In [1]:
```
### ID: 09893014
### Name: Anisjon Berdiev

#*****************************************************************************************
# Req: 5 parts
#part 1: Load the dataset

from inspect import stack
import pandas as pd
import pickle
import numpy as np
from sklearn import datasets


#Visuzlaztion
import seaborn as sns
sns.set(style='whitegrid')
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.preprocessing import scale

# machine learning techniques
from sklearn.decomposition import PCA
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn import tree
from sklearn.svm import SVC
from xgboost import XGBClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import confusion_matrix

###
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from sklearn.multiclass import OneVsRestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.metrics import classification_report
from sklearn.linear_model import LinearRegression

from __future__ import print_function
from ipywidgets import interact, interactive, fixed, interact_manual
import ipywidgets as widgets

import warnings
warnings.filterwarnings('ignore')


#*****************************************************************
```

In [2]:
```
# Part One: Load a dataset and Look at the summary of the dataset
df = pd.read_csv("glass_hw.csv")
df.head()

shape = df.shape
print(shape)   #2 shape of df
```

```
columns = df.columns
print(columns)


for i in list(df):
    print(df[i].tolist())     #3 date columns values

result = df.dtypes            #4data types of each column
print("Data types: ")
print(result)

stats_numeric = df.describe().astype(int)   #5 basic statistics of all numerical colum
print(stats_numeric)
print(df.info())

#*****************************************************************
```

```
(214, 11)
Index(['Id_number', 'RI', 'Na', 'Mg', 'Al', 'Si', 'K', 'Ca', 'Ba', 'Fe',
       'Type_of_glass'],
      dtype='object')
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 2
4, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 4
5, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 6
6, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 8
7, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106,
107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123,
124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140,
141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157,
158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174,
175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191,
192, 193, 194, 195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207, 208,
209, 210, 211, 212, 213, 214]
[1.52101, 1.51761, 1.51618, 1.51766, 1.51742, 1.51596, 1.51743, 1.51756, 1.51918, 1.5
1755, 1.51571, 1.51763, 1.51589, 1.51748, 1.51763, 1.51761, 1.51784, 1.52196, 1.5191
1, 1.51735, 1.5175, 1.51966, 1.51736, 1.51751, 1.5172, 1.51764, 1.51793, 1.51721, 1.5
1768, 1.51784, 1.51768, 1.51747, 1.51775, 1.51753, 1.51783, 1.51567, 1.51909, 1.5179
7, 1.52213, 1.52213, 1.51793, 1.51755, 1.51779, 1.5221, 1.51786, 1.519, 1.51869, 1.52
667, 1.52223, 1.51898, 1.5232, 1.51926, 1.51808, 1.51837, 1.51778, 1.51769, 1.51215,
1.51824, 1.51754, 1.51754, 1.51905, 1.51977, 1.52172, 1.52227, 1.52172, 1.52099, 1.52
152, 1.52152, 1.52152, 1.523, 1.51574, 1.51848, 1.51593, 1.51631, 1.51596, 1.5159, 1.
51645, 1.51627, 1.51613, 1.5159, 1.51592, 1.51593, 1.51646, 1.51594, 1.51409, 1.5162
5, 1.51569, 1.51645, 1.51618, 1.5164, 1.51841, 1.51605, 1.51588, 1.5159, 1.51629, 1.5
186, 1.51841, 1.51743, 1.51689, 1.51811, 1.51655, 1.5173, 1.5182, 1.52725, 1.5241, 1.
52475, 1.53125, 1.53393, 1.52222, 1.51818, 1.52664, 1.52739, 1.52777, 1.51892, 1.5184
7, 1.51846, 1.51829, 1.51708, 1.51673, 1.51652, 1.51844, 1.51663, 1.51687, 1.51707,
1.52177, 1.51872, 1.51667, 1.52081, 1.52068, 1.5202, 1.52177, 1.52614, 1.51813, 1.51
8, 1.51811, 1.51789, 1.51806, 1.51711, 1.51674, 1.51674, 1.5169, 1.51851, 1.51662, 1.
51709, 1.5166, 1.51839, 1.51769, 1.5161, 1.5167, 1.51643, 1.51665, 1.52127, 1.51779,
1.5161, 1.51694, 1.51646, 1.51655, 1.52121, 1.51776, 1.51796, 1.51832, 1.51934, 1.522
11, 1.51514, 1.51915, 1.52171, 1.52151, 1.51969, 1.51666, 1.51994, 1.52369, 1.51316,
1.51321, 1.52043, 1.52058, 1.52119, 1.51905, 1.51937, 1.51829, 1.51852, 1.51299, 1.51
888, 1.51916, 1.51969, 1.51115, 1.51131, 1.51838, 1.52315, 1.52247, 1.52365, 1.51613,
1.51602, 1.51623, 1.51719, 1.51683, 1.51545, 1.51556, 1.51727, 1.51531, 1.51609, 1.51
508, 1.51653, 1.51514, 1.51658, 1.51617, 1.51732, 1.51645, 1.51831, 1.5164, 1.51623,
1.51685, 1.52065, 1.51651, 1.51711]
[13.64, 13.89, 13.53, 13.21, 13.27, 12.79, 13.3, 13.15, 14.04, 13.0, 12.72, 12.8, 12.
88, 12.86, 12.61, 12.81, 12.68, 14.36, 13.9, 13.02, 12.82, 14.77, 12.78, 12.81, 13.3
8, 12.98, 13.21, 12.87, 12.56, 13.08, 12.65, 12.84, 12.85, 12.57, 12.69, 13.29, 13.8
9, 12.74, 14.21, 14.21, 12.79, 12.71, 13.21, 13.73, 12.73, 13.49, 13.19, 13.99, 13.2
1, 13.58, 13.72, 13.2, 13.43, 13.14, 13.21, 12.45, 12.99, 12.87, 13.48, 13.39, 13.6,
13.81, 13.51, 14.17, 13.48, 13.69, 13.05, 13.05, 13.12, 13.31, 14.86, 13.64, 13.09, 1
3.34, 13.02, 13.02, 13.44, 13.0, 13.92, 12.82, 12.86, 13.25, 13.41, 13.09, 14.25, 13.
36, 13.24, 13.4, 13.01, 12.55, 12.93, 12.9, 13.12, 13.24, 12.71, 13.36, 13.02, 12.2,
12.67, 12.96, 12.75, 12.35, 12.62, 13.8, 13.83, 11.45, 10.73, 12.3, 14.43, 13.72, 11.
23, 11.02, 12.64, 13.46, 13.1, 13.41, 13.24, 13.72, 13.3, 13.56, 13.25, 12.93, 13.23,
13.48, 13.2, 12.93, 12.94, 13.78, 13.55, 13.98, 13.75, 13.7, 13.43, 13.71, 13.33, 13.
19, 13.0, 12.89, 12.79, 12.87, 13.33, 13.2, 12.85, 13.0, 12.99, 12.85, 13.65, 13.33,
13.24, 12.16, 13.14, 14.32, 13.64, 13.42, 12.86, 13.04, 13.41, 14.03, 13.53, 13.5, 1
3.33, 13.64, 14.19, 14.01, 12.73, 11.56, 11.03, 12.64, 12.86, 13.27, 13.44, 13.02, 1
3.0, 13.38, 12.85, 12.97, 14.0, 13.79, 14.46, 14.09, 14.4, 14.99, 14.15, 14.56, 17.3
8, 13.69, 14.32, 13.44, 14.86, 15.79, 13.88, 14.85, 14.2, 14.75, 14.56, 14.14, 13.87,
14.7, 14.38, 15.01, 15.15, 11.95, 14.85, 14.8, 14.95, 14.95, 14.94, 14.39, 14.37, 14.
14, 14.92, 14.36, 14.38, 14.23]
[4.49, 3.6, 3.55, 3.69, 3.62, 3.61, 3.6, 3.61, 3.58, 3.6, 3.46, 3.66, 3.43, 3.56, 3.5
9, 3.54, 3.67, 3.85, 3.73, 3.54, 3.55, 3.75, 3.62, 3.57, 3.5, 3.54, 3.48, 3.48, 3.52,
3.49, 3.56, 3.5, 3.48, 3.47, 3.54, 3.45, 3.53, 3.48, 3.82, 3.82, 3.5, 3.42, 3.39, 3.8
```

```
4, 3.43, 3.48, 3.37, 3.7, 3.77, 3.35, 3.72, 3.33, 2.87, 2.84, 2.81, 2.71, 3.47, 3.48,
3.74, 3.66, 3.62, 3.58, 3.86, 3.81, 3.74, 3.59, 3.65, 3.65, 3.58, 3.58, 3.67, 3.87,
3.59, 3.57, 3.56, 3.58, 3.61, 3.58, 3.52, 3.52, 3.52, 3.45, 3.55, 3.52, 3.09, 3.58,
3.49, 3.49, 3.5, 3.48, 3.74, 3.44, 3.41, 3.34, 3.33, 3.43, 3.62, 3.25, 2.88, 2.96, 2.
85, 2.72, 2.76, 3.15, 2.9, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 3.83, 3.97, 3.89,
3.9, 3.68, 3.64, 3.57, 3.76, 3.54, 3.54, 3.48, 3.68, 3.66, 3.61, 2.28, 2.09, 1.35, 1.
01, 0.0, 3.98, 3.93, 3.85, 3.9, 3.8, 3.62, 3.52, 3.56, 3.54, 3.63, 3.51, 3.47, 3.18,
3.67, 3.66, 3.53, 3.57, 3.52, 3.45, 3.9, 3.65, 3.4, 3.58, 3.4, 3.39, 3.76, 3.41, 3.3
6, 3.34, 3.54, 3.78, 2.68, 1.85, 1.88, 1.71, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.61,
0.33, 2.39, 2.41, 2.24, 2.19, 1.74, 0.78, 0.0, 0.0, 0.0, 3.2, 3.26, 3.34, 2.2, 1.83,
1.78, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
[1.1, 1.36, 1.54, 1.29, 1.24, 1.62, 1.14, 1.05, 1.37, 1.36, 1.56, 1.27, 1.4, 1.27, 1.
31, 1.23, 1.16, 0.89, 1.18, 1.69, 1.49, 0.29, 1.29, 1.35, 1.15, 1.21, 1.41, 1.33, 1.4
3, 1.28, 1.3, 1.14, 1.23, 1.38, 1.34, 1.21, 1.32, 1.35, 0.47, 0.47, 1.12, 1.2, 1.33,
0.72, 1.19, 1.35, 1.18, 0.71, 0.79, 1.23, 0.51, 1.28, 1.19, 1.28, 1.29, 1.29, 1.12,
1.29, 1.17, 1.19, 1.11, 1.32, 0.88, 0.78, 0.9, 1.12, 0.87, 0.87, 0.9, 0.82, 1.74, 1.2
7, 1.52, 1.57, 1.54, 1.51, 1.54, 1.54, 1.25, 1.9, 2.12, 1.43, 1.25, 1.55, 2.08, 1.49,
1.47, 1.52, 1.48, 1.87, 1.11, 1.45, 1.58, 1.47, 1.49, 1.43, 1.06, 1.16, 1.71, 1.43,
1.44, 1.63, 0.83, 0.66, 1.17, 1.88, 2.1, 1.0, 1.0, 0.56, 0.77, 0.75, 0.67, 1.26, 1.1
9, 1.33, 1.41, 1.81, 1.53, 1.47, 1.32, 1.62, 1.48, 1.71, 1.15, 1.56, 1.26, 1.43, 1.6
7, 1.63, 1.36, 1.36, 1.18, 1.54, 1.25, 1.3, 1.08, 1.57, 1.54, 1.64, 1.61, 1.07, 1.44,
1.79, 1.23, 1.24, 1.11, 1.34, 1.38, 1.35, 1.76, 0.83, 0.65, 1.22, 1.31, 1.26, 1.28,
0.58, 1.52, 1.63, 1.54, 0.75, 0.91, 3.5, 1.86, 1.56, 1.56, 1.65, 1.83, 1.76, 1.58, 3.
04, 3.02, 1.4, 2.17, 1.51, 1.56, 1.19, 1.62, 1.66, 1.54, 1.74, 2.09, 0.56, 0.34, 1.8
1, 2.22, 1.23, 2.06, 1.31, 1.79, 2.38, 2.79, 2.0, 1.98, 2.68, 2.54, 2.34, 2.66, 2.51,
2.25, 1.19, 2.42, 1.99, 2.27, 1.8, 1.87, 1.82, 2.74, 2.88, 1.99, 2.02, 1.94, 2.08]
[71.78, 72.73, 72.99, 72.61, 73.08, 72.97, 73.09, 73.24, 72.08, 72.99, 73.2, 73.01, 7
3.28, 73.21, 73.29, 73.24, 73.11, 71.36, 72.12, 72.73, 72.75, 72.02, 72.79, 73.02, 7
2.85, 73.0, 72.64, 73.04, 73.15, 72.86, 73.08, 73.27, 72.97, 73.39, 72.95, 72.74, 71.
81, 72.96, 71.77, 71.77, 73.03, 73.2, 72.76, 71.76, 72.95, 71.95, 72.72, 71.57, 71.9
9, 72.08, 71.75, 72.36, 72.84, 72.85, 72.98, 73.7, 72.98, 72.95, 72.99, 72.79, 72.64,
71.72, 71.79, 71.35, 72.01, 71.96, 72.22, 72.32, 72.2, 71.99, 71.87, 71.96, 73.1, 72.
87, 73.11, 73.12, 72.39, 72.83, 72.88, 72.86, 72.66, 73.17, 72.81, 72.87, 72.28, 72.7
2, 73.25, 72.65, 72.89, 73.23, 72.28, 73.06, 73.26, 73.1, 73.28, 72.26, 72.34, 73.55,
73.21, 72.92, 73.27, 72.87, 73.81, 70.57, 71.15, 72.19, 69.81, 70.16, 72.67, 74.45, 7
3.21, 73.08, 72.02, 72.55, 72.44, 72.38, 72.33, 72.06, 72.53, 72.45, 72.4, 72.96, 72.
84, 72.52, 72.75, 72.51, 72.75, 71.99, 72.18, 71.76, 72.19, 71.24, 72.49, 71.81, 72.7
8, 72.33, 73.07, 72.96, 73.36, 73.14, 72.54, 72.83, 73.01, 72.72, 72.97, 72.57, 72.7
7, 72.67, 72.7, 72.89, 72.48, 71.5, 73.0, 72.69, 72.61, 73.01, 72.64, 71.79, 72.04, 7
1.94, 72.14, 72.65, 71.36, 69.89, 72.69, 72.86, 73.44, 73.75, 73.88, 73.03, 72.22, 7
0.48, 70.7, 72.25, 72.18, 73.39, 72.37, 72.76, 72.38, 72.67, 74.55, 72.5, 72.74, 73.4
8, 75.41, 72.81, 71.25, 72.38, 70.26, 70.43, 73.1, 73.28, 73.46, 73.02, 73.29, 73.39,
73.23, 73.28, 73.1, 73.05, 73.5, 75.18, 73.72, 73.11, 73.3, 72.99, 73.11, 72.86, 72.8
5, 72.61, 73.06, 73.42, 73.61, 73.36]
[0.06, 0.48, 0.39, 0.57, 0.55, 0.64, 0.58, 0.57, 0.56, 0.57, 0.67, 0.6, 0.69, 0.54,
0.58, 0.58, 0.61, 0.15, 0.06, 0.54, 0.54, 0.03, 0.59, 0.62, 0.5, 0.65, 0.59, 0.56, 0.
57, 0.6, 0.61, 0.56, 0.61, 0.6, 0.57, 0.56, 0.51, 0.64, 0.11, 0.11, 0.64, 0.59, 0.59,
0.17, 0.62, 0.55, 0.57, 0.02, 0.13, 0.59, 0.09, 0.6, 0.55, 0.55, 0.51, 0.56, 0.62, 0.
6, 0.59, 0.57, 0.14, 0.12, 0.23, 0.0, 0.18, 0.09, 0.19, 0.19, 0.23, 0.12, 0.16, 0.54,
0.67, 0.61, 0.72, 0.69, 0.66, 0.61, 0.37, 0.69, 0.69, 0.61, 0.68, 0.68, 1.1, 0.45, 0.
38, 0.67, 0.6, 0.63, 0.64, 0.44, 0.07, 0.39, 0.67, 0.51, 0.64, 0.62, 0.73, 0.6, 0.57,
0.7, 0.35, 0.08, 0.08, 0.81, 0.58, 0.12, 0.1, 0.0, 0.0, 0.0, 0.06, 0.57, 0.6, 0.51,
0.55, 0.64, 0.65, 0.64, 0.58, 0.64, 0.56, 0.62, 0.54, 0.58, 0.56, 0.49, 0.53, 0.39,
0.33, 0.19, 0.58, 0.54, 0.52, 0.55, 0.56, 0.61, 0.66, 0.65, 0.68, 0.57, 0.68, 0.66,
0.58, 0.62, 0.11, 0.56, 0.56, 0.57, 0.6, 0.0, 0.06, 0.59, 0.61, 0.52, 0.52, 0.11, 0.5
8, 0.57, 0.56, 0.16, 0.23, 1.68, 0.6, 0.47, 0.58, 0.38, 0.97, 0.47, 0.32, 6.21, 6.21,
0.33, 0.76, 0.13, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.76, 1.46, 0.6, 0.76,
0.31, 0.0, 0.0, 0.04, 0.0, 0.0, 0.08, 0.14, 0.0, 0.04, 0.05, 0.0, 2.7, 0.0, 0.0, 0.0,
0.0, 0.0, 1.41, 0.0, 0.08, 0.0, 0.0, 0.0, 0.0]
```

```
[8.75, 7.83, 7.78, 8.22, 8.07, 8.07, 8.17, 8.24, 8.3, 8.4, 8.09, 8.56, 8.05, 8.38, 8.
5, 8.39, 8.7, 9.15, 8.89, 8.44, 8.52, 9.0, 8.7, 8.59, 8.43, 8.53, 8.43, 8.43, 8.54,
8.49, 8.69, 8.55, 8.56, 8.55, 8.75, 8.57, 8.78, 8.68, 9.57, 9.57, 8.77, 8.64, 8.59,
9.74, 8.76, 9.0, 8.83, 9.82, 10.02, 8.91, 10.06, 9.14, 9.03, 9.07, 9.02, 9.06, 8.35,
8.43, 8.03, 8.27, 8.76, 8.67, 9.54, 9.69, 9.61, 9.4, 9.85, 9.85, 9.82, 10.17, 7.36,
8.32, 7.83, 7.89, 7.9, 7.96, 8.03, 8.04, 7.94, 7.97, 7.97, 7.86, 8.1, 8.05, 7.08, 8.2
1, 8.03, 8.08, 8.12, 8.08, 8.96, 8.27, 8.39, 8.22, 8.24, 8.6, 9.13, 8.9, 8.54, 8.79,
8.79, 9.23, 9.42, 11.64, 10.79, 13.24, 13.3, 16.19, 11.52, 10.99, 14.68, 14.96, 14.4,
8.21, 8.43, 8.28, 8.31, 7.88, 8.03, 7.96, 8.42, 8.03, 8.1, 7.99, 8.52, 8.55, 8.6, 9.8
5, 9.57, 10.56, 11.14, 13.44, 8.15, 8.21, 8.12, 8.44, 8.38, 8.11, 7.9, 7.99, 8.11, 8.
41, 8.23, 8.18, 8.81, 8.68, 8.6, 8.33, 8.44, 8.53, 8.38, 9.49, 8.93, 8.32, 8.79, 8.5
8, 8.65, 9.65, 8.79, 8.81, 8.99, 8.89, 9.14, 5.87, 10.09, 11.41, 11.62, 11.53, 10.17,
11.32, 12.24, 6.96, 6.93, 12.5, 9.7, 11.27, 9.57, 9.77, 9.26, 9.32, 7.59, 9.95, 10.8
8, 11.22, 6.65, 5.43, 5.79, 8.83, 9.76, 8.61, 8.67, 8.76, 9.04, 8.53, 8.52, 9.07, 9.4
1, 8.95, 9.08, 8.83, 8.34, 8.93, 8.39, 8.28, 8.71, 8.61, 8.67, 6.47, 9.45, 9.18, 8.4,
8.44, 8.48, 8.62]
[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.09, 0.0,
0.0, 0.0, 0.11, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.69, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.14, 0.11, 0.
0, 0.0, 0.0, 0.0, 0.0, 3.15, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.27, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.09, 0.06, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.
0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.15, 0.0, 2.2, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.24, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
1.19, 1.63, 0.0, 0.0, 1.68, 0.76, 0.64, 0.4, 1.59, 1.57, 0.61, 0.81, 0.66, 0.64, 0.5
3, 0.63, 0.0, 0.56, 1.71, 0.67, 1.55, 1.38, 2.88, 0.54, 1.06, 1.59, 1.64, 1.57, 1.67]
[0.0, 0.0, 0.0, 0.0, 0.0, 0.26, 0.0, 0.0, 0.0, 0.11, 0.24, 0.0, 0.24, 0.17, 0.0, 0.0,
0.0, 0.0, 0.0, 0.07, 0.19, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.14, 0.0, 0.
22, 0.06, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.3, 0.0, 0.16, 0.1, 0.0,
0.0, 0.16, 0.11, 0.0, 0.0, 0.09, 0.24, 0.31, 0.0, 0.0, 0.11, 0.0, 0.0, 0.11, 0.0, 0.0
7, 0.0, 0.17, 0.17, 0.16, 0.03, 0.12, 0.32, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.14, 0.0,
0.0, 0.0, 0.0, 0.09, 0.0, 0.0, 0.0, 0.1, 0.0, 0.09, 0.22, 0.0, 0.19, 0.0, 0.0, 0.0,
0.15, 0.24, 0.0, 0.0, 0.22, 0.0, 0.2, 0.0, 0.0, 0.34, 0.28, 0.24, 0.08, 0.0, 0.0, 0.
0, 0.0, 0.14, 0.0, 0.0, 0.1, 0.0, 0.29, 0.0, 0.0, 0.21, 0.0, 0.0, 0.0, 0.12, 0.0, 0.1
7, 0.17, 0.18, 0.0, 0.1, 0.0, 0.15, 0.0, 0.28, 0.12, 0.0, 0.0, 0.0, 0.0, 0.17, 0.25,
0.0, 0.24, 0.35, 0.0, 0.0, 0.1, 0.0, 0.17, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.
09, 0.0, 0.24, 0.37, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.51, 0.2
8, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.09,
0.09, 0.08, 0.07, 0.05, 0.01, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 5, 5, 5, 5, 5,
5, 5, 5, 5, 5, 5, 5, 5, 6, 6, 6, 6, 6, 6, 6, 6, 6, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7,
7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7]
Data types:
Id_number          int64
RI               float64
Na               float64
Mg               float64
Al               float64
Si               float64
K                float64
Ca               float64
```

```
Ba                float64
Fe                float64
Type_of_glass       int64
dtype: object
       Id_number   RI   Na   Mg   Al   Si    K   Ca   Ba   Fe  Type_of_glass
count        214  214  214  214  214  214  214  214  214  214            214
mean         107    1   13    2    1   72    0    8    0    0              2
std           61    0    0    1    0    0    0    1    0    0              2
min            1    1   10    0    0   69    0    5    0    0              1
25%           54    1   12    2    1   72    0    8    0    0              1
50%          107    1   13    3    1   72    0    8    0    0              2
75%          160    1   13    3    1   73    0    9    0    0              3
max          214    1   17    4    3   75    6   16    3    0              7
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 214 entries, 0 to 213
Data columns (total 11 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Id_number      214 non-null    int64  
 1   RI             214 non-null    float64
 2   Na             214 non-null    float64
 3   Mg             214 non-null    float64
 4   Al             214 non-null    float64
 5   Si             214 non-null    float64
 6   K              214 non-null    float64
 7   Ca             214 non-null    float64
 8   Ba             214 non-null    float64
 9   Fe             214 non-null    float64
 10  Type_of_glass  214 non-null    int64  
dtypes: float64(9), int64(2)
memory usage: 18.5 KB
None
```

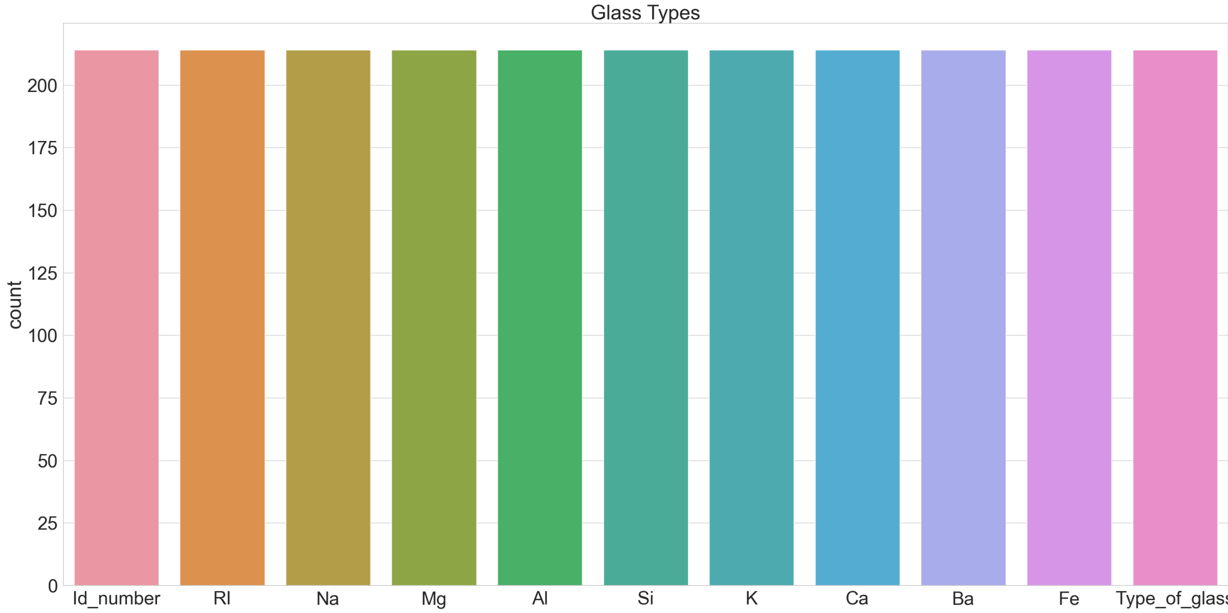
In [3]:

```python
print('*'*100)

#Part Two: EDA (Exploratory Data Analysis) of the dataset
#2.1 - Bars
sns.set(style="whitegrid", font_scale=2.8)
plt.subplots(figsize = (35,17))
sns.countplot(data=df).set_title('Glass Types')             #wrong
```

```
****************************************************************************************************
```

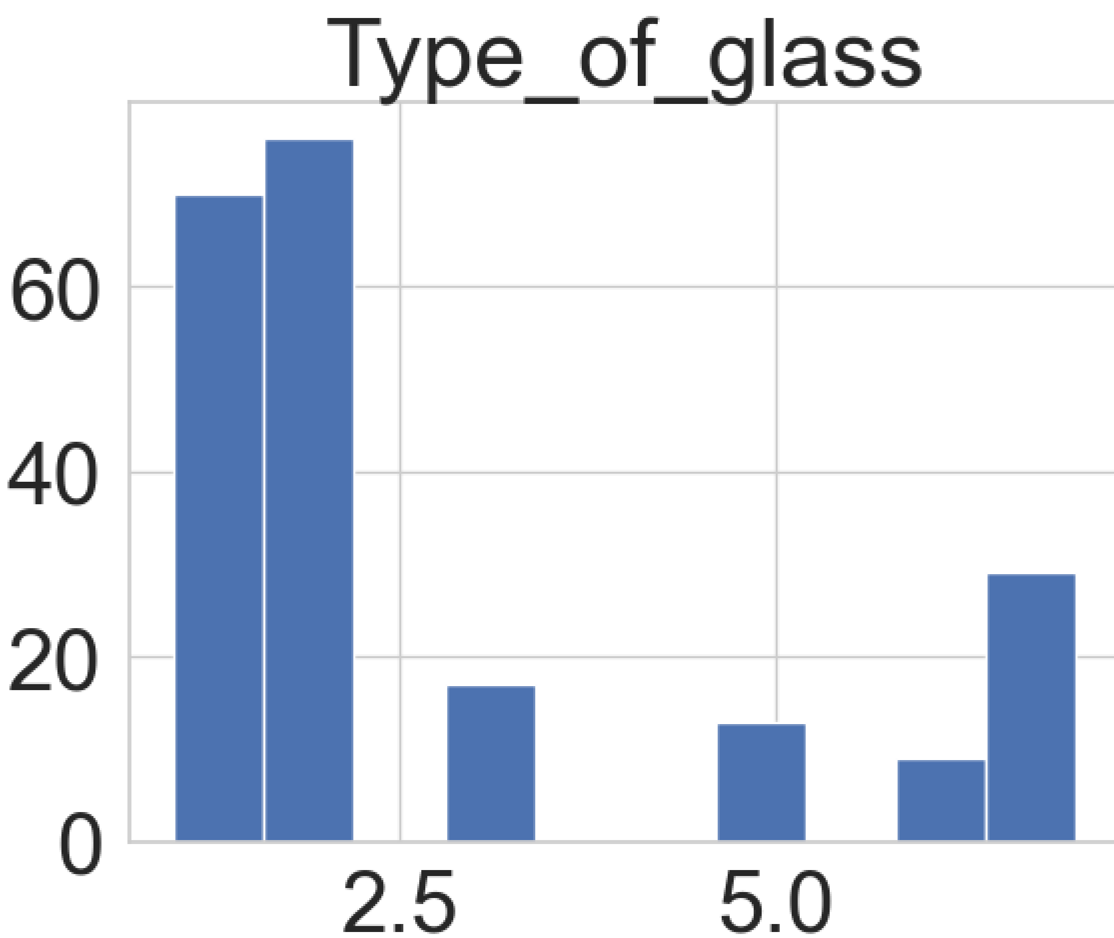
Out[3]:

```
Text(0.5, 1.0, 'Glass Types')
```



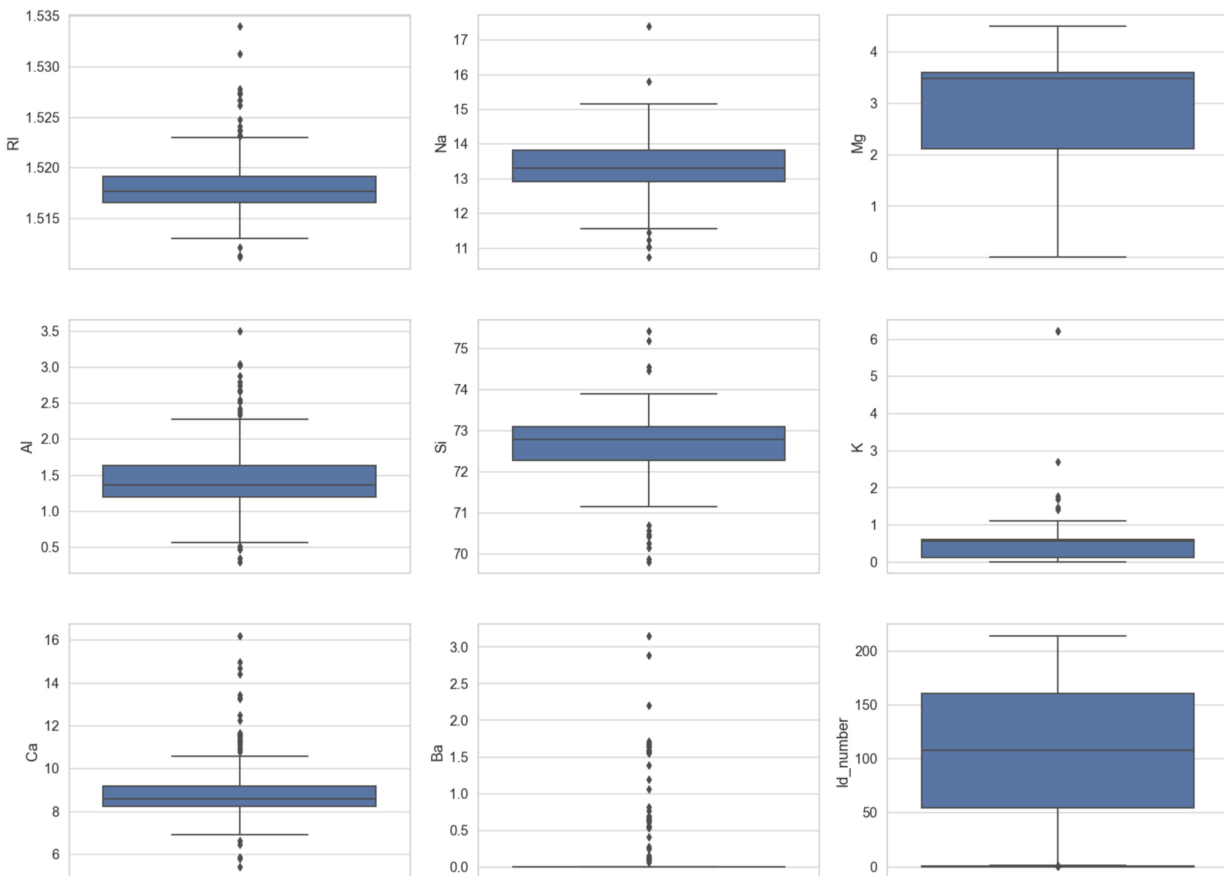
In [4]:
```
#2.2 - Histogram: distribution of class attributes: Type_of_glass
df.hist(column= 'Type_of_glass')
```

Out[4]:
```
array([[<AxesSubplot:title={'center':'Type_of_glass'}>]], dtype=object)
```



In [5]:
```
#2.3 - Box Plot: the IQR (interquartile range) of features and class attributes.
sns.set(style="whitegrid", font_scale=1.2)
plt.subplots(figsize = (20,15))
plt.subplot(3,3,1)
```

```
sns.boxplot( y='RI', data=df)
plt.subplot(3,3,2)
sns.boxplot(y='Na', data=df)
plt.subplot(3,3,3)
sns.boxplot(y='Mg', data=df)
plt.subplot(3,3,4)
sns.boxplot(y='Al', data=df)
plt.subplot(3,3,5)
sns.boxplot(y='Si', data=df)
plt.subplot(3,3,6)
sns.boxplot(y='K', data=df)
plt.subplot(3,3,7)
sns.boxplot(y='Ca', data=df)
plt.subplot(3,3,8)
sns.boxplot(y='Ba', data=df)
plt.subplot(3,3,9)
sns.boxplot(y='Fe', data=df)
sns.boxplot(y='Id_number', data=df)
plt.show()
```

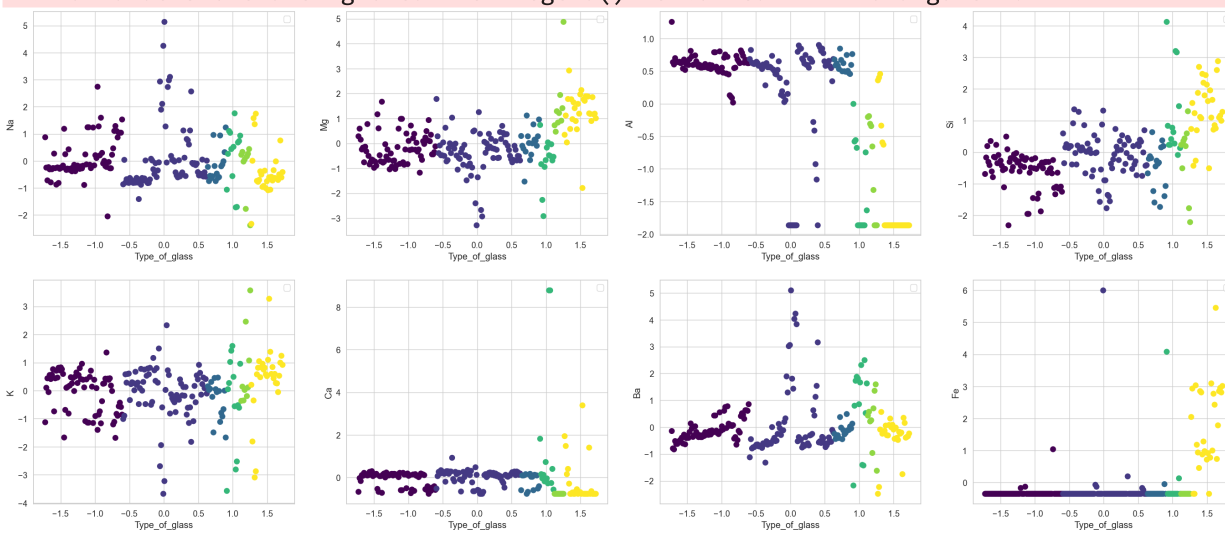


In [6]:

```
#2.4 - Scatter Plot: show different colors for different types of glass (e.g.Si vs. Ty

Y = df["Type_of_glass"].values
del df["Type_of_glass"]
X = df.values
X_scaled = scale(X)
Y = Y.astype(float)

plt.figure(figsize=(34 , 14))

plt.subplot(2 , 4, 1)
plt.scatter(X_scaled[:,0] , X_scaled[:,1] ,s = 70, c = Y, cmap = "viridis"  )
```

```
plt.legend(loc = 0)
plt.xlabel("Type_of_glass")
plt.ylabel("Na")

plt.subplot(2 , 4, 2)
plt.scatter(X_scaled[:,0] , X_scaled[:,2] ,s = 70, c = Y, cmap = "viridis"  )
plt.legend(loc = 0)
plt.xlabel("Type_of_glass")
plt.ylabel("Mg")

plt.subplot(2 , 4, 3)
plt.scatter(X_scaled[:,0] , X_scaled[:,3] ,s = 70, c = Y, cmap = "viridis"  )
plt.legend(loc = 0)
plt.xlabel("Type_of_glass")
plt.ylabel("Al")

plt.subplot(2 , 4, 4)
plt.scatter(X_scaled[:,0] , X_scaled[:,4] ,s = 70, c = Y, cmap = "viridis"  )
plt.legend(loc = 0)
plt.xlabel("Type_of_glass")
plt.ylabel("Si")

plt.subplot(2 , 4, 5)
plt.scatter(X_scaled[:,0] , X_scaled[:,5] ,s = 70, c = Y, cmap = "viridis"  )
plt.legend(loc = 0)
plt.xlabel("Type_of_glass")
plt.ylabel("K")

plt.subplot(2 , 4, 6)
plt.scatter(X_scaled[:,0] , X_scaled[:,6] ,s = 70, c = Y, cmap = "viridis"  )
plt.legend(loc = 0)
plt.xlabel("Type_of_glass")
plt.ylabel("Ca")

plt.subplot(2 , 4, 7)
plt.scatter(X_scaled[:,0] , X_scaled[:,7] ,s = 70, c = Y, cmap = "viridis"  )
plt.legend(loc = 0)
plt.xlabel("Type_of_glass")
plt.ylabel("Ba")

plt.subplot(2 , 4, 8)
plt.scatter(X_scaled[:,0] , X_scaled[:,8] ,s = 70, c = Y, cmap = "viridis"  )
plt.legend(loc = 0)
plt.xlabel("Type_of_glass")
plt.ylabel("Fe")

plt.show()
```

```
No artists with labels found to put in legend.  Note that artists whose label start w
ith an underscore are ignored when legend() is called with no argument.
No artists with labels found to put in legend.  Note that artists whose label start w
ith an underscore are ignored when legend() is called with no argument.
No artists with labels found to put in legend.  Note that artists whose label start w
ith an underscore are ignored when legend() is called with no argument.
No artists with labels found to put in legend.  Note that artists whose label start w
ith an underscore are ignored when legend() is called with no argument.
No artists with labels found to put in legend.  Note that artists whose label start w
ith an underscore are ignored when legend() is called with no argument.
No artists with labels found to put in legend.  Note that artists whose label start w
ith an underscore are ignored when legend() is called with no argument.
No artists with labels found to put in legend.  Note that artists whose label start w
ith an underscore are ignored when legend() is called with no argument.
No artists with labels found to put in legend.  Note that artists whose label start w
ith an underscore are ignored when legend() is called with no argument.
```



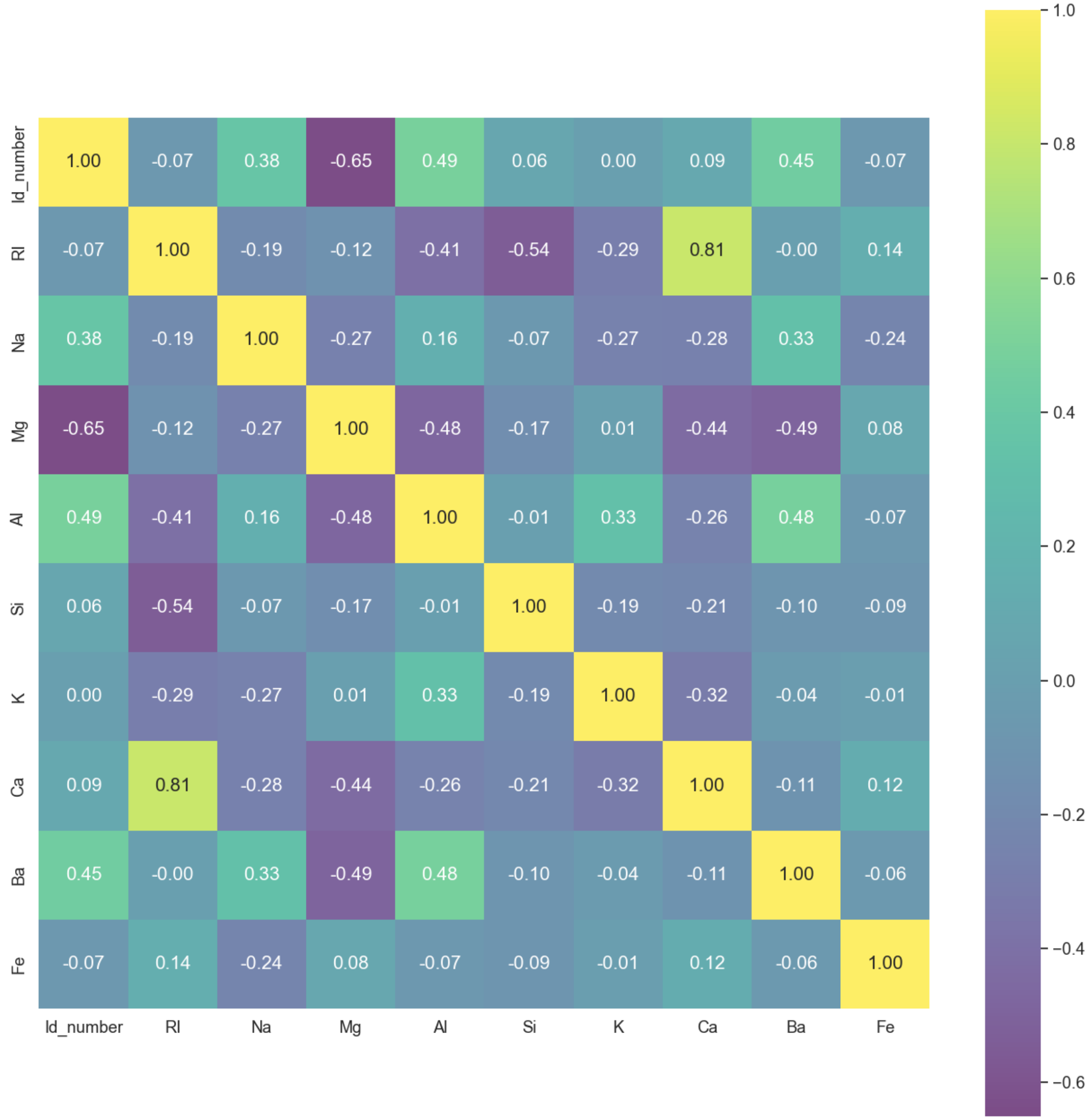
In [7]:

```
#2.5 - Correlation Matrix: correlations among the features and class
corrM = df.corr()

corrM
```

Out[7]:

	Id_number	RI	Na	Mg	Al	Si	K	Ca	
Id_number	1.000000	-0.072209	0.375722	-0.650328	0.490113	0.061232	0.003149	0.090800	0
RI	-0.072209	1.000000	-0.191885	-0.122274	-0.407326	-0.542052	-0.289833	0.810403	-0
Na	0.375722	-0.191885	1.000000	-0.273732	0.156794	-0.069809	-0.266087	-0.275442	0
Mg	-0.650328	-0.122274	-0.273732	1.000000	-0.481799	-0.165927	0.005396	-0.443750	-0
Al	0.490113	-0.407326	0.156794	-0.481799	1.000000	-0.005524	0.325958	-0.259592	0
Si	0.061232	-0.542052	-0.069809	-0.165927	-0.005524	1.000000	-0.193331	-0.208732	-0
K	0.003149	-0.289833	-0.266087	0.005396	0.325958	-0.193331	1.000000	-0.317836	-0
Ca	0.090800	0.810403	-0.275442	-0.443750	-0.259592	-0.208732	-0.317836	1.000000	-0
Ba	0.451001	-0.000386	0.326603	-0.492262	0.479404	-0.102151	-0.042618	-0.112841	1
Fe	-0.072794	0.143010	-0.241346	0.083060	-0.074402	-0.094201	-0.007719	0.124968	-0

In [8]:
```
#2.6 - Heat Map: correlations among the features and class attributes
corr = df.corr()
plt.figure(figsize=(16,16))
sns.heatmap(corr, cbar = True,  square = True, annot=True, fmt= '.2f',annot_kws={'size
            xticklabels=df.columns.values, yticklabels= df.columns.values, alpha = 0.7,
plt.show()
```



In [9]:
```
# Part Three: Train and Build Machine Learning Models, you should
# split the dataset into train and test dataset, and train at least three multiclassif
# machine learning models, such as logistic regression, SVC, KNN,
# Naïve Bayes, Decision Tree Classification, Random Forest
# Classification, etc. AND->>>


# Part Four: Model Evaluation, you should evaluate your trained multiclass classifier
# results, such as classification report (precision, recall, F1 score,
# support) AND ->>>

#Part Five: Make Predictions,
```

```
X_train, X_test, y_train, y_test = train_test_split(X_scaled, Y,train_size=0.75, test_

model = LinearRegression()
model.fit(X_train, y_train)

result = model.predict(X_test)
print(result)


#1 - Logistic Regression
cl = OneVsRestClassifier(LogisticRegression(C =10 , random_state=1367))
cl.fit(X_train , y_train)
y1 = cl.predict(X_test)
print (classification_report(y_test , y1))
print ("Classification Accuracy = " + str(accuracy_score(y_test , y1)))
```

```
[0.6862696  3.26145317 3.12150226 1.0117551  2.41177413 7.41505081
 4.76876973 2.50339691 0.64265563 1.52126319 1.42230861 1.17630563
 3.28061012 6.92375497 5.378381   6.81311628 0.77586657 2.04156889
 7.21341903 2.54675255 2.59984229 2.6724625  2.94846506 1.70882107
 0.382379   6.63541833 2.01314146 1.55602054 2.2122645  4.15899453
 1.56850277 6.86666829 6.95278735 4.21470698 1.31715529 2.0060586
 0.64920897 0.38754561 2.95164776 1.20604674 8.05102344 1.85981456
 4.59669084 2.86058343 5.85884824 2.24745408 5.33398564 2.42093107
 5.8304069  3.23599629 7.1330419  0.95377948 3.1551956  1.73615759]
              precision    recall  f1-score   support

         1.0       1.00      1.00      1.00        14
         2.0       0.64      0.94      0.76        17
         3.0       0.00      0.00      0.00         5
         5.0       0.50      0.25      0.33         4
         6.0       1.00      0.50      0.67         2
         7.0       0.92      0.92      0.92        12

    accuracy                           0.80        54
   macro avg       0.68      0.60      0.61        54
weighted avg       0.74      0.80      0.75        54

Classification Accuracy = 0.7962962962962963
```

In [10]:

```
#2 - Classification with SVC
clsvc = OneVsRestClassifier(SVC(kernel='rbf',C = 100, gamma = 0.1, probability=True,ra
clsvc.fit(X_train , y_train)
y3 = cl.predict(X_test)
print (classification_report(y_test , y3))
print ("Classification Accuracy = " + str(accuracy_score(y_test , y3)))
```

```
              precision    recall  f1-score   support

         1.0       1.00      1.00      1.00        14
         2.0       0.64      0.94      0.76        17
         3.0       0.00      0.00      0.00         5
         5.0       0.50      0.25      0.33         4
         6.0       1.00      0.50      0.67         2
         7.0       0.92      0.92      0.92        12

    accuracy                           0.80        54
   macro avg       0.68      0.60      0.61        54
weighted avg       0.74      0.80      0.75        54

Classification Accuracy = 0.7962962962962963
```

In [11]:

```
#3 - Random Forest
clf = OneVsRestClassifier(RandomForestClassifier(n_estimators=200 ,min_samples_split=2
                                                  min_samples_leaf=5, n_jobs=-1,  ranc
clf.fit(X_train , y_train)
y4 = clf.predict(X_test)
print (classification_report(y_test , y4))
print ("Classification Accuracy = " + str(accuracy_score(y_test , y4)))
```

```
              precision    recall  f1-score   support

         1.0       1.00      1.00      1.00        14
         2.0       1.00      1.00      1.00        17
         3.0       1.00      1.00      1.00         5
         5.0       1.00      1.00      1.00         4
         6.0       1.00      1.00      1.00         2
         7.0       1.00      1.00      1.00        12

    accuracy                           1.00        54
   macro avg       1.00      1.00      1.00        54
weighted avg       1.00      1.00      1.00        54

Classification Accuracy = 1.0
```

In [12]:

```
# Part Five: Make Predictions, you should save your trained multi-class
# classifier model with joblib or pickle object, and then you can load your
# model and make a prediction with new data

#model is saved at glass_class
filename = 'glass_class'
pickle.dump(model, open(filename, 'wb'))
```

In [13]:

```
#making prediction with new model
loaded_model = pickle.load(open(filename, 'rb'))
loaded_model.predict(X_test)
```

Out[13]:

```
array([0.6862696 , 3.26145317, 3.12150226, 1.0117551 , 2.41177413,
       7.41505081, 4.76876973, 2.50339691, 0.64265563, 1.52126319,
       1.42230861, 1.17630563, 3.28061012, 6.92375497, 5.378381  ,
       6.81311628, 0.77586657, 2.04156889, 7.21341903, 2.54675255,
       2.59984229, 2.6724625 , 2.94846506, 1.70882107, 0.382379  ,
       6.63541833, 2.01314146, 1.55602054, 2.2122645 , 4.15899453,
       1.56850277, 6.86666829, 6.95278735, 4.21470698, 1.31715529,
       2.0060586 , 0.64920897, 0.38754561, 2.95164776, 1.20604674,
       8.05102344, 1.85981456, 4.59669084, 2.86058343, 5.85884824,
       2.24745408, 5.33398564, 2.42093107, 5.8304069 , 3.23599629,
       7.1330419 , 0.95377948, 3.1551956 , 1.73615759])
```