In [1]:
```
from google.colab import drive
drive.mount('/content/MyDrive/')
```

```
Drive already mounted at /content/MyDrive/; to attempt to forcibly remou
nt, call drive.mount("/content/MyDrive/", force_remount=True).
```

In [2]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
%matplotlib inline
pd.set_option('display.max_rows',None)
pd.set_option('display.max_columns',None)
```

In [3]:
```
train = pd.read_csv('/content/MyDrive/MyDrive/Datasets/ML Task Datas/Task
2 Data/train.csv')
```

In [4]:
```
test = pd.read_csv('/content/MyDrive/MyDrive/Datasets/ML Task Datas/Task
2 Data/test.csv')
```

In [5]:
```
train.shape
```

Out[5]:
```
(9557, 143)
```

In [6]:
```
test.shape
```

Out[6]:
```
(23856, 142)
```

In [7]:
```
train.dtypes.value_counts()
```

Out[7]:
```
int64      130
float64      8
object       5
dtype: int64
```

In [8]:
```
test.dtypes.value_counts()
```

Out[8]:
```
int64      129
float64      8
object       5
dtype: int64
```

In [9]:
```
train.isnull().any().sum()
```

Out[9]:
```
5
```

In [10]:
```
test.isnull().any().sum()
```

Out[10]:
```
5
```

In [11]:
```
train.head()
```

Out[11]:

	Id	v2a1	hacdor	rooms	hacapo	v14a	refrig	v18q	v18q1	r4h1	r4h2
0	ID_279628684	190000.0	0	3	0	1	1	0	NaN	0	1
1	ID_f29eb3ddd	135000.0	0	4	0	1	1	1	1.0	0	1
2	ID_68de51c94	NaN	0	8	0	1	1	0	NaN	0	0
3	ID_d671db89c	180000.0	0	5	0	1	1	1	1.0	0	2
4	ID_d56d6f5f5	180000.0	0	5	0	1	1	1	1.0	0	2

In [12]:
```
test.head()
```

Out[12]:

	Id	v2a1	hacdor	rooms	hacapo	v14a	refrig	v18q	v18q1	r4h1	r4h2
0	ID_2f6873615	NaN	0	5	0	1	1	0	NaN	1	1
1	ID_1c78846d2	NaN	0	5	0	1	1	0	NaN	1	1
2	ID_e5442cf6a	NaN	0	5	0	1	1	0	NaN	1	1
3	ID_a8db26a79	NaN	0	14	0	1	1	1	1.0	0	1
4	ID_a62966799	175000.0	0	4	0	1	1	1	1.0	0	0

In [13]:
```
for column in train:
  if train[column].dtype == 'object':
    print(column,end=',')
```

```
Id,idhogar,dependency,edjefe,edjefa,
```

In [14]:
```
for column in train:
  if train[column].dtype == 'float64':
    print(column,end=',')
```

```
v2a1,v18q1,rez_esc,meaneduc,overcrowding,SQBovercrowding,SQBdependency,S
QBmeaned,
```

In [15]:
```
for column in train:
  if train[column].dtype == 'int64':
    print(column,end=',')
```

```
hacdor,rooms,hacapo,v14a,refrig,v18q,r4h1,r4h2,r4h3,r4m1,r4m2,r4m3,r4t1,
r4t2,r4t3,tamhog,tamviv,escolari,hhsize,paredblolad,paredzocalo,paredpre
b,pareddes,paredmad,paredzinc,paredfibras,paredother,pisomoscer,pisoceme
nto,pisoother,pisonatur,pisonotiene,pisomadera,techozinc,techoentrepiso,
techocane,techootro,cielorazo,abastaguadentro,abastaguafuera,abastaguan
o,public,planpri,noelec,coopele,sanitario1,sanitario2,sanitario3,sanitar
io5,sanitario6,energcocinar1,energcocinar2,energcocinar3,energcocinar4,e
limbasu1,elimbasu2,elimbasu3,elimbasu4,elimbasu5,elimbasu6,epared1,epare
d2,epared3,etecho1,etecho2,etecho3,eviv1,eviv2,eviv3,dis,male,female,est
adocivil1,estadocivil2,estadocivil3,estadocivil4,estadocivil5,estadocivi
l6,estadocivil7,parentesco1,parentesco2,parentesco3,parentesco4,parentes
co5,parentesco6,parentesco7,parentesco8,parentesco9,parentesco10,parente
sco11,parentesco12,hogar_nin,hogar_adul,hogar_mayor,hogar_total,instleve
l1,instlevel2,instlevel3,instlevel4,instlevel5,instlevel6,instlevel7,ins
tlevel8,instlevel9,bedrooms,tipovivi1,tipovivi2,tipovivi3,tipovivi4,tipo
vivi5,computer,television,mobilephone,qmobilephone,lugar1,lugar2,lugar3,
lugar4,lugar5,lugar6,area1,area2,age,SQBescolari,SQBage,SQBhogar_total,S
QBedjefe,SQBhogar_nin,agesq,Target,
```

In [16]:
```
for column in test:
  if test[column].dtype == 'object':
    print(column,end=',')
```

```
Id,idhogar,dependency,edjefe,edjefa,
```

In [17]:
```
for column in test:
  if test[column].dtype == 'float64':
    print(column,end=',')
```

```
v2a1,v18q1,rez_esc,meaneduc,overcrowding,SQBovercrowding,SQBdependency,S
QBmeaned,
```

In [18]:
```
for column in test:
  if test[column].dtype == 'int64':
    print(column,end=',')
```

```
hacdor,rooms,hacapo,v14a,refrig,v18q,r4h1,r4h2,r4h3,r4m1,r4m2,r4m3,r4t1,
r4t2,r4t3,tamhog,tamviv,escolari,hhsize,paredblolad,paredzocalo,paredpre
b,pareddes,paredmad,paredzinc,paredfibras,paredother,pisomoscer,pisoceme
nto,pisoother,pisonatur,pisonotiene,pisomadera,techozinc,techoentrepiso,
techocane,techootro,cielorazo,abastaguadentro,abastaguafuera,abastaguan
o,public,planpri,noelec,coopele,sanitario1,sanitario2,sanitario3,sanitar
io5,sanitario6,energcocinar1,energcocinar2,energcocinar3,energcocinar4,e
limbasu1,elimbasu2,elimbasu3,elimbasu4,elimbasu5,elimbasu6,epared1,epare
d2,epared3,etecho1,etecho2,etecho3,eviv1,eviv2,eviv3,dis,male,female,est
adocivil1,estadocivil2,estadocivil3,estadocivil4,estadocivil5,estadocivi
l6,estadocivil7,parentesco1,parentesco2,parentesco3,parentesco4,parentes
co5,parentesco6,parentesco7,parentesco8,parentesco9,parentesco10,parente
sco11,parentesco12,hogar_nin,hogar_adul,hogar_mayor,hogar_total,instleve
l1,instlevel2,instlevel3,instlevel4,instlevel5,instlevel6,instlevel7,ins
tlevel8,instlevel9,bedrooms,tipovivi1,tipovivi2,tipovivi3,tipovivi4,tipo
vivi5,computer,television,mobilephone,qmobilephone,lugar1,lugar2,lugar3,
lugar4,lugar5,lugar6,area1,area2,age,SQBescolari,SQBage,SQBhogar_total,S
QBedjefe,SQBhogar_nin,agesq,
```

In [19]:
```
for column in train:
  if train[column].dtype == 'object':
    print(f'{column}:  {train[column].unique()}')
```

```
Id:  ['ID_279628684' 'ID_f29eb3ddd' 'ID_68de51c94' ... 'ID_85fc658f8'
 'ID_ced540c61' 'ID_a38c64491']
idhogar:  ['21eb7fcc1' '0e5d7a658' '2c7317ea8' ... 'a8eeafc29' '212db6f6
c'
 'd6c086aa3']
dependency:  ['no' '8' 'yes' '3' '.5' '.25' '2' '.66666669' '.33333334'
'1.5'
 '.40000001' '.75' '1.25' '.2' '2.5' '1.2' '4' '1.3333334' '2.25'
 '.22222222' '5' '.83333331' '.80000001' '6' '3.5' '1.6666666' '.2857143
'
 '1.75' '.71428573' '.16666667' '.60000002']
edjefe:  ['10' '12' 'no' '11' '9' '15' '4' '6' '8' '17' '7' '16' '14' '5
' '21' '2'
 '19' 'yes' '3' '18' '13' '20']
edjefa:  ['no' '11' '4' '10' '9' '15' '7' '14' '13' '8' '17' '6' '5' '3'
'16' '19'
 'yes' '21' '12' '2' '20' '18']
```

In [20]:
```
for column in test:
  if test[column].dtype == 'object':
    print(f'{column}:  {test[column].unique()}')
```

```
Id:  ['ID_2f6873615' 'ID_1c78846d2' 'ID_e5442cf6a' ... 'ID_07dbb4be2'
 'ID_34d2ed046' 'ID_34754556f']
idhogar:  ['72958b30c' '5b598fbc9' '1e2fc704e' ... '2edb6f51e' '3aa78c56
b'
 'd237404b6']
dependency:  ['.5' 'no' '8' 'yes' '.25' '2' '.33333334' '.375' '.6000000
2' '1.5' '.2'
 '.75' '.66666669' '3' '.14285715' '.40000001' '.80000001' '1.6666666'
 '.2857143' '1.25' '2.5' '5' '.85714287' '1.3333334' '.16666667' '4'
 '.125' '.83333331' '2.3333333' '7' '1.2' '3.5' '2.25' '3.3333333' '6']
edjefe:  ['no' '16' '10' '6' '11' '8' '13' '14' '5' '3' '9' '17' '15' '7
' '21' '4'
 '12' '2' '20' 'yes' '19' '18']
edjefa:  ['17' 'no' '11' '14' '10' '15' '9' '6' '8' '3' '2' '5' '16' '12
' 'yes' '7'
 '13' '21' '4' '19' '18' '20']
```

In [21]:
```
c = ['Id','idhogar']
```

In [22]:
```
train = train.drop(c,axis=1)
test = test.drop(c,axis=1)
```

In [23]:
```
for column in train:
  if train[column].dtype == 'object':
    print(f'{column}:  {train[column].unique()}')
```

```
dependency:  ['no' '8' 'yes' '3' '.5' '.25' '2' '.66666669' '.33333334'
'1.5'
 '.40000001' '.75' '1.25' '.2' '2.5' '1.2' '4' '1.3333334' '2.25'
 '.22222222' '5' '.83333331' '.80000001' '6' '3.5' '1.6666666' '.2857143
'
 '1.75' '.71428573' '.16666667' '.60000002']
edjefe:  ['10' '12' 'no' '11' '9' '15' '4' '6' '8' '17' '7' '16' '14' '5
' '21' '2'
 '19' 'yes' '3' '18' '13' '20']
edjefa:  ['no' '11' '4' '10' '9' '15' '7' '14' '13' '8' '17' '6' '5' '3'
'16' '19'
 'yes' '21' '12' '2' '20' '18']
```

In [24]:
```
def map(x):
  if x == 'yes':
    return 1
  elif x == 'no':
    return 0
  else:
    return float(x)
```

In [25]:
```
train['dependency'] = train['dependency'].apply(map)
train['edjefe'] = train['edjefe'].apply(map)
train['edjefa'] = train['edjefa'].apply(map)
test['dependency'] = test['dependency'].apply(map)
test['edjefe'] = test['edjefe'].apply(map)
test['edjefa'] = test['edjefa'].apply(map)
```

In [26]:
```
for column in train:
  if train[column].dtype == 'int64':
    print(f'{column}:    {train[column].isnull().sum()}')
    print(train[column].isnull().any().sum())
```

```
hacdor:    0
0
rooms:    0
0
hacapo:    0
0
v14a:    0
0
refrig:    0
0
v18q:    0
0
r4h1:    0
0
r4h2:    0
0
r4h3:    0
0
r4m1:    0
0
r4m2:    0
0
r4m3:    0
0
r4t1:    0
0
r4t2:    0
0
r4t3:    0
0
tamhog:    0
0
tamviv:    0
0
escolari:    0
0
hhsize:    0
0
paredblolad:    0
0
paredzocalo:    0
0
paredpreb:    0
0
pareddes:    0
0
paredmad:    0
0
paredzinc:    0
0
paredfibras:    0
0
paredother:    0
0
pisomoscer:    0
0
pisocemento:    0
0
pisoother:    0
0
```

```
pisonatur:    0
0
pisonotiene:    0
0
pisomadera:    0
0
techozinc:    0
0
techoentrepiso:    0
0
techocane:    0
0
techootro:    0
0
cielorazo:    0
0
abastaguadentro:    0
0
abastaguafuera:    0
0
abastaguano:    0
0
public:    0
0
planpri:    0
0
noelec:    0
0
coopele:    0
0
sanitario1:    0
0
sanitario2:    0
0
sanitario3:    0
0
sanitario5:    0
0
sanitario6:    0
0
energcocinar1:    0
0
energcocinar2:    0
0
energcocinar3:    0
0
energcocinar4:    0
0
elimbasu1:    0
0
elimbasu2:    0
0
elimbasu3:    0
0
elimbasu4:    0
0
elimbasu5:    0
0
elimbasu6:    0
0
```

```
epared1:    0
0
epared2:    0
0
epared3:    0
0
etecho1:    0
0
etecho2:    0
0
etecho3:    0
0
eviv1:    0
0
eviv2:    0
0
eviv3:    0
0
dis:    0
0
male:    0
0
female:    0
0
estadocivil1:    0
0
estadocivil2:    0
0
estadocivil3:    0
0
estadocivil4:    0
0
estadocivil5:    0
0
estadocivil6:    0
0
estadocivil7:    0
0
parentesco1:    0
0
parentesco2:    0
0
parentesco3:    0
0
parentesco4:    0
0
parentesco5:    0
0
parentesco6:    0
0
parentesco7:    0
0
parentesco8:    0
0
parentesco9:    0
0
parentesco10:    0
0
parentesco11:    0
0
```

```
parentesco12:    0
0
hogar_nin:    0
0
hogar_adul:    0
0
hogar_mayor:    0
0
hogar_total:    0
0
instlevel1:    0
0
instlevel2:    0
0
instlevel3:    0
0
instlevel4:    0
0
instlevel5:    0
0
instlevel6:    0
0
instlevel7:    0
0
instlevel8:    0
0
instlevel9:    0
0
bedrooms:    0
0
tipovivi1:    0
0
tipovivi2:    0
0
tipovivi3:    0
0
tipovivi4:    0
0
tipovivi5:    0
0
computer:    0
0
television:    0
0
mobilephone:    0
0
qmobilephone:    0
0
lugar1:    0
0
lugar2:    0
0
lugar3:    0
0
lugar4:    0
0
lugar5:    0
0
lugar6:    0
0
```

```
area1:    0
0
area2:    0
0
age:    0
0
SQBescolari:    0
0
SQBage:    0
0
SQBhogar_total:    0
0
SQBedjefe:    0
0
SQBhogar_nin:    0
0
agesq:    0
0
Target:    0
```

In [27]:
```
for column in train:
  if train[column].dtype == 'float64':
    print(f'{column}:    {train[column].isnull().sum()}')
```

```
v2a1:    6860
v18q1:    7342
rez_esc:    7928
dependency:    0
edjefe:    0
edjefa:    0
meaneduc:    5
overcrowding:    0
SQBovercrowding:    0
SQBdependency:    0
SQBmeaned:    5
```

In [28]:
```
for column in test:
  if test[column].dtype == 'int64':
    print(f'{column}:    {test[column].isnull().sum()}')
```

```
hacdor:    0
rooms:    0
hacapo:    0
v14a:    0
refrig:    0
v18q:    0
r4h1:    0
r4h2:    0
r4h3:    0
r4m1:    0
r4m2:    0
r4m3:    0
r4t1:    0
r4t2:    0
r4t3:    0
tamhog:    0
tamviv:    0
escolari:    0
hhsize:    0
paredblolad:    0
paredzocalo:    0
paredpreb:    0
pareddes:    0
paredmad:    0
paredzinc:    0
paredfibras:    0
paredother:    0
pisomoscer:    0
pisocemento:    0
pisoother:    0
pisonatur:    0
pisonotiene:    0
pisomadera:    0
techozinc:    0
techoentrepiso:    0
techocane:    0
techootro:    0
cielorazo:    0
abastaguadentro:    0
abastaguafuera:    0
abastaguano:    0
public:    0
planpri:    0
noelec:    0
coopele:    0
sanitario1:    0
sanitario2:    0
sanitario3:    0
sanitario5:    0
sanitario6:    0
energcocinar1:    0
energcocinar2:    0
energcocinar3:    0
energcocinar4:    0
elimbasu1:    0
elimbasu2:    0
elimbasu3:    0
elimbasu4:    0
elimbasu5:    0
elimbasu6:    0
```

```
epared1:    0
epared2:    0
epared3:    0
etecho1:    0
etecho2:    0
etecho3:    0
eviv1:    0
eviv2:    0
eviv3:    0
dis:    0
male:    0
female:    0
estadocivil1:    0
estadocivil2:    0
estadocivil3:    0
estadocivil4:    0
estadocivil5:    0
estadocivil6:    0
estadocivil7:    0
parentesco1:    0
parentesco2:    0
parentesco3:    0
parentesco4:    0
parentesco5:    0
parentesco6:    0
parentesco7:    0
parentesco8:    0
parentesco9:    0
parentesco10:    0
parentesco11:    0
parentesco12:    0
hogar_nin:    0
hogar_adul:    0
hogar_mayor:    0
hogar_total:    0
instlevel1:    0
instlevel2:    0
instlevel3:    0
instlevel4:    0
instlevel5:    0
instlevel6:    0
instlevel7:    0
instlevel8:    0
instlevel9:    0
bedrooms:    0
tipovivi1:    0
tipovivi2:    0
tipovivi3:    0
tipovivi4:    0
tipovivi5:    0
computer:    0
television:    0
mobilephone:    0
qmobilephone:    0
lugar1:    0
lugar2:    0
lugar3:    0
lugar4:    0
lugar5:    0
lugar6:    0
```

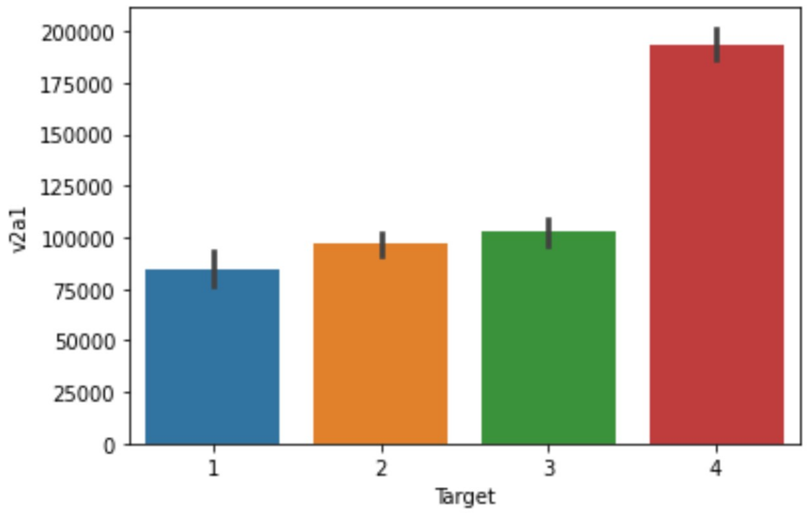
```
area1:    0
area2:    0
age:    0
SQBescolari:    0
SQBage:    0
SQBhogar_total:    0
SQBedjefe:    0
SQBhogar_nin:    0
agesq:    0
```

In [29]:
```
for column in test:
  if test[column].dtype == 'float64':
    print(f'{column}:    {test[column].isnull().sum()}')
```

```
v2a1:    17403
v18q1:    18126
rez_esc:    19653
dependency:    0
edjefe:    0
edjefa:    0
meaneduc:    31
overcrowding:    0
SQBovercrowding:    0
SQBdependency:    0
SQBmeaned:    31
```

In [30]:
```
sns.barplot(x='Target',y='v2a1',data=train)
```

Out[30]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd853deb410>
```

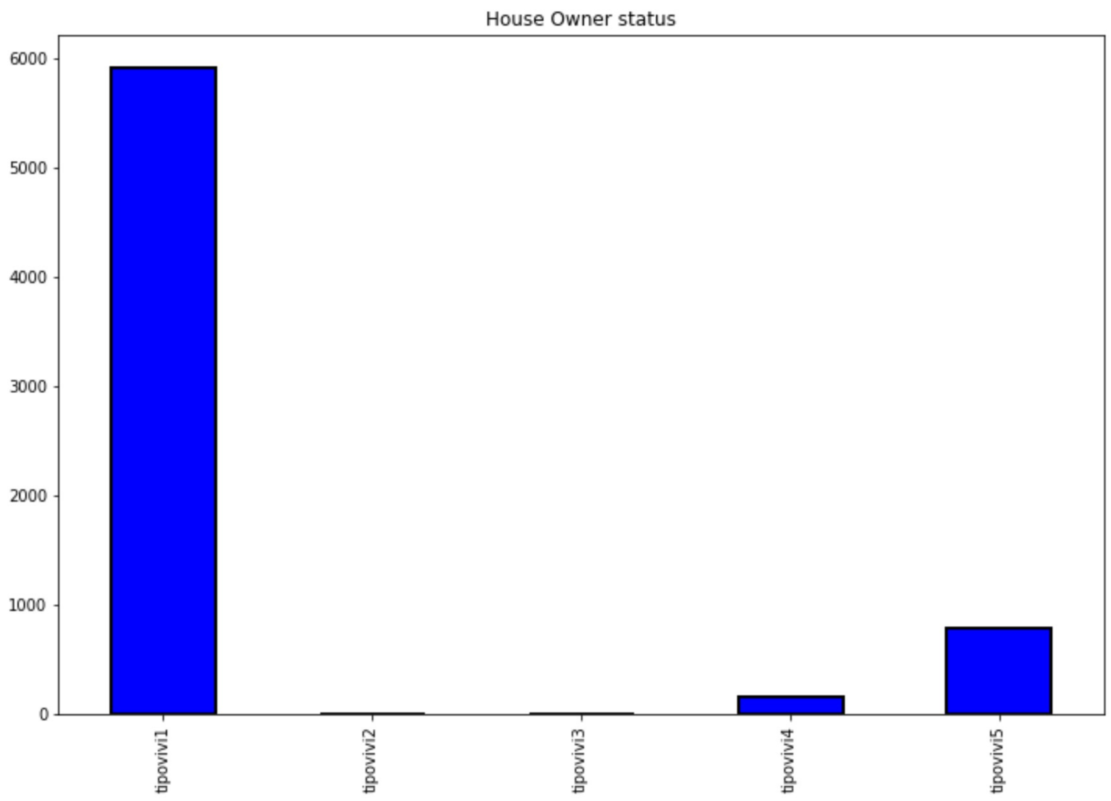


tipovivi1, =1 own and fully paid house, tipovivi2, "=1 own, paying in installments", tipovivi3, =1 rented, tipovivi4, =1 precarious, tipovivi5, "=1 other(assigned, borrowed)"

In [31]:
```
t = ['tipovivi1','tipovivi2','tipovivi3','tipovivi4','tipovivi5']
```

In [32]:
```
train.loc[train['v2a1'].isnull(),t].sum().plot.bar(figsize=(12,8),color='
b',edgecolor='k',linewidth=2)
plt.title('House Owner status')
```

Out[32]:
```
Text(0.5, 1.0, 'House Owner status')
```



In [33]:
```
train['v2a1'].fillna(0,inplace=True)
```

In [34]:
```
test['v2a1'].fillna(0,inplace=True)
```

In [35]:
```
train['v2a1'].isnull().sum()
```

Out[35]:
```
0
```

In [36]:
```
test['v2a1'].isnull().sum()
```

Out[36]:
```
0
```

Check if there is a house without a family head (parentesco1, =1 if household head)

In [37]:
```
train['parentesco1'].value_counts()
```

Out[37]:
```
0    6584
1    2973
Name: parentesco1, dtype: int64
```

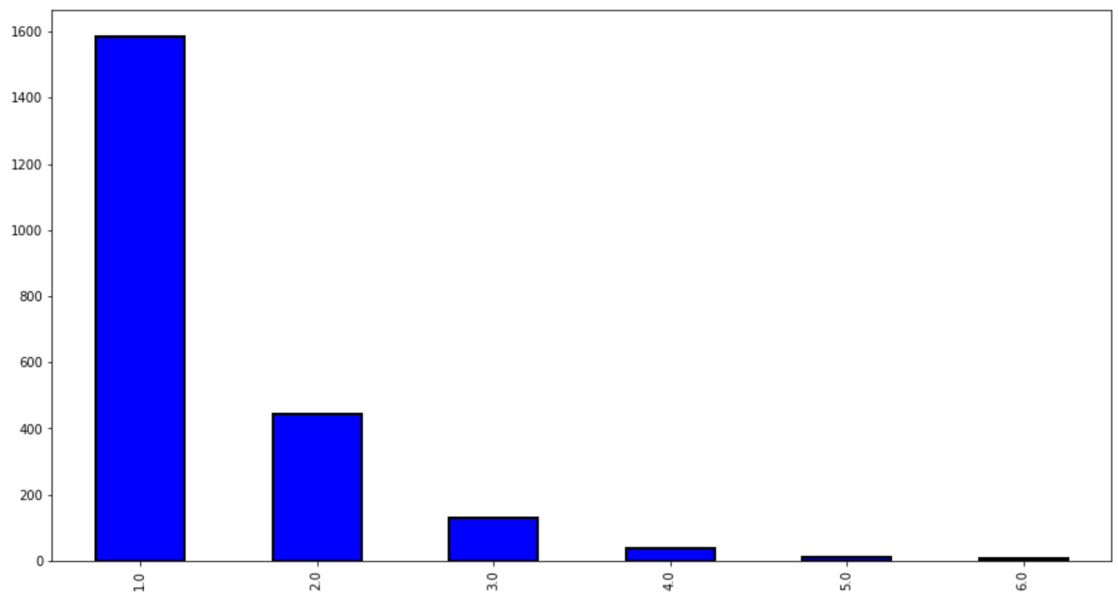
In [38]:
```
train['v18q1'].value_counts()
```

Out[38]:
```
1.0    1586
2.0     444
3.0     129
4.0      37
5.0      13
6.0       6
Name: v18q1, dtype: int64
```

In [39]:
```
plt.figure(figsize=(15,8))

train['v18q1'].value_counts().sort_index().plot.bar(color='blue',edgecolo
r='k',linewidth=2)
```

Out[39]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd853822350>
```



In [40]:
```
train['v18q1'].fillna(0,inplace=True)
```

In [41]:
```
test['v18q1'].fillna(0,inplace=True)
```

In [42]:
```
train['v18q1'].isnull().sum()
```

Out[42]:
```
0
```

In [43]:
```
test['v18q1'].isnull().sum()
```

Out[43]:
```
0
```

1. SQBescolari= escolari squared
2. SQBage, age squared
3. SQBhogar_total, hogar_total squared
4. SQBedjefe, edjefe squared
5. SQBhogar_nin, hogar_nin squared
6. SQBovercrowding, overcrowding squared
7. SQBdependency, dependency squared
8. SQBmeaned, square of the mean years of education of adults (>=18) in the household
9. agesq= Age squared , removing all this columns

```
In [44]: train= train.drop(['SQBescolari','SQBage','SQBhogar_total','SQBedjefe','S
QBhogar_nin','SQBovercrowding','SQBdependency','SQBmeaned','agesq'],axis=
1)
```

```
In [45]: test= test.drop(['SQBescolari','SQBage','SQBhogar_total','SQBedjefe','SQB
hogar_nin','SQBovercrowding','SQBdependency','SQBmeaned','agesq'],axis=1)
```

```
In [46]: train.shape
```

```
Out[46]: (9557, 132)
```

```
In [47]: test.shape
```

```
Out[47]: (23856, 131)
```

In [48]:
```
train['dependency'].value_counts()
```

Out[48]:
```
1.000000    2192
0.000000    1747
0.500000    1497
2.000000     730
1.500000     713
0.333333     598
0.666667     487
8.000000     378
0.250000     260
3.000000     236
4.000000     100
0.750000      98
0.200000      90
0.400000      84
1.333333      84
2.500000      77
5.000000      24
1.250000      18
3.500000      18
0.800000      18
2.250000      13
0.714286      12
1.750000      11
1.200000      11
0.833333      11
0.222222      11
0.285714       9
1.666667       8
0.600000       8
6.000000       7
0.166667       7
Name: dependency, dtype: int64
```

In [49]:
```
#Checking for 0 variance
for column in train:
  if train[column].var() == 0:
    print(column)
```

```
elimbasu5
```

In [50]:
```
#Checking for 0 variance
for column in test:
  if test[column].var() == 0:
    print(column)
```

In [51]:
```
train['rez_esc'].value_counts()
```

Out[51]:
```
0.0    1211
1.0     227
2.0      98
3.0      55
4.0      29
5.0       9
Name: rez_esc, dtype: int64
```

In [52]:
```
train['rez_esc'].isnull().sum()
```

Out[52]:
```
7928
```

3. Lets look at rez_esc (total nulls: 7928) : Years behind in school

why the null values, Lets look at few rows with nulls in rez_esc

Columns related to Years behind in school

Age in years

Lets look at the data with not null values first

In [53]:
```
train[train['rez_esc'].notnull()]['age'].describe()
```

Out[53]:
```
count    1629.000000
mean       12.258441
std         3.218325
min         7.000000
25%         9.000000
50%        12.000000
75%        15.000000
max        17.000000
Name: age, dtype: float64
```

In [54]:
```
train.loc[(train['rez_esc'].isnull() & ((train['age'] > 7) & (train['age
'] < 17)))]['age'].describe()
```

Out[54]:
```
count     1.0
mean     10.0
std       NaN
min      10.0
25%      10.0
50%      10.0
75%      10.0
max      10.0
Name: age, dtype: float64
```

In [55]:
```
train[(train['age'] ==10) & train['rez_esc'].isnull()].head()
```

Out[55]:

	v2a1	hacdor	rooms	hacapo	v14a	refrig	v18q	v18q1	r4h1	r4h2	r4h3	r4m
2514	160000.0	0	6	0	1	1	1	1.0	0	1	1	

In [56]:
```
train['rez_esc'].fillna(0,inplace=True)
test['rez_esc'].fillna(0,inplace=True)
```

In [57]:
```
train['rez_esc'].isnull().sum()
```
Out[57]:
```
0
```

In [58]:
```
test['rez_esc'].isnull().sum()
```
Out[58]:
```
0
```

Lets look at meaneduc (total nulls: 5) : average years of education for adults (18+)
why the null values, Lets look at few rows with nulls in meaneduc Columns related to average years of education for adults (18+)
edjefe, years of education of male head of household, based on the interaction of escolari (years of education), head of household and gender, yes=1 and no=0 edjefa, years of education of female head of household, based on the interaction of escolari (years of education), head of household and gender, yes=1 and no=0 instlevel1, =1 no level of education instlevel2, =1 incomplete primary

In [59]:
```
data = train[train['meaneduc'].isnull()].head()

columns=['edjefe','edjefa','instlevel1','instlevel2']
data[columns][data[columns]['instlevel1']>0].describe()
```
Out[59]:

	edjefe	edjefa	instlevel1	instlevel2
count	0.0	0.0	0.0	0.0
mean	NaN	NaN	NaN	NaN
std	NaN	NaN	NaN	NaN
min	NaN	NaN	NaN	NaN
25%	NaN	NaN	NaN	NaN
50%	NaN	NaN	NaN	NaN
75%	NaN	NaN	NaN	NaN
max	NaN	NaN	NaN	NaN

In [60]:
```
train['meaneduc'].fillna(0,inplace=True)
test['meaneduc'].fillna(0,inplace=True)
```

In [61]:
```
id = [ 'Target']

individual_bool = ['v18q', 'dis', 'male', 'female', 'estadocivil1', 'esta
docivil2', 'estadocivil3',
            'estadocivil4', 'estadocivil5', 'estadocivil6', 'estadocivil7
',
            'parentesco1', 'parentesco2',  'parentesco3', 'parentesco4',
'parentesco5',
            'parentesco6', 'parentesco7', 'parentesco8',  'parentesco9',
'parentesco10',
            'parentesco11', 'parentesco12', 'instlevel1', 'instlevel2', '
instlevel3',
            'instlevel4', 'instlevel5', 'instlevel6', 'instlevel7', 'inst
level8',
            'instlevel9', 'mobilephone']

individual_ordered = ['rez_esc', 'escolari', 'age']

hh_bool = ['hacdor', 'hacapo', 'v14a', 'refrig', 'paredblolad', 'paredzoc
alo',
           'paredpreb','pisocemento', 'pareddes', 'paredmad',
           'paredzinc', 'paredfibras', 'paredother', 'pisomoscer', 'pisoo
ther',
           'pisonatur', 'pisonotiene', 'pisomadera',
           'techozinc', 'techoentrepiso', 'techocane', 'techootro', 'ciel
orazo',
           'abastaguadentro', 'abastaguafuera', 'abastaguano',
            'public', 'planpri', 'noelec', 'coopele', 'sanitario1',
           'sanitario2', 'sanitario3', 'sanitario5',   'sanitario6',
           'energcocinar1', 'energcocinar2', 'energcocinar3', 'energcocin
ar4',
           'elimbasu1', 'elimbasu2', 'elimbasu3', 'elimbasu4',
           'elimbasu5', 'elimbasu6', 'epared1', 'epared2', 'epared3',
           'etecho1', 'etecho2', 'etecho3', 'eviv1', 'eviv2', 'eviv3',
           'tipovivi1', 'tipovivi2', 'tipovivi3', 'tipovivi4', 'tipovivi5
',
           'computer', 'television', 'lugar1', 'lugar2', 'lugar3',
           'lugar4', 'lugar5', 'lugar6', 'area1', 'area2']

hh_ordered = [ 'rooms', 'r4h1', 'r4h2', 'r4h3', 'r4m1','r4m2','r4m3', 'r4
t1',  'r4t2',
              'r4t3', 'v18q1', 'tamhog','tamviv','hhsize','hogar_nin',
              'hogar_adul','hogar_mayor','hogar_total',  'bedrooms', 'qmo
bilephone']

hh_cont = ['v2a1', 'dependency', 'edjefe', 'edjefa', 'meaneduc', 'overcro
wding']
```

In [62]:
```
#Check for redundant household variables
heads = train.loc[train['parentesco1'] == 1, :]
heads = heads[id + hh_bool + hh_cont + hh_ordered]
heads.shape
```

Out[62]:
```
(2973, 96)
```

In [63]:
```
# Create correlation matrix
corr_matrix = heads.corr()

# Select upper triangle of correlation matrix
upper = corr_matrix.where(np.triu(np.ones(corr_matrix.shape), k=1).astype
(np.bool))

# Find index of feature columns with correlation greater than 0.95
to_drop = [column for column in upper.columns if any(abs(upper[column]) >
0.95)]

to_drop
```

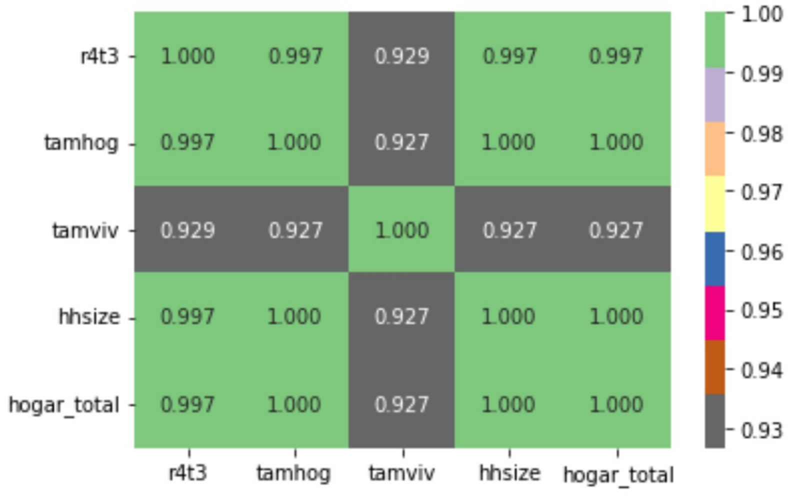
Out[63]:
```
['coopele', 'area2', 'tamhog', 'hhsize', 'hogar_total']
```

In [64]:
```
corr_matrix.loc[corr_matrix['tamhog'].abs() > 0.9, corr_matrix['tamhog'].
abs() > 0.9]
```

Out[64]:

	r4t3	tamhog	tamviv	hhsize	hogar_total
r4t3	1.000000	0.996884	0.929237	0.996884	0.996884
tamhog	0.996884	1.000000	0.926667	1.000000	1.000000
tamviv	0.929237	0.926667	1.000000	0.926667	0.926667
hhsize	0.996884	1.000000	0.926667	1.000000	1.000000
hogar_total	0.996884	1.000000	0.926667	1.000000	1.000000

In [65]:
```
sns.heatmap(corr_matrix.loc[corr_matrix['tamhog'].abs() > 0.9, corr_matri
x['tamhog'].abs() > 0.9],
            annot=True, cmap = plt.cm.Accent_r, fmt='.3f');
```



There are several variables here having to do with the size of the house: r4t3, Total persons in the household tamhog, size of the household tamviv, number of persons living in the household hhsize, household size hogar_total, # of total individuals in the household These variables are all highly correlated with one another.

In [66]:
```
train = train.drop(['tamhog', 'hogar_total', 'r4t3'],axis=1)
test = test.drop(['tamhog', 'hogar_total', 'r4t3'],axis=1)
```

In [67]:
```
train.shape
```
Out[67]:
```
(9557, 129)
```

In [69]:
```
#Check for redundant Individual variables
individual = train[id + individual_bool + individual_ordered]
individual.shape
```
Out[69]:
```
(9557, 37)
```

In [71]:
```
# Create correlation matrix
corr_matrix = individual.corr()

# Select upper triangle of correlation matrix
upper = corr_matrix.where(np.triu(np.ones(corr_matrix.shape), k=1).astype
(np.bool))

# Find index of feature columns with correlation greater than 0.95
to_drop = [column for column in upper.columns if any(abs(upper[column]) >
0.95)]

to_drop
```
Out[71]:
```
['female']
```

In [72]:
```
# This is simply We can remove the female flag.
train = train.drop('female',axis=1)
test = test.drop('female',axis=1)
```

In [73]:
```
#lets check area1 and area2 also
# area1, =1 zona urbana
# area2, =2 zona rural
#area2 redundant because we have a column indicating if the house is in a
urban zone

train = train.drop('area2',axis=1)
test = test.drop('area2',axis=1)
```

In [74]:
```
X = train.drop('Target',axis=1)
Y = train['Target']
```

In [75]:
```
from sklearn.model_selection import train_test_split
```

In [77]:
```
X_train,X_test,y_train,y_test = train_test_split(X,Y,test_size=0.3,random
_state=1)
```

In [78]:
```
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```
```
(6689, 126)
(2868, 126)
(6689,)
(2868,)
```

In [79]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score,f1_score,classification_repor
t,confusion_matrix
```

In [80]:
```
RFC = RandomForestClassifier()
```

In [81]:
```
RFC.fit(X_train,y_train)
```

Out[81]:
```
RandomForestClassifier()
```

In [82]:
```
pred = RFC.predict(X_test)
```

In [83]:
```
print(accuracy_score(y_test,pred))
print(confusion_matrix(y_test,pred))
print(classification_report(y_test,pred))
```

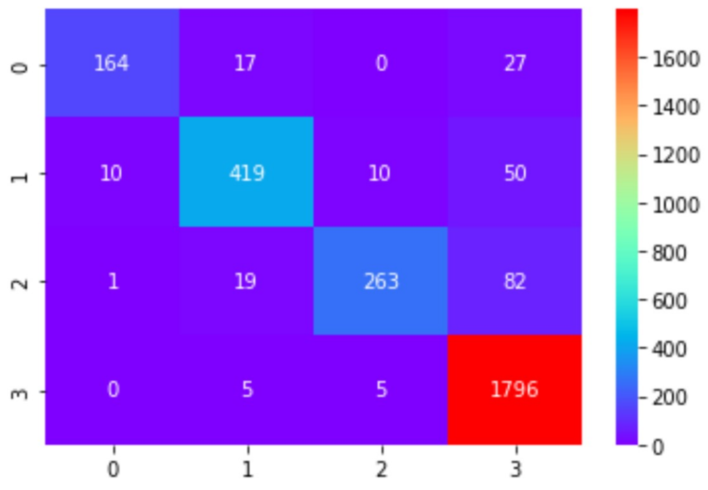
```
0.9211994421199442
[[ 164   17    0   27]
 [   10  419   10   50]
 [    1   19  263   82]
 [    0    5    5 1796]]
              precision    recall  f1-score   support

           1       0.94      0.79      0.86       208
           2       0.91      0.86      0.88       489
           3       0.95      0.72      0.82       365
           4       0.92      0.99      0.96      1806

    accuracy                           0.92      2868
   macro avg       0.93      0.84      0.88      2868
weighted avg       0.92      0.92      0.92      2868
```

In [84]:
```
sns.heatmap(confusion_matrix(y_test,pred),annot=True,fmt='d',cmap = plt.c
m.rainbow)
```

Out[84]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd84f800590>
```



In [85]:
```
y_preds = RFC.predict(test)
```

In [86]:
```
y_preds
```

Out[86]:
```
array([4, 4, 4, ..., 4, 4, 4])
```

Step 6: Check the accuracy using random forest with cross validation.

In [89]:
```
from sklearn.model_selection import KFold,cross_val_score
```

In [90]:
```
kfolds = KFold(n_splits=5,random_state=7,shuffle=True)
rmclassifier=RandomForestClassifier(random_state=10,n_jobs = -1)
print(cross_val_score(rmclassifier,X,Y,cv=kfolds,scoring='accuracy'))
results=cross_val_score(rmclassifier,X,Y,cv=kfolds,scoring='accuracy')
print(results.mean()*100)
```

```
[0.92834728 0.93148536 0.92830979 0.92255364 0.93040293]
92.82197977356375
```

Checking the score using 100 trees

In [91]:
```
rmclassifier=RandomForestClassifier(n_estimators=100,random_state=10,n_jo
bs = -1)
print(cross_val_score(rmclassifier,X,Y,cv=kfolds,scoring='accuracy'))
results=cross_val_score(rmclassifier,X,Y,cv=kfolds,scoring='accuracy')
print(results.mean()*100)
```

```
[0.92834728 0.93148536 0.92830979 0.92255364 0.93040293]
92.82197977356375
```

In [93]:
```
y_predict_testdata = RFC.predict(test)
y_predict_testdata
```

Out[93]:
```
array([4, 4, 4, ..., 4, 4, 4])
```

In [94]:
```
rmclassifier.fit(X,Y)
labels = list(X)
feature_importances = pd.DataFrame({'feature': labels, 'importance': rmcl
assifier.feature_importances_})
feature_importances=feature_importances[feature_importances.importance>0.
015]
feature_importances.head()
```

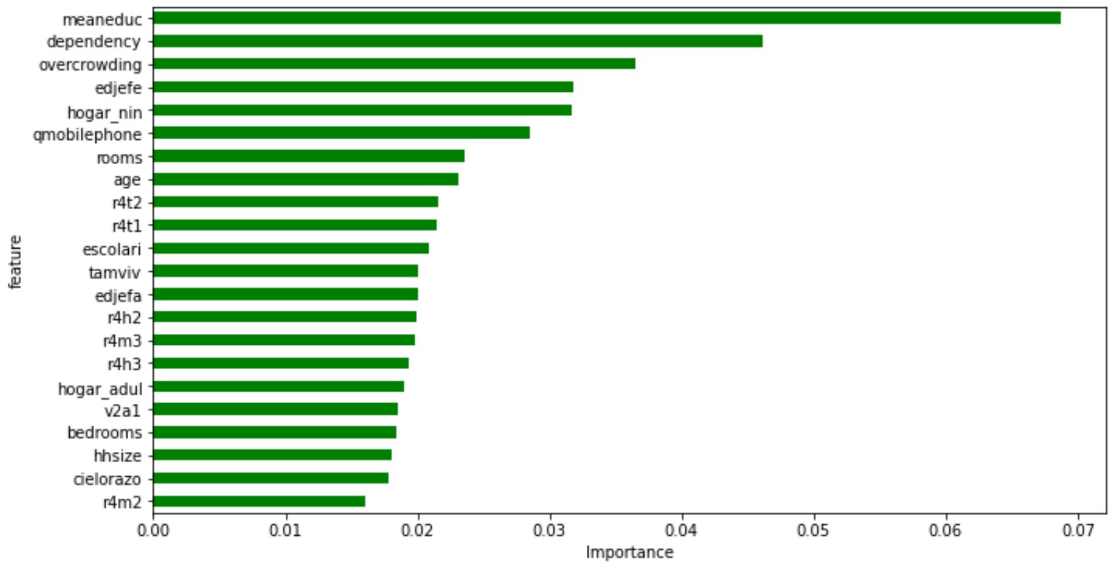
Out[94]:

	feature	importance
0	v2a1	0.018565
2	rooms	0.023626
9	r4h2	0.019918
10	r4h3	0.019287
12	r4m2	0.016104

In [95]:
```
feature_importances.sort_values(by=['importance'], ascending=True, inplac
e=True)
feature_importances['positive'] = feature_importances['importance'] > 0
feature_importances.set_index('feature',inplace=True)
feature_importances.head()

feature_importances.importance.plot(kind='barh', figsize=(11, 6),color =
feature_importances.positive.map({True: 'Green', False: 'Red'}))
plt.xlabel('Importance')
```

Out[95]:
```
Text(0.5, 0, 'Importance')
```



In [ ]: