Part 2 Analysis

This Python code aims to explore and analyze a dataset related to glass identification using various data visualization and preprocessing techniques. Below is an explanation and analysis of the key parts of the code:

In the first lines of code we focused on loading all the needed libraries and the dataset. The column names are **Id, Ri, Na, Mg, Al, Si, K, Ca, Ba, Fe, and Type**. However, we dropped the columns Id and Type focusing the analysis on 9 chemical properties.

```
# loading all the needed libraries
from matplotlib import pyplot
import numpy
from pandas import read_csv
from pandas import set_option
from pandas.plotting import scatter_matrix
from numpy import set_printoptions
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import Normalizer
from sklearn.preprocessing import Binarizer

# loading the dataset
filename = 'C:\\Users\\fabio\\Desktop\\glass+identification\\glass.data'
columns = ['Id', 'Ri', 'Na', 'Mg', 'Al', 'Si', 'K', 'Ca', 'Ba', 'Fe', 'Type']
columns_after_drop = ['Ri', 'Na', 'Mg', 'Al', 'Si', 'K', 'Ca', 'Ba', 'Fe']
dataset_before_drop = read_csv(filename, names=columns, sep=',')
dataset = dataset_before_drop.drop(['Id', 'Type'], axis=1)
print("haad:" , dataset.head())
print("Shape of dataset:", dataset.shape)
```

The two variables are dropped in this analysis for the following reasons:

- The Id column typically contains unique identifiers for each row that have no bearing on the chemical properties of the samples themselves. Including Id in the analysis could introduce noise without adding any useful information. It doesn't contribute to understanding the relationships between the chemical properties since it's just a label.
- The Type column likely contains categorical labels indicating the class or category of each observation. For example, 1 identifies building_windows_float_processed, 2 identifies building_windows_non_float_processed, 5 identifies containers, and so on. For unsupervised analysis, we focus only on the chemical properties.

Explanation of chemical properties:

- Refractive index (Ri)
- Sodium (Na)
- Magnesium (Mg)
- Aluminum (Al)
- Silicon (Si)
- Potassium (K)
- Calcium (Ca)
- Barium (Ba)
- Iron (Fe)

After performing the head() and shape operations, we can examine the first few rows of the dataset.

```
haad:         Ri     Na    Mg    Al     Si     K    Ca   Ba   Fe
0  1.52101  13.64  4.49  1.10  71.78  0.06  8.75  0.0  0.0
1  1.51761  13.89  3.60  1.36  72.73  0.48  7.83  0.0  0.0
2  1.51618  13.53  3.55  1.54  72.99  0.39  7.78  0.0  0.0
3  1.51766  13.21  3.69  1.29  72.61  0.57  8.22  0.0  0.0
4  1.51742  13.27  3.62  1.24  73.08  0.55  8.07  0.0  0.0
Shape of dataset: (214, 9)
```

From the first few rows we can get some ideas of the values we'll be encountering:

1. **Ri**: The values are very close to one another, ranging between 1.516 and 1.521. This indicates that the refractive index might not have a high degree of variability among the samples.
2. **Na**: The values hover around 13-14. Sodium content might have a narrow range in glass composition.
3. **Mg and Al**: These features vary a bit more, and differences in these levels could provide insight into the types of glass or the processes used in their production.
4. **Si:** The silicon values are around 71-73, which makes sense, as glass primarily consists of silica. We can suppose that it is the dominant component of the dataset.
5. **K and Ca**: There is more variability in Potassium (as low as 0.06 to 0.57) and Calcium content, which could indicate different types of additives or manufacturing processes.
6. **Ba and Fe**: The values for these are mostly zero, meaning many samples do not contain detectable levels of Barium and Iron.

However, we can notice that the readability of the head() of our dataset is not the best, for this reason we will use pandas to better it.

```
# Set pandas display options for better output visibility
set_option('display.max_rows', 500)
set_option('display.width', 100)
set_option('display.precision', 3)  # Corrected option

# Print dataset types to check data types of each column
print("Data types of each column:\n", dataset.dtypes)

# Show the first 20 rows of the dataset
print("First 20 rows of the dataset:\n", dataset.head(20))
```

After performing these lines of code, we can notice that the visibility gets way better:

```
First 20 rows of the dataset:
        Ri     Na    Mg    Al     Si     K    Ca   Ba    Fe
0   1.521  13.64  4.49  1.10  71.78  0.06  8.75  0.0  0.00
1   1.518  13.89  3.60  1.36  72.73  0.48  7.83  0.0  0.00
2   1.516  13.53  3.55  1.54  72.99  0.39  7.78  0.0  0.00
3   1.518  13.21  3.69  1.29  72.61  0.57  8.22  0.0  0.00
4   1.517  13.27  3.62  1.24  73.08  0.55  8.07  0.0  0.00
5   1.516  12.79  3.61  1.62  72.97  0.64  8.07  0.0  0.26
6   1.517  13.30  3.60  1.14  73.09  0.58  8.17  0.0  0.00
7   1.518  13.15  3.61  1.05  73.24  0.57  8.24  0.0  0.00
8   1.519  14.04  3.58  1.37  72.08  0.56  8.30  0.0  0.00
9   1.518  13.00  3.60  1.36  72.99  0.57  8.40  0.0  0.11
10  1.516  12.72  3.46  1.56  73.20  0.67  8.09  0.0  0.24
11  1.518  12.80  3.66  1.27  73.01  0.60  8.56  0.0  0.00
12  1.516  12.88  3.43  1.40  73.28  0.69  8.05  0.0  0.24
13  1.517  12.86  3.56  1.27  73.21  0.54  8.38  0.0  0.17
14  1.518  12.61  3.59  1.31  73.29  0.58  8.50  0.0  0.00
15  1.518  12.81  3.54  1.23  73.24  0.58  8.39  0.0  0.00
16  1.518  12.68  3.67  1.16  73.11  0.61  8.70  0.0  0.00
17  1.522  14.36  3.85  0.89  71.36  0.15  9.15  0.0  0.00
18  1.519  13.90  3.73  1.18  72.12  0.06  8.89  0.0  0.00
19  1.517  13.02  3.54  1.69  72.73  0.54  8.44  0.0  0.07
```

To get more insights of our data we can now take advantage of the describe() function which will list 8 statistical properties of each attribute:

```
print(dataset.describe())
```

Resulting in this table:

	Ri	Na	Mg	Al	...	K	Ca	Ba	Fe
count	214.000	214.000	214.000	214.000	...	214.000	214.000	214.000	214.000
mean	1.518	13.408	2.685	1.445	...	0.497	8.957	0.175	0.057
std	0.003	0.817	1.442	0.499	...	0.652	1.423	0.497	0.097
min	1.511	10.730	0.000	0.290	...	0.000	5.430	0.000	0.000
25%	1.517	12.908	2.115	1.190	...	0.122	8.240	0.000	0.000
50%	1.518	13.300	3.480	1.360	...	0.555	8.600	0.000	0.000
75%	1.519	13.825	3.600	1.630	...	0.610	9.172	0.000	0.100
max	1.534	17.380	4.490	3.500	...	6.210	16.190	3.150	0.510

From this table we can definitely notice that Silicon, Calcium, and Sodium are the most consistent and dominant components of the glass, with narrow ranges and higher mean values. Also, the chemicals Potassium, Magnesium, and Barium show significant variability across samples, indicating that some glass formulations include these elements while others exclude them entirely. However, the **max values** for Potassium, Barium, and Iron indicate a few samples with much higher concentrations than the typical glass sample, suggesting the presence of **outliers** or specialized glass types. It is also important to notice that many chemical properties, particularly Barium and Iron, have **zero values** in several samples. This indicates the absence of these elements in a majority of glass types.

We now decided to group the chemicals by type:

```
print(dataset_before_drop.groupby('Type').size())
```

```
Type
1    70
2    76
3    17
5    13
6     9
7    29
```

For further understanding of our result, here is an explanation of the numbers:

Type of glass: (class attribute)

- 1 building_windows_float_processed
- 2 building_windows_non_float_processed
- 3 vehicle_windows_float_processed
- 4 vehicle_windows_non_float_processed (none in this database)
- 5 containers

- 6 tableware
- 7 headlamps

From this table we can notice that type 2 (building_windows_non_float_processed) has the highest count, while type 6 (tableware) has the lowest count. The high counts of Types 1 and 2 compared to Type 3 may suggest a stronger demand for building windows than vehicle windows in the context of what our dataset reflects.

We now utilize the Pearson correlation coefficient for our analysis because it measures the linear relationship between two continuous variables.

```
print(dataset.corr(method = 'pearson'))
```

```
            Ri     Na     Mg     Al     Si     K      Ca          Ba     Fe
Ri  1.000e+00 -0.192 -0.122 -0.407 -0.542 -0.290  0.810 -3.860e-04  0.143
Na -1.919e-01  1.000 -0.274  0.157 -0.070 -0.266 -0.275  3.266e-01 -0.241
Mg -1.223e-01 -0.274  1.000 -0.482 -0.166  0.005 -0.444 -4.923e-01  0.083
Al -4.073e-01  0.157 -0.482  1.000 -0.006  0.326 -0.260  4.794e-01 -0.074
Si -5.421e-01 -0.070 -0.166 -0.006  1.000 -0.193 -0.209 -1.022e-01 -0.094
K  -2.898e-01 -0.266  0.005  0.326 -0.193  1.000 -0.318 -4.262e-02 -0.008
Ca  8.104e-01 -0.275 -0.444 -0.260 -0.209 -0.318  1.000 -1.128e-01  0.125
Ba -3.860e-04  0.327 -0.492  0.479 -0.102 -0.043 -0.113  1.000e+00 -0.059
Fe  1.430e-01 -0.241  0.083 -0.074 -0.094 -0.008  0.125 -5.869e-02  1.000
```

We can notice some good insights. The strong positive correlation of Calcium and Rhodium shows that higher calcium concentrations correlate with increased rhodium levels. This finding suggests that calcium plays a crucial role in enhancing rhodium's presence or effectiveness in the glass matrix. On the other hand, the negative correlation of Aluminum and Rhodium suggests that higher aluminum concentrations may inhibit rhodium levels. Also, the negative correlation between Potassium and Sodium with Rhodium shows potential interactions that may be worth exploring in future studies. We also noticed some minimal correlations, for example Barium and Iron. These elements show negligible influence on rhodium levels, indicating that they may not significantly impact the glass properties related to rhodium.

In order to evaluate the distribution of each chemical property within the glass dataset we decided to use the skeweness.

```
print(dataset.skew())
```

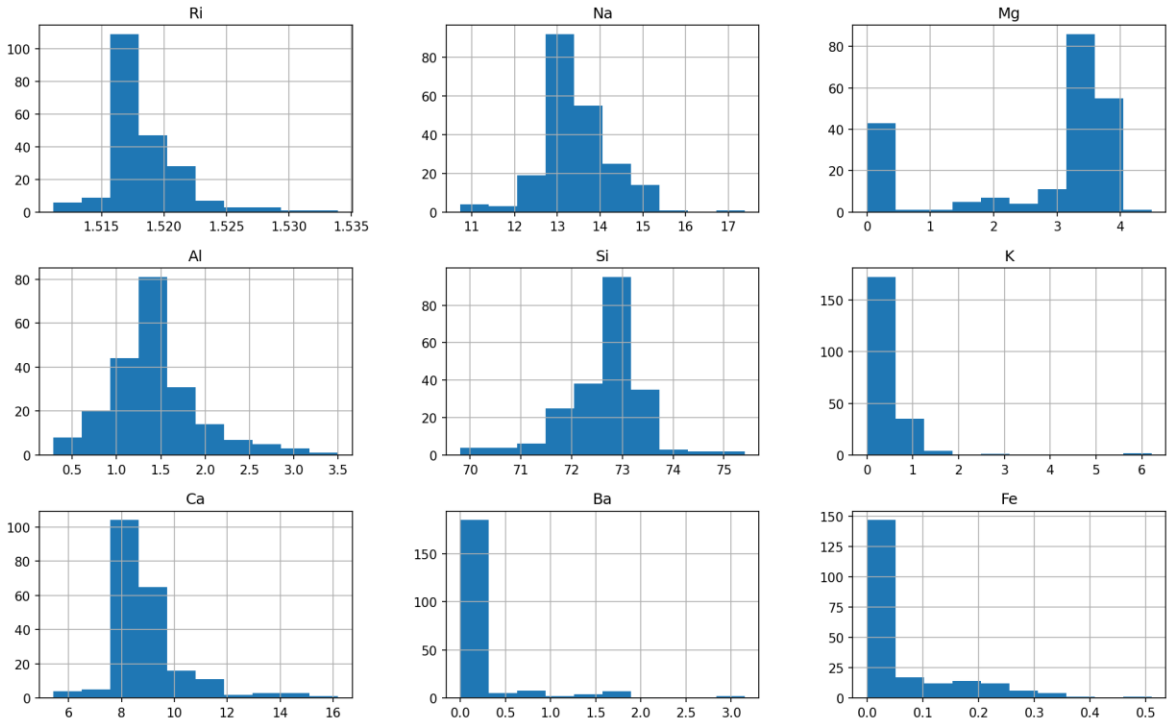
```
Ri     1.625
Na     0.454
Mg    -1.153
Al     0.907
Si    -0.730
K      6.552
Ca     2.047
Ba     3.416
Fe     1.754
```

Some important observations are the following:

- Potassium has a skewness value of 6.552, indicating very high concentrations. This could point to specific glass types or manufacturing processes that incorporate higher potassium levels.
- Barium (Ba) and Calcium (Ca) also show high positive skewness (3.416 and 2.047, respectively), suggesting that most samples have low concentrations of these elements while a few have much higher levels. This indicates potential specialized glass formulations that might use barium or calcium differently than the standard formulations.
- Sodium (Na) and Aluminum (Al) show a generally balanced distribution but with some influence from higher values.
- Magnesium (Mg) has a negative skew of -1.153, indicating that the majority of samples have low magnesium levels, which might suggest that its role in glass composition is less critical than other components.

We have now decided to have a visual representation of the distribution of data points across different ranges of values. Also, after our skewness analysis, histograms visually confirm whether a dataset is skewed to the left or the right. Moreover, a good reason to create a histogram is detecting outliers or extreme values that may affect our result.

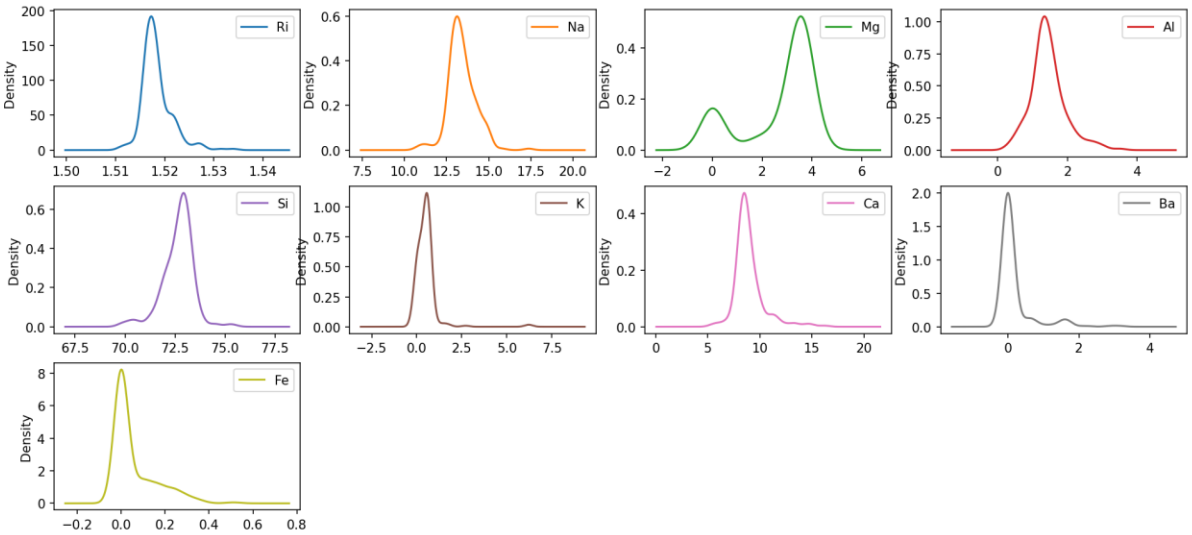
```
print(dataset.hist())
pyplot.figsize = (8,8)
pyplot.savefig('histograms.png', dpi=300)
pyplot.show()
```



From this graph we can notice that Sodium (Na) and Aluminum (Al) appear to have distributions that are closer to normal, though still slightly skewed. This suggests that these elements are present in more consistent amounts across the samples. For chemicals like Potassium (K), Barium (Ba), and Calcium (Ca) we have a strong right skewness in their distributions. This indicates that while most samples have low concentrations, a small number have significantly higher values. Silicon (Si) and Iron (Fe) could potentially have bimodal distributions, where there are two peaks in the data. This may suggest different types of glass compositions. Something important to notice is that this histogram reveal potential outliers, particularly in the distributions for potassium and barium. These outliers could have significant implications for understanding the properties of specific glass types, as they may relate to special formulations.

To get a more nuanced view of the distribution of the chemical properties compared to our histogram, we decided to use a density plot.

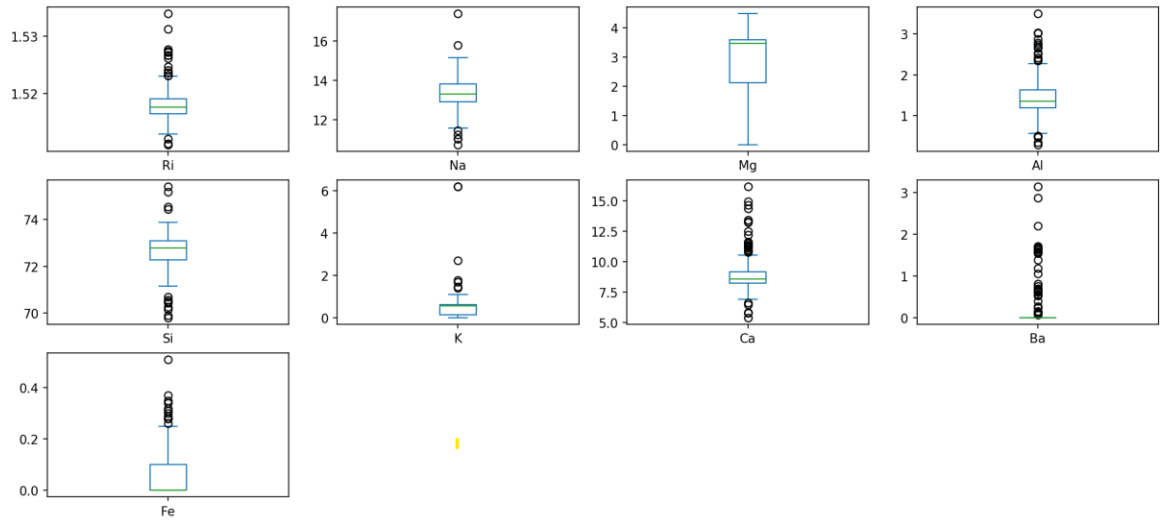
```
dataset.plot(kind='density' , subplots=True, layout=(4,4), sharex=False, figsize=(14,14))
pyplot.show()
```



Properties such as Potassium (K), Barium (Ba) and Calcium (Ca) show pronounced right skewness because the curves have a peak on the left side with a long tail extending to the right. The properties of Sodium (Na) and Magnesium (Mg) display peaks at specific values, suggesting more symmetry compared to K, Ca, and Ba, but still indicate some degree of skewness. Silicon (Si) and Aluminum (Al) have very defined peaks, indicating a high concentration of values around certain points, which can also be indicative of their distribution characteristics. The behavior of the Magnesium (Mg) where the density initially increases to a peak at 0, decreases, and then rises again at around 4, suggests a nuanced distribution of Magnesium values.

Another plot that could be relevant for our analysis could be a Boxplot. This type of graph is excellent for visualizing the spread and skewness of the data, showing also outliers, and giving a big clarity.

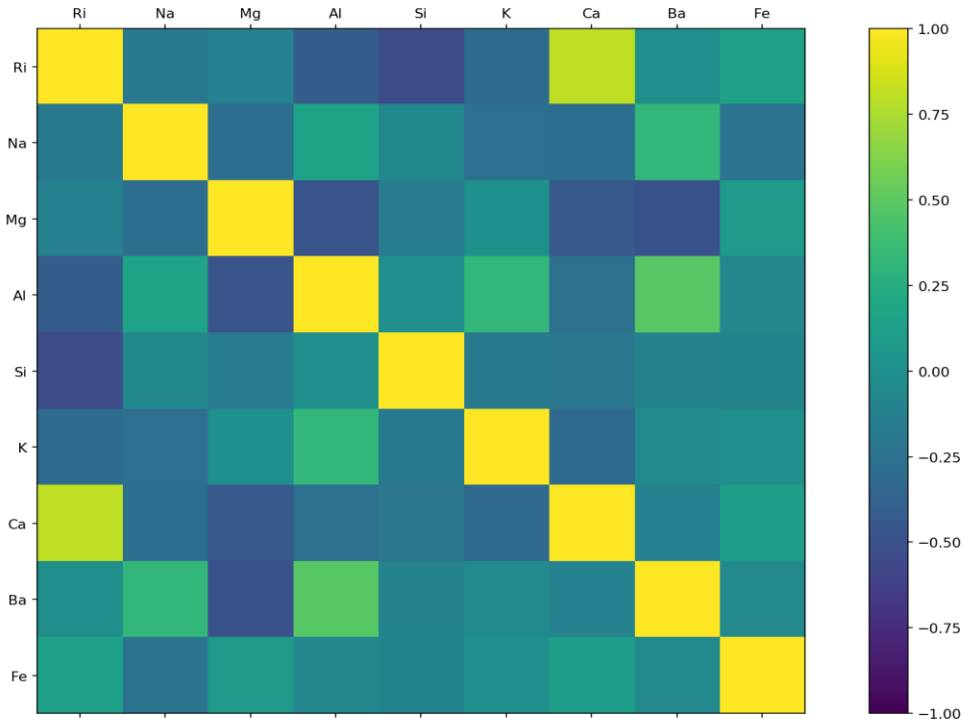
```
dataset.plot(kind='box', subplots=True, layout=(4,4), sharex=False, figsize=(14,14))
pyplot.show()
```



Elements such as Potassium (K), Barium (Ba), Calcium (Ca), and Iron (Fe) show significant right skewness, indicating the majority of samples have low concentrations, with a few samples containing much higher values. Consistent Elements: Silicon (Si) and Refractive Index (Ri) have relatively consistent distributions, showing less variability, which might indicate their consistent role in the glass-making process. Several elements, particularly Potassium (K), Barium (Ba), Aluminum (Al), and Calcium (Ca), show notable outliers. Silicon (Si) and Iron (Fe) may exhibit bimodal distributions, suggesting there are two distinct groups or types of glass that differ in their silicon and iron content. Magnesium (Mg) shows higher variability in its concentration levels across the samples, indicating this element may play a more variable role in the glass formulations.

A great way to visualize relationships between different variables can be done using a correlation matrix analysis. Using it can be also useful to detect dependencies. Some chemical properties may be independend or inversely related. For example, variables with near-zero correlation can help us understand which components behave independently of others.

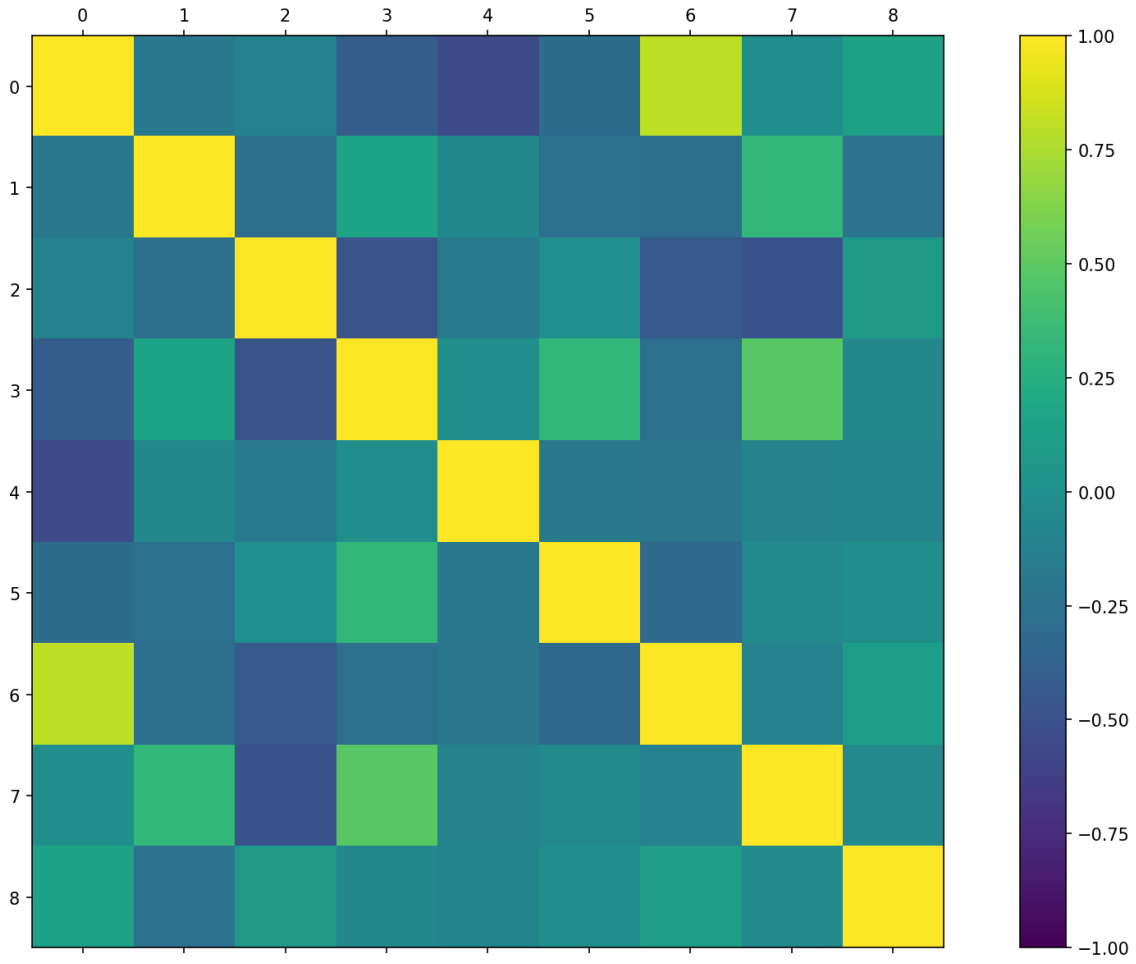
```
fig = pyplot.figure(figsize=(10,8))
ax = fig.add_subplot(111)
cax = ax.matshow(dataset.corr(), vmin=-1, vmax=1)
fig.colorbar(cax)
ticks = numpy.arange(0, 9, 1)
ax.set_xticks(ticks)
ax.set_yticks(ticks)
ax.set_xticklabels(columns_after_drop)
ax.set_yticklabels(columns_after_drop)
pyplot.show()
```



Calcium (Ca) and Refractive Index (Ri) also show a relatively strong positive correlation, suggesting that these variables tend to increase or decrease together. Silicon (Si) and Refractive Index (Ri) display near-zero correlation, as indicated by the dark blue color. This suggests that these two variables are independent of each other and do not exhibit a linear relationship. Negative correlations (closer to -1) are represented by shades of dark blue to green. The positive correlation (closer to 1) is instead represented by the bright yellow squares in the heatmap. Of course, we can notice many yellow boxes that happen when the chemical is compared to itself.

Similarly, we also decided to plot the Correlation Matrix General Plot.

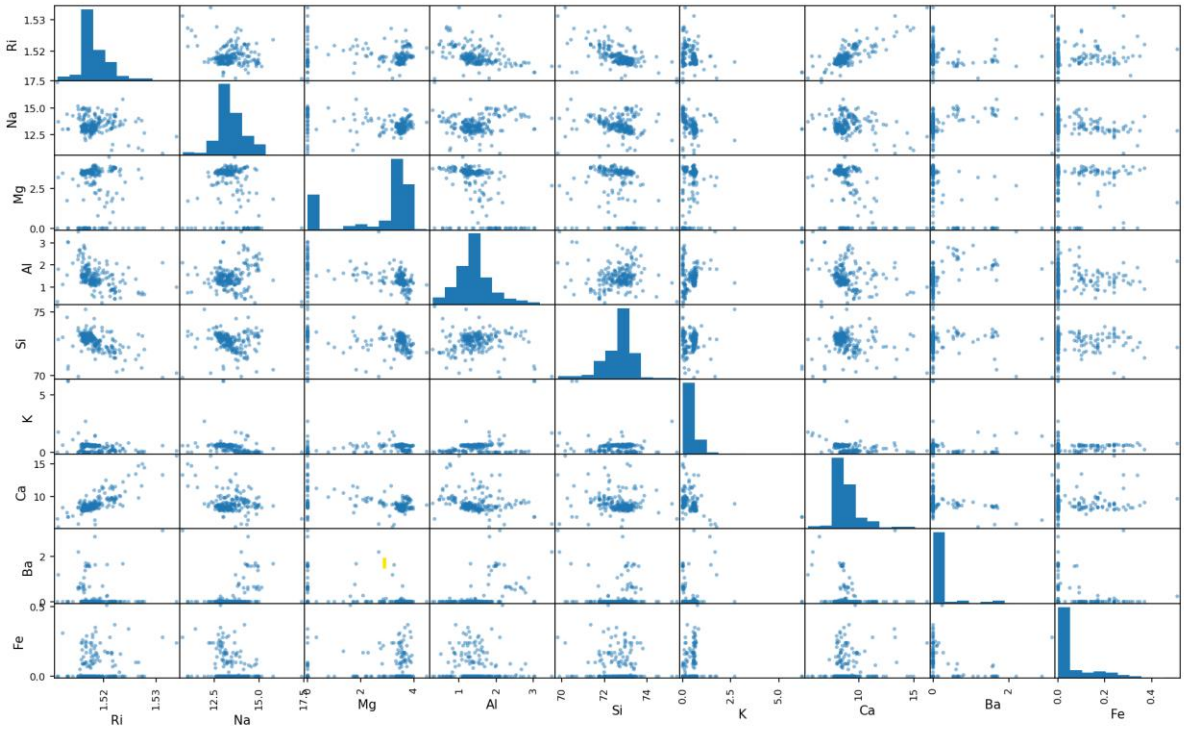
```
fig = pyplot.figure()
ax = fig.add_subplot(111)
cax = ax.matshow(dataset.corr(), vmin=-1, vmax=1)
fig.colorbar(cax)
pyplot.show()
```



This heatmap can be useful to spot general trends, that we have already found before, in a faster way. Otherwise, it is missing labels and clarity compared to the previous one.

With the aim of finding more relationships between every pair of variables we can use a Scatter Matrix. It also includes histograms for each variable along the diagonal. The matrix will also allow us to spot non-linear trends in the data which the correlation matrix might miss since it focuses on linear correlations. This is very useful to use especially with the aim of digging deeper into how variables interact and looking for potential patterns.

```
scatter_matrix(dataset)
pyplot.figure(figsize=(20,18))
pyplot.show()
```



From these graphs we can notice visible outliers, particularly in pairs such as BA and Fe or Ri and K. Other plots show a lot of randomness, for example Si and Fe, which aligns with their near zero correlation in the heat map. Other plots like Ca and Ri show a positive trend, with points showing a mild upward slope, which also aligns with their near one correlation in the heat map.

The next step that we decided to take is scaling the features using the MinMaxScaler.

```
array = dataset.values
X = array[:, 0:9]
scaler = MinMaxScaler(feature_range=(0, 1))
rescaledX = scaler.fit_transform(X)
set_printoptions(precision=3)
print(rescaledX[0:5, :])
```

```
[[0.433 0.438 1.    0.252 0.352 0.01  0.309 0.    0.   ]
 [0.284 0.475 0.802 0.333 0.521 0.077 0.223 0.    0.   ]
 [0.221 0.421 0.791 0.389 0.568 0.063 0.218 0.    0.   ]
 [0.286 0.373 0.822 0.312 0.5   0.092 0.259 0.    0.   ]
 [0.275 0.382 0.806 0.296 0.584 0.089 0.245 0.    0.   ]]
```

Each value is between 0 and 1, representing the scaling of each feature. A value of 1 indicates the maximum value for a that feature across the dataset, while a value of 0 indicates the minimum one. Some columns such as Na, Mg, and Al show a lot of variability suggesting that they might provide more information for analysis rather than the ones showing constant values or have low variance like Ba and Fe. Another thing we might focus on is that Mg is highly concentrated, and we can notice from the 1 and other values close to its maximum. This might tell us that the dataset has a significant focus on this element, and we could find specific trends. If I had to further investifate the relationships between some of these features I would probably focus on the chemicals with significant variability like Na, Mg, and Al.

We now decided to standardize our dataset using StandardScaler. The standardization transforms the features to have a mean of 0 and a standard deviation of 1.

```
X = array[:, 0:9]
scaler_standard = StandardScaler().fit(X)
rescaled_standardX = scaler_standard.transform(X)
set_printoptions(precision=3)
print(rescaled_standardX[0:5, :])
```

```
[[ 0.873  0.285  1.255 -0.692 -1.127 -0.672 -0.146 -0.353 -0.586]
 [-0.249  0.592  0.636 -0.17   0.102 -0.026 -0.794 -0.353 -0.586]
 [-0.721  0.15   0.601  0.191  0.439 -0.165 -0.829 -0.353 -0.586]
 [-0.233 -0.243  0.699 -0.311 -0.053  0.112 -0.519 -0.353 -0.586]
 [-0.312 -0.169  0.65  -0.411  0.555  0.081 -0.625 -0.353 -0.586]]
```

For example, in the first sample, we can see that we have a high value of 0.0873. This indicates that it has a significantly higher refractive index. When we see negative values like Aluminum it suggests having a lower concentration than average. This numbers are

very relevant because they will impact the final property of glass such as viscosity and melting temperature. For example, the presence of magnesium or aluminum in general improves the durability of glass.

When dealing with dataset where the scale of the feature varies significantly it's a good idea to use a normalizer.

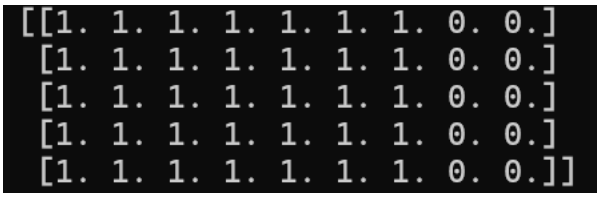
```
X = array[:, 0:9]
scaler = Normalizer().fit(X)
normalizedX = scaler.transform(X)
set_printoptions(precision=3)
print(normalizedX[0:5, :])
```

```
[[2.062e-02 1.850e-01 6.088e-02 1.492e-02 9.733e-01 8.136e-04 1.186e-01
  0.000e+00 0.000e+00]
 [2.035e-02 1.863e-01 4.827e-02 1.824e-02 9.753e-01 6.437e-03 1.050e-01
  0.000e+00 0.000e+00]
 [2.028e-02 1.810e-01 4.749e-02 2.060e-02 9.764e-01 5.217e-03 1.041e-01
  0.000e+00 0.000e+00]
 [2.040e-02 1.776e-01 4.961e-02 1.734e-02 9.762e-01 7.663e-03 1.105e-01
  0.000e+00 0.000e+00]
 [2.028e-02 1.773e-01 4.838e-02 1.657e-02 9.766e-01 7.350e-03 1.078e-01
  0.000e+00 0.000e+00]]
```

Something that we noticed immediately is that the normalized values of potassium are significantly higher than the other elements. This can suggest that potassium is the primary component in these glasses' composition. Other chemicals like Magnesium show that while they are present in moderate quantities, they play a crucial role in the glass structure. Other elements such as sodium and silicon are present at relatively low levels, suggesting that these elements may be less critical in the specific composition. We can also notice an absence of Calcium for example, its absence suggests that maybe the glass in meant to be clear with no color or impurities. It is important to have a consistency in the glass composition to ensure a good final product.

A good idea is now to use a Binarizer with the goal of categorizing samples based on the presence or absence of certain chemicals.

```
binarizer = Binarizer(threshold=0.0).fit(X)
binaryX = binarizer.transform(X)
set_printoptions(precision=3)
print(binaryX[0:5, :])
```



Basically, any value greater than 0.0 is automatically converted to 1, while values less than 0.0 are converted to 0. This is just a simplified view of the composition of the glass samples.