Income Qualification(ML)

January 28, 2023

0.1 Load the Dataset

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
sns.set()
```

0.2 Exploring the Dataset

[2]:
```
df_income_train=pd.read_csv('trainbook.csv')
df_income_test=pd.read_csv('testbook.csv')
```

[3]:
```
df_income_train.head()
```

[3]:

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

	r4h1	…	SQBescolari	SQBage	SQBhogar_total	SQBedjefe	SQBhogar_nin	\
0	0	…	100	1849	1	100	0	
1	0	…	144	4489	1	144	0	
2	0	…	121	8464	1	0	0	
3	0	…	81	289	16	121	4	
4	0	…	121	1369	16	121	4	

	SQBovercrowding	SQBdependency	SQBmeaned	agesq	Target
0	1.000000	0.0	100.0	1849	4
1	1.000000	64.0	144.0	4489	4
2	0.250000	64.0	121.0	8464	4
3	1.777778	1.0	121.0	289	4
4	1.777778	1.0	121.0	1369	4

```
[5 rows x 143 columns]
```

[4]:
```
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
```

[5]:
```
df_income_test.head()
```

[5]:

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

	r4h1	…	age	SQBescolari	SQBage	SQBhogar_total	SQBedjefe	\
0	1	…	4	0	16	9	0	
1	1	…	41	256	1681	9	0	
2	1	…	41	289	1681	9	0	
3	0	…	59	256	3481	1	256	
4	0	…	18	121	324	1	0	

	SQBhogar_nin	SQBovercrowding	SQBdependency	SQBmeaned	agesq
0	1	2.25	0.25	272.25	16
1	1	2.25	0.25	272.25	1681
2	1	2.25	0.25	272.25	1681
3	0	1.00	0.00	256.00	3481
4	1	0.25	64.00	NaN	324

```
[5 rows x 142 columns]
```

[6]:
```
df_income_train.select_dtypes('int64').head()
```

[6]:

	hacdor	rooms	hacapo	v14a	refrig	v18q	r4h1	r4h2	r4h3	r4m1	…	\
0	0	3	0	1	1	0	0	1	1	0	…	
1	0	4	0	1	1	1	0	1	1	0	…	
2	0	8	0	1	1	0	0	0	0	0	…	
3	0	5	0	1	1	1	0	2	2	1	…	
4	0	5	0	1	1	1	0	2	2	1	…	

	area1	area2	age	SQBescolari	SQBage	SQBhogar_total	SQBedjefe	\
0	1	0	43	100	1849	1	100	

```
1      1      0   67          144    4489               1         144
2      1      0   92          121    8464               1           0
3      1      0   17           81     289              16         121
4      1      0   37          121    1369              16         121

   SQBhogar_nin  agesq  Target
0             0   1849       4
1             0   4489       4
2             0   8464       4
3             4    289       4
4             4   1369       4

[5 rows x 130 columns]
```

[7]:
```
# Find columns with null values
null_counts=df_income_train.select_dtypes('int64').isnull().sum()
null_counts[null_counts > 0]
```

[7]:
```
Series([], dtype: int64)
```

[8]:
```
df_income_train.select_dtypes('float64').head()
```

[8]:
```
        v2a1  v18q1  rez_esc  meaneduc  overcrowding  SQBovercrowding  \
0  190000.0    NaN      NaN      10.0      1.000000         1.000000
1  135000.0    1.0      NaN      12.0      1.000000         1.000000
2       NaN    NaN      NaN      11.0      0.500000         0.250000
3  180000.0    1.0      1.0      11.0      1.333333         1.777778
4  180000.0    1.0      NaN      11.0      1.333333         1.777778

   SQBdependency  SQBmeaned
0            0.0      100.0
1           64.0      144.0
2           64.0      121.0
3            1.0      121.0
4            1.0      121.0
```

[9]:
```
# Find columns with null values
null_counts=df_income_train.select_dtypes('float64').isnull().sum()
null_counts[null_counts > 0]
```

[9]:
```
v2a1          6860
v18q1         7342
rez_esc       7928
meaneduc         5
SQBmeaned        5
dtype: int64
```

[10]:
```
df_income_train.select_dtypes('object').head()
```

[10]:

	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

[11]:
```
# Find columns with null values
null_counts=df_income_train.select_dtypes('object').isnull().sum()
null_counts[null_counts > 0]
```

[11]:
```
Series([], dtype: int64)
```

0.3 Data Cleaning

[12]:
```
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()
```

[12]:

	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

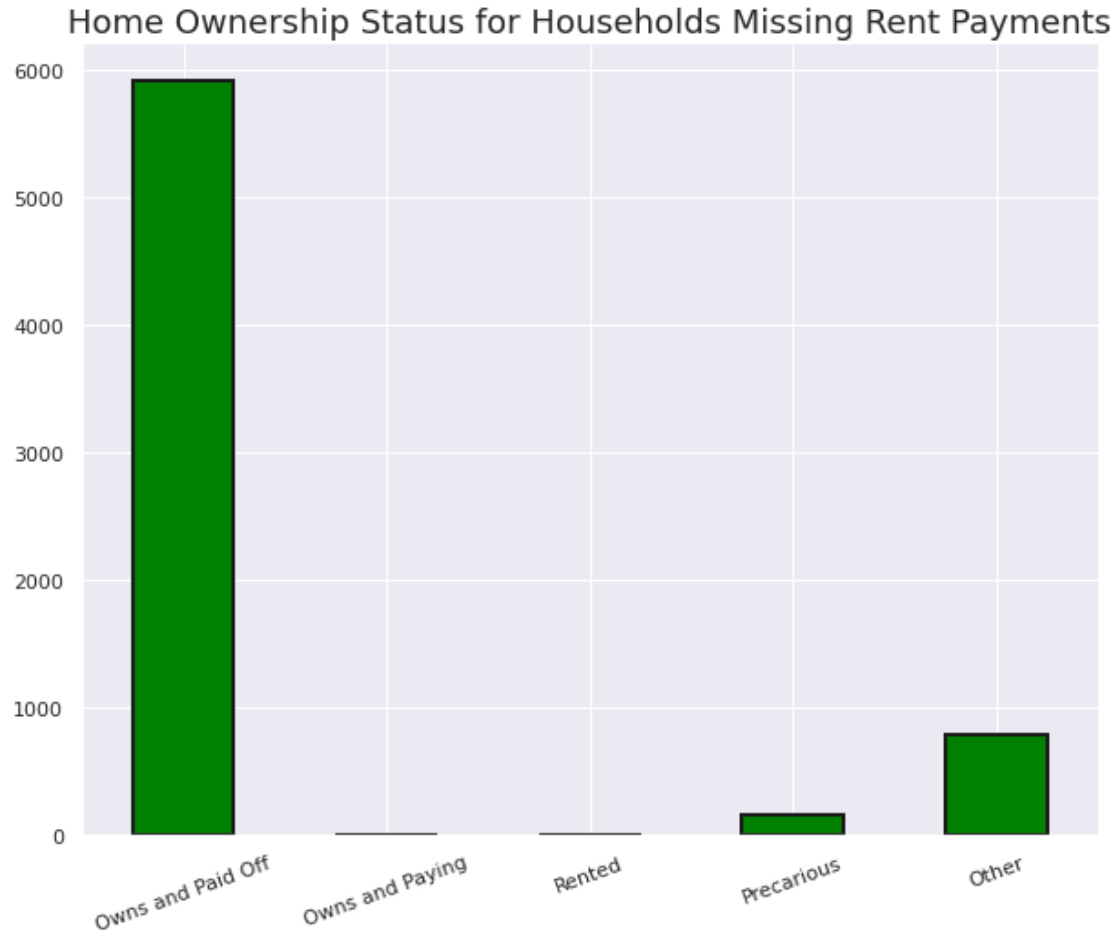
[13]:
```
data=df_income_train[df_income_train['v2a1'].isnull()].head()
columns=['tipovivi1', 'tipovivi2', 'tipovivi3', 'tipovivi4', 'tipovivi5']
data[columns]
```

[13]:

	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

[14]:
```
# variable 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(figsize=(10,8),
color='green',
                                                                              ␣
 ↪    edgecolor=
'k', linewidth=2);
plt.xticks([0, 1, 2, 3, 4],
           ['Owns and Paid Off','Owns and␣
 ↪Paying','Rented','Precarious','Other'],
       rotation = 20)
plt.title('Home Ownership Status for Households Missing Rent Payments',␣
 ↪size=18);
```



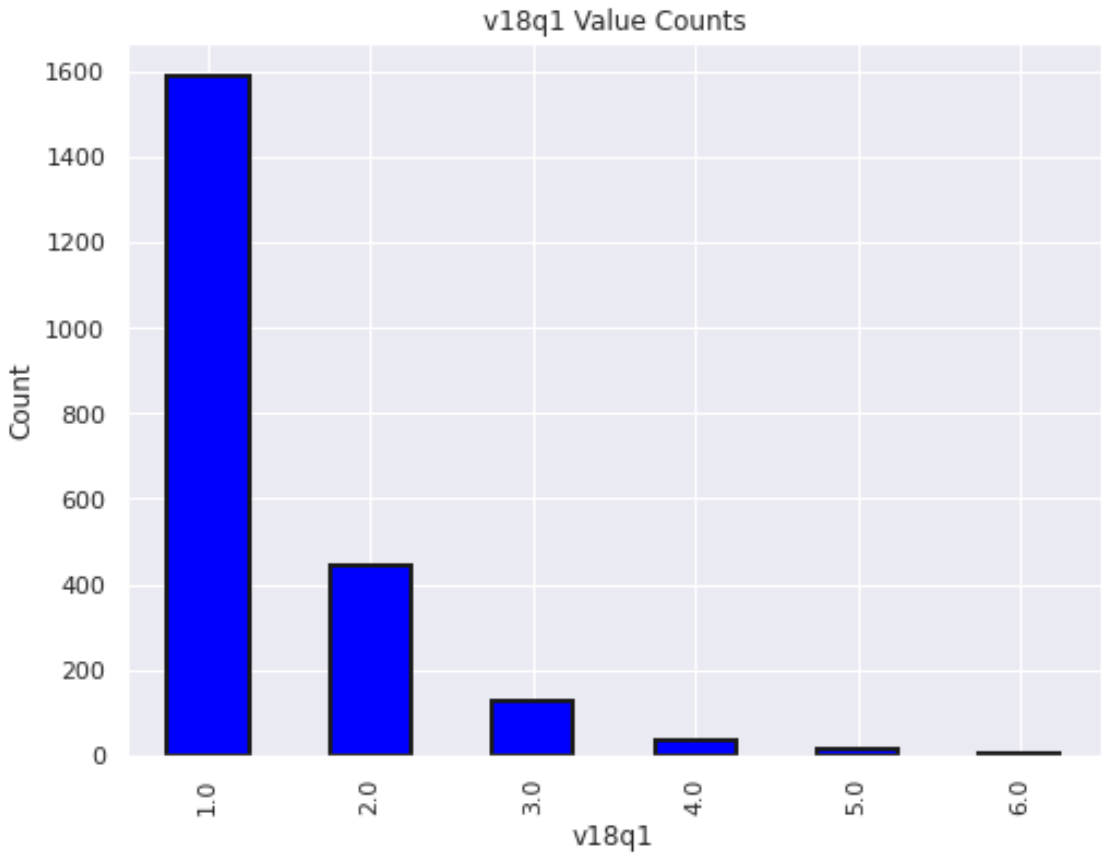
[15]:
```
# Add 0 for all null values
for df in [df_income_train, df_income_test]:
    df['v2a1'].fillna(value=0, inplace=True)

df_income_train[['v2a1']].isnull().sum()
```

[15]:
```
v2a1    0
dtype: int64
```

[16]:
```
# Heads of household
heads=df_income_train.loc[df_income_train['parentesco1']==1].copy()
heads.groupby('v18q')['v18q1'].apply(lambda x : x.isnull().sum())
```

[16]:
```
v18q
0    2318
1       0
Name: v18q1, dtype: int64
```

[17]:
```
plt.figure(figsize=(8,6))
col='v18q1'
df_income_train[col].value_counts().sort_index().plot.bar(color = 'blue',
                                                          edgecolor='k',
                                                          linewidth=2)
plt.xlabel(f'{col}'); plt.title(f'{col} Value Counts'); plt.ylabel('Count')
plt.show();
```



[18]:
```
for df in [df_income_train, df_income_test]:
    df['v18q1'].fillna(value=0, inplace=True)

df_income_train[['v18q1']].isnull().sum()
```

[18]:
```
v18q1    0
dtype: int64
```

[19]:
```
# data with no null values
df_income_train[df_income_train['rez_esc'].notnull()]['age'].describe()
```

[19]:
```
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
```

[20]:
```
df_income_train.loc[df_income_train['rez_esc'].isnull()]['age'].describe()
```

[20]:
```
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
```

[21]:
```
df_income_train.loc[df_income_train['rez_esc'].isnull() &
                    ((df_income_train['age']>7) & (df_income_train
['age'] <17))]['age'].describe()
```

[21]:
```
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
```

[22]:
```
df_income_train[(df_income_train['age']==10) & df_income_train['rez_esc'].
 ↪isnull()].head()
df_income_train[(df_income_train['Id']=='ID_f012e4242')].head()
```

[22]:
```
                Id      v2a1  hacdor  rooms  hacapo  v14a  refrig  v18q  \
2514  ID_f012e4242  160000.0       0      6       0     1       1     1

      v18q1  r4h1  …  SQBescolari  SQBage  SQBhogar_total  SQBedjefe  \
2514    1.0     0  …            0     100               9        121

      SQBhogar_nin  SQBovercrowding  SQBdependency  SQBmeaned  agesq  Target
2514             1             2.25           0.25     182.25    100       4

[1 rows x 143 columns]
```

[23]:
```
for df in [df_income_train, df_income_test]:
    df['rez_esc'].fillna(value=0, inplace=True)
df_income_train[['rez_esc']].isnull().sum()
```

[23]:
```
rez_esc    0
dtype: int64
```

[24]:
```
data=df_income_train[df_income_train['meaneduc'].isnull()].head()

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

[24]:

	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

[25]:
```
# fix the data
for df in [df_income_train, df_income_test]:
    df['meaneduc'].fillna(value=0, inplace=True)
df_income_train[['meaneduc']].isnull().sum()
```

[25]:
```
meaneduc    0
dtype: int64
```

[26]:
```
data=df_income_train[df_income_train['SQBmeaned'].isnull()].head()

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

[26]:

	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

[27]:
```
for df in [df_income_train, df_income_test]:
    df['SQBmeaned'].fillna(value=0, inplace=True)
df_income_train[['SQBmeaned']].isnull().sum()
```

[27]:
```
SQBmeaned    0
dtype: int64
```

[28]:
```
null_counts=df_income_train.isnull().sum()
null_counts[null_counts > 0].sort_values(ascending=False)
```

[28]:
```
Series([], dtype: int64)
```

[29]:
```
for df in [df_income_train, df_income_test]:
    df['v18q1'].fillna(value=0, inplace=True)
df_income_train[['v18q1']].isnull().sum()
```

[29]:
```
v18q1    0
dtype: int64
```

[30]:
```
for df in [df_income_train, df_income_test]:
    df['v2a1'].fillna(value=0, inplace=True)
df_income_train[['v2a1']].isnull().sum()
```

[30]:
```
v2a1    0
dtype: int64
```

[31]:
```
null_counts=df_income_train.isnull().sum()
null_counts[null_counts > 0].sort_values(ascending=False)
```

[31]:
```
Series([], dtype: int64)
```

[32]:
```
# 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 equal
not_equal=all_equal[all_equal !=True]
print('There are {} households where all the family members do not have the␣
 ↪same target.'.format(len(not_equal)))
```

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

[33]:
```
# Lets check one household
df_income_train[df_income_train['idhogar']==not_equal.index[0]][['idhogar',␣
 ↪'parentesco1', 'Target' ]]
```

[33]:

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

[34]:
```
# 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.
```

[35]:
```
households_no_head_equal=households_no_head.groupby('idhogar')['Target'].
 ↪apply(lambda x : x.nunique()==1)
print('{} Households with no head heave different Target value.'.
 ↪format(sum(households_no_head_equal==False)))
```

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

[36]:
```
for household in not_equal.index:
    # Find correct label for the head of household
    true_target=int(df_income_train[(df_income_train['idhogar']==household) &␣
 ↪(df_income_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_target

# Groupby thr 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 all the family members do not have the␣
 ↪same target.'.format(len(not_equal)))
```

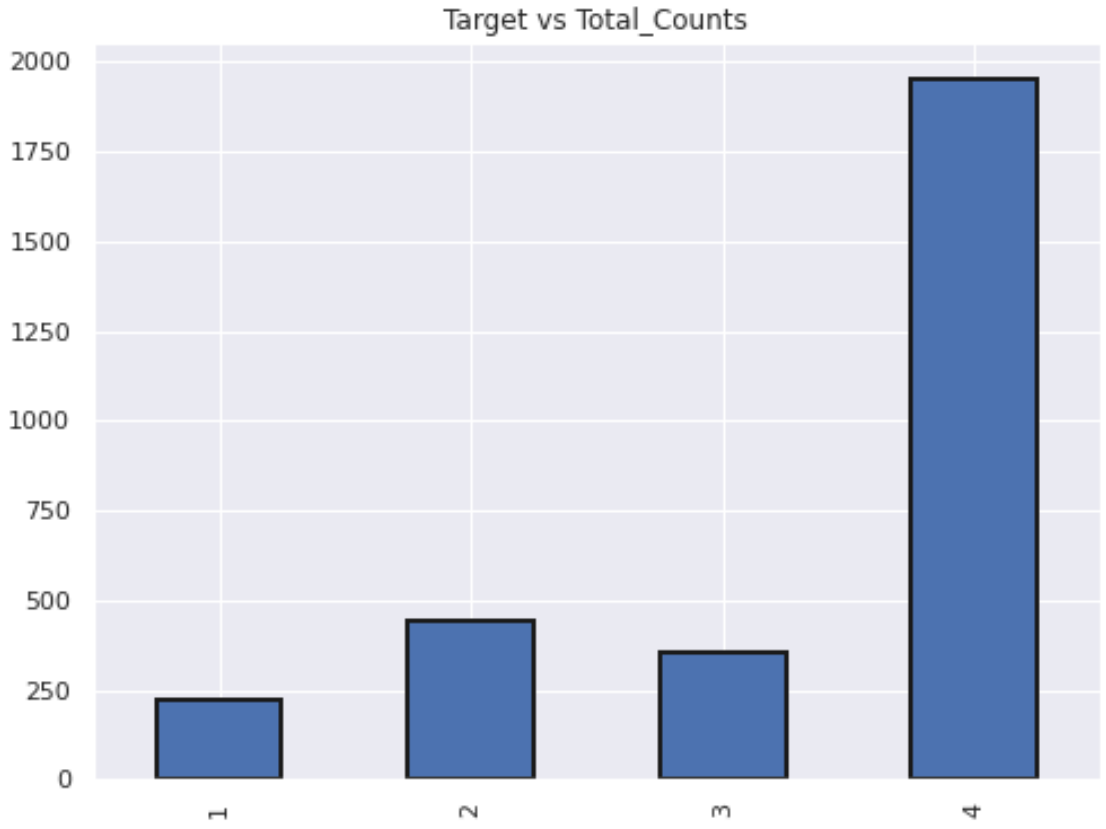
```
There are 84 households where all the family members do not have the same
target.
```

[37]:
```
target_counts=heads['Target'].value_counts().sort_index()
target_counts
```

[37]:
```
1     222
2     442
3     355
4    1954
Name: Target, dtype: int64
```

[38]:
```
target_counts.plot.bar(figsize=(8,6),linewidth=2,edgecolor='k',
                       title='Target vs Total_Counts')
```

```
plt.show()
```



[39]:
```
# 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)
```

[40]:
```
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']
```

[41]:
```
# 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
```

[41]:
```
(2973, 98)
```

[42]:
```
# 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
```

```
/usr/local/lib/python3.7/site-packages/ipykernel_launcher.py:5:
DeprecationWarning: `np.bool` is a deprecated alias for the builtin `bool`. To
silence this warning, use `bool` by itself. Doing this will not modify any
behavior and is safe. If you specifically wanted the numpy scalar type, use
`np.bool_` here.
Deprecated in NumPy 1.20; for more details and guidance:
https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  """
```

[42]:
```
['coopele', 'area2', 'tamhog', 'hhsize', 'hogar_total']
```

[43]:
```
['coopele', 'area2', 'tamhog', 'hhsize', 'hogar_total']
```

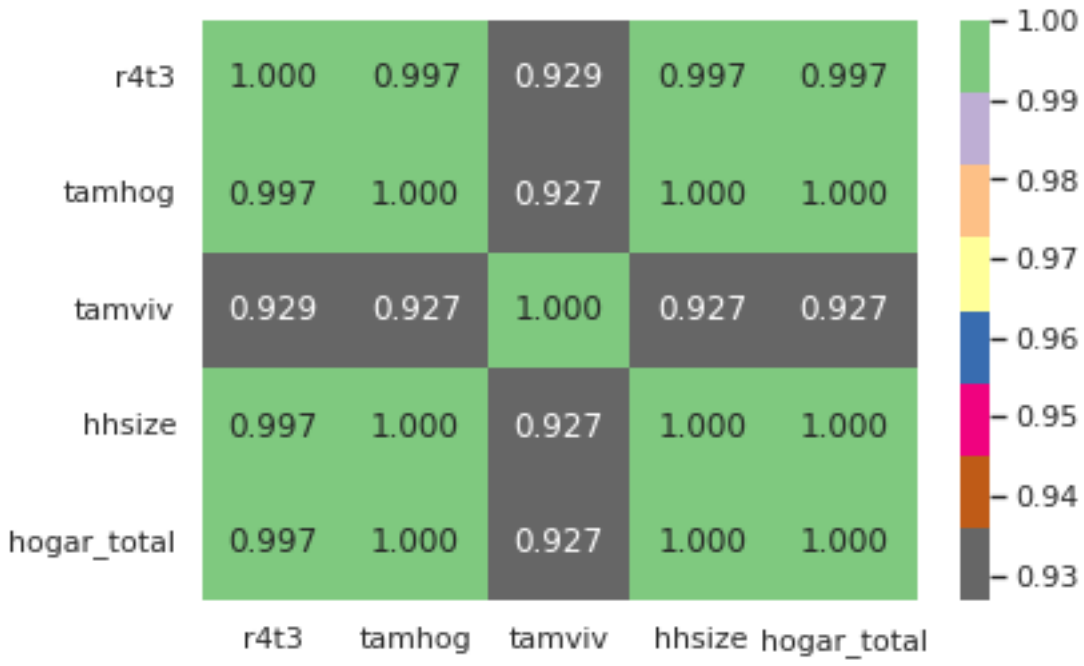
[43]:
```
['coopele', 'area2', 'tamhog', 'hhsize', 'hogar_total']
```

[44]:
```
corr_matrix.loc[corr_matrix['tamhog'].abs() > 0.9, corr_matrix['tamhog'].abs()␣
↪> 0.9]
```

[44]:

	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

[45]:
```
sns.heatmap(corr_matrix.loc[corr_matrix['tamhog'].abs() > 0.9,␣
↪corr_matrix['tamhog'].abs() > 0.9],
            annot=True, cmap = plt.cm.Accent_r, fmt='.3f');
```



[46]:
```
cols=['tamhog', 'hogar_total', 'r4t3']
for df in [df_income_train, df_income_test]:
    df.drop(columns = cols,inplace=True)

df_income_train.shape
```

[46]: (9557, 131)

[47]:
```
#Check for redundant Individual variables
ind = df_income_train[id_ + ind_bool + ind_ordered]
ind.shape
```

[47]: (9557, 39)

[48]:
```
# 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
```

```
/usr/local/lib/python3.7/site-packages/ipykernel_launcher.py:5:
DeprecationWarning: `np.bool` is a deprecated alias for the builtin `bool`. To
silence this warning, use `bool` by itself. Doing this will not modify any
behavior and is safe. If you specifically wanted the numpy scalar type, use
`np.bool_` here.
Deprecated in NumPy 1.20; for more details and guidance:
https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  """
```

[48]:
```
['female']
```

[49]:
```
# This is simply the opposite of male! 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
```

[49]:
```
(9557, 130)
```

[50]:
```
for df in [df_income_train, df_income_test]:
    df.drop(columns = 'area2',inplace=True)

df_income_train.shape
```

[50]:
```
(9557, 129)
```

[51]:
```
#Finally lets 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
```

[51]:
```
(9557, 127)
```

0.4 Predict the accuracy using Random Forest Classifier

[52]:
```
df_income_train.iloc[:,0:-1]
```

[52]:

	v2a1	hacdor	rooms	hacapo	v14a	refrig	v18q	v18q1	r4h1	r4h2	\
0	190000.0	0	3	0	1	1	0	0.0	0	1	
1	135000.0	0	4	0	1	1	1	1.0	0	1	
2	0.0	0	8	0	1	1	0	0.0	0	0	
3	180000.0	0	5	0	1	1	1	1.0	0	2	
4	180000.0	0	5	0	1	1	1	1.0	0	2	
…	…	…	…	…	…	…	…	…	…	…	…

```
9552   80000.0       0      6       0     1       1     0    0.0     0     2
9553   80000.0       0      6       0     1       1     0    0.0     0     2
9554   80000.0       0      6       0     1       1     0    0.0     0     2
9555   80000.0       0      6       0     1       1     0    0.0     0     2
9556   80000.0       0      6       0     1       1     0    0.0     0     2

      …  mobilephone  qmobilephone  lugar1  lugar2  lugar3  lugar4  lugar5  \
0     …             1             1       1       0       0       0       0
1     …             1             1       1       0       0       0       0
2     …             0             0       1       0       0       0       0
3     …             1             3       1       0       0       0       0
4     …             1             3       1       0       0       0       0
…   …           …           …     …     …     …     …     …
9552  …             1             3       0       0       0       0       0
9553  …             1             3       0       0       0       0       0
9554  …             1             3       0       0       0       0       0
9555  …             1             3       0       0       0       0       0
9556  …             1             3       0       0       0       0       0

      lugar6  area1  age
0          0      1   43
1          0      1   67
2          0      1   92
3          0      1   17
4          0      1   37
…       …     …  …
9552       1      0   46
9553       1      0    2
9554       1      0   50
9555       1      0   26
9556       1      0   21

[9557 rows x 126 columns]
```

[53]:
```
df_income_train.iloc[:,-1]
```

[53]:
```
0       4
1       4
2       4
3       4
4       4
       ..
9552    2
9553    2
9554    2
9555    2
9556    2
```

```
Name: Target, Length: 9557, dtype: int64
```

[54]:
```
x_features=df_income_train.iloc[:,0:-1] # feature without target
y_features=df_income_train.iloc[:,-1] # only target
print(x_features.shape)
print(y_features.shape)
```

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

[55]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import␣
 ↪accuracy_score,confusion_matrix,f1_score,classification_report

x_train,x_test,y_train,y_test=train_test_split(x_features,y_features,test_size=0.
 ↪2,random_state=1)
rmclassifier = RandomForestClassifier()
```

[57]:
```
rmclassifier.fit(x_train,y_train)
```

[57]:
```
RandomForestClassifier()
```

[58]:
```
y_predict = rmclassifier.predict(x_test)
```

[59]:
```
print(accuracy_score(y_test,y_predict))
print(confusion_matrix(y_test,y_predict))
print(classification_report(y_test,y_predict))
```

```
0.9398535564853556
[[ 117    8    0   22]
 [   5  296    4   28]
 [   0    5  184   40]
 [   0    2    1 1200]]
              precision    recall  f1-score   support

           1       0.96      0.80      0.87       147
           2       0.95      0.89      0.92       333
           3       0.97      0.80      0.88       229
           4       0.93      1.00      0.96      1203

    accuracy                           0.94      1912
   macro avg       0.95      0.87      0.91      1912
weighted avg       0.94      0.94      0.94      1912
```

[60]:
```
y_predict_testdata = rmclassifier.predict(df_income_test)
```

[61]:
```
y_predict_testdata
```

[61]:
```
array([4, 4, 4, …, 4, 4, 4])
```

0.5 Check the accuracy using Random Forest with cross validation

[62]:
```
from sklearn.model_selection import KFold,cross_val_score
```

[63]:
```
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)
```

```
[0.92991632 0.92991632 0.92726321 0.92255364 0.93301936]
92.85337694781808
```

[64]:
```
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)
```

```
[0.92991632 0.92991632 0.92726321 0.92255364 0.93301936]
92.85337694781808
```

[65]:
```
rmclassifier.fit(x_features,y_features)
labels = list(x_features)
feature_importances = pd.DataFrame({'feature': labels, 'importance':␣
↪rmclassifier.feature_importances_})
feature_importances=feature_importances[feature_importances.importance>0.015]
feature_importances.head()
```

[65]:

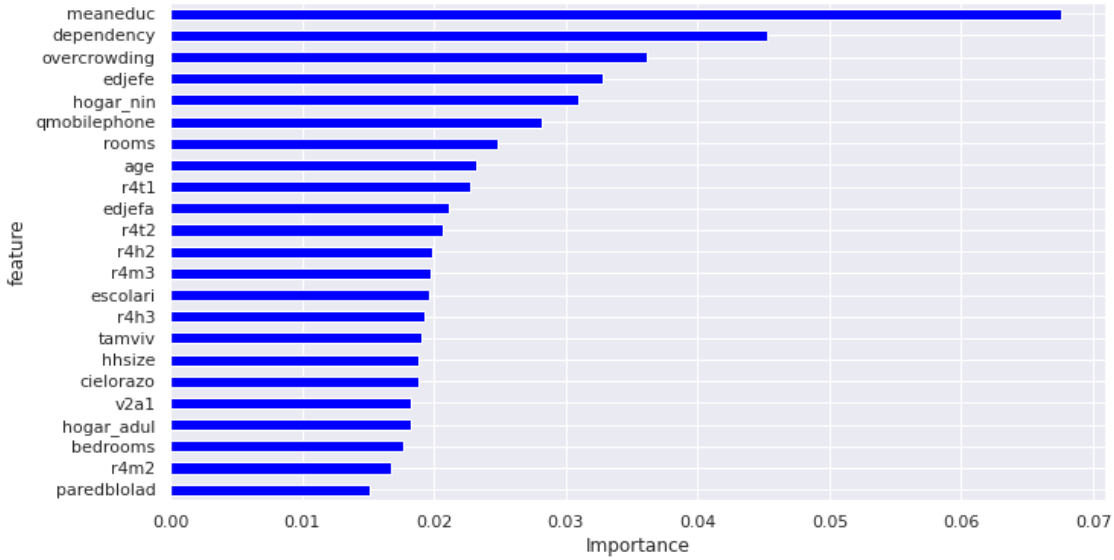
	feature	importance
0	v2a1	0.018302
2	rooms	0.024884
9	r4h2	0.019924
10	r4h3	0.019244
12	r4m2	0.016720

[66]:
```
y_predict_testdata = rmclassifier.predict(df_income_test)
y_predict_testdata
```

[66]:
```
array([4, 4, 4, …, 4, 4, 4])
```

[67]:
```
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_importances.positive.map({True: 'blue', False: 'red'}))
plt.xlabel('Importance')
```

[67]:
```
Text(0.5, 0, 'Importance')
```



[ ]: