In [2]:

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

In [11]:

```
import warnings
warnings.filterwarnings('ignore')
```

In [12]:

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

In [13]:

```
df_income_train.head(10)
```

Out[13]:

	Id	v2a1	hacdor	rooms	hacapo	v14a	refrig	v18q	v18q1	r4h1	...	SQ
0	ID_279628684	190000.0	0	3	0	1	1	0	NaN	0	...	
1	ID_f29eb3ddd	135000.0	0	4	0	1	1	1	1.0	0	...	
2	ID_68de51c94	NaN	0	8	0	1	1	0	NaN	0	...	
3	ID_d671db89c	180000.0	0	5	0	1	1	1	1.0	0	...	
4	ID_d56d6f5f5	180000.0	0	5	0	1	1	1	1.0	0	...	
5	ID_ec05b1a7b	180000.0	0	5	0	1	1	1	1.0	0	...	
6	ID_e9e0c1100	180000.0	0	5	0	1	1	1	1.0	0	...	
7	ID_3e04e571e	130000.0	1	2	0	1	1	0	NaN	0	...	
8	ID_1284f8aad	130000.0	1	2	0	1	1	0	NaN	0	...	
9	ID_51f52fdd2	130000.0	1	2	0	1	1	0	NaN	0	...	

10 rows × 143 columns

In [14]:

```
df_income_test.head(10)
```

Out[14]:

	Id	v2a1	hacdor	rooms	hacapo	v14a	refrig	v18q	v18q1	r4h1	...	ag
0	ID_2f6873615	NaN	0	5	0	1	1	0	NaN	1	...	
1	ID_1c78846d2	NaN	0	5	0	1	1	0	NaN	1	...	4
2	ID_e5442cf6a	NaN	0	5	0	1	1	0	NaN	1	...	4
3	ID_a8db26a79	NaN	0	14	0	1	1	1	1.0	0	...	5
4	ID_a62966799	175000.0	0	4	0	1	1	1	1.0	0	...	1
5	ID_e77d38d45	400000.0	0	3	0	1	1	1	1.0	0	...	3
6	ID_3c5f4bd51	400000.0	0	3	0	1	1	1	1.0	0	...	4
7	ID_a849c29bd	300000.0	0	6	0	1	1	1	1.0	0	...	2
8	ID_472fa82da	300000.0	0	6	0	1	1	1	1.0	0	...	2
9	ID_24864adcc	NaN	0	6	0	1	1	0	NaN	0	...	7

10 rows × 142 columns

In [15]:

```
df_income_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 [16]:

```
df_income_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 23856 entries, 0 to 23855
Columns: 142 entries, Id to agesq
dtypes: float64(8), int64(129), object(5)
memory usage: 25.8+ MB
```

In [17]:

```
df_income_test.shape
```

Out[17]:

```
(23856, 142)
```

In [18]:

```
df_income_train.shape
```

Out[18]:

```
(9557, 143)
```

In [19]:

```
print('Integer Type: ')
print(df_income_train.select_dtypes(np.int64).columns)
print('/n')
print('Float Type: ')
print(df_income_train.select_dtypes(np.float64).columns)
print('/n')
print('Object Type: ')
print(df_income_train.select_dtypes(np.object).columns)
```

```
Integer Type:
Index(['hacdor', 'rooms', 'hacapo', 'v14a', 'refrig', 'v18q', 'r4h1', 'r4h
2',
       'r4h3', 'r4m1',
       ...
       'area1', 'area2', 'age', 'SQBescolari', 'SQBage', 'SQBhogar_total',
       'SQBedjefe', 'SQBhogar_nin', 'agesq', 'Target'],
      dtype='object', length=130)
/n
Float Type:
Index(['v2a1', 'v18q1', 'rez_esc', 'meaneduc', 'overcrowding',
       'SQBovercrowding', 'SQBdependency', 'SQBmeaned'],
      dtype='object')
/n
Object Type:
Index(['Id', 'idhogar', 'dependency', 'edjefe', 'edjefa'], dtype='object')
```

In [20]:

```
df_income_train.select_dtypes('int64').head(10)
```

Out[20]:

	hacdor	rooms	hacapo	v14a	refrig	v18q	r4h1	r4h2	r4h3	r4m1	...	area1	area2	ag
0	0	3	0	1	1	0	0	1	1	0	...	1	0	4
1	0	4	0	1	1	1	0	1	1	0	...	1	0	6
2	0	8	0	1	1	0	0	0	0	0	...	1	0	9
3	0	5	0	1	1	1	0	2	2	1	...	1	0	1
4	0	5	0	1	1	1	0	2	2	1	...	1	0	3
5	0	5	0	1	1	1	0	2	2	1	...	1	0	3
6	0	5	0	1	1	1	0	2	2	1	...	1	0	
7	1	2	0	1	1	0	0	1	1	2	...	1	0	
8	1	2	0	1	1	0	0	1	1	2	...	1	0	3
9	1	2	0	1	1	0	0	1	1	2	...	1	0	2

10 rows × 130 columns

In [21]:

```
null_counts=df_income_train.select_dtypes('int64').isnull().sum()
null_counts[null_counts>0]
```

Out[21]:

```
Series([], dtype: int64)
```

In [22]:

```
df_income_train.select_dtypes('float64').head(10)
```

Out[22]:

	v2a1	v18q1	rez_esc	meaneduc	overcrowding	SQBovercrowding	SQBdependency
0	190000.0	NaN	NaN	10.0	1.000000	1.000000	0.0
1	135000.0	1.0	NaN	12.0	1.000000	1.000000	64.0
2	NaN	NaN	NaN	11.0	0.500000	0.250000	64.0
3	180000.0	1.0	1.0	11.0	1.333333	1.777778	1.0
4	180000.0	1.0	NaN	11.0	1.333333	1.777778	1.0
5	180000.0	1.0	NaN	11.0	1.333333	1.777778	1.0
6	180000.0	1.0	0.0	11.0	1.333333	1.777778	1.0
7	130000.0	NaN	0.0	10.0	4.000000	16.000000	1.0
8	130000.0	NaN	NaN	10.0	4.000000	16.000000	1.0
9	130000.0	NaN	NaN	10.0	4.000000	16.000000	1.0

In [23]:

```
null_counts=df_income_train.select_dtypes('float64').isnull().sum()
null_counts[null_counts > 0]
```

Out[23]:

```
v2a1          6860
v18q1         7342
rez_esc       7928
meaneduc         5
SQBmeaned        5
dtype: int64
```

In [24]:

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

Out[24]:

	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 [153]:

```
null_counts=df_income_train.select_dtypes('float64').isnull().sum()
null_counts[null_counts > 0]
```

Out[153]:

```
v2a1         6860
v18q1        7342
rez_esc      7928
meaneduc        5
SQBmeaned       5
dtype: int64
```

In [154]:

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

Out[154]:

	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 [25]:

```
null_counts=df_income_train.select_dtypes('object').isnull().sum()
null_counts[null_counts > 0]
```

Out[25]:

```
Series([], dtype: int64)
```

In [26]:

```
mapping={'yes':1,'no':0}
for df in [df_income_train, df_income_test]:
    df['dependency'] =df['dependency'].replace(mapping).astype(np.float64)
    df['edjefe'] =df['edjefe'].replace(mapping).astype(np.float64)
    df['edjefa'] =df['edjefa'].replace(mapping).astype(np.float64)
df_income_train[['dependency','edjefe','edjefa']].describe()
```

Out[26]:

	dependency	edjefe	edjefa
count	9557.000000	9557.000000	9557.000000
mean	1.149550	5.096788	2.896830
std	1.605993	5.246513	4.612056
min	0.000000	0.000000	0.000000
25%	0.333333	0.000000	0.000000
50%	0.666667	6.000000	0.000000
75%	1.333333	9.000000	6.000000
max	8.000000	21.000000	21.000000

In [27]:

```
# 1. Lets look at v2a1 (total nulls: 6860) : Monthly rent payment
# why the null values, Lets look at few rows with nulls in v2a1
# Columns related to Monthly rent payment
# 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 [28]:

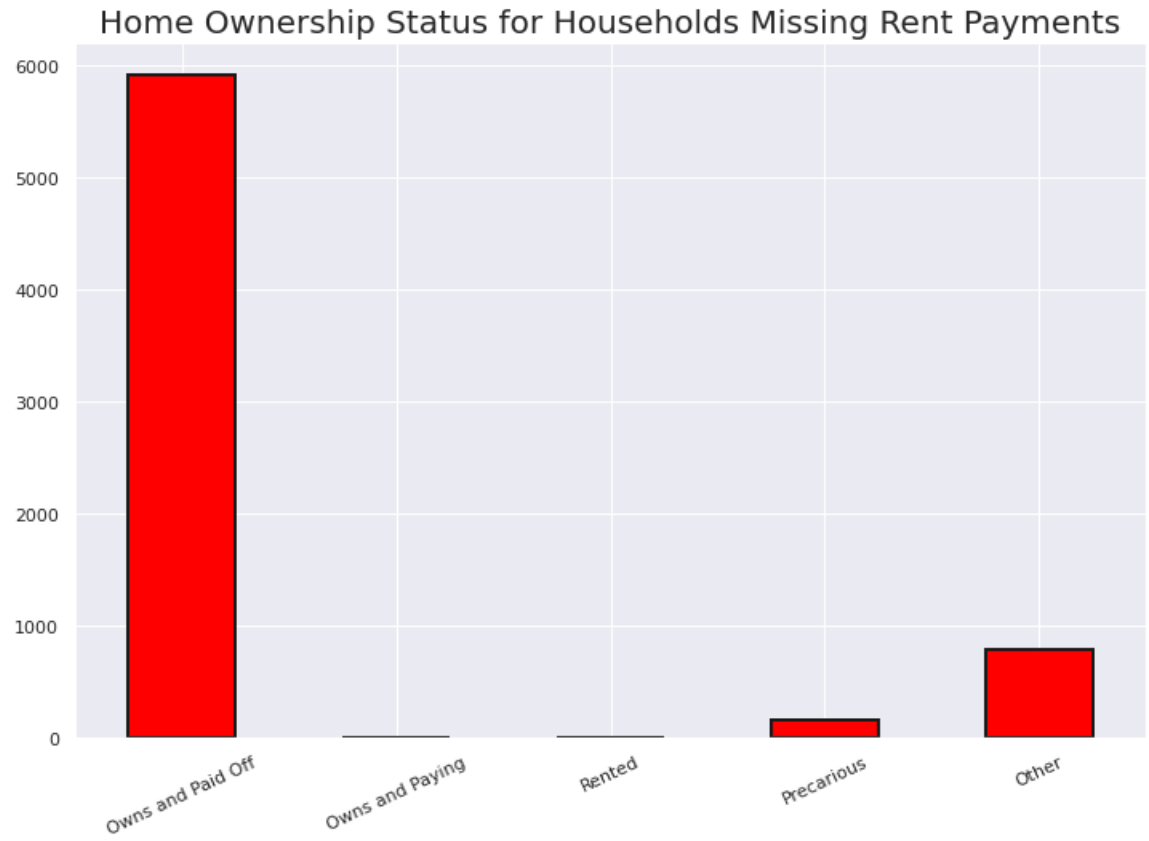
```
data = df_income_train[df_income_train['v2a1'].isnull()].head(10)
columns=['tipovivi1','tipovivi2','tipovivi3','tipovivi4','tipovivi5']
data[columns]
```

Out[28]:

	tipovivi1	tipovivi2	tipovivi3	tipovivi4	tipovivi5
2	1	0	0	0	0
13	1	0	0	0	0
14	1	0	0	0	0
26	1	0	0	0	0
32	1	0	0	0	0
33	1	0	0	0	0
34	1	0	0	0	0
35	1	0	0	0	0
36	1	0	0	0	0
42	1	0	0	0	0

In [29]:

```
# Variables indicating home ownership
own_variables = [x for x in df_income_train if x.startswith('tipo')]
# Plot of the home ownership variables for home missing rent payments
df_income_train.loc[df_income_train['v2a1'].isnull(), own_variables].sum().plot.bar(fig
size = (12, 8),
color = 'red',
edgecolor = 'k', linewidth = 2);
plt.xticks([0, 1, 2, 3, 4],
['Owns and Paid Off', 'Owns and Paying', 'Rented', 'Precarious', 'Other'],
rotation = 25)
plt.title('Home Ownership Status for Households Missing Rent Payments', size = 20);
```



In [30]:

```
#Looking at the above data it makes sense that when the house is fully paid, there will
be no monthly rent payment.
#Lets add 0 for all the null values.

for df in [df_income_train, df_income_test]:
    df['v2a1'].fillna(value=0, inplace=True)
df_income_train[['v2a1']].isnull().sum()
```

Out[30]:

```
v2a1    0
dtype: int64
```

In [31]:

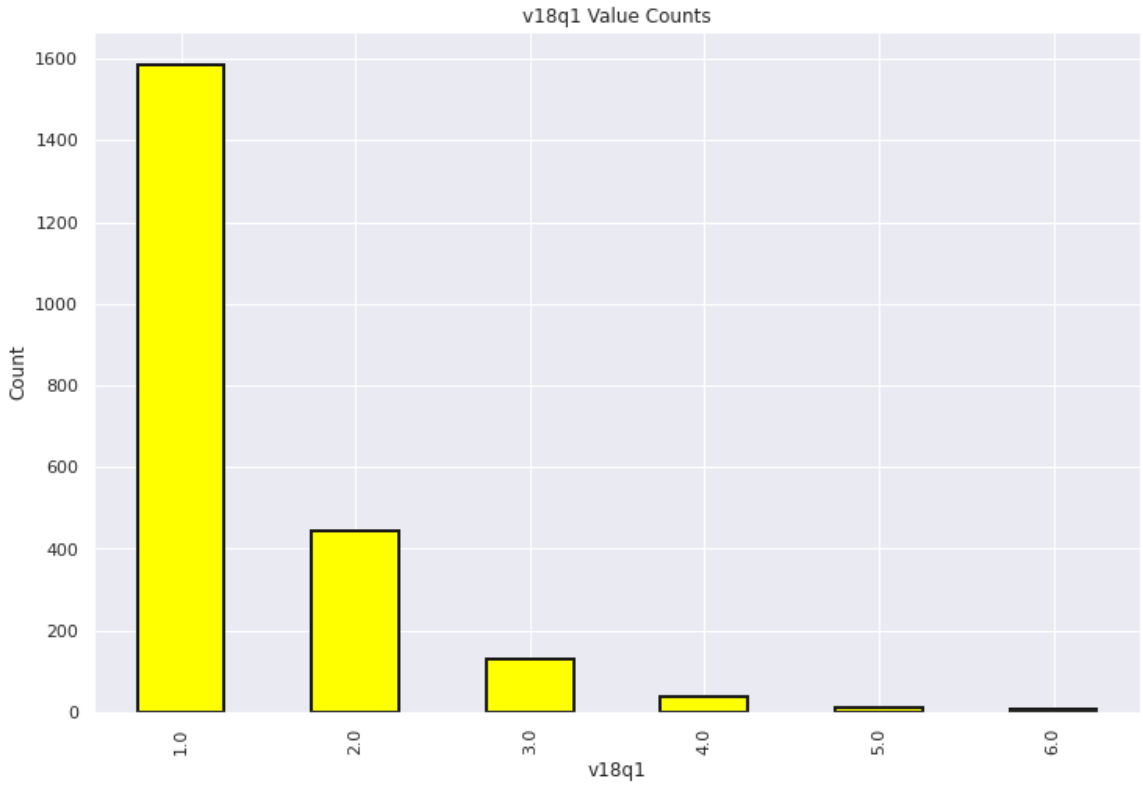
```
heads = df_income_train.loc[df_income_train['parentesco1'] == 1].copy()
heads.groupby('v18q')['v18q1'].apply(lambda x: x.isnull().sum())
```

Out[31]:

```
v18q
0    2318
1       0
Name: v18q1, dtype: int64
```

In [32]:

```
plt.figure(figsize = (12, 8))
col='v18q1'
df_income_train[col].value_counts().sort_index().plot.bar(color = 'yellow',edgecolor =
'k',linewidth = 2)
plt.xlabel(f'{col}'); plt.title(f'{col} Value Counts'); plt.ylabel('Count')
plt.show();
```



In [34]:

```
for df in [df_income_train, df_income_test]:
    df['v18q1'].fillna(value=0, inplace=True)
df_income_train[['v18q1']].isnull().sum()
```

Out[34]:

```
v18q1    0
dtype: int64
```

In [35]:

```
df_income_train[df_income_train['rez_esc'].notnull()]['age'].describe()
```

Out[35]:

```
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 [36]:

```
df_income_train.loc[df_income_train['rez_esc'].isnull()]['age'].describe()
```

Out[36]:

```
count    7928.000000
mean       38.833249
std        20.989486
min         0.000000
25%        24.000000
50%        38.000000
75%        54.000000
max        97.000000
Name: age, dtype: float64
```

In [37]:

```
df_income_train.loc[(df_income_train['rez_esc'].isnull() & ((df_income_train['age'] > 7
) & (df_income_train['age'] < 17)))]['age'].describe()
```

Out[37]:

```
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 [38]:

```
df_income_train[(df_income_train['age'] ==10) & df_income_train['rez_esc'].isnull()].he
ad()
df_income_train[(df_income_train['Id'] =='ID_f012e4242')].head()
```

Out[38]:

	Id	v2a1	hacdor	rooms	hacapo	v14a	refrig	v18q	v18q1	r4h1	...
2514	ID_f012e4242	160000.0	0	6	0	1	1	1	1.0	0	...

1 rows × 143 columns

In [39]:

```
for df in [df_income_train, df_income_test]:
    df['rez_esc'].fillna(value=0, inplace=True)
df_income_train[['rez_esc']].isnull().sum()
```

Out[39]:

```
rez_esc    0
dtype: int64
```

In [40]:

```
data = df_income_train[df_income_train['meaneduc'].isnull()].head()
columns=['edjefe','edjefa','instlevel1','instlevel2']
data[columns][data[columns]['instlevel1']>0].describe()
```

Out[40]:

	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 [41]:

```
for df in [df_income_train, df_income_test]:
    df['meaneduc'].fillna(value=0, inplace=True)
df_income_train[['meaneduc']].isnull().sum()
```

Out[41]:

```
meaneduc    0
dtype: int64
```

In [171]:

```
data = df_income_train[df_income_train['SQBmeaned'].isnull()].head()
columns=['edjefe','edjefa','instlevel1','instlevel2']
data[columns][data[columns]['instlevel1']>0].describe()
```

Out[171]:

	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 [42]:

```
for df in [df_income_train, df_income_test]:
    df['SQBmeaned'].fillna(value=0, inplace=True)
df_income_train[['SQBmeaned']].isnull().sum()
```

Out[42]:

```
SQBmeaned    0
dtype: int64
```

In [43]:

```
null_counts = df_income_train.isnull().sum()
null_counts[null_counts > 0].sort_values(ascending=False)
```

Out[43]:

```
Series([], dtype: int64)
```

Lets look at the target column

In [44]:

```
# Groupby the household and figure out the number of unique values

all_equal = df_income_train.groupby('idhogar')['Target'].apply(lambda x: x.nunique() == 1)

# Households where targets are not all equal
not_equal = all_equal[all_equal != True]

print('There are {} households where the family members do not all have the same target.'.format(len(not_equal)))
```

```
There are 85 households where the family members do not all have the same
target.
```

In [45]:

```
#Lets check one household

df_income_train[df_income_train['idhogar'] == not_equal.index[0]][['idhogar', 'parentesco1', 'Target']]
```

Out[45]:

	idhogar	parentesco1	Target
7651	0172ab1d9	0	3
7652	0172ab1d9	0	2
7653	0172ab1d9	0	3
7654	0172ab1d9	1	3
7655	0172ab1d9	0	2

In [46]:

```
#Lets use Target value of the parent record (head of the household) and update rest. But before that lets check
# if all families has a head.

households_head = df_income_train.groupby('idhogar')['parentesco1'].sum()

# Find households without a head

households_no_head = df_income_train.loc[df_income_train['idhogar'].isin(households_head[households_head == 0].index), :]

print('There are {} households without a head.'.format(households_no_head['idhogar'].nunique()))
```

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

In [47]:

```
households_no_head_equal = households_no_head.groupby('idhogar')['Target'].apply(lambda
x: x.nunique() == 1)
print('{} Households with no head have different Target value.'.format(sum(households_n
o_head_equal == False)))
```

```
0 Households with no head have different Target value.
```

In [48]:

```
#Set poverty level of the members and the head of the house within a family.
# Iterate through each household

for household in not_equal.index:

    # Find the correct label (for the head of household)

    true_target = int(df_income_train[(df_income_train['idhogar'] == household) & (df_i
ncome_train['parentesco1'] == 1.0)]['Target'])

    # Set the correct label for all members in the household

    df_income_train.loc[df_income_train['idhogar'] == household, 'Target'] = true_targe
t

    # Groupby the household and figure out the number of unique values
    all_equal = df_income_train.groupby('idhogar')['Target'].apply(lambda x: x.nunique
() == 1)

    # Households where targets are not all equal
    not_equal = all_equal[all_equal != True]

    print('There are {} households where the family members do not all have the same ta
rget.'.format(len(not_equal)))
```

```
There are 84 households where the family members do not all have the same
target.
There are 83 households where the family members do not all have the same
target.
There are 82 households where the family members do not all have the same
target.
There are 81 households where the family members do not all have the same
target.
There are 80 households where the family members do not all have the same
target.
There are 79 households where the family members do not all have the same
target.
There are 78 households where the family members do not all have the same
target.
There are 77 households where the family members do not all have the same
target.
There are 76 households where the family members do not all have the same
target.
There are 75 households where the family members do not all have the same
target.
There are 74 households where the family members do not all have the same
target.
There are 73 households where the family members do not all have the same
target.
There are 72 households where the family members do not all have the same
target.
There are 71 households where the family members do not all have the same
target.
There are 70 households where the family members do not all have the same
target.
There are 69 households where the family members do not all have the same
target.
There are 68 households where the family members do not all have the same
target.
There are 67 households where the family members do not all have the same
target.
There are 66 households where the family members do not all have the same
target.
There are 65 households where the family members do not all have the same
target.
There are 64 households where the family members do not all have the same
target.
There are 63 households where the family members do not all have the same
target.
There are 62 households where the family members do not all have the same
target.
There are 61 households where the family members do not all have the same
target.
There are 60 households where the family members do not all have the same
target.
There are 59 households where the family members do not all have the same
target.
There are 58 households where the family members do not all have the same
target.
There are 57 households where the family members do not all have the same
target.
There are 56 households where the family members do not all have the same
target.
There are 55 households where the family members do not all have the same
target.
There are 54 households where the family members do not all have the same
```

```
target.
There are 53 households where the family members do not all have the same
target.
There are 52 households where the family members do not all have the same
target.
There are 51 households where the family members do not all have the same
target.
There are 50 households where the family members do not all have the same
target.
There are 49 households where the family members do not all have the same
target.
There are 48 households where the family members do not all have the same
target.
There are 47 households where the family members do not all have the same
target.
There are 46 households where the family members do not all have the same
target.
There are 45 households where the family members do not all have the same
target.
There are 44 households where the family members do not all have the same
target.
There are 43 households where the family members do not all have the same
target.
There are 42 households where the family members do not all have the same
target.
There are 41 households where the family members do not all have the same
target.
There are 40 households where the family members do not all have the same
target.
There are 39 households where the family members do not all have the same
target.
There are 38 households where the family members do not all have the same
target.
There are 37 households where the family members do not all have the same
target.
There are 36 households where the family members do not all have the same
target.
There are 35 households where the family members do not all have the same
target.
There are 34 households where the family members do not all have the same
target.
There are 33 households where the family members do not all have the same
target.
There are 32 households where the family members do not all have the same
target.
There are 31 households where the family members do not all have the same
target.
There are 30 households where the family members do not all have the same
target.
There are 29 households where the family members do not all have the same
target.
There are 28 households where the family members do not all have the same
target.
There are 27 households where the family members do not all have the same
target.
There are 26 households where the family members do not all have the same
target.
There are 25 households where the family members do not all have the same
target.
There are 24 households where the family members do not all have the same
target.
```

```
There are 23 households where the family members do not all have the same
target.
There are 22 households where the family members do not all have the same
target.
There are 21 households where the family members do not all have the same
target.
There are 20 households where the family members do not all have the same
target.
There are 19 households where the family members do not all have the same
target.
There are 18 households where the family members do not all have the same
target.
There are 17 households where the family members do not all have the same
target.
There are 16 households where the family members do not all have the same
target.
There are 15 households where the family members do not all have the same
target.
There are 14 households where the family members do not all have the same
target.
There are 13 households where the family members do not all have the same
target.
There are 12 households where the family members do not all have the same
target.
There are 11 households where the family members do not all have the same
target.
There are 10 households where the family members do not all have the same
target.
There are 9 households where the family members do not all have the same t
arget.
There are 8 households where the family members do not all have the same t
arget.
There are 7 households where the family members do not all have the same t
arget.
There are 6 households where the family members do not all have the same t
arget.
There are 5 households where the family members do not all have the same t
arget.
There are 4 households where the family members do not all have the same t
arget.
There are 3 households where the family members do not all have the same t
arget.
There are 2 households where the family members do not all have the same t
arget.
There are 1 households where the family members do not all have the same t
arget.
There are 0 households where the family members do not all have the same t
arget.
```

Lets check for any bias in the dataset

In [49]:

```
target_counts = heads['Target'].value_counts().sort_index()
target_counts
```

Out[49]:

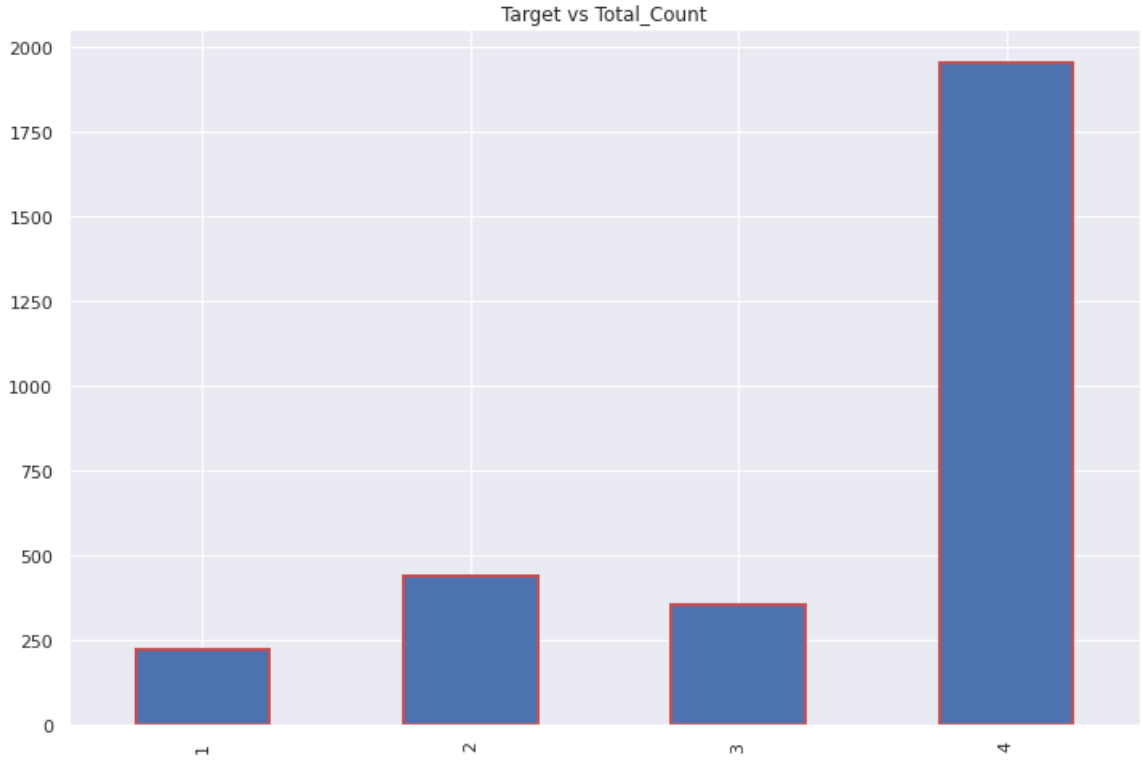
```
1     222
2     442
3     355
4    1954
Name: Target, dtype: int64
```

In [50]:

```
target_counts.plot.bar(figsize = (12, 8),linewidth = 2,edgecolor = 'r',title="Target vs Total_Count")
```

Out[50]:

```
<AxesSubplot:title={'center':'Target vs Total_Count'}>
```



In [51]:

```
#Lets remove them
print(df_income_train.shape)
cols=['SQBescolari', 'SQBage', 'SQBhogar_total', 'SQBedjefe',
'SQBhogar_nin', 'SQBovercrowding', 'SQBdependency', 'SQBmeaned', 'agesq']
for df in [df_income_train, df_income_test]:
    df.drop(columns = cols,inplace=True)
print(df_income_train.shape)
```

```
(9557, 143)
(9557, 134)
```

In [52]:

```
id_ = ['Id', 'idhogar', 'Target']

ind_bool = ['v18q', 'dis', 'male', 'female', 'estadocivil1', 'estadocivil2', 'estadocivil3',
            'estadocivil4', 'estadocivil5', 'estadocivil6', 'estadocivil7',
            'parentesco1', 'parentesco2', 'parentesco3', 'parentesco4', 'parentesco5',
            'parentesco6', 'parentesco7', 'parentesco8', 'parentesco9', 'parentesco10',
            'parentesco11', 'parentesco12', 'instlevel1', 'instlevel2', 'instlevel3',
            'instlevel4', 'instlevel5', 'instlevel6', 'instlevel7', 'instlevel8',
            'instlevel9', 'mobilephone']

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

hh_bool = ['hacdor', 'hacapo', 'v14a', 'refrig', 'paredblolad', 'paredzocalo',
    'paredpreb','pisocemento', 'pareddes', 'paredmad',
    'paredzinc', 'paredfibras', 'paredother', 'pisomoscer', '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',
    'tipovivi1', 'tipovivi2', 'tipovivi3', 'tipovivi4', 'tipovivi5',
    'computer', 'television', 'lugar1', 'lugar2', 'lugar3',
    'lugar4', 'lugar5', 'lugar6', 'area1', 'area2']

hh_ordered = [ 'rooms', 'r4h1', 'r4h2', 'r4h3', 'r4m1','r4m2','r4m3', 'r4t1', 'r4t2',
    'r4t3', 'v18q1', 'tamhog','tamviv','hhsize','hogar_nin',
    'hogar_adul','hogar_mayor','hogar_total', 'bedrooms', 'qmobilephone']

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

In [53]:

```
#Check for redundant household variables
heads = df_income_train.loc[df_income_train['parentesco1'] == 1, :]
heads = heads[id_ + hh_bool + hh_cont + hh_ordered]
heads.shape
```

Out[53]:

```
(2973, 98)
```

In [54]:

```
# 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[54]:

```
['coopele', 'area2', 'tamhog', 'hhsize', 'hogar_total']
```

In [55]:

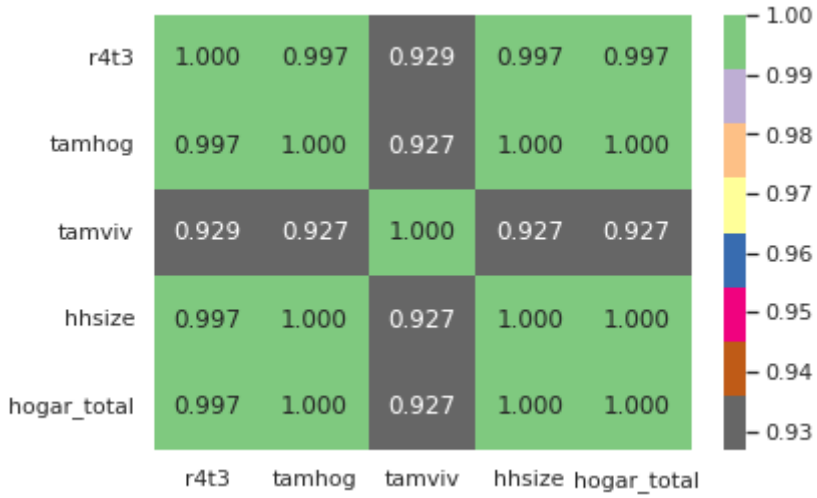
```
corr_matrix.loc[corr_matrix['tamhog'].abs() > 0.9, corr_matrix['tamhog'].abs() > 0.9]
```

Out[55]:

	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 [56]:

```
sns.heatmap(corr_matrix.loc[corr_matrix['tamhog'].abs() > 0.9, corr_matrix['tamhog'].ab
s() > 0.9],
annot=True, cmap = plt.cm.Accent_r, fmt='.3f');
```



In [57]:

```
cols=['tamhog', 'hogar_total', 'r4t3']
for df in [df_income_train, df_income_test]:
    df.drop(columns = cols,inplace=True)
df_income_train.shape
```

Out[57]:

(9557, 131)

In [58]:

```
#Check for redundant Individual variables
ind = df_income_train[id_ + ind_bool + ind_ordered]
ind.shape
```

Out[58]:

(9557, 39)

In [59]:

```
# Create correlation matrix
corr_matrix = ind.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[59]:

['female']

In [60]:

```
# We can remove the male flag.
for df in [df_income_train, df_income_test]:
    df.drop(columns = 'male',inplace=True)
df_income_train.shape
```

Out[60]:

(9557, 130)

In [61]:

```
#lets check area1 and area2 also
for df in [df_income_train, df_income_test]:
    df.drop(columns = 'area2',inplace=True)
df_income_train.shape
```

Out[61]:

(9557, 129)

In [62]:

```
#Finally delete 'Id', 'idhogar'
cols=['Id','idhogar']
for df in [df_income_train, df_income_test]:
    df.drop(columns = cols,inplace=True)
df_income_train.shape
```

Out[62]:

```
(9557, 127)
```

Predicting the accuracy using random forest classifier.

In [63]:

```
x_features=df_income_train.iloc[:,0:-1]
y_features=df_income_train.iloc[:,-1]
print(x_features.shape)
print(y_features.shape)
```

```
(9557, 126)
(9557,)
```

In [64]:

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score,confusion_matrix,f1_score,classification_rep
ort

x_train,x_test,y_train,y_test=train_test_split(x_features,y_features,test_size=0.2,rand
om_state=1)
rmclassifier = RandomForestClassifier()
```

In [65]:

```
rmclassifier.fit(x_train,y_train)
```

Out[65]:

```
RandomForestClassifier()
```

In [67]:

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
    max_depth=None, max_features='auto', max_leaf_nodes=None,
    min_impurity_decrease=0.0,min_samples_leaf=1, min_samples_split=2,
    min_weight_fraction_leaf=0.0, n_estimators=10,
    n_jobs=None, oob_score=False, random_state=None,
    verbose=0, warm_start=False)
```

Out[67]:

```
RandomForestClassifier(n_estimators=10)
```

In [70]:

```
y_predict = rmclassifier.predict(x_test)
```

In [71]:

```
print(accuracy_score(y_test,y_predict))
print(confusion_matrix(y_test,y_predict))
print(classification_report(y_test,y_predict))
```

```
0.952928870292887
[[ 136    1    0   20]
 [   1  287    1   28]
 [   0    1  197   35]
 [   0    2    1 1202]]
              precision    recall  f1-score   support

           1       0.99      0.87      0.93       157
           2       0.99      0.91      0.94       317
           3       0.99      0.85      0.91       233
           4       0.94      1.00      0.97      1205

    accuracy                           0.95      1912
   macro avg       0.98      0.90      0.94      1912
weighted avg       0.96      0.95      0.95      1912
```

In [72]:

```
y_predict_testdata = rmclassifier.predict(df_income_test)
```

In [73]:

```
y_predict_testdata
```

Out[73]:

```
array([4, 4, 4, ..., 4, 4, 4])
```

Checking the accuracy using random forest.

In [75]:

```
from sklearn.model_selection import KFold,cross_val_score
```

In [76]:

```
seed=7
kfold=KFold(n_splits=5,random_state=seed,shuffle=True)
rmclassifier=RandomForestClassifier(random_state=10,n_jobs = -1)
print(cross_val_score(rmclassifier,x_features,y_features,cv=kfold,scoring='accuracy'))
results=cross_val_score(rmclassifier,x_features,y_features,cv=kfold,scoring='accuracy')
print(results.mean()*100)
```

```
[0.94246862 0.94979079 0.94557823 0.94243851 0.94976452]
94.60081361157272
```

Checking the score using 100 trees

In [77]:

```
num_trees= 100
rmclassifier=RandomForestClassifier(n_estimators=100, random_state=10,n_jobs = -1)
print(cross_val_score(rmclassifier,x_features,y_features,cv=kfold,scoring='accuracy'))
results=cross_val_score(rmclassifier,x_features,y_features,cv=kfold,scoring='accuracy')
print(results.mean()*100)
```

```
[0.94246862 0.94979079 0.94557823 0.94243851 0.94976452]
94.60081361157272
```

Looking at the accuracy score, RandomForestClassifier with cross validation has the highest accuracy score of 94.60%

In [84]:

```
rmclassifier.fit(x_features,y_features)
labels = list(x_features)
feature_importances = pd.DataFrame({'feature': labels, 'importance': rmclassifier.featu
re_importances_})
feature_importances=feature_importances[feature_importances.importance>0.015]
feature_importances.head()
```

Out[84]:

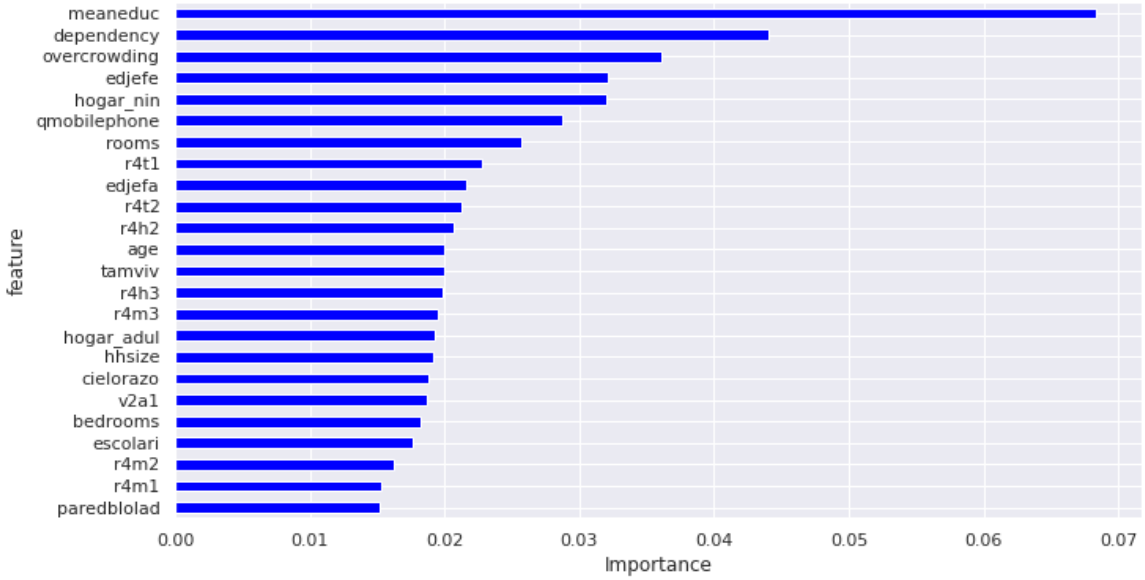
	feature	importance
0	v2a1	0.018653
2	rooms	0.025719
9	r4h2	0.020706
10	r4h3	0.019808
11	r4m1	0.015271

In [85]:

```
feature_importances.sort_values(by=['importance'], ascending=True, inplace=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_import
ances.positive.map({True: 'blue', False: 'red'}))
plt.xlabel('Importance')
```

Out[85]:

```
Text(0.5, 0, 'Importance')
```



In [ ]: