In [70]:

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

In [71]:

```
data = pd.read_csv('smart_houses.csv')
```

In [72]:

```
data.head()
```

Out[72]:

	n_moradores	idosos	def_loc	def_b_v	def_cog	def_aud	comodos	r_anual	classe	
0	2	1	0	1	0	0	3	29644.0	baixa	569
1	2	1	0	1	0	0	3	29600.0	baixa	569
2	2	2	0	0	0	0	4	30000.0	baixa	405
3	5	1	1	0	0	0	12	170800.0	alta	1579
4	5	1	1	0	0	0	12	190800.0	alta	1575

In [73]:

```
cols_to_use = ['n_moradores','idosos','def_loc','def_b_v','def_cog','def_aud','comodos',
               'r_anual','A','B','C','D','E','instalacao','preco','crianca']
```

In [74]:

```
name_le = LabelEncoder()
classe = name_le.fit_transform(data['classe'].values)
classe
```

Out[74]:

```
array([1, 1, 1, 0, 0, 0, 2, 2, 1, 2, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 1, 0,
       0, 0, 2, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 2, 2, 2, 2, 1, 2, 1, 2, 0,
       2, 0, 2, 2, 1, 1, 0, 2, 2, 2, 2, 2, 2, 2, 1, 0, 0, 2, 2, 2, 2, 2,
       2, 0, 2, 2, 2, 2, 2, 2, 2, 2, 1, 1, 0, 2, 1, 2, 2, 0, 0, 0, 2, 2,
       2, 2, 2, 1, 1, 2, 2, 2, 1, 1, 1, 1, 1, 2, 2, 2])
```

In [75]:

```
X = data[cols_to_use]
X.insert(15,'classe',classe)
X
```

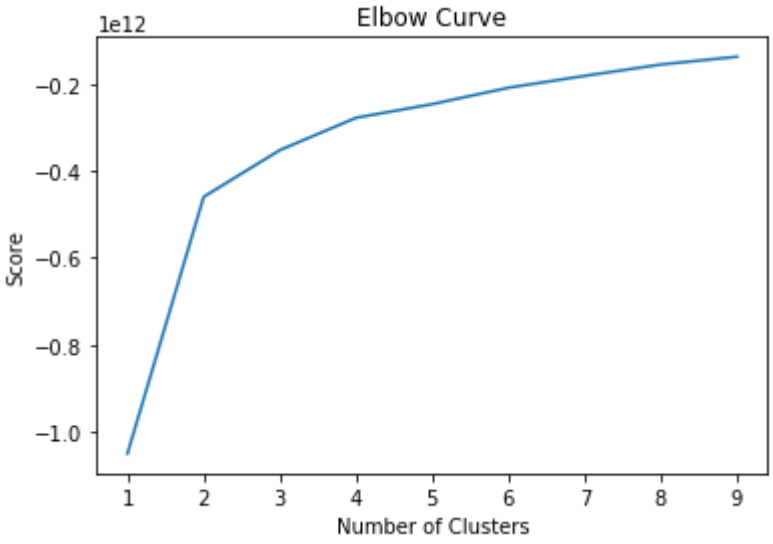
Out[75]:

	n_moradores	idosos	def_loc	def_b_v	def_cog	def_aud	comodos	r_anual	A
0	2	1	0	1	0	0	3	29644.0	56985.70
1	2	1	0	1	0	0	3	29600.0	56985.70
2	2	2	0	0	0	0	4	30000.0	40510.00
3	5	1	1	0	0	0	12	170800.0	157904.00
4	5	1	1	0	0	0	12	190800.0	157564.00
...	...	...	...	...	...	...	...	...	...
99	1	0	0	1	0	0	4	14860.0	37600.00
100	1	0	0	1	0	0	4	14800.0	37600.00
101	5	0	1	0	0	1	5	105600.0	98777.00
102	4	0	1	0	0	1	5	103600.0	98753.00
103	5	0	1	1	0	0	7	146700.0	8965.14

104 rows × 17 columns

In [76]:

```
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 [77]:

```
name_le = LabelEncoder()
y = name_le.fit_transform(data['tipo_sh'].values)
#y1 = name_le.inverse_transform(y)
```

In [78]:

```
y
```

Out[78]:

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

In [79]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30)
```

In [80]:

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

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

In [81]:

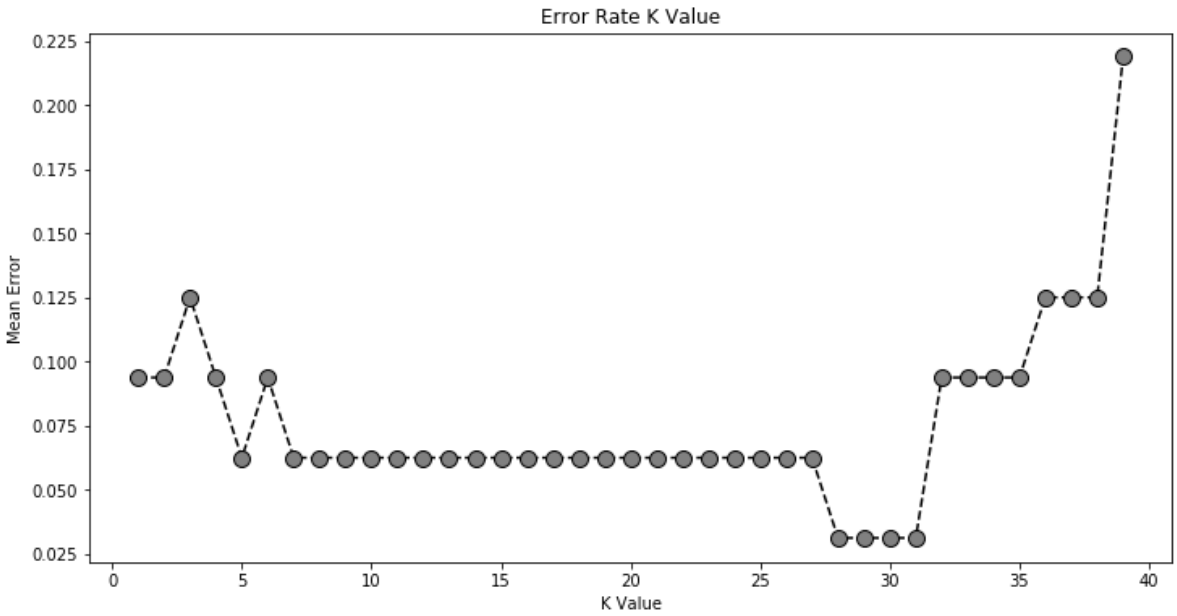
```
error = []

# 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 [82]:

```
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[82]:

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



In [83]:

```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors=3)
classifier.fit(X_train, y_train)
```

Out[83]:

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=3, p=2,
                     weights='uniform')
```

In [84]:

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

In [85]:

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

```
[[8 0 0 0 1]
 [0 8 1 0 0]
 [0 0 5 0 0]
 [0 0 1 4 0]
 [1 0 0 0 3]]
              precision    recall  f1-score   support

           1       0.89      0.89      0.89         9
           2       1.00      0.89      0.94         9
           3       0.71      1.00      0.83         5
           4       1.00      0.80      0.89         5
           5       0.75      0.75      0.75         4

    accuracy                           0.88        32
   macro avg       0.87      0.87      0.86        32
weighted avg       0.89      0.88      0.88        32
```

In [86]:

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

Out[86]:

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

In [87]:

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

In [88]:

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

```
[[9 0 0 0 0]
 [0 9 0 0 0]
 [0 0 5 0 0]
 [0 0 1 4 0]
 [0 1 0 0 3]]
              precision    recall  f1-score   support

           1       1.00      1.00      1.00         9
           2       0.90      1.00      0.95         9
           3       0.83      1.00      0.91         5
           4       1.00      0.80      0.89         5
           5       1.00      0.75      0.86         4

    accuracy                           0.94        32
   macro avg       0.95      0.91      0.92        32
weighted avg       0.95      0.94      0.94        32
```

In [89]:

```
print (pd.crosstab(y_test, y_pred, rownames=['Real'], colnames=['Predito'], margins=True))
```

```
Predito  1   2  3  4  5  All
Real
1        9   0  0  0  0    9
2        0   9  0  0  0    9
3        0   0  5  0  0    5
4        0   0  1  4  0    5
5        0   1  0  0  3    4
All      9  10  6  4  3   32
```

In [90]:

```
print ("KMeans accuracy score : ",accuracy_score(y_test, y_pred))
```

```
KMeans accuracy score :  0.9375
```

In [91]:

```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors=6)
classifier.fit(X_train, y_train)

y_pred = classifier.predict(X_test)

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

```
[[8 0 0 0 1]
 [0 9 0 0 0]
 [0 0 5 0 0]
 [0 0 1 4 0]
 [0 1 0 0 3]]
              precision    recall  f1-score   support

           1       1.00      0.89      0.94         9
           2       0.90      1.00      0.95         9
           3       0.83      1.00      0.91         5
           4       1.00      0.80      0.89         5
           5       0.75      0.75      0.75         4

    accuracy                           0.91        32
   macro avg       0.90      0.89      0.89        32
weighted avg       0.91      0.91      0.91        32
```

In [92]:

```
print (pd.crosstab(y_test, y_pred, rownames=['Real'], colnames=['Predito'], margins=True))
```

```
Predito  1   2  3  4  5  All
Real
1        8   0  0  0  1    9
2        0   9  0  0  0    9
3        0   0  5  0  0    5
4        0   0  1  4  0    5
5        0   1  0  0  3    4
All      8  10  6  4  4   32
```

In [93]:

```
print ("KMeans accuracy score : ",accuracy_score(y_test, y_pred))
```

```
KMeans accuracy score :  0.90625
```

In [94]:

```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors=11)
classifier.fit(X_train, y_train)

y_pred = classifier.predict(X_test)

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

```
[[9 0 0 0 0]
 [0 9 0 0 0]
 [0 0 5 0 0]
 [0 0 1 4 0]
 [0 0 1 0 3]]
              precision    recall  f1-score   support

           1       1.00      1.00      1.00         9
           2       1.00      1.00      1.00         9
           3       0.71      1.00      0.83         5
           4       1.00      0.80      0.89         5
           5       1.00      0.75      0.86         4

    accuracy                           0.94        32
   macro avg       0.94      0.91      0.92        32
weighted avg       0.96      0.94      0.94        32
```

In [95]:

```
print (pd.crosstab(y_test, y_pred, rownames=['Real'], colnames=['Predito'], margins=True))
```

```
Predito  1  2  3  4  5  All
Real
1        9  0  0  0  0    9
2        0  9  0  0  0    9
3        0  0  5  0  0    5
4        0  0  1  4  0    5
5        0  0  1  0  3    4
All      9  9  7  4  3   32
```

In [96]:

```
print ("KMeans accuracy score : ",accuracy_score(y_test, y_pred))
```

```
KMeans accuracy score :  0.9375
```