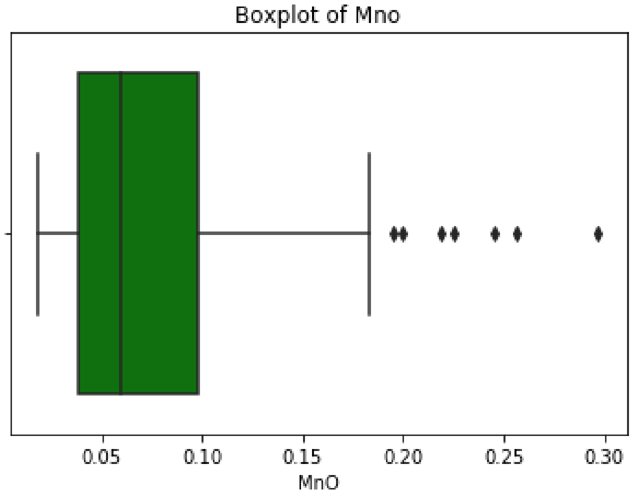




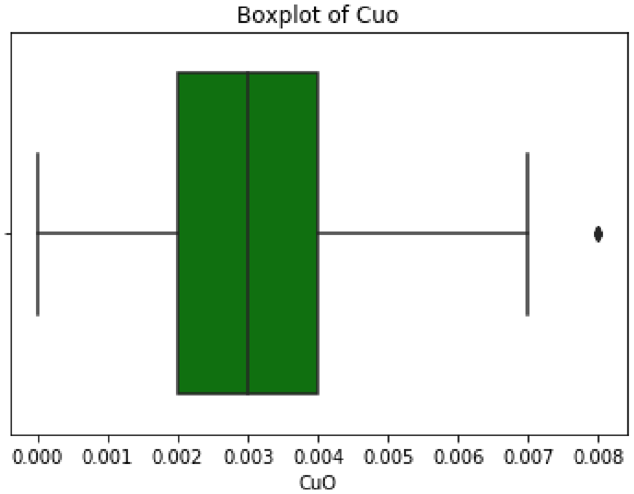
Boxplot of Fe2O3



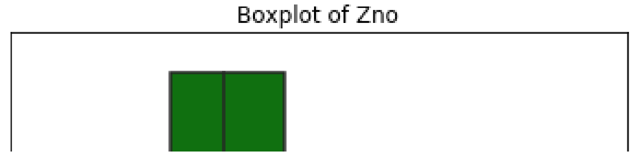
Boxplot of Mno

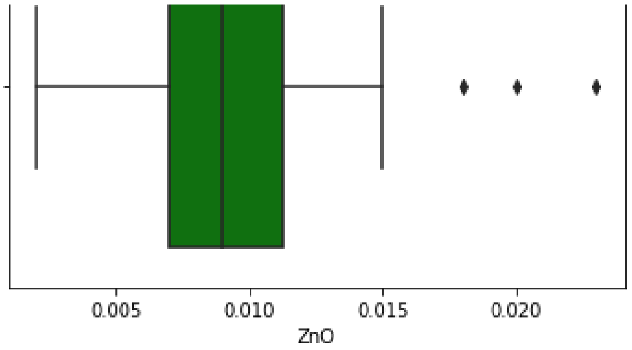


Boxplot of Cuo

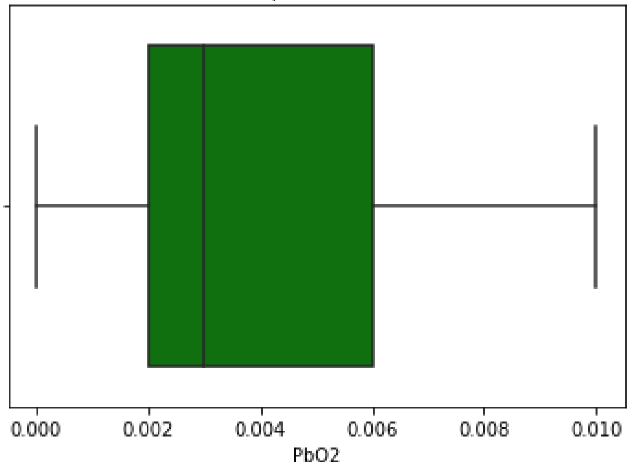


Boxplot of Zno

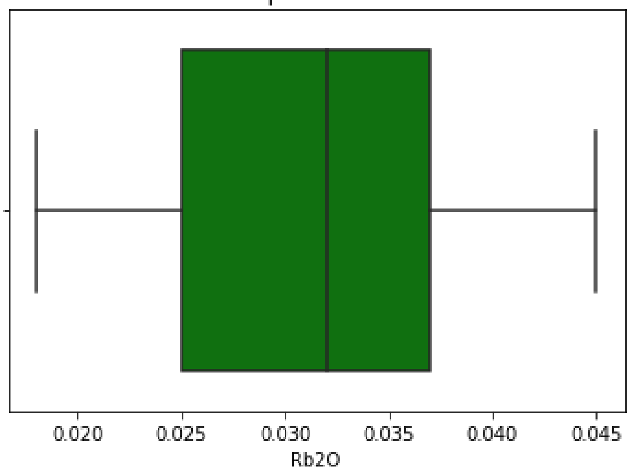




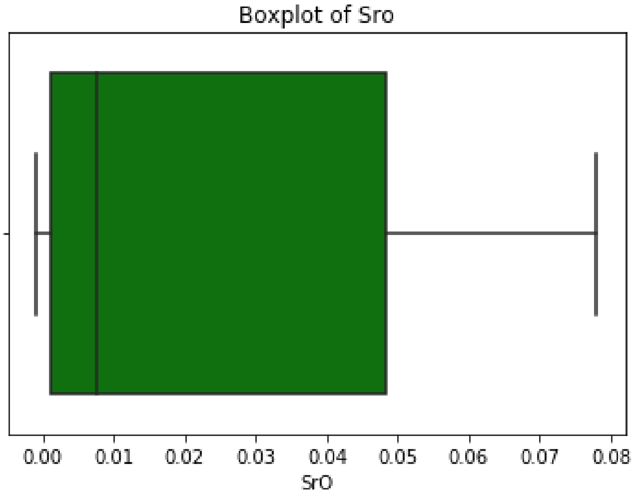
Boxplot of Pbo2



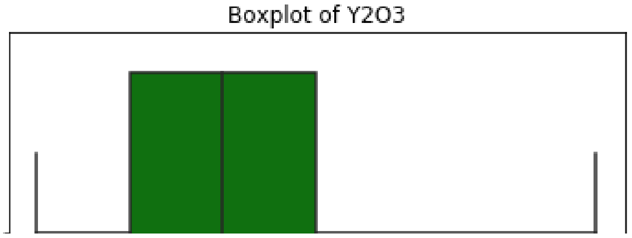
Boxplot of Rb2O

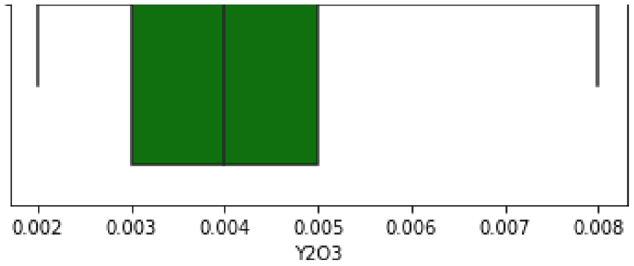


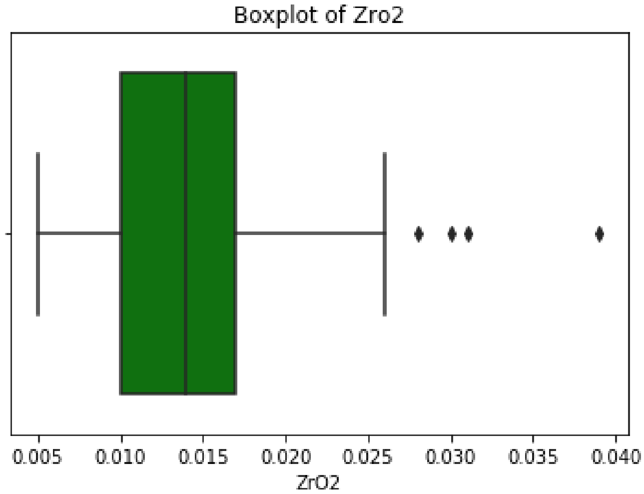
Boxplot of Sro

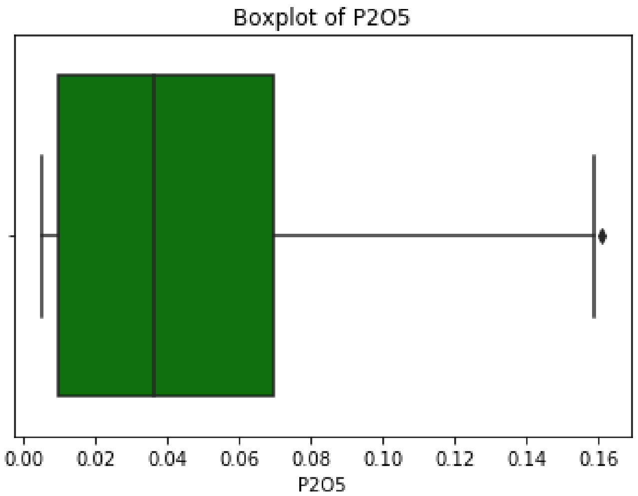


Boxplot of Y2O3









In [15]:

```
composition_b.head()
```

Out[15]:

	id	Ceramic Name	Part	Na2O	MgO	Al2O3	SiO2	K2O	CaO	TiO2	Fe2O3	MnO	CuO	ZnO	PbO2	Rb2O	SrO	Y2O3
0	1	FLQ-1-b	Body	0.620	0.380	19.610	71.990	4.840	0.310	0.070	1.180	0.063	0.001	0.007	0.001	0.043	0.000	0.004
1	2	FLQ-2-b	Body	0.570	0.470	21.190	70.090	4.980	0.490	0.090	1.120	0.038	0.002	0.008	0.004	0.043	-0.001	0.004
2	3	FLQ-3-b	Body	0.490	0.190	18.600	74.700	3.470	0.430	0.060	1.070	0.042	0.002	0.005	0.005	0.038	0.004	0.004
3	4	FLQ-4-b	Body	0.890	0.300	18.010	74.190	4.010	0.270	0.090	1.230	0.046	0.002	0.007	0.006	0.038	0.001	0.004
4	5	FLQ-5-b	Body	0.030	0.360	18.410	73.990	4.330	0.650	0.050	1.190	0.038	0.004	0.009	0.004	0.036	0.001	0.003

In [16]:

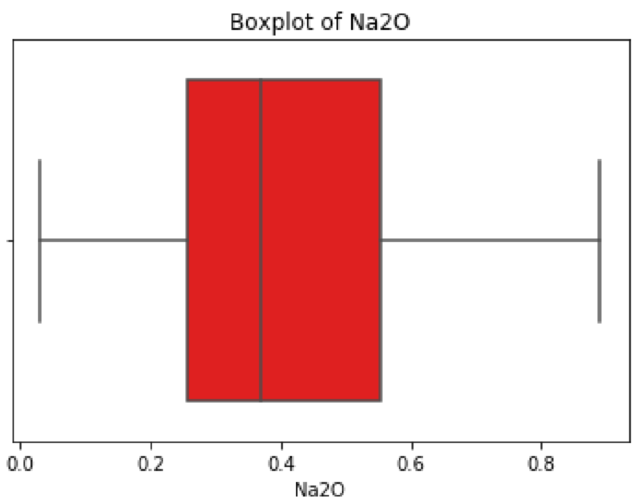
```
pd.set_option('display.float_format', lambda x: '%.3f' % x)
composition_b.describe().T
```

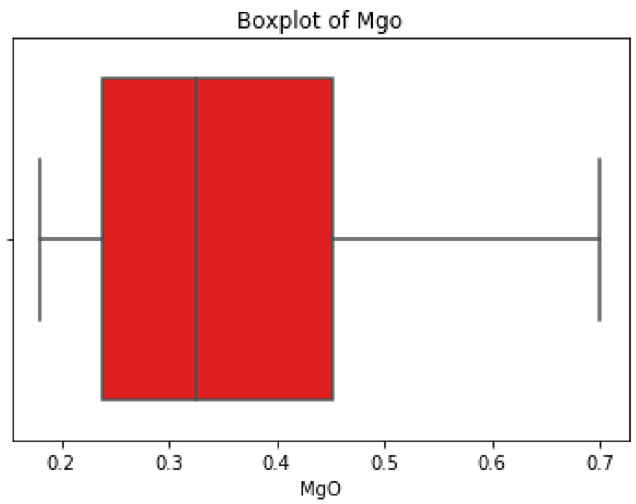
Out[16]:

	count	mean	std	min	25%	50%	75%	max
id	44.000	22.500	12.845	1.000	11.750	22.500	33.250	44.000
Na2O	44.000	0.397	0.211	0.030	0.258	0.370	0.552	0.890
MgO	44.000	0.342	0.125	0.180	0.237	0.325	0.453	0.700
Al2O3	44.000	21.812	2.301	17.650	19.590	21.835	23.310	26.480
SiO2	44.000	69.222	3.088	63.880	66.625	68.990	71.588	74.990
K2O	44.000	4.949	0.846	2.730	4.490	5.005	5.500	6.560
CaO	44.000	0.277	0.221	0.120	0.150	0.180	0.285	1.260
TiO2	44.000	0.122	0.063	0.040	0.070	0.105	0.152	0.290
Fe2O3	44.000	1.878	0.584	0.960	1.270	1.860	2.155	3.110
MnO	44.000	0.039	0.010	0.018	0.033	0.038	0.042	0.065
CuO	44.000	0.003	0.002	0.000	0.002	0.003	0.004	0.008
ZnO	44.000	0.009	0.002	0.004	0.008	0.009	0.011	0.012
PbO2	44.000	0.005	0.003	0.000	0.003	0.005	0.007	0.010
Rb2O	44.000	0.036	0.005	0.023	0.032	0.037	0.038	0.045
SrO	44.000	0.002	0.001	-0.001	0.001	0.001	0.002	0.007
Y2O3	44.000	0.005	0.001	0.003	0.004	0.005	0.006	0.008
ZrO2	44.000	0.017	0.007	0.007	0.013	0.017	0.020	0.039
P2O5	44.000	0.011	0.007	0.005	0.007	0.009	0.014	0.044

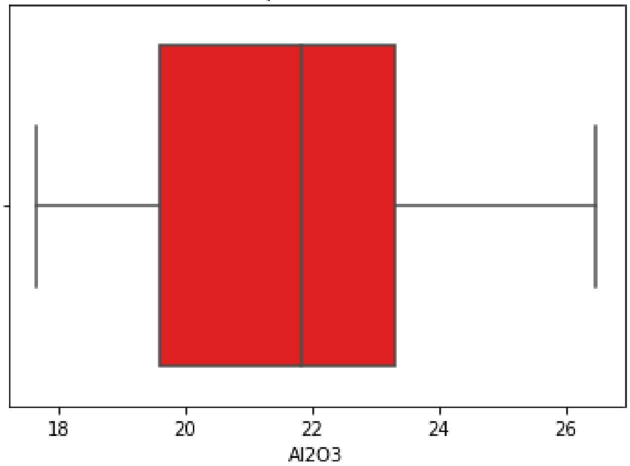
In [17]:

```
for column in composition[list]:
    fig, ax = plt.subplots()
    sns.boxplot(x=composition_b[column], ax=ax, color="red");
    plt.title(f'Boxplot of {column.title().replace("_", " ")}')
```

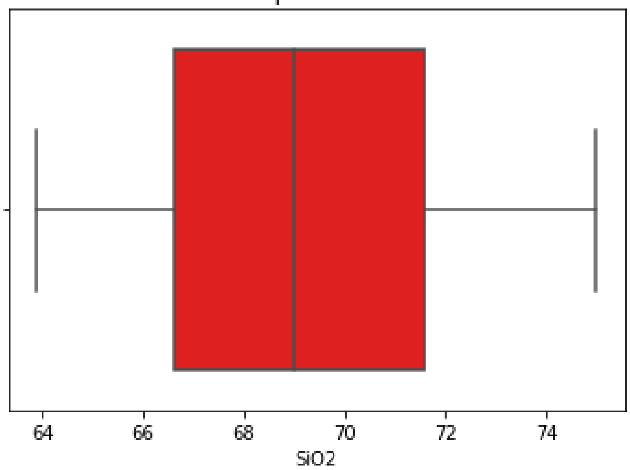




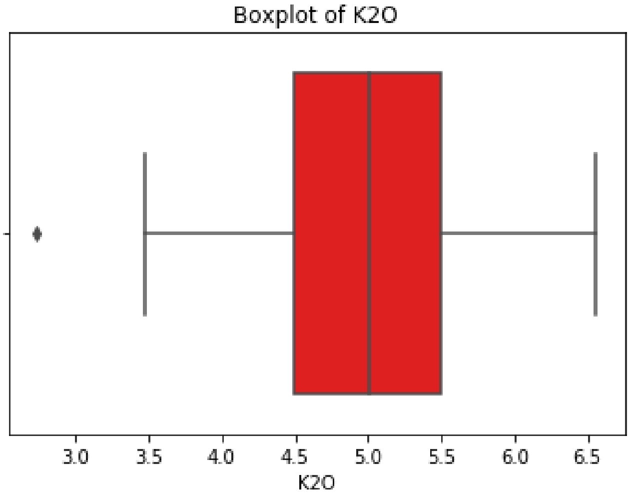
Boxplot of Al2O3



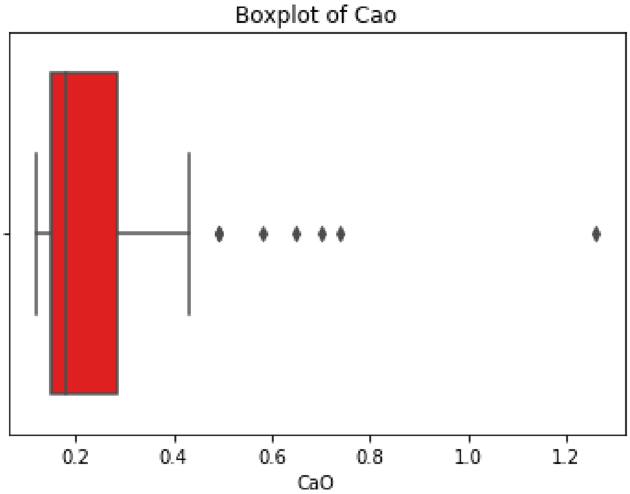
Boxplot of Sio2



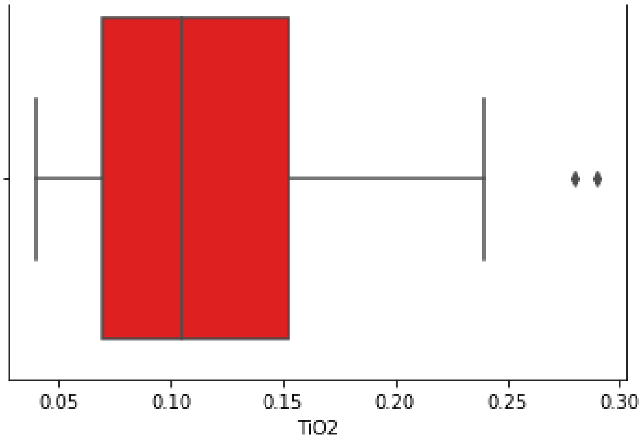
Boxplot of K2O



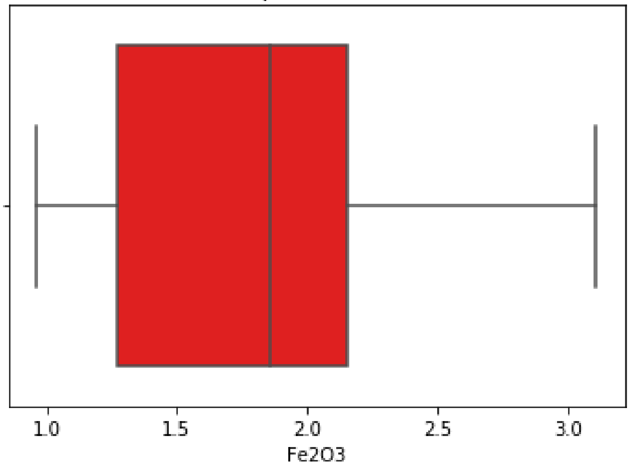
Boxplot of Cao



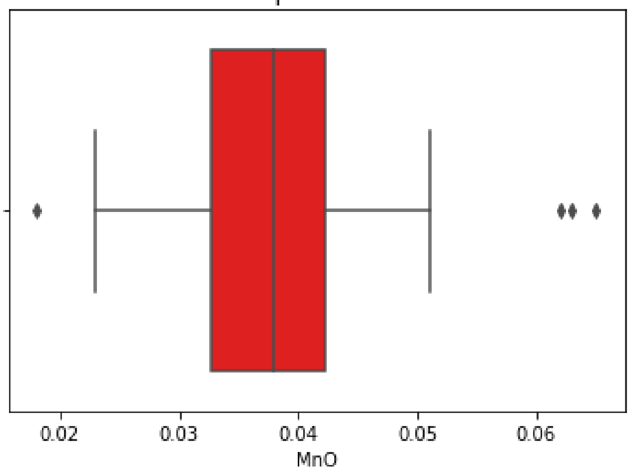
Boxplot of Tio2



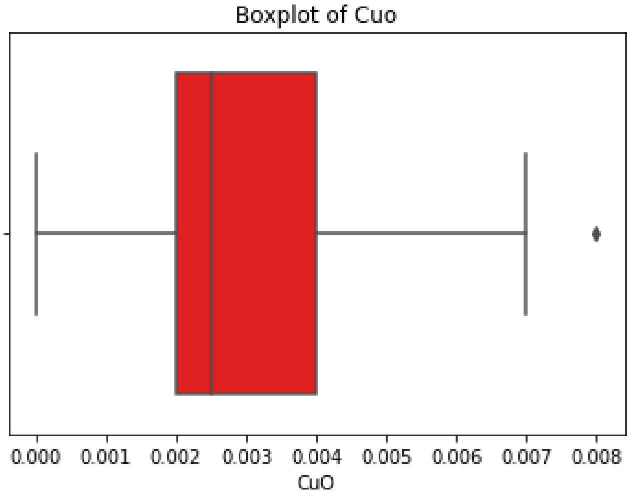
Boxplot of Fe2O3



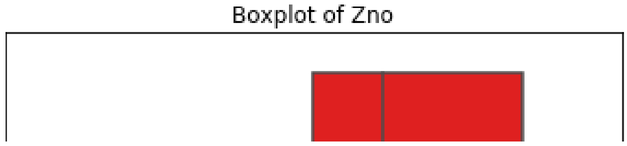
Boxplot of Mno

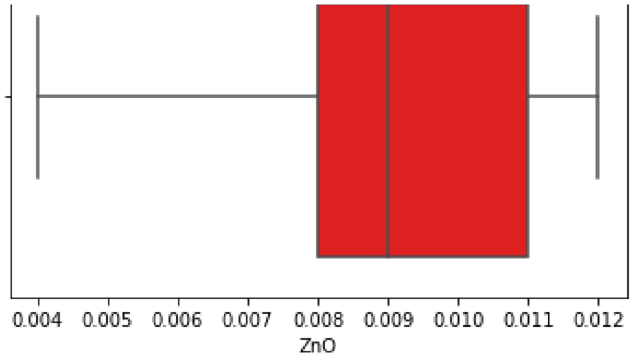


Boxplot of Cuo

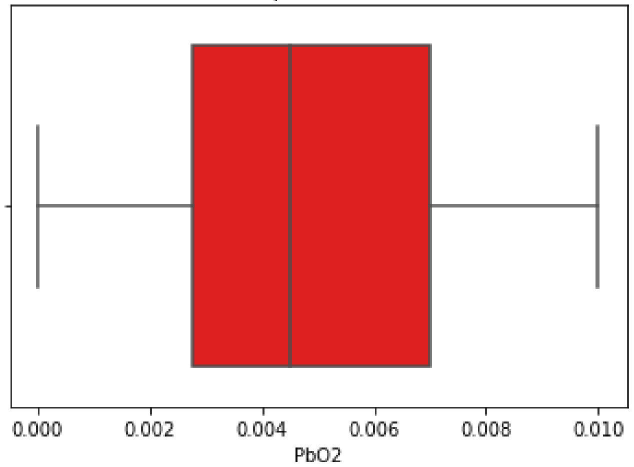


Boxplot of Zno

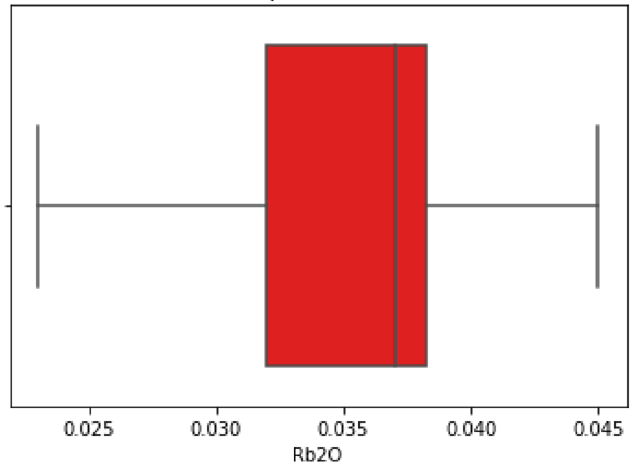




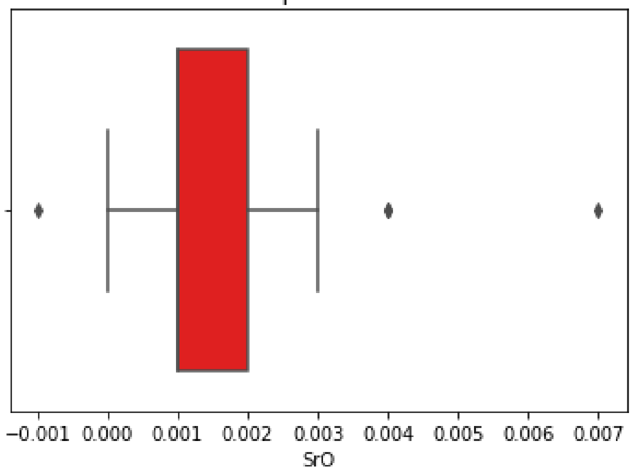
Boxplot of Pbo2



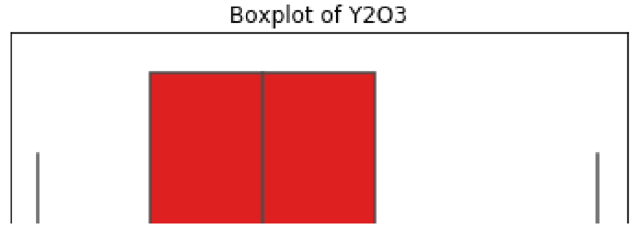
Boxplot of Rb2O

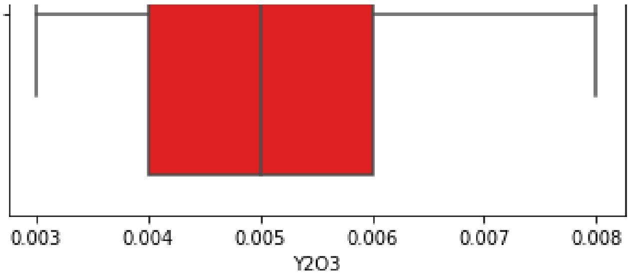


Boxplot of Sro

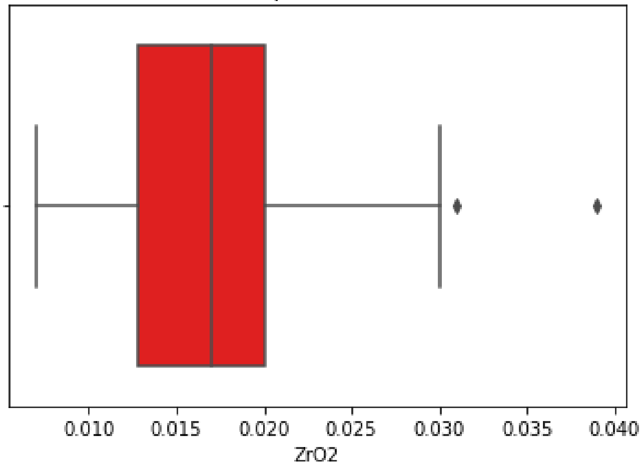


Boxplot of Y2O3

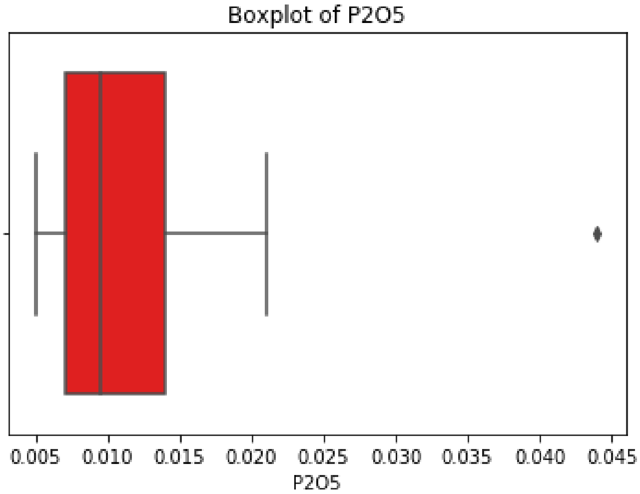




Boxplot of Zro2



Boxplot of P2O5



In [18]:

```
composition_g.head()
```

Out[18]:

	id	Ceramic Name	Part	Na2O	MgO	Al2O3	SiO2	K2O	CaO	TiO2	Fe2O3	MnO	CuO	ZnO	PbO2	Rb2O	SrO	Y2
44	1	FLQ-1-g	Glaze	0.970	0.070	11.420	74.410	5.700	5.340	0.050	1.040	0.055	0.002	0.006	0.002	0.031	0.015	0.
45	2	FLQ-2-g	Glaze	1.460	0.470	12.960	68.790	4.850	8.880	0.110	1.490	0.095	0.003	0.004	0.000	0.035	0.025	0.
46	3	FLQ-3-g	Glaze	1.050	0.230	13.640	69.900	4.460	8.430	0.070	1.220	0.059	0.002	0.009	0.004	0.037	0.009	0.
47	4	FLQ-4-g	Glaze	0.140	0.410	12.420	67.240	4.290	12.860	0.060	1.580	0.096	0.008	0.007	0.004	0.033	0.016	0.
48	5	FLQ-5-g	Glaze	0.370	1.030	13.150	68.980	5.580	7.910	0.080	1.900	0.080	0.006	0.012	0.000	0.032	0.008	0.

In [19]:

```
pd.set_option('display.float_format', lambda x: '%.3f' % x)
composition_g.describe().T
```

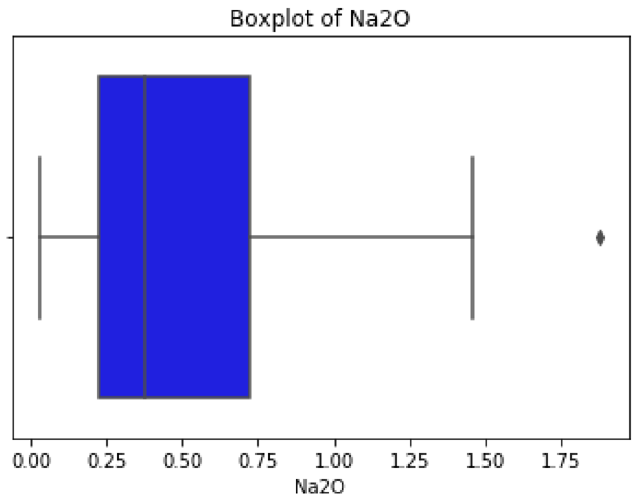
Out[19]:

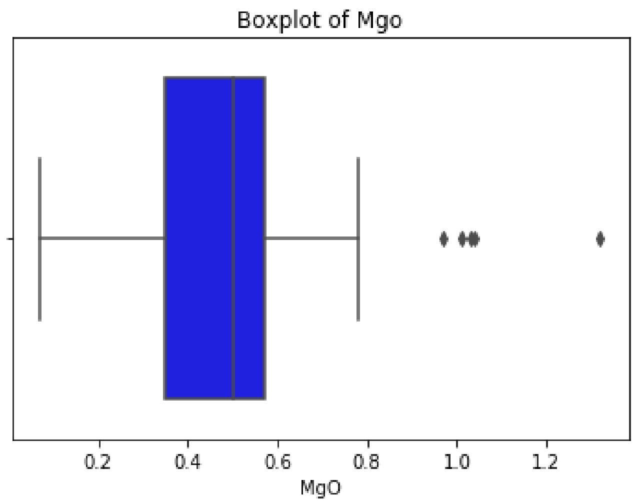
	count	mean	std	min	25%	50%	75%	max

id	count	mean	std	min	25%	50%	75%	max
	44.000	22.500	12.845	1.000	11.750	22.500	33.250	44.000
Na2O	44.000	0.546	0.436	0.030	0.223	0.375	0.723	1.880
MgO	44.000	0.518	0.249	0.070	0.350	0.500	0.573	1.320
Al2O3	44.000	13.110	0.849	11.300	12.707	13.005	13.715	14.760
SiO2	44.000	70.428	2.251	65.530	68.755	70.650	71.840	75.950
K2O	44.000	5.008	0.920	2.980	4.285	5.105	5.720	6.740
CaO	44.000	8.066	2.534	4.120	6.090	7.915	9.503	13.690
TiO2	44.000	0.080	0.031	0.040	0.060	0.070	0.090	0.190
Fe2O3	44.000	1.245	0.439	0.580	0.948	1.180	1.580	2.610
MnO	44.000	0.125	0.061	0.052	0.079	0.101	0.163	0.297
CuO	44.000	0.003	0.002	0.000	0.002	0.003	0.004	0.008
ZnO	44.000	0.010	0.004	0.002	0.007	0.009	0.012	0.023
PbO2	44.000	0.003	0.002	0.000	0.002	0.003	0.004	0.010
Rb2O	44.000	0.027	0.006	0.018	0.022	0.025	0.031	0.040
SrO	44.000	0.044	0.020	0.008	0.024	0.049	0.061	0.078
Y2O3	44.000	0.004	0.001	0.002	0.003	0.004	0.004	0.005
ZrO2	44.000	0.012	0.003	0.005	0.010	0.012	0.014	0.017
P2O5	44.000	0.077	0.032	0.036	0.051	0.071	0.097	0.161

