In [1]:

```
import pandas as  pd
import numpy as np
import matplotlib.pyplot as plt
import  seaborn as sns
from sklearn.feature_selection import chi2
from sklearn.feature_selection import SelectKBest
import plotly.express as px
import plotly.graph_objects as go
```

In [5]:

```
data=pd.read_csv(r'C:\Users\sunny\AppData\Local\Temp\Temp1_Dataset for the project (1).zip\Dataset for the project\train.csv')
data
```

Out[5]:

	Id	v2a1	hacdor	rooms	hacapo	v14a	refrig	v18q	v18q1	r4h1	...	SQBescolari	SQBage	SQBhogar_total	S
0	ID_279628684	190000.0	0	3	0	1	1	0	NaN	0	...	100	1849	1	
1	ID_f29eb3ddd	135000.0	0	4	0	1	1	1	1.0	0	...	144	4489	1	
2	ID_68de51c94	NaN	0	8	0	1	1	0	NaN	0	...	121	8464	1	
3	ID_d671db89c	180000.0	0	5	0	1	1	1	1.0	0	...	81	289	16	
4	ID_d56d6f5f5	180000.0	0	5	0	1	1	1	1.0	0	...	121	1369	16	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
9552	ID_d45ae367d	80000.0	0	6	0	1	1	0	NaN	0	...	81	2116	25	
9553	ID_c94744e07	80000.0	0	6	0	1	1	0	NaN	0	...	0	4	25	
9554	ID_85fc658f8	80000.0	0	6	0	1	1	0	NaN	0	...	25	2500	25	
9555	ID_ced540c61	80000.0	0	6	0	1	1	0	NaN	0	...	121	676	25	
9556	ID_a38c64491	80000.0	0	6	0	1	1	0	NaN	0	...	64	441	25	

9557 rows × 143 columns

In [6]:

```
data.describe()
```

Out[6]:

	v2a1	hacdor	rooms	hacapo	v14a	refrig	v18q	v18q1	r4h1	
count	2.697000e+03	9557.000000	9557.000000	9557.000000	9557.000000	9557.000000	9557.000000	2215.000000	9557.000000	9557.00
mean	1.652316e+05	0.038087	4.955530	0.023648	0.994768	0.957623	0.231767	1.404063	0.385895	1.55
std	1.504571e+05	0.191417	1.468381	0.151957	0.072145	0.201459	0.421983	0.763131	0.680779	1.03
min	0.000000e+00	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000	1.000000	0.000000	0.00
25%	8.000000e+04	0.000000	4.000000	0.000000	1.000000	1.000000	0.000000	1.000000	0.000000	1.00
50%	1.300000e+05	0.000000	5.000000	0.000000	1.000000	1.000000	0.000000	1.000000	0.000000	1.00
75%	2.000000e+05	0.000000	6.000000	0.000000	1.000000	1.000000	0.000000	2.000000	1.000000	2.00
max	2.353477e+06	1.000000	11.000000	1.000000	1.000000	1.000000	1.000000	6.000000	5.000000	8.00

8 rows × 138 columns

In [7]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9557 entries, 0 to 9556
```

```
RangeIndex: 9557 entries, 0 to 9556
Columns: 143 entries, Id to Target
dtypes: float64(8), int64(130), object(5)
memory usage: 10.4+ MB
```

In [8]:

```
data.columns[1:100]
```

Out[8]:

```
Index(['v2a1', 'hacdor', 'rooms', 'hacapo', 'v14a', 'refrig', 'v18q', 'v18q1',
       'r4h1', 'r4h2', 'r4h3', 'r4m1', 'r4m2', 'r4m3', 'r4t1', 'r4t2', 'r4t3',
       'tamhog', 'tamviv', 'escolari', 'rez_esc', 'hhsize', 'paredblolad',
       'paredzocalo', 'paredpreb', 'pareddes', 'paredmad', 'paredzinc',
       'paredfibras', 'paredother', 'pisomoscer', 'pisocemento', 'pisoother',
       'pisonatur', 'pisonotiene', 'pisomadera', 'techozinc', 'techoentrepiso',
       'techocane', 'techootro', 'cielorazo', 'abastaguadentro',
       'abastaguafuera', 'abastaguano', 'public', 'planpri', 'noelec',
       'coopele', 'sanitario1', 'sanitario2', 'sanitario3', 'sanitario5',
       'sanitario6', 'energcocinar1', 'energcocinar2', 'energcocinar3',
       'energcocinar4', 'elimbasu1', 'elimbasu2', 'elimbasu3', 'elimbasu4',
       'elimbasu5', 'elimbasu6', 'epared1', 'epared2', 'epared3', 'etecho1',
       'etecho2', 'etecho3', 'eviv1', 'eviv2', 'eviv3', 'dis', 'male',
       'female', 'estadocivil1', 'estadocivil2', 'estadocivil3',
       'estadocivil4', 'estadocivil5', 'estadocivil6', 'estadocivil7',
       'parentesco1', 'parentesco2', 'parentesco3', 'parentesco4',
       'parentesco5', 'parentesco6', 'parentesco7', 'parentesco8',
       'parentesco9', 'parentesco10', 'parentesco11', 'parentesco12',
       'idhogar', 'hogar_nin', 'hogar_adul', 'hogar_mayor', 'hogar_total'],
      dtype='object')
```

In [9]:

```
data.columns[100:143]
```

Out[9]:

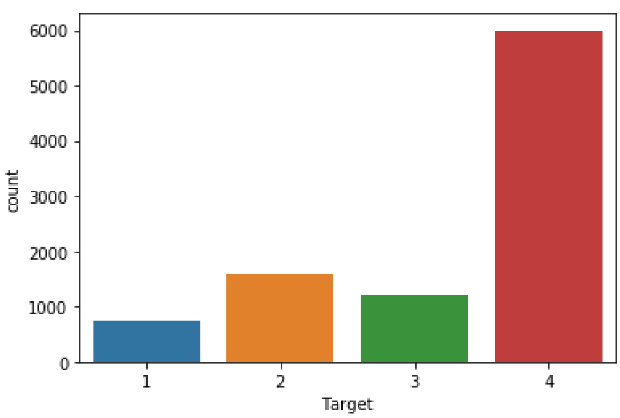
```
Index(['dependency', 'edjefe', 'edjefa', 'meaneduc', 'instlevel1',
       'instlevel2', 'instlevel3', 'instlevel4', 'instlevel5', 'instlevel6',
       'instlevel7', 'instlevel8', 'instlevel9', 'bedrooms', 'overcrowding',
       'tipovivi1', 'tipovivi2', 'tipovivi3', 'tipovivi4', 'tipovivi5',
       'computer', 'television', 'mobilephone', 'qmobilephone', 'lugar1',
       'lugar2', 'lugar3', 'lugar4', 'lugar5', 'lugar6', 'area1', 'area2',
       'age', 'SQBescolari', 'SQBage', 'SQBhogar_total', 'SQBedjefe',
       'SQBhogar_nin', 'SQBovercrowding', 'SQBdependency', 'SQBmeaned',
       'agesq', 'Target'],
      dtype='object')
```

In [10]:

```
sns.countplot(x='Target', data=data)
```

Out[10]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x24db45c0308>
```

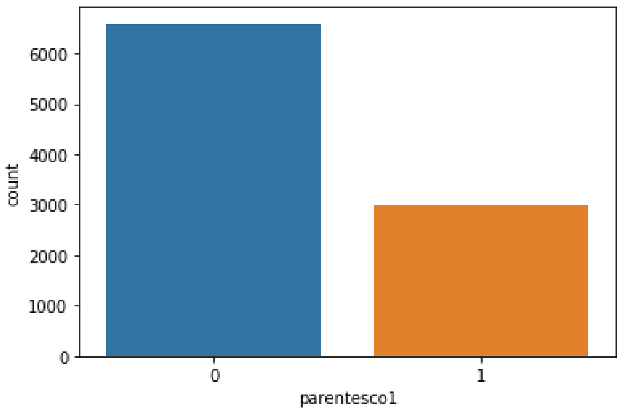


In [11]:

```python
sns.countplot(x='parentesco1', data=data)
```

Out[11]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x24db51e7348>
```



In [12]:

```python
data.isnull().sum()
```

Out[12]:

```
Id                      0
v2a1                 6860
hacdor                  0
rooms                   0
hacapo                  0
                     ...
SQBovercrowding         0
SQBdependency           0
SQBmeaned               5
agesq                   0
Target                  0
Length: 143, dtype: int64
```

In [13]:

```python
data=data.dropna(axis=1)
```

In [14]:

```python
data.columns
```

Out[14]:

```
Index(['Id', 'hacdor', 'rooms', 'hacapo', 'v14a', 'refrig', 'v18q', 'r4h1',
       'r4h2', 'r4h3',
       ...
       'age', 'SQBescolari', 'SQBage', 'SQBhogar_total', 'SQBedjefe',
       'SQBhogar_nin', 'SQBovercrowding', 'SQBdependency', 'agesq', 'Target'],
      dtype='object', length=138)
```

In [15]:

```python
data.corr()
```

Out[15]:

	hacdor	rooms	hacapo	v14a	refrig	v18q	r4h1	r4h2	r4h3	r4m1	...	age	SC
hacdor	1.000000	-0.233369	0.652594	-0.175011	-0.101965	-0.084680	0.232508	0.059313	0.184857	0.268978	...	-0.118168	

	hacdor	rooms	hacapo	v14a	refrig	v18q	r4h1	r4h2	r4h3	r4m1	...	age	SC
rooms	-0.233369	1.000000	-0.213368	0.129183	0.130531	0.254256	-0.066578	0.267627	0.195222	-0.032558	...	0.077046	
hacapo	0.652594	-0.213368	1.000000	-0.150986	-0.124506	-0.067529	0.226378	0.126645	0.240056	0.241452	...	-0.087773	
v14a	-0.175011	0.129183	-0.150986	1.000000	0.143143	0.036396	-0.054769	0.018133	-0.015552	-0.006370	...	0.027193	
refrig	-0.101965	0.130531	-0.124506	0.143143	1.000000	0.086002	-0.047087	-0.022819	-0.046860	-0.023502	...	0.029801	
...	...	...	...	...	...	...	...	...	...	...	...	...	
SQBhogar_nin	0.388043	0.007952	0.367025	-0.015193	-0.108718	-0.050562	0.565494	0.124701	0.432550	0.550488	...	-0.316034	
SQBovercrowding	0.794699	-0.355526	0.640096	-0.174969	-0.123054	-0.125936	0.355660	0.144478	0.329636	0.348197	...	-0.240636	
SQBdependency	0.005278	-0.027575	0.014411	0.005712	-0.034080	-0.071504	-0.036977	-0.157357	-0.158375	-0.012610	...	0.303847	
agesq	-0.102725	0.068288	-0.075528	0.023831	0.025846	-0.054670	-0.272690	-0.054712	-0.203856	-0.281166	...	0.958090	
Target	-0.191714	0.226208	-0.138008	0.063382	0.126792	0.238864	-0.229889	0.101253	-0.043359	-0.253163	...	0.117620	

133 rows × 133 columns

In [18]:

```
dat_1=data[data['parentesco1']==1]
dat_1
```

Out[18]:

	Id	hacdor	rooms	hacapo	v14a	refrig	v18q	r4h1	r4h2	r4h3	...	age	SQBescolari	SQBage	SQBhogar_total	S
0	ID_279628684	0	3	0	1	1	0	0	1	1	...	43	100	1849	1	
1	ID_f29eb3ddd	0	4	0	1	1	1	0	1	1	...	67	144	4489	1	
2	ID_68de51c94	0	8	0	1	1	0	0	0	0	...	92	121	8464	1	
5	ID_ec05b1a7b	0	5	0	1	1	1	0	2	2	...	38	121	1444	16	
8	ID_1284f8aad	1	2	0	1	1	0	0	1	1	...	30	81	900	16	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
9535	ID_18b0a845b	0	4	0	1	0	0	1	1	2	...	26	25	676	25	
9541	ID_a31274054	0	3	0	0	0	0	2	2	4	...	40	4	1600	25	
9545	ID_32a00a8bf	0	5	0	1	1	0	1	2	3	...	45	4	2025	25	
9551	ID_79d39dddc	0	3	0	1	1	0	0	1	1	...	67	0	4489	4	
9552	ID_d45ae367d	0	6	0	1	1	0	0	2	2	...	46	81	2116	25	

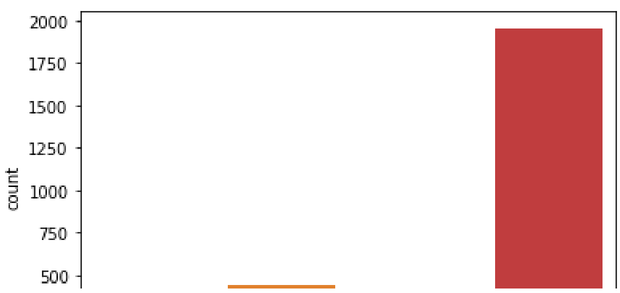
2973 rows × 138 columns

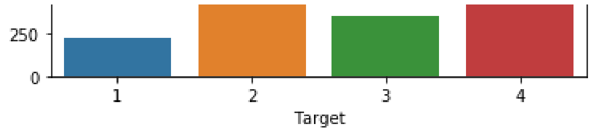
In [19]:

```
sns.countplot(x='Target', data=dat_1)
```

Out[19]:

<matplotlib.axes._subplots.AxesSubplot at 0x21444c6b808>





In [20]:

```
data=data.dropna(axis=1)
```

In [21]:

```
data.columns
```

Out[21]:

```
Index(['Id', 'hacdor', 'rooms', 'hacapo', 'v14a', 'refrig', 'v18q', 'r4h1',
       'r4h2', 'r4h3',
       ...
       'age', 'SQBescolari', 'SQBage', 'SQBhogar_total', 'SQBedjefe',
       'SQBhogar_nin', 'SQBovercrowding', 'SQBdependency', 'agesq', 'Target'],
      dtype='object', length=138)
```

Predict the accuracy using random forest classifier

In [19]:

```
#taining model
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, roc_auc_score, roc_curve,auc,classification_report
from sklearn.preprocessing import StandardScaler
y=data['Target']
x=data.corr().columns
x=data[x]
x=x.drop(['Target'],axis=1)
x_train,x_test,y_train,y_test=train_test_split(x,y)
scl=StandardScaler()
x_train = scl.fit_transform(x_train)
x_test  =scl.fit_transform(x_test)
rf = RandomForestClassifier(n_estimators=100, oob_score=True, random_state=123456)
rf.fit(x_train, y_train)
```

Out[19]:

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                       max_depth=None, max_features='auto', max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=100,
                       n_jobs=None, oob_score=True, random_state=123456,
                       verbose=0, warm_start=False)
```

In [26]:

```
predicted = rf.predict(x_test)
accuracy = accuracy_score(y_test, predicted)
print(f'Mean accuracy score: {accuracy:.3}')
print(classification_report(y_test,predicted))
```

```
Mean accuracy score: 0.881
              precision    recall  f1-score   support

           1       0.91      0.69      0.79       187
           2       0.93      0.74      0.82       409
           3       0.95      0.62      0.75       307
           4       0.86      1.00      0.92      1487

    accuracy                           0.88      2390
   macro avg       0.91      0.76      0.82      2390
```

```
weighted avg       0.89      0.88      0.87      2390
```

Check the accuracy using random forest with cross validation

In [28]:

```
from sklearn.model_selection import cross_val_score
k=cross_val_score(rf,x_train,y_train,cv=10,scoring='accuracy')
print(k)
print('the accuracy using random forest with cross validation',k.mean())
```

```
[0.86629526 0.87325905 0.87447699 0.89121339 0.87587169 0.87308229
 0.87726639 0.86610879 0.88391608 0.88935574]
the accuracy using random forest with cross validation 0.8770845669563702
```

In [ ]:

```

```

In [ ]:

```

```