In [13]:

```
import pandas as pd
import numpy as np
import pylab as pl
from sklearn import datasets
import matplotlib.pyplot as plt
import sklearn.metrics as sm
from sklearn.cluster import KMeans
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
%matplotlib inline

#https://github.com/VenkateshUV/Comprehending-K-Means-and-KNN-Algorithms/blob/master/Knn%20
```

In [14]:

```
family = pd.read_csv("Family2.csv")
```

In [15]:

```
family
```

Out[15]:

	n_residents	n_small_kids	n_growup_kids	n_elderly	n_disable	family	first_securit
0	1	0	0	0	1	one_eld_dis	surveillanc
1	3	1	0	0	0	wkid	surveillanc
2	2	1	0	0	0	mkid	lack_ener
3	4	1	1	0	0	wkid	lack_ener
4	1	0	0	0	0	one_house	surveillanc
5	5	0	2	1	0	wkid_ed	alarr
6	2	0	0	2	0	eld_dis	surveillanc
7	1	0	0	0	0	one_house	surveillanc
8	2	1	0	0	0	mkid	alarr
9	2	0	0	0	0	cohab	surveillanc
10	4	0	0	2	0	eld_dis	surveillanc
11	2	0	0	0	1	eld_dis	surveillanc
12	5	2	0	0	1	wkid_ed	surveillanc
13	3	0	2	0	0	mkid	lack_ener
14	2	0	0	0	0	cohab	surveillanc
15	2	0	0	0	1	cohab	surveillanc
16	1	0	0	1	0	one_house	surveillanc
17	3	0	0	1	0	eld_dis	surveillanc
18	4	0	0	2	0	eld_dis	surveillanc
19	1	0	0	1	0	one_eld_dis	surveillanc
20	4	1	2	0	0	wkid	surveillanc
21	2	1	0	0	0	mkid	lack_ener
22	4	1	1	0	0	wkid	lack_ener
23	1	0	0	0	0	one_house	surveillanc
24	5	0	2	1	0	wkid_ed	alarr
25	2	0	0	2	0	eld_dis	lack_ener
26	1	0	0	0	0	one_house	surveillanc
27	2	1	0	0	0	mkid	alarr
28	2	0	0	0	0	cohab	surveillanc
29	4	0	0	2	0	eld_dis	surveillanc
30	2	0	0	0	1	eld_dis	surveillanc
31	6	2	0	0	2	wkid_ed	alarr
32	3	0	2	0	0	mkid	lack_ener
33	2	0	0	0	0	cohab	surveillanc

	n_residents	n_small_kids	n_growup_kids	n_elderly	n_disable	family	first_securit
34	2	0	0	0	1	cohab	surveillanc
35	1	0	0	1	0	one_eld_dis	surveillanc
36	3	0	0	1	0	eld_dis	surveillanc
37	4	0	0	2	0	eld_dis	surveillanc
38	4	0	0	0	2	eld_dis	alarr
39	2	0	0	0	0	cohab	lack_ener
40	1	0	0	0	0	one_house	surveillanc
41	2	0	1	0	0	mkid	lack_ener
42	2	0	0	0	0	cohab	surveillanc
43	1	0	0	0	0	one_house	surveillanc
44	1	0	0	0	0	one_house	surveillanc
45	7	0	2	1	0	wkid_ed	alarr
46	2	0	0	2	0	eld_dis	lack_ener
47	1	0	0	0	0	one_house	surveillanc
48	6	0	2	1	0	wkid_ed	alarr
49	2	0	0	2	0	eld_dis	lack_ener
50	2	0	0	0	0	cohab	lack_ener
51	3	0	0	1	0	eld_dis	surveillanc
52	6	2	0	0	0	wkid	alarr
53	1	0	0	0	1	one_eld_dis	surveillanc
54	1	0	0	0	0	one_house	surveillanc

In [16]:

```
family_features = ['n_residents', 'n_small_kids', 'n_growup_kids', 'n_elderly',
       'n_disable']
X = family[family_features]
X
```

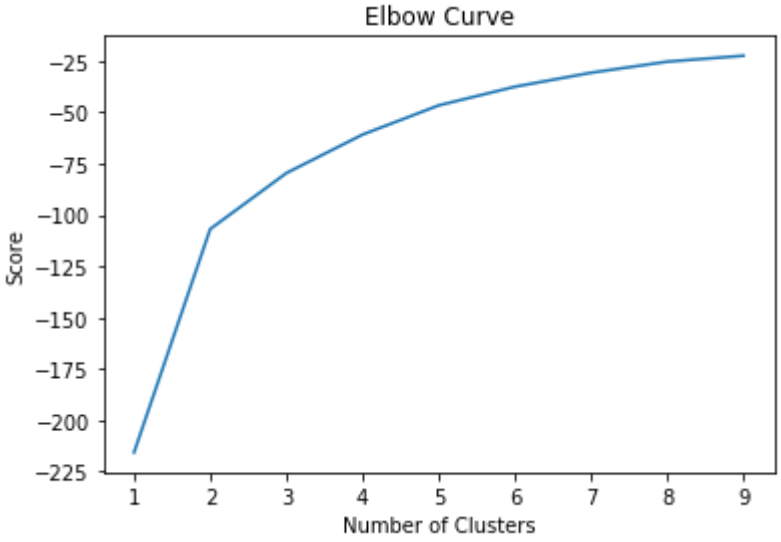
Out[16]:

	n_residents	n_small_kids	n_growup_kids	n_elderly	n_disable
0	1	0	0	0	1
1	3	1	0	0	0
2	2	1	0	0	0
3	4	1	1	0	0
4	1	0	0	0	0
5	5	0	2	1	0
6	2	0	0	2	0
7	1	0	0	0	0
8	2	1	0	0	0
9	2	0	0	0	0
10	4	0	0	2	0
11	2	0	0	0	1
12	5	2	0	0	1
13	3	0	2	0	0
14	2	0	0	0	0
15	2	0	0	0	1
16	1	0	0	1	0
17	3	0	0	1	0
18	4	0	0	2	0
19	1	0	0	1	0
20	4	1	2	0	0
21	2	1	0	0	0
22	4	1	1	0	0
23	1	0	0	0	0
24	5	0	2	1	0
25	2	0	0	2	0
26	1	0	0	0	0
27	2	1	0	0	0
28	2	0	0	0	0
29	4	0	0	2	0
30	2	0	0	0	1
31	6	2	0	0	2

	n_residents	n_small_kids	n_growup_kids	n_elderly	n_disable
32	3	0	2	0	0
33	2	0	0	0	0
34	2	0	0	0	1
35	1	0	0	1	0
36	3	0	0	1	0
37	4	0	0	2	0
38	4	0	0	0	2
39	2	0	0	0	0
40	1	0	0	0	0
41	2	0	1	0	0
42	2	0	0	0	0
43	1	0	0	0	0
44	1	0	0	0	0
45	7	0	2	1	0
46	2	0	0	2	0
47	1	0	0	0	0
48	6	0	2	1	0
49	2	0	0	2	0
50	2	0	0	0	0
51	3	0	0	1	0
52	6	2	0	0	0
53	1	0	0	0	1
54	1	0	0	0	0

In [17]:

```
Nc = range(1, 10)
kmeans = [KMeans(n_clusters=i) for i in Nc]
kmeans
score = [kmeans[i].fit(X).score(X) for i in range(len(kmeans))]
score
pl.plot(Nc,score)
pl.xlabel('Number of Clusters')
pl.ylabel('Score')
pl.title('Elbow Curve')
pl.show()
```



In [18]:

```
family_features = ['n_residents', 'n_small_kids', 'n_growup_kids', 'n_elderly',
       'n_disable']
X = family[family_features]
X.head()
```

Out[18]:

	n_residents	n_small_kids	n_growup_kids	n_elderly	n_disable
0	1	0	0	0	1
1	3	1	0	0	0
2	2	1	0	0	0
3	4	1	1	0	0
4	1	0	0	0	0

In [19]:

```
from sklearn.preprocessing import LabelEncoder
```

In [20]:

```
name_le = LabelEncoder()
y = name_le.fit_transform(family['family'].values)
y1 = name_le.inverse_transform(y)
```

In [21]:

```
d = {'LabelEncoder' : y,
   'Family Type' : y1}
pd.DataFrame(d)
```

Out[21]:

	LabelEncoder	Family Type
0	3	one_eld_dis
1	5	wkid
2	2	mkid
3	5	wkid
4	4	one_house
5	6	wkid_ed
6	1	eld_dis
7	4	one_house
8	2	mkid
9	0	cohab
10	1	eld_dis
11	1	eld_dis
12	6	wkid_ed
13	2	mkid
14	0	cohab
15	0	cohab
16	4	one_house
17	1	eld_dis
18	1	eld_dis
19	3	one_eld_dis
20	5	wkid
21	2	mkid
22	5	wkid
23	4	one_house
24	6	wkid_ed
25	1	eld_dis
26	4	one_house
27	2	mkid
28	0	cohab
29	1	eld_dis
30	1	eld_dis
31	6	wkid_ed
32	2	mkid

	LabelEncoder	Family Type
33	0	cohab
34	0	cohab
35	3	one_eld_dis
36	1	eld_dis
37	1	eld_dis
38	1	eld_dis
39	0	cohab
40	4	one_house
41	2	mkid
42	0	cohab
43	4	one_house
44	4	one_house
45	6	wkid_ed
46	1	eld_dis
47	4	one_house
48	6	wkid_ed
49	1	eld_dis
50	0	cohab
51	1	eld_dis
52	5	wkid
53	3	one_eld_dis
54	4	one_house

In [22]:

```
y
```

Out[22]:

```
array([3, 5, 2, 5, 4, 6, 1, 4, 2, 0, 1, 1, 6, 2, 0, 0, 4, 1, 1, 3, 5, 2,
       5, 4, 6, 1, 4, 2, 0, 1, 1, 6, 2, 0, 0, 3, 1, 1, 1, 0, 4, 2, 0, 4,
       4, 6, 1, 4, 6, 1, 0, 1, 5, 3, 4])
```

In [23]:

```
family.head(10)
```

Out[23]:

	n_residents	n_small_kids	n_growup_kids	n_elderly	n_disable	family	first_security	s
0	1	0	0	0	1	one_eld_dis	surveillance	
1	3	1	0	0	0	wkid	surveillance	
2	2	1	0	0	0	mkid	lack_energ	
3	4	1	1	0	0	wkid	lack_energ	
4	1	0	0	0	0	one_house	surveillance	
5	5	0	2	1	0	wkid_ed	alarm	
6	2	0	0	2	0	eld_dis	surveillance	
7	1	0	0	0	0	one_house	surveillance	
8	2	1	0	0	0	mkid	alarm	
9	2	0	0	0	0	cohab	surveillance	

In [24]:

```
X.insert(5,'type',y)
```

In [25]:

```
X.head()
```

Out[25]:

	n_residents	n_small_kids	n_growup_kids	n_elderly	n_disable	type
0	1	0	0	0	1	3
1	3	1	0	0	0	5
2	2	1	0	0	0	2
3	4	1	1	0	0	5
4	1	0	0	0	0	4

In [26]:

```
df_family = pd.DataFrame(X, columns = family_features + ['type'])
```

In [27]:

```
df_family.replace({'type': {0: 'Cohabit', 1:'Idoso-Defic',2:'Monoparental-kid',3:'Idoso-Soz
```

Out[27]:

	n_residents	n_small_kids	n_growup_kids	n_elderly	n_disable	type
0	1	0	0	0	1	Idoso-Sozinho
1	3	1	0	0	0	Pais-kid
2	2	1	0	0	0	Monoparental-kid
3	4	1	1	0	0	Pais-kid
4	1	0	0	0	0	Sozinho
5	5	0	2	1	0	Idoso-Defic-kid
6	2	0	0	2	0	Idoso-Defic
7	1	0	0	0	0	Sozinho
8	2	1	0	0	0	Monoparental-kid
9	2	0	0	0	0	Cohabit
10	4	0	0	2	0	Idoso-Defic
11	2	0	0	0	1	Idoso-Defic
12	5	2	0	0	1	Idoso-Defic-kid
13	3	0	2	0	0	Monoparental-kid
14	2	0	0	0	0	Cohabit
15	2	0	0	0	1	Cohabit
16	1	0	0	1	0	Sozinho
17	3	0	0	1	0	Idoso-Defic
18	4	0	0	2	0	Idoso-Defic
19	1	0	0	1	0	Idoso-Sozinho
20	4	1	2	0	0	Pais-kid
21	2	1	0	0	0	Monoparental-kid
22	4	1	1	0	0	Pais-kid
23	1	0	0	0	0	Sozinho
24	5	0	2	1	0	Idoso-Defic-kid
25	2	0	0	2	0	Idoso-Defic
26	1	0	0	0	0	Sozinho
27	2	1	0	0	0	Monoparental-kid
28	2	0	0	0	0	Cohabit
29	4	0	0	2	0	Idoso-Defic
30	2	0	0	0	1	Idoso-Defic
31	6	2	0	0	2	Idoso-Defic-kid
32	3	0	2	0	0	Monoparental-kid
33	2	0	0	0	0	Cohabit

	n_residents	n_small_kids	n_growup_kids	n_elderly	n_disable	type
34	2	0	0	0	1	Cohabit
35	1	0	0	1	0	Idoso-Sozinho
36	3	0	0	1	0	Idoso-Defic
37	4	0	0	2	0	Idoso-Defic
38	4	0	0	0	2	Idoso-Defic
39	2	0	0	0	0	Cohabit
40	1	0	0	0	0	Sozinho
41	2	0	1	0	0	Monoparental-kid
42	2	0	0	0	0	Cohabit
43	1	0	0	0	0	Sozinho
44	1	0	0	0	0	Sozinho
45	7	0	2	1	0	Idoso-Defic-kid
46	2	0	0	2	0	Idoso-Defic
47	1	0	0	0	0	Sozinho
48	6	0	2	1	0	Idoso-Defic-kid
49	2	0	0	2	0	Idoso-Defic
50	2	0	0	0	0	Cohabit
51	3	0	0	1	0	Idoso-Defic
52	6	2	0	0	0	Pais-kid
53	1	0	0	0	1	Idoso-Sozinho
54	1	0	0	0	0	Sozinho

In [28]:

```
X = df_family.iloc[:, :-1].values
y = df_family.iloc[:, 5].values
```

In [29]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20)
```

In [30]:

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(X_train)

X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

In [31]:

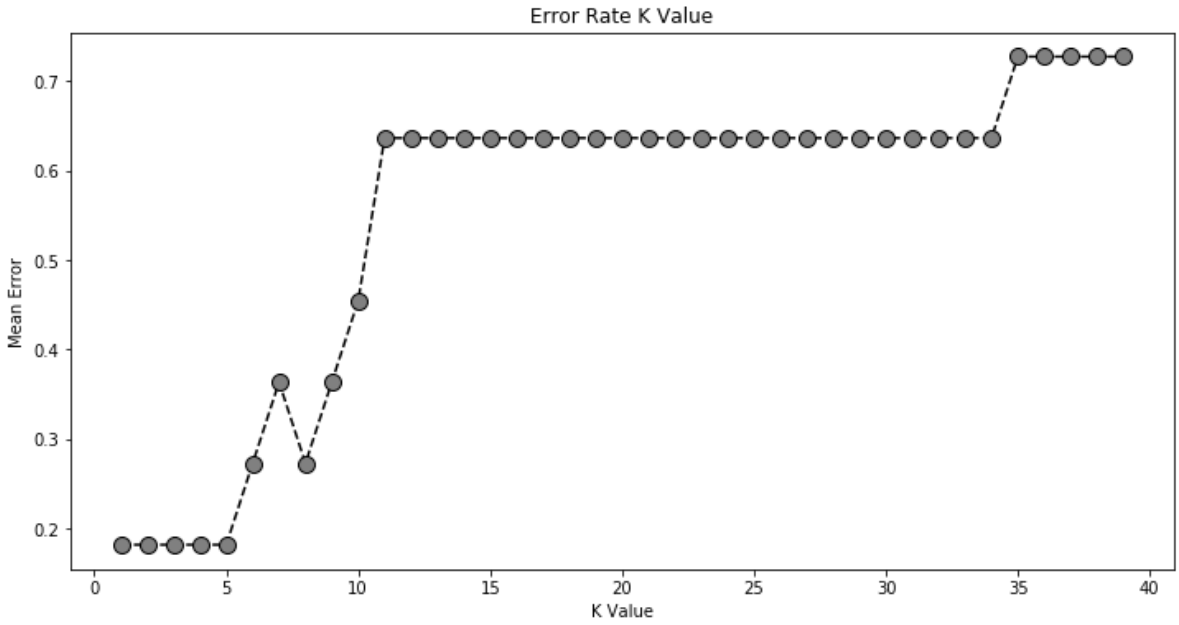
```
error = []

# Calculating error for K values between 1 and 40
for i in range(1, 40):
    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(X_train, y_train)
    pred_i = knn.predict(X_test)
    error.append(np.mean(pred_i != y_test))
```

In [32]:

```
plt.figure(figsize=(12, 6))
plt.plot(range(1, 40), error, color='black', linestyle='dashed', marker='o',
         markerfacecolor='grey', markersize=10)
plt.title('Error Rate K Value')
plt.xlabel('K Value')
plt.ylabel('Mean Error')
```

Out[32]:

```
Text(0, 0.5, 'Mean Error')
```



In [33]:

```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors=5)
classifier.fit(X_train, y_train)
```

Out[33]:

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=5, p=2,
                     weights='uniform')
```

In [34]:

```
y_pred = classifier.predict(X_test)
```

In [35]:

```
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
[[2 0 0 0 0 0]
 [1 2 0 0 0 0]
 [0 0 1 0 0 0]
 [0 0 0 2 0 0]
 [0 0 0 0 1 0]
 [0 0 0 0 1 1]]
              precision    recall  f1-score   support

           0       0.67      1.00      0.80         2
           1       1.00      0.67      0.80         3
           2       1.00      1.00      1.00         1
           4       1.00      1.00      1.00         2
           5       0.50      1.00      0.67         1
           6       1.00      0.50      0.67         2

    accuracy                           0.82        11
   macro avg       0.86      0.86      0.82        11
weighted avg       0.89      0.82      0.82        11
```

In [36]:

```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors=7)
classifier.fit(X_train, y_train)
```

Out[36]:

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=7, p=2,
                     weights='uniform')
```

In [37]:

```
y_pred = classifier.predict(X_test)
```

In [38]:

```
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
[[2 0 0 0 0 0]
 [1 2 0 0 0 0]
 [0 0 0 0 0 1]
 [0 0 0 2 0 0]
 [0 0 0 0 0 1]
 [0 0 0 0 1 1]]
              precision    recall  f1-score   support

           0       0.67      1.00      0.80         2
           1       1.00      0.67      0.80         3
           2       0.00      0.00      0.00         1
           4       1.00      1.00      1.00         2
           5       0.00      0.00      0.00         1
           6       0.33      0.50      0.40         2

    accuracy                           0.64        11
   macro avg       0.50      0.53      0.50        11
weighted avg       0.64      0.64      0.62        11


C:\Users\gabid\anaconda3\lib\site-packages\sklearn\metrics\_classification.p
y:1272: UndefinedMetricWarning: Precision and F-score are ill-defined and be
ing set to 0.0 in labels with no predicted samples. Use `zero_division` para
meter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [ ]:

```
```