

```
In [1]: 1 import pandas as pd  
2 import numpy as np  
3 import matplotlib.pyplot as plt  
4 import seaborn as sns  
5 %matplotlib inline  
6  
7 import warnings  
8 warnings.filterwarnings('ignore')
```

```
In [2]: 1 train=pd.read_csv('train.csv')  
2 test=pd.read_csv('test.csv')
```

Let us explore the dataset before moving forward

```
In [3]: 1 print('The shape of the train dataset is: {}'.format(train.shape))  
2 print('The shape of the test dataset is: {}'.format(test.shape))
```

The shape of the train dataset is: (9557, 143)
The shape of the test dataset is: (23856, 142)

Let us identify our target variable

```
In [4]: 1 for i in train.columns:  
2     if i not in test.columns:  
3         print('Target variable is {}'.format(i))
```

Target variable is Target

```
In [5]: 1 #Let us understand the type of data type  
2 print(train.dtypes.value_counts())
```

int64 130
float64 8
object 5
dtype: int64

```
In [6]: 1 print(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
None

```
In [7]: 1 #We have mixed data types. Specified as below:  
2  
3 #float64 : 8 variables  
4 #int64 : 130 variables  
5 #object :5 variables
```

```
In [8]: 1 #Let us explore each different type of datasets  
2 for i in train.columns:  
3     a=train[i].dtype  
4     if a == 'object':  
5         print(i)
```

Id
idhogar
dependency
edjefe
edjefa

Below is Data dictionary for above object variables

->ID = Unique ID

->idhogar, Household level identifier

->dependency, Dependency rate, calculated = (number of members of the household younger than 19 or older than 64)/(number of member of household between 19 and 64)

->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

Let's drop ID variable

```
In [9]: 1 train.drop(['Id','idhogar'],axis=1,inplace=True)
```

```
In [10]: 1 train.columns
```

```
Out[10]: Index(['v2a1', 'hacdor', 'rooms', 'hacapo', 'v14a', 'refrig', 'v18q', 'v18q1',  
   'r4h1', 'r4h2',  
   ...  
   'SQBescalar1', 'SQBage', 'SQBhogar_total', 'SQBedjefe', 'SQBhogar_nin',  
   'SQBovercrowding', 'SQBdependency', 'SQBmeaned', 'agesq', 'Target'],  
  dtype='object', length=141)
```

```
In [11]: 1 train['dependency'].value_counts()
```

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

```
In [12]: 1 def map(i):
2
3     if i=='yes':
4         return(float(1))
5     elif i=='no':
6         return(float(0))
7     else:
8         return(float(i))
```

```
In [13]: 1 train['dependency'] = train['dependency'].apply(map)
2 train['dependency'].value_counts()
```

```
Out[13]: 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.----- 1
0.222222 11
0.285714  9
1.666667 8
0.600000 8
6.000000 7
0.166667 7
Name: dependency, dtype: int64
```

```
In [14]: 1 for i in train.columns:
2     a = train[i].dtypes
3     if a == 'object':
4         print(i)

edjefe
edjefa
```

```
In [15]: 1 train['edjefe']=train['edjefe'].apply(map)
2 train['edjefa']=train['edjefa'].apply(map)
```

```
In [16]: 1 train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9557 entries, 0 to 9556
Columns: 141 entries, v2a1 to Target
dtypes: float64(11), int64(130)
memory usage: 10.3 MB
```

```
In [17]: 1 #Now all data is numerical
```

Let us identify variable with 0 variance

```
In [18]: 1 var_df = pd.DataFrame(np.var(train,0),columns=['variance'])
2 var_df.sort_values(by='variance').head(15)
3 print('Below are columns with 0 variance')
4 col=list((var_df[var_df['variance']==0]).index)
5 print(col)
```

Below are columns with 0 variance
['elimbasu5']

#elimbasu5 : 1 if rubbish disposal mainly by throwing in river, creek or sea.

Interpretation :From above it is shown that all values of elimbasu5 is same so there is no variability in dataset therefore we will drop this variable

Check if there are any biases in our dataset

```
In [19]: 1 contingency_tab=pd.crosstab(train['r4t3'],train['hogar_total'])
2 Observed_Values=contingency_tab.values
```

```
In [20]: 1 import scipy.stats
2 b=scipy.stats.chi2_contingency(contingency_tab)
3 Expected_Values = b[3]
4 no_of_rows=len(contingency_tab.iloc[0:2,0])
5 no_of_columns=len(contingency_tab.iloc[0,0:2])
6 df=(no_of_rows-1)*(no_of_columns-1)
7 print("Degree of Freedom:-",df)
```

Degree of Freedom:- 1

```
In [21]: 1 from scipy.stats import chi2
2 chi_square=sum([(o-e)**2./e for o,e in zip(Observed_Values,Expected_Values)])
3 chi_square_statistic=chi_square[0]+chi_square[1]
4 print("chi-square statistic:-",chi_square_statistic)
5 alpha=0.05
6 critical_value=chi2.ppf(q=1-alpha,df=df)
7 print('critical_value:',critical_value)
8 p_value=1-chi2.cdf(x=chi_square_statistic,df=df)
9 print('p-value:',p_value)
10 print('Significance level: ',alpha)
11 print('Degree of Freedom: ',df)
12 print('chi-square statistic:',chi_square_statistic)
13 print('critical_value:',critical_value)
14 print('p-value:',p_value)
```

chi-square statistic:- 17022.072400560897

critical_value: 3.841458820694124

p-value: 0.0

Significance level: 0.05

Degree of Freedom: 1

chi-square statistic: 17022.072400560897

critical_value: 3.841458820694124

p-value: 0.0

```
In [22]: 1 if chi_square_statistic>=critical_value:
2     print("Reject H0, There is a relationship between 2 categorical variables")
3 else:
4     print("Retain H0, There is no relationship between 2 categorical variables")
5
6 if p_value<=alpha:
7     print("Reject H0, There is a relationship between 2 categorical variables")
8 else:
9     print("Retain H0, There is no relationship between 2 categorical variables")
```

Reject H0, There is a relationship between 2 categorical variables

Reject H0, There is a relationship between 2 categorical variables

```
In [23]: 1 #2
2 contingency_tab=pd.crosstab(train['tipovivi3'],train['v2a1'])
3 Observed_Values=contingency_tab.values
```

```
In [24]: 1 import scipy.stats
2 b=scipy.stats.chi2_contingency(contingency_tab)
3 Expected_Values = b[3]
4 no_of_rows=len(contingency_tab.iloc[0:2,0])
5 no_of_columns=len(contingency_tab.iloc[0,0:2])
6 df=(no_of_rows-1)*(no_of_columns-1)
7 print("Degree of Freedom:-",df)
```

Degree of Freedom:- 1

```
In [25]: 1 from scipy.stats import chi2
2 chi_square=sum([(o-e)**2./e for o,e in zip(Observed_Values,Expected_Values)])
3 chi_square_statistic=chi_square[0]+chi_square[1]
4 print("chi-square statistic:-",chi_square_statistic)
5 alpha=0.05
6 critical_value=chi2.ppf(q=1-alpha,df=df)
7 print('critical_value:',critical_value)
8 p_value=1-chi2.cdf(x=chi_square_statistic,df=df)
9 print('p-value:',p_value)
10 print('Significance level: ',alpha)
11 print('Degree of Freedom: ',df)
12 print('chi-square statistic:',chi_square_statistic)
13 print('critical_value:',critical_value)
14 print('p-value:',p_value)
```

chi-square statistic:- 54.04781105990782

critical_value: 3.841458820694124

p-value: 1.9562129693895258e-13

Significance level: 0.05

Degree of Freedom: 1

chi-square statistic: 54.04781105990782

critical_value: 3.841458820694124

p-value: 1.9562129693895258e-13

```
In [26]: 1 if chi_square_statistic>=critical_value:
2     print("Reject H0,There is a relationship between 2 categorical variables")
3 else:
4     print("Retain H0,There is no relationship between 2 categorical variables")
5
6 if p_value<=alpha:
7     print("Reject H0,There is a relationship between 2 categorical variables")
8 else:
9     print("Retain H0,There is no relationship between 2 categorical variables")
```

Reject H0,There is a relationship between 2 categorical variables
Reject H0,There is a relationship between 2 categorical variables

```
In [27]: 1 #3
2 contingency_tab=pd.crosstab(train['v18q'],train['v18q1'])
3 Observed_Values=contingency_tab.values
```

```
In [28]: 1 import scipy.stats
2 b=scipy.stats.chi2_contingency(contingency_tab)
3 Expected_Values = b[3]
4 no_of_rows=len(contingency_tab.iloc[0:2,0])
5 no_of_columns=len(contingency_tab.iloc[0,0:2])
6 df=(no_of_rows-1)*(no_of_columns-1)
7 print("Degree of Freedom:-",df)
```

Degree of Freedom:- 0

```
In [29]: 1 from scipy.stats import chi2
2 chi_square=sum([(o-e)**2./e for o,e in zip(Observed_Values,Expected_Values)])
3 chi_square_statistic=chi_square[0]+chi_square[1]
4 print("chi-square statistic:-",chi_square_statistic)
5 alpha=0.05
6 critical_value=chi2.ppf(q=1-alpha,df=df)
7 print('critical_value:',critical_value)
8 p_value=1-chi2.cdf(x=chi_square_statistic,df=df)
9 print('p-value:',p_value)
10 print('Significance level: ',alpha)
11 print('Degree of Freedom: ',df)
12 print('chi-square statistic:',chi_square_statistic)
13 print('critical_value:',critical_value)
14 print('p-value:',p_value)
```

chi-square statistic:- 0.0
critical_value: nan
p-value: nan
Significance level: 0.05
Degree of Freedom: 0
chi-square statistic: 0.0
critical_value: nan
p-value: nan

```
In [30]: 1 if chi_square_statistic>=critical_value:
2     print("Reject H0,There is a relationship between 2 categorical variables")
3 else:
4     print("Retain H0,There is no relationship between 2 categorical variables")
5
6 if p_value<=alpha:
7     print("Reject H0,There is a relationship between 2 categorical variables")
8 else:
9     print("Retain H0,There is no relationship between 2 categorical variables")
```

Retain H0,There is no relationship between 2 categorical variables
Retain H0,There is no relationship between 2 categorical variables

##Therefore,variables ('v18q','v18q1') have relationship between them. For good result we can use any one of them.

Conclusion : Therefore, there is bias in our dataset.

```
In [31]: 1 train.drop('r4t3',axis=1,inplace=True)
```

Check if there is a house without a family head

"parentesco1" =1 if household head

```
In [32]: 1 train['parentesco1'].value_counts()
```

```
Out[32]: 0    6584
1    2973
Name: parentesco1, dtype: int64
```

```
In [33]: 1 pd.crosstab(train['edjefa'],train['edjefe'])
```

```
Out[33]: edjefe  0.0  1.0  2.0  3.0  4.0  5.0  6.0  7.0  8.0  9.0 ... 12.0  13.0  14.0  15.0  16.0  17.0  18.0  19.0  20.0  21.0
edjefe
 0.0  435  123  194  307  137  222  1845  234  257  486 ... 113  103  208  285  134  202  19  14  7  43
 1.0   69  0  0  0  0  0  0  0  0 ... 0  0  0  0  0  0  0  0  0  0
 2.0   84  0  0  0  0  0  0  0  0 ... 0  0  0  0  0  0  0  0  0  0
 3.0  152  0  0  0  0  0  0  0  0 ... 0  0  0  0  0  0  0  0  0  0
 4.0  136  0  0  0  0  0  0  0  0 ... 0  0  0  0  0  0  0  0  0  0
 5.0  176  0  0  0  0  0  0  0  0 ... 0  0  0  0  0  0  0  0  0  0
 6.0  947  0  0  0  0  0  0  0  0 ... 0  0  0  0  0  0  0  0  0  0
 7.0  179  0  0  0  0  0  0  0  0 ... 0  0  0  0  0  0  0  0  0  0
 8.0  217  0  0  0  0  0  0  0  0 ... 0  0  0  0  0  0  0  0  0  0
 9.0  237  0  0  0  0  0  0  0  0 ... 0  0  0  0  0  0  0  0  0  0
10.0  96  0  0  0  0  0  0  0  0 ... 0  0  0  0  0  0  0  0  0  0
11.0  399  0  0  0  0  0  0  0  0 ... 0  0  0  0  0  0  0  0  0  0
```

10.0	96	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11.0	399	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12.0	72	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13.0	52	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14.0	120	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15.0	188	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16.0	113	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17.0	76	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18.0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19.0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20.0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21.0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

22 rows × 22 columns

Interpretation : Above cross tab shows 0 male head and 0 female head which implies that there are 435 families with no family head.

Count how many null values are existing in columns.

In [34]: 1 train.isna().sum().value_counts()

Out[34]: 0 135
5 2
6860 1
7342 1
7928 1
dtype: int64

In [35]: 1 train['Target'].isna().sum()

Out[35]: 0

Interpretation : There are no null values in Target variable. Now lets proceed further and identify and fillna of other variable.

In [36]: 1 float_col = []
2 for i in train.columns:
3 a=train[i].dtype
4 if a=='float64':
5 float_col.append(i)
6 print(float_col)

['v2a1', 'v18q1', 'rez_esc', 'dependency', 'edjefe', 'edjefa', 'meaneduc', 'overcrowding', 'SQBovercrowding', 'SQBdependency', 'SQBmeaned']

In [37]: 1 train[float_col].isna().sum()

Out[37]: v2a1 6860
v18q1 7342
rez_esc 7928
dependency 0
edjefe 0
edjefa 0
meaneduc 5
overcrowding 0
SQBovercrowding 0
SQBdependency 0
SQBmeaned 5
dtype: int64

In [38]: 1 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]: 1 pd.crosstab(train['tipovivi1'],train['v2a1'])

Out[39]: v2a1 0.0 12000.0 13000.0 14000.0 15000.0 16000.0 17000.0 20000.0 23000.0 25000.0 ... 570540.0 600000.0 620000.0 684648.0 700000.0 770229.0
tipovivi1
0 29 3 4 3 3 2 4 22 5 21 ... 25 11 3 3 7 3

1 rows × 157 columns

In [40]: 1 pd.crosstab(train['v18q1'],train['v18q'])

Out[40]: v18q 1
v18q1
1.0 1586
2.0 444
3.0 129
4.0 37
5.0 13
6.0 6

Interpretation and action : 'v2a1', 'v18q1', 'rez_esc' have more than 50% null values, because for v18q1, there are families with their own house so they won't pay rent in that case it should be 0 and similar is for v18q1 there can be families with 0 tablets.

Instead we can drop a column tipovivi3,v18q

tipovivi3, =1 rented v18q, owns a tablet as v2a1 alone can show both **as v18q1 alone can show that if respondent owns a tablet or not

```
In [41]: 1 train['v2a1'].fillna(0,inplace=True)
2 train['v18q1'].fillna(0,inplace=True)

In [42]: 1 train.drop(['tipovivi3', 'v18q','rez_esc','elimbasu5'],axis=1,inplace=True)

In [43]: 1 train['meaneduc'].fillna(np.mean(train['meaneduc']),inplace=True)
2 train['SQBmeaned'].fillna(np.mean(train['SQBmeaned']),inplace=True)
3 print(train.isna().sum().value_counts())

0    136
dtype: int64

In [44]: 1 int_col=[]
2 for i in train.columns:
3     a=train[i].dtype
4     if a == 'int64':
5         int_col.append(i)
6 print(int_col)

['hacdor', 'rooms', 'hacapo', 'v14a', 'refrig', 'r4h1', 'r4h2', 'r4h3', 'r4m1', 'r4m2', 'r4m3', 'r4t1', 'r4t2', 'tamhog', 'tamviv', 'escolari', 'hhsize', 'paredblolad', 'paredzocalo', 'paredpreb', 'pareddeds', 'paredmad', 'paredzin c', 'paredfibras', 'paredother', 'pisomoscer', 'pisocemento', 'pisother', 'pisonatur', 'pisonotiene', 'pisomadera', 'techozinc', 'techoentrempiso', 'techocane', 'techootro', 'cielorazo', 'abastaguardentro', 'abastaguafuera', 'abast aguano', 'public', 'planpri', 'coopele', 'sanitario1', 'sanitario2', 'sanitario3', 'sanitario5', 'sanitar io6', 'energococinar1', 'energococinar2', 'energococinar3', 'energococinar4', 'elimbasu1', 'elimbasu2', 'elimbasu3', 'elimbasu4', 'elimbasu5', 'epared2', 'epared3', 'etecho1', 'etecho2', 'etecho3', 'eviv1', 'eviv2', 'eviv3', 'dis', 'male', 'female', 'estadocivil1', 'estadocivil2', 'estadocivil3', 'estadocivil4', 'estadocivil5', 'estadoc ivil6', 'estadocivil7', 'parentesco1', 'parentesco2', 'parentesco3', 'parentesco4', 'parentesco5', 'parentesco6', 'parentesco7', 'parentesco8', 'parentesco9', 'parentesco10', 'parentesco11', 'parentesco12', 'hogar_min', 'hogar_adu l', 'hogar_mayor', 'hogar_total', 'instlevel1', 'instlevel2', 'instlevel3', 'instlevel4', 'instlevel5', 'instlevel6 ', 'instlevel7', 'instlevel8', 'instlevel9', 'bedrooms', 'tipovivi1', 'tipovivi2', 'tipovivi4', 'tipovivi5', 'compu ter', 'television', 'mobilephone', 'qmobilephone', 'lugar1', 'lugar2', 'lugar3', 'lugar4', 'lugar5', 'lugar6', 'are a1', 'area2', 'age', 'SQBesculari', 'SQBage', 'SQBhogar_total', 'SQBbedjefe', 'SQBhogar_min', 'agesq', 'Target']

In [45]: 1 train[int_col].isna().sum().value_counts()

Out[45]: 0    126
dtype: int64
```

Interpretation: Now there is no null value in our dataset

```
In [46]: 1 train.Target.value_counts()

Out[46]: 4    5996
2    1597
3   1209
1    755
Name: Target, dtype: int64
```

Set the poverty level of the members and the head of the house same in a family.

Now for people below poverty level can be people paying less rent and don't own a house. and it also depends on whether a house is in urban area or rural area.

```
In [47]: 1 Poverty_level = train[train['v2a1']!=0]

In [48]: 1 Poverty_level.shape

Out[48]: (2668, 136)

In [49]: 1 poverty_level=Poverty_level.groupby('area1')['v2a1'].apply(np.median)
2 poverty_level

Out[49]: area1
0    80000.0
1   140000.0
Name: v2a1, dtype: float64
```

For rural area level if people paying rent less than 8000 is under poverty level.

For Urban area level if people having rent less than 140000 is under poverty level.

```
In [50]: 1 def povert(x):
2     if x<8000:
3         return('Below poverty level')
4
5     elif x>140000:
6         return('Above poverty level')
7     elif x<140000:
8         return('Below poverty level: Ur-ban ; Above poverty level : Rural ')
```

```
In [51]: 1 c=Poverty_level['v2a1'].apply(povert)
2 c.shape

Out[51]: (2668,)

In [52]: 1 pd.crosstab(c,Poverty_level['area1'])

Out[52]: area1      0      1
                    ...
v2a1
Above poverty level  139  1103
Below poverty level: Ur-ban ; Above poverty level : Rural  306  1081
```

Interpretation :

There are total 1242 people above poverty level independent of area whether rural or Urban Remaining 1111 people level depends on their area Rural :

Above poverty level= 445

Urban :

Above poverty level=1103

Below poverty level=1081

```
In [53]: 1 from sklearn.ensemble import RandomForestClassifier  
2 from sklearn.model_selection import train_test_split
```

```
In [54]: 1 X_data=train.drop('Target',axis=1)  
2 Y_data=train.Target
```

```
In [55]: 1 X_data_col=X_data.columns
```

Applying standard scaling to dataset

```
In [56]: 1 from sklearn.preprocessing import StandardScaler  
2 SS=StandardScaler()  
3 X_data_1=SS.fit_transform(X_data)  
4 X_data_1=pd.DataFrame(X_data_1,columns=X_data_col)
```

Now let us perform model fitting

```
In [57]: 1 X_train,X_test,Y_train,Y_test=train_test_split(X_data_1,Y_data,test_size=0.25,stratify=Y_data,random_state=0)
```

```
In [58]: 1 from sklearn.pipeline import Pipeline  
2 from sklearn.model_selection import GridSearchCV
```

```
In [59]: 1 rfc=RandomForestClassifier(random_state=0)  
2 parameters={'n_estimators':[10,50,100,300], 'max_depth':[3,5,10,15]}  
3 grid=zip([rfc],[parameters])  
4  
5 best_=None
```

```
In [60]: 1 for i, j in grid:  
2     a=GridSearchCV(i,param_grid=j, cv=3,n_jobs=1)  
3     a.fit(X_train,Y_train)  
4     if best_ is None:  
5         best_=a  
6     elif a.best_score_>best_.best_score_:  
7         best_=a
```

```
In [61]: 1 print ("Best CV Score",best_.best_score_)  
2 print ("Model Parameters",best_.best_params_)  
3 print("Best Estimator",best_.best_estimator_)
```

```
Best CV Score 0.8507046183898423  
Model Parameters {'max_depth': 15, 'n_estimators': 300}  
Best Estimator RandomForestClassifier(max_depth=15, n_estimators=300, random_state=0)
```

```
In [62]: 1 RFC=best_.best_estimator_  
2 Model=RFC.fit(X_train,Y_train)  
3 pred=Model.predict(X_test)
```

```
In [63]: 1 print('Model Score of train data : {}'.format(Model.score(X_train,Y_train)))  
2 print('Model Score of test data : {}'.format(Model.score(X_test,Y_test)))
```

```
Model Score of train data : 0.9831170643225896  
Model Score of test data : 0.8824267782426778
```

```
In [64]: 1 Important_features=pd.DataFrame(Model.feature_importances_,X_data_col,columns=['feature_importance'])
```

```
In [65]: 1 Top50Features=Important_features.sort_values(by='feature_importance',ascending=False).head(50).index  
2 Top50Features
```

```
Out[65]: Index(['SQBmeaned', 'meaneduc', 'SQBdependency', 'dependency', 'overcrowding',  
 'SQBovercrowding', 'qmobilephone', 'SQBhogar_nin', 'SQBedjefe',  
 'edjefe', 'hogar_nin', 'rooms', 'cielorazo', 'r4tl', 'v2a1', 'edjefa',  
 'agesq', 'r4m3', 'r4h2', 'SQBage', 'age', 'escolari', 'r4t2', 'r4h3',  
 'hogar_adul', 'SQBescolari', 'eviv3', 'bedrooms', 'r4m1', 'epared3',  
 'r4m2', 'tamviv', 'paredblolad', 'v18q1', 'SQBhogar_total', 'tamhog',  
 'hhsize', 'hogar_total', 'pisomoscer', 'etecho3', 'r4h1', 'lugar1',  
 'eviv2', 'tipovivi1', 'energocinar2', 'energcocinar3', 'epared2',  
 'television', 'area2', 'area1'],  
 dtype='object')
```

```
In [66]: 1 for i in Top50Features:  
2     if i not in X_data_col:  
3         print(i)  
4  
5 X_data_Top50=X_data[Top50Features]
```

```
In [67]: 1 X_train,X_test,Y_train,Y_test=train_test_split(X_data_Top50,Y_data,test_size=0.25,stratify=Y_data,random_state=0  
2  
3 Model_1=RFC.fit(X_train,Y_train)  
4 pred=Model_1.predict(X_test)  
5  
6 from sklearn.metrics import confusion_matrix,f1_score,accuracy_score
```

```
In [68]: 1 confusion_matrix(Y_test,pred)
```

```
Out[68]: array([[ 143,    17,     0,    29],  
 [     8,   324,     4,    63],  
 [     1,    12,   214,    75],  
 [     2,    10,     3, 1485]])
```

```
In [69]: 1 f1_score(Y_test,pred,average='weighted')
```

```
Out[69]: 0.9026906492316511
```

```
In [70]: 1 accuracy_score(Y_test,pred)
```

```
Out[70]: 0.906276150627615
```

Let us clean the test data and then predict values from the test data

```
In [71]: 1 # lets drop Id variable.  
2 test.drop('r4t3',axis=1,inplace=True)  
3 test.drop(['Id','idhogar'],axis=1,inplace=True)  
4 test['dependency']=test['dependency'].apply(map)  
5 test['edjefe']=test['edjefe'].apply(map)  
6 test['edjefa']=test['edjefa'].apply(map)
```

```
In [72]: 1 test['v2a1'].fillna(0,inplace=True)  
2 test['v18q1'].fillna(0,inplace=True)  
3  
4 test.drop(['tipovivi3', 'v18q','rez_esc','elimbasu5'],axis=1,inplace=True)  
5  
6 train['meaneduc'].fillna(np.mean(train['meaneduc']),inplace=True)  
7 train['SQBmeaned'].fillna(np.mean(train['SQBmeaned']),inplace=True)
```

```
In [73]: 1 test_data=test[Top50Features]
```

```
In [74]: 1 test_data.isna().sum().value_counts()
```

```
Out[74]: 0      48  
31      2  
dtype: int64
```

```
In [76]: 1 test_data.SQBmeaned.fillna(np.mean(test_data['SQBmeaned']),inplace=True)  
2 test_data.meaneduc.fillna(np.mean(test_data['meaneduc']),inplace=True)  
3  
4 Test_data_1=SS.fit_transform(test_data)  
5 X_data_1=pd.DataFrame(Test_data_1)  
6 test_prediction=Model_1.predict(test_data)  
7 test_prediction
```

```
Out[76]: array([4, 4, 4, ..., 4, 4, 4])
```

```
In [76]: 1 test_data.SQBmeaned.fillna(np.mean(test_data['SQBmeaned']),inplace=True)  
2 test_data.meaneduc.fillna(np.mean(test_data['meaneduc']),inplace=True)  
3  
4 Test_data_1=SS.fit_transform(test_data)  
5 X_data_1=pd.DataFrame(Test_data_1)  
6 test_prediction=Model_1.predict(test_data)  
7 test_prediction
```

```
Out[76]: array([4, 4, 4, ..., 4, 4, 4])
```

Interpretation : Above is our prediction for test data.

Conclusion :

Using RandomForest Classifier we can predict test_data with accuracy of 90%.

