```
import pandas as pd
import numpy as np
import seaborn as sns
```

```
columns = ['RI','Na', 'Mg', 'Al', 'Si', 'K', 'Ca', 'Ba','Fe','Type']
Abc = pd.read_csv('https://archive.ics.uci.edu/ml/machine-learning-databases/glass/glass.data',
                 names=columns, header=None)
```

```
print(Abc)
```

```
          RI     Na    Mg    Al     Si     K    Ca    Ba   Fe  Type
1    1.52101  13.64  4.49  1.10  71.78  0.06  8.75  0.00  0.0     1
2    1.51761  13.89  3.60  1.36  72.73  0.48  7.83  0.00  0.0     1
3    1.51618  13.53  3.55  1.54  72.99  0.39  7.78  0.00  0.0     1
4    1.51766  13.21  3.69  1.29  72.61  0.57  8.22  0.00  0.0     1
5    1.51742  13.27  3.62  1.24  73.08  0.55  8.07  0.00  0.0     1
..       ...    ...   ...   ...    ...   ...   ...   ...  ...   ...
210  1.51623  14.14  0.00  2.88  72.61  0.08  9.18  1.06  0.0     7
211  1.51685  14.92  0.00  1.99  73.06  0.00  8.40  1.59  0.0     7
212  1.52065  14.36  0.00  2.02  73.42  0.00  8.44  1.64  0.0     7
213  1.51651  14.38  0.00  1.94  73.61  0.00  8.48  1.57  0.0     7
214  1.51711  14.23  0.00  2.08  73.36  0.00  8.62  1.67  0.0     7

[214 rows x 10 columns]
```

```
# finding null values
Abc.isnull().sum()
```

```
RI      0
Na      0
Mg      0
Al      0
Si      0
K       0
Ca      0
Ba      0
Fe      0
Type    0
dtype: int64
```

```
X,Y = Abc.iloc[:,1:-1].values, Abc.iloc[:,-1].values
```

```
print(X)
```

```
[[13.64  4.49  1.1  ...  8.75  0.    0.  ]
 [13.89  3.6   1.36 ...  7.83  0.    0.  ]
 [13.53  3.55  1.54 ...  7.78  0.    0.  ]
 ...
 [14.36  0.    2.02 ...  8.44  1.64  0.  ]
 [14.38  0.    1.94 ...  8.48  1.57  0.  ]
 [14.23  0.    2.08 ...  8.62  1.67  0.  ]]
```

```
#split the data into train and test
from sklearn.model_selection import train_test_split
X_train,X_test, Y_train, Y_test =train_test_split(X,Y,test_size=0.3,random_state=0,stratify=Y)
```

```
print(X_train,X_test)
```

```
 [1.415e+01 0.000e+00 2.090e+00 7.274e+01 0.000e+00 1.088e+01 0.000e+00
  0.000e+00]
 [1.372e+01 3.680e+00 1.810e+00 7.206e+01 6.400e-01 7.880e+00 0.000e+00
  0.000e+00]
 [1.309e+01 3.520e+00 1.550e+00 7.287e+01 6.800e-01 8.050e+00 0.000e+00
  9.000e-02]
 [1.477e+01 3.750e+00 2.900e-01 7.202e+01 3.000e-02 9.000e+00 0.000e+00
  0.000e+00]
 [1.145e+01 0.000e+00 1.880e+00 7.219e+01 8.100e-01 1.324e+01 0.000e+00
  3.400e-01]
 [1.324e+01 3.570e+00 1.380e+00 7.270e+01 5.600e-01 8.440e+00 0.000e+00
  1.000e-01]
 [1.379e+01 2.410e+00 1.190e+00 7.276e+01 0.000e+00 9.770e+00 0.000e+00
  0.000e+00]
 [1.425e+01 3.090e+00 2.080e+00 7.228e+01 1.100e+00 7.080e+00 0.000e+00
  0.000e+00]
 [1.421e+01 3.820e+00 4.700e-01 7.177e+01 1.100e-01 9.570e+00 0.000e+00
  0.000e+00]
 [1.274e+01 3.480e+00 1.350e+00 7.296e+01 6.400e-01 8.680e+00 0.000e+00
  0.000e+00]
 [1.330e+01 3.640e+00 1.530e+00 7.253e+01 6.500e-01 8.030e+00 0.000e+00
  2.900e-01]
 [1.419e+01 3.780e+00 9.100e-01 7.136e+01 2.300e-01 9.140e+00 0.000e+00
  3.700e-01]
 [1.348e+01 3.740e+00 9.000e-01 7.201e+01 1.800e-01 9.610e+00 0.000e+00
  7.000e-02]
 [1.369e+01 3.200e+00 1.810e+00 7.281e+01 1.760e+00 5.430e+00 1.190e+00
  0.000e+00]
 [1.579e+01 1.830e+00 1.310e+00 7.043e+01 3.100e-01 8.610e+00 1.680e+00
  0.000e+00]
 [1.499e+01 7.800e-01 1.740e+00 7.250e+01 0.000e+00 9.950e+00 0.000e+00
  0.000e+00]
 [1.486e+01 3.670e+00 1.740e+00 7.187e+01 1.600e-01 7.360e+00 0.000e+00
  1.200e-01]
 [1.494e+01 0.000e+00 1.870e+00 7.311e+01 0.000e+00 8.670e+00 1.380e+00
  0.000e+00]]
```

```
print(Y_train)
```

```
[7 1 2 6 2 5 6 1 5 2 2 5 7 2 2 1 2 2 2 1 2 2 3 7 2 6 1 1 2 3 1 2 1 2 3 1 2
 2 2 2 7 2 2 1 2 1 3 1 7 1 2 2 6 3 7 6 2 1 2 2 1 1 2 3 2 1 1 7 7 2 7 2 5 2
 2 2 1 7 2 5 2 5 1 3 7 2 1 1 3 7 1 2 7 1 1 3 1 2 7 2 1 5 1 2 2 7 2 5 2 1 1
 7 1 1 2 5 1 1 2 2 2 7 2 1 7 1 1 1 1 3 1 7 1 2 1 1 2 1 2 2 1 1 3 6 7 3 1 1
 1]
```

```
print(Y_test)
```

```
[2 3 2 1 2 1 1 2 5 7 1 2 5 5 1 1 2 1 7 2 1 2 1 2 7 2 2 1 2 1 3 1 7 2 2 1 2
 5 1 7 1 2 2 7 1 3 1 6 2 2 1 2 3 6 2 1 1 2 3 1 7 7 6 2 7]
```

```
#normalizing the data
from sklearn.preprocessing import MinMaxScaler
mms = MinMaxScaler()
X_train_norm = mms.fit_transform(X_train)
X_test_norm = mms.transform(X_test)
```

```
print(X_train_norm)
```

```
[[0.55037594 0.         0.54814815 ... 0.         0.91428571 0.        ]
 [0.36390977 0.80400891 0.26296296 ... 0.18209877 0.         0.        ]
 [0.32180451 0.79287305 0.48148148 ... 0.1563786  0.         0.        ]
 ...
 [0.27518797 0.78396437 0.4037037  ... 0.21296296 0.         0.        ]
 [0.32180451 0.77505568 0.35185185 ... 0.20164609 0.         0.        ]
 [0.30977444 0.80400891 0.47407407 ... 0.16460905 0.         0.50980392]]
```

```
print(X_test_norm)
```

```
   0.          0.        ]
 [ 0.4962406   0.83741648  0.08888889  0.35357143  0.01771337  0.32716049
   0.          0.        ]
 [ 0.41353383  0.83296214  0.30740741  0.56785714  0.09500805  0.16049383
   0.          0.        ]
 [ 0.51428571  0.          0.64814815  0.52321429  0.          0.4537037
   0.          0.        ]
 [ 0.44962406  0.81959911  0.54444444  0.40178571  0.10305958  0.14506173
   0.          0.        ]
 [ 0.35488722  0.78396437  0.44814815  0.54642857  0.10950081  0.16255144
   0.          0.17647059]
 [ 0.6075188   0.83518931 -0.01851852  0.39464286  0.00483092  0.26028807
   0.          0.        ]
 [ 0.10827068  0.          0.57037037  0.425       0.13043478  0.69650206
   0.          0.66666667]
 [ 0.37744361  0.79510022  0.38518519  0.51607143  0.09017713  0.2026749
   0.          0.19607843]
 [ 0.46015038  0.53674833  0.31481481  0.52678571  0.          0.33950617
   0.          0.        ]
 [ 0.52932331  0.68819599  0.64444444  0.44107143  0.17713366  0.0627572
   0.          0.        ]
 [ 0.52330827  0.85077951  0.04814815  0.35        0.01771337  0.31893004
   0.          0.        ]
 [ 0.30225564  0.77505568  0.37407407  0.5625      0.10305958  0.22736626
   0.          0.        ]
 [ 0.38646617  0.81069042  0.44074074  0.48571429  0.10466989  0.16049383
   0.          0.56862745]
 [ 0.52030075  0.84187082  0.21111111  0.27678571  0.03703704  0.27469136
   0.          0.7254902 ]
 [ 0.41353383  0.83296214  0.20740741  0.39285714  0.02898551  0.32304527
   0.          0.1372549 ]
 [ 0.44511278  0.71269488  0.54444444  0.53571429  0.28341385 -0.10699588
   0.37777778  0.        ]
 [ 0.76090226  0.40757238  0.35925926  0.11071429  0.04991948  0.22016461
   0.53333333  0.        ]
 [ 0.6406015   0.17371938  0.51851852  0.48035714  0.          0.35802469
   0.          0.        ]
 [ 0.62105263  0.81737194  0.51851852  0.36785714  0.0257649   0.09156379
   0.          0.23529412]
 [ 0.63308271  0.          0.56666667  0.58928571  0.          0.22633745
   0.43809524  0.        ]]
```

```
#Describing the data as Minimum, Maximum, mean, standard deviating for each feature
Abc.describe()
```

	RI	Na	Mg	Al	Si	K	Ca	Ba	Fe
count	214.000000	214.000000	214.000000	214.000000	214.000000	214.000000	214.000000	214.000000	214.000000
mean	1.518365	13.407850	2.684533	1.444907	72.650935	0.497056	8.956963	0.175047	0.057009
std	0.003037	0.816604	1.442408	0.499270	0.774546	0.652192	1.423153	0.497219	0.097439
min	1.511150	10.730000	0.000000	0.290000	69.810000	0.000000	5.430000	0.000000	0.000000
25%	1.516522	12.907500	2.115000	1.190000	72.280000	0.122500	8.240000	0.000000	0.000000
50%	1.517680	13.300000	3.480000	1.360000	72.790000	0.555000	8.600000	0.000000	0.000000
75%	1.519157	13.825000	3.600000	1.630000	73.087500	0.610000	9.172500	0.000000	0.100000
max	1.533930	17.380000	4.490000	3.500000	75.410000	6.210000	16.190000	3.150000	0.510000

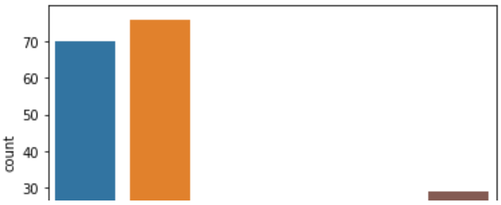
```
#Apply Simple linear Regression to training set
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, Y_train)
```

```
LinearRegression()
```

```
#predecting the test result
y_predict = regressor.predict(X_test)
```

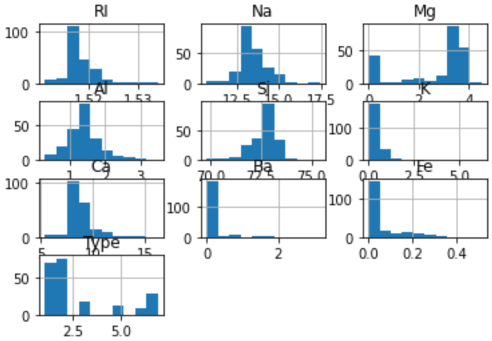
```
#finding the count of target feature using seaborn plot
import matplotlib.pyplot as plt
sns.countplot(Abc["Type"])
plt.show()
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variable as
  warnings.warn(
```

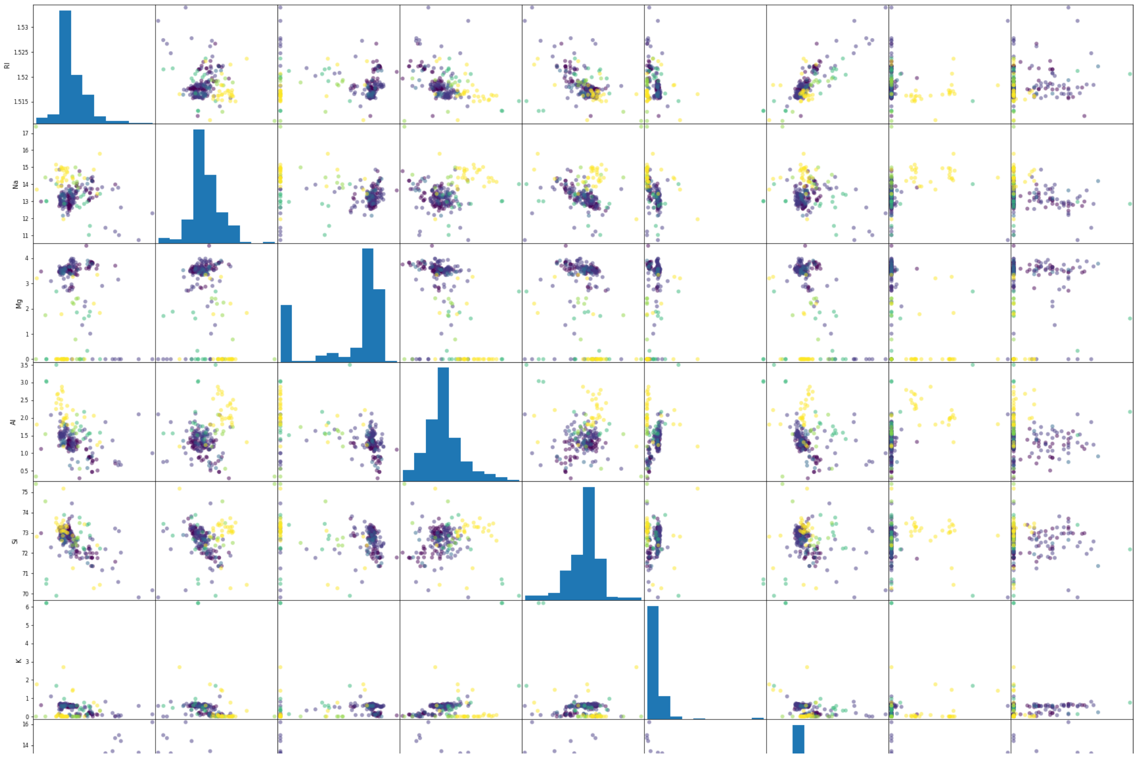


```
Abc.hist()
```

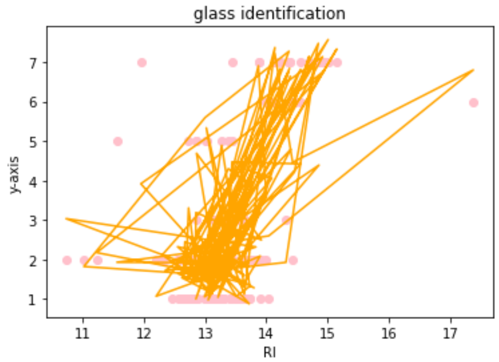
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7fd010b35310>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fd010ae95b0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fd010affbe0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7fd010aaffa0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fd010a693d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fd010a15700>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7fd010a157f0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fd010a44c40>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fd0109ad3d0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7fd0109597c0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fd010987af0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fd0109330d0>]],
      dtype=object)
```

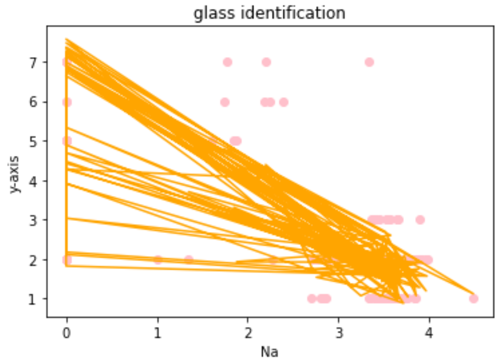


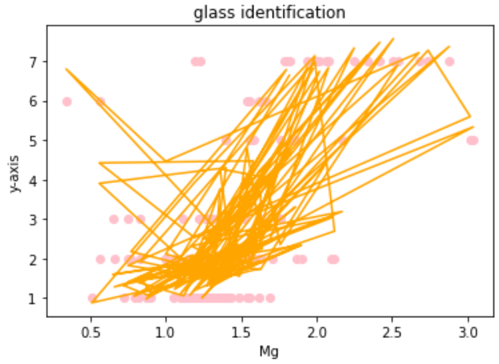
```
import matplotlib.pyplot as plt
pd.plotting.scatter_matrix(Abc.iloc[:,:-1], c=Abc.iloc[:,-1], figsize=(30, 30), marker='o')
plt.legend(Abc["Type"].unique())
plt.show()
```

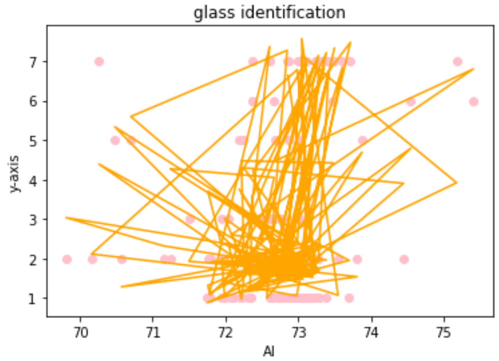


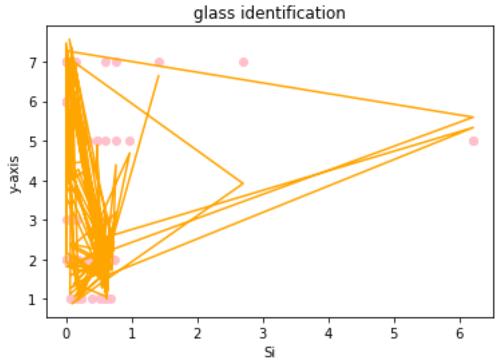
```
import matplotlib.pyplot as pt
#Visualization of the training set results
for i in range(len(columns)-2):
  pt.scatter(X_train[:,i:i+1], Y_train, color = 'pink')
  pt.plot(X_train[:,i:i+1], regressor.predict(X_train), color = 'orange')
  pt.title('glass identification')
  pt.xlabel(columns[i])
  pt.ylabel('y-axis')
  pt.show()
```

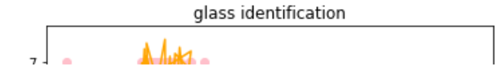




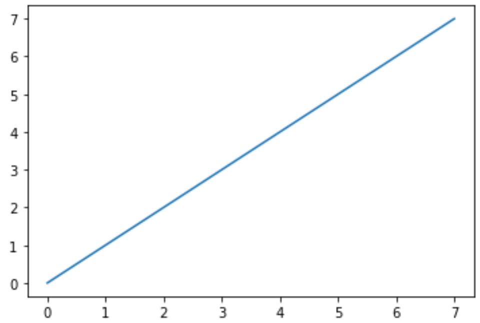


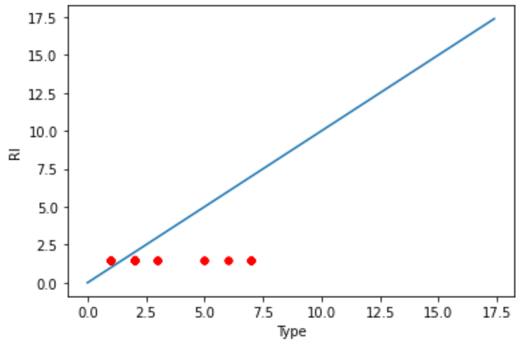


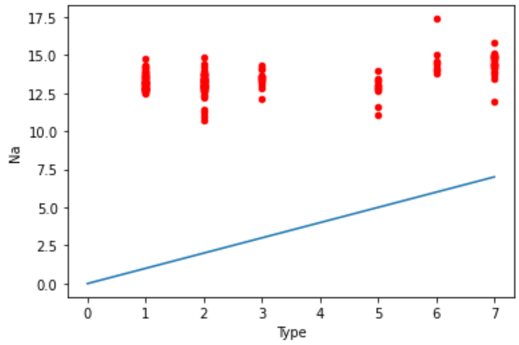


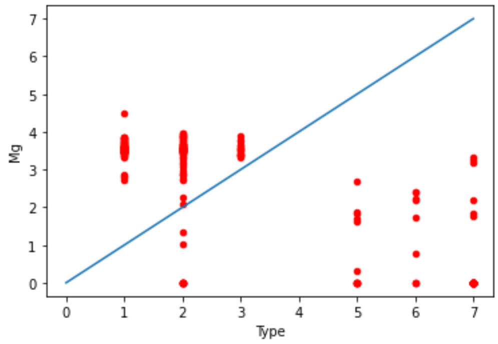


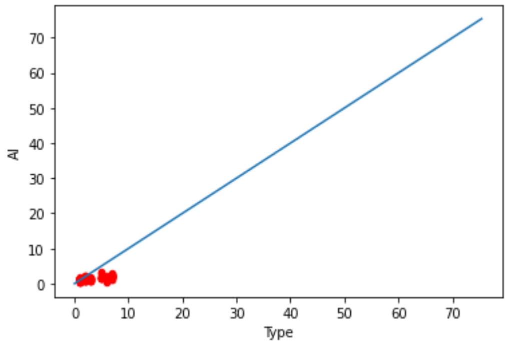
```
for i in range(len(columns)):
  mymax = max(max(Abc['Type']),max(Abc[Abc.columns[i]]))
  plt.plot([0,mymax],[0,mymax])
  Abc.plot(kind="scatter",x="Type",y=Abc.columns[i],color ="red")
```

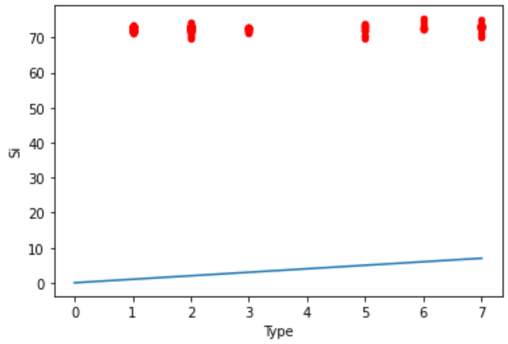

0
1
2
3
4
5
6
7

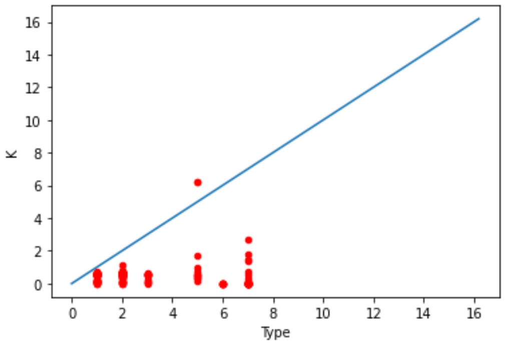

RI
Type
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5

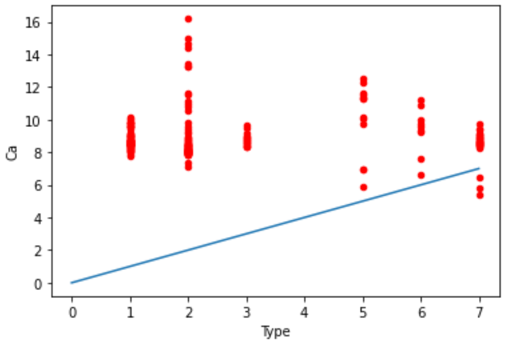

Na
Type
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
0
1
2
3
4
5
6
7

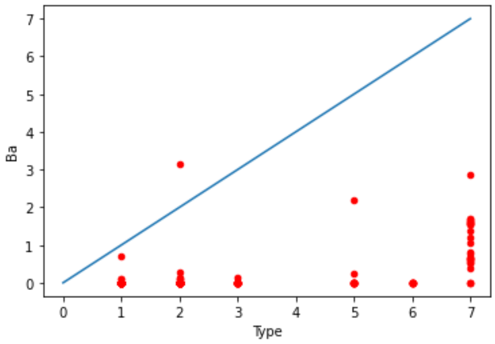

Mg
Type
0
1
2
3
4
5
6
7

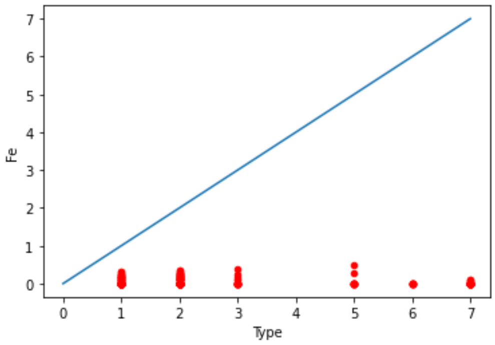

Al
Type
0
10
20
30
40
50
60
70

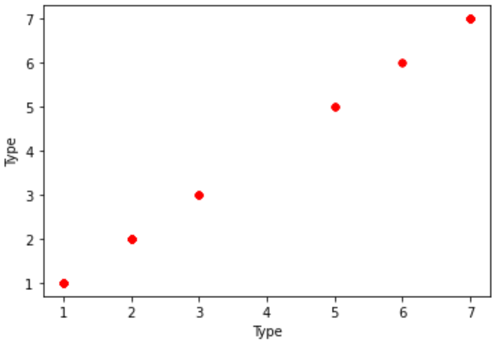

Si
Type
0
10
20
30
40
50
60
70
0
1
2
3
4
5
6
7











```
import numpy as np
spot = np.round(y_predict)
G = spot.astype(int)
G
sac = []

for spot in Y_test:
    if spot in G:
        sac.append(spot)
        np.delete(G,spot)
```

```
from collections import Counter
```

```
Counter(sac).keys()
Counter(sac).values()
```

```
     dict_values([23, 5, 21, 4, 9, 3])
```

```
test_keys, test_values = np.unique(sac, return_counts=True)
```

```
import numpy as np
lkta = {}
i=0
for key in test_keys:
    lkta[key] = test_values[i]
    i=i+1
lkta
```

```
     {1: 21, 2: 23, 3: 5, 5: 4, 6: 3, 7: 9}
```

```
tr = Counter(lkta).most_common(3)
```

```
print("")
for i in tr:
  print(i)
```

```
     (2, 23)
     (1, 21)
     (7, 9)
```

```
# by applying all the visualizations and plots out of the 7 features given according to my knowledge and coding done by me i got 2 1 and
```

✓ 0s completed at 10:37 PM