Problem

Identify the level of income qualification needed for the families in Latin America.

Problem Statement Scenario:

Many social programs have a hard time ensuring that the right people are given enough aid. It's tricky when a program focuses on the poorest segment of the population. This segment of the population can't provide the necessary income and expense records to prove that they qualify.

In Latin America, a popular method called Proxy Means Test (PMT) uses an algorithm to verify income qualification. With PMT, agencies use a model that considers a family's observable household attributes like the material of their walls and ceiling or the assets found in their homes to classify them and predict their level of need.

While this is an improvement, accuracy remains a problem as the region's population grows and poverty declines.

The Inter-American Development Bank (IDB)believes that new methods beyond traditional econometrics, based on a dataset of Costa Rican household characteristics, might help improve PMT's performance. Following actions should be performed:

1.Identify the output variable.

2. Understand the type of data.

3.Check if there are any biases in your dataset.

4.Check whether all members of the house have the same poverty level.

5.Check if there is a house without a family head.

6.Set poverty level of the members and the head of the house within a family.

7.Count how many null values are existing in columns.

8.Remove null value rows of the target variable.

9.Predict the accuracy using random forest classifier.

10.Check the accuracy using random forest with cross validation.

In [79]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
sns.set()
import warnings
warnings.filterwarnings('ignore')
```

In [80]:
```
Income_DF=pd.read_csv('Income-qualification-project.csv')
```

In [81]:
```
Income=pd.DataFrame(Income_DF)
```

In [82]:
```
Income_train=pd.read_csv('train_income.csv')
Income_test=pd.read_csv('test_income.csv')
Income_train.head()
```

Out[82]:

	Id	v2a1	hacdor	rooms	hacapo	v14a	refrig	v18q	v18q1	r4h1	...	SQBesco
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 rows × 143 columns

In [83]:
```
Income_test.head()
```

Out[83]:

	Id	v2a1	hacdor	rooms	hacapo	v14a	refrig	v18q	v18q1	r4h1	...	age	SQI
0	ID_2f6873615	NaN	0	5	0	1	1	0	NaN	1	...	4	
1	ID_1c78846d2	NaN	0	5	0	1	1	0	NaN	1	...	41	
2	ID_e5442cf6a	NaN	0	5	0	1	1	0	NaN	1	...	41	
3	ID_a8db26a79	NaN	0	14	0	1	1	1	1.0	0	...	59	
4	ID_a62966799	175000.0	0	4	0	1	1	1	1.0	0	...	18	

5 rows × 142 columns

In [84]:
```
dir(Income_train)
```

```
 '__annotations__',
 '__array__',
 '__array_priority__',
 '__array_wrap__',
 '__bool__',
 '__class__',
 '__contains__',
 '__copy__',
 '__deepcopy__',
 '__delattr__',
 '__delitem__',
 '__dict__',
 '__dir__',
 '__div__',
 '__doc__',
 '__eq__',
 '__finalize__',
 '__floordiv__',
 '__format__',
 '__ge__',
```

1. Explore the Dataset

In [85]:
```
print('Shape of train dataset is {}'.format(Income_train.shape))
print('Shape of test dataset is {}'.format(Income_test.shape))
```

```
Shape of train dataset is (9557, 143)
Shape of test dataset is (23856, 142)
```

2.Identifying the Target

In [86]:
```
for i in Income_train.columns:
    if i not in Income_test.columns:
        print("Our Target variable is {}".format(i))
```

```
Our Target variable is Target
```

Ans of (1). Our Target variable is Target

3.Understanding the datatype

In [87]:
```
print(Income_train.dtypes.value_counts())
print(Income_train.info())
```

```
int64      130
float64      8
object       5
dtype: int64
<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
```

Ans of (2). We have mixed data types. Specified as below:

float64 : 8 variables

int64 : 130 variables

object :5 variables

In [88]:
```
#lets explore each different types of datasets
for i in Income_train.columns:
    a=Income_train[i].dtype
    if a == 'object':
        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

In [89]:
```
# lets drop Id variable.

Income_train.drop(['Id','idhogar'],axis=1,inplace=True)
```

In [90]:
```
Income_train['dependency'].value_counts()
```

Out[90]:
```
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
1.3333334       84
.40000001       84
2.5             77
5               24
.80000001       18
1.25            18
3.5             18
2.25            13
.71428573       12
1.2             11
.22222222       11
1.75            11
.83333331       11
.2857143         9
1.6666666        8
.60000002        8
6                7
.16666667        7
Name: dependency, dtype: int64
```

Lets Convert object variables into numerical data.

In [91]:
```
def map(i):

    if i=='yes':
        return(float(1))
    elif i=='no':
        return(float(0))
    else:
        return(float(i))
```

In [92]:
```
Income_train['dependency']=Income_train['dependency'].apply(map)
```

In [93]:
```
for i in Income_train.columns:
    a=Income_train[i].dtype
    if a == 'object':
        print(i)
```

```
edjefe
edjefa
```

In [94]:
```
Income_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9557 entries, 0 to 9556
Columns: 141 entries, v2a1 to Target
dtypes: float64(9), int64(130), object(2)
memory usage: 10.3+ MB
```

In [95]:
```
Income_train['edjefe']=Income_train['edjefe'].apply(map)
Income_train['edjefa']=Income_train['edjefa'].apply(map)
```

In [96]:
```
Income_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
```

Now we have made all the data in numeric

Now we'll identify variable with 0 variance

In [97]:
```
var_df=pd.DataFrame(np.var(Income_train,0),columns=['variance'])
var_df.sort_values(by='variance').head(15)
print('Below are columns with variance 0.')
col=list((var_df[var_df['variance']==0]).index)
print(col)
```

```
Below are columns with variance 0.
['elimbasu5']
```

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

Now we'll check if there is any bias in the dataset

In [98]:
```
contingency_tab=pd.crosstab(Income_train['r4t3'],Income_train['hogar_total'])
Observed_Values=contingency_tab.values
import scipy.stats
b=scipy.stats.chi2_contingency(contingency_tab)
Expected_Values = b[3]
no_of_rows=len(contingency_tab.iloc[0:2,0])
no_of_columns=len(contingency_tab.iloc[0,0:2])
df=(no_of_rows-1)*(no_of_columns-1)
print("Degree of Freedom:-",df)
from scipy.stats import chi2
chi_square=sum([(o-e)**2./e for o,e in zip(Observed_Values,Expected_Values)])
chi_square_statistic=chi_square[0]+chi_square[1]
print("chi-square statistic:-",chi_square_statistic)
alpha=0.05
critical_value=chi2.ppf(q=1-alpha,df=df)
print('critical_value:',critical_value)
p_value=1-chi2.cdf(x=chi_square_statistic,df=df)
print('p-value:',p_value)
print('Significance level: ',alpha)
print('Degree of Freedom: ',df)
print('chi-square statistic:',chi_square_statistic)
print('critical_value:',critical_value)
print('p-value:',p_value)
if chi_square_statistic>=critical_value:
    print("Reject H0,There is a relationship between 2 categorical variables")
else:
    print("Retain H0,There is no relationship between 2 categorical variables")

if p_value<=alpha:
    print("Reject H0,There is a relationship between 2 categorical variables")
else:
    print("Retain H0,There is no relationship between 2 categorical variables")
```

```
Degree of Freedom:- 1
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
Reject H0,There is a relationship between 2 categorical variables
Reject H0,There is a relationship between 2 categorical variables
```

Therefore,variables ('r4t3','hogar_total') have relationship between them. For good result we can use any one of them.

In [99]:
```
contingency_tab=pd.crosstab(Income_train['tipovivi3'],Income_train['v2a1'])
Observed_Values=contingency_tab.values
import scipy.stats
b=scipy.stats.chi2_contingency(contingency_tab)
Expected_Values = b[3]
no_of_rows=len(contingency_tab.iloc[0:2,0])
no_of_columns=len(contingency_tab.iloc[0,0:2])
df=(no_of_rows-1)*(no_of_columns-1)
print("Degree of Freedom:-",df)
from scipy.stats import chi2
chi_square=sum([(o-e)**2./e for o,e in zip(Observed_Values,Expected_Values)])
chi_square_statistic=chi_square[0]+chi_square[1]
print("chi-square statistic:-",chi_square_statistic)
alpha=0.05
critical_value=chi2.ppf(q=1-alpha,df=df)
print('critical_value:',critical_value)
p_value=1-chi2.cdf(x=chi_square_statistic,df=df)
print('p-value:',p_value)
print('Significance level: ',alpha)
print('Degree of Freedom: ',df)
print('chi-square statistic:',chi_square_statistic)
print('critical_value:',critical_value)
print('p-value:',p_value)
if chi_square_statistic>=critical_value:
    print("Reject H0,There is a relationship between 2 categorical variables")
else:
    print("Retain H0,There is no relationship between 2 categorical variables")
if p_value<=alpha:
    print("Reject H0,There is a relationship between 2 categorical variables")
else:
    print("Retain H0,There is no relationship between 2 categorical variables")
```

```
Degree of Freedom:- 1
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
Reject H0,There is a relationship between 2 categorical variables
Reject H0,There is a relationship between 2 categorical variables
```

Variables ('tipovivi3','v2a1') have relationship between them. For good result we can use any one of them.

In [100]:
```
contingency_tab=pd.crosstab(Income_train['v18q'],Income_train['v18q1'])
Observed_Values=contingency_tab.values
import scipy.stats
b=scipy.stats.chi2_contingency(contingency_tab)
Expected_Values = b[3]
no_of_rows=len(contingency_tab.iloc[0:2,0])
no_of_columns=len(contingency_tab.iloc[0,0:2])
df=(no_of_rows-1)*(no_of_columns-1)
print("Degree of Freedom:-",df)
from scipy.stats import chi2
chi_square=sum([(o-e)**2./e for o,e in zip(Observed_Values,Expected_Values)])
chi_square_statistic=chi_square[0]+chi_square[1]
print("chi-square statistic:-",chi_square_statistic)
alpha=0.05
critical_value=chi2.ppf(q=1-alpha,df=df)
print('critical_value:',critical_value)
p_value=1-chi2.cdf(x=chi_square_statistic,df=df)
print('p-value:',p_value)
print('Significance level: ',alpha)
print('Degree of Freedom: ',df)
print('chi-square statistic:',chi_square_statistic)
print('critical_value:',critical_value)
print('p-value:',p_value)
if chi_square_statistic>=critical_value:
    print("Reject H0,There is a relationship between 2 categorical variables")
else:
    print("Retain H0,There is no relationship between 2 categorical variables")

if p_value<=alpha:
    print("Reject H0,There is a relationship between 2 categorical variables")
else:
    print("Retain H0,There is no relationship between 2 categorical variables")
```

```
Degree of Freedom:- 0
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
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.

Ans(3).Therefore, there is bias in our dataset.

In [101]:
```
Income_train.drop('r4t3',axis=1,inplace=True)
```

Now we would check if there is a house without a family

head.

"parentesco1" =1 if household head

In [102]:
```
Income_train.parentesco1.value_counts()
```

Out[102]:
```
0    6584
1    2973
Name: parentesco1, dtype: int64
```

Ans(5).Thus, we have houses without family head

In [103]:
```
pd.crosstab(Income_train['edjefa'],Income_train['edjefe'])
```

Out[103]:

edjefe edjefa	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
0.0	435	123	194	307	137	222	1845	234	257	486	...	113	103	208	285	134	202
1.0	69	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
3.0	152	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
5.0	176	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
7.0	179	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
9.0	237	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
11.0	399	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
13.0	52	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
15.0	188	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
17.0	76	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
19.0	4	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
21.0	5	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 [104]:
```
Income_train.isna().sum().value_counts()
```

Out[104]:
```
0       135
5         2
7928      1
6860      1
7342      1
dtype: int64
```

Lets Identify number of null values in Target variable

In [105]:
```
Income_train['Target'].isna().sum()
```

Out[105]:
```
0
```

Ans(8). There are no null values in Target variable. Now lets proceed further and identify and fillna of other variable.

In [106]:
```
float_col=[]
for i in Income_train.columns:
    a=Income_train[i].dtype
    if a == 'float64':
        float_col.append(i)
print(float_col)
```

```
['v2a1', 'v18q1', 'rez_esc', 'dependency', 'edjefe', 'edjefa', 'meaneduc', 'ove
rcrowding', 'SQBovercrowding', 'SQBdependency', 'SQBmeaned']
```

In [107]:
```
Income_train[float_col].isna().sum()
```

Out[107]:
```
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 [108]:
```
Income_train['v18q1'].value_counts()
```

Out[108]:
```
1.0    1586
2.0     444
3.0     129
4.0      37
5.0      13
6.0       6
Name: v18q1, dtype: int64
```

In [109]:
```
pd.crosstab(Income_train['tipovivi1'],Income_train['v2a1'])
```

Out[109]:

v2a1 tipovivi1	0.0	12000.0	13000.0	14000.0	15000.0	16000.0	17000.0	20000.0	23000.0	25000.0	...
0	29	3	4	3	3	2	4	22	5	21	...

1 rows × 157 columns

In [110]:
```
pd.crosstab(Income_train['v18q1'],Income_train['v18q'])
```

Out[110]:

v18q v18q1	1
1.0	1586
2.0	444
3.0	129
4.0	37
5.0	13
6.0	6

'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.

Istead 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 [111]:
```
Income_train['v2a1'].fillna(0,inplace=True)
Income_train['v18q1'].fillna(0,inplace=True)
```

In [112]:
```
Income_train.drop(['tipovivi3', 'v18q','rez_esc','elimbasu5'],axis=1,inplace=True
```

In [113]:
```
Income_train['meaneduc'].fillna(np.mean(Income_train['meaneduc']),inplace=True)
Income_train['SQBmeaned'].fillna(np.mean(Income_train['SQBmeaned']),inplace=True)
print(Income_train.isna().sum().value_counts())
```

```
0    136
dtype: int64
```

In [114]:
```
int_col=[]
for i in Income_train.columns:
    a=Income_train[i].dtype
    if a == 'int64':
        int_col.append(i)
print(int_col)
```

```
['hacdor', 'rooms', 'hacapo', 'v14a', 'refrig', 'r4h1', 'r4h2', 'r4h3', 'r4m1',
'r4m2', 'r4m3', 'r4t1', 'r4t2', 'tamhog', 'tamviv', 'escolari', 'hhsize', 'pare
dblolad', 'paredzocalo', 'paredpreb', 'pareddes', 'paredmad', 'paredzinc', 'par
edfibras', 'paredother', 'pisomoscer', 'pisocemento', 'pisoother', 'pisonatur',
'pisonotiene', 'pisomadera', 'techozinc', 'techoentrepiso', 'techocane', 'techo
otro', 'cielorazo', 'abastaguadentro', 'abastaguafuera', 'abastaguano', 'publi
c', 'planpri', 'noelec', 'coopele', 'sanitario1', 'sanitario2', 'sanitario3',
'sanitario5', 'sanitario6', 'energcocinar1', 'energcocinar2', 'energcocinar3',
'energcocinar4', 'elimbasu1', 'elimbasu2', 'elimbasu3', 'elimbasu4', 'elimbasu
6', 'epared1', 'epared2', 'epared3', 'etecho1', 'etecho2', 'etecho3', 'eviv1',
'eviv2', 'eviv3', 'dis', 'male', 'female', 'estadocivil1', 'estadocivil2', 'est
adocivil3', 'estadocivil4', 'estadocivil5', 'estadocivil6', 'estadocivil7', 'pa
rentesco1', 'parentesco2', 'parentesco3', 'parentesco4', 'parentesco5', 'parent
esco6', 'parentesco7', 'parentesco8', 'parentesco9', 'parentesco10', 'parentesc
o11', 'parentesco12', 'hogar_nin', 'hogar_adul', 'hogar_mayor', 'hogar_total',
'instlevel1', 'instlevel2', 'instlevel3', 'instlevel4', 'instlevel5', 'instleve
l6', 'instlevel7', 'instlevel8', 'instlevel9', 'bedrooms', 'tipovivi1', 'tipovi
vi2', 'tipovivi4', 'tipovivi5', 'computer', 'television', 'mobilephone', 'qmobi
lephone', 'lugar1', 'lugar2', 'lugar3', 'lugar4', 'lugar5', 'lugar6', 'area1',
'area2', 'age', 'SQBescolari', 'SQBage', 'SQBhogar_total', 'SQBedjefe', 'SQBhog
ar_nin', 'agesq', 'Target']
```

In [115]:
```
Income_train[int_col].isna().sum().value_counts()
```

Out[115]:
```
0    126
dtype: int64
```

In [116]:
```
Income_train.isna().sum().value_counts()
```

Out[116]:
```
0    136
dtype: int64
```

Now there is no null value in our datset.

In [117]:
```
Income_train.Target.value_counts()
```

Out[117]:
```
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 [118]:
```
Poverty_level=Income_train[Income_train['v2a1'] !=0]
```

In [119]:
```
Poverty_level.shape
```

Out[119]:
```
(2668, 136)
```

In [120]:
```
poverty_level=Poverty_level.groupby('area1')['v2a1'].apply(np.median)
poverty_level
```

Out[120]:
```
area1
0     80000.0
1    140000.0
Name: v2a1, dtype: float64
```

Ans(6).

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

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

In [121]:
```
def povert(x):
    if x<8000:
        return('Below poverty level')

    elif x>140000:
        return('Above poverty level')
    elif x<140000:
        return('Below poverty level: Ur-ban ; Above poverty level : Rural ')
```

In [122]:
```
c=Poverty_level['v2a1'].apply(povert)
c.shape
```

Out[122]:
```
(2668,)
```

In [123]:
```
pd.crosstab(c,Poverty_level['area1'])
```

Out[123]:

area1	0	1
v2a1		
Above poverty level	139	1103
Below poverty level: Ur-ban ; Above poverty level : Rural	306	1081

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 [124]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
```

In [125]:
```
X_data=Income_train.drop('Target',axis=1)
Y_data=Income_train.Target
```

In [126]:
```
X_data_col=X_data.columns
```

Applying Standard Scalling to dataset

In [127]:
```
from sklearn.preprocessing import StandardScaler
SS=StandardScaler()
X_data_1=SS.fit_transform(X_data)
X_data_1=pd.DataFrame(X_data_1,columns=X_data_col)
```

Towards Model Fitting

In [128]:
```
X_train,X_test,Y_train,Y_test=train_test_split(X_data_1,Y_data,test_size=0.25,str
```

Lets identify best parameters for our model using GridSearchCv

In [129]:
```
from sklearn.pipeline import Pipeline
from sklearn.model_selection import GridSearchCV

rfc=RandomForestClassifier(random_state=0)
parameters={'n_estimators':[10,50,100,300],'max_depth':[3,5,10,15]}
grid=zip([rfc],[parameters])

best_=None

for i, j in grid:
    a=GridSearchCV(i,param_grid=j,cv=3,n_jobs=1)
    a.fit(X_train,Y_train)
    if best_ is None:
        best_=a
    elif a.best_score_>best_.best_score_:
        best_=a


print ("Best CV Score",best_.best_score_)
print ("Model Parameters",best_.best_params_)
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_st
ate=0)
```

In [130]:
```
RFC=best_.best_estimator_
Model=RFC.fit(X_train,Y_train)
pred=Model.predict(X_test)
```

In [131]:
```
print('Model Score of train data : {}'.format(Model.score(X_train,Y_train)))
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 [132]:
```
Important_features=pd.DataFrame(Model.feature_importances_,X_data_col,columns=['f
```

In [133]:
```
Top50Features=Important_features.sort_values(by='feature_importance',ascending=Fa
Top50Features
```

Out[133]:
```
Index(['SQBmeaned', 'meaneduc', 'SQBdependency', 'dependency', 'overcrowding',
       'SQBovercrowding', 'qmobilephone', 'SQBhogar_nin', 'SQBedjefe',
       'edjefe', 'hogar_nin', 'rooms', 'cielorazo', 'r4t1', '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', 'energcocinar2', 'energcocinar3', 'epared2',
       'television', 'area2', 'area1'],
      dtype='object')
```

In [134]:
```
for i in Top50Features:
    if i not in X_data_col:
        print(i)
```

In [135]:
```
X_data_Top50=X_data[Top50Features]
```

In [136]:
```
X_train,X_test,Y_train,Y_test=train_test_split(X_data_Top50,Y_data,test_size=0.25
```

In [137]:
```
Model_1=RFC.fit(X_train,Y_train)
pred=Model_1.predict(X_test)
```

In [138]:
```
from sklearn.metrics import confusion_matrix,f1_score,accuracy_score
```

In [139]:
```
confusion_matrix(Y_test,pred)
```

Out[139]:
```
array([[ 143,   17,    0,   29],
       [   8,  324,    4,   63],
       [   1,   12,  214,   75],
       [   2,   10,    3, 1485]], dtype=int64)
```

In [140]:
```
f1_score(Y_test,pred,average='weighted')
```

Out[140]:
```
0.9026906492316511
```

In [141]:
```
accuracy_score(Y_test,pred)
```

Out[141]:
```
0.906276150627615
```

Lets apply cleaning on test data and then find prediction for that.

In [142]:
```
# lets drop Id variable.
Income_test.drop('r4t3',axis=1,inplace=True)
Income_test.drop(['Id','idhogar'],axis=1,inplace=True)
Income_test['dependency']=Income_test['dependency'].apply(map)
Income_test['edjefe']=Income_test['edjefe'].apply(map)
Income_test['edjefa']=Income_test['edjefa'].apply(map)
```

In [143]:
```
Income_test['v2a1'].fillna(0,inplace=True)
Income_test['v18q1'].fillna(0,inplace=True)
```

In [144]:
```
Income_test.drop(['tipovivi3', 'v18q','rez_esc','elimbasu5'],axis=1,inplace=True)
Income_train['meaneduc'].fillna(np.mean(Income_train['meaneduc']),inplace=True)
Income_train['SQBmeaned'].fillna(np.mean(Income_train['SQBmeaned']),inplace=True)
```

In [145]:
```
Income_test_data=Income_test[Top50Features]
Income_test_data.isna().sum().value_counts()
```

Out[145]:
```
0     48
31     2
dtype: int64
```

In [146]:
```
Income_test_data.SQBmeaned.fillna(np.mean(Income_test_data['SQBmeaned']),inplace=
Income_test_data.meaneduc.fillna(np.mean(Income_test_data['meaneduc']),inplace=Tr
Test_data_1=SS.fit_transform(Income_test_data)
X_data_1=pd.DataFrame(Test_data_1)
test_prediction=Model_1.predict(Income_test_data)
test_prediction
```

Out[146]:
```
array([4, 4, 4, ..., 4, 4, 4], dtype=int64)
```

Ans(10). Using RandomForest Classifier we can predict test_data with accuracy of 90%.