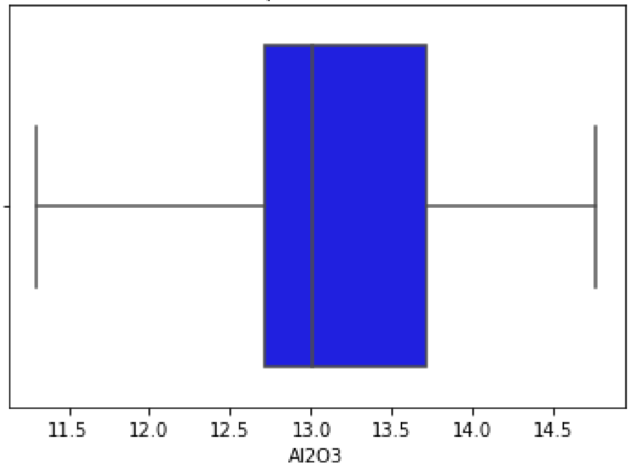
In [20]:

```
for column in composition[list]:
    fig, ax = plt.subplots()
    sns.boxplot(x=composition_g[column], ax=ax, color="blue");
    plt.title(f'Boxplot of {column.title().replace("_", " ")}')
```

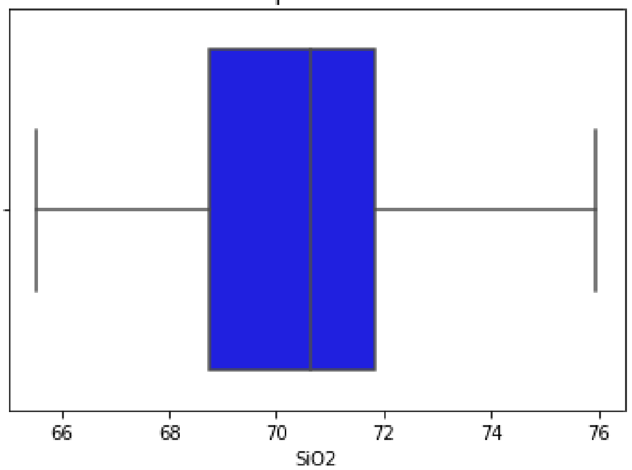




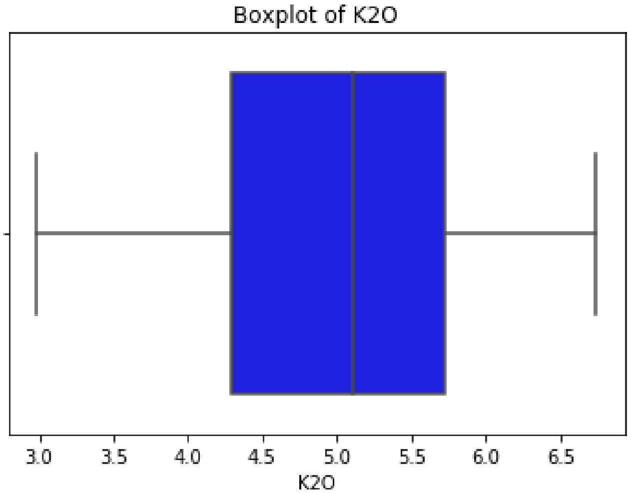
Boxplot of Al2O3



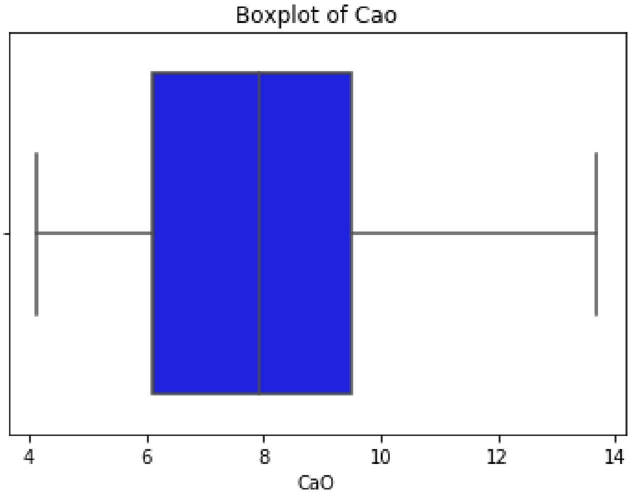
Boxplot of Sio2



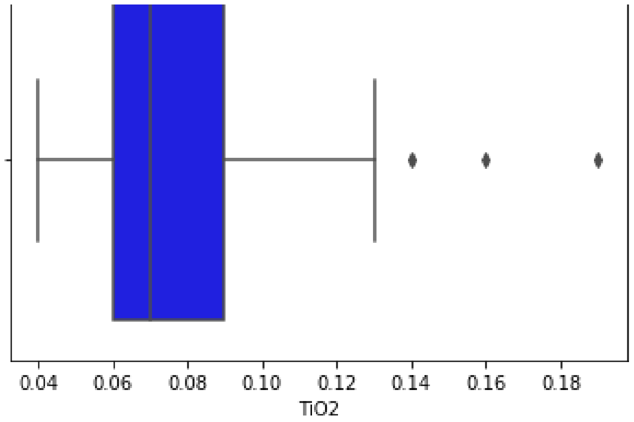
Boxplot of K2O



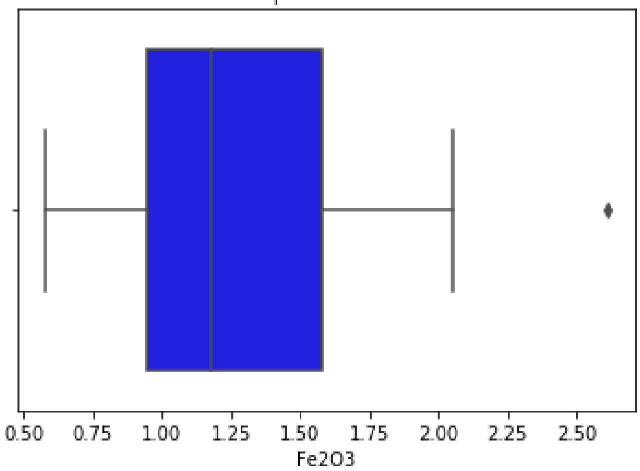
Boxplot of Cao



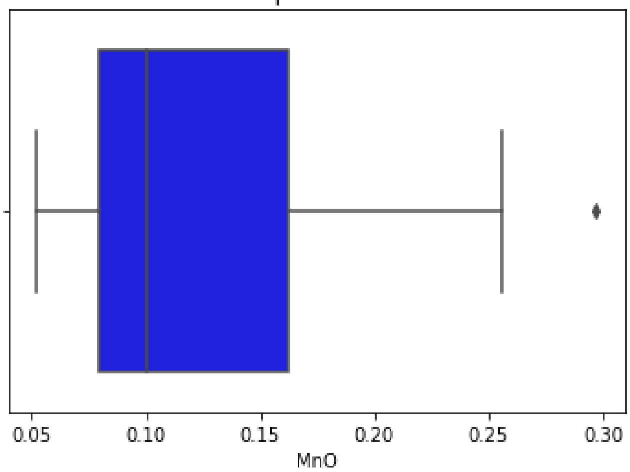
Boxplot of Tio2



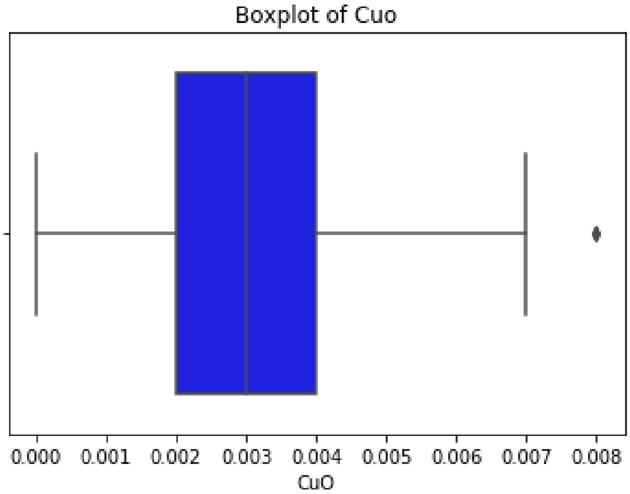
Boxplot of Fe2O3



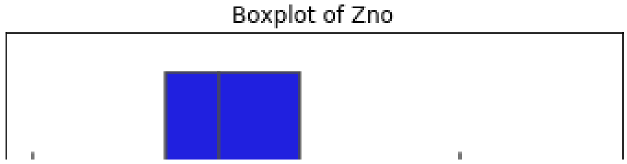
Boxplot of Mno

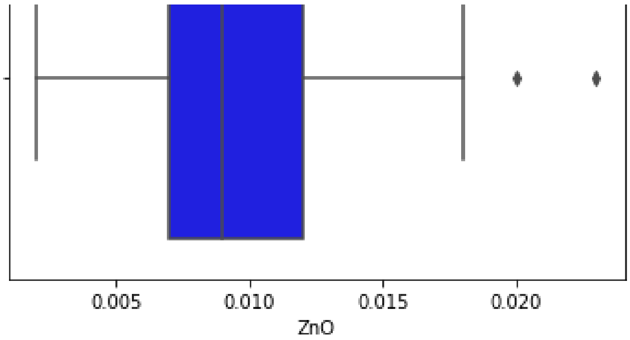


Boxplot of Cuo

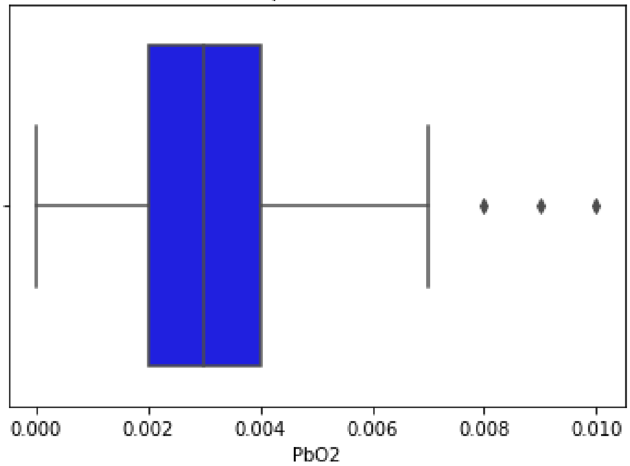


Boxplot of Zno

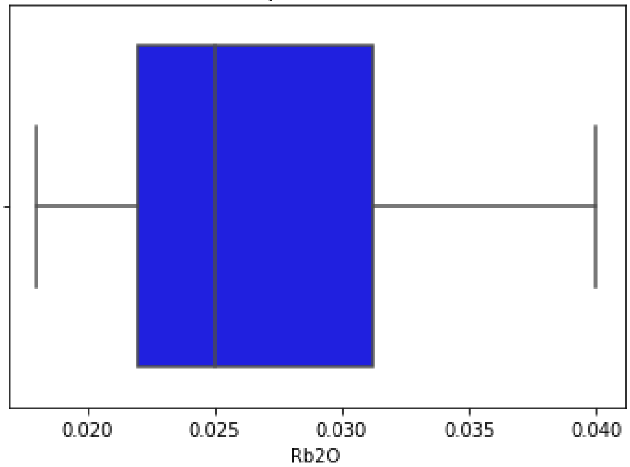




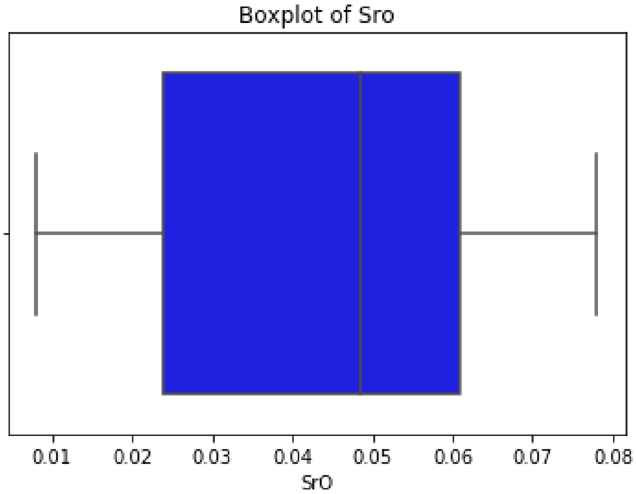
Boxplot of Pbo2



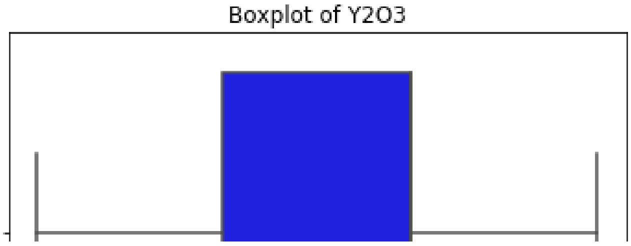
Boxplot of Rb2O

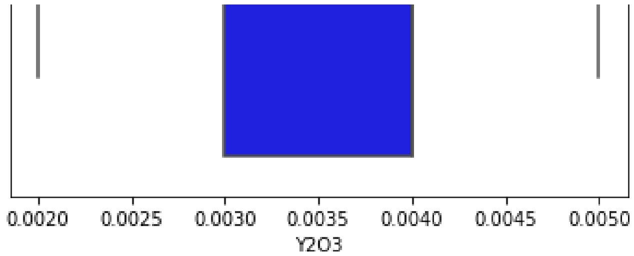


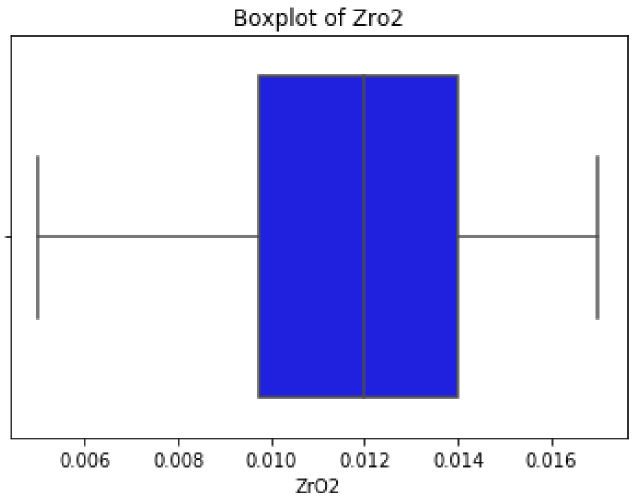
Boxplot of Sro

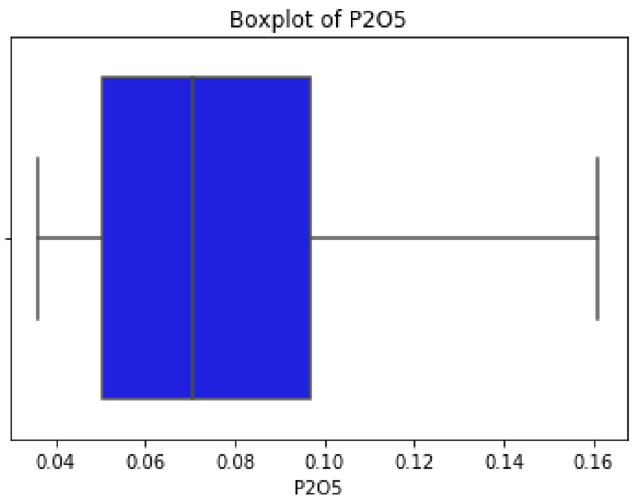


Boxplot of Y2O3





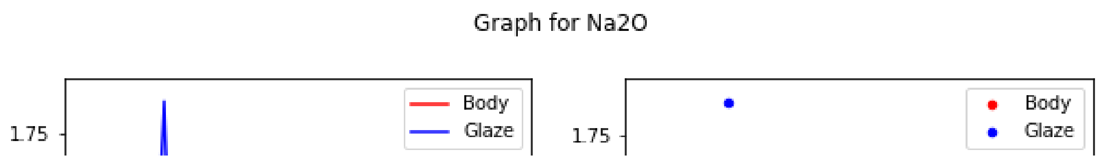


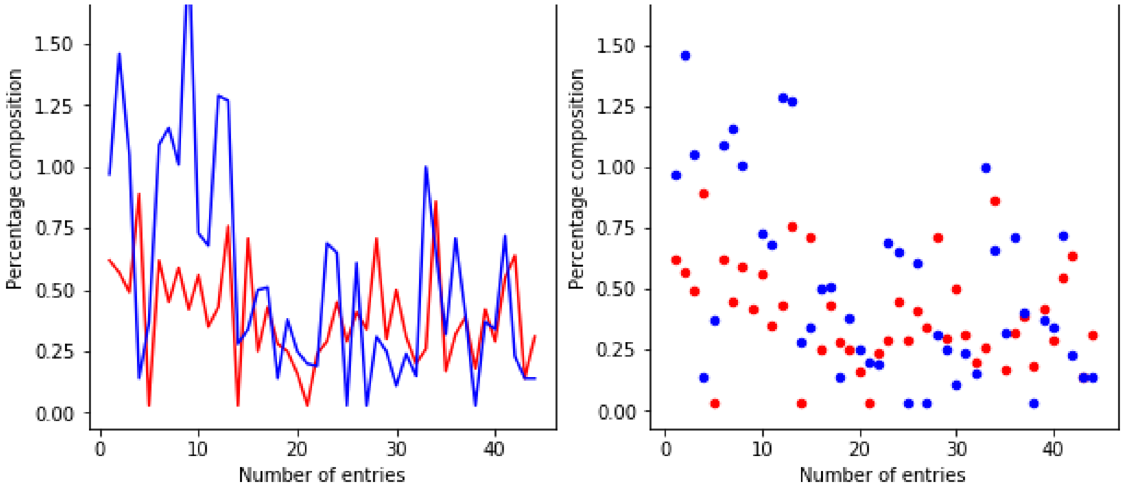


Now i will plot all components of body and glaze so as to get a birds eye view of the varying composition for ceramics

