Glass Identification

In [24]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
import pickle as pkl
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn import preprocessing
from sklearn.metrics import accuracy_score
```

In [25]:
```
df = pd.read_csv('glass.csv')
```

In [26]:
```
df.head()
```

Out[26]:

	1	1.52101	13.64	4.49	1.10	71.78	0.06	8.75	0.00	0.00.1	1.1
0	2	1.51761	13.89	3.60	1.36	72.73	0.48	7.83	0.0	0.00	1
1	3	1.51618	13.53	3.55	1.54	72.99	0.39	7.78	0.0	0.00	1
2	4	1.51766	13.21	3.69	1.29	72.61	0.57	8.22	0.0	0.00	1
3	5	1.51742	13.27	3.62	1.24	73.08	0.55	8.07	0.0	0.00	1
4	6	1.51596	12.79	3.61	1.62	72.97	0.64	8.07	0.0	0.26	1

In [27]:
```
# Rename specific columns
df = df.rename(columns={'1.52101': 'RI', '13.64': 'Na', '4.49': 'Mg', '1.10': 'Al',

print(df.head())
```

```
   1       RI     Na    Mg    Al     Si     K    Ca   Ba    Fe  Type of glass
0  2  1.51761  13.89  3.60  1.36  72.73  0.48  7.83  0.0  0.00              1
1  3  1.51618  13.53  3.55  1.54  72.99  0.39  7.78  0.0  0.00              1
2  4  1.51766  13.21  3.69  1.29  72.61  0.57  8.22  0.0  0.00              1
3  5  1.51742  13.27  3.62  1.24  73.08  0.55  8.07  0.0  0.00              1
4  6  1.51596  12.79  3.61  1.62  72.97  0.64  8.07  0.0  0.26              1
```

In [28]:
```
df.head()
```

Out[28]:

	1	RI	Na	Mg	Al	Si	K	Ca	Ba	Fe	Type of glass
0	2	1.51761	13.89	3.60	1.36	72.73	0.48	7.83	0.0	0.00	1
1	3	1.51618	13.53	3.55	1.54	72.99	0.39	7.78	0.0	0.00	1
2	4	1.51766	13.21	3.69	1.29	72.61	0.57	8.22	0.0	0.00	1
3	5	1.51742	13.27	3.62	1.24	73.08	0.55	8.07	0.0	0.00	1
4	6	1.51596	12.79	3.61	1.62	72.97	0.64	8.07	0.0	0.26	1

In [29]:
```
df.shape
```

Out[29]:
```
(213, 11)
```

In [30]:
```
df.drop(['1'], axis=1, inplace=True)
```

In [31]:
```
df['Type of glass'].value_counts()
```

Out[31]:
```
2    76
1    69
7    29
3    17
5    13
6     9
Name: Type of glass, dtype: int64
```

In [32]:
```
df.dtypes
```

Out[32]:
```
RI              float64
Na              float64
Mg              float64
Al              float64
Si              float64
K               float64
Ca              float64
Ba              float64
Fe              float64
Type of glass     int64
dtype: object
```

In [33]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 213 entries, 0 to 212
Data columns (total 10 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   RI             213 non-null    float64
 1   Na             213 non-null    float64
 2   Mg             213 non-null    float64
 3   Al             213 non-null    float64
 4   Si             213 non-null    float64
 5   K              213 non-null    float64
 6   Ca             213 non-null    float64
 7   Ba             213 non-null    float64
 8   Fe             213 non-null    float64
 9   Type of glass  213 non-null    int64  
dtypes: float64(9), int64(1)
memory usage: 16.8 KB
```

In [34]:
```
# Checking for null values
df.isnull().sum()
```

Out[34]:
```
RI              0
Na              0
Mg              0
Al              0
Si              0
K               0
Ca              0
Ba              0
Fe              0
Type of glass   0
dtype: int64
```

In [35]:
```
df.describe()
```

Out[35]:

	RI	Na	Mg	Al	Si	K	Ca	Ba
count	213.000000	213.000000	213.000000	213.000000	213.000000	213.000000	213.000000	213.000000
mean	1.518353	13.406761	2.676056	1.446526	72.655023	0.499108	8.957934	0.175869
std	0.003039	0.818371	1.440453	0.499882	0.774052	0.653035	1.426435	0.498245
min	1.511150	10.730000	0.000000	0.290000	69.810000	0.000000	5.430000	0.000000
25%	1.516520	12.900000	2.090000	1.190000	72.280000	0.130000	8.240000	0.000000
50%	1.517680	13.300000	3.480000	1.360000	72.790000	0.560000	8.600000	0.000000
75%	1.519150	13.830000	3.600000	1.630000	73.090000	0.610000	9.180000	0.000000
max	1.533930	17.380000	3.980000	3.500000	75.410000	6.210000	16.190000	3.150000

In [36]:
```
#correaltion matrix
corr=df.corr().style.background_gradient(cmap='winter')
corr
```

Out[36]:

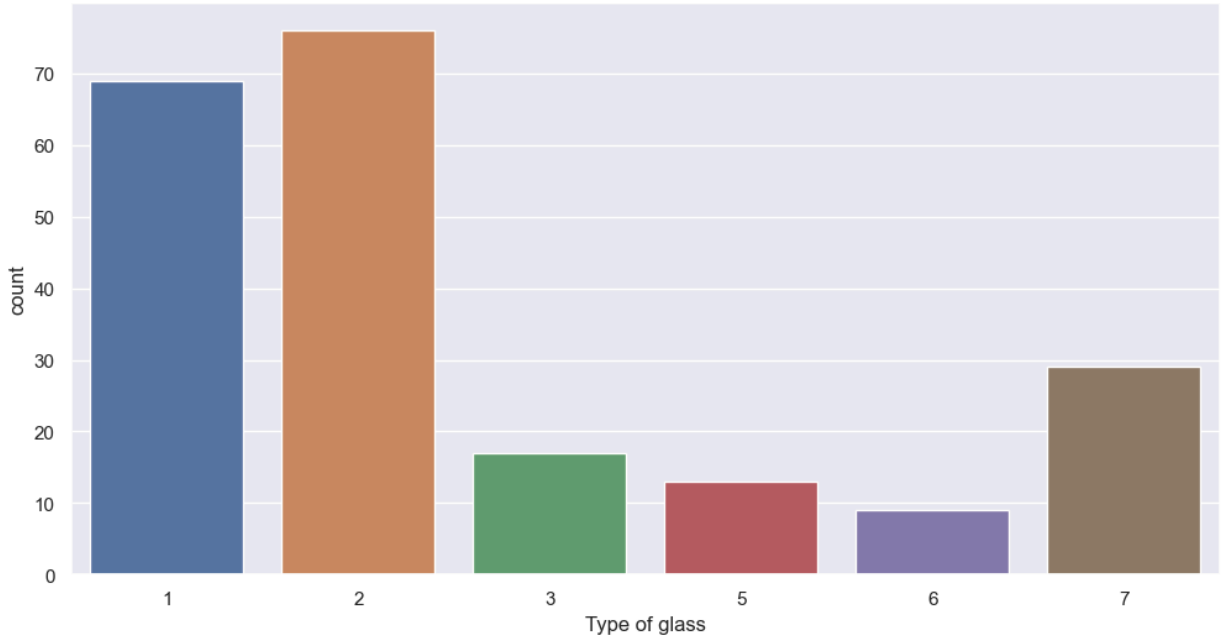
	RI	Na	Mg	Al	Si	K	Ca	Ba	Fe
RI	1.000000	-0.193436	-0.128118	-0.405671	-0.540010	-0.287900	0.812495	0.001062	0.145791
Na	-0.193436	1.000000	-0.276486	0.157928	-0.068519	-0.265520	-0.275314	0.327233	-0.240802
Mg	-0.128118	-0.276486	1.000000	-0.480035	-0.160359	0.009397	-0.444559	-0.492149	0.086906
Al	-0.405671	0.157928	-0.480035	1.000000	-0.009226	0.324484	-0.260372	0.478936	-0.076456
Si	-0.540010	-0.068519	-0.160359	-0.009226	1.000000	-0.197684	-0.210141	-0.104361	-0.097674
K	-0.287900	-0.265520	0.009397	0.324484	-0.197684	1.000000	-0.318649	-0.043790	-0.009586
Ca	0.812495	-0.275314	-0.444559	-0.260372	-0.210141	-0.318649	1.000000	-0.113121	0.124674
Ba	0.001062	0.327233	-0.492149	0.478936	-0.104361	-0.043790	-0.113121	1.000000	-0.059729
Fe	0.145791	-0.240802	0.086906	-0.076456	-0.097674	-0.009586	0.124674	-0.059729	1.000000
Type of glass	-0.161322	0.504983	-0.744004	0.597754	0.147767	-0.012765	0.000372	0.574896	-0.191090

In [37]:
```
#visualize the heatmap
sns.set(rc={'figure.figsize':(16,8)})
sns.heatmap(df.corr(),annot=True,cmap='YlGnBu')
```

Out[37]: <Axes: >

In [38]:
```
plt.figure(figsize=(12,6))
sns.countplot(x='Type of glass', data=df)
```

Out[38]: <Axes: xlabel='Type of glass', ylabel='count'>



Data Modeling

In [42]:
```
X = df.drop(['Type of glass'], axis=1)
y = df['Type of glass']
```

In [51]:
```
X_train,X_test,y_train,y_test =train_test_split(X, y,test_size=0.25, random_state=1
normalizer = preprocessing.Normalizer().fit(X_train)
X_train = normalizer.transform(X_train)
X_test = normalizer.transform(X_test)
```

In [52]:
```
Dtc = DecisionTreeClassifier(max_depth = 6)
Dtc.fit(X_train, y_train)
y_pred = Dtc.predict(X_test)
Acc = accuracy_score(y_test,y_pred)
print('Test Data accuracy: {:.2f}%'.format(Acc*100))
```

```
Test Data accuracy: 70.37%
```

In [53]:
```
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
result = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(result)
result1 = classification_report(y_test, y_pred)
print("Classification Report:",)
print (result1)
```

```
Confusion Matrix:
[[19  4  0  0  0  1]
 [ 2 11  1  0  1  2]
 [ 3  1  0  0  0  0]
 [ 0  0  0  2  0  0]
 [ 0  1  0  0  2  0]
 [ 0  0  0  0  0  4]]
Classification Report:
              precision    recall  f1-score   support

           1       0.79      0.79      0.79        24
           2       0.65      0.65      0.65        17
           3       0.00      0.00      0.00         4
           5       1.00      1.00      1.00         2
           6       0.67      0.67      0.67         3
           7       0.57      1.00      0.73         4

    accuracy                           0.70        54
   macro avg       0.61      0.68      0.64        54
weighted avg       0.67      0.70      0.68        54
```

In [56]:
```
#Import Random Forest Model
from sklearn.ensemble import RandomForestClassifier

#Create a Gaussian Classifier
rfc=RandomForestClassifier(n_estimators=100, random_state = 0)

#Train the model using the training sets y_pred=clf.predict(X_test)
rfc.fit(X_train,y_train)

y_pred=rfc.predict(X_test)
acc = accuracy_score(y_test, y_pred)
print("Test set accuracy: {:.2f}".format(acc*100))

#Now we can predict which type of glass type it is.
print("The type of glass is :",rfc.predict([[3, 2, 4, 2, 1, 0, 3, 1, 0]]))
```

```
Test set accuracy: 75.93
The type of glass is : [7]
```

Model Validation

In [58]:
```
from sklearn.model_selection import cross_val_predict
predict = cross_val_predict(estimator = rfc, X = X_train, y = y_train, cv = 5)
print("Classification Report: \n",classification_report(y_train, predict))
```

```
Classification Report:
               precision    recall  f1-score   support

           1       0.63      0.76      0.69        45
           2       0.71      0.85      0.78        59
           3       0.50      0.08      0.13        13
           5       0.83      0.45      0.59        11
           6       1.00      0.83      0.91         6
           7       0.95      0.84      0.89        25

    accuracy                           0.73       159
   macro avg       0.77      0.63      0.66       159
weighted avg       0.73      0.73      0.71       159
```

Thankyou

In [ ]:
```

```