In [1]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.neighbors import KNeighborsClassifier
```

In [2]:
```
df = pd.read_csv("glass.csv")
df
```

Out[2]:

	RI	Na	Mg	Al	Si	K	Ca	Ba	Fe	Type
0	1.52101	13.64	4.49	1.10	71.78	0.06	8.75	0.00	0.0	1
1	1.51761	13.89	3.60	1.36	72.73	0.48	7.83	0.00	0.0	1
2	1.51618	13.53	3.55	1.54	72.99	0.39	7.78	0.00	0.0	1
3	1.51766	13.21	3.69	1.29	72.61	0.57	8.22	0.00	0.0	1
4	1.51742	13.27	3.62	1.24	73.08	0.55	8.07	0.00	0.0	1
...	...	...	...	...	...	...	...	...	...	...
209	1.51623	14.14	0.00	2.88	72.61	0.08	9.18	1.06	0.0	7
210	1.51685	14.92	0.00	1.99	73.06	0.00	8.40	1.59	0.0	7
211	1.52065	14.36	0.00	2.02	73.42	0.00	8.44	1.64	0.0	7
212	1.51651	14.38	0.00	1.94	73.61	0.00	8.48	1.57	0.0	7
213	1.51711	14.23	0.00	2.08	73.36	0.00	8.62	1.67	0.0	7

214 rows × 10 columns

In [3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 214 entries, 0 to 213
Data columns (total 10 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   RI      214 non-null    float64
 1   Na      214 non-null    float64
 2   Mg      214 non-null    float64
 3   Al      214 non-null    float64
 4   Si      214 non-null    float64
 5   K       214 non-null    float64
 6   Ca      214 non-null    float64
 7   Ba      214 non-null    float64
 8   Fe      214 non-null    float64
 9   Type    214 non-null    int64  
dtypes: float64(9), int64(1)
memory usage: 16.8 KB
```

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 214 entries, 0 to 213
Data columns (total 10 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   RI      214 non-null    float64
 1   Na      214 non-null    float64
 2   Mg      214 non-null    float64
 3   Al      214 non-null    float64
 4   Si      214 non-null    float64
 5   K       214 non-null    float64
 6   Ca      214 non-null    float64
 7   Ba      214 non-null    float64
 8   Fe      214 non-null    float64
 9   Type    214 non-null    int64  
dtypes: float64(9), int64(1)
memory usage: 16.8 KB
```

In [5]:
```
df.describe().round(2).style.background_gradient(cmap = 'Oranges')
```

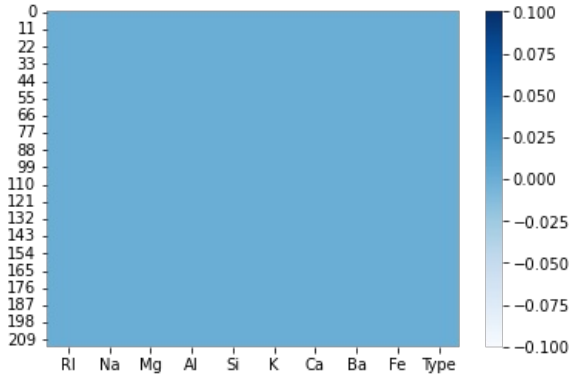
Out[5]:

	RI	Na	Mg	Al	Si	K	Ca	Ba	Fe	Type

count	214.000000	214.000000	214.000000	214.000000	214.000000	214.000000	214.000000	214.000000	214.000000	214.000000
mean	1.520000	13.410000	2.680000	1.440000	72.650000	0.500000	8.960000	0.180000	0.060000	2.780000
std	0.000000	0.820000	1.440000	0.500000	0.770000	0.650000	1.420000	0.500000	0.100000	2.100000
min	1.510000	10.730000	0.000000	0.290000	69.810000	0.000000	5.430000	0.000000	0.000000	1.000000
25%	1.520000	12.910000	2.110000	1.190000	72.280000	0.120000	8.240000	0.000000	0.000000	1.000000
50%	1.520000	13.300000	3.480000	1.360000	72.790000	0.560000	8.600000	0.000000	0.000000	2.000000
75%	1.520000	13.820000	3.600000	1.630000	73.090000	0.610000	9.170000	0.000000	0.100000	3.000000
max	1.530000	17.380000	4.490000	3.500000	75.410000	6.210000	16.190000	3.150000	0.510000	7.000000

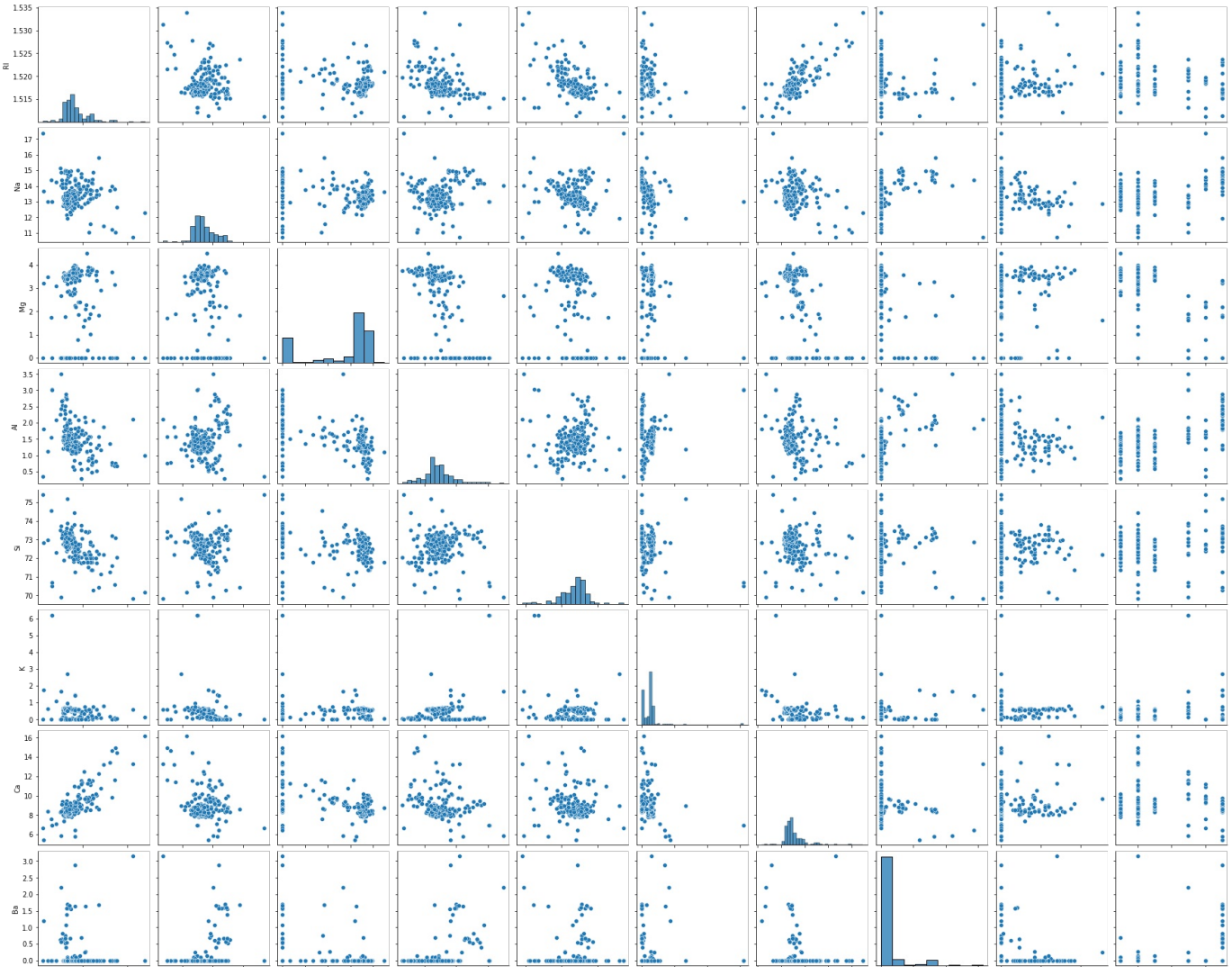
In [6]:
```
import seaborn as sns
sns.heatmap(df.isnull(),cmap='Blues')
```

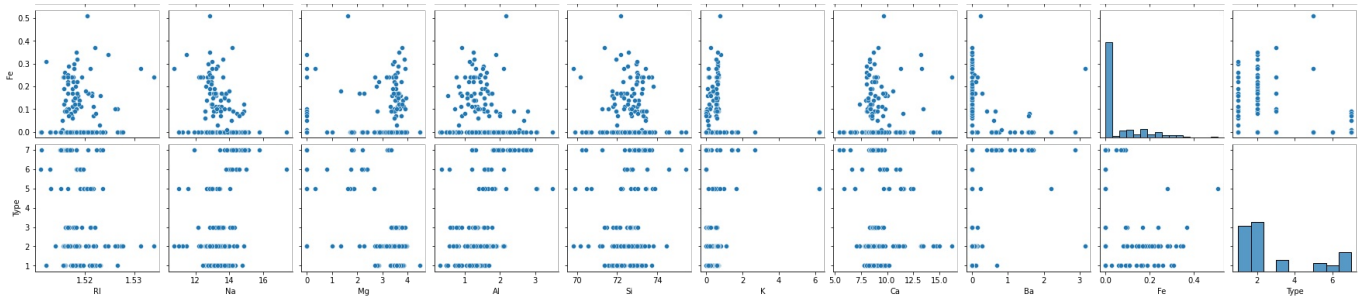
Out[6]:
<AxesSubplot:>



In [7]:
```
sns.pairplot(df)
```

Out[7]:
<seaborn.axisgrid.PairGrid at 0x152e44c96d0>





In [8]:
```
df.duplicated()
```

Out[8]:
```
0      False
1      False
2      False
3      False
4      False
       ...
209    False
210    False
211    False
212    False
213    False
Length: 214, dtype: bool
```

In [9]:
```
array = df.values
x= array[:,0:9]
y= array[:,9]
```

In [10]:
```
x
```

Out[10]:
```
array([[ 1.52101, 13.64   ,  4.49   , ...,  8.75   ,  0.     ,  0.     ],
       [ 1.51761, 13.89   ,  3.6    , ...,  7.83   ,  0.     ,  0.     ],
       [ 1.51618, 13.53   ,  3.55   , ...,  7.78   ,  0.     ,  0.     ],
       ...,
       [ 1.52065, 14.36   ,  0.     , ...,  8.44   ,  1.64   ,  0.     ],
       [ 1.51651, 14.38   ,  0.     , ...,  8.48   ,  1.57   ,  0.     ],
       [ 1.51711, 14.23   ,  0.     , ...,  8.62   ,  1.67   ,  0.     ]])
```

In [11]:
```
y
```

Out[11]:
```
array([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., 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.])
```

In [12]:
```
x = (x-x.min(axis=0))/(x.max(axis=0))-x.min(axis=0)
```

In [13]:
```
x
```

Out[13]:
```
array([[ -1.50472207, -10.56256617,   1.        , ...,  -5.22493515,
            0.        ,   0.        ],
       [ -1.5069386 , -10.54818182,   0.80178174, ...,  -5.28176035,
            0.        ,   0.        ],
       [ -1.50787084, -10.56889528,   0.79064588, ...,  -5.28484867,
            0.        ,   0.        ],
       ...,
       [ -1.50495676, -10.52113924,   0.        , ...,  -5.24408277,
            0.52063492,   0.        ],
       [ -1.50765571, -10.51998849,   0.        , ...,  -5.24161211,
```

```
   0.4984127 ,   0.        ],
[ -1.50726456, -10.5286191 ,   0.        , ...,  -5.23296479,
   0.53015873,   0.        ]])
```

In [14]:
```
num_folds = 30
KFold = KFold(n_splits=30)
```

In [15]:
```
model=KNeighborsClassifier(n_neighbors=20)
results=cross_val_score(model,x,y,cv=KFold)
```

In [16]:
```
print(results.mean())
```

```
0.531547619047619
```

grid search for algorithm tunning

In [17]:
```
import numpy
from pandas import read_csv
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import GridSearchCV
```

In [18]:
```
n_neighbors = numpy.array(range(1,40))
param_grid = dict(n_neighbors=n_neighbors)
```

In [19]:
```
n_neighbors
```

Out[19]:
```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17,
       18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34,
       35, 36, 37, 38, 39])
```

In [20]:
```
model = KNeighborsClassifier()
grid = GridSearchCV(estimator=model, param_grid=param_grid)
grid.fit(x, y)
```

Out[20]:
```
GridSearchCV(estimator=KNeighborsClassifier(),
             param_grid={'n_neighbors': array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17
,
       18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34,
       35, 36, 37, 38, 39])})
```

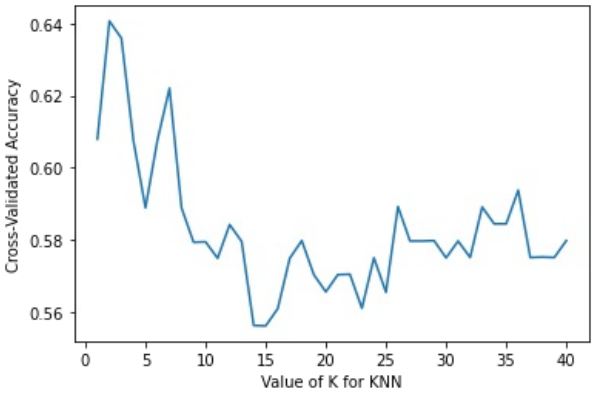
In [21]:
```
print(grid.best_score_)
print(grid.best_params_)
```

```
0.6406423034330011
{'n_neighbors': 2}
```

visualizing the CV results

In [22]:
```
import matplotlib.pyplot as plt
%matplotlib inline
# choose k between 1 to 41
k_range = range(1, 41)
k_scores = []
# use iteration to caclulator different k in models, then return the average accuracy based on the cross validati
for k in k_range:
    knn = KNeighborsClassifier(n_neighbors=k)
    scores = cross_val_score(knn, x, y, cv=5)
    k_scores.append(scores.mean())
# plot to see clearly
plt.plot(k_range, k_scores)
plt.xlabel('Value of K for KNN')
plt.ylabel('Cross-Validated Accuracy')
```

```
plt.show()
```



hence from the above graph we casn see that best value for k was 2 which gave us classification accuracy as 0.640531561461794

In [ ]: