In [76]:
```
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from sklearn.model_selection import train_test_split,KFold,cross_val_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score,confusion_matrix

import warnings
warnings.filterwarnings('ignore')
```

In [77]:
```
data = pd.read_csv('glass.csv')
data
```

Out[77]:

	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 [78]:
```
data.shape
```

Out[78]:
```
(214, 10)
```

In [79]:
```
data.isna().sum()
```

Out[79]:
```
RI      0
Na      0
Mg      0
Al      0
Si      0
K       0
Ca      0
Ba      0
Fe      0
Type    0
dtype: int64
```

In [80]:
```
data.dtypes
```

Out[80]:
```
RI      float64
Na      float64
Mg      float64
Al      float64
Si      float64
K       float64
Ca      float64
Ba      float64
Fe      float64
Type      int64
dtype: object
```

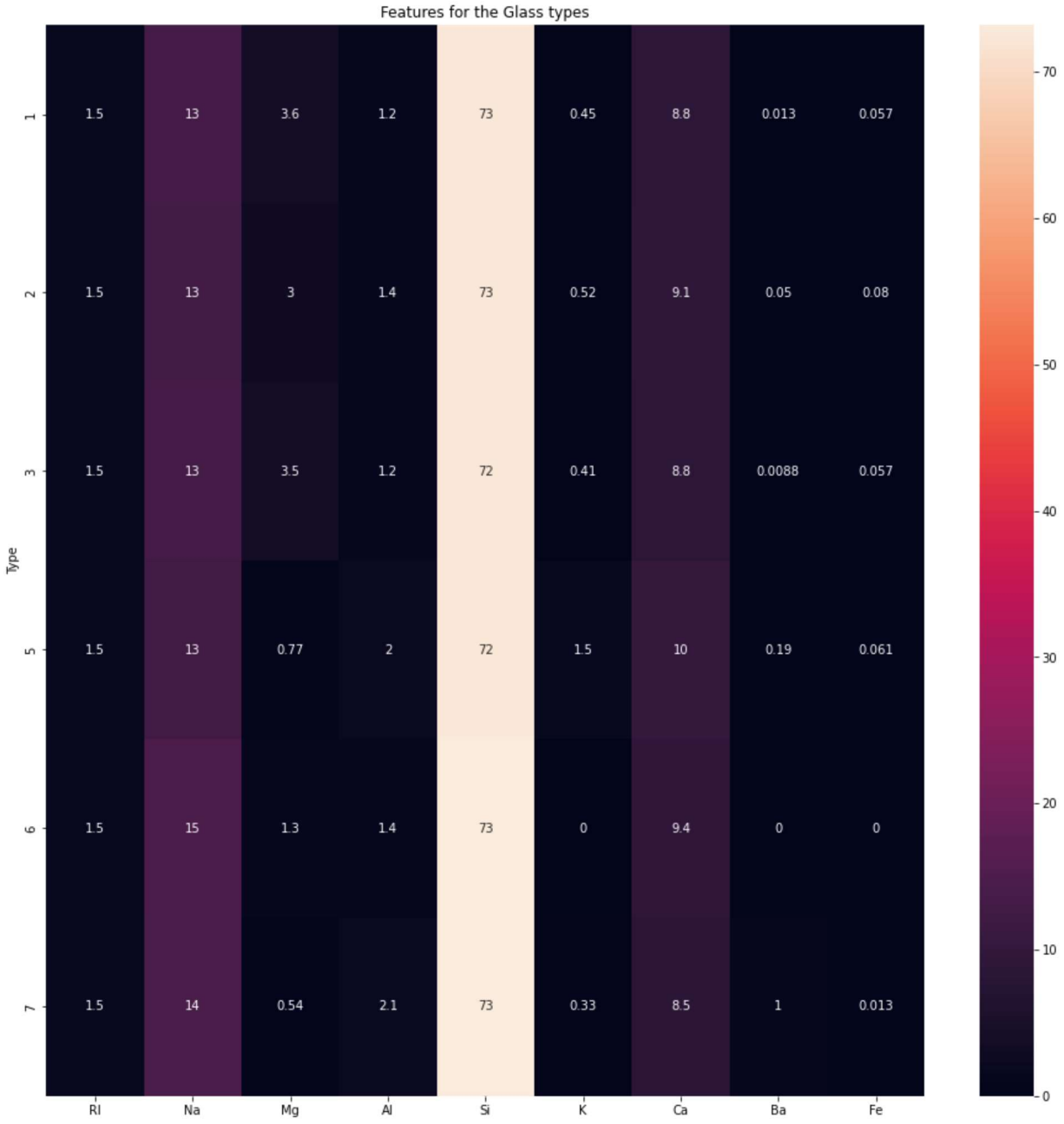
In [81]:
```
data.describe()
```

Out[81]:

	RI	Na	Mg	Al	Si	K	Ca	
count	214.000000	214.000000	214.000000	214.000000	214.000000	214.000000	214.000000	214.00
mean	1.518365	13.407850	2.684533	1.444907	72.650935	0.497056	8.956963	0.17
std	0.003037	0.816604	1.442408	0.499270	0.774546	0.652192	1.423153	0.49
min	1.511150	10.730000	0.000000	0.290000	69.810000	0.000000	5.430000	0.00
25%	1.516522	12.907500	2.115000	1.190000	72.280000	0.122500	8.240000	0.00
50%	1.517680	13.300000	3.480000	1.360000	72.790000	0.555000	8.600000	0.00
75%	1.519157	13.825000	3.600000	1.630000	73.087500	0.610000	9.172500	0.00
max	1.533930	17.380000	4.490000	3.500000	75.410000	6.210000	16.190000	3.15

In [82]:
```
data_temp = data.groupby(by='Type').mean()
plt.figure(figsize=(16,16))
sns.heatmap(data_temp,annot = True)
plt.title('Features for the Glass types')
```

Out[82]: Text(0.5, 1.0, 'Features for the Glass types')



In [83]:
```
x = data.drop('Type',axis =1)
y = data[['Type']]
```

In [84]:
```
_test = train_test_split(x,y,test_size=0.20,random_state=12,stratify=y,shuffle=Tru
```

In [85]:
```
x_train.shape,y_train.shape
```

Out[85]:
```
((171, 9), (171, 1))
```

In [86]:
```
x_test.shape,y_test.shape
```

Out[86]:
```
((43, 9), (43, 1))
```

In [87]:
```
knn = KNeighborsClassifier()
knn.fit(x_train,y_train)
```

Out[87]:
```
KNeighborsClassifier()
```

In [88]:
```
y_pred_train = knn.predict(x_train)
y_pred_train
```

Out[88]:
```
array([2, 7, 1, 2, 2, 2, 2, 2, 1, 1, 2, 1, 3, 1, 2, 1, 2, 2, 7, 2, 1, 1,
       1, 2, 1, 1, 2, 1, 3, 7, 7, 1, 5, 6, 2, 2, 2, 2, 2, 2, 1, 5, 3, 7,
       3, 2, 6, 2, 1, 1, 1, 2, 7, 2, 3, 1, 1, 2, 2, 1, 1, 1, 7, 2, 1, 1,
       2, 1, 7, 1, 2, 2, 2, 2, 6, 1, 7, 1, 2, 1, 2, 1, 2, 2, 1, 2, 2, 2,
       2, 1, 1, 1, 7, 1, 1, 2, 5, 2, 1, 3, 1, 2, 1, 1, 5, 1, 5, 1, 7, 2,
       1, 1, 2, 1, 1, 2, 1, 1, 1, 2, 7, 5, 2, 2, 2, 1, 2, 3, 1, 3, 7, 2,
       1, 1, 1, 2, 1, 2, 1, 2, 7, 2, 7, 1, 2, 2, 2, 2, 1, 1, 1, 1, 2, 6,
       1, 2, 6, 2, 3, 1, 1, 7, 1, 2, 7, 1, 7, 1, 1, 7, 2], dtype=int64)
```

In [89]:
```
print('Accuracy         :',accuracy_score(y_train,y_pred_train))
print('Confusion metrix :\n',confusion_matrix(y_train,y_pred_train))
```

```
Accuracy         : 0.7543859649122807
Confusion metrix :
 [[49  5  2  0  0  0]
 [10 49  1  0  1  0]
 [ 8  0  6  0  0  0]
 [ 0  4  0  5  0  1]
 [ 0  3  0  0  3  1]
 [ 1  3  0  1  1 17]]
```

In [90]:
```
std_scalar = StandardScaler()
std_scalar = std_scalar.fit_transform(x)
x_scaled = pd.DataFrame(data = std_scalar,columns=x.columns)
x_scaled
```

Out[90]:

	RI	Na	Mg	Al	Si	K	Ca	Ba	F
0	0.872868	0.284953	1.254639	-0.692442	-1.127082	-0.671705	-0.145766	-0.352877	-0.58645
1	-0.249333	0.591817	0.636168	-0.170460	0.102319	-0.026213	-0.793734	-0.352877	-0.58645
2	-0.721318	0.149933	0.601422	0.190912	0.438787	-0.164533	-0.828949	-0.352877	-0.58645
3	-0.232831	-0.242853	0.698710	-0.310994	-0.052974	0.112107	-0.519052	-0.352877	-0.58645
4	-0.312045	-0.169205	0.650066	-0.411375	0.555256	0.081369	-0.624699	-0.352877	-0.58645
...	...	...	...	...	...	...	...	...	
209	-0.704815	0.898681	-1.865511	2.881125	-0.052974	-0.640968	0.157088	1.783978	-0.58645
210	-0.500178	1.856097	-1.865511	1.094342	0.529374	-0.763919	-0.392276	2.852405	-0.58645
211	0.754046	1.168721	-1.865511	1.154570	0.995252	-0.763919	-0.364103	2.953200	-0.58645
212	-0.612399	1.193270	-1.865511	0.993960	1.241133	-0.763919	-0.335931	2.812087	-0.58645
213	-0.414363	1.009152	-1.865511	1.275028	0.917606	-0.763919	-0.237327	3.013677	-0.58645

214 rows × 9 columns

In [91]:
```
x_train,x_test,y_train,y_test = train_test_split(x_scaled,y,test_size=0.20,random
```

In [92]:
```
x_train.shape,y_train.shape
```

Out[92]:
```
((171, 9), (171, 1))
```

In [93]:
```
x_test.shape,y_test.shape
```

Out[93]:
```
((43, 9), (43, 1))
```

In [94]:
```
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(x_train,y_train)

y_pred = knn.predict(x_train)
print('Accuracy score :',accuracy_score(y_train,y_pred))
```

```
Accuracy score : 0.7251461988304093
```

In [95]:
```
kfold = KFold(n_splits=5,shuffle=True,random_state=14)
cv_scores = []


for i in range(1,50,2):
    knn_model = KNeighborsClassifier(n_neighbors=i)
    cross_val_scores = cross_val_score(estimator = knn_model,X = x_scaled,y=y,cv=
    print(i,'th Iteration:\n',cross_val_scores.mean().round(4))
    cv_scores.append(cross_val_scores.mean().round(4))
```

```
1 th Iteration:
 0.701
3 th Iteration:
 0.6966
5 th Iteration:
 0.6498
7 th Iteration:
 0.6592
9 th Iteration:
 0.6404
11 th Iteration:
 0.6495
13 th Iteration:
 0.645
15 th Iteration:
 0.6169
17 th Iteration:
 0.6261
19 th Iteration:
 0.6355
21 th Iteration:
 0.6309
23 th Iteration:
 0.6031
25 th Iteration:
 0.5938
27 th Iteration:
 0.589
29 th Iteration:
 0.5796
31 th Iteration:
 0.5657
33 th Iteration:
 0.5797
35 th Iteration:
 0.5703
37 th Iteration:
 0.5564
39 th Iteration:
 0.5704
41 th Iteration:
 0.5425
43 th Iteration:
 0.5333
45 th Iteration:
```

```
 0.5286
47 th Iteration:
 0.524
49 th Iteration:
 0.4822
```

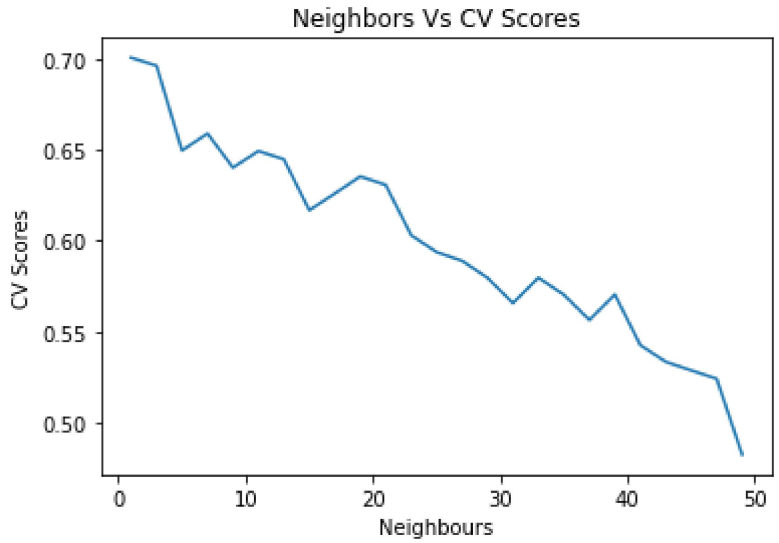
In [96]:
```
cv_scores
```

Out[96]:
```
[0.701,
 0.6966,
 0.6498,
 0.6592,
 0.6404,
 0.6495,
 0.645,
 0.6169,
 0.6261,
 0.6355,
 0.6309,
 0.6031,
 0.5938,
 0.589,
 0.5796,
 0.5657,
 0.5797,
 0.5703,
 0.5564,
 0.5704,
 0.5425,
 0.5333,
 0.5286,
 0.524,
 0.4822]
```

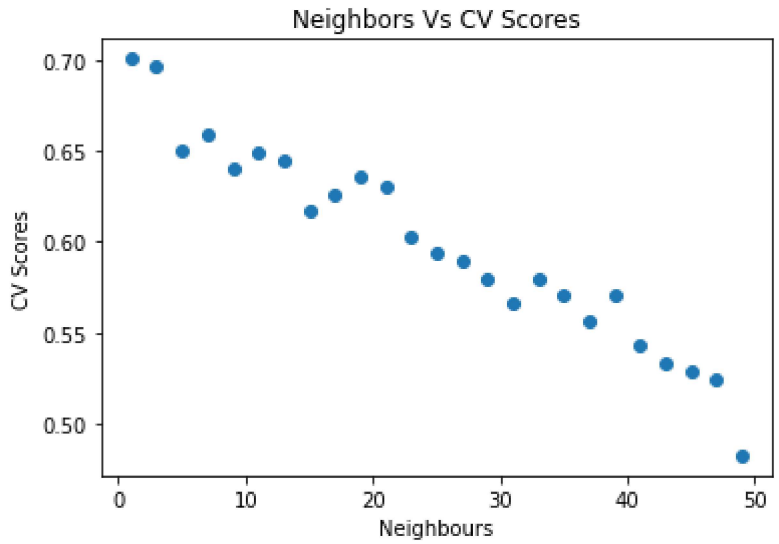
In [97]:
```
max(cv_scores)
```

Out[97]:
```
0.701
```

In [98]:
```
plt.plot(range(1,50,2),cv_scores)
plt.xlabel('Neighbours')
plt.ylabel('CV Scores')
plt.title('Neighbors Vs CV Scores')
plt.show()
```



In [99]:
```
plt.scatter(range(1,50,2),cv_scores)
plt.xlabel('Neighbours')
plt.ylabel('CV Scores')
plt.title('Neighbors Vs CV Scores')
plt.show()
```



In [100]:
```
knn.score(x_train,y_train)
```

Out[100]: 0.7251461988304093

In [101]:
```
knn.score(x_test,y_test)
```

Out[101]: 0.6511627906976745

In [ ]: