In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
from matplotlib import pyplot as plt

from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split,cross_val_scor
from sklearn.metrics import accuracy_score,classification_report,co
from sklearn.preprocessing import StandardScaler
```

In [2]:
```
glass_data=pd.read_csv('glass.csv')
glass_data
```

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]:
```
glass_data.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]:
```
glass_data.dtypes
```

Out[4]:
```
RI      float64
Na      float64
Mg      float64
Al      float64
Si      float64
K       float64
Ca      float64
Ba      float64
Fe      float64
Type      int64
dtype: object
```

In [5]:
```
glass_data.isnull().sum()
```

Out[5]:
```
RI      0
Na      0
Mg      0
Al      0
Si      0
K       0
Ca      0
Ba      0
Fe      0
Type    0
dtype: int64
```

In [6]:
```
glass_data.duplicated().sum()
```

Out[6]:
```
1
```

In [7]:
```
glass_data[glass_data.duplicated()]
```

Out[7]:

	RI	Na	Mg	Al	Si	K	Ca	Ba	Fe	Type
39	1.52213	14.21	3.82	0.47	71.77	0.11	9.57	0.0	0.0	1

In [8]:
```
glass_data.drop_duplicates(inplace=True)
```

In [9]:
```
glass_data.duplicated().sum()
```

Out[9]:
```
0
```

In [10]:
```
glass_data.head()
```

Out[10]:

	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	0.0	0.0	1
1	1.51761	13.89	3.60	1.36	72.73	0.48	7.83	0.0	0.0	1
2	1.51618	13.53	3.55	1.54	72.99	0.39	7.78	0.0	0.0	1
3	1.51766	13.21	3.69	1.29	72.61	0.57	8.22	0.0	0.0	1
4	1.51742	13.27	3.62	1.24	73.08	0.55	8.07	0.0	0.0	1

In [11]:
```
glass_data.shape
```

Out[11]:
```
(213, 10)
```

In [12]:
```
glass_data['Type'].unique()
```

Out[12]:
```
array([1, 2, 3, 5, 6, 7])
```

In [13]:
```
glass_data.describe()
```

Out[13]:

	RI	Na	Mg	Al	Si	K	Ca
count	213.000000	213.000000	213.000000	213.000000	213.000000	213.000000	213.000000
mean	1.518348	13.404085	2.679202	1.449484	72.655070	0.498873	8.954085
std	0.003033	0.816662	1.443691	0.495925	0.773998	0.653185	1.425882
min	1.511150	10.730000	0.000000	0.290000	69.810000	0.000000	5.430000
25%	1.516520	12.900000	2.090000	1.190000	72.280000	0.130000	8.240000
50%	1.517680	13.300000	3.480000	1.360000	72.790000	0.560000	8.600000
75%	1.519150	13.810000	3.600000	1.630000	73.090000	0.610000	9.150000
max	1.533930	17.380000	4.490000	3.500000	75.410000	6.210000	16.190000

In [14]:
```
glass_data["Type"].value_counts()
```

Out[14]:
```
2    76
1    69
7    29
3    17
5    13
6     9
Name: Type, dtype: int64
```

In [15]:
```
X= glass_data.drop('Type',axis=1)
y=glass_data[['Type']]
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.20
X_train.shape,y_train.shape
```

Out[15]:
```
((170, 9), (170, 1))
```

In [16]:
```
X_test.shape,y_test.shape
```

Out[16]:
```
((43, 9), (43, 1))
```

In [17]:
```
knn = KNeighborsClassifier(n_neighbors=1)
knn.fit(X_train,y_train)
```

```
/opt/anaconda3/lib/python3.9/site-packages/sklearn/neighbors/_clas
sification.py:215: DataConversionWarning: A column-vector y was pa
ssed when a 1d array was expected. Please change the shape of y to
(n_samples,), for example using ravel().
  return self._fit(X, y)
```

Out[17]:
```
KNeighborsClassifier
KNeighborsClassifier(n_neighbors=1)
```

In [18]:
```
#prediction for training
pred_y= knn.predict(X_train)
```

In [19]:
```
#accuracy Score
accuracy_score(y_train,pred_y)
```

Out[19]:
```
1.0
```

In [20]:
```
#counfusion Matrix
confusion_matrix(y_train,pred_y)
```

Out[20]:
```
array([[47,  0,  0,  0,  0,  0],
       [ 0, 66,  0,  0,  0,  0],
       [ 0,  0, 13,  0,  0,  0],
       [ 0,  0,  0, 12,  0,  0],
       [ 0,  0,  0,  0,  6,  0],
       [ 0,  0,  0,  0,  0, 26]])
```

In [21]:
```
print(classification_report(y_train,pred_y))
```

```
              precision    recall  f1-score   support

           1       1.00      1.00      1.00        47
           2       1.00      1.00      1.00        66
           3       1.00      1.00      1.00        13
           5       1.00      1.00      1.00        12
           6       1.00      1.00      1.00         6
           7       1.00      1.00      1.00        26

    accuracy                           1.00       170
   macro avg       1.00      1.00      1.00       170
weighted avg       1.00      1.00      1.00       170
```

In [22]:
```
#prediction for testing data
y_pred=knn.predict(X_test)
```

In [23]:
```
#accuracy score for test data
accuracy_score(y_test,y_pred)
```

Out[23]:
```
0.6976744186046512
```

In [24]:
```
#confusion Matrix
confusion_matrix(y_test,y_pred)
```

Out[24]:
```
array([[12,  6,  4,  0,  0,  0],
       [ 0, 10,  0,  0,  0,  0],
       [ 1,  0,  3,  0,  0,  0],
       [ 0,  0,  0,  1,  0,  0],
       [ 0,  1,  0,  1,  1,  0],
       [ 0,  0,  0,  0,  0,  3]])
```

In [25]:
```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           1       0.92      0.55      0.69        22
           2       0.59      1.00      0.74        10
           3       0.43      0.75      0.55         4
           5       0.50      1.00      0.67         1
           6       1.00      0.33      0.50         3
           7       1.00      1.00      1.00         3

    accuracy                           0.70        43
   macro avg       0.74      0.77      0.69        43
weighted avg       0.80      0.70      0.69        43
```

In [26]:
```
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 th
for k in k_range:
    knn = KNeighborsClassifier(n_neighbors=k)
    scores = cross_val_score(knn, X, y, cv=5)
    k_scores.append(scores.mean())
```

```
/opt/anaconda3/lib/python3.9/site-packages/sklearn/neighbors/_clas
sification.py:215: DataConversionWarning: A column-vector y was pa
ssed when a 1d array was expected. Please change the shape of y to
(n_samples,), for example using ravel().
  return self._fit(X, y)
/opt/anaconda3/lib/python3.9/site-packages/sklearn/neighbors/_clas
sification.py:215: DataConversionWarning: A column-vector y was pa
ssed when a 1d array was expected. Please change the shape of y to
(n_samples,), for example using ravel().
  return self._fit(X, y)
/opt/anaconda3/lib/python3.9/site-packages/sklearn/neighbors/_clas
sification.py:215: DataConversionWarning: A column-vector y was pa
ssed when a 1d array was expected. Please change the shape of y to
(n_samples,), for example using ravel().
  return self._fit(X, y)
/opt/anaconda3/lib/python3.9/site-packages/sklearn/neighbors/_clas
sification.py:215: DataConversionWarning: A column-vector y was pa
ssed when a 1d array was expected. Please change the shape of y to
(n_samples,), for example using ravel().
```

In [27]:
```
# plot to see clearly
plt.figure(figsize=(10,6))
plt.plot(k_range, k_scores,color='blue',linestyle='dashed',marker='
plt.grid(True)
plt.title('CV Accuracy Vs k value for KNN')
plt.xlabel('Value of K for KNN')
plt.ylabel('Cross-Validated Accuracy')
plt.show()
```