In [95]:

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import cross_val_score
import seaborn as sns
import numpy as np
```

In [96]:

```
glass_data = pd.read_csv('glass.csv')
glass_data
```

Out[96]:

	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 [97]:

```
# initial analysis
glass_data.shape
```

Out[97]:

```
(214, 10)
```

In [98]:

```
glass_data.isna().sum()
```

Out[98]:

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

In [99]:
```
glass_data.Type.value_counts()
```

Out[99]:
```
2    76
1    70
7    29
3    17
5    13
6     9
Name: Type, dtype: int64
```

In [100]:
```
glass_data.dtypes
```

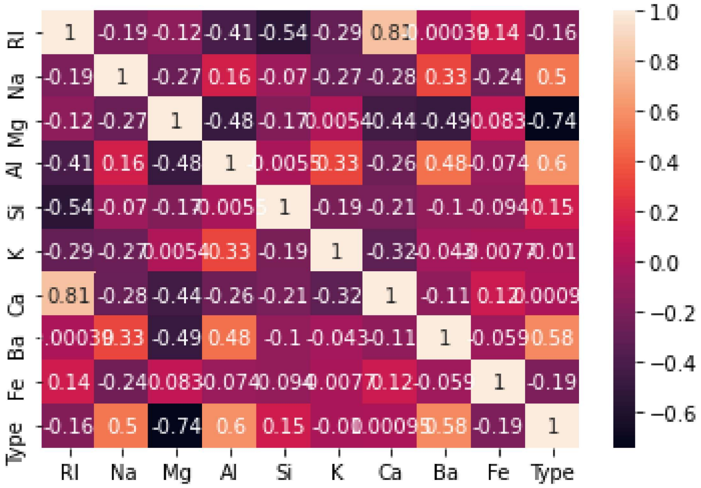
Out[100]:
```
RI      float64
Na      float64
Mg      float64
Al      float64
Si      float64
K       float64
Ca      float64
Ba      float64
Fe      float64
Type      int64
dtype: object
```

In [104]:
```
round(glass_data.corr(),2)
```

Out[104]:

	RI	Na	Mg	Al	Si	K	Ca	Ba	Fe	Type
RI	1.00	-0.19	-0.12	-0.41	-0.54	-0.29	0.81	-0.00	0.14	-0.16
Na	-0.19	1.00	-0.27	0.16	-0.07	-0.27	-0.28	0.33	-0.24	0.50
Mg	-0.12	-0.27	1.00	-0.48	-0.17	0.01	-0.44	-0.49	0.08	-0.74
Al	-0.41	0.16	-0.48	1.00	-0.01	0.33	-0.26	0.48	-0.07	0.60
Si	-0.54	-0.07	-0.17	-0.01	1.00	-0.19	-0.21	-0.10	-0.09	0.15
K	-0.29	-0.27	0.01	0.33	-0.19	1.00	-0.32	-0.04	-0.01	-0.01
Ca	0.81	-0.28	-0.44	-0.26	-0.21	-0.32	1.00	-0.11	0.12	0.00
Ba	-0.00	0.33	-0.49	0.48	-0.10	-0.04	-0.11	1.00	-0.06	0.58
Fe	0.14	-0.24	0.08	-0.07	-0.09	-0.01	0.12	-0.06	1.00	-0.19
Type	-0.16	0.50	-0.74	0.60	0.15	-0.01	0.00	0.58	-0.19	1.00

In [105]:
```
# Data visualization
sns.heatmap(glass_data.corr(), annot=True)
plt.show()
```



MODEL BUILDING

In [106]:
```
X = glass_data.drop(['Type'], axis=1)
y= glass_data['Type']
```

In [107]:
```
X
```

Out[107]:

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

214 rows × 9 columns

In [108]:
```
y
```

Out[108]:
```
0      1
1      1
2      1
3      1
4      1
      ..
209    7
210    7
211    7
212    7
213    7
Name: Type, Length: 214, dtype: int64
```

In [109]:
```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.20, random_state
```

In [110]:
```
X_train.shape , y_train.shape
```

Out[110]:
```
((171, 9), (171,))
```

In [111]:
```
X_test.shape, y_test.shape
```

Out[111]:
```
((43, 9), (43,))
```

Model training | Testing | Evaluation without NORMALIZATION

Generating a Model with K=3

In [112]:

```
knn_model = KNeighborsClassifier(n_neighbors=3)
knn_model.fit(X_train,y_train)
```

Out[112]:

```
KNeighborsClassifier(n_neighbors=3)
```

In [116]:

```
y_test_pred = knn_model.predict(X_test)
y_test_pred
print("Accuracy score : " , round(accuracy_score(y_test,y_test_pred),4))
```

```
Accuracy score :  0.7907
```

Generating a Model with K=5

In [117]:

```
knn_model = KNeighborsClassifier(n_neighbors=5)
knn_model.fit(X_train,y_train)
```

Out[117]:

```
KNeighborsClassifier()
```

In [118]:

```
y_test_pred = knn_model.predict(X_test)
y_test_pred
print("Accuracy score : " , round(accuracy_score(y_test,y_test_pred),4))
```

```
Accuracy score :  0.7907
```

Model Training | Testing | Evaluation with NORMALIZATION

Generating a Model with K = 3

In [119]:

```
scaler = StandardScaler()
scaled_X = scaler.fit_transform(X)
```

In [120]:

```
X_train,X_test,y_train,y_test = train_test_split(scaled_X,y,test_size=0.20, rando
X_train.shape , y_train.shape,X_test.shape,y_test.shape
```

Out[120]:

```
((171, 9), (171,), (43, 9), (43,))
```

In [121]:
```
X_train
```

Out[121]:
```
array([[-2.38151559,  4.87563749, -1.86551055, ..., -1.62482241,
        -0.35287683, -0.5864509 ],
       [-0.48697606, -0.90567874,  0.13583231, ..., -0.29367195,
        -0.35287683, -0.5864509 ],
       [-0.15691691, -0.26740179,  0.84464125, ..., -0.36410319,
        -0.35287683,  2.29388828],
       ...,
       [-0.08430389, -0.095558  ,  0.80989571, ..., -0.58948316,
        -0.35287683, -0.5864509 ],
       [-0.62890149, -0.45152014,  0.49718589, ..., -0.26549945,
        -0.35287683, -0.5864509 ],
       [-0.81373462, -0.47606926,  0.62226982, ..., -0.70217315,
        -0.35287683, -0.5864509 ]])
```

In [122]:
```
X_test
```

Out[122]:
```
array([[ 2.56607109e+00,  3.58600602e-01, -1.86551055e+00,
        -1.70460232e-01, -1.82589947e+00, -4.71910254e-01,
         3.15745888e+00, -3.52876828e-01,  4.42241664e-01],
       [-8.83047040e-01, -2.06029009e-01,  5.59727851e-01,
         5.03781754e-02,  7.75254394e-01, -1.79901870e-01,
        -6.52871279e-01, -3.52876828e-01, -5.86450902e-01],
       [ 3.31570637e-01,  4.69071613e-01, -1.90775722e-01,
        -5.11755954e-01,  1.41142259e-01, -7.63918639e-01,
         5.72632323e-01, -3.52876828e-01, -5.86450902e-01],
       [-2.95541751e-01, -6.97011279e-01,  5.66676959e-01,
        -6.12137048e-01,  8.01136522e-01,  9.67376522e-02,
        -2.86628823e-01, -3.52876828e-01, -5.86450902e-01],
       [-7.04815098e-01,  8.98681099e-01, -1.86551055e+00,
         2.88112504e+00, -5.29737012e-02, -6.40967740e-01,
         1.57087998e-01,  1.78397794e+00, -5.86450902e-01],
       [ 4.37189566e-01,  1.41421248e+00, -1.86551055e+00,
        -1.77655774e+00,  1.07289887e+00, -7.63918639e-01,
         1.59388532e+00, -3.52876828e-01, -5.86450902e-01],
       [-4.70473101e-01, -6.72462165e-01,  6.22269816e-01,
         2.70841327e-01,  5.29737012e-02,  1.72581064e-01,
```

In [123]:
```
knn_model = KNeighborsClassifier(n_neighbors=3)
knn_model.fit(X_train,y_train)
```

Out[123]:
```
KNeighborsClassifier(n_neighbors=3)
```

In [124]:
```
y_test_pred = knn_model.predict(X_test)
y_test_pred
print("Accuracy score : " , round(accuracy_score(y_test,y_test_pred),4))
```

```
Accuracy score :  0.7209
```

Generating a Model with K = 5

In [125]:
```
knn_model = KNeighborsClassifier(n_neighbors=5)
knn_model.fit(X_train,y_train)
```

Out[125]:
```
KNeighborsClassifier()
```

In [126]:
```
y_test_pred = knn_model.predict(X_test)
y_test_pred
print("Accuracy score : " , round(accuracy_score(y_test,y_test_pred),4))
```

```
Accuracy score :  0.7442
```

How to pickup Optimum no. of K?

In [127]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [137]:
```
neighbors = list(range(1,25))
cv_scores = []

for i in neighbors:
    knn_model = KNeighborsClassifier(n_neighbors = i)
    cv_score = cross_val_score(estimator = knn_model, X = scaled_X, y=y, cv = 5)
    cv_scores.append(cv_score.mean())
```

In [138]:
```
cv_scores
```

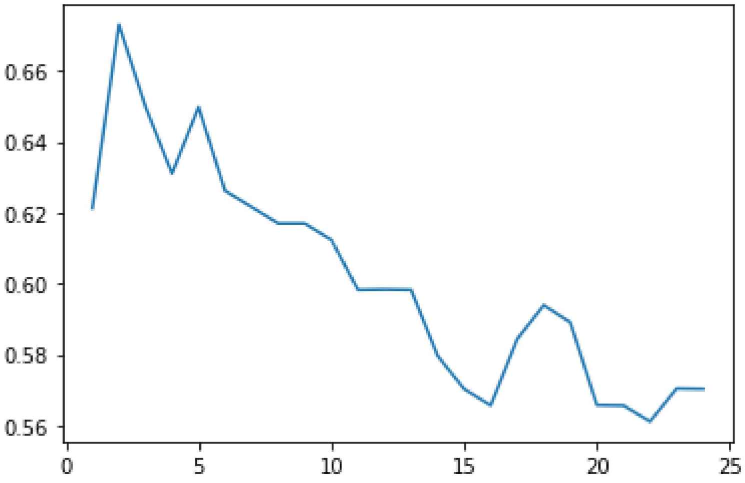
Out[138]:
```
[0.6212624584717608,
 0.6729789590254706,
 0.6498338870431895,
 0.6310077519379845,
 0.6497231450719824,
 0.6261351052048727,
 0.6215946843853821,
 0.6169435215946844,
 0.6169435215946844,
 0.6122923588039867,
 0.5982281284606865,
 0.5983388704318936,
 0.5982281284606865,
 0.5796234772978959,
 0.5703211517165006,
 0.5656699889258029,
 0.5843853820598006,
 0.5939091915836101,
 0.5890365448504983,
 0.56578073089701,
 0.5656699889258029,
 0.5611295681063122,
 0.5704318936877076,
 0.5703211517165006]
```

```
In [139]: neighbours[cv_scores.index(max(cv_scores))]
```

```
Out[139]: 2
```

Visualizing the K neighbors wrt CV

```
In [140]: plt.plot(neighbors, cv_scores)
          plt.show()
```



```
In [141]: 
          knn_model = KNeighborsClassifier(n_neighbors = 2)
          knn_model.fit(X_train,y_train)
          y_test_pred = knn_model.predict(X_test)
          print("Accuracy score : " , round(accuracy_score(y_test,y_test_pred),4))
```

```
Accuracy score :  0.814
```

```
In [ ]: 
```