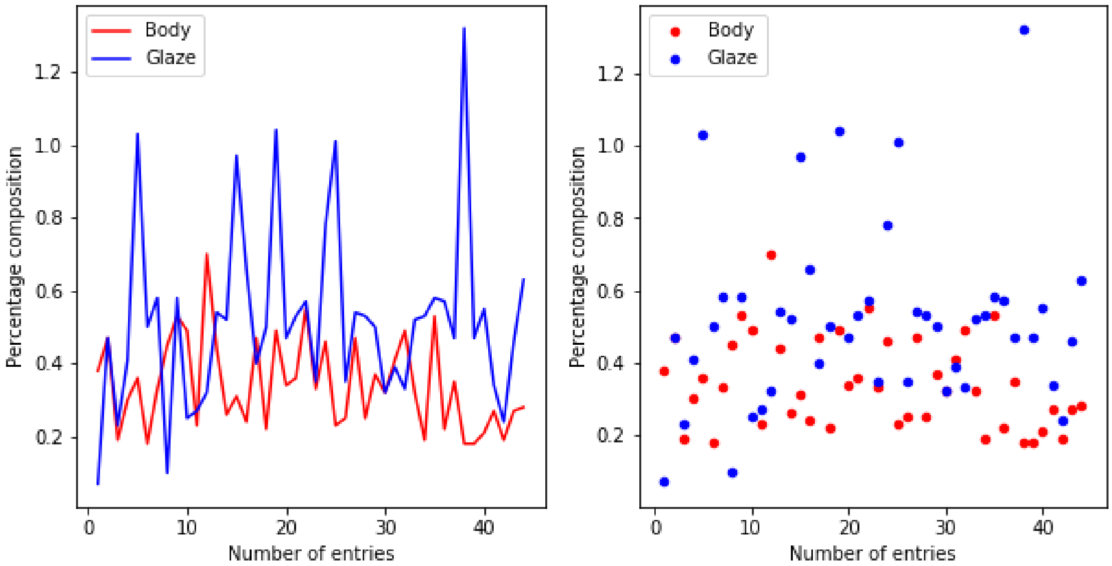
In [21]:

```
for column in composition_b[list]:
    figure, axes = plt.subplots(1, 2,figsize=(10,5))
    ax=composition_b.plot(kind='line',x='id',y=[column],color='red',legend=False,ax=axes
[0])
    composition_g.plot(kind='line',x='id',y=[column],color='blue',ax=ax,legend=False)
    ax.legend(["Body", "Glaze"]);
    ax.set_xlabel("Number of entries")
    ax.set_ylabel("Percentage composition")
    ax=composition_b.plot(kind='scatter',x='id',y=[column],color='red',legend=False,ax=a
xes[1])
    composition_g.plot(kind='scatter',x='id',y=[column],color='blue',ax=ax,legend=False)
    ax.legend(["Body", "Glaze"]);
    ax.set_xlabel("Number of entries")
    ax.set_ylabel("Percentage composition")
    p=f'Graph for {column.title().replace("_", " ")}'
    plt.suptitle(p)
```

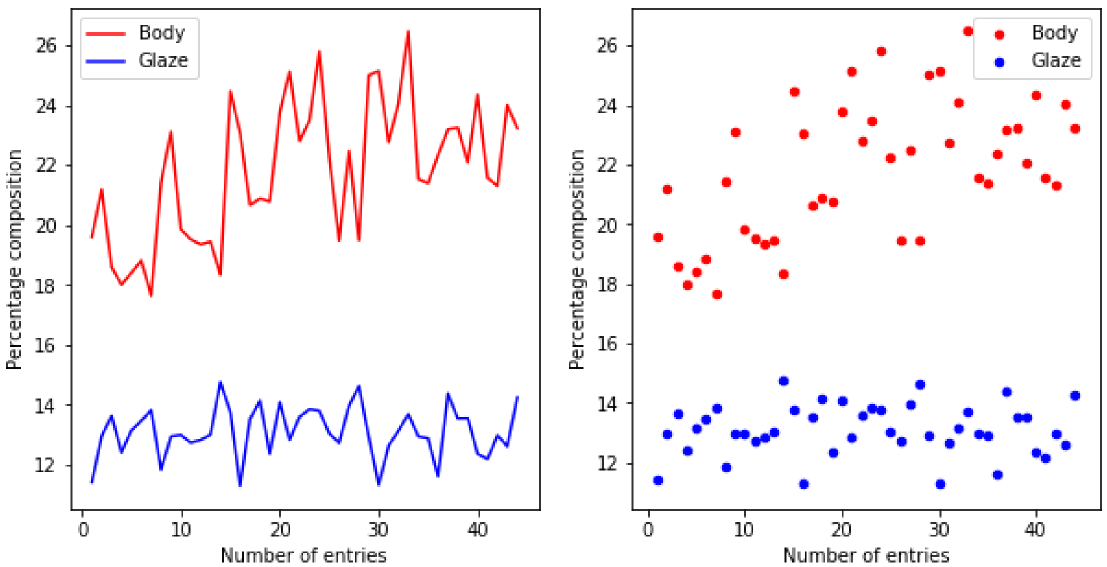




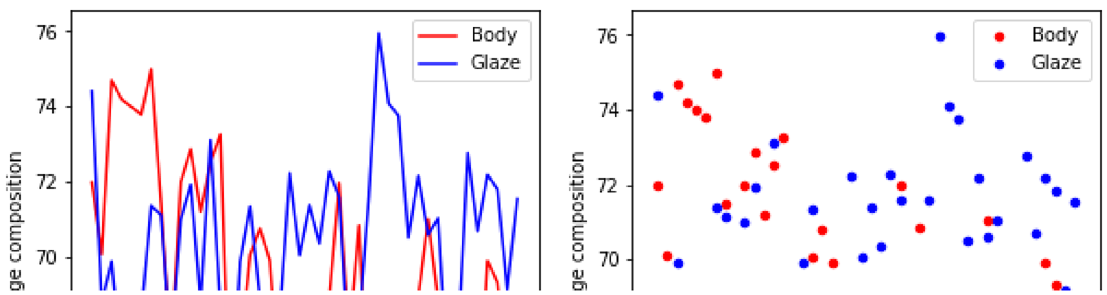
Graph for Mgo

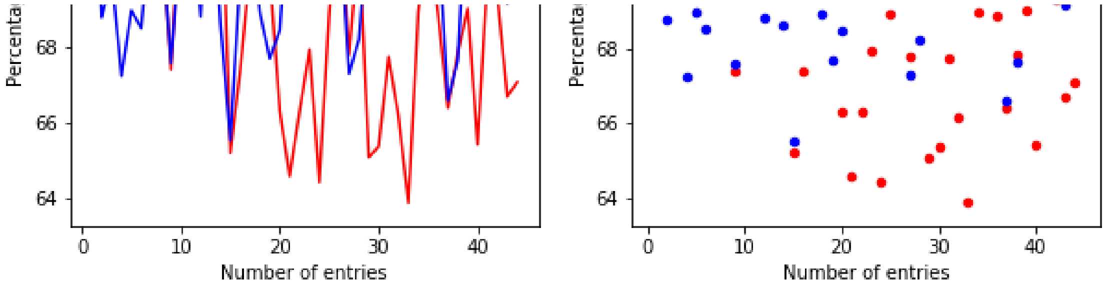


Graph for Al2O3

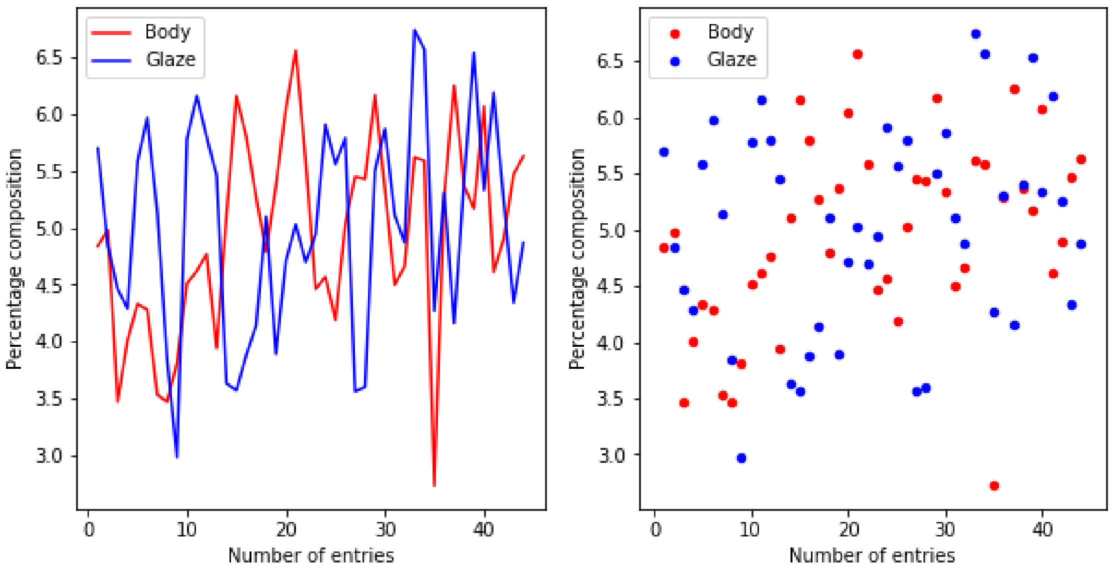


Graph for Sio2

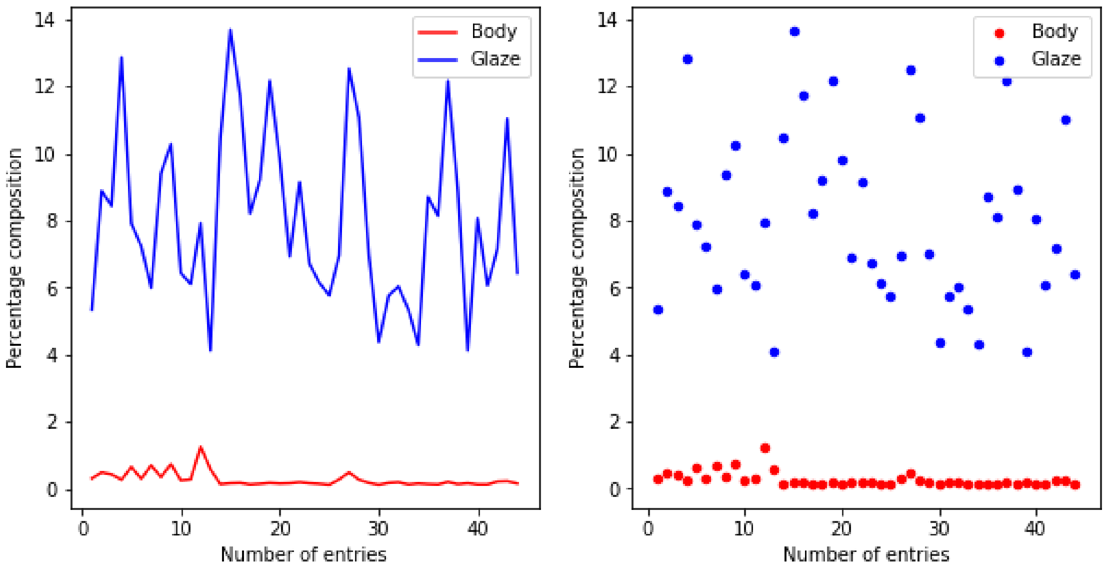




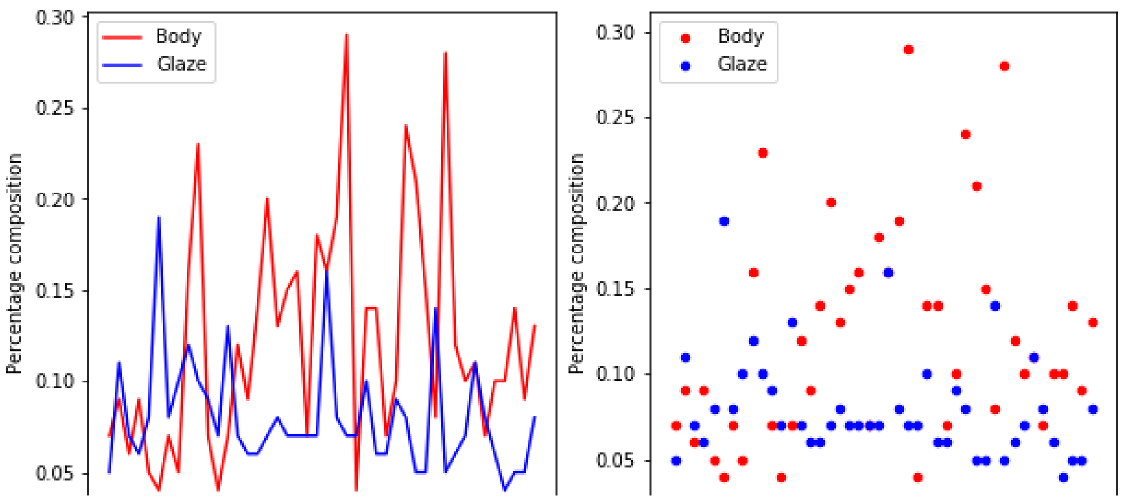
Graph for K2O



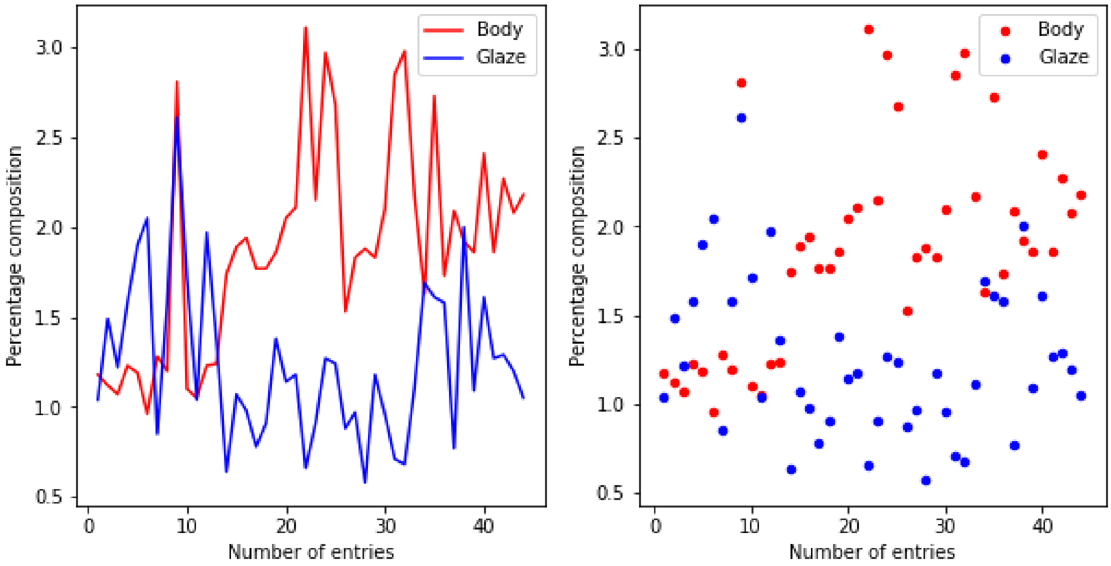
Graph for Cao



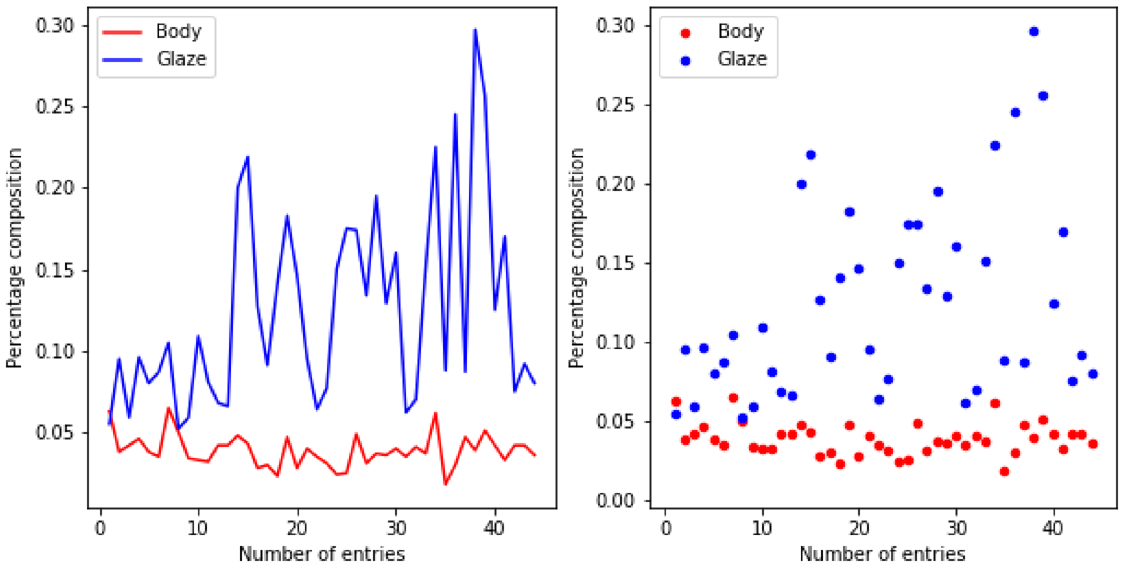
Graph for Tio2



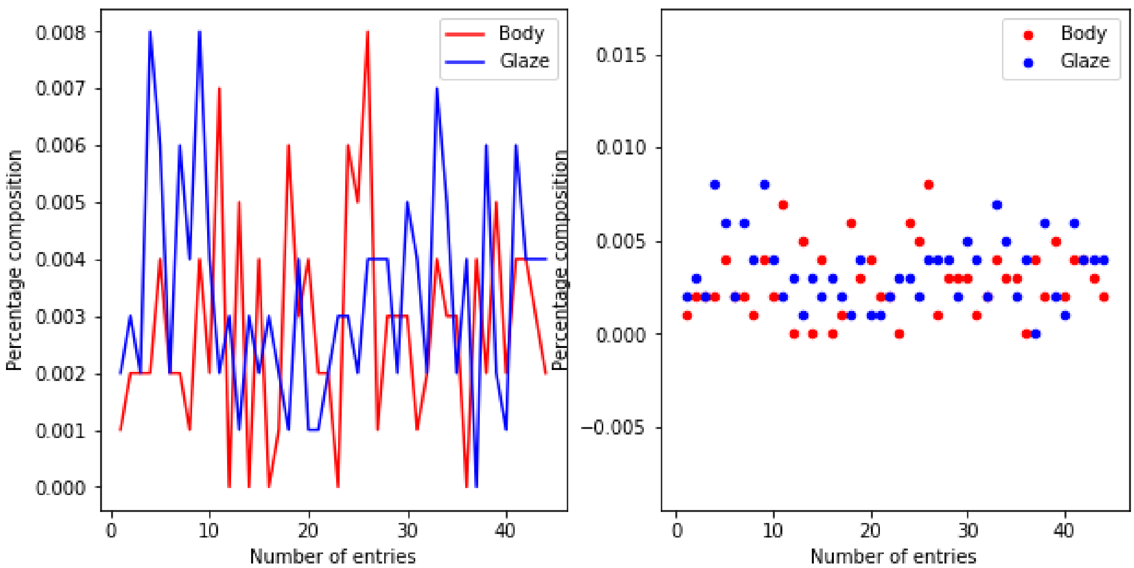
Graph for Fe2O3



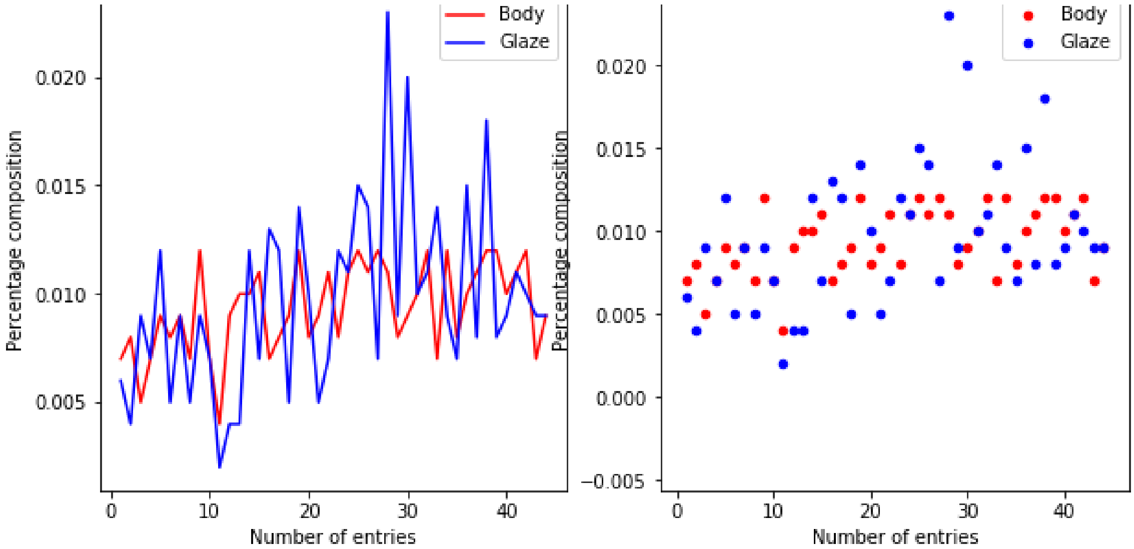
Graph for Mno



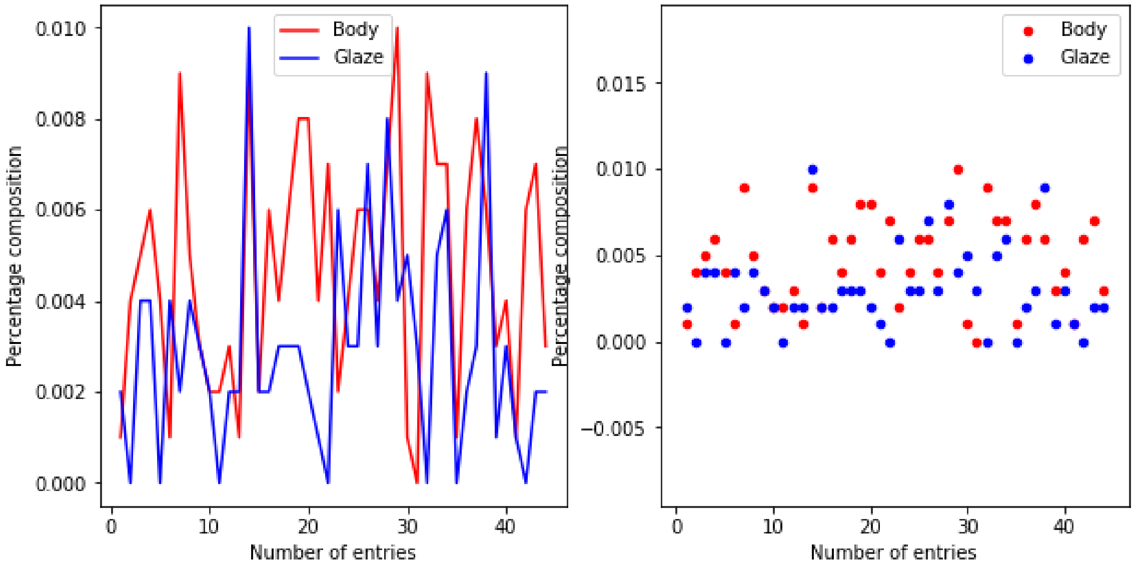
Graph for Cuo



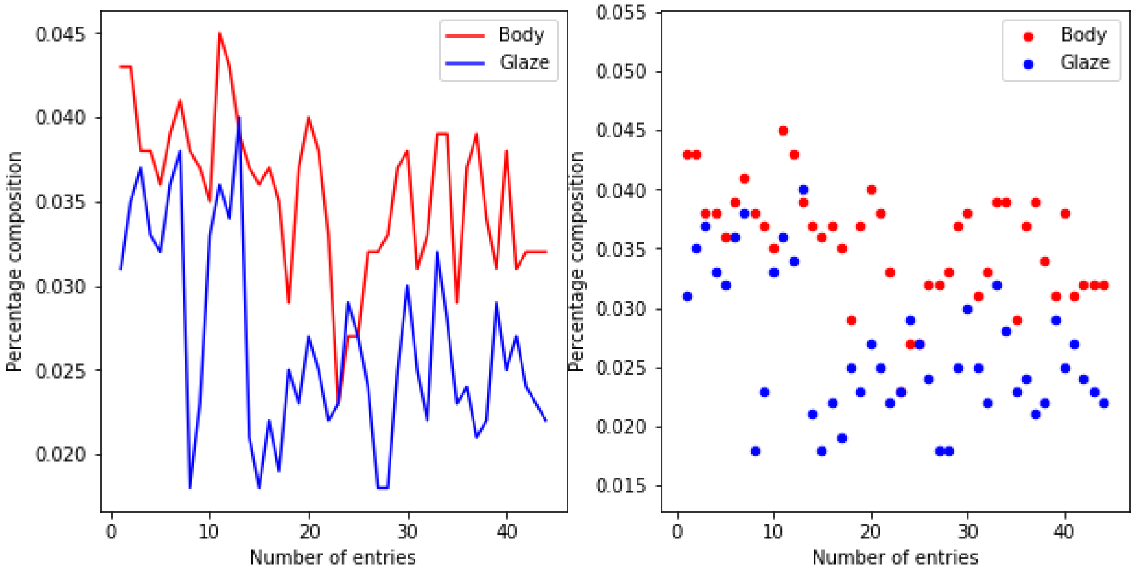
Graph for Zno



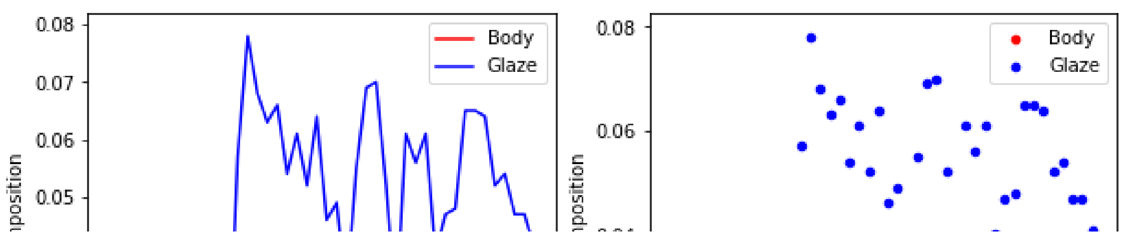
Graph for Pbo2

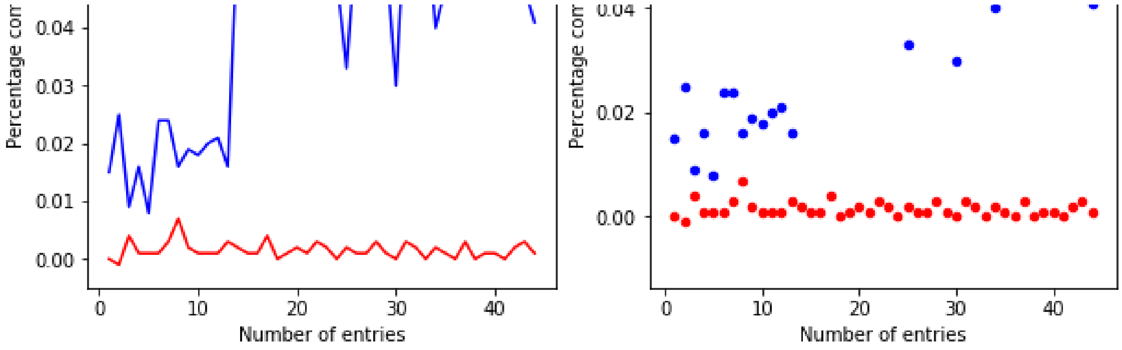


Graph for Rb2O

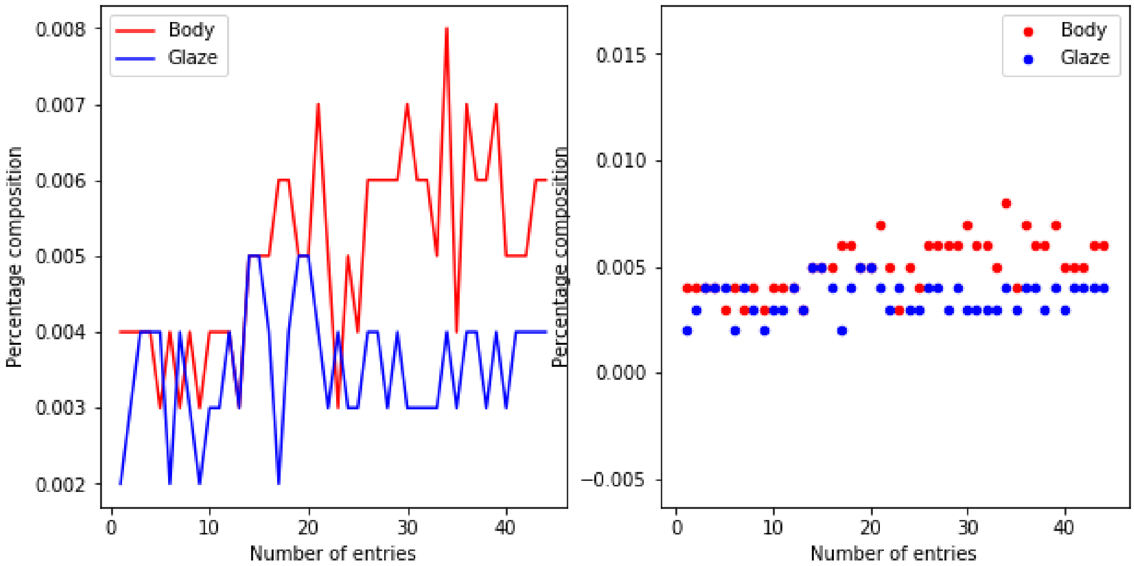


Graph for Sro

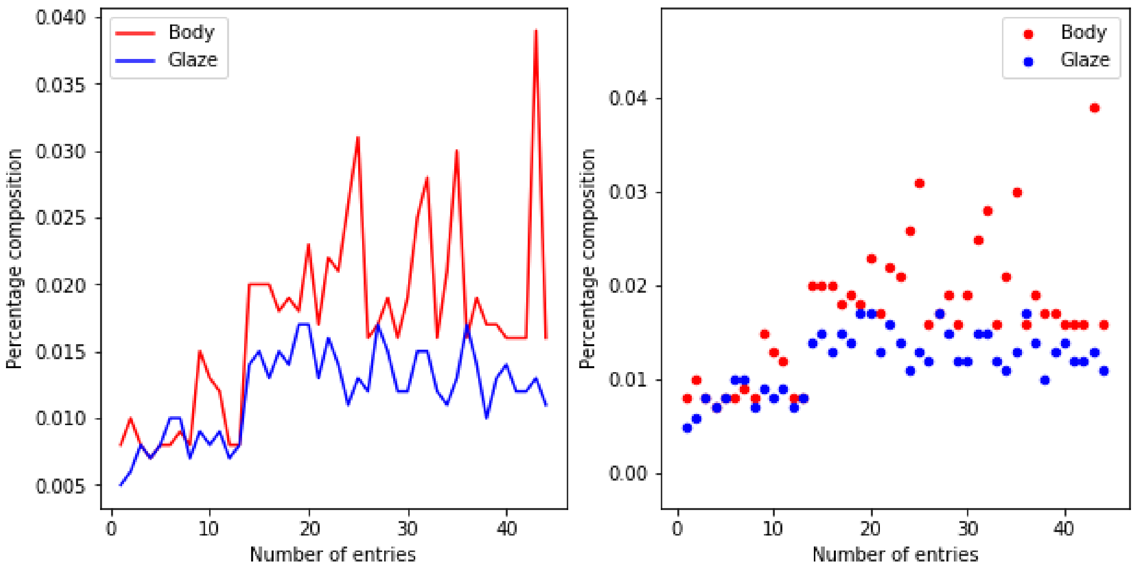




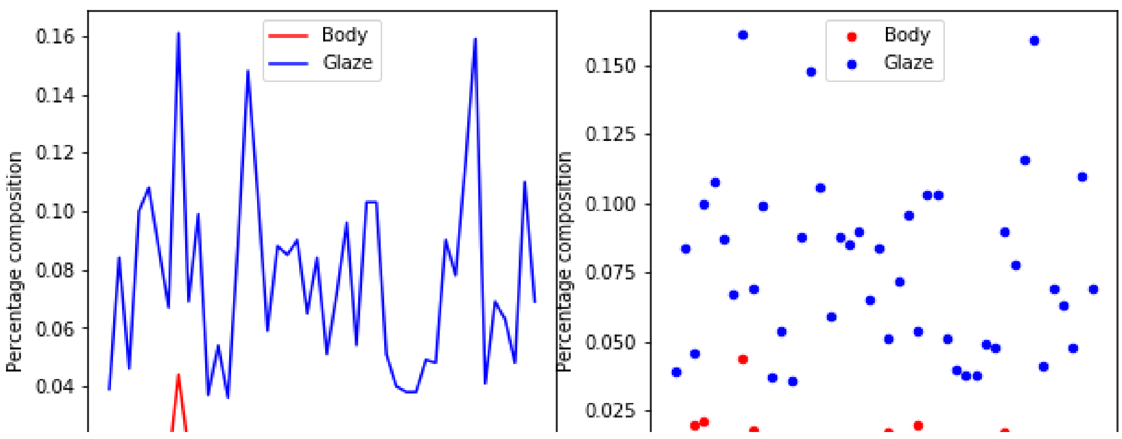
Graph for Y2O3

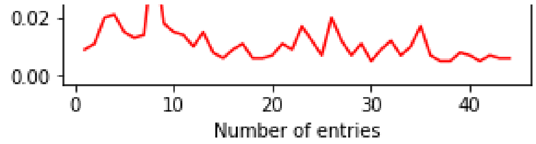


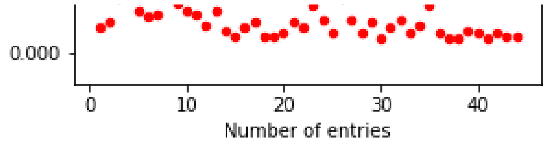
Graph for Zro2



Graph for P2O5







In [22]:

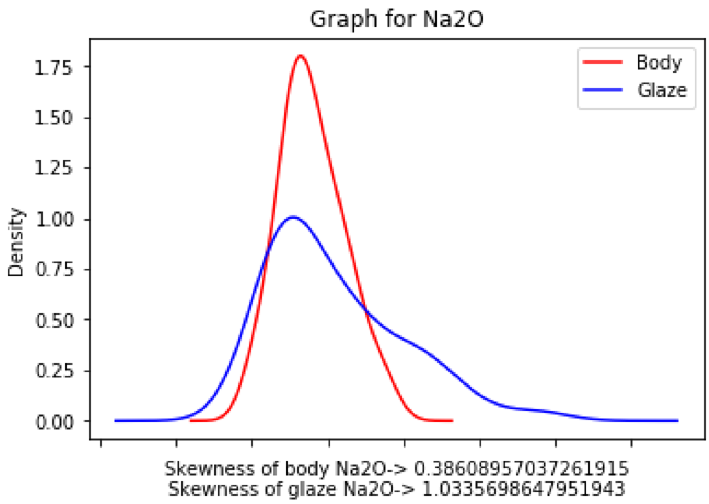
```
composition[list].skew()
```

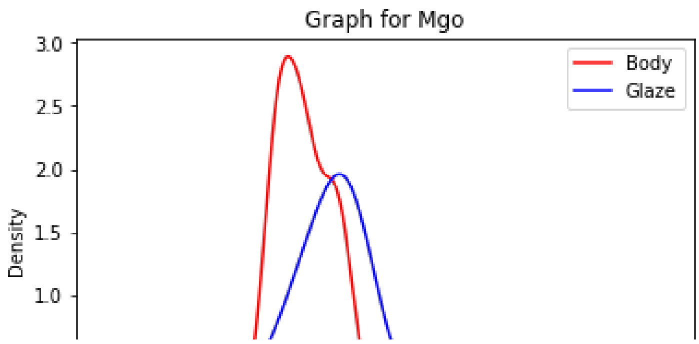
Out[22]:

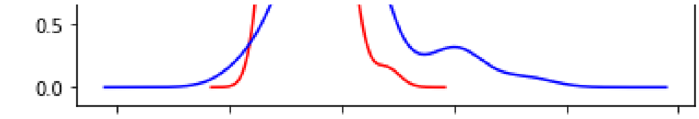
```
Na2O      1.439
MgO       1.561
Al2O3     0.289
SiO2     -0.079
K2O      -0.269
CaO       0.517
TiO2      1.512
Fe2O3     0.636
MnO       1.493
CuO       0.627
ZnO       0.965
PbO2      0.512
Rb2O     -0.161
SrO       0.674
Y2O3      0.549
ZrO2      1.249
P2O5      0.962
dtype: float64
```

In [23]:

```
for column in composition_b[list]:
    ax=composition_b.plot(kind='density',x='id',y=[column],color='red',legend=False)
    composition_g.plot(kind='density',x='id',y=[column],color='blue',ax=ax,legend=False)
    ax.legend(["Body", "Glaze"]);
    ax.set_xticklabels([])
    p=f'Graph for {column.title().replace("_", " ")}'
    plt.title(p)
    ax.set_xlabel(f'Skewness of body {column.title().replace("_", " ")}-> '+ str(skew(com
position_b[column]))+"\n"+f'Skewness of glaze {column.title().replace("_", " ")}-> '+ str
(skew(composition_g[column])))
```

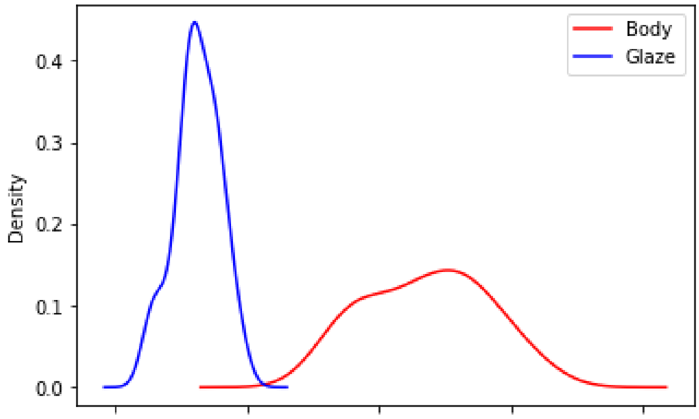






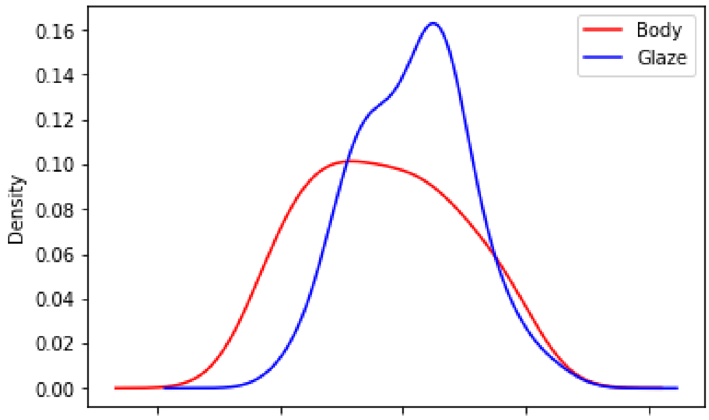
Skewness of body Mgo-> 0.6204176074995184
Skewness of glaze Mgo-> 1.1267689041221824

Graph for Al2O3



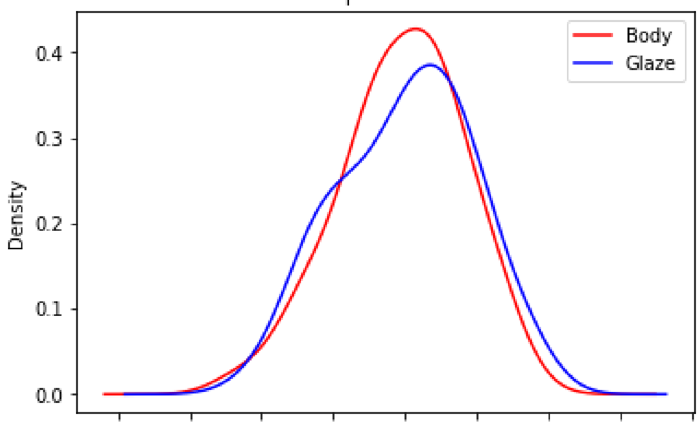
Skewness of body Al2O3-> 0.0009757822470763805
Skewness of glaze Al2O3-> -0.2949691761988996

Graph for Sio2



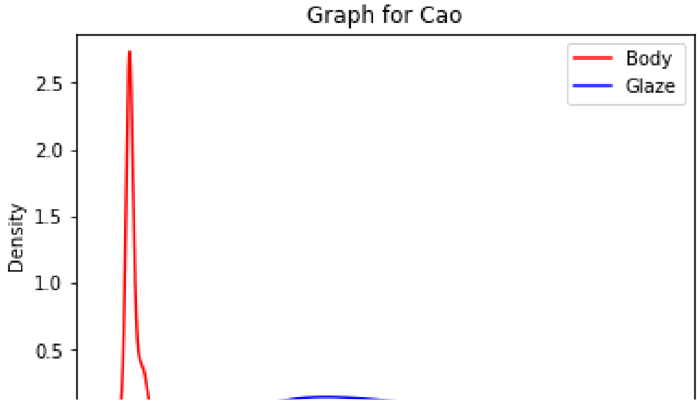
Skewness of body Sio2-> 0.13725205921909403
Skewness of glaze Sio2-> 0.07361597811273572

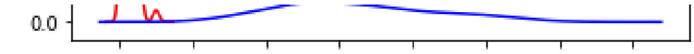
Graph for K2O



Skewness of body K2O-> -0.3821417769973489
Skewness of glaze K2O-> -0.18772712075121506

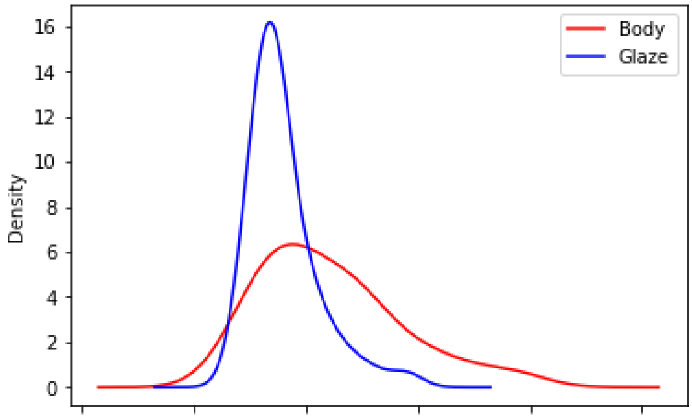
Graph for Cao





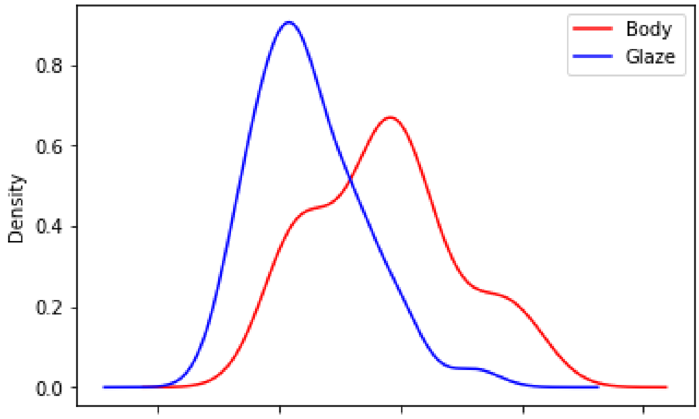
Skewness of body Cao-> 2.554423407480873
Skewness of glaze Cao-> 0.4388154667471102

Graph for Tio2



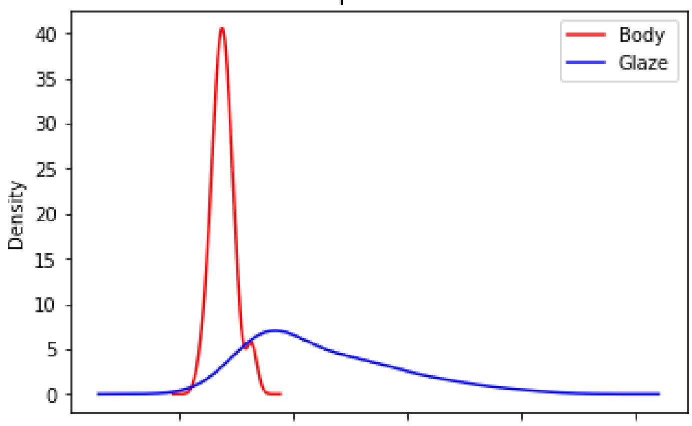
Skewness of body Tio2-> 0.9033527672514025
Skewness of glaze Tio2-> 1.6461378558498616

Graph for Fe2O3



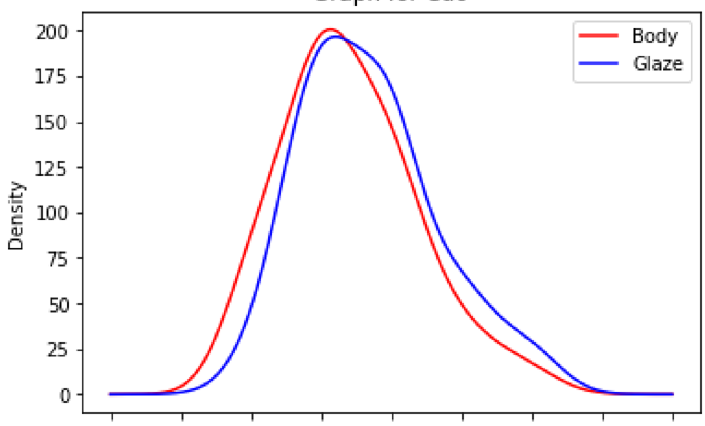
Skewness of body Fe2O3-> 0.3390626124539278
Skewness of glaze Fe2O3-> 0.8389495510322743

Graph for Mno



Skewness of body Mno-> 0.59982165790151
Skewness of glaze Mno-> 0.9401259857826482

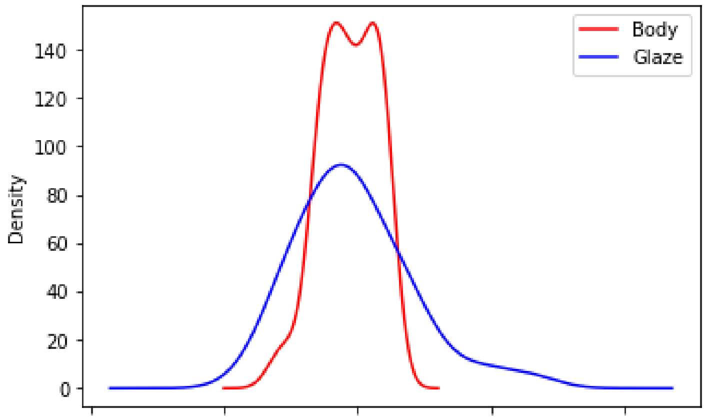
Graph for Cuo



Skewness of body Cuo-> 0.5957816295010149

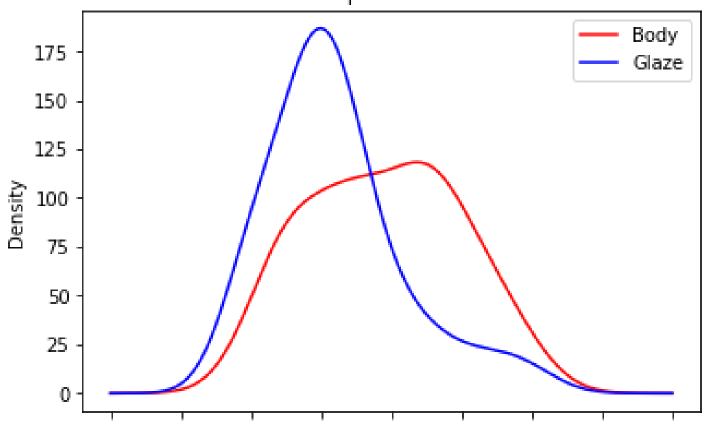
Skewness of glaze Cuo-> 0.6712075194839627

Graph for Zno



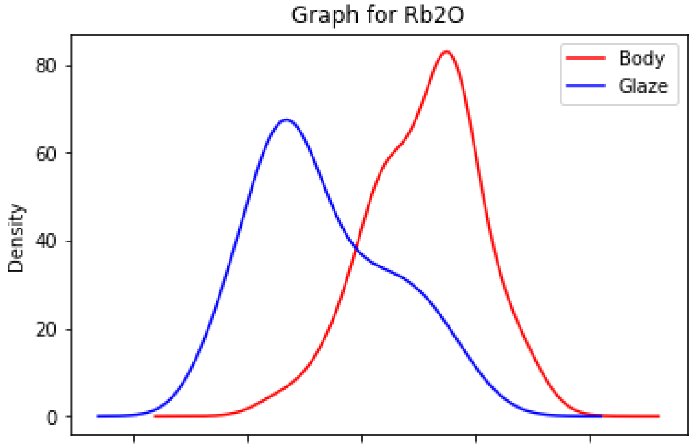
Skewness of body Zno-> -0.4008223994307452
Skewness of glaze Zno-> 0.8653116778536001

Graph for Pbo2



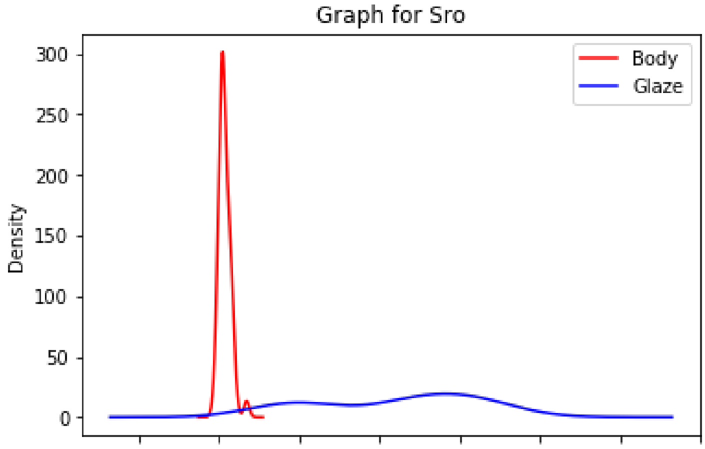
Skewness of body Pbo2-> 0.050147130302251604
Skewness of glaze Pbo2-> 1.0780857192663849

Graph for Rb2O

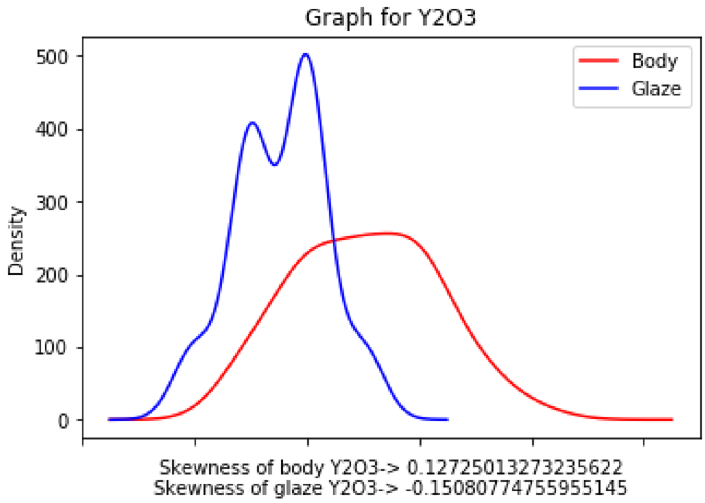


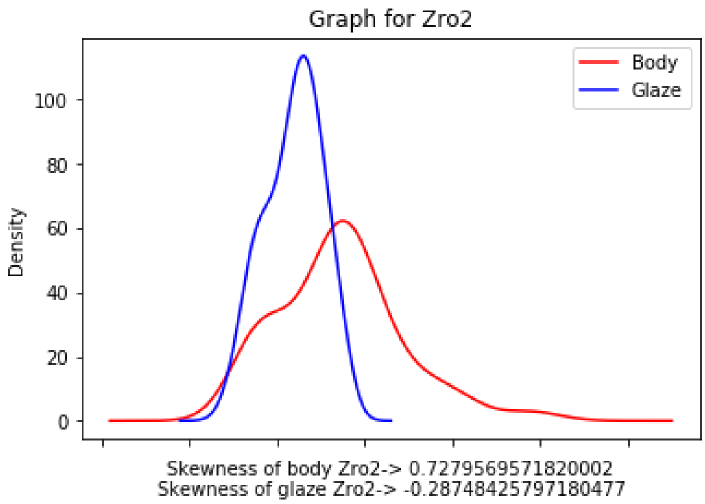
Skewness of body Rb2O-> -0.36746406402837867
Skewness of glaze Rb2O-> 0.5218090050912135

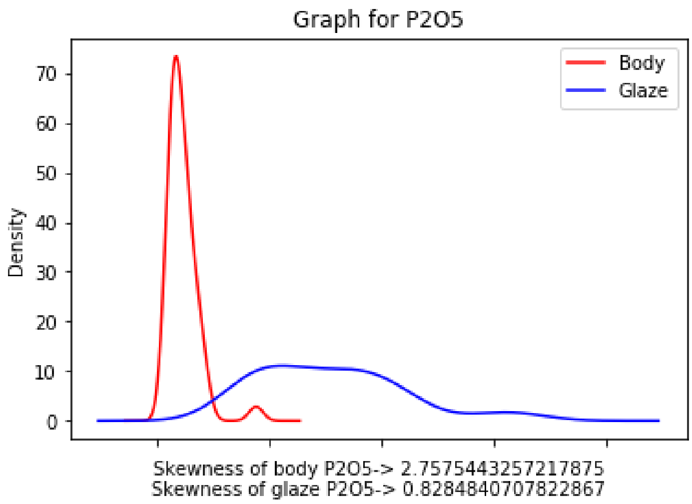
Graph for Sro



Skewness of body Sro-> 1.3084851618216253
Skewness of glaze Sro-> -0.3469509976037987







Logistic Regression

In [24]:

```
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
composition['Part'] = le.fit_transform(composition.Part)
composition['Part'].sample(2)
```

Out[24]:

```
63    1
27    0
Name: Part, dtype: int32
```

In [25]:

```
composition
```

Out[25]:

	Ceramic Name	Part	Na2O	MgO	Al2O3	SiO2	K2O	CaO	TiO2	Fe2O3	MnO	CuO	ZnO	PbO2	Rb2O	SrO	Y2O3
0	FLQ-1-b	0	0.620	0.380	19.610	71.990	4.840	0.310	0.070	1.180	0.063	0.001	0.007	0.001	0.043	0.000	0.004
1	FLQ-2-b	0	0.570	0.470	21.190	70.090	4.980	0.490	0.090	1.120	0.038	0.002	0.008	0.004	0.043	-0.001	0.004
2	FLQ-3-b	0	0.490	0.190	18.600	74.700	3.470	0.430	0.060	1.070	0.042	0.002	0.005	0.005	0.038	0.004	0.004
3	FLQ-4-b	0	0.890	0.300	18.010	74.190	4.010	0.270	0.090	1.230	0.046	0.002	0.007	0.006	0.038	0.001	0.004
4	FLQ-5-b	0	0.030	0.360	18.410	73.990	4.330	0.650	0.050	1.190	0.038	0.004	0.009	0.004	0.036	0.001	0.003
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
83	DY-M-3-g	1	0.340	0.550	12.370	70.700	5.330	8.060	0.060	1.610	0.125	0.001	0.009	0.003	0.025	0.052	0.003
84	DY-QC-1-g	1	0.720	0.340	12.200	72.190	6.190	6.060	0.040	1.270	0.170	0.006	0.011	0.001	0.027	0.054	0.004
85	DY-QC-2-g	1	0.230	0.240	12.990	71.810	5.250	7.150	0.050	1.290	0.075	0.004	0.010	0.000	0.024	0.047	0.004
86	DY-QC-3-g	1	0.140	0.460	12.620	69.160	4.340	11.030	0.050	1.200	0.092	0.004	0.009	0.002	0.023	0.047	0.004
87	DY-QC-4-g	1	0.140	0.630	14.250	71.550	4.870	6.430	0.080	1.050	0.080	0.004	0.009	0.002	0.022	0.041	0.004

88 rows × 19 columns

In [26]:

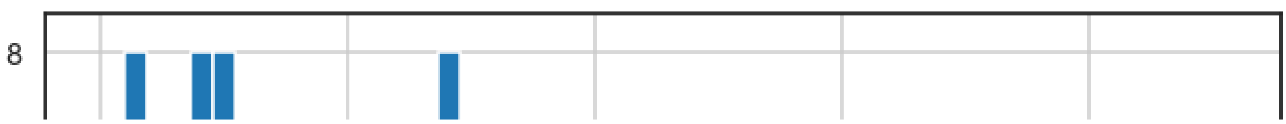
```
feature_cols = composition.columns[2:]
corr_values = composition[feature_cols].corr()

tril_index = np.tril_indices_from(corr_values)

for coord in zip(*tril_index):
    corr_values.iloc[coord[0], coord[1]] = np.NaN

corr_values = (corr_values
               .stack()
               .to_frame()
               .reset_index()
               .rename(columns={'level_0':'feature1',
                                'level_1':'feature2',
                                0:'correlation'}))

corr_values['abs_correlation'] = corr_values.correlation.abs()
```

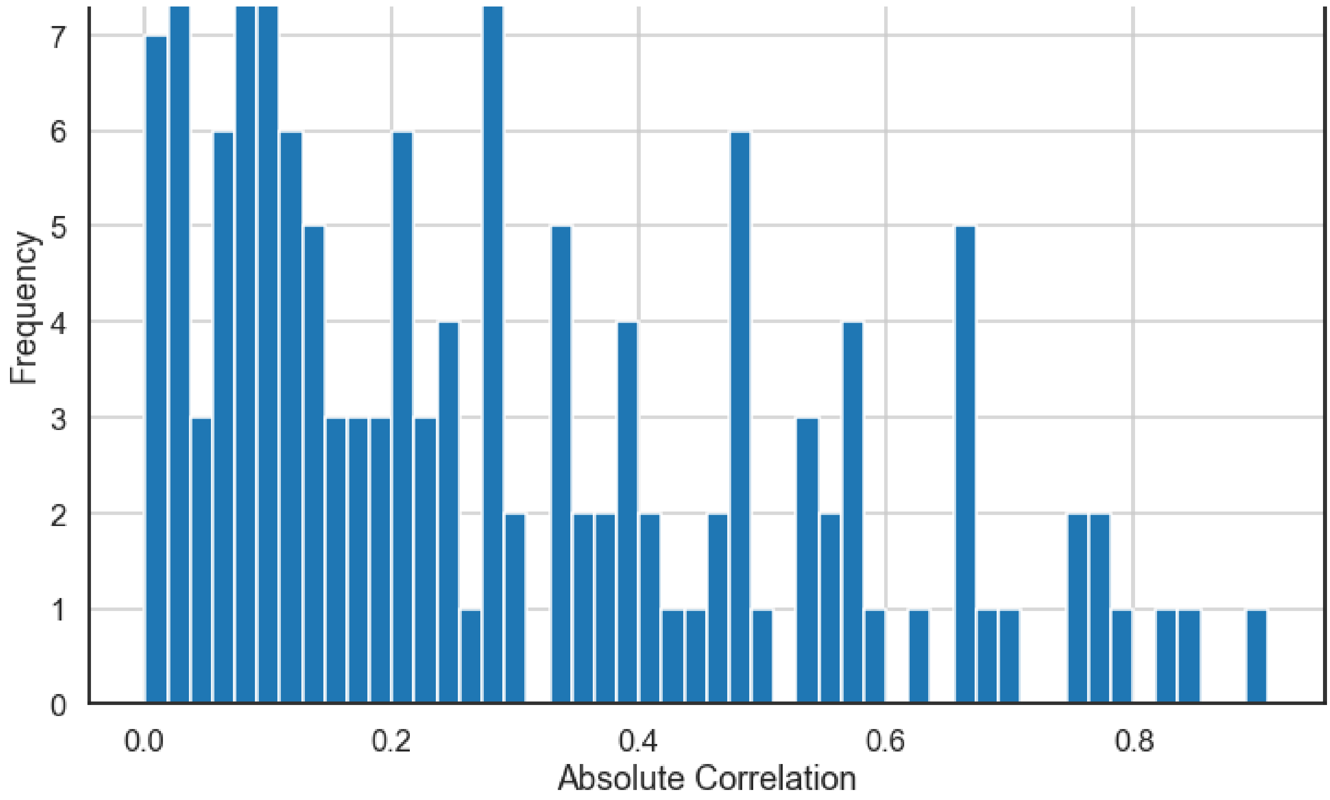
In [27]:

```
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [28]:

```
sns.set_context('talk')
sns.set_style('white')

ax = corr_values.abs_correlation.hist(bins=50, figsize=(12, 8))
ax.set(xlabel='Absolute Correlation', ylabel='Frequency');
```





In [29]:

```
corr_values.sort_values('correlation', ascending=False).query('abs_correlation>0.8')
```

Out[29]:

	feature1	feature2	correlation	abs_correlation
80	CaO	P2O5	0.908	0.908
77	CaO	SrO	0.832	0.832
33	Al2O3	CaO	-0.843	0.843

In [30]:

```
from sklearn.model_selection import StratifiedShuffleSplit

strat_shuf_split = StratifiedShuffleSplit(n_splits=1,
                                          test_size=0.3,
                                          random_state=42)

train_idx, test_idx = next(strat_shuf_split.split(composition[feature_cols], composition
.Part))

X_train = composition.loc[train_idx, feature_cols]
y_train = composition.loc[train_idx, 'Part']

X_test  = composition.loc[test_idx, feature_cols]
y_test  = composition.loc[test_idx, 'Part']
```

In [31]:

```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression(solver='liblinear').fit(X_train, y_train)
```

In [32]:

```
from sklearn.linear_model import LogisticRegressionCV
lr_l1 = LogisticRegressionCV(Cs=10, cv=4, penalty='l1', solver='liblinear').fit(X_train,
y_train)
```

```
C:\Users\Aahil\anaconda3\lib\site-packages\sklearn\svm\_base.py:947: ConvergenceWarning:
Liblinear failed to converge, increase the number of iterations.
  "the number of iterations.", ConvergenceWarning)
```

```
C:\Users\Aahil\anaconda3\lib\site-packages\sklearn\svm\_base.py:947: ConvergenceWarning:
Liblinear failed to converge, increase the number of iterations.
  "the number of iterations.", ConvergenceWarning)
C:\Users\Aahil\anaconda3\lib\site-packages\sklearn\svm\_base.py:947: ConvergenceWarning:
Liblinear failed to converge, increase the number of iterations.
  "the number of iterations.", ConvergenceWarning)
C:\Users\Aahil\anaconda3\lib\site-packages\sklearn\svm\_base.py:947: ConvergenceWarning:
Liblinear failed to converge, increase the number of iterations.
  "the number of iterations.", ConvergenceWarning)
```

In [33]:

```
lr_l2 = LogisticRegressionCV(Cs=10, cv=4, penalty='l2', solver='liblinear').fit(X_train,
y_train)
```

In [34]:

```
coefficients =[]

coeff_labels = ['lr', 'l1', 'l2']
coeff_models = [lr, lr_l1, lr_l2]

for lab,mod in zip(coeff_labels, coeff_models):
    coeffs = mod.coef_
    coeff_label = pd.MultiIndex(levels=[[lab], [0]],
                                 codes=[[0,], [0]])
    coefficients.append(pd.DataFrame(coeffs.T, columns=coeff_label))

coefficients = pd.concat(coefficients, axis=1)

coefficients.sample(10)
```

Out[34]:

	lr	l1	l2
	0	0	0
12	-0.001	0.000	-0.000
9	0.000	0.000	0.000
13	0.007	0.000	0.000
7	-0.009	0.000	-0.004
5	0.820	0.373	0.064
4	0.274	0.000	0.004
1	0.026	0.000	0.002
3	0.126	0.027	0.013
2	-0.809	-0.194	-0.070
8	0.016	0.000	0.001

In [35]:

```
y_pred = []
y_prob = []

coeff_labels = ['lr', 'l1', 'l2']
coeff_models = [lr, lr_l1, lr_l2]

for lab,mod in zip(coeff_labels, coeff_models):
    y_pred.append(pd.Series(mod.predict(X_test), name=lab))
    y_prob.append(pd.Series(mod.predict_proba(X_test).max(axis=1), name=lab))

y_pred = pd.concat(y_pred, axis=1)
y_prob = pd.concat(y_prob, axis=1)

y_prob.head()
```

Out[35]:

	lr	l1	l2
0	1.000	0.958	0.677
1	0.986	0.809	0.563
2	1.000	0.935	0.668
3	0.997	0.913	0.635
4	0.999	0.959	0.661

In [36]:

```
from sklearn.metrics import precision_recall_fscore_support as score
from sklearn.metrics import confusion_matrix, accuracy_score, roc_auc_score
from sklearn.preprocessing import label_binarize

metrics = []
cm = {}

for lab in coeff_labels:
    precision, recall, fscore, _ = score(y_test, y_pred[lab], average='weighted')
    accuracy = accuracy_score(y_test, y_pred[lab])
    cm[lab] = confusion_matrix(y_test, y_pred[lab])
    metrics.append(pd.Series({'precision':precision, 'recall':recall,
                             'fscore':fscore, 'accuracy':accuracy},
                            name=lab))

metrics = pd.concat(metrics, axis=1)
```

In [37]:

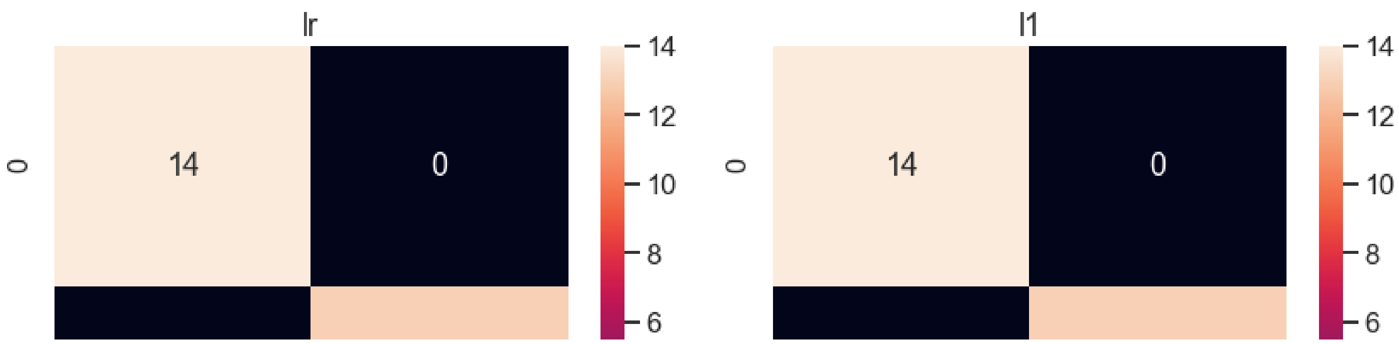
```
metrics
```

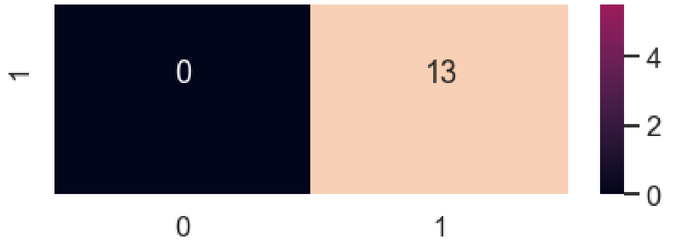
Out[37]:

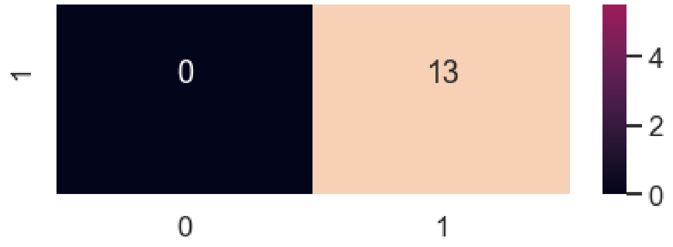
	lr	l1	l2
precision	1.000	1.000	1.000
recall	1.000	1.000	1.000
fscore	1.000	1.000	1.000
accuracy	1.000	1.000	1.000

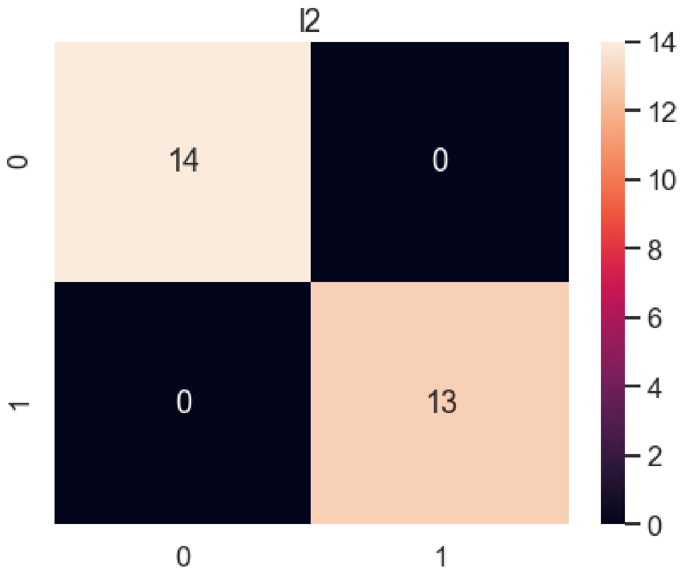
In [38]:

```
fig, axList = plt.subplots(nrows=2, ncols=2)
axList = axList.flatten()
fig.set_size_inches(12, 10)

axList[-1].axis('off')

for ax,lab in zip(axList[:-1], coeff_labels):
    sns.heatmap(cm[lab], ax=ax, annot=True, fmt='d');
    ax.set(title=lab);

plt.tight_layout()
```









KNN

In [39]:

```
composition=composition.drop(columns='Ceramic Name')
```

In [40]:

```
composition
```

Out[40]:

	Part	Na2O	MgO	Al2O3	SiO2	K2O	CaO	TiO2	Fe2O3	MnO	CuO	ZnO	PbO2	Rb2O	SrO	Y2O3	ZrO2	P2(
0	0	0.620	0.380	19.610	71.990	4.840	0.310	0.070	1.180	0.063	0.001	0.007	0.001	0.043	0.000	0.004	0.008	0.0
1	0	0.570	0.470	21.190	70.090	4.980	0.490	0.090	1.120	0.038	0.002	0.008	0.004	0.043	-0.001	0.004	0.010	0.0
2	0	0.490	0.190	18.600	74.700	3.470	0.430	0.060	1.070	0.042	0.002	0.005	0.005	0.038	0.004	0.004	0.008	0.0
3	0	0.890	0.300	18.010	74.190	4.010	0.270	0.090	1.230	0.046	0.002	0.007	0.006	0.038	0.001	0.004	0.007	0.0
4	0	0.030	0.360	18.410	73.990	4.330	0.650	0.050	1.190	0.038	0.004	0.009	0.004	0.036	0.001	0.003	0.008	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
83	1	0.340	0.550	12.370	70.700	5.330	8.060	0.060	1.610	0.125	0.001	0.009	0.003	0.025	0.052	0.003	0.014	0.0
84	1	0.720	0.340	12.200	72.190	6.190	6.060	0.040	1.270	0.170	0.006	0.011	0.001	0.027	0.054	0.004	0.012	0.0
85	1	0.230	0.240	12.990	71.810	5.250	7.150	0.050	1.290	0.075	0.004	0.010	0.000	0.024	0.047	0.004	0.012	0.0
86	1	0.140	0.460	12.620	69.160	4.340	11.030	0.050	1.200	0.092	0.004	0.009	0.002	0.023	0.047	0.004	0.013	0.1
87	1	0.140	0.630	14.250	71.550	4.870	6.430	0.080	1.050	0.080	0.004	0.009	0.002	0.022	0.041	0.004	0.011	0.0

88 rows × 18 columns

In [41]:

```
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
```

```
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report, f1_s
core
```

In [42]:

```
y, X = composition['Part'], composition.drop(columns='Part')
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=42
)
```

In [43]:

```
knn = KNeighborsClassifier(n_neighbors=3)
knn = knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
print(classification_report(y_test, y_pred))
print('Accuracy score: ', round(accuracy_score(y_test, y_pred), 2))
print('F1 Score: ', round(f1_score(y_test, y_pred), 2))
```

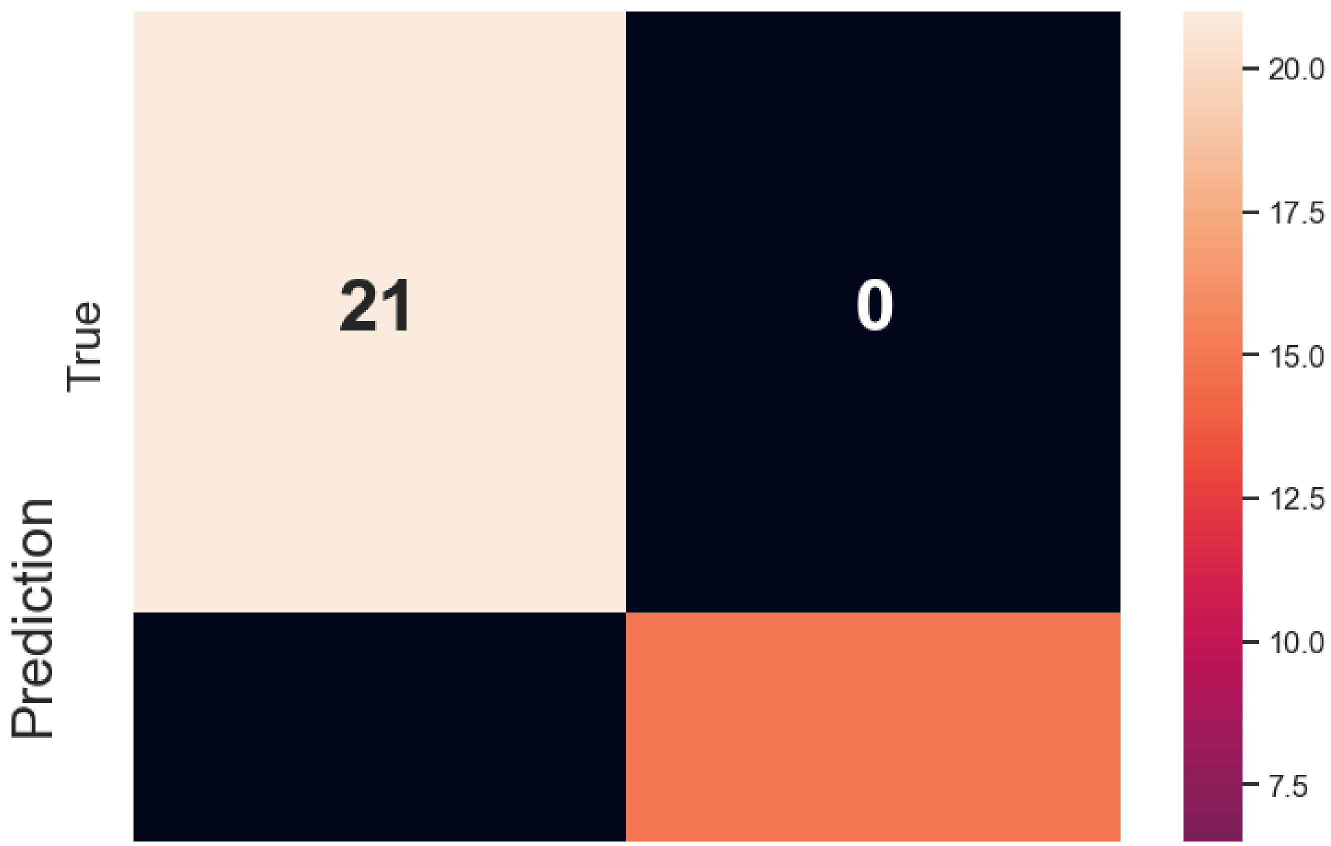
```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        21
           1       1.00      1.00      1.00        15

    accuracy                           1.00        36
   macro avg       1.00      1.00      1.00        36
weighted avg       1.00      1.00      1.00        36

Accuracy score:  1.0
F1 Score:  1.0
```

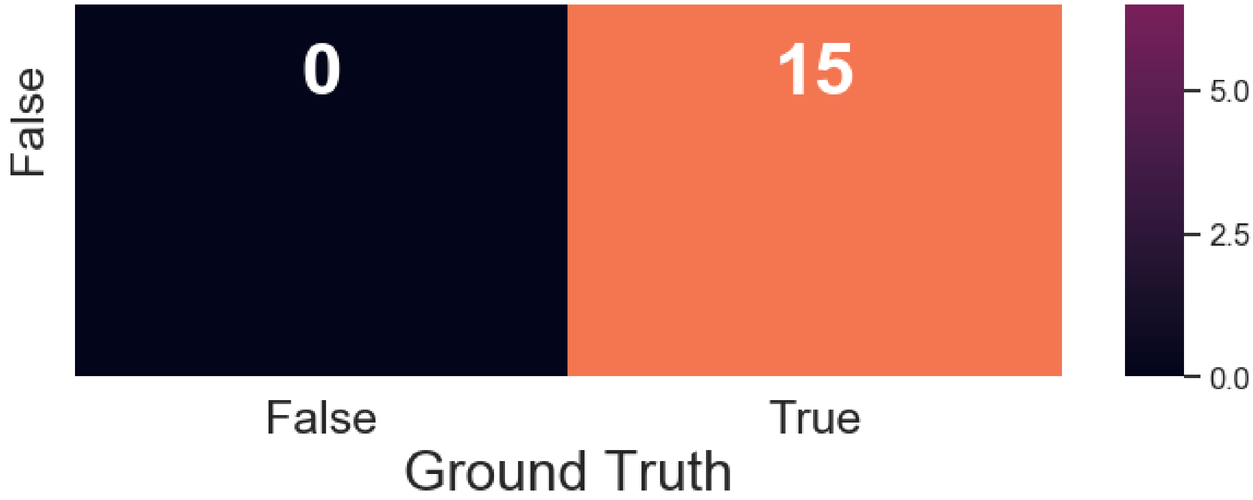
In [44]:

```
_, ax = plt.subplots(figsize=(12,12))
ax = sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, fmt='d',  annot_kws={"siz
e": 40, "weight": "bold"})
labels = ['False', 'True']
ax.set_xticklabels(labels, fontsize=25);
ax.set_yticklabels(labels[::-1], fontsize=25);
ax.set_ylabel('Prediction', fontsize=30);
ax.set_xlabel('Ground Truth', fontsize=30)
```

Out[44]:

```
Text(0.5, 76.5, 'Ground Truth')
```





SVM

In [45]:

```
correlations = composition[feature_cols].corrwith(y)
```

In [46]:

```
correlations
```

Out[46]:

```
Na2O      0.214
MgO       0.411
Al2O3    -0.930
SiO2      0.220
K2O       0.034
CaO       0.910
TiO2     -0.396
Fe2O3    -0.527
MnO       0.704
CuO       0.144
ZnO       0.051
PbO2     -0.331
Rb2O     -0.645
SrO       0.838
Y2O3     -0.586
ZrO2     -0.441
P2O5      0.821
dtype: float64
```

In [47]:

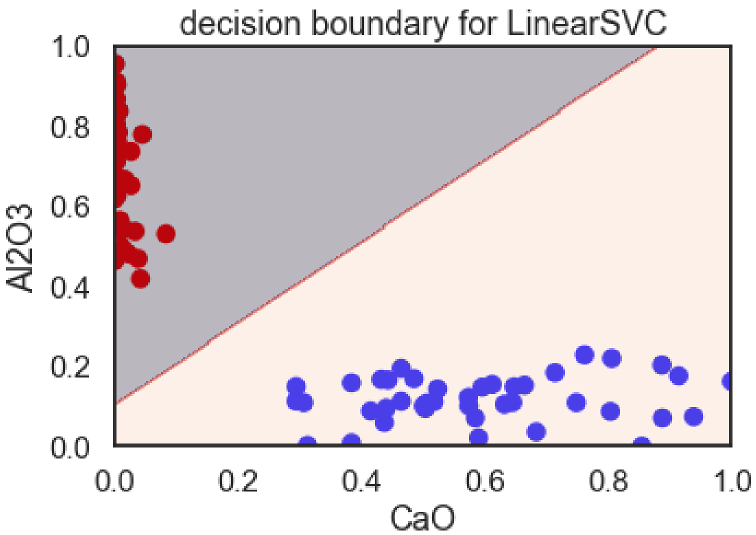
```
from sklearn.preprocessing import MinMaxScaler

feature_cols = correlations.map(abs).sort_values().iloc[-2:].index
print(feature_cols)
X = composition[feature_cols]
scaler = MinMaxScaler()
X = scaler.fit_transform(X)
X = pd.DataFrame(X, columns=['%s_scaled' % fld for fld in feature_cols])
print(X.columns)
```

```
Index(['CaO', 'Al2O3'], dtype='object')
Index(['CaO_scaled', 'Al2O3_scaled'], dtype='object')
```

In [48]:

```
from sklearn.svm import LinearSVC

LSVC = LinearSVC()
LSVC.fit(X, y)
```

```
X_color = X.sample(80, random_state=45)
y_color = y.loc[X_color.index]
y_color = y_color.map(lambda r: 'blue' if r == 1 else 'red')
ax = plt.axes()
ax.scatter(
    X_color.iloc[:, 0], X_color.iloc[:, 1],
    color=y_color, alpha=1)

x_axis, y_axis = np.arange(0, 1.005, .005), np.arange(0, 1.005, .005)
xx, yy = np.meshgrid(x_axis, y_axis)
xx_ravel = xx.ravel()
yy_ravel = yy.ravel()
X_grid = pd.DataFrame([xx_ravel, yy_ravel]).T
y_grid_predictions = LSVC.predict(X_grid)
y_grid_predictions = y_grid_predictions.reshape(xx.shape)
ax.contourf(xx, yy, y_grid_predictions, alpha=.3)

ax.set(
    xlabel=feature_cols[0],
    ylabel=feature_cols[1],
    xlim=[0, 1],
    ylim=[0, 1],
    title='decision boundary for LinearSVC');
```



Decision Trees

In [49]:

```
from sklearn.model_selection import StratifiedShuffleSplit
strat_shuff_split = StratifiedShuffleSplit(n_splits=1, test_size=0.3, random_state=42)

train_idx, test_idx = next(strat_shuff_split.split(composition[feature_cols], compositio
n['Part']))

X_train = composition.loc[train_idx, feature_cols]
y_train = composition.loc[train_idx, 'Part']

X_test = composition.loc[test_idx, feature_cols]
y_test = composition.loc[test_idx, 'Part']
```

In [50]:

```
from sklearn.tree import DecisionTreeClassifier

dt = DecisionTreeClassifier(random_state=42)
dt = dt.fit(X_train, y_train)
```

In [51]:

```
dt.tree_.node_count, dt.tree_.max_depth
```

Out[51]:

```
(3, 1)
```

In [52]:

```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

def measure_error(y_true, y_pred, label):
    return pd.Series({'accuracy':accuracy_score(y_true, y_pred),
                      'precision': precision_score(y_true, y_pred),
                      'recall': recall_score(y_true, y_pred),
                      'f1': f1_score(y_true, y_pred)},
                      name=label)
```

In [53]:

```
y_train_pred = dt.predict(X_train)
y_test_pred = dt.predict(X_test)

train_test_full_error = pd.concat([measure_error(y_train, y_train_pred, 'train'),
                                   measure_error(y_test, y_test_pred, 'test')],
                                   axis=1)

train_test_full_error
```

Out[53]:

	train	test
accuracy	1.000	1.000
precision	1.000	1.000
recall	1.000	1.000
f1	1.000	1.000

In [ ]: