In [30]:

```
import pandas as pd
```

In [31]:

```
data = pd.read_csv('Family2.csv')
```

In [41]:

```
data.head()
```

Out[41]:

	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	

In [32]:

```
cols_to_use = ['n_residents','n_small_kids','n_growup_kids','n_elderly','n_disable']
X = data[cols_to_use]
```

In [33]:

```
from sklearn.preprocessing import LabelEncoder
name_le = LabelEncoder()
y = name_le.fit_transform(data['family'].values)
#y1 = name_le.inverse_transform(y)
```

In [40]:

```
y
```

Out[40]:

```
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 [42]:

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

In [49]:

```
X.head(10)
```

Out[49]:

	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
5	5	0	2	1	0	6
6	2	0	0	2	0	1
7	1	0	0	0	0	4
8	2	1	0	0	0	2
9	2	0	0	0	0	0

In [43]:

```
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
```

In [44]:

```
si = SimpleImputer()
ss =  StandardScaler()
knc = KNeighborsClassifier(n_neighbors=5)
pipe = Pipeline(steps = [('preprocessamento',si),
                         ('standard',ss ),
                         ('model',knc )])
```

In [58]:

```
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 [59]:

```
knc.fit(X_train, y_train)
```

Out[59]:

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

In [60]:

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

In [61]:

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

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

           0       1.00      1.00      1.00         4
           1       1.00      1.00      1.00         3
           2       1.00      1.00      1.00         3
           4       0.80      1.00      0.89         4
           5       1.00      0.50      0.67         2
           6       1.00      1.00      1.00         1

    accuracy                           0.94        17
   macro avg       0.97      0.92      0.93        17
weighted avg       0.95      0.94      0.93        17
```

In [ ]:

```

```