KNN

Prepare a model for glass classification using KNN

Data Description:

RI : refractive index

Na: Sodium (unit measurement: weight percent in corresponding oxide, as are attributes 4-10)

Mg: Magnesium

AI: Aluminum

Si: Silicon

K:Potassium

Ca: Calcium

Ba: Barium

Fe: Iron

Type: Type of glass: (class attribute) 1 -- building_windows_float_processed
2 --building_windows_non_float_processed
3 --vehicle_windows_float_processed
4 --vehicle_windows_non_float_processed (none in this database)
5 --containers
6 --tableware
7 --headlamps

In [27]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
import seaborn as sns
from sklearn.preprocessing import normalize, LabelEncoder
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix



from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score

from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import classification_report, accuracy_score,precision_score,recall_score,f1_score,matthews_corrcoef
from sklearn.metrics import confusion_matrix
```

In [3]:
```
glass= pd.read_csv("glass.csv")
glass.head()
```

Out[3]:

	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

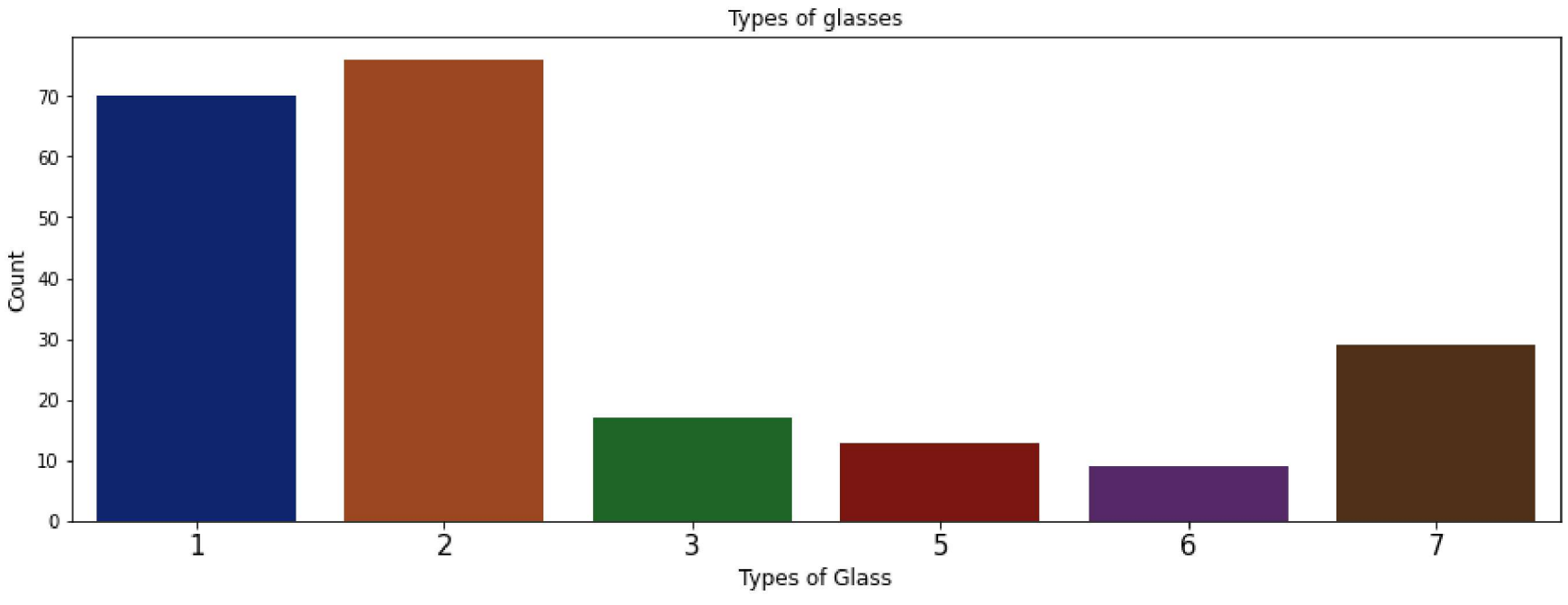
In [4]:
```
glass.info()
```

```
<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 [7]:
```
plt.figure(figsize=(15, 5))

plt.title("Types of glasses")
sns.countplot(data=glass, x="Type",palette = "dark")
plt.xticks(rotation = 0, size = 15)
```

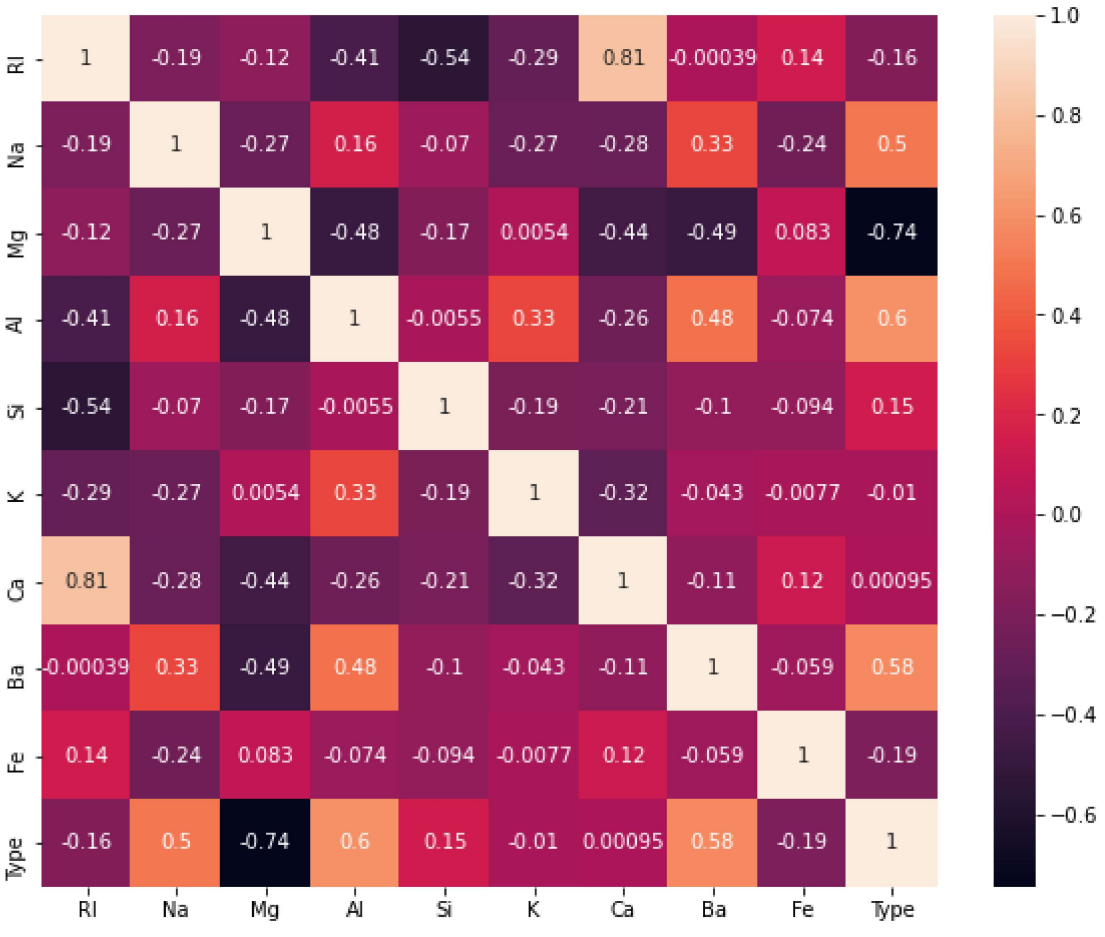
```
plt.xlabel("Types of Glass", fontsize=12)
plt.ylabel("Count", fontsize=12)
```

Out[7]: Text(0, 0.5, 'Count')



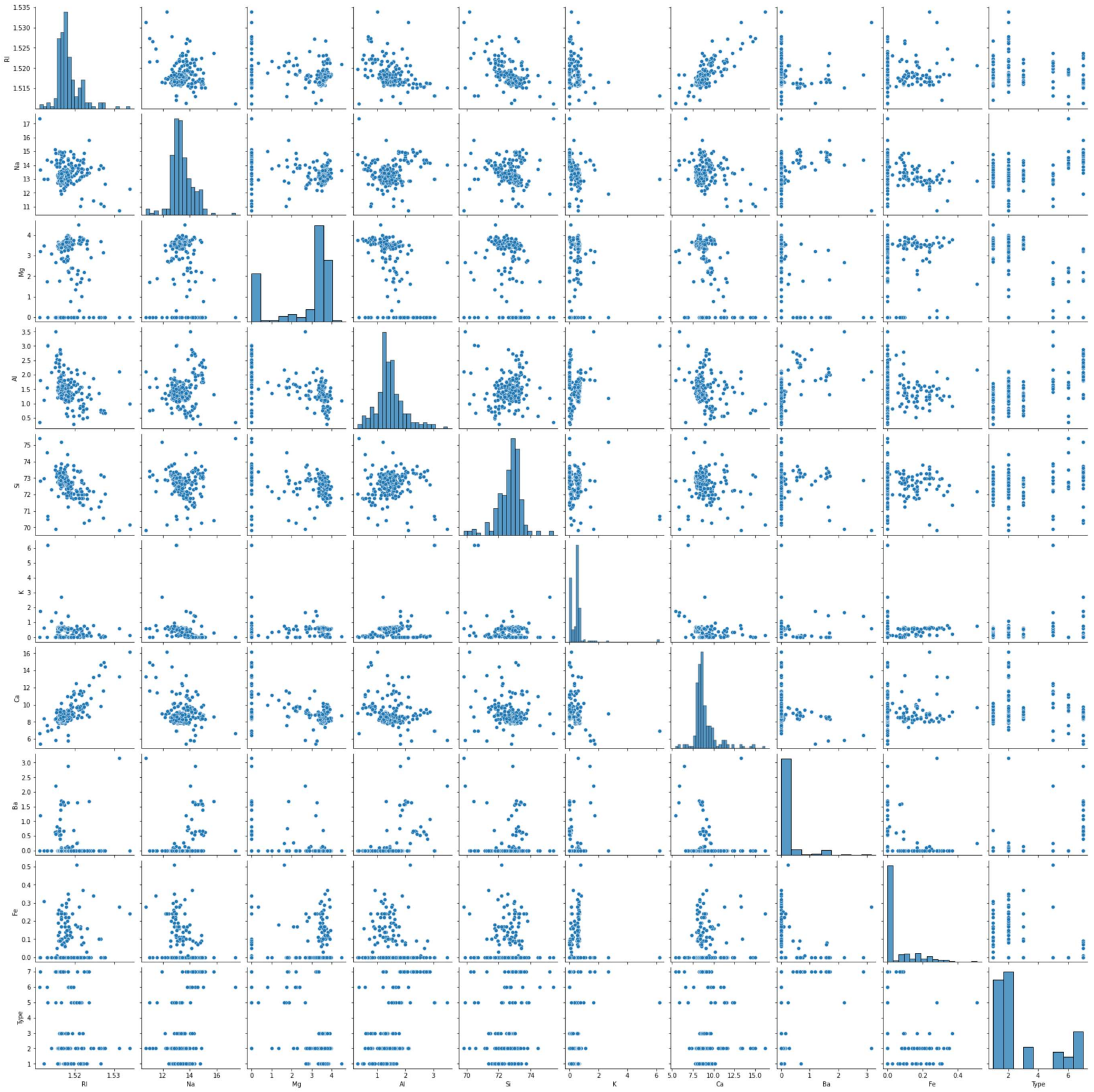
In [9]:
```
plt.figure(figsize=(10,8))
sns.heatmap(glass.corr(),annot=True)
```

Out[9]: <AxesSubplot:>

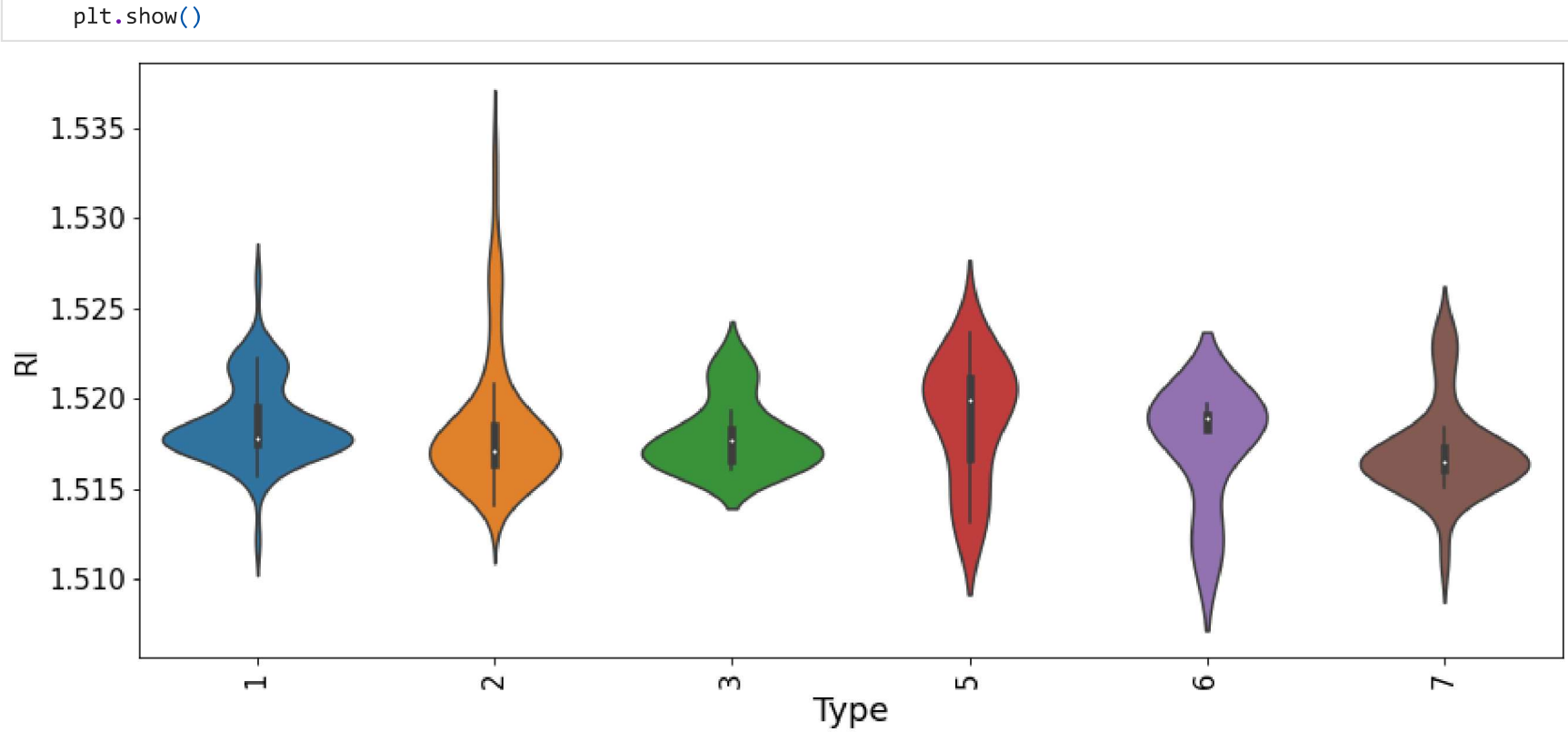


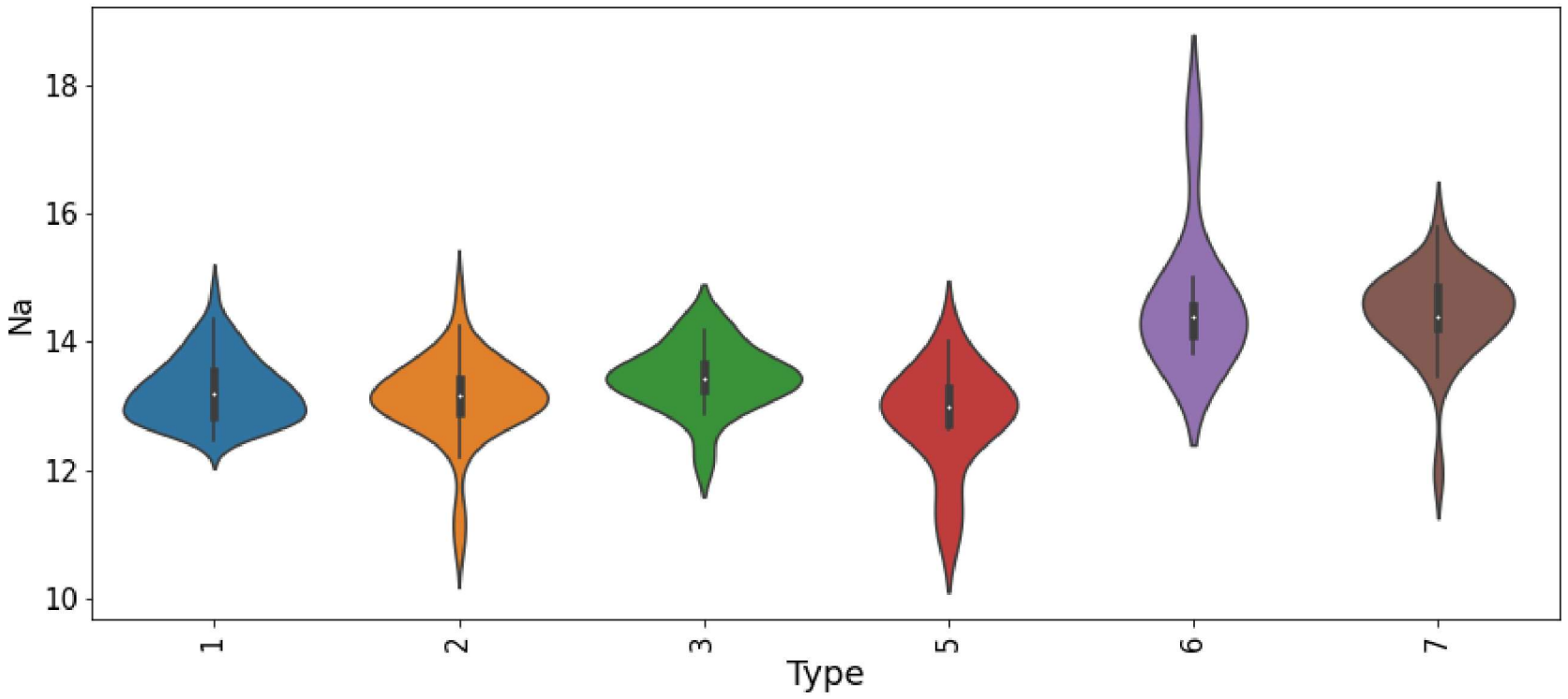
In [15]:
```
sns.pairplot(glass)
```

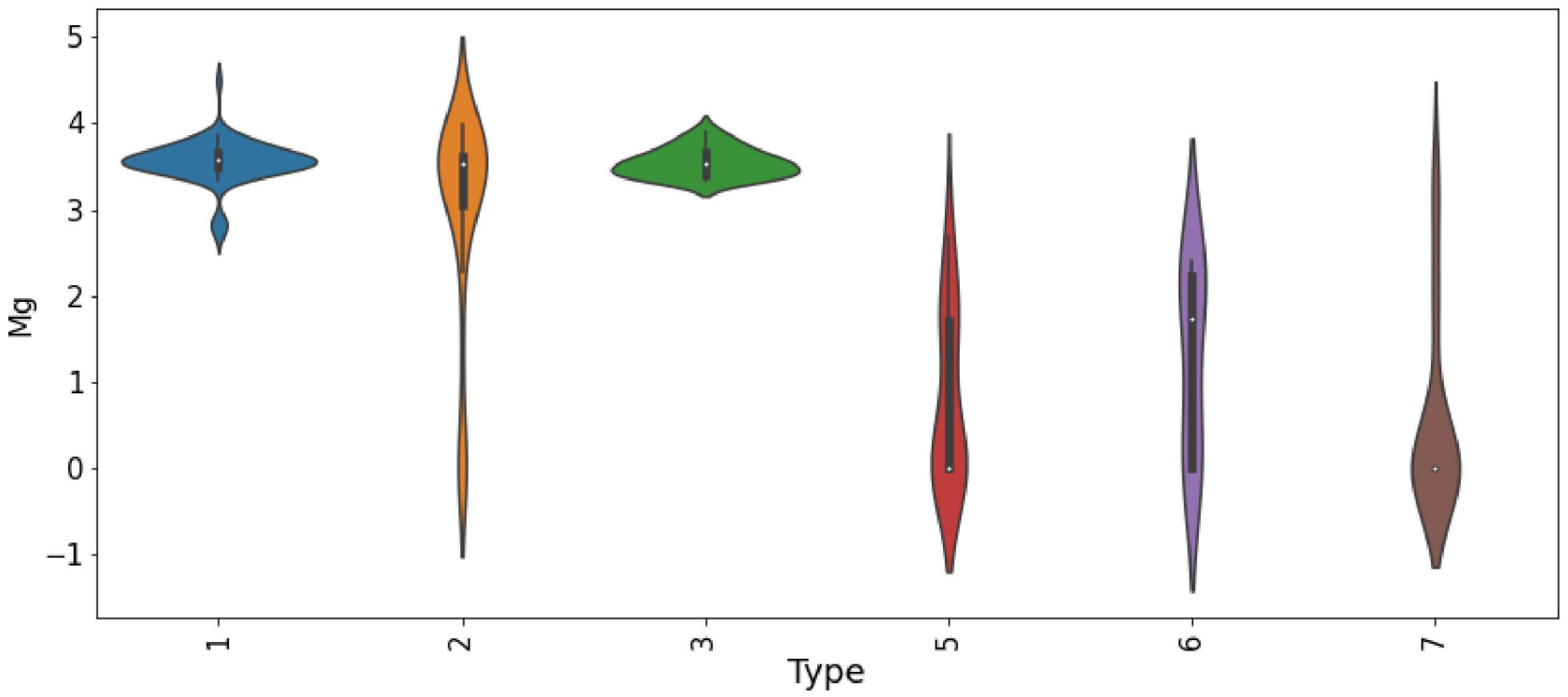
Out[15]: <seaborn.axisgrid.PairGrid at 0x1f4cfb69be0>

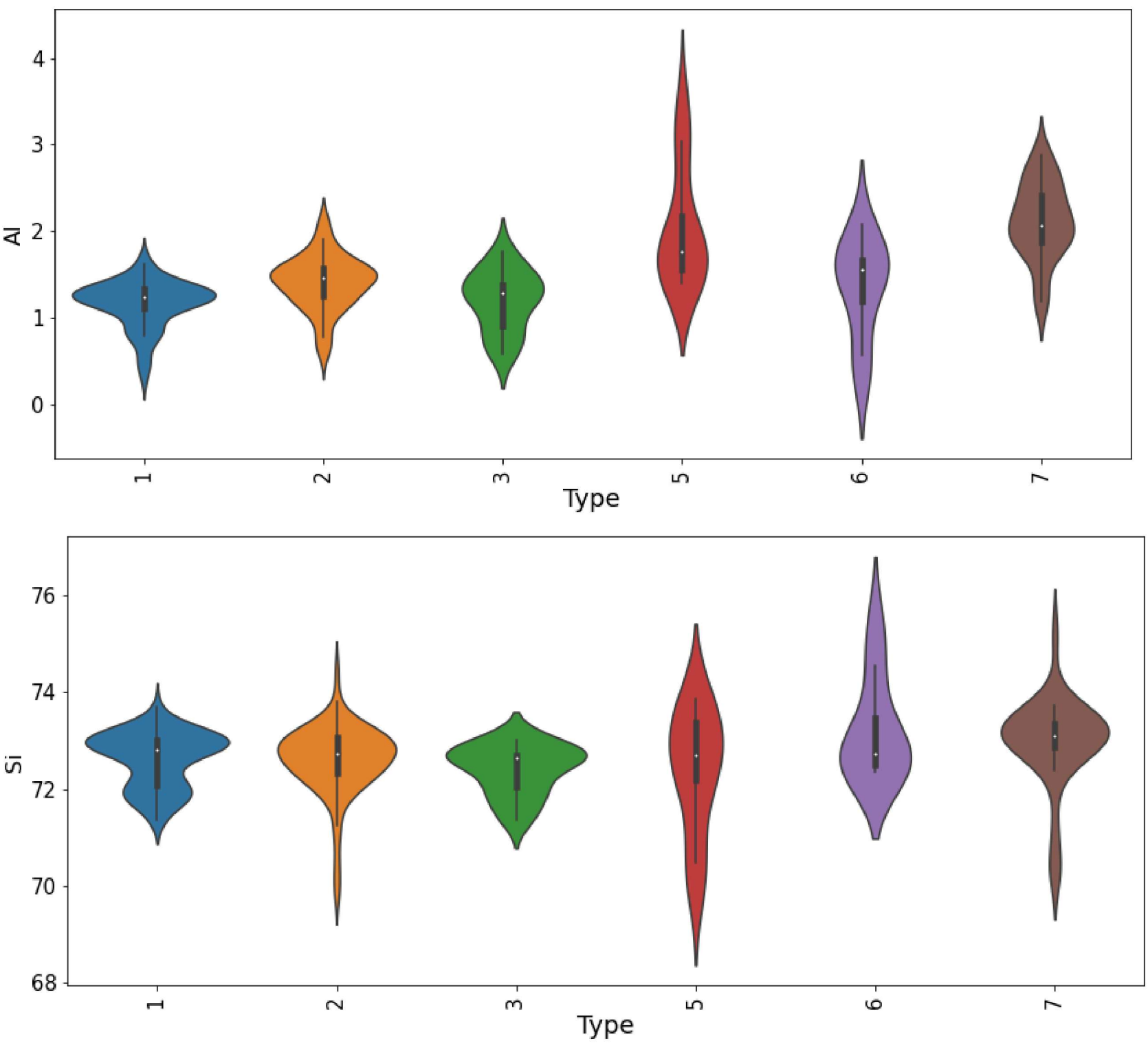


In [14]:
```
for i in glass.columns[:-1]:
    plt.figure(figsize=(14,6))
    sns.violinplot(x = 'Type', y= i,data = glass)
    plt.xticks(rotation = 90, size = 15)
    plt.yticks(size = 15)
    plt.xlabel('Type',fontsize=18)
    plt.ylabel(i, fontsize=16)

    plt.show()
```

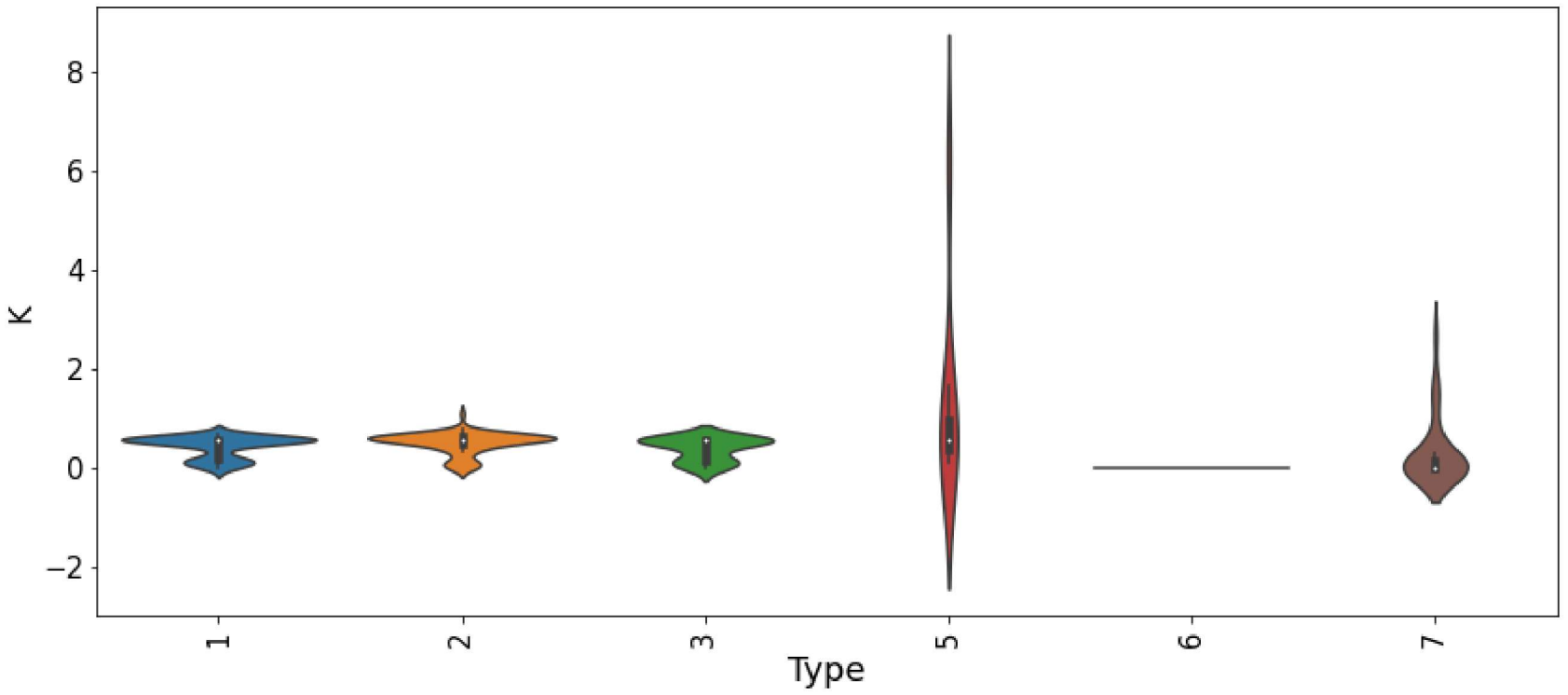


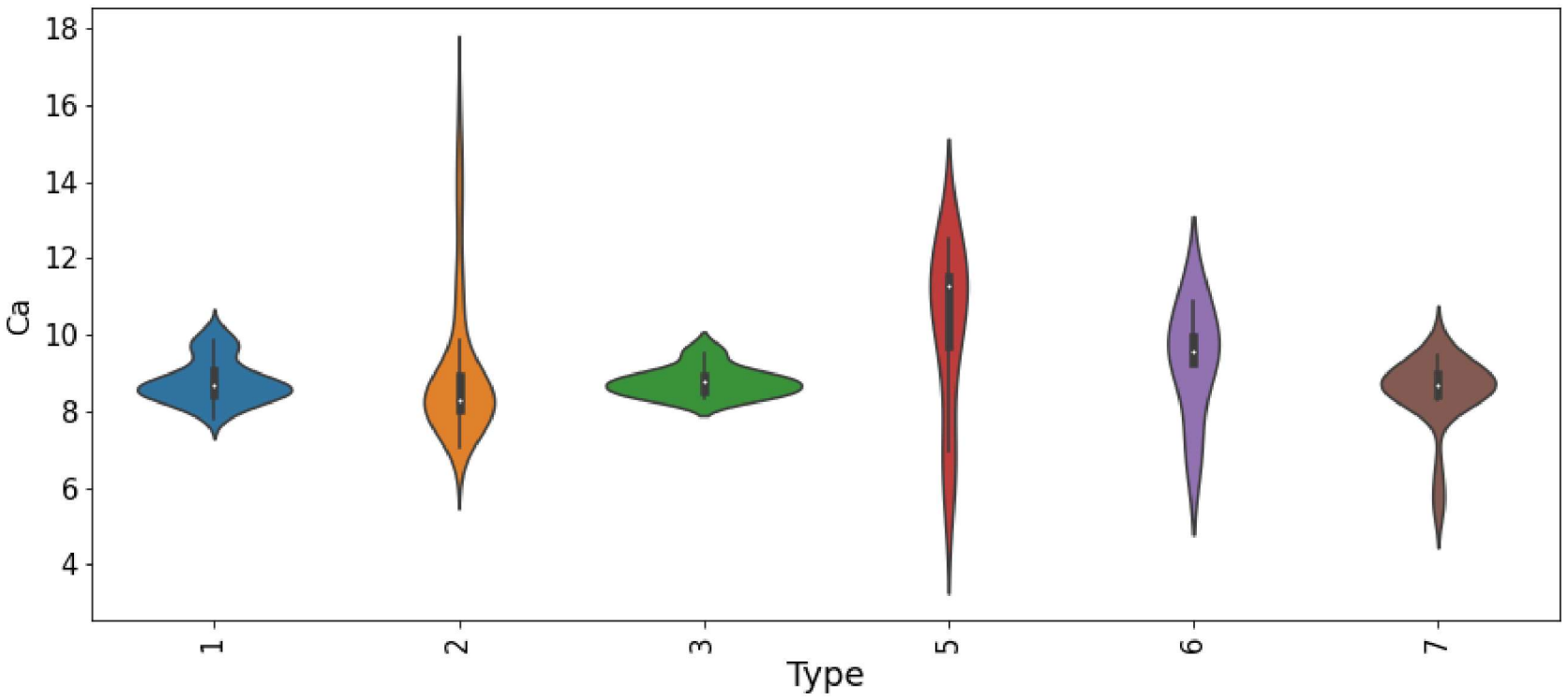

Na
18
16
14
12
10
1
2
3
5
6
7
Type

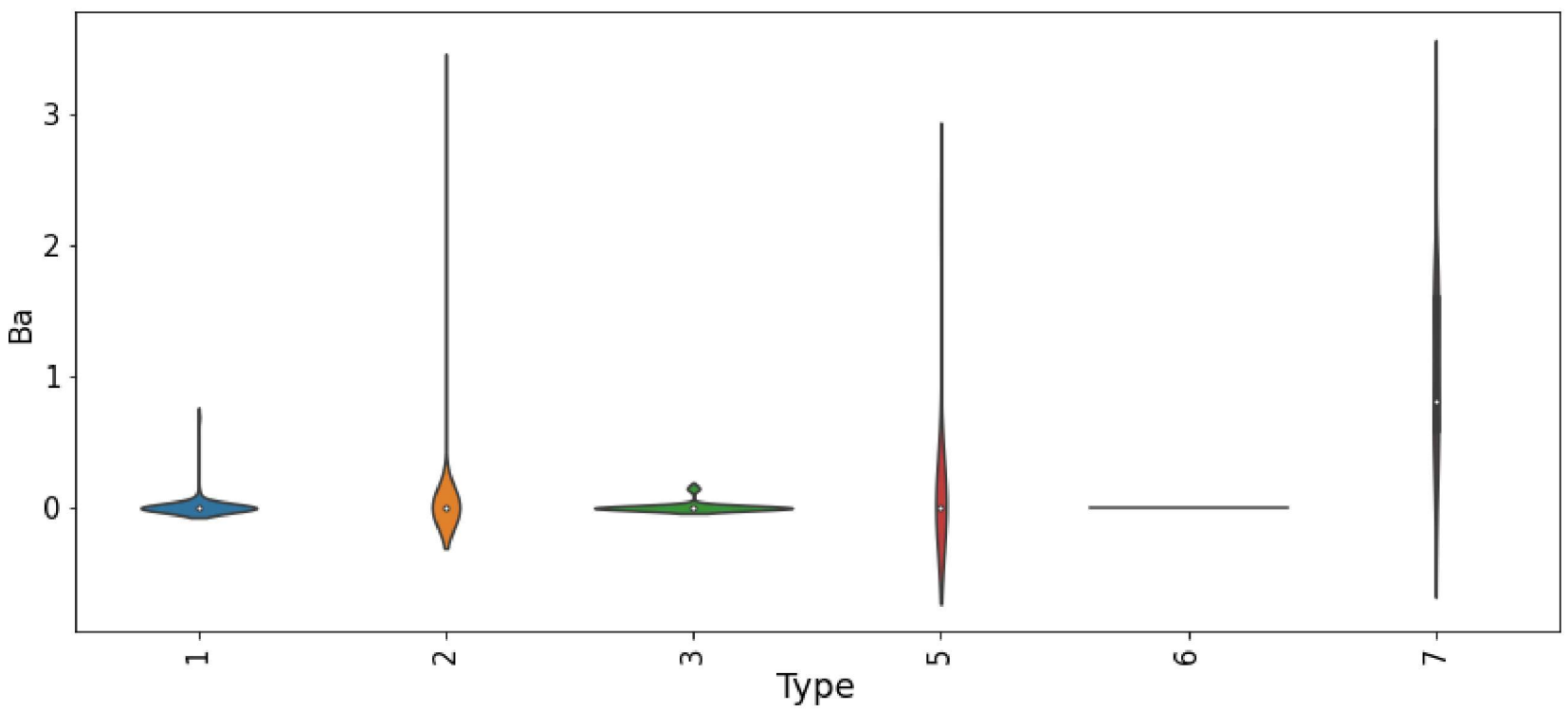

Mg
5
4
3
2
1
0
−1
1
2
3
5
6
7
Type

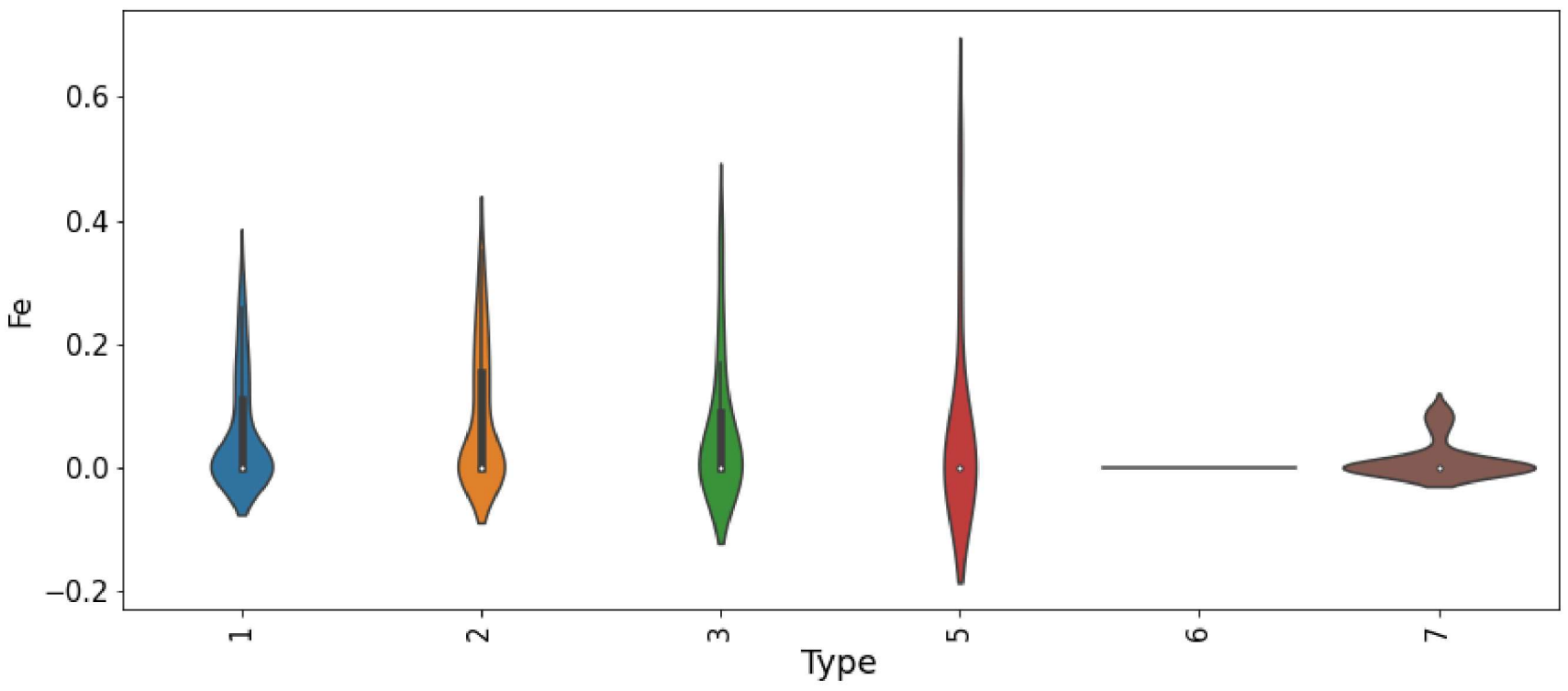

Al
4
3
2
1
0
1
2
3
5
6
7
Type

Si
76
74
72
70
68
1
2
3
5
6
7
Type


K
8
6
4
2
0
−2
1
2
3
5
6
7
Type

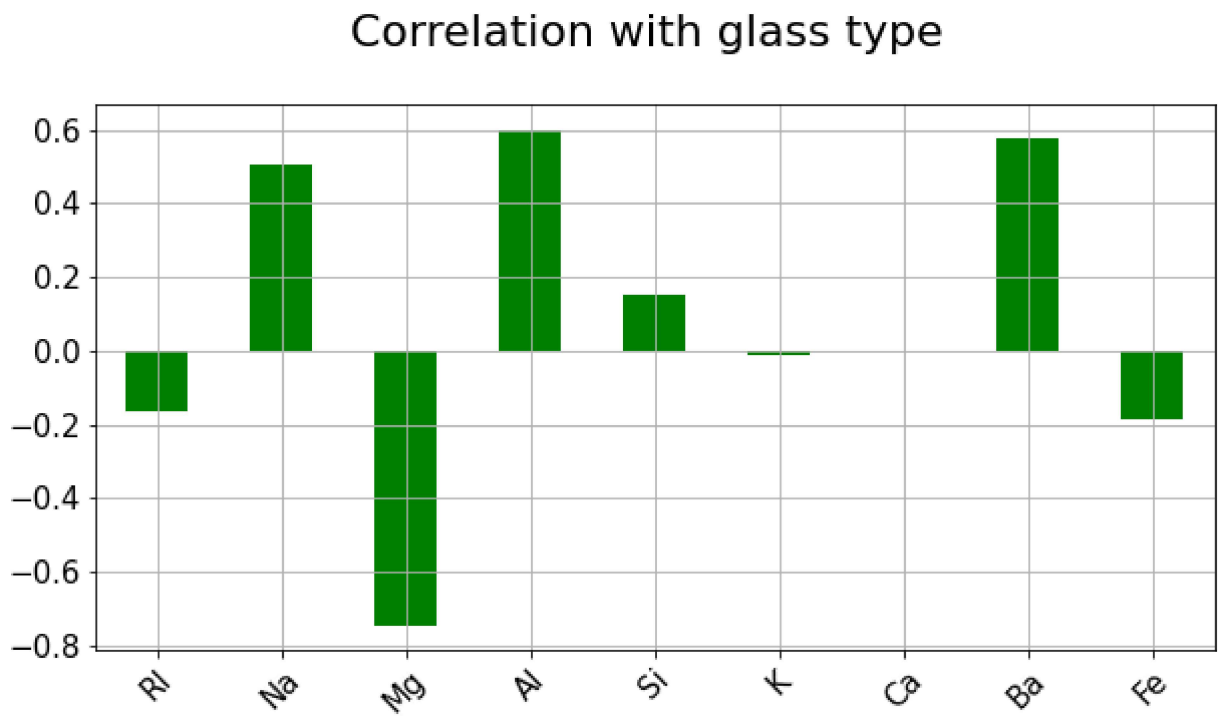

Ca
18
16
14
12
10
8
6
4
1
2
3
5
6
7
Type


Ba
3
2
1
0
1
2
3
5
6
7
Type


Fe
0.6
0.4
0.2
0.0
−0.2
1
2
3
5
6
7
Type

In [26]:
```
data2 = glass.iloc[:,:-1]

correlations = data2.corrwith(glass.Type)
correlations = correlations[correlations!=1]
positive_correlations = correlations[correlations >0].sort_values(ascending = False)
negative_correlations =correlations[correlations<0].sort_values(ascending = False)

correlations.plot.bar(
        figsize = (10, 5),
        fontsize = 15,
        color = 'green',
        rot = 45, grid = True)
plt.title('Correlation with glass type \n',
horizontalalignment="center", fontstyle = "normal",
fontsize = "22", fontfamily = "sans-serif")
```

Out[26]: Text(0.5, 1.0, 'Correlation with glass type \n')



In [16]:
```
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
```

In [20]:
```
scaled_features=scaler.fit_transform(glass.drop('Type',axis=1))
data_head=pd.DataFrame(scaled_features,columns=glass.columns[:-1])
data_head.head()
```

Out[20]:

	RI	Na	Mg	Al	Si	K	Ca	Ba	Fe
0	0.872868	0.284953	1.254639	-0.692442	-1.127082	-0.671705	-0.145766	-0.352877	-0.586451
1	-0.249333	0.591817	0.636168	-0.170460	0.102319	-0.026213	-0.793734	-0.352877	-0.586451
2	-0.721318	0.149933	0.601422	0.190912	0.438787	-0.164533	-0.828949	-0.352877	-0.586451
3	-0.232831	-0.242853	0.698710	-0.310994	-0.052974	0.112107	-0.519052	-0.352877	-0.586451
4	-0.312045	-0.169205	0.650066	-0.411375	0.555256	0.081369	-0.624699	-0.352877	-0.586451

In [21]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(data_head,glass['Type'], test_size=0.3, random_state=42)
```

In [22]:
```
print('Shape of X_train: ', X_train.shape)
print('Shape of X_test: ', X_test.shape)
print('Shape of y_train: ', y_train.shape)
print('Shape of y_test: ', y_test.shape)
```

```
Shape of X_train:  (149, 9)
Shape of X_test:  (65, 9)
Shape of y_train:  (149,)
Shape of y_test:  (65,)
```

KNN model

In [23]:
```
model = KNeighborsClassifier(n_neighbors=5)
model.fit(X_train, y_train)
```

Out[23]: KNeighborsClassifier()

In [28]:
```
pred = model.predict(X_test)
acc= accuracy_score(y_test,pred)
print("The accuracy is  {}".format(acc))
```

```
The accuracy is  0.6461538461538462
```

In [29]:
```
kfold = KFold(n_splits=10)
results = cross_val_score(model, X_train, y_train, cv=kfold)
print(results.mean())
```

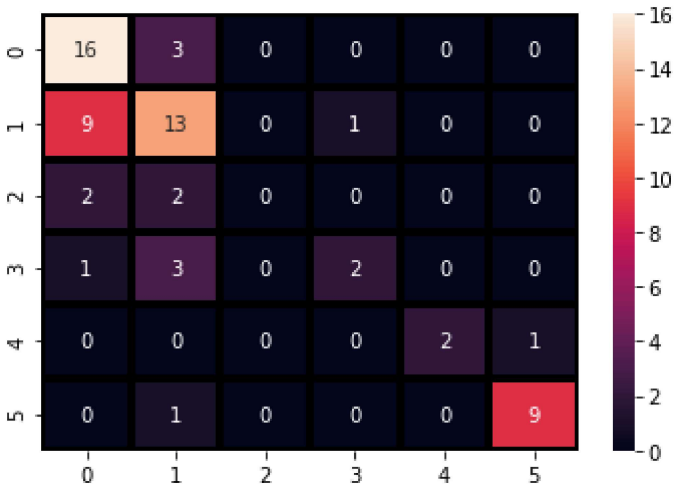
```
0.650952380952381
```

In [31]:
```
sns.heatmap(confusion_matrix(y_test, pred),annot=True,fmt = "d",linecolor="k",linewidths=3)
print('Classification Report ',classification_report(y_test,pred))
```

```
C:\Users\Admin\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1221: UndefinedMetricWarning: Precision and
F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to contro
l this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

Classification Report

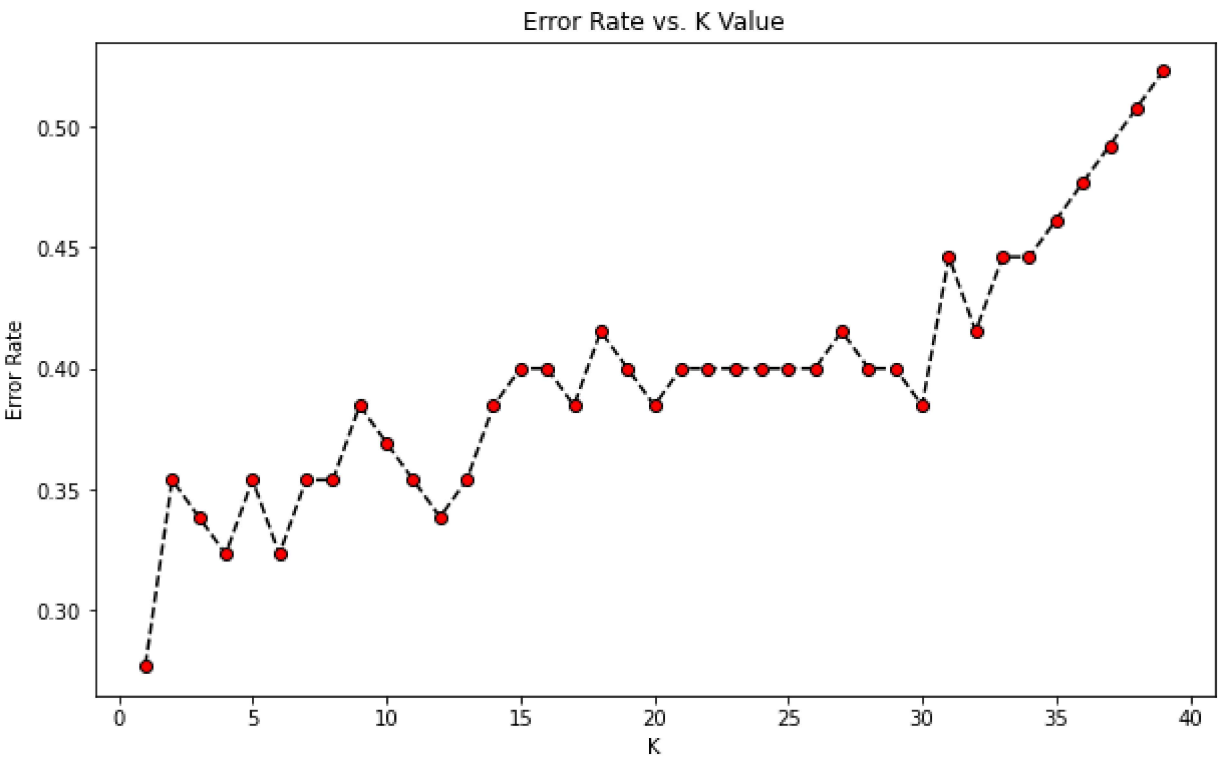
	precision	recall	f1-score	support
1	0.57	0.84	0.68	19
2	0.59	0.57	0.58	23
3	0.00	0.00	0.00	4
5	0.67	0.33	0.44	6
6	1.00	0.67	0.80	3
7	0.90	0.90	0.90	10
accuracy			0.65	65
macro avg	0.62	0.55	0.57	65
weighted avg	0.62	0.65	0.62	65



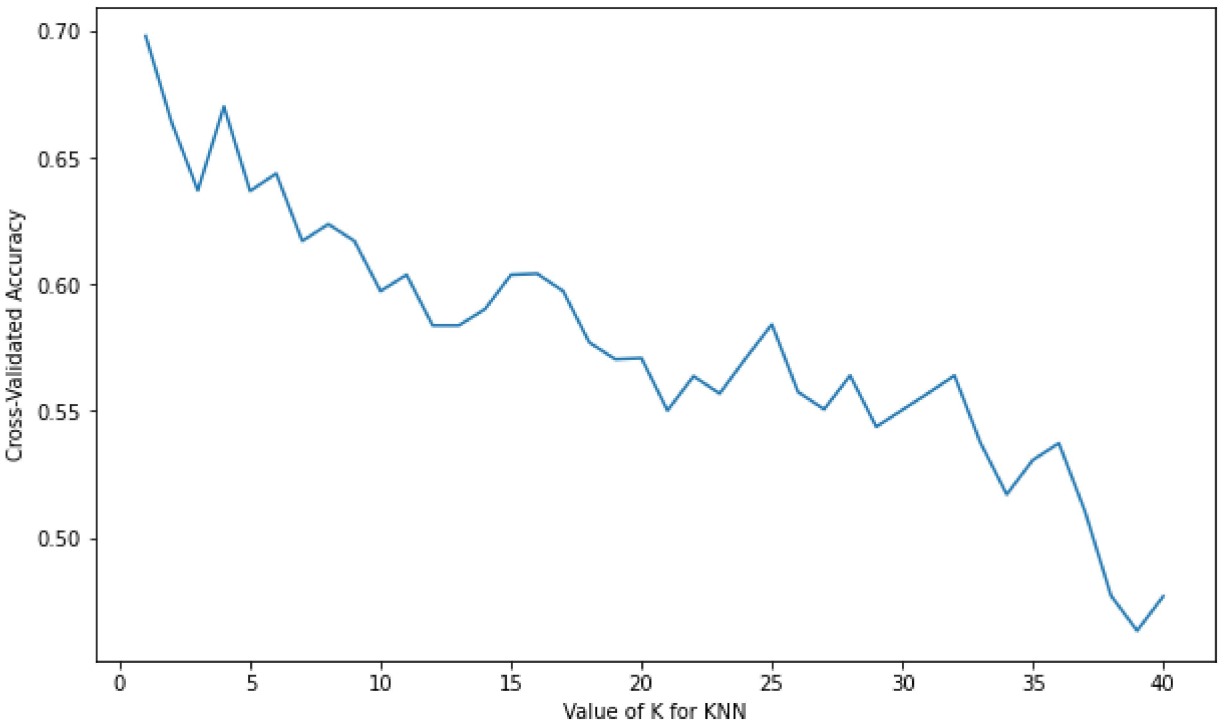
In [33]:
```
err_rate=[]
for i in range(1,40):
    knn=KNeighborsClassifier(n_neighbors=i)
    knn.fit(X_train,y_train)
    predict=knn.predict(X_test)
    err_rate.append(np.mean(predict!=y_test))
```

In [36]:
```
plt.figure(figsize=(10,6))
plt.plot(range(1,40),err_rate,color='black', linestyle='dashed', marker='o',
         markerfacecolor='red', markersize=6)
plt.title('Error Rate vs. K Value')
plt.xlabel('K')
plt.ylabel('Error Rate')
```

Out[36]: Text(0, 0.5, 'Error Rate')



In [39]:
```
import matplotlib.pyplot as plt
%matplotlib inline

# choose k between 1 to 41
k_range = range(1, 41)
k_scores = []

for k in k_range:
    knn = KNeighborsClassifier(n_neighbors=k)
    scores = cross_val_score(knn, X_train, y_train, cv=5)
    k_scores.append(scores.mean())
# plot to see clearly
plt.figure(figsize=(10,6))
plt.plot(k_range, k_scores)
plt.xlabel('Value of K for KNN')
plt.ylabel('Cross-Validated Accuracy')
plt.show()
```



In [58]:
```
knn_model=KNeighborsClassifier(n_neighbors=4)
knn_model.fit(X_train,y_train)
pred_knn=knn_model.predict(X_test)

acc_knn= accuracy_score(y_test,pred_knn)
print("The accuracy is  {}".format(acc_knn))
```

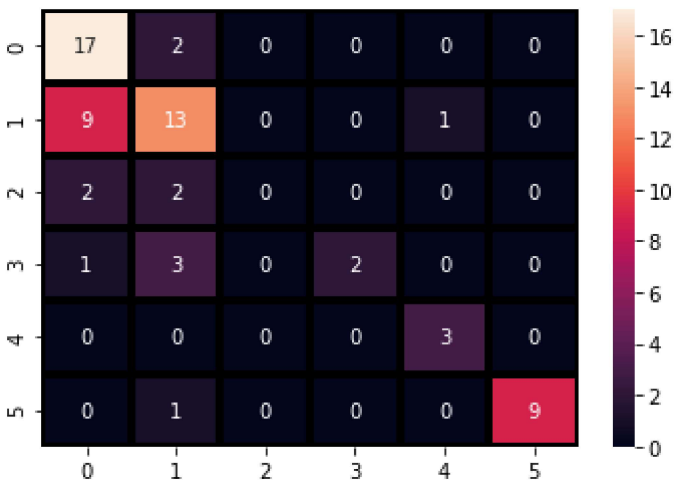
```
The accuracy is  0.676923076923077
```

In [59]:
```
kfold_knn = KFold(n_splits=10)
results_knn = cross_val_score(knn_model, X_train, y_train, cv=kfold_knn)
print(results_knn.mean())
```

```
0.7042857142857143
```

In [60]:
```
sns.heatmap(confusion_matrix(y_test, pred_knn),annot=True,fmt = "d",linecolor="k",linewidths=3)
print('Classification Report ',classification_report(y_test,pred_knn))
```

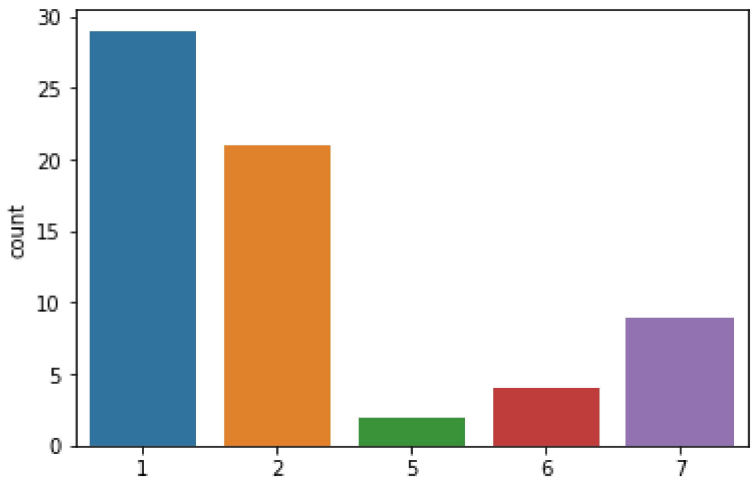
```
C:\Users\Admin\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1221: UndefinedMetricWarning: Precision and
F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to contro
l this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
Classification Report                precision    recall  f1-score   support

           1       0.59      0.89      0.71        19
           2       0.62      0.57      0.59        23
           3       0.00      0.00      0.00         4
           5       1.00      0.33      0.50         6
           6       0.75      1.00      0.86         3
           7       1.00      0.90      0.95        10

    accuracy                           0.68        65
   macro avg       0.66      0.62      0.60        65
weighted avg       0.67      0.68      0.65        65
```



In [61]:
```
sns.countplot(pred_knn)
```

```
C:\Users\Admin\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a key
word arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[61]: <AxesSubplot:ylabel='count'>



In [80]:
```
# parameters selection
kf = KFold(n_splits=15)
grid_params ={
    'n_neighbors':[1,2,3,4,5,6],
    'weights':['uniform','distance'],
    'metric' :['eclidean', 'manhattan']
}

gs = GridSearchCV(KNeighborsClassifier(), grid_params, verbose =10,cv=kf, n_jobs=-1)
gs_results = gs.fit(X_train, y_train)
```

```
Fitting 15 folds for each of 24 candidates, totalling 360 fits
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 8 concurrent workers.
[Parallel(n_jobs=-1)]: Done   2 tasks      | elapsed:    4.8s
[Parallel(n_jobs=-1)]: Done   9 tasks      | elapsed:    4.9s
[Parallel(n_jobs=-1)]: Done  16 tasks      | elapsed:    5.0s
[Parallel(n_jobs=-1)]: Done  25 tasks      | elapsed:    5.0s
[Parallel(n_jobs=-1)]: Done  34 tasks      | elapsed:    5.0s
[Parallel(n_jobs=-1)]: Batch computation too fast (0.1963s.) Setting batch_size=2.
[Parallel(n_jobs=-1)]: Done  45 tasks      | elapsed:    5.1s
[Parallel(n_jobs=-1)]: Done  56 tasks      | elapsed:    5.1s
[Parallel(n_jobs=-1)]: Batch computation too fast (0.0380s.) Setting batch_size=4.
[Parallel(n_jobs=-1)]: Done  74 tasks      | elapsed:    5.1s
[Parallel(n_jobs=-1)]: Batch computation too fast (0.0500s.) Setting batch_size=8.
[Parallel(n_jobs=-1)]: Done 104 tasks      | elapsed:    5.2s
[Parallel(n_jobs=-1)]: Done 168 tasks      | elapsed:    5.2s
[Parallel(n_jobs=-1)]: Batch computation too fast (0.0590s.) Setting batch_size=16.
[Parallel(n_jobs=-1)]: Done 253 tasks      | elapsed:    5.4s
[Parallel(n_jobs=-1)]: Done 298 tasks      | elapsed:    5.4s
[Parallel(n_jobs=-1)]: Done 322 tasks      | elapsed:    5.5s
[Parallel(n_jobs=-1)]: Done 341 tasks      | elapsed:    5.5s
[Parallel(n_jobs=-1)]: Done 360 out of 360 | elapsed:    5.5s finished
```

In [81]:
```
gs_results.best_score_
```

Out[81]:
```
0.711851851851852
```

In [82]:
```
model_final = gs_results.best_estimator_
model_final
```

Out[82]:
```
KNeighborsClassifier(metric='manhattan', n_neighbors=1)
```

In [83]:
```
gs_results.get_params
```

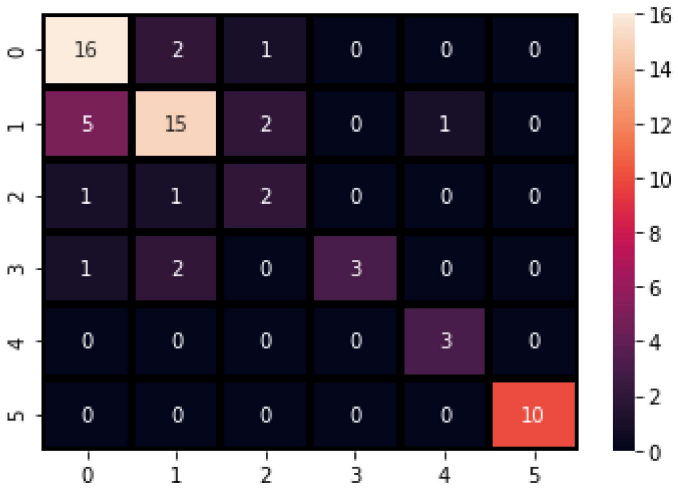
Out[83]:
```
<bound method BaseEstimator.get_params of GridSearchCV(cv=KFold(n_splits=15, random_state=None, shuffle=False),
             estimator=KNeighborsClassifier(), n_jobs=-1,
             param_grid={'metric': ['eclidean', 'manhattan'],
                         'n_neighbors': [1, 2, 3, 4, 5, 6],
                         'weights': ['uniform', 'distance']},
             verbose=10)>
```

In [84]:
```
pred_final = model_final.predict(X_test)
acc_final= accuracy_score(y_test,pred_final)
print("The accuracy is  {}".format(acc_final))
```

```
The accuracy is  0.7538461538461538
```

In [85]:
```
sns.heatmap(confusion_matrix(y_test, pred_final),annot=True,fmt = "d",linecolor="k",linewidths=3)
print('Classification Report ',classification_report(y_test,pred_final))
```

```
Classification Report               precision    recall  f1-score   support

           1       0.70      0.84      0.76        19
           2       0.75      0.65      0.70        23
           3       0.40      0.50      0.44         4
           5       1.00      0.50      0.67         6
           6       0.75      1.00      0.86         3
           7       1.00      1.00      1.00        10

    accuracy                           0.75        65
   macro avg       0.77      0.75      0.74        65
weighted avg       0.77      0.75      0.75        65
```



In [86]:
```
sns.countplot(pred_final)
```

```
C:\Users\Admin\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a key word arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[86]: <AxesSubplot:ylabel='count'>