In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [2]:

```
train=pd.read_csv('train.csv')
test=pd.read_csv('test.csv')
```

In [4]:

```
train.shape
```

Out[4]:

```
(9557, 143)
```

In [5]:

```
test.shape
```

Out[5]:

```
(23856, 142)
```

In [7]:

```
train.head(15)
```

Out[7]:

	Id	v2a1	hacdor	rooms	hacapo	v14a	refrig	v18q	v18q1	r4h1	...	SQBescolari	SQBage	SQBhogar_to
0	ID_279628684	190000.0	0	3	0	1	1	0	NaN	0	...	100	1849	
1	ID_f29eb3ddd	135000.0	0	4	0	1	1	1	1.0	0	...	144	4489	
2	ID_68de51c94	NaN	0	8	0	1	1	0	NaN	0	...	121	8464	
3	ID_d671db89c	180000.0	0	5	0	1	1	1	1.0	0	...	81	289	
4	ID_d56d6f5f5	180000.0	0	5	0	1	1	1	1.0	0	...	121	1369	
5	ID_ec05b1a7b	180000.0	0	5	0	1	1	1	1.0	0	...	121	1444	
6	ID_e9e0c1100	180000.0	0	5	0	1	1	1	1.0	0	...	4	64	
7	ID_3e04e571e	130000.0	1	2	0	1	1	0	NaN	0	...	0	49	
8	ID_1284f8aad	130000.0	1	2	0	1	1	0	NaN	0	...	81	900	
9	ID_51f52fdd2	130000.0	1	2	0	1	1	0	NaN	0	...	121	784	
10	ID_db44f5c59	130000.0	1	2	0	1	1	0	NaN	0	...	9	121	
11	ID_de822510c	100000.0	0	3	0	1	1	0	NaN	0	...	144	324	
12	ID_d94071d7c	100000.0	0	3	0	1	1	0	NaN	0	...	121	1156	
13	ID_064b57869	NaN	0	4	0	1	1	1	1.0	0	...	16	6241	
14	ID_5c837d8a4	NaN	0	4	0	1	1	1	1.0	0	...	225	1521	

15 rows × 143 columns

In [8]:

```
test.head(2)
```

Out[8]:

Out[8]:

	Id	v2a1	hacdor	rooms	hacapo	v14a	refrig	v18q	v18q1	r4h1	...	age	SQBescolari	SQBage	SQBhogar_to
0	ID_2f6873615	NaN	0	5	0	1	1	0	NaN	1	...	4	0	16	
1	ID_1c78846d2	NaN	0	5	0	1	1	0	NaN	1	...	41	256	1681	

2 rows × 142 columns

In [9]:

```
# The Output Variable is Target
print("The output variable is: Target")
```

```
The output variable is: Target
```

In [10]:

```
# Understanding the type of data we are dealing with.
```

In [11]:

```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
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 [13]:

```
# categorial data
train.select_dtypes('object').head()
```

Out[13]:

	Id	idhogar	dependency	edjefe	edjefa
0	ID_279628684	21eb7fcc1	no	10	no
1	ID_f29eb3ddd	0e5d7a658	8	12	no
2	ID_68de51c94	2c7317ea8	8	no	11
3	ID_d671db89c	2b58d945f	yes	11	no
4	ID_d56d6f5f5	2b58d945f	yes	11	no

In [14]:

```
train["idhogar"].value_counts()
```

Out[14]:

```
fd8a6d014    13
0c7436de6    12
ae6cf0558    12
3fe29a56b    11
6b35cdcf0    11
             ..
ee3d80cb6     1
3be430e4c     1
00e443b00     1
0aed192d8     1
8230d4e9c     1
Name: idhogar, Length: 2988, dtype: int64
```

In [16]:

```
train["dependency"].value_counts()
```

Out[16]:

```
yes          2192
no           1747
.5           1497
2             730
1.5           713
.33333334     598
.66666669     487
8             378
.25           260
3             236
4             100
.75            98
.2             90
.40000001      84
1.3333334      84
2.5            77
5              24
3.5            18
.80000001      18
1.25           18
2.25           13
.71428573      12
1.75           11
.22222222      11
.83333331      11
1.2            11
.2857143        9
1.6666666       8
.60000002       8
6               7
.16666667       7
Name: dependency, dtype: int64
```

In [18]:

```
train['edjefe'].value_counts()
```

Out[18]:

```
no      3762
6       1845
11       751
9        486
3        307
15       285
8        257
7        234
5        222
14       208
17       202
2        194
4        137
16       134
yes      123
12       113
10       111
13       103
21        43
18        19
19        14
20         7
Name: edjefe, dtype: int64
```

In [19]:

```
train['edjefa'].value_counts()
```

Out[19]:

```
no      6230
6        947
```

```
11       399
9        237
8        217
15       188
7        179
5        176
3        152
4        136
14       120
16       113
10        96
2         84
17        76
12        72
yes       69
13        52
21         5
19         4
18         3
20         2
Name: edjefa, dtype: int64
```

In [22]:

```
print("the ID and idhogar are identifiers hence we drop them as they will not contribute to the model")
print("dependency, edjefe, edjefa are mixed of categorial data and numerical data")
print("Hence we convert the categorial data to numerical data")
```

```
the ID and idhogar are identifiers hence we drop them as they will not contribute to the
model
dependency, edjefe, edjefa are mixed of categorial data and numerical data
Hence we convert the categorial data to numerical data
```

In [25]:

```
# convert to numeric as c_t_n
def c_t_n(i):
    if i == "yes":
        return(float(1))
    elif i == "no":
        return(float(0))
    else:
        return(float(i))
```

In [26]:

```
train['dependency']=train['dependency'].apply(c_t_n)
train['edjefe']=train['edjefe'].apply(c_t_n)
train['edjefa']=train['edjefa'].apply(c_t_n)
```

In [27]:

```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9557 entries, 0 to 9556
Columns: 143 entries, Id to Target
dtypes: float64(11), int64(130), object(2)
memory usage: 10.4+ MB
```

In [28]:

```
train.select_dtypes('object').head()
```

Out[28]:

	Id	idhogar
0	ID_279628684	21eb7fcc1

	Id	idhogar
1	ID_f29eb3ddd	0e5d7a658
~~2~~	~~ID_68de51c94~~	~~2c7317ea8~~
3	ID_d671db89c	2b58d945f
4	ID_d56d6f5f5	2b58d945f

In [29]:

```
print("The two categorial data are identifiers or IDs, we drop them as they wont contribu
te to the model")
```

```
The two categorial data are identifiers or IDs, we drop them as they wont contribute to t
he model
```

In [30]:

```
# Now checking if theres is any biases in the data set
print("we do this by using statistical tools")
```

```
we do this by using statistical tools
```

In [34]:

```
import scipy.stats as stats
import statsmodels.api as sm
from statsmodels.formula.api import ols

from scipy.stats import chi2_contingency
from scipy.stats import chi2
```

In [35]:

```
print("There is always a bias in a dataset")
print("Machine learning is a representation of probability density functions correlated t
o each other")
print("We can use statistical tools to compare and see how other variables are correlated
")
print("We can achieve this by using hypothesis testing")
print("Null Hypothesis: There is a relation between the variables")
print("Alternative Hypothesis : There is no relationship between the variables")
```

```
There is always a bias in a dataset
Machine learning is a representation of probability density functions correlated to each
other
We can use statistical tools to compare and see how other variables are correlated
We can achieve this by using hypothesis testing
Null Hypothesis: There is a relation between the variables
Alternative Hypothesis : There is no relationship between the variables
```

In [36]:

```
data_cont_table = pd.crosstab(train["bedrooms"] , train["overcrowding"])
table = data_cont_table
stat, p, dof, expected = chi2_contingency(table)
print("dof=%d" % dof)
print(expected)
probability = 0.99
critical = chi2.ppf(probability, dof)
print("probability=%.3f, critical=%.3f, stat=%.3f" % (probability, critical, stat))
if abs(stat) >= critical:
    print("Reject the Null Hypothesis")
    print("There is a relation between the variables")
else:
    print("Fail to reject The Null Hypothesis")
    print("There is no relationship between the variables")
alpha = 1.0 - probability
print('significance=%.3f, p=%.3f' % (alpha, p))
if p <= alpha:
    print("Reject the Null Hypothesis")
    print("There is a relation between the variables")
else:
```

```python
    print("Fail to reject The Null Hypothesis")
    print("There is no relationship between the variables")
```

```
dof=222
[[6.29904782e-02 9.44857173e-01 3.84241917e+00 9.44857173e-01
  1.56846291e+01 1.63775243e+00 2.46292770e+01 9.13361934e+00
  3.02354295e+00 1.00784765e+00 1.28059642e+02 1.25980956e-01
  2.26765721e+00 1.98420006e+01 7.53366119e+01 3.77942869e+00
  7.30059642e+01 8.81866695e-01 4.95735063e+01 7.87380977e+00
  5.66914304e-01 1.13382861e+00 8.82496599e+01 6.92895260e-01
  1.70074291e+00 9.76352412e+00 2.57001151e+01 1.76373339e+00
  2.82197342e+01 1.70074291e+00 2.64560008e+00 6.92895260e-01
  7.18091451e+00 1.88971435e+00 5.66914304e-01 4.66129539e+00
  9.44857173e-01 2.26765721e+00]
 [3.53981375e-01 5.30972062e+00 2.15928639e+01 5.30972062e+00
  8.81413624e+01 9.20351575e+00 1.38406718e+02 5.13272994e+01
  1.69911060e+01 5.66370200e+00 7.19644135e+02 7.07962750e-01
  1.27433295e+01 1.11504133e+02 4.23361724e+02 2.12388825e+01
  4.10264414e+02 4.95573925e+00 2.78583342e+02 4.42476719e+01
  3.18583237e+00 6.37166475e+00 4.95927906e+02 3.89379512e+00
  9.55749712e+00 5.48671131e+01 1.44424401e+02 9.91147850e+00
  1.58583656e+02 9.55749712e+00 1.48672177e+01 3.89379512e+00
  4.03538767e+01 1.06194412e+01 3.18583237e+00 2.61946217e+01
  5.30972062e+00 1.27433295e+01]
 [4.16971853e-01 6.25457780e+00 2.54352830e+01 6.25457780e+00
  1.03825991e+02 1.08412682e+01 1.63035995e+02 6.04609187e+01
  2.00146489e+01 6.67154965e+00 8.47703777e+02 8.33943706e-01
  1.50109867e+01 1.31346134e+02 4.98698336e+02 2.50183112e+01
  4.83270378e+02 5.83760594e+00 3.28156848e+02 5.21214816e+01
  3.75274668e+00 7.50549336e+00 5.84177566e+02 4.58669038e+00
  1.12582400e+01 6.46306372e+01 1.70124516e+02 1.16752119e+01
  1.86803390e+02 1.12582400e+01 1.75128178e+01 4.58669038e+00
  4.75347913e+01 1.25091556e+01 3.75274668e+00 3.08559171e+01
  6.25457780e+00 1.50109867e+01]
 [1.23888249e-01 1.85832374e+00 7.55718322e+00 1.85832374e+00
  3.08481741e+01 3.22109449e+00 4.84403055e+01 1.79637962e+01
  5.94663597e+00 1.98221199e+00 2.51864811e+02 2.47776499e-01
  4.45997698e+00 3.90247986e+01 1.48170346e+02 7.43329497e+00
  1.43586481e+02 1.73443549e+00 9.75000523e+01 1.54860312e+01
  1.11499425e+00 2.22998849e+00 1.73567437e+02 1.36277074e+00
  3.34498274e+00 1.92026787e+01 5.05464058e+01 3.46887098e+00
  5.55019358e+01 3.34498274e+00 5.20330648e+00 1.36277074e+00
  1.41232604e+01 3.71664748e+00 1.11499425e+00 9.16773046e+00
  1.85832374e+00 4.45997698e+00]
 [3.12859684e-02 4.69289526e-01 1.90844407e+00 4.69289526e-01
  7.79020613e+00 8.13435178e-01 1.22328136e+01 4.53646542e+00
  1.50172648e+00 5.00575494e-01 6.36043738e+01 6.25719368e-02
  1.12629486e+00 9.85508005e+00 3.74180182e+01 1.87715810e+00
  3.62604374e+01 4.38003558e-01 2.46220571e+01 3.91074605e+00
  2.81573716e-01 5.63147431e-01 4.38316417e+01 3.44145652e-01
  8.44721147e-01 4.84932510e+00 1.27646751e+01 8.76007115e-01
  1.40161138e+01 8.44721147e-01 1.31401067e+00 3.44145652e-01
  3.56660040e+00 9.38579052e-01 2.81573716e-01 2.31516166e+00
  4.69289526e-01 1.12629486e+00]
 [1.04635346e-02 1.56953019e-01 6.38275610e-01 1.56953019e-01
  2.60542011e+00 2.72051899e-01 4.09124202e+00 1.51721251e+00
  5.02249660e-01 1.67416553e-01 2.12723658e+01 2.09270692e-02
  3.76687245e-01 3.29601339e+00 1.25143874e+01 6.27812075e-01
  1.21272366e+01 1.46489484e-01 8.23480172e+00 1.30794182e+00
  9.41718112e-02 1.88343622e-01 1.46594119e+01 1.15098880e-01
  2.82515434e-01 1.62184786e+00 4.26912211e+00 2.92978968e-01
  4.68766349e+00 2.82515434e-01 4.39468452e-01 1.15098880e-01
  1.19284294e+00 3.13906037e-01 9.41718112e-02 7.74301559e-01
  1.56953019e-01 3.76687245e-01]
 [4.18541383e-04 6.27812075e-03 2.55310244e-02 6.27812075e-03
  1.04216804e-01 1.08820760e-02 1.63649681e-01 6.06885006e-02
  2.00899864e-02 6.69666213e-03 8.50894632e-01 8.37082767e-04
  1.50674898e-02 1.31840536e-01 5.00575494e-01 2.51124830e-02
  4.85089463e-01 5.85957937e-03 3.29392069e-01 5.23176729e-02
  3.76687245e-03 7.53374490e-03 5.86376478e-01 4.60395522e-03
  1.13006173e-02 6.48739144e-02 1.70764884e-01 1.17191587e-02
  1.87506540e-01 1.13006173e-02 1.75787381e-02 4.60395522e-03
```

```
  4.77137177e-02 1.25562415e-02 3.76687245e-03 3.09720624e-02
  6.27812075e-03 1.50674898e-02]]
probability=0.990, critical=273.939, stat=24077.038
Reject the Null Hypothesis
There is a relation between the variables
significance=0.010, p=0.000
Reject the Null Hypothesis
There is a relation between the variables
```

In [37]:

```
data_cont_table = pd.crosstab(train["r4h1"] , train["r4h3"])
table = data_cont_table
stat, p, dof, expected = chi2_contingency(table)
print("dof=%d" % dof)
print(expected)
probability = 0.99
critical = chi2.ppf(probability, dof)
print("probability=%.3f, critical=%.3f, stat=%.3f" % (probability, critical, stat))
if abs(stat) >= critical:
    print("Reject the Null Hypothesis")
    print("There is a relation between the variables")
else:
    print("Fail to reject The Null Hypothesis")
    print("There is no relationship between the variables")
alpha = 1.0 - probability
print('significance=%.3f, p=%.3f' % (alpha, p))
if p <= alpha:
    print("Reject the Null Hypothesis")
    print("There is a relation between the variables")
else:
    print("Fail to reject The Null Hypothesis")
    print("There is no relationship between the variables")
```

```
dof=40
[[4.41446479e+02 2.20863828e+03 2.25714115e+03 1.23717485e+03
  3.56390708e+02 1.27232186e+02 6.60763838e+01 1.68705661e+01
  7.02940253e+00]
 [1.40687245e+02 7.03884273e+02 7.19341948e+02 3.94282725e+02
  1.13580308e+02 4.05483938e+01 2.10582819e+01 5.37658261e+00
  2.24024275e+00]
 [3.98865753e+01 1.99559904e+02 2.03942346e+02 1.11784033e+02
  3.22014230e+01 1.14959715e+01 5.97028356e+00 1.52432772e+00
  6.35136549e-01]
 [3.48268285e+00 1.74245056e+01 1.78071571e+01 9.76038506e+00
  2.81165638e+00 1.00376687e+00 5.21293293e-01 1.33096160e-01
  5.54567333e-02]
 [1.05137595e+00 5.26022811e+00 5.37574553e+00 2.94653134e+00
  8.48801925e-01 3.03023961e-01 1.57371560e-01 4.01799728e-02
  1.67416553e-02]
 [1.44564194e+00 7.23281364e+00 7.39165010e+00 4.05148059e+00
  1.16710265e+00 4.16657947e-01 2.16385895e-01 5.52474626e-02
  2.30197761e-02]]
probability=0.990, critical=63.691, stat=6859.280
Reject the Null Hypothesis
There is a relation between the variables
significance=0.010, p=0.000
Reject the Null Hypothesis
There is a relation between the variables
```

In [38]:

```
data_cont_table = pd.crosstab(train["r4h1"] , train["bedrooms"])
table = data_cont_table
stat, p, dof, expected = chi2_contingency(table)
print("dof=%d" % dof)
print(expected)
probability = 0.99
critical = chi2.ppf(probability, dof)
print("probability=%.3f, critical=%.3f, stat=%.3f" % (probability, critical, stat))
if abs(stat) >= critical:
    print("Reject the Null Hypothesis")
```

```
    print("There is a relation between the variables")
else:
    print("Fail to reject The Null Hypothesis")
    print("There is no relationship between the variables")
alpha = 1.0 - probability
print('significance=%.3f, p=%.3f' % (alpha, p))
if p <= alpha:
    print("Reject the Null Hypothesis")
    print("There is a relation between the variables")
else:
    print("Fail to reject The Null Hypothesis")
    print("There is no relationship between the variables")
```

```
dof=30
[[4.23170032e+02 2.37804688e+03 2.80121691e+03 8.32281260e+02
  2.10179136e+02 7.02940253e+01 2.81176101e+00]
 [1.34862614e+02 7.57874124e+02 8.92736737e+02 2.65244742e+02
  6.69832583e+01 2.24024275e+01 8.96097102e-01]
 [3.82352203e+01 2.14866695e+02 2.53101915e+02 7.52001674e+01
  1.89905828e+01 6.35136549e+00 2.54054620e-01]
 [3.33849534e+00 1.87610129e+01 2.20995082e+01 6.56607722e+00
  1.65815633e+00 5.54567333e-01 2.21826933e-02]
 [1.00784765e+00 5.66370200e+00 6.67154965e+00 1.98221199e+00
  5.00575494e-01 1.67416553e-01 6.69666213e-03]
 [1.38579052e+00 7.78759025e+00 9.17338077e+00 2.72554149e+00
  6.88291305e-01 2.30197761e-01 9.20791043e-03]]
probability=0.990, critical=50.892, stat=170.534
Reject the Null Hypothesis
There is a relation between the variables
significance=0.010, p=0.000
Reject the Null Hypothesis
There is a relation between the variables
```

In [39]:

```
data_cont_table = pd.crosstab(train["v18q"] , train["v18q1"])
table = data_cont_table
stat, p, dof, expected = chi2_contingency(table)
print("dof=%d" % dof)
print(expected)
probability = 0.99
critical = chi2.ppf(probability, dof)
print("probability=%.3f, critical=%.3f, stat=%.3f" % (probability, critical, stat))
if abs(stat) >= critical:
    print("Reject the Null Hypothesis")
    print("There is a relation between the variables")
else:
    print("Fail to reject The Null Hypothesis")
    print("There is no relationship between the variables")
alpha = 1.0 - probability
print('significance=%.3f, p=%.3f' % (alpha, p))
if p <= alpha:
    print("Reject the Null Hypothesis")
    print("There is a relation between the variables")
else:
    print("Fail to reject The Null Hypothesis")
    print("There is no relationship between the variables")
```

```
dof=0
[[1586.  444.  129.   37.   13.    6.]]
probability=0.990, critical=nan, stat=0.000
Fail to reject The Null Hypothesis
There is no relationship between the variables
significance=0.010, p=1.000
Fail to reject The Null Hypothesis
There is no relationship between the variables
```

In [40]:

```
print("There is a bias in the data set.")
```

```
There is a bias in the data set.
```

In [41]:

```
# CHECKING IF ALL MEMBERS OF THE HOUSE HOLD HAVE THE SAME POVERTY LEVEL.
```

In [43]:

```
unique_values = train.groupby('idhogar')['Target'].apply(lambda x: x.nunique() == 1)
different_households = unique_values[unique_values != True]
print('There are {} households where members of the house dont have the same poverty leve
l.'.format(len(different_households)))
```

```
There are 85 households where members of the house dont have the same poverty level.
```

In [45]:

```
# CHECK IF THERE IS A HOUSE WITHOUT A FAMILY HEAD.
family_head = train.groupby('idhogar')['parentesco1'].sum()
no_head = train.loc[train['idhogar'].isin(family_head[family_head == 0].index), :]
print('There are {} households without a family head.'.format(no_head['idhogar'].nunique
()))
```

```
There are 15 households without a family head.
```

In [46]:

```
# Set poverty level of the members and the head of the house within a family.
```

In [47]:

```
for individuals_household in different_households.index:
    true_target = int(train[(train["idhogar"] == individuals_household) & (train["parent
esco1"] == 1.0)]["Target"])
    train.loc[train["idhogar"] == individuals_household, "Target"] = true_target
unique_values = train.groupby("idhogar")["Target"].apply(lambda x: x.nunique() == 1)
different_households = unique_values[unique_values != True]
print("There are {} households where the family members do not all have the same target."
.format(len(different_households)))
```

```
There are 0 households where the family members do not all have the same target.
```

In [78]:

```
poverty_level = train[train["v2a1"] != 0]
```

In [79]:

```
poverty_level.shape
```

Out[79]:

```
(9528, 143)
```

In [80]:

```
poverty_level=poverty_level.groupby("area2")["v2a1"].apply(np.median)
```

In [81]:

```
poverty_level
```

Out[81]:

```
area2
0    NaN
1    NaN
Name: v2a1, dtype: float64
```

In [82]:

```
print("there are Null values in the v2a1 column")
```

```
print("v2a1 translates - Monthly rent payment")
print("this means that other family own the houses and dont pay rent")
print("we can replace Null values by 0 rent payment")
train['v2a1'].fillna(0,inplace=True)
```

```
there are Null values in the v2a1 column
v2a1 translates - Monthly rent payment
this means that other family own the houses and dont pay rent
we can replace Null values by 0 rent payment
```

In [83]:

```
# Count how many null values are existing in columns.
train.isna().sum().value_counts()
```

Out[83]:

```
0       139
5         2
7928      1
7342      1
dtype: int64
```

In [94]:

```
Poverty_level = train[train["v2a1"] != 0]
```

In [95]:

```
Poverty_level.shape
```

Out[95]:

```
(2668, 143)
```

In [96]:

```
poverty_level=Poverty_level.groupby("area2")["v2a1"].apply(np.median)
print(poverty_level)
```

```
area2
0    140000.0
1     80000.0
Name: v2a1, dtype: float64
```

In [97]:

```
print("area2 - we see the median of the rent of 140000 then people are below this value i
n urban area are Below POVERTY LEVEL")
print("area2 - we see the median of rula area of povety line at 80000")
```

```
area2 - we see the median of the rent of 140000 then people are below this value in urban
area are Below POVERTY LEVEL
area2 - we see the median of rula area of povety line at 80000
```

In [98]:

```
# we can define a function to filter give poverty levels depending on the rend median val
ues
def povertyID(x):
    if x < 80000:
        return("BELOW POVERTY LEVEL")
    elif x > 140000:
        return("ABOVE POVERTY LEVEL")
    else:
        return("BELOW POVERTY LEVEL OF URBAN AREA, BUT ABOVE POVERTY LEVEL OF A RURAL ARE
A")
```

In [100]:

```
P_L= Poverty_level["v2a1"].apply(povertyID)
```

In [101]:

```
P_L.shape
```

Out[101]:

```
(2668,)
```

In [102]:

```
pd.crosstab(P_L,Poverty_level["area2"])
```

Out[102]:

area2	0	1
v2a1		
ABOVE POVERTY LEVEL	1103	139
BELOW POVERTY LEVEL	418	208
BELOW POVERTY LEVEL OF URBAN AREA, BUT ABOVE POVERTY LEVEL OF A RURAL AREA	702	98

In [103]:

```
# Remove null value rows of the target variable
train["Target"].isna().sum()
```

Out[103]:

```
0
```

In [104]:

```
print("There is no need to remove null values as they dont exist in this target")
```

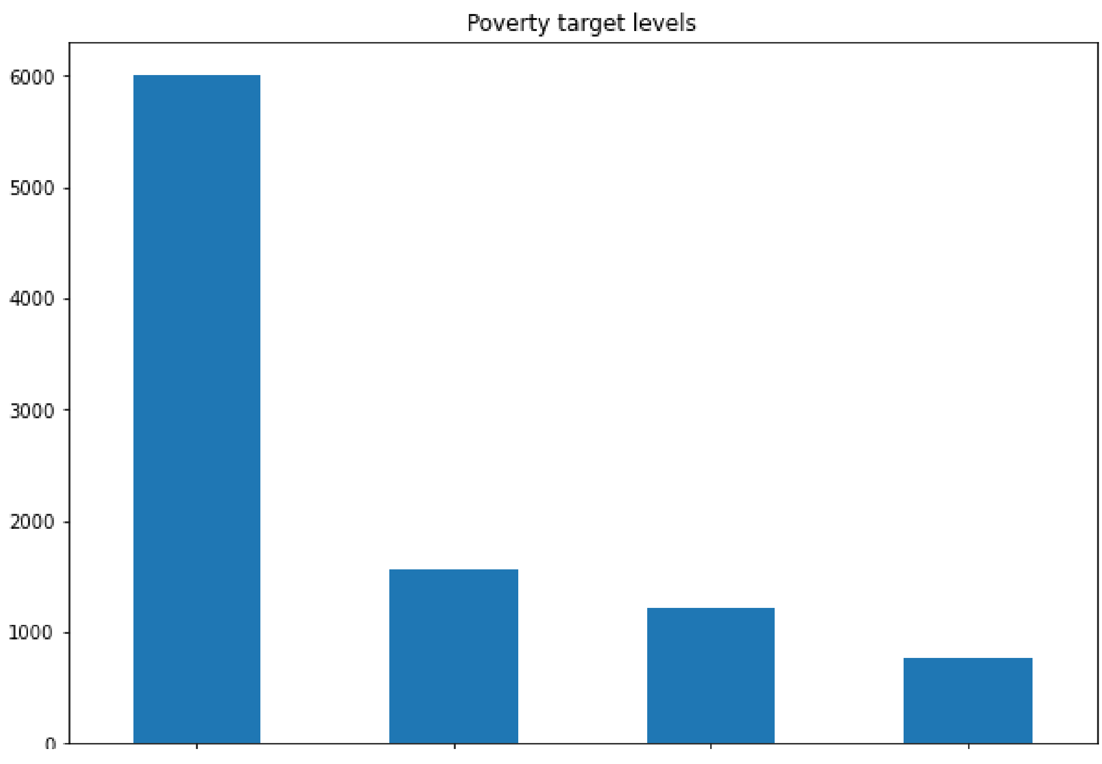
```
There is no need to remove null values as they dont exist in this target
```

In [113]:

```
#Visualising the "Target" column
(train["Target"].value_counts()).head().plot(kind="bar",figsize=(10,7), title = "Poverty target levels")
```

Out[113]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1e6664d67c0>
```



In [114]:

```
train.head(2)
```

Out[114]:

	Id	v2a1	hacdor	rooms	hacapo	v14a	refrig	v18q	v18q1	r4h1	...	SQBescolari	SQBage	SQBhogar_tota
0	ID_279628684	190000.0	0	3	0	1	1	0	NaN	0	...	100	1849	
1	ID_f29eb3ddd	135000.0	0	4	0	1	1	1	1.0	0	...	144	4489	

2 rows × 143 columns

In [115]:

```
train.drop(["Id","idhogar"],axis=1,inplace=True)
```

In [118]:

```
X=train.drop("Target",axis=1)
y=train.Target
```

In [119]:

```
# Predict the accuracy using random forest classifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1)
```

In [127]:

```
from sklearn.impute import SimpleImputer
imp = SimpleImputer(missing_values=np.nan,strategy="most_frequent")
X_train = imp.fit_transform(X_train)
X_test = imp.fit_transform(X_test)
```

In [128]:

```
clf = RandomForestClassifier()
```

In [141]:

```
from sklearn.metrics import accuracy_score
model = clf.fit(X_train,y_train)
y_predict = model.predict(X_test)
accuracy = accuracy_score(y_predict,y_test)
print(accuracy)
```

```
0.9382845188284519
```

In [142]:

```
# we now clean and fit the test data
#train.select_dtypes('object').head()
test["dependency"]=test["dependency"].apply(c_t_n)
test["edjefe"]=test["edjefe"].apply(c_t_n)
test["edjefa"]=test["edjefa"].apply(c_t_n)
test["v2a1"].fillna(0,inplace=True)
```

In [143]:

```
test_data = imp.fit_transform(test)
test_prediction=model.predict(test_data)
print(test_prediction)
```

```
[4 4 4 ... 4 4 4]
```

In [144]:

```
print("END")
```

END

In [ ]:

```

```