In [0]:
```
%load_ext autoreload
%autoreload 2

%matplotlib inline
```

In [3]:
```
!ls Machine-Learning--Projects/Projects/'Projects for Submission'/'Project 2 - Income Qualification'
```

```
'Dataset for the project.zip'  'Income Qualification.txt'
```

In [4]:
```
!ls data
```

```
test.csv  train.csv
```

In [0]:
```
import pandas as pd
import numpy as np

data=pd.read_csv('data/train.csv')
```

In [7]:
```
data.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 [8]:
```
data.columns[data.dtypes==object]
```

Out[8]:
```
Index(['Id', 'idhogar', 'dependency', 'edjefe', 'edjefa'], dtype='object')
```

In [9]:
```
data['dependency'].unique()
```

Out[9]:
```
array(['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'], dtype=object)
```

In [10]:
```
data[(data['dependency']=='no') & (data['SQBdependency']!=0)]
```

Out[10]:

Id	v2a1	hacdor	rooms	hacapo	v14a	refrig	v18q	v18q1	r4h1	r4h2	r4h3	r4m1	r4m2	r4m3	r4t

0 rows × 143 columns

In [11]:
```
data[(data['dependency']=='yes') & (data['SQBdependency']!=1)]
```

Out[11]:

Id	v2a1	hacdor	rooms	hacapo	v14a	refrig	v18q	v18q1	r4h1	r4h2	r4h3	r4m1	r4m2	r4m3	r4t

0 rows × 143 columns

In [12]:
```
data[(data['dependency']=='3') & (data['SQBdependency']!=9)]
```

Out[12]:

Id	v2a1	hacdor	rooms	hacapo	v14a	refrig	v18q	v18q1	r4h1	r4h2	r4h3	r4m1	r4m2	r4m3	r4t

0 rows × 143 columns

In [0]:
```
data['dependency']=np.sqrt(data['SQBdependency'])
```

In [67]:
```
data['edjefe'].unique()
```

Out[67]:
```
array(['10', '12', 'no', '11', '9', '15', '4', '6', '8', '17', '7', '16',
       '14', '5', '21', '2', '19', 'yes', '3', '18', '13', '20'],
      dtype=object)
```

In [14]:
```
data['edjefa'].unique()
```

Out[14]:
```
array(['no', '11', '4', '10', '9', '15', '7', '14', '13', '8', '17', '6',
       '5', '3', '16', '19', 'yes', '21', '12', '2', '20', '18'],
      dtype=object)
```

In [15]:
```
data['SQBedjefe'].unique()
```

Out[15]:
```
array([100, 144,   0, 121,  81, 225,  16,  36,  64, 289,  49, 256, 196,
        25, 441,   4, 361,   1,   9, 324, 169, 400])
```

In [16]:
```
data[['edjefe', 'edjefa', 'SQBedjefe']][:20]
```

Out[16]:

	edjefe	edjefa	SQBedjefe
0	10	no	100
1	12	no	144
2	no	11	0
3	11	no	121
4	11	no	121
5	11	no	121
6	11	no	121
7	9	no	81
8	9	no	81
9	9	no	81
10	9	no	81
11	no	11	0
12	no	11	0
13	no	4	0
14	no	4	0
15	no	10	0
16	no	10	0
17	no	10	0
18	no	10	0
19	15	no	225

In [17]:
```
data[['edjefe', 'edjefa', 'SQBedjefe']][data['edjefe']=='yes']
```

Out[17]:

	edjefe	edjefa	SQBedjefe
601	yes	no	1
602	yes	no	1
2392	yes	no	1
2393	yes	no	1
2394	yes	no	1
2422	yes	no	1
2423	yes	no	1
2424	yes	no	1
2829	yes	no	1
2830	yes	no	1
2831	yes	no	1
3015	yes	no	1
3016	yes	no	1
3017	yes	no	1
3741	yes	no	1
3742	yes	no	1
3743	yes	no	1
3744	yes	no	1
3745	yes	no	1
4490	yes	no	1
4491	yes	no	1
4492	yes	no	1
4493	yes	no	1
4494	yes	no	1
4903	yes	no	1
4904	yes	no	1
4905	yes	no	1
4979	yes	no	1
4980	yes	no	1
4981	yes	no	1
...	...	...	...
8385	yes	no	1
8585	yes	no	1
8903	yes	no	1
8904	yes	no	1
8905	yes	no	1
8951	yes	no	1
8952	yes	no	1
8953	yes	no	1

	edjefe	edjefa	SQBedjefe
8954	yes	no	1
8955	yes	no	1
8956	yes	no	1
8957	yes	no	1
9040	yes	no	1
9041	yes	no	1
9042	yes	no	1
9043	yes	no	1
9044	yes	no	1
9045	yes	no	1
9046	yes	no	1
9047	yes	no	1
9197	yes	no	1
9198	yes	no	1
9199	yes	no	1
9257	yes	no	1
9415	yes	no	1
9416	yes	no	1
9417	yes	no	1
9463	yes	no	1
9464	yes	no	1
9465	yes	no	1

123 rows × 3 columns

In [18]:
```
data[(data['edjefe']=='yes') & (data['edjefa']!='no')]
```

Out[18]:

	Id	v2a1	hacdor	rooms	hacapo	v14a	refrig	v18q	v18q1	r4h1	r4h2	r4h3	r4m1	r4m2	r4m3	r4t

0 rows × 143 columns

In [19]:
```
data[(data['edjefa']=='yes') & (data['parentesco1']==1)][['edjefe', 'edjefa',
'parentesco1', 'escolari']]
```

Out[19]:

	edjefe	edjefa	parentesco1	escolari
432	no	yes	1	1
568	no	yes	1	1
595	no	yes	1	1
976	no	yes	1	1
1574	no	yes	1	1
2464	no	yes	1	1
2614	no	yes	1	1
3469	no	yes	1	1
3855	no	yes	1	1
3952	no	yes	1	1
4197	no	yes	1	1
4487	no	yes	1	1
4867	no	yes	1	1
4888	no	yes	1	1
4920	no	yes	1	1
6030	no	yes	1	1
6067	no	yes	1	1
6427	no	yes	1	1
7117	no	yes	1	1
7887	no	yes	1	1
8134	no	yes	1	1
8164	no	yes	1	1
8437	no	yes	1	1
8509	no	yes	1	1
9166	no	yes	1	1

In [20]:
```
data[data['edjefe']=='yes'][['edjefe', 'edjefa','age', 'escolari', 'parentesco
1','male', 'female', 'idhogar']]
```

Out[20]:

	edjefe	edjefa	age	escolari	parentesco1	male	female	idhogar
601	yes	no	81	1	1	1	0	3641ce2d1
602	yes	no	73	3	0	0	1	3641ce2d1
2392	yes	no	42	6	0	0	1	a7abf59cc
2393	yes	no	12	5	0	0	1	a7abf59cc
2394	yes	no	46	1	1	1	0	a7abf59cc
2422	yes	no	41	6	0	1	0	698318dd7
2423	yes	no	74	1	0	0	1	698318dd7
2424	yes	no	78	1	1	1	0	698318dd7
2829	yes	no	72	4	0	0	1	4559a5af0
2830	yes	no	76	1	1	1	0	4559a5af0
2831	yes	no	49	7	0	0	1	4559a5af0
3015	yes	no	10	4	0	1	0	78f517dc7
3016	yes	no	35	4	0	1	0	78f517dc7
3017	yes	no	73	1	1	1	0	78f517dc7
3741	yes	no	0	0	0	1	0	0b3d5369b
3742	yes	no	2	0	0	0	1	0b3d5369b
3743	yes	no	6	0	0	1	0	0b3d5369b
3744	yes	no	31	1	1	1	0	0b3d5369b
3745	yes	no	29	6	0	0	1	0b3d5369b
4490	yes	no	56	5	0	0	1	f2ed4871d
4491	yes	no	67	1	1	1	0	f2ed4871d
4492	yes	no	16	6	0	0	1	f2ed4871d
4493	yes	no	5	0	0	0	1	f2ed4871d
4494	yes	no	28	9	0	0	1	f2ed4871d
4903	yes	no	22	6	0	1	0	e26adc10d
4904	yes	no	50	6	0	0	1	e26adc10d
4905	yes	no	59	1	1	1	0	e26adc10d
4979	yes	no	17	11	0	1	0	0519a23f5
4980	yes	no	46	1	1	1	0	0519a23f5
4981	yes	no	52	6	0	0	1	0519a23f5
...	...	...	...	...	...	...	...	...
8385	yes	no	33	6	0	1	0	41f4d7a4b
8585	yes	no	78	1	1	1	0	12a23bffe
8903	yes	no	30	1	1	1	0	180d40332
8904	yes	no	24	6	0	0	1	180d40332
8905	yes	no	7	0	0	1	0	180d40332
8951	yes	no	13	6	0	0	1	d4bce9879
8952	yes	no	48	6	0	0	1	d4bce9879
8953	yes	no	17	6	0	0	1	d4bce9879

	edjefe	edjefa	age	escolari	parentesco1	male	female	idhogar
8954	yes	no	60	1	1	1	0	d4bce9879
8955	yes	no	28	9	0	1	0	d4bce9879
8956	yes	no	1	0	0	0	1	d4bce9879
8957	yes	no	26	11	0	0	1	d4bce9879
9040	yes	no	30	4	0	0	1	da2ecdcfd
9041	yes	no	13	6	0	1	0	da2ecdcfd
9042	yes	no	9	2	0	1	0	da2ecdcfd
9043	yes	no	15	6	0	1	0	da2ecdcfd
9044	yes	no	23	1	1	1	0	da2ecdcfd
9045	yes	no	1	0	0	1	0	da2ecdcfd
9046	yes	no	11	3	0	0	1	da2ecdcfd
9047	yes	no	4	0	0	0	1	da2ecdcfd
9197	yes	no	43	3	0	0	1	5fd40ddd3
9198	yes	no	23	6	0	1	0	5fd40ddd3
9199	yes	no	60	1	1	1	0	5fd40ddd3
9257	yes	no	75	1	1	1	0	7e0dc2f87
9415	yes	no	73	0	0	0	1	7dde74368
9416	yes	no	47	2	0	1	0	7dde74368
9417	yes	no	82	1	1	1	0	7dde74368
9463	yes	no	68	1	1	1	0	be108a783
9464	yes	no	57	3	0	0	1	be108a783
9465	yes	no	35	0	0	1	0	be108a783

123 rows × 8 columns

In [21]:
```
data[(data['edjefe']=='no') & (data['edjefa']=='no')][['edjefe', 'edjefa', 'age', 'escolari', 'female', 'male', 'Id', 'parentesco1', 'idhogar']]
```

Out[21]:

	edjefe	edjefa	age	escolari	female	male	Id	parentesco1	idhogar
86	no	no	89	0	1	0	ID_48c10ba80	1	2b1a06ddc
87	no	no	55	13	0	1	ID_1a283d51b	0	2b1a06ddc
121	no	no	21	14	1	0	ID_fdf18bbee	0	d9e9b591a
122	no	no	47	8	0	1	ID_cba836ed8	0	d9e9b591a
123	no	no	49	9	1	0	ID_6031cb88d	0	d9e9b591a
124	no	no	68	0	1	0	ID_0bc817cb2	1	d9e9b591a
125	no	no	21	14	1	0	ID_1cfb76926	0	d9e9b591a
166	no	no	6	0	1	0	ID_f435c5e46	0	71bec40bc
167	no	no	9	1	0	1	ID_a51262dae	0	71bec40bc
168	no	no	26	5	1	0	ID_7e6d23b28	0	71bec40bc
169	no	no	47	6	1	0	ID_4c409e16e	0	71bec40bc
170	no	no	4	0	1	0	ID_0704f4d9f	0	71bec40bc
171	no	no	60	0	0	1	ID_b0adca84e	1	71bec40bc
172	no	no	25	7	0	1	ID_0e4c312aa	0	71bec40bc
416	no	no	34	0	0	1	ID_2399e1e3e	1	fd22b4607
417	no	no	34	11	1	0	ID_301d72522	0	fd22b4607
418	no	no	7	0	1	0	ID_d643847f2	0	fd22b4607
419	no	no	14	7	1	0	ID_02a405b3b	0	fd22b4607
498	no	no	36	0	1	0	ID_460e4f901	1	7bf86807a
499	no	no	4	0	0	1	ID_0701248a7	0	7bf86807a
500	no	no	11	5	0	1	ID_a643ace5b	0	7bf86807a
501	no	no	35	9	0	1	ID_fcdff98ad	0	7bf86807a
502	no	no	17	10	0	1	ID_3a60d3642	0	7bf86807a
634	no	no	21	8	0	1	ID_01f647aaf	0	fd10905bc
635	no	no	85	0	1	0	ID_6a13e748d	1	fd10905bc
636	no	no	8	1	1	0	ID_e7fa38760	0	fd10905bc
637	no	no	36	5	1	0	ID_0d3b0550d	0	fd10905bc
669	no	no	0	0	0	1	ID_76517e7d8	0	e8725f7cb
670	no	no	73	6	0	1	ID_c30b3f5c4	0	e8725f7cb
671	no	no	68	0	1	0	ID_a5bedc6c4	1	e8725f7cb
...	...	...	...	...	...	...	...	...	...
9342	no	no	73	0	0	1	ID_b75c1d362	1	152d719e2
9343	no	no	72	0	1	0	ID_01462b000	0	152d719e2
9374	no	no	20	6	0	1	ID_b4f231ccb	0	7b7ebaf70
9375	no	no	5	0	0	1	ID_46eb2842b	0	7b7ebaf70
9376	no	no	46	2	1	0	ID_d7f45a459	0	7b7ebaf70
9377	no	no	25	6	1	0	ID_5b5fbe311	0	7b7ebaf70
9378	no	no	26	11	0	1	ID_aed542da7	0	7b7ebaf70
9379	no	no	8	1	1	0	ID_12c10b214	0	7b7ebaf70

	edjefe	edjefa	age	escolari	female	male	Id	parentesco1	idhogar
9380	no	no	15	6	1	0	ID_cd64c1af9	0	7b7ebaf70
9381	no	no	52	0	0	1	ID_d43b03397	1	7b7ebaf70
9389	no	no	97	0	0	1	ID_06caac58e	0	b4bfd5115
9390	no	no	14	7	1	0	ID_635e2f20e	0	b4bfd5115
9391	no	no	68	0	1	0	ID_2129e7f05	1	b4bfd5115
9431	no	no	21	5	0	1	ID_65d174f99	0	0966c7521
9432	no	no	70	0	0	1	ID_3ef316d48	1	0966c7521
9433	no	no	57	6	1	0	ID_317e9a221	0	0966c7521
9434	no	no	17	6	0	1	ID_adfb4cf14	0	0966c7521
9450	no	no	40	6	0	1	ID_f0886bae3	0	2ca73280c
9451	no	no	69	0	0	1	ID_3f66348ab	1	2ca73280c
9452	no	no	33	6	1	0	ID_c51d7c66b	0	2ca73280c
9453	no	no	8	2	1	0	ID_2f5ad837a	0	2ca73280c
9454	no	no	41	6	0	1	ID_d74271567	0	2ca73280c
9455	no	no	59	0	1	0	ID_dc7068500	0	2ca73280c
9456	no	no	16	6	0	1	ID_cd592016b	0	2ca73280c
9481	no	no	48	0	1	0	ID_5f199ef06	1	339954bdc
9482	no	no	47	1	0	1	ID_61845138f	0	339954bdc
9490	no	no	77	0	1	0	ID_de4b00f08	1	5ba48ed17
9506	no	no	82	0	1	0	ID_14dbfebf2	1	6548edd92
9550	no	no	61	6	1	0	ID_90a399a51	0	212db6f6c
9551	no	no	67	0	0	1	ID_79d39dddc	1	212db6f6c

435 rows × 9 columns

In [22]:
```
data[(data['edjefe']=='yes') & data['parentesco1']==1][['escolari']]
```

Out[22]:

	escolari
601	1
2394	1
2424	1
2830	1
3017	1
3744	1
4491	1
4905	1
4980	1
5154	1
5191	1
5296	1
5338	1
5387	1
5562	1
5579	1
5588	1
5728	1
6342	1
6605	1
6665	1
6743	1
6869	1
6884	1
7056	1
7079	1
7361	1
7368	1
7759	1
7833	1
7907	1
8384	1
8585	1
8903	1
8954	1
9044	1
9199	1
9257	1
9417	1

	escolari
9463	1

Basically:

'edjefe' and 'edjefa' are both 'no' when the head of the household had 0 years of school there's 'edjefe'= 'yes' and 'edjefa'='no' in some cases, all these cases the head of the household had 1 year of school there's 'edjefe'= 'no' and 'edjefa'='yes' in some cases, all these cases the head of the household had 1 year of school most of the time either 'edjefe' or 'edjefa' is a number while the other is a 'no' Let's merge the jefe and jefa education into one, undependent of gender

In [68]:
```
conditions = [
    (data['edjefe']=='no') & (data['edjefa']=='no'), #both no
    (data['edjefe']=='yes') & (data['edjefa']=='no'), # yes and no
    (data['edjefe']=='no') & (data['edjefa']=='yes'), #no and yes
    (data['edjefe']!='no') & (data['edjefe']!='yes') & (data['edjefa']=='no'),
# number and no
    (data['edjefe']=='no') & (data['edjefa']!='no') # no and number
]
choices = [0, 1, 1, data['edjefe'], data['edjefa']]
data['edjefx']=np.select(conditions, choices)
data['edjefx']=data['edjefx'].astype(int)
data[['edjefe', 'edjefa', 'edjefx']][:15]
```

Out[68]:

	edjefe	edjefa	edjefx
0	10	no	10
1	12	no	12
2	no	11	11
3	11	no	11
4	11	no	11
5	11	no	11
6	11	no	11
7	9	no	9
8	9	no	9
9	9	no	9
10	9	no	9
11	no	11	11
12	no	11	11
13	no	4	4
14	no	4	4

In [24]:
```
data.describe()
```

Out[24]:

	v2a1	hacdor	rooms	hacapo	v14a	refrig	v18q	
count	2.697000e+03	9557.000000	9557.000000	9557.000000	9557.000000	9557.000000	9557.000000	221
mean	1.652316e+05	0.038087	4.955530	0.023648	0.994768	0.957623	0.231767	
std	1.504571e+05	0.191417	1.468381	0.151957	0.072145	0.201459	0.421983	
min	0.000000e+00	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000	
25%	8.000000e+04	0.000000	4.000000	0.000000	1.000000	1.000000	0.000000	
50%	1.300000e+05	0.000000	5.000000	0.000000	1.000000	1.000000	0.000000	
75%	2.000000e+05	0.000000	6.000000	0.000000	1.000000	1.000000	0.000000	
max	2.353477e+06	1.000000	11.000000	1.000000	1.000000	1.000000	1.000000	

8 rows × 139 columns

Missing values

In [25]:
```
data.columns[data.isna().sum()!=0]
```

Out[25]:
```
Index(['v2a1', 'v18q1', 'rez_esc', 'meaneduc', 'SQBmeaned'], dtype='object')
```

Columns with nans:

- v2a1 - monthly rent
- v18q1 - number of tablets
- rez_esc - years behind school
- meaneduc - mean education for adults
- SQBmeaned - square of meaned

'meaneduc' and 'SQBmeaned' are related, let's start with those.

In [26]:
```
data[data['meaneduc'].isnull()]
```

Out[26]:

	Id	v2a1	hacdor	rooms	hacapo	v14a	refrig	v18q	v18q1	r4h1	r4h2	r4h3	r4m
1291	ID_bd8e11b0f	NaN	0	7	0	1	1	0	NaN	0	0	0	
1840	ID_46ff87316	110000.0	0	1	0	1	1	0	NaN	0	1	1	
1841	ID_69f50bf3e	110000.0	0	1	0	1	1	0	NaN	0	1	1	
2049	ID_db3168f9f	180000.0	0	3	0	1	1	0	NaN	0	2	2	
2050	ID_2a7615902	180000.0	0	3	0	1	1	0	NaN	0	2	2	

5 rows × 144 columns

that's not a lot of rows

In [27]:
```
data[data['meaneduc'].isnull()][['Id','idhogar','edjefe','edjefa', 'hogar_adul', 'hogar_mayor', 'hogar_nin', 'age', 'escolari']]
```

Out[27]:

	Id	idhogar	edjefe	edjefa	hogar_adul	hogar_mayor	hogar_nin	age	escolari
1291	ID_bd8e11b0f	1b31fd159	no	10	0	0	1	18	10
1840	ID_46ff87316	a874b7ce7	4	no	0	0	2	18	6
1841	ID_69f50bf3e	a874b7ce7	4	no	0	0	2	18	4
2049	ID_db3168f9f	faaebf71a	12	no	0	0	2	19	12
2050	ID_2a7615902	faaebf71a	12	no	0	0	2	19	12

In [28]:
```
print(len(data[data['idhogar']==data.iloc[1291]['idhogar']]))
print(len(data[data['idhogar']==data.iloc[1840]['idhogar']]))
print(len(data[data['idhogar']==data.iloc[2049]['idhogar']]))
```

```
1
2
2
```

So, the 5 rows with Nan for 'meaneduc' is just 3 households, where 18-19 year-olds live. No other people live in these households. Then we can just take the education levels of these kids ('escolari') and put them into 'meaneduc' and 'SQBmeaned'.

In [0]:
```
meaneduc_nan=data[data['meaneduc'].isnull()][['Id','idhogar','escolari']]
```

In [0]:
```
me=meaneduc_nan.groupby('idhogar')['escolari'].mean().reset_index()
```

In [32]:
```
me
```

Out[32]:

	idhogar	escolari
0	1b31fd159	10
1	a874b7ce7	5
2	faaebf71a	12

In [0]:
```
for row in meaneduc_nan.iterrows():
    idx=row[0]
    idhogar=row[1]['idhogar']
    m=me[me['idhogar']==idhogar]['escolari'].tolist()[0]
    data.at[idx, 'meaneduc']=m
    data.at[idx, 'SQBmeaned']=m*m
```

Next, let's look at 'v2a1', the monthly rent payment, that also has missing values.

In [34]:
```
data['v2a1'].isnull().sum()
```

Out[34]: 6860

That's a lot of missing values.

But maybe they own their houses. We can look what type of housing these people with missing values live..

In [35]:
```
norent=data[data['v2a1'].isnull()]
print("Owns his house:", norent[norent['tipovivi1']==1]['Id'].count())
print("Owns his house paying installments", norent[norent['tipovivi2']==1]['I
d'].count())
print("Rented ", norent[norent['tipovivi3']==1]['Id'].count())
print("Precarious ", norent[norent['tipovivi4']==1]['Id'].count())
print("Other ", norent[norent['tipovivi5']==1]['Id'].count())
print("Total ", 6860)
```

```
Owns his house: 5911
Owns his house paying installments 0
Rented  0
Precarious  163
Other  786
Total  6860
```

The majority in fact owns their houses, only a few have odd situations. We can probably just assume they don't pay rent, and put 0 in these cases.

In [0]:
```
data['v2a1']=data['v2a1'].fillna(0)
```

Now, let's look at 'v18q1', which indicates how many tablets the household owns.

In [37]:
```
data['v18q1'].isna().sum()
```

Out[37]:
```
7342
```

That's also a lot rows with missing values... However, there's a column, 'v18q', which indicates whether there's a tablet in the household at all, that might help!

In [38]:
```
tabletnan=data[data['v18q1'].isnull()]
tabletnan[tabletnan['v18q']==0]['Id'].count()
```

Out[38]:
```
7342
```

In [39]:
```
data['v18q1'].unique()
```

Out[39]:
```
array([nan,  1.,  2.,  3.,  4.,  5.,  6.])
```

That's exactly the number of rows with missing values! There's also no 0 among the values of 'v18q1'. So all the nans in 'v18q1' just means they don't own a tablet! So we can just change them to 0.

In [0]:
```
data['v18q1']=data['v18q1'].fillna(0)
```

Next up is 'rez_esc', which indicates if a person is behind in school.

In [41]:
```
data['rez_esc'].isnull().sum()
```

Out[41]:
```
7928
```

that's a lot of rows, so I will explore more

In [42]:
```
data['rez_esc'].describe()
```

Out[42]:
```
count    1629.000000
mean        0.459791
std         0.946550
min         0.000000
25%         0.000000
50%         0.000000
75%         1.000000
max         5.000000
Name: rez_esc, dtype: float64
```

In [43]:
```
data['rez_esc'].unique()
```

Out[43]:
```
array([nan,  1.,  0.,  3.,  2.,  4.,  5.])
```

In [44]:
```
data[data['rez_esc']>1][['age', 'escolari', 'rez_esc']][:20]
```

Out[44]:

	age	escolari	rez_esc
185	13	3	3.0
190	16	7	2.0
240	16	6	3.0
242	11	2	2.0
312	15	6	2.0
317	17	6	4.0
319	17	7	3.0
320	15	6	2.0
333	16	7	2.0
510	16	7	2.0
617	16	6	3.0
629	16	6	3.0
714	17	6	4.0
769	17	6	4.0
837	16	6	3.0
937	16	6	3.0
978	16	7	2.0
1146	16	6	3.0
1150	15	6	2.0
1151	16	7	2.0

In [45]:
```
rez_esc_nan=data[data['rez_esc'].isnull()]
rez_esc_nan[(rez_esc_nan['age']<18) & rez_esc_nan['escolari']>0][['age', 'esco
lari']]
```

Out[45]:

	age	escolari

So all the nans here are either adults or children before school age. We can input 0 again.

In [0]:
```
data['rez_esc']=data['rez_esc'].fillna(0)
```

Normalizing same households with different target values

In [0]:
```
d={}
weird=[]
for row in data.iterrows():
    idhogar=row[1]['idhogar']
    target=row[1]['Target']
    if idhogar in d:
        if d[idhogar]!=target:
            weird.append(idhogar)
    else:
        d[idhogar]=target
```

In [49]:
```
len(set(weird))
```

Out[49]:
```
85
```

There are 85 households like that.

setting target value = head of the household value

In [50]:
```
data[data['idhogar']==weird[2]][['idhogar','parentesco1', 'Target']]
```

Out[50]:

	idhogar	parentesco1	Target
285	6833ac5dc	0	2
286	6833ac5dc	0	2
287	6833ac5dc	0	2
288	6833ac5dc	1	2
289	6833ac5dc	0	2
290	6833ac5dc	0	1

In [0]:
```
for i in set(weird):
    hhold=data[data['idhogar']==i][['idhogar', 'parentesco1', 'Target']]
    target=hhold[hhold['parentesco1']==1]['Target'].tolist()[0]
    for row in hhold.iterrows():
        idx=row[0]
        if row[1]['parentesco1']!=1:
            data.at[idx, 'Target']=target
```

In [52]:
```
data[data['idhogar']==weird[1]][['idhogar','parentesco1', 'Target']]
```

Out[52]:

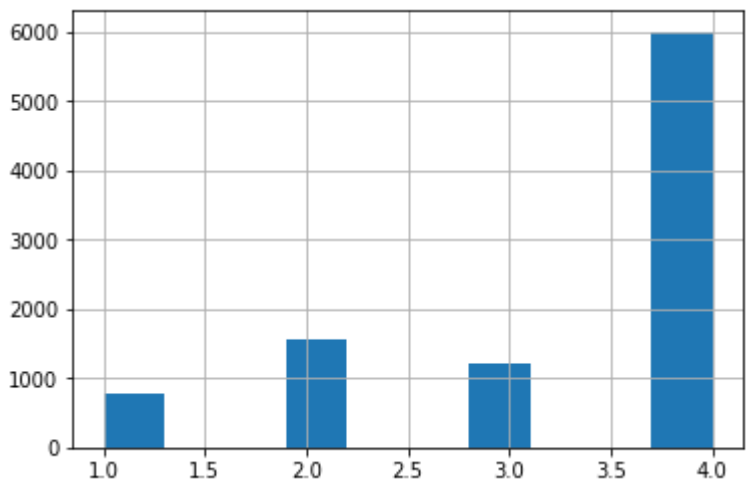
	idhogar	parentesco1	Target
282	4b6077882	1	1
283	4b6077882	0	1
284	4b6077882	0	1

In [0]:
```
def data_cleaning(data):
    data['dependency']=np.sqrt(data['SQBdependency'])
    data['rez_esc']=data['rez_esc'].fillna(0)
    data['v18q1']=data['v18q1'].fillna(0)
    data['v2a1']=data['v2a1'].fillna(0)

    conditions = [
    (data['edjefe']=='no') & (data['edjefa']=='no'), #both no
    (data['edjefe']=='yes') & (data['edjefa']=='no'), # yes and no
    (data['edjefe']=='no') & (data['edjefa']=='yes'), #no and yes
    (data['edjefe']!='no') & (data['edjefe']!='yes') & (data['edjefa']=='no'),
# number and no
    (data['edjefe']=='no') & (data['edjefa']!='no') # no and number
    ]
    choices = [0, 1, 1, data['edjefe'], data['edjefa']]
    data['edjefx']=np.select(conditions, choices)
    data['edjefx']=data['edjefx'].astype(int)
    data.drop(['edjefe', 'edjefa'], axis=1, inplace=True)

    meaneduc_nan=data[data['meaneduc'].isnull()][['Id','idhogar','escolari']]
    me=meaneduc_nan.groupby('idhogar')['escolari'].mean().reset_index()
    for row in meaneduc_nan.iterrows():
        idx=row[0]
        idhogar=row[1]['idhogar']
        m=me[me['idhogar']==idhogar]['escolari'].tolist()[0]
        data.at[idx, 'meaneduc']=m
        data.at[idx, 'SQBmeaned']=m*m

    return data
```

In [0]:
```
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
```

In [55]:
```
data['Target'].hist()
```

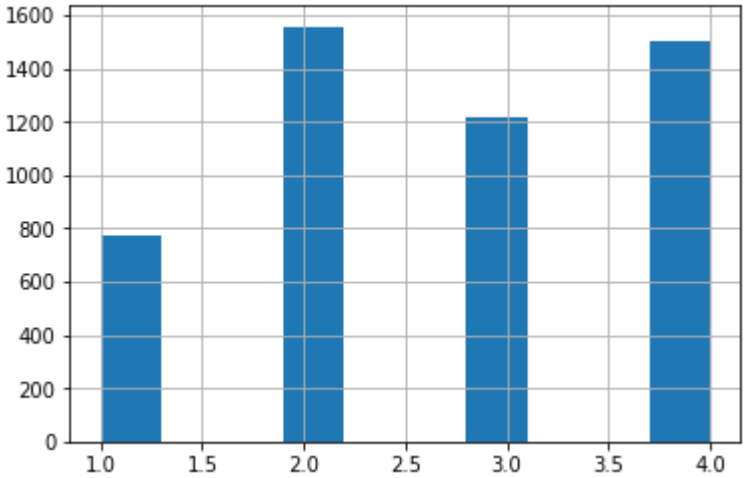
Out[55]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f585360b208>
```



In [0]:
```
data_undersampled=data.drop(data.query('Target == 4').sample(frac=.75).index)
```

In [57]:
```
data_undersampled['Target'].hist()
```

Out[57]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5850d14be0>
```



In [0]:
```
data_undersampled['dependency'].hist()
```

Random Forest

In [0]:
```
X=data_undersampled.drop(['Id', 'idhogar', 'Target', 'edjefe', 'edjefa'], axis
=1)
y=data_undersampled['Target']
```

In [0]:
```
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)
```

In [71]:
```
X_train.shape
```

Out[71]:
```
(4043, 139)
```

In [72]:
```
y_train.shape
```

Out[72]:
```
(4043,)
```

In [0]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
```

In [0]:
```
clf = RandomForestClassifier()
params={'n_estimators': list(range(40,61, 1))}
gs = GridSearchCV(clf, params, cv=5)
```

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

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-81-dd947cb3301f> in <module>()
----> 1 gs.fit(X_train, y_train)

/usr/local/lib/python3.6/dist-packages/sklearn/model_selection/_search.py in
fit(self, X, y, groups, **fit_params)
    685                 return results
    686
--> 687             self._run_search(evaluate_candidates)
    688
    689         # For multi-metric evaluation, store the best_index_, best_pa
rams_ and

/usr/local/lib/python3.6/dist-packages/sklearn/model_selection/_search.py in
_run_search(self, evaluate_candidates)
   1146     def _run_search(self, evaluate_candidates):
   1147         """Search all candidates in param_grid"""
-> 1148         evaluate_candidates(ParameterGrid(self.param_grid))
   1149
   1150

/usr/local/lib/python3.6/dist-packages/sklearn/model_selection/_search.py in
evaluate_candidates(candidate_params)
    664                                   for parameters, (train, test)
    665                                   in product(candidate_params,
--> 666                                              cv.split(X, y, groups)))
    667
    668                 if len(out) < 1:

/usr/local/lib/python3.6/dist-packages/joblib/parallel.py in __call__(self, i
terable)
    919             # remaining jobs.
    920             self._iterating = False
--> 921             if self.dispatch_one_batch(iterator):
    922                 self._iterating = self._original_iterator is not None
    923

/usr/local/lib/python3.6/dist-packages/joblib/parallel.py in dispatch_one_bat
ch(self, iterator)
    757                 return False
    758             else:
--> 759                 self._dispatch(tasks)
    760                 return True
    761

/usr/local/lib/python3.6/dist-packages/joblib/parallel.py in _dispatch(self,
 batch)
    714         with self._lock:
    715             job_idx = len(self._jobs)
--> 716             job = self._backend.apply_async(batch, callback=cb)
    717             # A job can complete so quickly than its callback is
    718             # called before we get here, causing self._jobs to

/usr/local/lib/python3.6/dist-packages/joblib/_parallel_backends.py in apply_
async(self, func, callback)
    180     def apply_async(self, func, callback=None):
    181         """Schedule a func to be run"""
--> 182         result = ImmediateResult(func)
    183         if callback:
    184             callback(result)

/usr/local/lib/python3.6/dist-packages/joblib/_parallel_backends.py in __init
__(self, batch)
    547         # Don't delay the application, to avoid keeping the input
```

```
    548         # arguments in memory
--> 549         self.results = batch()
    550
    551     def get(self):

/usr/local/lib/python3.6/dist-packages/joblib/parallel.py in __call__(self)
    223         with parallel_backend(self._backend, n_jobs=self._n_jobs):
    224             return [func(*args, **kwargs)
--> 225                     for func, args, kwargs in self.items]
    226
    227     def __len__(self):

/usr/local/lib/python3.6/dist-packages/joblib/parallel.py in <listcomp>(.0)
    223         with parallel_backend(self._backend, n_jobs=self._n_jobs):
    224             return [func(*args, **kwargs)
--> 225                     for func, args, kwargs in self.items]
    226
    227     def __len__(self):

/usr/local/lib/python3.6/dist-packages/sklearn/model_selection/_validation.py
in _fit_and_score(estimator, X, y, scorer, train, test, verbose, parameters,
 fit_params, return_train_score, return_parameters, return_n_test_samples, re
turn_times, return_estimator, error_score)
    512             estimator.fit(X_train, **fit_params)
    513         else:
--> 514             estimator.fit(X_train, y_train, **fit_params)
    515
    516     except Exception as e:

/usr/local/lib/python3.6/dist-packages/sklearn/ensemble/forest.py in fit(sel
f, X, y, sample_weight)
    247
    248         # Validate or convert input data
--> 249         X = check_array(X, accept_sparse="csc", dtype=DTYPE)
    250         y = check_array(y, accept_sparse='csc', ensure_2d=False, dtyp
e=None)
    251         if sample_weight is not None:

/usr/local/lib/python3.6/dist-packages/sklearn/utils/validation.py in check_a
rray(array, accept_sparse, accept_large_sparse, dtype, order, copy, force_all
_finite, ensure_2d, allow_nd, ensure_min_samples, ensure_min_features, warn_o
n_dtype, estimator)
    494             try:
    495                 warnings.simplefilter('error', ComplexWarning)
--> 496                 array = np.asarray(array, dtype=dtype, order=order)
    497             except ComplexWarning:
    498                 raise ValueError("Complex data not supported\n"

/usr/local/lib/python3.6/dist-packages/numpy/core/numeric.py in asarray(a, dt
ype, order)
    536
    537     """
--> 538     return array(a, dtype, copy=False, order=order)
    539
    540

ValueError: could not convert string to float: 'no'
```

In [0]:

In [0]:

In [0]:

In [0]:

In [0]:

In [0]:

In [0]:

In [0]: