|  |  |
| --- | --- |
|  | Income Qualification |
|  |  |
|  | Margil Shah  MACHINE LEARNING  5/26/21 |

Source Code – Full Project

Contents

[DESCRIPTION 2](#_Toc72922566)

[Problem Statement Scenario: 2](#_Toc72922567)

[Screen Shots: 2](#_Toc72922568)

[Load data 3](#_Toc72922569)

[Let us explore our dataset before moving further 3](#_Toc72922570)

[Analysis 1: Identify the output variable. 3](#_Toc72922571)

[Analysis 2: Understand the type of data. 3](#_Toc72922572)

[We have mixed data types. Specified as below: 4](#_Toc72922573)

[Lets Convert object variables into numerical data. 5](#_Toc72922574)

[Analysis 3: Check if there are any biases in your dataset. 6](#_Toc72922575)

[Analysis 4: Check whether all members of the house have the same poverty level. 11](#_Toc72922576)

[Analysis 5: Check if there is a house without a family head. 11](#_Toc72922577)

[Analysis 7: Count how many null values are existing in columns. 13](#_Toc72922578)

[Analysis 6: Set poverty level of the members and the head of the house within a family. 16](#_Toc72922579)

[Analysis 9: Predict the accuracy using random forest classifier. 17](#_Toc72922580)

[Applying Standard Scalling to dataset 17](#_Toc72922581)

[Now we will proceed to model fitting 18](#_Toc72922582)

[Analysis 10: Check the accuracy using random forest with cross validation. 19](#_Toc72922583)

[Lets apply cleaning on test data and then find prediction for that. 20](#_Toc72922584)

[Conclusion :¶ 20](#_Toc72922585)

# DESCRIPTION

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:

* Identify the output variable.
* Understand the type of data.
* Check if there are any biases in your dataset.
* Check whether all members of the house have the same poverty level.
* Check if there is a house without a family head.
* Set poverty level of the members and the head of the house within a family.
* Count how many null values are existing in columns.
* Remove null value rows of the target variable.
* Predict the accuracy using random forest classifier.
* Check the accuracy using random forest with cross validation.

Find the datasets here.

Find the data dictionary here:

# Screen Shots:

Before proceed let’s upload the datasets in SimpliLearn Lab

test.csv

train.csv

## Load data

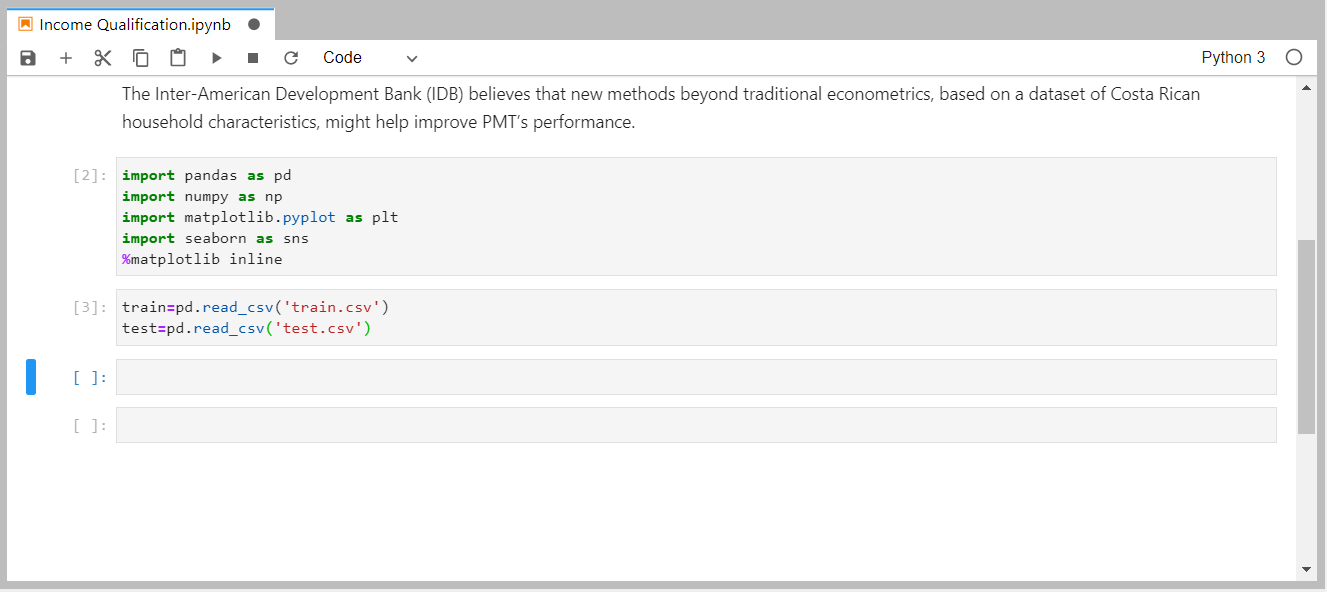


Figure 1 Load train and test Data

## Let us explore our dataset before moving further

print('Shape of train dataset is {}'.format(train.shape))

print('Shape of test dataset is {}'.format(test.shape))

## Analysis 1: Identify the output variable.

for i in train.columns:

if i not in test.columns:

print("Our Target variable is {}".format(i))

Output:

**Our Target variable is Target**

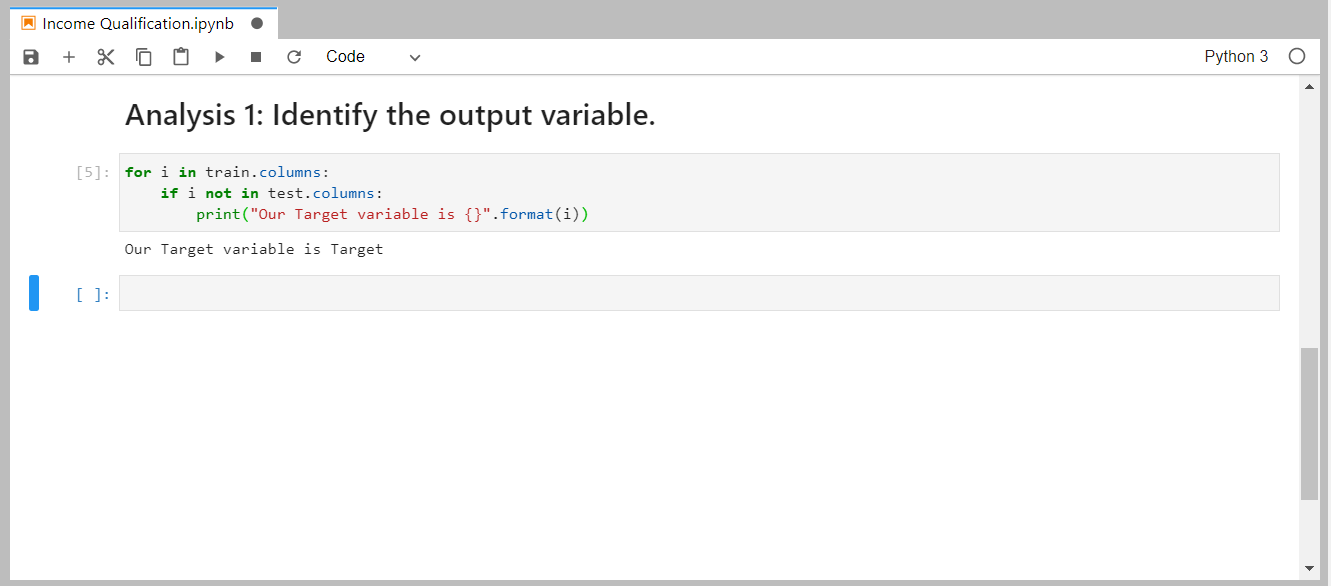


Figure 2 Identify the output variable.

## Analysis 2: Understand the type of data.

print(train.dtypes.value\_counts())

print(train.info())

### We have mixed data types. Specified as below:

* float64 : 8 variables
* int64 : 130 vriables
* object :5 variables

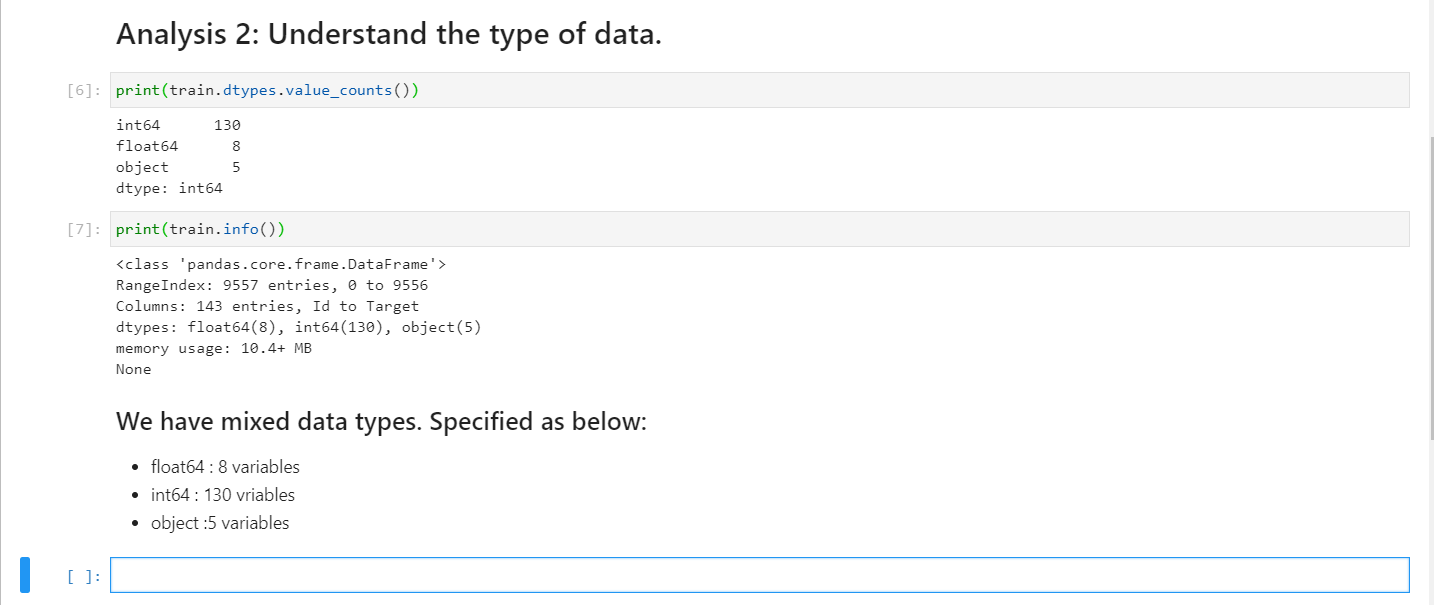


Figure 3 Understand the type of data.

#lets explore each different types of datasets

for i in train.columns:

a=train[i].dtype

if a == 'object':

print(i)

Id

idhogar

dependency

edjefe

edjefa

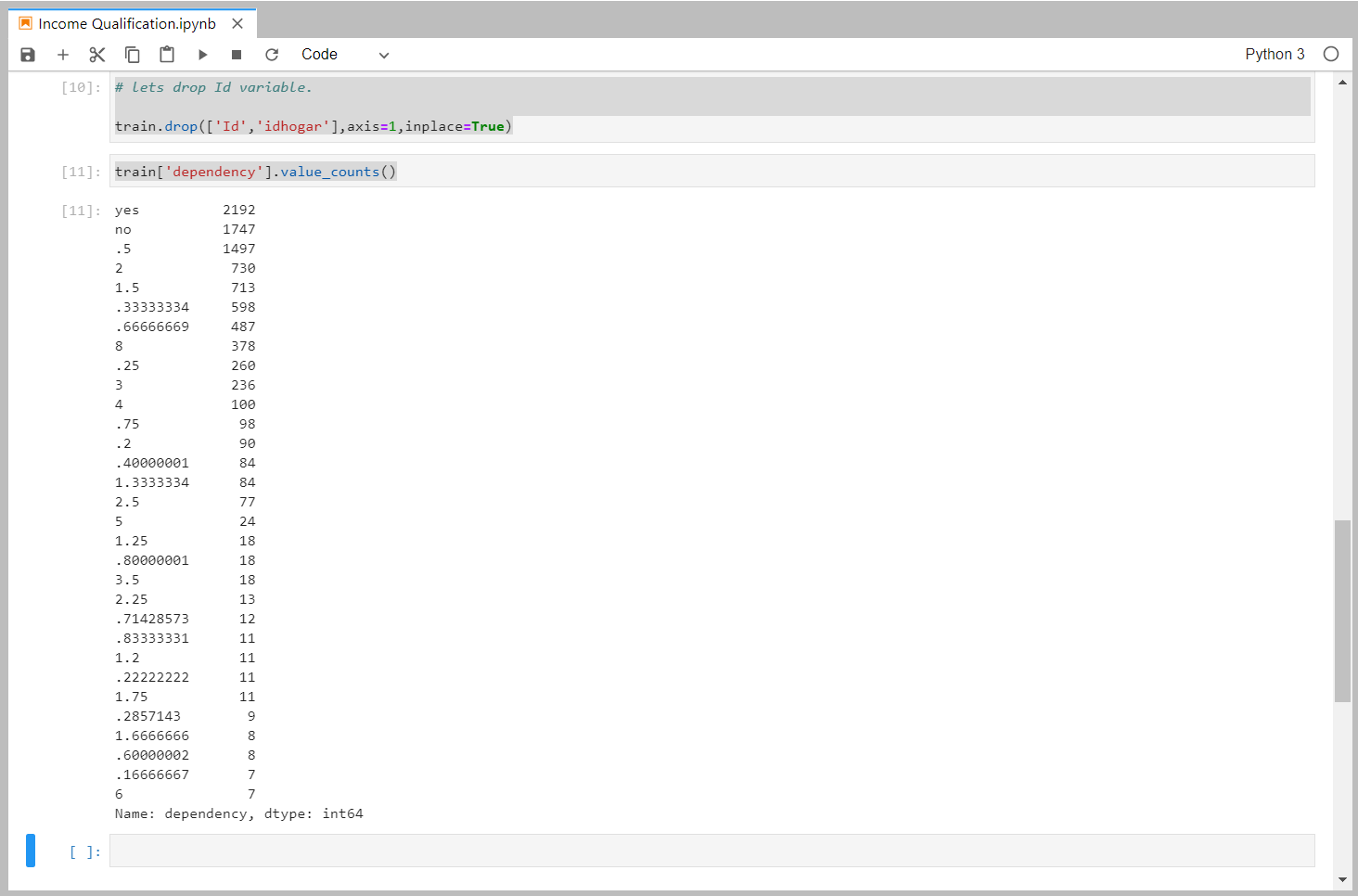
Below is Data dictionary for above object variables

* 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

# lets drop Id variable.

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

train['dependency'].value\_counts()



### Lets Convert object variables into numerical data.

def map(i):

if i=='yes':

return(float(1))

elif i=='no':

return(float(0))

else:

return(float(i))

train['dependency']=train['dependency'].apply(map)

for i in train.columns:

a=train[i].dtype

if a == 'object':

print(i)

edjefe

edjefa

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

train['edjefe']=train['edjefe'].apply(map)

train['edjefa']=train['edjefa'].apply(map)

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 all data is in numerical form

**Lets identify variable with 0 varinace**

var\_df=pd.DataFrame(np.var(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']

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 variablity in dataset therefor we will drop this variable**

## Analysis 3: Check if there are any biases in your dataset.

contingency\_tab=pd.crosstab(train['r4t3'],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")

**Output:**

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

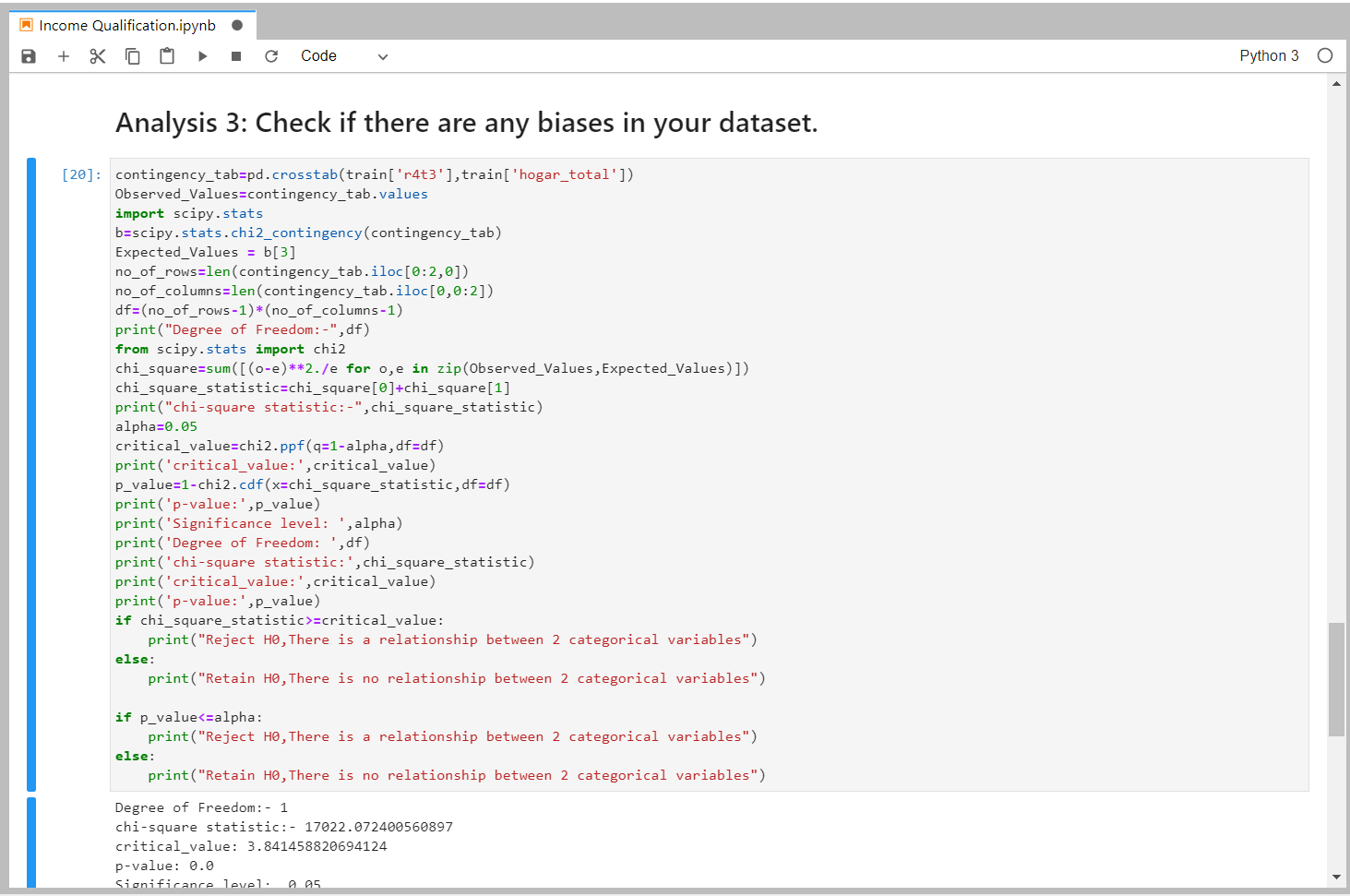


Figure 4 Variables ('r4t3','hogar\_total') have relationship between them.

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

contingency\_tab=pd.crosstab(train['tipovivi3'],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")

Output:

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

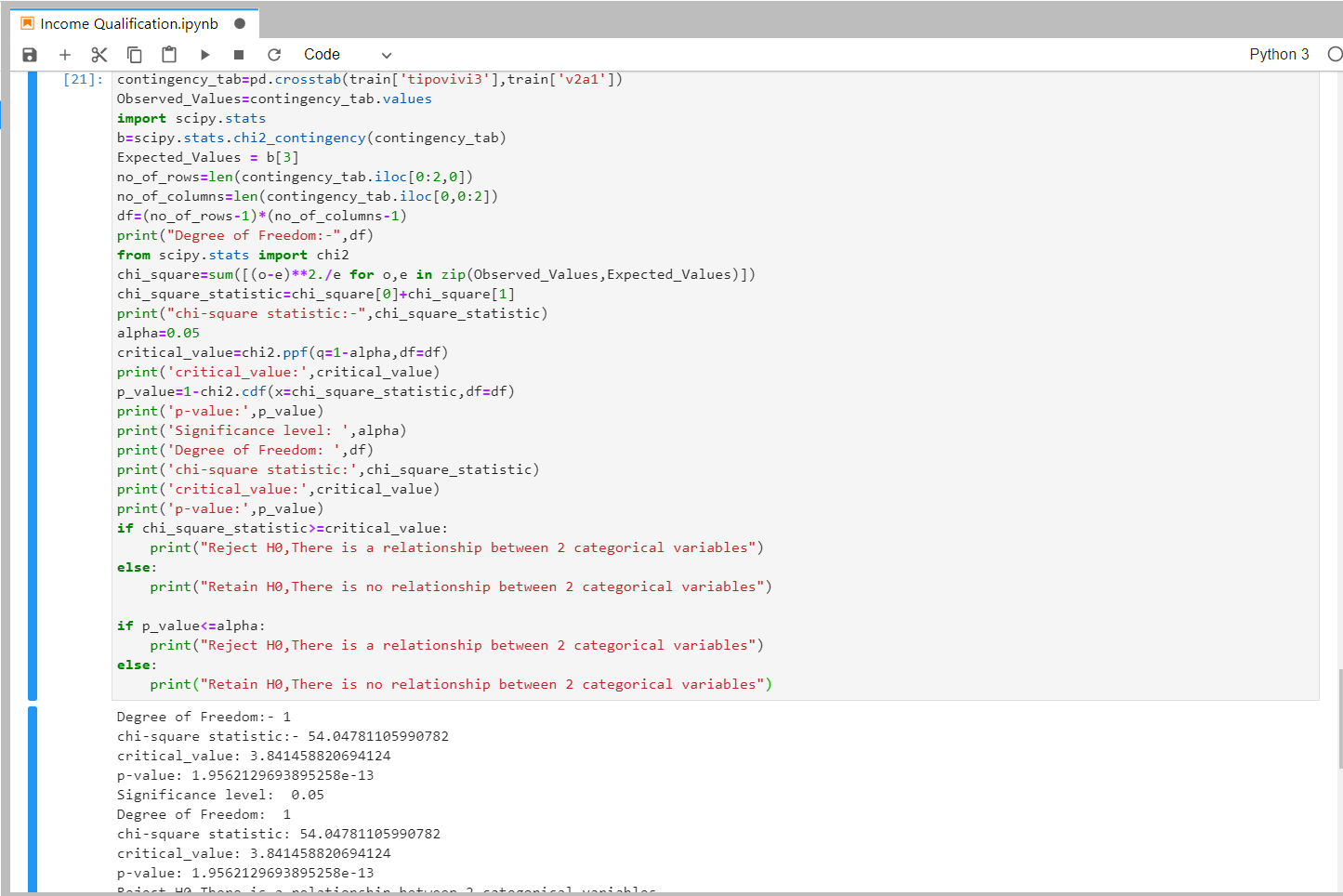


Figure 5 Variables ('tipovivi3','v2a1') have relationship between them.

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

contingency\_tab=pd.crosstab(train['v18q'],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")

**Output:**

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

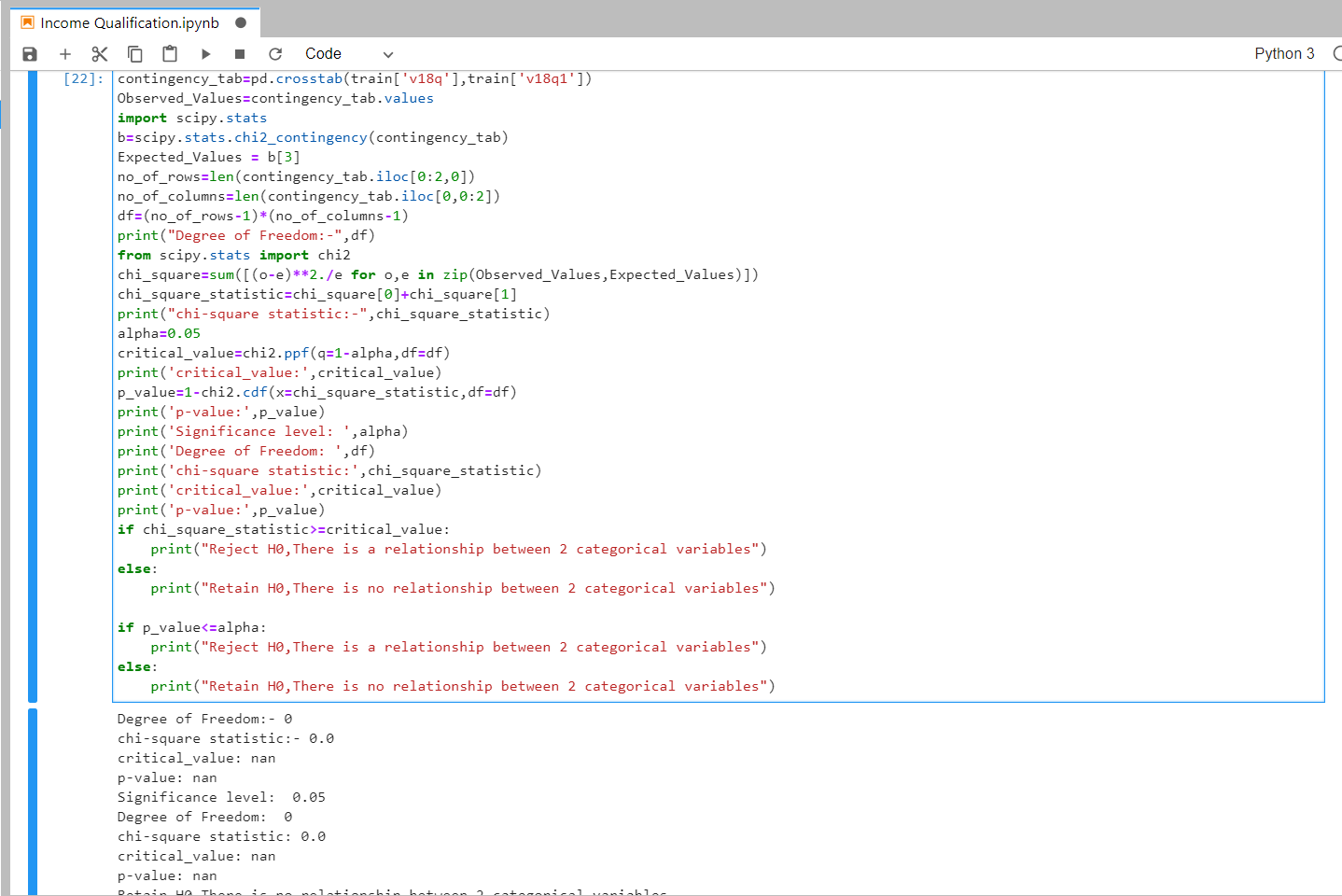


Figure 5 Variables ('v18q','v18q1') have relationship between them.

**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.**

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

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

## Analysis 5: Check if there is a house without a family head.

"parentesco1" =1 if household head

train.parentesco1.value\_counts()

0 6584

1 2973

Name: parentesco1, dtype: int64

pd.crosstab(train['edjefa'],train['edjefe'])

| **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** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **edjefa** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **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 | 0 |
| **2.0** | 84 | 0 | 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 | 0 |
| **4.0** | 136 | 0 | 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 | 0 |
| **6.0** | 947 | 0 | 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 | 0 |
| **8.0** | 217 | 0 | 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 | 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.**

## Analysis 7: Count how many null values are existing in columns.

**train.isna().sum().value\_counts()**

0 135

5 2

7928 1

6860 1

7342 1

dtype: int64

Lets Identify number of null values in Target variable

**train['Target'].isna().sum()**

0

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

**float\_col=[]**

**for i in train.columns:**

**a=train[i].dtype**

**if a == 'float64':**

**float\_col.append(i)**

**print(float\_col)**

['v2a1', 'v18q1', 'rez\_esc', 'dependency', 'edjefe', 'edjefa', 'meaneduc', 'overcrowding', 'SQBovercrowding', 'SQBdependency', 'SQBmeaned']

**train[float\_col].isna().sum()**

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

**train['v18q1'].value\_counts()**

1.0 1586

2.0 444

3.0 129

4.0 37

5.0 13

6.0 6

Name: v18q1, dtype: int64

**pd.crosstab(train['tipovivi1'],train['v2a1'])**

| **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** | **800000.0** | **855810.0** | **1000000.0** | **2353477.0** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **tipovivi1** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **0** | 29 | 3 | 4 | 3 | 3 | 2 | 4 | 22 | 5 | 21 | ... | 25 | 11 | 3 | 3 | 7 | 3 | 4 | 11 | 7 | 2 |

1 rows × 157 columns

**pd.crosstab(train['v18q1'],train['v18q'])**

| **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.***

***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

**train['v2a1'].fillna(0,inplace=True)**

**train['v18q1'].fillna(0,inplace=True)**

**train.drop(['tipovivi3', 'v18q','rez\_esc','elimbasu5'],axis=1,inplace=True)**

**train['meaneduc'].fillna(np.mean(train['meaneduc']),inplace=True)**

**train['SQBmeaned'].fillna(np.mean(train['SQBmeaned']),inplace=True)**

**print(train.isna().sum().value\_counts())**

0 136

dtype: int64

int\_col=[]

for i in train.columns:

a=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', 'paredblolad', 'paredzocalo', 'paredpreb', 'pareddes', 'paredmad', 'paredzinc', 'paredfibras', 'paredother', 'pisomoscer', 'pisocemento', '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', 'elimbasu6', 'epared1', 'epared2', 'epared3', 'etecho1', 'etecho2', 'etecho3', 'eviv1', 'eviv2', 'eviv3', 'dis', 'male', 'female', 'estadocivil1', 'estadocivil2', 'estadocivil3', 'estadocivil4', 'estadocivil5', 'estadocivil6', 'estadocivil7', 'parentesco1', 'parentesco2', 'parentesco3', 'parentesco4', 'parentesco5', 'parentesco6', 'parentesco7', 'parentesco8', 'parentesco9', 'parentesco10', 'parentesco11', 'parentesco12', 'hogar\_nin', 'hogar\_adul', 'hogar\_mayor', 'hogar\_total', 'instlevel1', 'instlevel2', 'instlevel3', 'instlevel4', 'instlevel5', 'instlevel6', 'instlevel7', 'instlevel8', 'instlevel9', 'bedrooms', 'tipovivi1', 'tipovivi2', 'tipovivi4', 'tipovivi5', 'computer', 'television', 'mobilephone', 'qmobilephone', 'lugar1', 'lugar2', 'lugar3', 'lugar4', 'lugar5', 'lugar6', 'area1', 'area2', 'age', 'SQBescolari', 'SQBage', 'SQBhogar\_total', 'SQBedjefe', 'SQBhogar\_nin', 'agesq', 'Target']

train[int\_col].isna().sum().value\_counts()

0 126

dtype: int64

Interpretation : **Now there is no null value in our datset.**

train.Target.value\_counts()

4 5996

2 1597

3 1209

1 755

Name: Target, dtype: int64

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

Poverty\_level=train[train['v2a1'] !=0]

Poverty\_level.shape

(2668, 136)

poverty\_level=Poverty\_level.groupby('area1')['v2a1'].apply(np.median)

poverty\_level

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 paying rent less than 140000 is under poverty level.

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 ')

c=Poverty\_level['v2a1'].apply(povert)

c.shape

(2668,)

pd.crosstab(c,Poverty\_level['area1'])

| **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

## Analysis 9: Predict the accuracy using random forest classifier.

from sklearn.ensemble import RandomForestClassifier

from sklearn.model\_selection import train\_test\_split

X\_data=train.drop('Target',axis=1)

Y\_data=train.Target

### Applying Standard Scalling to dataset

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)

### Now we will proceed to model fitting

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)

Lets identify best parameters for our model using GridSearchCv

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(bootstrap=True, ccp\_alpha=0.0, class\_weight=None,

criterion='gini', max\_depth=15, max\_features='auto',

max\_leaf\_nodes=None, max\_samples=None,

min\_impurity\_decrease=0.0, min\_impurity\_split=None,

min\_samples\_leaf=1, min\_samples\_split=2,

min\_weight\_fraction\_leaf=0.0, n\_estimators=300,

n\_jobs=None, oob\_score=False, random\_state=0, verbose=0,

warm\_start=False)

RFC=best\_.best\_estimator\_

Model=RFC.fit(X\_train,Y\_train)

pred=Model.predict(X\_test)

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

Important\_features=pd.DataFrame(Model.feature\_importances\_,X\_data\_col,columns=['feature\_importance'])

Top50Features=Important\_features.sort\_values(by='feature\_importance',ascending=False).head(50).index

Top50Features

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')

for i in Top50Features:

if i not in X\_data\_col:

print(i)

X\_data\_Top50=X\_data[Top50Features]

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)

Model\_1=RFC.fit(X\_train,Y\_train)

pred=Model\_1.predict(X\_test)

from sklearn.metrics import confusion\_matrix,f1\_score,accuracy\_score

confusion\_matrix(Y\_test,pred)

array([[ 143, 17, 0, 29],

[ 8, 324, 4, 63],

[ 1, 12, 214, 75],

[ 2, 10, 3, 1485]])

f1\_score(Y\_test,pred,average='weighted')

0.9026906492316511

## Analysis 10: Check the accuracy using random forest with cross validation.

accuracy\_score(Y\_test,pred)

0.906276150627615

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

# lets drop Id variable.

test.drop('r4t3',axis=1,inplace=True)

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

test['dependency']=test['dependency'].apply(map)

test['edjefe']=test['edjefe'].apply(map)

test['edjefa']=test['edjefa'].apply(map)

test['v2a1'].fillna(0,inplace=True)

test['v18q1'].fillna(0,inplace=True)

test.drop(['tipovivi3', 'v18q','rez\_esc','elimbasu5'],axis=1,inplace=True)

train['meaneduc'].fillna(np.mean(train['meaneduc']),inplace=True)

train['SQBmeaned'].fillna(np.mean(train['SQBmeaned']),inplace=True)

test\_data=test[Top50Features]

test\_data.isna().sum().value\_counts()

0 48

31 2

dtype: int64

test\_data.SQBmeaned.fillna(np.mean(test\_data['SQBmeaned']),inplace=True)

test\_data.meaneduc.fillna(np.mean(test\_data['meaneduc']),inplace=True)

Test\_data\_1=SS.fit\_transform(test\_data)

X\_data\_1=pd.DataFrame(Test\_data\_1)

test\_prediction=Model\_1.predict(test\_data)

test\_prediction

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

Interpretation : **Above is our prediction for test data.**

# Conclusion :[¶](https://proxy-5.vocareum.com/lab?#Conclusion-:)

**Using RandomForest Classifier we can predict test\_data with accuracy of 90%.**